Making Links on Your Web Pages Last Longer Than You

Ayush Goel
University of Michigan

Jingyuan Zhu
University of Michigan

Harsha V. Madhyastha
University of Michigan

Abstract

It is common for the authors of a web page to include links to related pages on other sites. However, when users visit a page several years after it was last updated, they often find that some of the external links either do not work or point to unrelated content. To combat these problems of link rot and content drift, the solution used today is to capture a copy of the linked page when a link is created and serve this copy to users who choose to visit the link.

We argue that this status quo ignores the reality that one does not always link to a page in order to point visitors to the content that existed on that page when the link was created. The utility of linking to a web page by simply directing users to that page's URL is that they can benefit from any updates to the page's content (e.g., corrections to news articles and new comments on a blog post) or access rich app-like functionality on the page (e.g., search). In this paper, we present a sketch of what it would take to make web links resilient while accounting for the dynamism of web pages.

CCS Concepts

• **Information systems → World Wide Web**;

Keywords

Web Link Persistence, Web Archives

ACM Reference Format:
Ayush Goel, Jingyuan Zhu, and Harsha V. Madhyastha. 2022. Making Links on Your Web Pages Last Longer Than You. In *The 21st ACM Workshop on Hot Topics in Networks (HotNets '22), November 14–15, 2022, Austin, TX, USA.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3563766.3564103

1 Introduction

When we write on any topic or produce any new content, we typically build on the work of others. For example, authors of books and research papers include citations to support their claims and to refer readers to relevant prior work. Similarly, software developers link to libraries written by others, so as to not have to implement all functionality from scratch.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
HotNets '22, November 14–15, 2022, Austin, TX, USA
© 2022 Association for Computing Machinery.
ACM ISBN 978-1-4503-9899-2/22/11…$15.00
https://doi.org/10.1145/3563766.3564103

In the context of the web, authors of web pages often link to pages hosted on other sites in order to point visitors to related information and services [20]. When an author of a page wishes to link to another page, they do so by including the URL u for the latter in the source code of their page. When any user chooses to follow this link, the user's browser loads the page that exists at u.

While we take this modus operandi on the web for granted today, linking to a page using that page's URL results in two significant problems over the long term.

- **Link rot.** Any citation to a book or paper includes the title and list of authors, and a reader is free to look up a copy in any library. In contrast, a link to a web page mentions the specific hostname and path where the page is hosted. As a result, several years after a link is created, it is often the case that requests to the specified URL results in an error (e.g., failed DNS resolution or 'Page not found' HTTP 404 response) because there is no longer any page at that URL.
- **Content drift.** Even if a web page does exist at the URL specified in a link, the content at that link might have significantly changed. Unlike books and research papers, whose content is static, any website is free to vary its response across multiple requests to the same URL on its site. Consequently, when a user visits an old page and follows a link on it, they might find that the linked page contains unrelated information because the content at a specific URL often significantly diverges over time.

Prior studies have found that both problems commonly occur across the web [13, 16, 18, 19, 22]. For example, millions of external links included in Wikipedia articles no longer work [6, 14], and 3 out of 4 references in a corpus of 240 thousand pages exhibited content drift [15]. The net result is that the work that authors of web pages put into identifying which external pages they should link to is going to waste over time. In turn, users end up missing out on the carefully selected context and pointers to related information/services that web publishers intended to provide them.

To make external links on web pages resilient to link rot and content drift, the state-of-the-art solution is to capture a snapshot of every linked page when a link is created and to serve this page snapshot to users who choose to visit this link in the future. For example, when a web publisher wishes to link to a page, they can ask a service like *perma.cc* [7, 21] to crawl the page and store a copy. The publisher can then link to this stored page snapshot, which remains unmodified over time. By storing multiple copies of every page snapshot across many locations, services like *perma.cc* ensure that at least one copy will likely exist even when the original page is no longer on the web.

We observe that this reliance on static page snapshots is based on the assumption that the purpose of linking to a page is to point users to the content that existed on that page *at the time the link was created*. This assumption is invalid in a number of cases. First, many updates to pages are critical for users to see, e.g., corrections added to news articles or updates appended to a blog post about an ongoing investigation. Second, a page snapshot cannot preserve any functionality on the page that relies on client browsers interacting with the page's origin servers, e.g., on any snapshot of a product page, users cannot buy the product. Moreover, when an author links to a page that they know significantly changes over time (e.g., a page that lists the top 10 movies of the month), they are explicitly choosing to point any user to whatever content happens to exist on the page at the time of the user's visit. After all, unlike software applications, there is a reason why web links do not refer to a specific version of the page they link to.

In this paper, we ask: what would it take to enable web publishers to protect the links on their pages from link rot and content drift, without robbing users of page functionality or exposing them to stale information? Like current solutions, we seek a backward-compatible approach that requires only the sites who wish to fortify the links on their pages to make any changes. Our proposal is based on three principles.

1. Cope with site reorganizations. When a link no longer works (i.e., requests to the specified URL return an error), a previously captured snapshot of the linked page should be used as a last resort. Often, a page does not exist at its original URL only because the page has been moved and it now resides at an alternate URL. In such cases, discovering the page's new URL and directing users to it will help ensure that any critical functionality on the page is still usable.

2. Direct users to latest relevant page snapshot. When a page no longer exists anywhere on the web and using a snapshot of it is the only recourse, using the snapshot captured when the page was linked to is seldom the right thing to do. Instead, to ensure that users can see all relevant updates to the page, it is necessary to repeatedly capture new snapshots of every linked page. When a user wishes to visit a link that does not work now, they should be directed to the most recent non-erroneous snapshot of the linked page which reflects the original purpose of the link.

3. Embrace diversity of web pages. Lastly, in order to identify whether a page's content has drifted or to select the latest page snapshot in which its content had not drifted, it is insufficient to use a one-size-fits-all solution. Web pages should be classified based on the purpose they serve – e.g., informational, navigational, and functional [11] – and handled accordingly.

In the rest of the paper, we discuss the shortcomings of the current page snapshotting based solutions in more detail (§2), outline our envisioned approach and the challenges that need to be addressed to realize it (§3), and finally describe potential solutions to these challenges (§4).

2 Problems with status quo

Web publishers who wish to protect the external links they include on their pages often rely on third-party snapshotting services offered by perma.cc and other web archives [4, 5]. In this section, we describe the various shortcomings posed by this approach of relying on static page snapshots to cope with link rot and content drift.

Dataset. To support our arguments, we analyze all external links on a corpus of web pages which were last updated in 2021 or earlier but are still relevant today. We assemble our corpus as follows. On June 1, 2022, we queried the Google Trends API [3] and obtained the 128 search queries that it returns. We issued each of these queries to Google and crawled the top 20 results for every query, resulting in a total of 2,560 pages. On each of these pages, we pruned out the boilerplate content (e.g., recommended stories, navigation bar, header and footer), if any, using Chrome's open source DOM distiller [1]. From the remaining part of the page, we extracted all outgoing links to third-party domains.

After ignoring pages on which no external links appear in the non-boilerplate portion of the page, we sample at random 100 pages which were last updated in 2021 or earlier. The fact that these pages appear in the top 20 results of popular queries shows that they are still relevant today, thereby making it important to preserve the links on these pages. Our corpus of 100 web pages cumulatively contain 2,273 external links. We manually examined all of these links and classified each link into one of the following three categories [11].

- **Informational:** The content on the linked page is largely static, and the purpose of the link is to point users to the information contained on the page.
- **Navigational:** Flux in the linked page's content is expected; the intention underlying the link is to point users to a page from where they can find other relevant pages.
- **Transactional:** The link helps users discover some app-like functionality that is offered on the linked page.

Transactional pages become dysfunctional when accessed via snapshots. 2% of all external links in our dataset were transactional, and 21% of the pages in our corpus contained atleast one such link. For example, the civilian application process page on the US Capitol Police website https://www.uscp.gov/careers/civilian-application links to their career portal https://uscp.csod.com/ux/ats/careersite/1/home?c=uscp using which civilians can read various job descriptions and also apply to them, by uploading the necessary documents. Such transactional pages become non-functional when accessed via a static page snapshot, since interactions with the page's origin servers no longer work.

Page snapshots poorly represent navigational pages. On the other hand, 39% of the external links were navigational, and 40% of the pages had such links. For example, to point to a partnership with the Centre for Risk Studies at the University of Cambridge, the page at https://www.wtwco.com/en-US/About-Us/environmental-social-and-governance

links to the Centre's home page, which is at https://www.jbs.cam.ac.uk/faculty-research/centres/risk/. Since the content on such navigational pages changes often, a page snapshot that was captured when the page was originally linked to is likely to contain stale information, e.g., the latest media coverage of the Centre's work.

Informational pages are not static. Finally, 59% of links were informational, and 73% of pages had at least one such link. In such cases, relying on an old snapshot of a linked page can result in users missing out on critical updates. For example, the page at https://kisscincinnati.iheart.com/featured/josh-martinez/content/2021-11-06-report-astroworld-deaths-were-result-of-a-targeted-attacks-with-syringe/ includes a link to https://www.tmz.com/2021/11/06/travis-scott-astroworld-concert-stampede-dead-drake/, an article about a targeted attack at a Houston music festival. Internet Archive's copies of this article reveal that, if a user relied on a snapshot captured when this link was created, it would be missing critical updates made to the article thereafter based on comments from Houston's Police Chief.

3 Envisioned approach

To overcome the above-mentioned limitations of existing solutions for fortifying web links, one might be tempted to rethink how pages on the web link to one other. For example, creators of the web had considered bi-directional links [9], which would enable any site to notify other sites which links to its pages of any updates. Similarly, prior work has proposed content-based networking [12] where web documents are identified and linked based on a signature of their content, rather than where they are hosted. However, with any such proposal to rearchitect the web, a site that seeks to shield its users from link rot and content drift would be at the mercy of when other sites adopt the proposed changes.

Therefore, in keeping with existing solutions, our proposed link management service, *DuraLink*, requires changes only on any site that seeks to protect the links on its pages. Like existing services such as *perma.cc*, *DuraLink* will require a web publisher to register all linked URLs with it whenever a page is created or updated. For any link, *DuraLink* will allow web publishers to specify whether only that link need be protected or all links of a certain depth starting from the specified link need to be preserved. Thereafter, *DuraLink*'s operation will differ from existing services primarily in three ways. Figure 1 summarizes the high-level decision tree that dictates where *DuraLink* will redirect a user to when the user chooses to visit a link registered with it.

Repeated snapshotting. First, instead of taking a single snapshot of a link when it is first created, *DuraLink* will repeatedly recrawl every link at regular intervals. How frequently a particular page should be recrawled can be guided by the rate of content flux on the page, which can be determined based on the first few snapshots of the page. The benefit of repeatedly capturing new snapshots of any linked page is that, when the page is no longer on the web, *DuraLink* will be able to direct users to the latest snapshot which preserves the intent of the link, instead of the original snapshot, which might contain stale information.

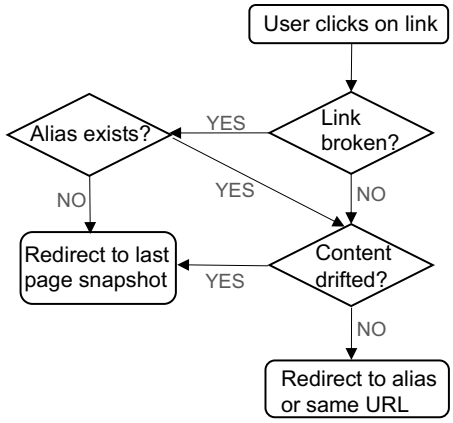

Figure 1: When a user chooses to visit a link, logic that *DuraLink* will use to identify where to direct the user to.

Alias detection. When the URL specified in a link ceases to function, it is not always the case that the page at that URL no longer exists on the web. It can often be the case that the site hosting that page has been reorganized and the page now resides at an alternate URL, which we refer to as an *alias* of the original URL. Table 1 presents some examples. Therefore, when a user chooses to visit a link that is now dysfunctional, *DuraLink* should serve a previously captured snapshot only if a functional alias for the link does not exist. Doing so is particularly important for transactional pages, since the functionality on those pages will not work on their snapshots. Note that, in order to honor the choice made by an author regarding which pages to link to, *DuraLink* should redirect any broken link only to the new URL for the same page previously available at that link, not to an alternate page with similar content/functionality.

Content drift detection. Finally, if either a linked URL is not broken or if an alias for it exists, *DuraLink* must examine if the content on the live page has drifted sufficiently that the original intention associated with the link is now violated. To detect content drift, *DuraLink* will have to take into account both the category the linked page belongs to (i.e., informational, navigational, or transactional) and the context associated with the link on the source page (e.g., anchor text and the content surrounding it). Once *DuraLink* determines that the content at a target link has drifted, it can stop recrawling the link and serve its last snapshot of that page henceforth.

4 Research challenges and potential solutions

To realize the above vision, we identify the following three challenges.

- **Discovering URL aliases.** When requests to a previously functional URL begin to fail, how to tell if the page that was available at that URL still exists elsewhere on the

Domain	Old URL	New URL
edx.org	www.edx.org/blog/edx-welcomes-library-alexandria	blog.edx.org/edx-welcomes-library-alexandria
	www.edx.org/blog/art-poetry-outside-classroom	blog.edx.org/art-poetry-outside-classroom
	www.edx.org/blog/columbia-university-joins-edx	blog.edx.org/columbia-university-joins-edx
reviewjournal.com	lvrj.com/news/16976731.html	reviewjournal.com/news/patients-vent-emotions
	Title: Patients vent emotions	
	lvrj.com/news/7358326.html	reviewjournal.com/news/clinton-has-huge-lead-in-Nevada
	Title: Clinton has huge lead in Nevada	

Table 1: On a couple of sites, examples showing the patterns in how broken URLs map to their aliases.

same site? On the one hand, unlike images, videos, and other static files [8], the hash of a page's content is not a unique identifier for it, since a page's content can vary over time. Crawling the entire site to discover the page's new location is impractical. On the other hand, even if we find a page whose content is similar to the last captured snapshot of the URL, it might be a different page than the one that was originally linked to.

- **Detecting content drift given different page types.** To detect whether the content on a page has drifted, it is insufficient to merely always compare the difference between the current version of the page and its originally captured snapshot. As mentioned earlier, it is perfectly acceptable for a linked page to be updated in a number of cases. Detecting content drift requires a precise understanding of the intent of the author who included a third party link on their page and whether changes on the linked page violate the purpose of the link.
- **Seamless integration into the web.** Enabling our envisioned approach calls for modifications to how websites operate. The web stack has a number of parts that could be modified, e.g., content management systems, page serving infrastructure, and client-side JavaScript. It is critical to enable link resilience in a manner that minimizes the burden on website providers and optimizes user experience.

Next, we elaborate on why each of these challenges is non-trivial to address, and we outline potential solutions.

4.1 Re-discovering pages after site reorganizations

When *DuraLink*'s attempt to recrawl a URL registered with it fails, it will attempt to find an alias for the URL, i.e., the new URL at which the same page now exists. The most straightforward way to do so is by following the strategy that a human would use. *DuraLink* could query web search engines such as Google or Bing using keywords extracted from its last successfully crawled snapshot and restrict results to the same site as the original URL. For an informational page, top-N words from the page's content [17] are likely to suffice for the search query. Whereas, for navigational pages, querying with the page title is likely to work better. If any of the search results has a similar title or content as *DuraLink*'s last snapshot for the URL, it can be considered the alias.

However, such an approach is fundamentally limited in its accuracy. The content/title between *DuraLink*'s last snapshot for a page and a current page on the same site might match only because they happen to be two similar pages. For example, the Internet Archive reveals that the page previously available at http://www.ew.com/article/1993/10/29/unplugged, which is now a broken URL, had the title "Unplugged". The page at https://ew.com/article/2002/01/07/unplugged/ today has the same title. It is clear from manual inspection that these are two different pages. But, by comparing the page previously available at a now dysfunctional URL and a page currently available on the same site, how can an algorithm reason about whether these two pages are the same?

Patterns in URL changes. We posit that, to automate the accurate identification of aliases for broken URLs at scale, the key is to leverage the following property: when a site is reorganized, it is rarely the case that a single page is moved; rather, a collection of pages with similar pages end up with new URLs. Therefore, if a broken URL has an alias, it is usually the case that other similar URLs (e.g., those in the same directory) also have aliases. Consequently, there is usually a pattern in how the old URLs for pages on a site map to their corresponding new URLs (i.e., their aliases).

Leveraging patterns for accuracy. We therefore argue that one can confirm that a URL u' is an alias of u only if the transformation from u to u' is the same as that from other dysfunctional URLs similar to u to their corresponding aliases. For example, we can confirm that the two URLs in the first row of Table 1 are the old and new URLs for the same page because other URLs on edx.org have similarly transformed to their new versions, as shown in the remaining rows of the table. This transformation can often be fairly complex, e.g., as also shown in Table 1, the site previously hosted at lvrj.com has now moved to reviewjournal.com (which we can confirm because the home page for the former redirects to the home page for the latter) and the new page URLs contain page titles in place of a unique page ID.

In order to leverage these URL replacement patterns for confirming the accuracy of any particular alias, *DuraLink* will have to discover the aliases for other similar URLs. Therefore, when a link is registered with *DuraLink*, it must also identify a few similar URLs and capture snapshots of those pages. These URLs might have already been registered with *DuraLink* as part of other links. If not, URLs similar to u can be discovered in several efficient ways: 1) on the page linked from u, e.g., many articles have a "related topics" section which links to URLs in the same directory; 2) the prefix of u ignoring the portion after the last '/' is often a navigational page which links to many URLs

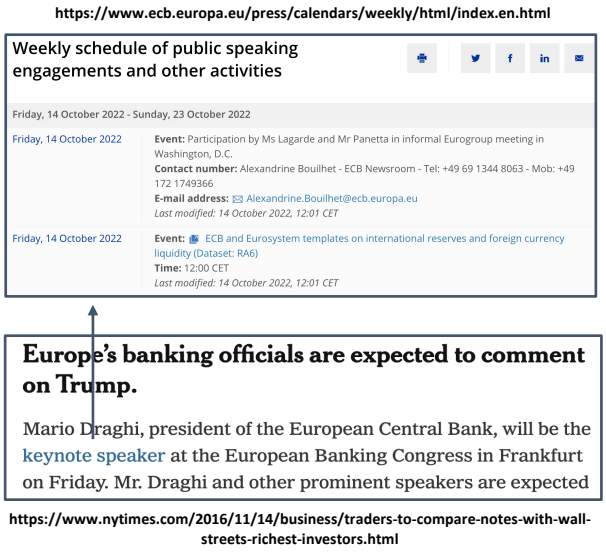

Figure 2: An example of how content drift occurs because a link refers to specific information on a page that is updated frequently.

similar to u, e.g., https://abcnews.go.com/US/, the prefix of https://abcnews.go.com/US/thousands-cattle-dead-amid-continuing-heat-wave/story?id=85434516, includes links to many similar URLs; or 3) from a URL prefix based web search.

Thus, when *DuraLink* obtains an error when it attempts to recrawl a particular URL, it can try to find the alias not just for that URL but also for other similar URLs that it has previously identified. For each of those URLs, *DuraLink*'s web search using the title/content from its last successful snapshot will yield a number of candidate aliases. By deriving the transformation function from each URL to each of its candidate aliases, *DuraLink* can identify that transformation which is common across all of the similar URLs.

Leveraging patterns for efficiency. Beyond helping confirm the accuracy of discovered aliases, the patterns that *DuraLink* learns from old to new URLs can also help improve its efficiency. After having attempted to discover aliases for a number of broken URLs in a particular site/directory, *DuraLink* can learn the manner in which this site has modified the URLs for its pages. Thereafter, for any other broken URL on the site, *DuraLink* can simply use the patterns it has learned to *infer* the corresponding alias.

4.2 Detect content drift for different page types

Once *DuraLink* identifies a broken URL's alias or, if no alias exists and a snapshot of the link must be used, *DuraLink* must determine whether the content at the linked page has drifted. Prior work has either relied on manual analysis to identify content drift [23] or considered any significant change to the linked page as content drift [15]. The former approach cannot be employed in a system and the latter is prone to false declarations of content drift.

To understand how content drift occurs in practice and what approach is necessary to accurately identify it in an automated manner, we conducted the following study. Instead of using our earlier corpus from §2, which mostly contained pages created in the last few years, we recreated the study from[23] at a smaller scale. For each year from 1996 to 2019, we crawled 50 articles at random from nytimes.com. For each article, we extracted all outgoing links to external domains, and focused only on links that contained a path in the URL. We were left with a total of 780 external URLs, spanning 463 domains originating from 291 NYTimes articles.

We manually categorized each of these links as having suffered content drift or not as follows. From the NYTimes page containing a URL u, we read the text surrounding the anchor text of u in order to understand the author's intention for including that particular URL. Then, we visited the page hosted at u, and read through the page content to determine whether the relevant information is still on the page.

Ignoring the 200 URLs that had suffered from link rot, we observed that 53 URLs (9.1%) out of the remaining 580 URLs had undergone content drift, a fraction comparable to the one observed by prior work [23]. We observe that all of these cases of content drift can be categorized into one of two cases. First, most (68%) cases of content drift correspond to soft-404s, i.e., it is not the case that the page at the original link has been modified; rather, the site is serving an incorrect response. Second, the remaining cases of content drift (32%) were cases where a web page's author linked to a page that is expected to change over time, but the context associated with the link refers to specific information that existed at the time the link was created; Figure 2 shows one such example.

These findings suggest that identifying content drift on the web is likely easier than evident from prior work [15, 23]. Though on a small dataset, our analysis suggests that most instances of what appears content drift instead corresponds to link rot. To automate the detection of soft-404s, *DuraLink* can compare the response it receives for a particular URL on a site to that site's responses for other similar URLs [10], e.g., if a site redirects a number of similar URLs to the same URL, that shows that all of these redirections are broken.

To identify remaining instances of content drift, *DuraLink* would need to employ a binary classifier. Given a link to a page whose content is known to change over time, the classifier would take as input the text surrounding the link's anchor text and the text on the linked page. The purpose of this classifier would be determine whether the intention of the link is to refer to specific information on the linked page or to simply point visitors to the linked page. If the former, *DuraLink* would know that this link is prone to content drift when the linked page's content changes.

4.3 Integration with the modern web stack

Lastly, we consider the question of how to integrate *DuraLink* into any website.

To see how existing services which are comparable to *DuraLink* work, let us consider the example of perma.cc [7].

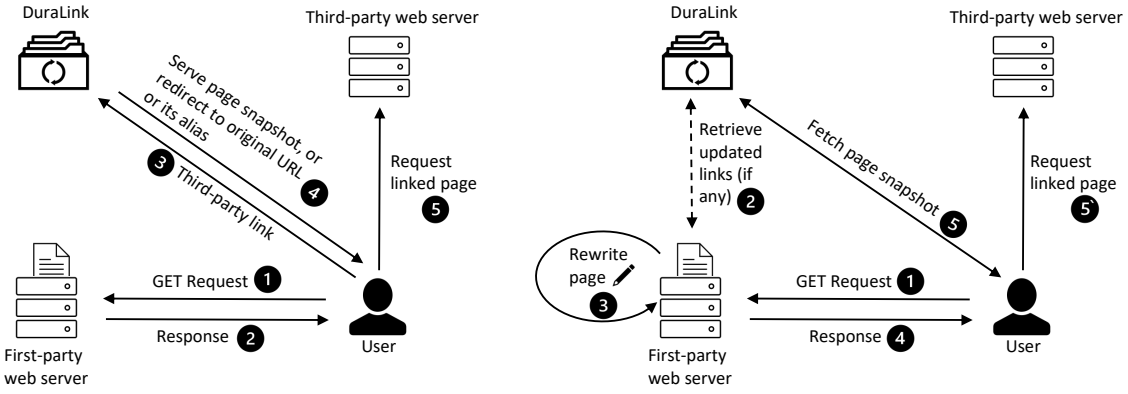

Figure 3: (a) For a page served by the first-party server, illustration of how user's visits to third-party links will be served if *DuraLink* is used in a manner identical to perma.cc. (b) We instead propose leveraging dynamic generation of HTMLs.

When a web page's author wishes to link to a URL u, they submit this URL to perma.cc, which crawls the page at u and returns a link akin to perma.cc/u. The author then links to this alternate URL on their page. When users click on this link in the future, their browser fetches and renders perma.cc's snapshot of the page at u.

Problem: Availability/privacy vs. user experience. *DuraLink* could be integrated into any website in a similar manner (Figure 3(a)) except that, when users visit a link, *DuraLink* would use the logic in Figure 1 to determine where to direct the user to. However, such an architecture would not be ideal with respect to both availability and privacy. Unlike perma.cc, which always returns stored page snapshots *DuraLink* may forward the user onto the originally linked URL if there is no link rot or content drift. In such cases, having *DuraLink* serve as a relay results in it learning about users' browsing activity for no resultant benefit. Moreover, such links will be inaccessible to users when *DuraLink* is down. One could sidestep these problems by having users always visit the originally linked URL and leave it to them to invoke *DuraLink* when they encounter a broken link or observe content drift. But, that would result in a poor user experience.

Solution: Dynamic rewriting of HTMLs. To address the tradeoff described above, we observe that every link on the page need not have to either always use the original URL (which punts the detection of link rot and content drift to users) or always use a URL hosted by *DuraLink* (which makes it a single point of failure). Modern web servers already generate dynamic HTML responses. Every page has a HTML template, which is populated in response to a user request based on server-side state (e.g., the visitor count of a page), the user's cookies, etc. [2].

This dynamic generation of HTML responses facilitates the server-side integration of *DuraLink* as follows. When a web server receives a client's request for a page, apart from querying its back-end database for server-side state, the server can also query *DuraLink* for how it wishes to handle every external link on the page. Links that suffer from neither link rot nor content drift (which *DuraLink* determines offline when recrawling links registered with it) can be left unmodified. Whereas, links for which *DuraLink* wishes to direct users to either an alias or a snapshot can be rewritten appropriately. As shown in Figure 3(b), when users visit links on the page, they need to interact with *DuraLink* only to fetch page snapshots.

The risk of *DuraLink* being offline can be mitigated if several providers independently offer instances of the service and publishers register every link with multiple service providers. In the rare case when all *DuraLink* deployments are down, users may temporarily be exposed to broken links or linked pages whose content has diverged. However, *DuraLink*'s unavailability will have no impact on users' ability to visit functional links. Moreover, once a server has learned from *DuraLink* that a particular link needs to be rewritten to point to the link's alias or a snapshot, the server can cache this information.

5 Conclusions

On the web, a page can include links to pages in any other domain. This flexibility, however, comes with the risk that those external pages may later either no longer exist at their original URLs or have significantly changed. In this paper, we highlighted that existing solutions to these problems ignore the fact that snapshots of web pages are not equivalent to the original pages: functionality that relies on client-server interactions do not work and the information they contain can be stale. To overcome these limitations without having to redesign the web, we presented our vision for *DuraLink*, a service which will serve page snapshots only when absolutely necessary to combat link rot or content drift. We identified three key enabling capabilities to realize *DuraLink*: 1) repeatedly recrawl linked pages, 2) when a link ceases to work, identify whether the page still exists at an alternate URL, and 3) account for the intent of a link and the category of the linked page to identify content drift.

References

[1] chromium/dom-distiller: Distills the DOM. https://github.com/chromium/dom-distiller.

[2] EJS – Embedded JavaScript templates. https://ejs.co/.

[3] Google trends API. https://trends.google.com/trends/.

[4] Internet archive. https://www.archive.org/about/.

[5] Library of congress. https://www.loc.gov/web-archives/.

[6] More than 9 million broken links on Wikipedia are now rescued. https://blog.archive.org/2018/10/01/more-than-9-million-broken-links-on-wikipedia-are-now-rescued/.

[7] Perma.cc. https://perma.cc/.

[8] A. Anand, A. Balachandran, A. Akella, V. Sekar, and S. Seshan. Enhancing video accessibility and availability using information-bound references. *IEEE/ACM Transactions on Networking*, 24(2):1223–1236, 2015.

[9] M. Appleton. A short history of bi-directional links. https://maggieappleton.com/bidirectionals.

[10] Z. Bar-Yossef, A. Z. Broder, R. Kumar, and A. Tomkins. Sic transit gloria telae: Towards an understanding of the web's decay. In *WWW*, 2004.

[11] A. Broder. A taxonomy of web search. In *ACM Sigir forum*. ACM New York, NY, USA.

[12] A. Carzaniga and A. L. Wolf. Content-based networking: A new communication infrastructure. In *Workshop on Infrastruture for Mobile and Wireless Systems*, 2001.

[13] R. P. Dellavalle, E. J. Hester, L. F. Heilig, A. L. Drake, J. W. Kuntzman, M. Graber, and L. M. Schilling. Going, going, gone: Lost internet references. *Science*, 2003.

[14] M. Graham. Turn all references blue. https://archive.org/details/mark-graham-presentation.

[15] S. M. Jones, H. Van de Sompel, H. Shankar, M. Klein, R. Tobin, and C. Grover. Scholarly context adrift: Three out of four URI references lead to changed content. *PloS one*, 2016.

[16] M. Klein, H. Van de Sompel, R. Sanderson, H. Shankar, L. Balakireva, K. Zhou, and R. Tobin. Scholarly context not found: One in five articles suffers from reference rot. *PloS one*, 9(12):e115253, 2014.

[17] T. A. Phelps and R. Wilensky. *Robust hyperlinks cost just five words each*. University of California, Berkeley, Computer Science Division, 2000.

[18] S. Rhodes. Breaking down link rot: The chesapeake project legal information archive's examination of url stability. *Law Library Journal*, 2010.

[19] D. Spinellis. The decay and failures of web references. *Communications of the ACM*, 46(1):71–77, 2003.

[20] T. Urban, M. Degeling, T. Holz, and N. Pohlmann. Beyond the front page: Measuring third party dynamics in the field. In *WWW*, 2020.

[21] J. Zittrain, K. Albert, and L. Lessig. Perma: Scoping and addressing the problem of link and reference rot in legal citations. *Legal Information Management*, 14(2):88–99, 2014.

[22] J. L. Zittrain, J. Bowers, and C. Stanton. The paper of record meets an ephemeral web: An examination of linkrot and content drift within the new york times. *Available at SSRN 3833133*, 2021.

[23] J. L. Zittrain, J. Bowers, and C. Stanton. The paper of record meets an ephemeral web: An examination of linkrot and content drift within the new york times. *Available at SSRN 3833133*, 2021.