DTWiki: A Disconnection and Intermittency Tolerant Wiki

Bowei Du
Computer Science Division
445 Soda Hall
University of California, Berkeley
Berkeley, CA 94720-1776
bowei@cs.berkeley.edu

Eric A. Brewer
Computer Science Division
623 Soda Hall
University of California, Berkeley
Berkeley, CA 94720-1776
brewer@cs.berkeley.edu

ABSTRACT

Wikis have proven to be a valuable tool for collaboration and content generation on the web. Simple semantics and ease-of-use make wiki systems well suited for meeting many emerging region needs in the areas of education, collaboration and local content generation. Despite their usefulness, current wiki software does not work well in the network environments found in emerging regions. For example, it is common to have long-lasting network partitions due to cost, power and poor connectivity. Network partitions make a traditional centralized wiki architecture unusable due to the unavailability of the central server. Existing solutions towards addressing connectivity problems include web-caching proxies and snapshot distribution. While proxies and snapshots allow wiki data to be read while disconnected, they prevent users from contributing updates back to the wiki.

In this paper we detail the design and implementation of DTWiki, a wiki system which explicitly addresses the problem of operating a wiki system in an intermittent environment. The DTWiki system is able to cope with long-lasting partitions and bad connectivity while providing the functionality of popular wiki software such as MediaWiki and TWiki.

Categories and Subject Descriptors

H.4.0 [**Information Systems**]: Information System Applications-General; J.0 [**Computer Applications**]: General

General Terms

Design

Keywords

Wiki, Delay-Tolerant Networking, Developing Regions

1. INTRODUCTION

The online collaborative encyclopedia Wikipedia [27] constitutes (as of January, 2008) approximately 7.5 million user-contributed articles and is among the top ten most visited websites in the world. Although article growth has tapered off with the project's maturity, the web site has grown organically since its inception in 2001. A critical piece of the success of Wikipedia has been the ease with which the users can participate in writing Wikipedia articles. The popularity of "wikis" [4] as website content management software also speaks to the usefulness of the medium. Generally speaking, wikis have proven to be an excellent tool for collaboration and content generation.

Several aspects of wikis are interesting for the emerging regions context. First, there is a great need in emerging regions for content that is locally relevant. Local content can take a variety of forms, from a local translation of existing online material to the community knowledge bases. Most of the content on the web today is written in English and targeted towards in the industrialized world. Many information communications and technology (ICT) for development programs seek to address this issue with a local content generation effort using website creation tools [2, 12]. Training is needed to become proficient in website authoring. Wikis are well suited to fill this niche because they combine both content creation and content sharing into a tool that is easy to use. The Wikipedia experience shows that interested user base can quickly generate a large amount of useful wiki content.

Second, the wiki model satisfies many organizational needs for collaboration and document sharing. Because wiki documents are unstructured, users can use shared pages as a whiteboard without being overly burdened by system-enforced semantics and schemas. Annotations and discussions are easily incorporated into a wiki page along with the main content. Many wikis also implement some form of revision control system, which is a powerful tool for managing multiple concurrent user modifications.

Finally, there is a large demand for the content generated in existing online wikis. Many educationally-focused non-governmental organizations (NGOs) distribute snapshots of Wikipedia content as part of their digital classroom material. [28, 15, 11] Although snapshots are useful, they result in a unidirectional flow of information. A wiki that has a bidirectional information flow would allow local student discussions to connect with the wider community on the Internet. This is something that is not possible with a snapshot or caching-based approach.

To date, the use of wiki software in the emerging region context has been hampered by a lack of reliable Internet connectivity. This is primarily due to the centralized web architecture of wiki software. For example, the data backend for the popular Wikipedia website consists of a single database master replicated in a cluster environment. This kind of architecture precludes the use of wikis in environments where there are long lasting network partitions separating "well-connected" islands of users from each other and the Internet. Common examples of such environments include Internet cafés and schools that are disconnected from the network due to poor infrastructure or a desire to reduce connection costs. In these scenarios, users are able to communicate within the local net-

This material is based upon work supported by the National Science Foundation under Grant No. 0326582.

WWW 2008, April 21–25, 2008, Beijing, China.
ACM 978-1-60558-085-2/08/04.

work, but cannot always reach the Internet. More esoteric examples include locations serviced by "sneaker net" or a mechanical backhaul [19, 16].

Despite wiki's centralized provenance, the basic wiki data model is a good candidate for intermittent networks and disconnected operation. Wiki systems already have to handle the possibility of conflicting concurrent user edits that can occur due to stale browser data. In addition, the central metaphor of the wiki, that of a *page*, provides an obvious boundary for data sharing and data consistency. The loose semantics of the wiki application also tempers user expectations for conflict resolution. On the Wikipedia website for example, users frequently perform minor edits to "fix" content such as correcting spelling or grammar. User intervention in resolving conflicts fits this usage pattern.

Finally, we note that many of the auxiliary operations of wikis, e.g. presentation and search, are functions that can be implemented with the information available locally with no need for distributed state.

We build on the work of *Delay-tolerant Networking* (DTN) [3, 7], which provides a messaging layer that handles intermittency and disconnection, manages communications links as they go up and down, and provides durable storage for messages in transit. On top of DTN, we built a replicated file system, TierStore, that manages file consistency and replication. DTWiki is built as an application on top of this file system.

This paper describes the design and implementation of DTWiki, a wiki that enables bi-directional collaboration among users working in disconnected network environments. The paper is organized as follows: First, in Section 2, we make the term "wiki" more precise by distilling a functional specification from a survey of popular wiki systems and discuss what features are desired in a disconnected wiki. We then describe how DTWiki is implemented in Section 3. Finally, we demonstrate that our prototype performs adequately in Section 4, review related work in Section 5 and conclude in Section 6.

2. WIKI NATURE

Because wiki software ranges in implementation size from a two-hundred byte shell script to full-blown content-management systems, we took the top five wikis as referenced by the popular wiki website c2 [26] as a basis for determining the key functionality needed for wiki software. The methodology used by c2 to rank wiki engines is to compare Google search hit counts for each of the wiki engines. We are careful to note that this is not at all a scientific sampling, however, it should serve to provide a flavor of the feature sets of wiki software. We note that most of the code in wiki software stack concerns rendering and formatting. Although presentation is important, it is the semantics of the shared data that affects how the system is constructed. Table 1 summarizes the features of the top five wiki platforms [8, 24, 22, 18, 17].

At the most basic level, wikis comprise a set of user-editable web pages written in a simplified markup language. Referenced pages in the wiki are automatically generated as they are linked. The general aim of the wiki application is fast and easy content creation. All of the top five wiki systems track edit history in addition to page contents. This version control system allows users to retrieve previous versions of a page. Version control brings all of the benefits of automatic revisioning to concurrent multi-user environments.

As wikis have moved away from their original open, "free for all", model of user contribution to one with more structure, the management of user accounts has become part of the wiki system. User accounts determine access control for pages on the wiki site.

Content creation is the product of a collaboration among participating users. Most of the top five popular wikis devote a separate content namespace for discussions among users. The discussion page could be implemented as an ordinary wiki page, although it may be more appropriate to structure it as an append-only chat log, which more closely matches the desired semantics.

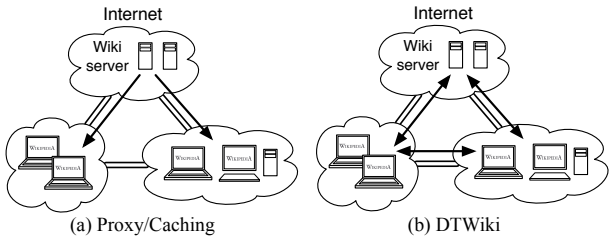

Figure 1: A comparison between the proxy/caching architecture and the DTWiki architecture. Each cloud represents a potentially long-lasting partition of the network. In the proxy architecture, updates only flow from the Internet-based server, while in the DTWiki architecture, wiki updates can be made and disseminated from any of the clouds.

Finally, all five wiki systems have search and indexing functionality to enable users to browse created content efficiently. Thus for DTWiki to be considered to be feature complete, we need to have revision tracked hypertext pages, some form of integrated user account storage, a discussion oriented set of pages, and text search and indexing capabilities.

2.1 Wikis For Intermittent Environments

The current approach to bringing wikis and wiki content to poorly networked environments is to create a mirror of the wiki site, either by taking a static web crawl of the site or by running a copy of the wiki software on a server on the partitioned side of the network from a database snapshot. Figure 1a depicts such a setup. Content hosted on the Internet based wiki server is cached or pushed out to the disconnected clients in each network partition. Clients view wiki content on the local server on the intranet but have no interaction with the site hosted on the Internet.

The problem with using snapshots and local caching schemes is that the local content has become divorced from updates to the main content. In addition, contributions of the local users are either not possible or they are difficult to share with other parts of the network. This turns the user contribution based wiki model into a static, read-only web page.

In the DTWiki system architecture that is depicted in Figure 1b, clients can locally modify their wiki state at any time. Changed content from the Internet wiki as well as those in the intermittently connected networks are synchronized across temporary network partitions and bad communication links. Entry into the wiki system only involves connecting the local DTWiki server to an upstream DTWiki host via DTN. Once the underlying network routing has been established, wiki content is transparently shared among DTWiki hosts.

Finally we note that using a external synchronization mechanism such as rsync [23] to keep wiki databases in each partition up to date does not work well because it is not integrated into the wiki semantics nor does it handle concurrent update conflicts. Also, replacing the underlying IP protocol with a Delay-Tolerant transport is not sufficient because the HTTP protocol between the client and server behaves poorly given long network roundtrip times.

Wiki	Implementation	Features
MediaWiki	PHP, SQL storage	Revision history, ACLs, discussion pages, rich media, search, plugins
TWiki	PHP, RCS, SQL	Revision history, ACLs, SQL database integration (forms), rich media, search, plugins
TikiWiki	PHP, SQL	Revision history, ACLs, full fledged content management system, search, plugins
PukiWiki	PHP, file system	Revision history, file attachments
PhpWiki	PHP, SQL	Revision history, file attachments

Table 1: Popular wiki packages and their feature set.

3. IMPLEMENTATION

In this section we detail how DTWiki is implemented. First, we briefly summarize the interface of TierStore/DTN, which is the file system platform we used to construct DTWiki. We then describe how each component of a wiki as described in Section 2 is implemented in DTWiki.

3.1 TierStore

TierStore [6] is a file system for building distributed, delay-tolerant applications. TierStore uses the Delay-Tolerant Networking (DTN) stack [5] for network transport, which enables the file system to be robust across intermittent connectivity and to work over diverse kinds of network connectivity. The abstraction that TierStore presents to the application is that of a transparently synchronized shared file system. TierStore has three major components:

- Transparent synchronization of file system contents.
- Partial replication of shared data.
- Detection and resolution of concurrent update conflicts on single files.

Changes to the shared TierStore file system are automatically synchronized with other TierStore hosts in the network, using any available DTN mechanisms for transport. TierStore guarantees that changes to a file will not be visible to remote hosts until all local file handles have been closed. This provides the application with a mechanism to ensure that inconsistent partial edits to a file never appear remotely.

Portions of the file system namespace are divided into disjoint *publications* that are shared among *subscribing* TierStore nodes. Publication boundaries are delineated by a directory and depth. For example, subscribers of a publication rooted at folder `images/` with depth two will share files in `images/` and its first level of subdirectories. Figure 2 shows the three TierStore file systems sharing two publications: `text/` and `images/`. In Figure 2, node A is subscribed to `images/` and `text/` while nodes B is only subscribed to `text/` and C is only subscribed to the `images/` publication.

Because hosts in the TierStore system may be temporarily partitioned, applications using TierStore will inevitably concurrently update shared state. TierStore is a system that only guarantees *coherence* rather than general *consistency*. This means that it can only detect conflicts resulting from concurrent updates to the same file or file name but cannot impose any ordering constraints across updates to a set of multiple files. When concurrent updates occur, the TierStore system detects and informs the local application through application configured *resolver* functions.

In default case, the TierStore system resolves conflicting file versions by appending a unique suffix to the remote file in the file system. For example, as shown in Figure 2, suppose two TierStore nodes A and B concurrently update the same file `/notes.txt`. After synchronization, the TierStore application on node A will

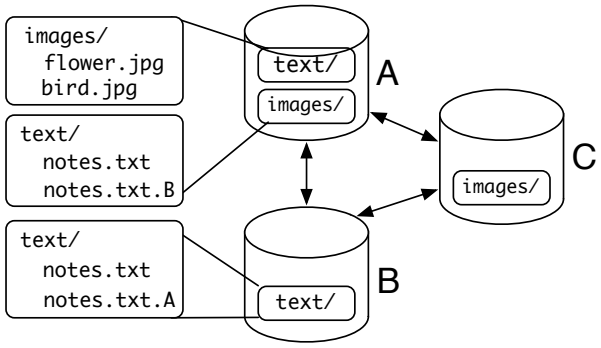

Figure 2: Three TierStore nodes A, B and C sharing publications `images/` and `text/`. The file `test/notes.txt` received a conflicting update on nodes A and B resulting in a conflict files `notes.txt.A` and `notes.txt.B`

see local edits in `/notes.txt` while the edits from node B will be in `/notes.txt.B`. However, at node B, local edits will be in `notes.txt` and node A's edits in `notes.txt.A`. Although the default resolution procedure is primitive, it turns out that this is sufficient for implementing all of the functionality needed by DTWiki.

Finally, TierStore has support for application hooks that function similar to the `inotify` function in Linux for notifying the application to take action when changes to the file system arrive. These are useful for maintaining structured data (such as indices) derived from the state stored within TierStore.

3.2 Data Schema

Most existing wiki software store data using a SQL database. In the case of DTWiki the data layout is complicated due to three concerns. First, because the underlying storage is a file system, we need to organize the layout in a manner that is efficient to access. Second, because TierStore only supports object coherence, we need to guarantee that the wiki does not require consistency across multiple files. As stated earlier, the TierStore system cannot give any ordering guarantee with respect to updates of two different files as they are received on remote hosts. Finally, shared files in the TierStore system will result in conflicting versions. Conflicts must be dealt with in a way that makes sense to the user and does not result in invalid states for the application.

Our strategies for dealing with these requirements are as follows: We attempt to make each piece of data (e.g. file) shared in TierStore as self-contained as possible. For instance, all attributes related to a data object are pushed into the object itself. The primary way by which an object is named is used to organize the file system hierarchy. Auxiliary methods of reference (such as external databases indexing the content) are handled by inserting hooks into the TierStore update notification system to update an external database with indexing information whenever updates arrive. Finally, conflicts in

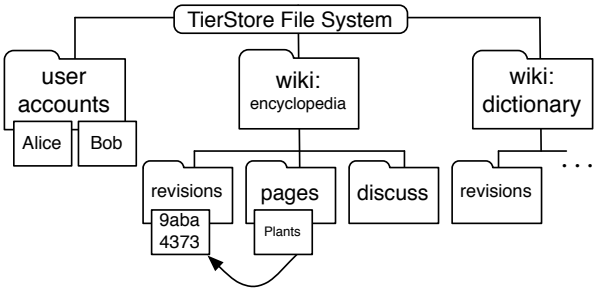

Figure 3: The data layout of the DTWiki. The arrow indicates a symbolic link.

shared state are either 1) handled via a semantically well-defined merge process, or 2) avoided by the use of techniques such as globally unique names or content-based naming.

We now describe each component of DTWiki data backend in detail. Figure 3 is a summary of the general data layout of DTWiki in the TierStore file system.

3.3 Pages and Revisions

The central data object in DTWiki is that of a page revision. A new page revision is created whenever a change is made to a wiki page. A revision stores three things: metadata about the page edit, the contents of the wiki page itself and the name of the preceding revisions that the current revision is derived from. Each revision is a separate file in a `revisions/` directory in the TierStore file system. All revision files are named with a globally unique id. Figure 4 shows an example revision file.

A separate directory `pages/` contains symbolic links with a name of a page to the revision representing the most current revision in `revisions/`. For instance, the page with the title "Wiki Page" would be symbolic linked to the appropriate revision file in the revisions directory.

When two users concurrently update the same wiki page, they create conflicting symbolic links in the `pages/` directory. The default TierStore conflict resolver renders this by appending a suffix to each remotely conflicting copy. The wiki software detects the presence of the conflict and renders to the user a message stating that a conflict had occurred and displays a merge of the contents of the conflicting revisions. Note that this is handled similarly to the case that occurs in current wiki software when a user edit on their local web browser has become stale due to a concurrent edit, so the user does not need to learn any new concepts in order to deal with concurrency.

Previous revision pointers in the metadata allow the wiki system to derive a revision history of a page. A page may have multiple previous revisions if the edit occurred during a conflict. When an edit occurs on a page with conflicting updates, each of the previously conflicting revisions is added to the `previous revisions` field.

3.4 User Accounts

Information about user accounts is stored in the `users/` directory. Figure 4 shows an example user account file. The user account file contains user login and passwords as well as their access-control groups. Although update conflicts in the user account file should be rare, there are natural conflict resolution rules for each of the mutable fields. For example, the access-control groups that a user belongs to are a union of the groups in the conflicting files.

Revision File

```
revision id:  9aba4373
previous revisions:  2487a9e3 544bafe2
date:  Wed, 24 Oct 07 8:35:25
user:  wikizen@dtn.com 3a424353
page title:  Example Page
tags:  example wikizen
read:  ALL
write:  admin

page content...
```

User file

```
user name:  wikizen@dtn.com
user id:  3a424353
password hash:  ####
groups:  admin
```

Discussion Entry

```
user name:  wikizen@dtn.com
date:  Wed, 24 Oct 07 8:35:25

conversation...
```

Figure 4: Example revision file, user file and discussion file.

3.5 Attachments and Media

Each wiki page can contain attachments of binary files. Rich media and file attachments are stored in the `media/` directory in a file named by the MD5 hash of its contents. The metadata describing the attached file is stored in the linking wiki page. This scheme accomplishes two things. First, the same attachment can be linked multiple times without requiring additional storage space. Second, the content hash ensures that no conflicts can be created by a file upload, eliminating the need for revision control of the media files in addition to the wiki content.

3.6 Discussion Pages

Creation of wiki content requires some amount of coordination among users of the system. DTWiki facilitates these discussions by creating a separate user conversation page for each wiki page in the system. Some wiki systems implement the conversation page as a wiki page. We choose to have different semantics for discussions because multiple updates to the conversation page by multiple users is expected to be the norm rather than the exception. This means that the active conversations will be almost constantly in benign conflict as each user's conversations should be trivially mergable.

Conversations are stored in `discussion/`*page name*/ directory. Each new comment post by a user about the wiki page is stored in the directory in a new file named by a new globally unique id. When viewing the conversation, DTWiki software concatenates all of the conversation file entries in the discussion directory sorted by time stamp. The conversation entries are, in effect, organized as an append-only log.

3.7 Search/Indexing

Searching and indexing are commonly implemented by leveraging search functionality in the SQL database backend of the wiki. We choose not to replicate search index state via the shared TierStore file system because all of the category and search indexes can

# Revisions	Time per Revision (seconds)
5,000	0.079
10,000	0.084
40,000	0.095

Table 2: Local scalability of the system with respect to the number of revisions during an import of revisions from the WikiVersity trace, measured in terms of the time taken per revision imported.

be derived locally from the shared portion of the wiki. Thus, an off-the-shelf text search engine can be used. All that is required are the appropriate hooks to freshen the search index when new content arrives.

TierStore has a script hook system for specifying external scripts to be run when an update to the file system arrives. DTWiki uses the update notifications to run an external indexing engine. Our preliminary implementation uses the Apache Lucene search engine, but any similar text search application could be used.

3.8 Partial Sharing

Wiki sites with a great deal of content such as Wikipedia have mechanisms to organize their data into disjoint namespaces. For example, the Wikipedia website is actually a conglomeration of several separate wikis split by language and by function. The dictionary Wiktionary is occupies a separate namespace from Wikipedia. In DTWiki, each sub-wiki is placed on a separate TierStore publication, which enables participating TierStore nodes to subscribe to the subset of content that they are interested in.

4. EVALUATION

We evaluate our DTWiki prototype for scalability and efficiency. First, we show that local system can scale to handle content on the order of Wikipedia-sized loads and that the overhead imposed by our data schema is not an issue. Second, we show that the DTWiki system does not incur much additional overhead in terms of bandwidth used. Finally, we give the results for a wiki trace replay on a small simulated network.

All scalability and bandwidth experiments were run on machines with a Intel Xeon 2.80 GHz processor with 1 gigabyte of memory. The DTWiki software itself never exceeded 200 MB of resident RAM during the runs. The trace replay was performed on a set of 1.8GHz Pentium 4 with 1GB of memory.

4.1 Scalability

We tested the scalability of the DTWiki software by taking a portion of Wikipedia and measuring the time required to import the data into a local DTWiki system. The data file consists of the WikiVersity section of the website, 1.4 gigabytes of data total consisting of over 41,470 revisions. Our import software takes each Wikipedia revision and create a DTWiki revision with the same content. Table 2 is a graph of number of revisions imported to the amount of time needed for the import to finish. As shown in the graph, the DTWiki system scales linearly with the number of imported articles.

We also tested the response time of the DTWiki after the import of the data in terms of latency per page load. This was done by picking 1,000 pages randomly from the wiki and loading their contents. (Note: this does not include time spend rendering the web page in the browser.) This experiment was performed ten times. Page loads took on the average 7.8 ms to retrieve.

Revisions	Network	Content	Overhead
100	1,000,829	982,369	1.8%
500	5,564,302	5,438,636	2.3%
1000	8,009,046	7,749,412	3.2%

Table 3: Overhead of DTWiki in network traffic for synchronization with various number of revisions. Network and Content sizes are measured in bytes.

4.2 Bandwidth

Our DTWiki prototype has be competitive with existing approaches for web caching and file synchronization. To test whether this is true, we compare the bandwidth consumed by two DTWiki synchronizing their contents versus the size of the contents itself. The test data was generated from the WikiVersity dump used above. The experiment was run using 100, 500 and 1,000 revisions. Table 3 summarizes the result of the experiment. Overhead is measured as the percentage of extra network traffic versus the total size of the shared content.

4.3 Simulated Usage

In order to obtain a sense of the usability of the DTWiki system in a real network environment, we ran a time-scaled simulated usage pattern from a three month segment of the same WikiVersity trace on a real DTWiki system. The network consisted of a star topology with a single central DTWiki node and three leaf DTWiki nodes. The network connectivity between the nodes was controlled using a network traffic shaper to simulate disconnections. Two of the leaf nodes was given nightly connectivity (from 18:00 to 6:00) to emulate the conditions of a nightly scheduled dial-up connection of a remote school. The remaining leaf node was left always connected the central node to simulate an Internet connected client. In order for the simulation to finish in a reasonable amount of time, each second in the experiment was simulated by 0.005 seconds of wall clock time.

Revision edit history from the months of October, November and December in year 2007 was used to generate the simulated page edits. The edits were distributed among the three leaf nodes by randomly dividing the user population in three equal parts and assigning each set of users to a particular node. The trace data contained 12,964 revisions on 2,677 different pages from a total of 792 authors over the three month period.

Figure 5 shows the average number of updates to pages at a node during the 3-month period versus the number of conflicts detected by the DTWiki system. The conflict rate was 10 pages or less for 63 of the days in the simulation; however, there were bursts in the conflict rate above 30 conflicts for four days during the trace run. We note that the high conflict rate may be due to the random assignment of authors to nodes. A real-world assignment may exhibit more more author locality in edits which would reduce the edit conflict rate. In addition the conflict rate is conservative due to the fact we don't distinguish between conflicts that are automatically mergeable versus those that require user intervention.

5. RELATED WORK

DTWiki is closely related to a class of proxy applications that aim to make traditionally network-based services available when the hosts are disconnected. WWWOFFLE [29], NOFFLE [13] and OfflineIMAP [14], are respectively, proxies which provide clients cached offline access for HTTP, NEWS and IMAP servers. The TEK [21] project is a web caching proxy which has been designed

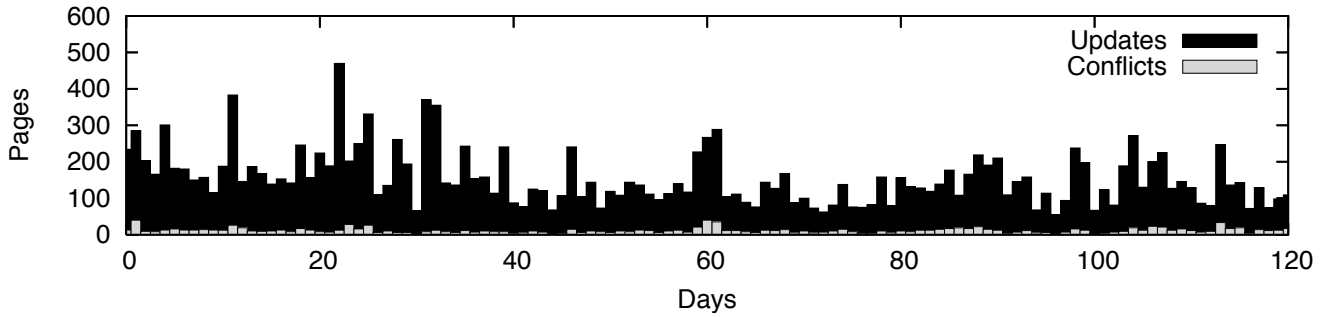

Figure 5: The average number of pages updates and conflicts that occur over a period of 120 days at a node using a replayed WikiVersity trace and simulated network with scheduled disconnections.

specifically for addressing connectivity issues of emerging regions. TEK uses e-mail as transport and supports transcoding and compression.

The design of these systems has focused mostly on the single disconnection scenario in which the intermittent connection separates the user(s) from the Internet. DTWiki provides wiki functionality for arbitrary combination of network topologies and intermittency. With regard to flexible use of network topology, the DTWiki system is related to the Lotus Notes [10], Bayou [20] peer-to-peer groupware systems. Both systems predate the modern conception of wikis. Lotus Notes has similar structure and goals. The central data structure of the Lotus system is the *note*, which is a semi-structured document that can cross reference other notes in the system. Bayou is a general system for building applications in intermittent networks. The authors of the Bayou system implement several structured collaboration applications such as a shared calendar and a shared bibliographic database, but did not investigate shared intermittently connected hypertext systems.

The concept of revisions and concurrent edit management is similar to the functionality of version control systems such as CVS and Mercurial [25, 9] which allow disconnected locals edits and resolution of conflicts cause by remote users. The difference is that our use of the DTN and TierStore software architecture allows us to handle arbitrary network topologies and more exotic forms of transport. We also note that the semantics of the wiki page is simpler than that of an arbitrary file system operation. For instance, we don't support tracking atomic changes across multiple pages.

6. CONCLUSION

In this paper, we present DTWiki, a system for collaborative content creation in networks where disconnection is the expected norm. DTWiki offers the full power of an online wiki coupled with the ability to perform local edits and local content creation while partitioned from the network. The DTWiki system is adaptable to arbitrary kinds of network topologies and disconnection patterns. The adaptation of the wiki feature set to work in a disconnected was surprisingly straightforward. One factor that worked to our advantage is the simple semantics of the wiki application, which were easy to map to the TierStore coherency model.

On the technical front, we intend to investigate whether other pieces of online content management systems are conducive to being built in a manner similar to DTWiki. There are also opportunities in improving the integration of the data model to traffic prioritization. As the system is currently implemented, revision data is not prioritized for newer copies of the wiki page. Ideally, fresher pages should be delivered in front of older pages, presumably because the older revisions are less useful. Also, there may also be a better data partitioning to enable finer grain sharing than whole wiki namespaces.

Our experience with DTWiki has found that while the consistency model of TierStore is appropriate for the wiki application, the file system programming interface is somewhat awkward for application programmers. We are investigating a separate database oriented interface to TierStore to allow for easier integration with existing database-based software.

It is our suspicion that there is much to be gained by enabling bidirectional sharing of information in the unstructured, easy-to-use format of a wiki. Not only will such a system enable the generation of local content, but the presence of a globally shared "white board" can be adapted by hand to a myriad of different applications. We plan to integrate the DTWiki system with efforts such as the AVOIR [1] in a real-world deployment to test our hypothesis.

7. REFERENCES

[1] AFRICAN VIRTUAL OPEN INITIATIVES AND RESOURCES. http://avoir.uwc.ac.za/avoir/index.php.

[2] ASIA FOUNDATION. Cambodia Information Centers. http://www.cambodia.org.

[3] CERF, V. ET AL. Delay-tolerant network architecture internet draft, Sept. 2006. draft-irtf-dtnrg-arch-05.txt.

[4] CUNNINGHAM, W., AND LEUF, B. *The Wiki Way: Quick Collaboration on the Web*. Addison-Wesley Professional, April 2001.

[5] DEMMER, M., BREWER, E., FALL, K., JAIN, S., HO, M., AND PATRA, R. Implementing Delay Tolerant Networking. Tech. Rep. IRB-TR-04-020, Intel Research Berkeley, Dec. 2004.

[6] DEMMER, M., DU, B., AND BREWER, E. TierStore: A Distributed File-System for Challenged Networks in Developing Regions. In *FAST: File and Storage Technologies* (Feburary 2008).

[7] FALL, K. A delay-tolerant network architecture for challenged internets. In *SIGCOMM 2003*.

[8] MEDIAWIKI. http://www.mediawiki.org.

[9] MERCURIAL: A LIGHTWEIGHT SOURCE CONTROL MANAGEMENT SYSTEM. http://www.selenic.com/mercurial/wiki/.

[10] MOHAN, C. A database perspective on lotus domino/notes. In *SIGMOD Conference* (1999), p. 507.

[11] MOULIN. Wikimedia by moulin. http://www.moulinwiki.org/.

[12] M.S. SWAMINATHAN RESEARCH FOUNDATION. http://www.mssrf.org/.

[13] NOFFLE NEWS SERVER. http://noffle.sourceforge.net.

[14] OFFLINEIMAP. http://software.complete.org/offlineimap.

[15] ONE LAPTOP PER CHILD PROJECT. http://www.laptop.org/.

[16] PENTLAND, A. S., FLETCHER, R., AND HASSON, A. Daknet: Rethinking connectivity in developing nations. *IEEE Computer* (Jan. 2004).

[17] PHPWIKI. http://phpwiki.sourceforge.net.

[18] PUKIWIKI. http://pukiwiki.sourceforge.jp.

[19] SETH, A., KROEKER, D., ZAHARIA, M., GUO, S., AND KESHAV, S. Low-cost communication for rural internet kiosks using mechanical backhaul. In *MobiCom '06: Proceedings of the 12th annual international conference on Mobile computing and networking* (2006).

[20] TERRY, D. B., THEIMER, M. M., PETERSEN, K., DEMERS, A. J., SPREITZER, M. J., AND HAUSER, C. H. Managing update conflicts in Bayou, a weakly connected replicated storage system. In *SOSP 1995*.

[21] THIES, W., PREVOST, J., MAHTAB, T., CUEVAS, G., SHAKHSHIR, S., ARTOLA, A., VO, B., LITVAK, Y., CHAN, S., HENDERSON, S., HALSEY, M., LEVISON, L., AND AMARASINGHE, S. Searching the world wide web in low-connectivity communities. In *WWW 2002*.

[22] TIKIWIKI. http://tikiwiki.org.

[23] TRIDGELL, A., AND MACKERRAS, P. The rsync algorithm. Tech. Rep. TR-CS-96-05, Australian National Univ., June 1996.

[24] TWIKI. http://twiki.org.

[25] VERPERMAN, J. *Essential CVS*. O'Reilly Books, 2003.

[26] WARD CUNNINGHAM. Wiki Engines. http://c2.com/cgi/wiki?WikiEngines.

[27] WIKIPEDIA. http://www.wikipedia.org/.

[28] Wizzy Digital Courier. http://www.wizzy.org.za/.

[29] WWWOFFLE: WORLD WIDE WEB OFFLINE EXPLORER. http://www.gedanken.demon.co.uk/wwwoffle/.