The C-LINK System for Collaborative Web Usage: A Real-World Deployment in Rural Nicaragua

Sibren Isaacman and Margaret Martonosi
Princeton University
{isaacman, mrm}@princeton.edu

Abstract

Information exchange is one of the most crucial elements of education and business in the modern world. Therefore, equipping developing regions with access to the internet is becoming increasingly important. Though many regions will not receive broadband, wired connections in the near future, limited connectivity is rapidly become available. The connectivity may come from the increasing density of mobile devices or from novel delay-tolerant approaches. In this paper we demonstrate a collaborative caching system we have developed that allows for such connectivity. Further, we report on our experiences deploying our system in rural Nicaragua. We show that collaborative caching allows miss rates to drop in half through the course of the deployment, and that user satisfaction remains high despite long round trip times. Although general web access is sometimes viewed as not amenable to collaborative caching and prefetching, our data shows value in collaboration, with 74% of pages accessed by more than one user.

1 Introduction

The modern internet serves as a hub of information regarding health care, education, and other topics, and is a vital communication pathway throughout the developed world. In North America, for instance, over 70% of the population has access to the internet's resources. In stark contrast, the world average is 20%, with some areas having internet penetration rates as low as 5% [13]. Closing this "digital divide" will become an ever more important and efficient method for alleviating poverty and making developing regions competitive on the world stage [14]. Thus, bringing technology to developing regions is one of the "Millennium Development Goals" named by the United Nations [21].

Though connectivity's importance is immediately evident, difficulties in bringing connectivity to these regions are daunting. The cost and impracticality of installing wired internet access makes ubiquitous wired connections a virtual impossibility [3]. Even wireless solutions, however, have been slow in adoption due to difficulties in erecting towers, licensing appropriate portions of the spectrum, negotiating land use, and navigating bureaucracy or corruption. Despite these issues, it appears that largely-disconnected regions will slowly gain access to the internet with the emergence of delay-tolerant and wireless approaches.

Nevertheless, for decades to come, we expect large numbers of people in rural regions to be frequently disconnected or operating on constrained bandwidth links (e.g. cellular). This paper explores collaboration and prefetching techniques for mitigating the latency and bandwidth constraints facing such users. In particular, we have built a transport-layer-agnostic system for supporting cooperative caching and prefetching of web data.

Why Collaboration? Collaboration offers several benefits over other approaches that may be considered in developing regions. First, by caching content collaboratively across many user machines, we reduce the reliance on any single machine. By removing the single point of failure, we make a more robust system. This is important since machines and even kiosks are likely to be unreliable and also due to the frequency with which power surges and losses occur. The relative instability of the power supply means that any system that is to find long term use needs to be to able adapt to the loss of any of its components. Second, a distributed approach allows user mobility; this is important given that cell phones and mobile devices are becoming the dominant computing platform in these regions. As the devices move from region to region, a distributed approach allows these users to continue to participate in the collaborative cache. In addition, by allowing these mobile users to participate, we provide a secondary source for content dispersal.

Having explored the potential for such a system in [9], we built a full implementation and evaluated our design with a real system deployment in Cinco Pinos, Nicaragua. This paper offers the following conclusions and contributions:

- We demonstrate that a web browser employing a collaborative caching system is both very useful and enjoyable to rural internet users, despite the high underlying latencies for web requests that miss in the collaborative cache.
- With the collaborative cache, miss rates as low as 22% are achievable even after only a few days of deployment.
- We show that minor changes to the definition of a cache hit and what constitutes cachability can result in highly improved usability.

The rest of this paper is structured as follows: Section 2 examines related work. Section 3 then describes our own system in more detail. The deployment of the system is discussed in Section 4 and some design decisions are discussed in more depth in Section 5. Finally, Section 6 offers some conclusions.

2 Related Work

User Applications in Developing Regions There have been a number of successful applications designed with developing regions in mind. The CAM project [15] developed region-specific finance and other applications for mobile phones. It demonstrated

NSDR'09, October 11, 2009, Big Sky, Montana, USA.

that it is possible to effectively remove barriers to entry and bring useful applications to rural users.

More closely related to our own project, TEK [20] allows for disconnected Web Browsing via email. Similarly, RuralCafe [4] attempts to perform local query refinement to reduce the number of required trips. Each of these approaches, however, uses a large local per-node store. By replacing these local stores with collaborative cache, we can either improve the effectiveness of the storage, or get the same effectiveness from a smaller store.

Delay-Tolerant Networks Typical networking applications assume that at all times an end-to-end connection exists from the client to the server. In many areas, even a low bandwidth link (e.g., cellular) does not exist. In such cases a DTN store-and-forward protocol can be used such that the data is stored in intermediate nodes until connections can be made that will allow the data to make progress toward its destination.

In a DTN, packets can be flooded to all encountered nodes (or a subset thereof) [11] or social networks can be learned and routing thus deduced [15]. Though such previous work focused on custom built hardware to forward specifically collected data, the Tetherless Computing Lab [1] and DakNet [16] have each built a DTN upon which more general internet traffic can move.

As deployed, C-LINK ran on top of KioskNet [1] but is flexible enough to be used with any network layer. We envision, ultimately, that connections between the kiosk and city may occur over hybrids of cellular networks and DTN using protocols such as Delay Tolerant Link State Routing [7]. Our solution is designed with this type of hybrid in mind.

Other early efforts at disconnected operation include the Coda [12] and Rover [10] projects. Each of these projects uses local caches to hide network disconnections. However, the reliance on specialized file systems (and thereby code changes in applications of interest) makes both these approaches unsuitable for developing regions with heterogeneous machines. Further, these approaches do not make use of the well-connected nature of the network that may exist within the villages themselves.

Web Caching Another area of related work is web caching. In the most general web-caching model, a client checks whether a page is locally cached at that machine. If the page is not cached, a request is sent to a proxy which in turn checks its own cache. Only if that fails will the request be sent further upstream to the servers. Naturally, multiple levels of proxies may exist between a client and server and the proxies can be either explicitly-called or transparent [2]. For our disconnected village model, we do not assume the highly-capable proxy/kiosk that would normally exist. Instead we show that a more collaborative cache technique can be employed. In collaborative caching, caches make requests to one another for data [8, 17, 22]. If the data requested exists in a cache with which the current cache is collaborating, the data can be sent from one cache to the other, obviating the need to go further out.

Hierarchical collaborative caches such as Squid [22] are distinct from the type of collaborative caching that we propose and do not address directly the needs of the developing world, frequently breaking under disconnection. Squid, in particular, is known to function poorly even in the case of intermittent connections, and not at all in offline mode. Unlike in other hierarchical caches, we assume there is only a single proxy at a time, and that the machine has relatively low storage. Except for having an internet connection (albeit weak), the proxy (kiosk) is largely indistinguishable from other participating nodes. Such assumptions do not fit typical hierarchical web caching, so we instead extend on collaborative caching and employ a peer-to-peer model, as in DitTorrent [18]. In this model, client peers that have the requested data can serve the data rather then requiring a trip over a slower upstream link.

Though collaborative caching was found by Wolman et al. [23] to provide only limited benefits, Wolman concluded that smaller populations (<20K users) saw the most benefits. Our system is designed for classrooms and internet cafes in rural settings where user counts less than 100 per village are expected. In this space, we are the first to study the effects of collaborative caching.

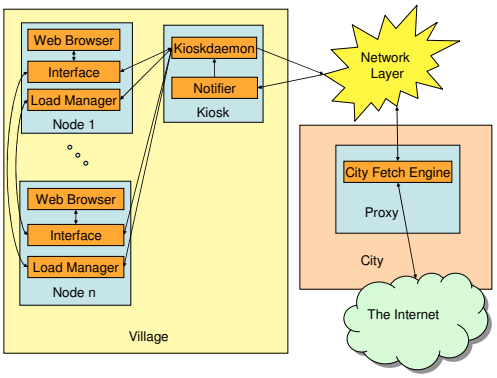

Figure 1. Diagram of the system. Nodes in the village are well connected (when the connection is functioning). The kiosk, in turn, is connected over a high latency (or high cost) link to the city, which in turn is connected to the internet.

3 The C-LINK System

Figure 1 is an abstract view of our system. The *village* is composed of devices (*nodes*) well-connected to a router (*kiosk*) which is in turn connected via a high latency or high cost link (or a combination of the two) to the *city* in which there is a proxy to the internet. The name C-LINK is derived from the five programs which make up the C-LINK system. These programs are: *City Fetch Engine*, *Load Manager*, *Interface*, *Notifier*, *Kioskdaemon*.

All web requests begin when a user makes a request (either by clicking a link or typing directing into the address bar) in a web browser which is configured to use the Interface as a proxy. By developing the front end as a stand-alone program that works with arbitrary browsers, rather than as an integrated component of a specific web browser, we ensure the flexibility of the system. Any browser can be used, so long as it is configured to connect to the internet via the C-LINK Interface. (Most modern browsers are easily configurable to connect via a proxy.)

The Interface then contacts the Kioskdaemon, which determines the availability of the page by consulting a mapping of hashed URLs to nodes in the network. Any data included in a POST request is also hashed and appended to the URL hash when the mapping is checked. What happens next is largely determined by whether the page exists in the cache. Section 3.1 follows a request that is found somewhere in the collaborative cache, while Section 3.2 explores what happens in the event of a cache miss. Section 3.3 discusses other issues that may arise in the functioning of the system.

3.1 Example Flow: Cache Hit

When the Kioskdaemon finds the hashed URL (plus data included in a POST request, if appropriate) in its mapping table, it replies back to the Interface with the last known good IP address for the machine holding the data. The Interface, in turn, contacts the Load Manager at the given IP address. The data is then sent from the Load Manager to the Interface and subsequently back to the web browser. The Load Manager is also responsible for LRU tracking on the requesting machine. Since this page is newly-accessed, it moves to the front (MRU) of the queue.

3.2 Example Flow: Cache Miss

If the hashed URL and data are not found in the Kioskdaemon's mapping table, the Interface is informed that the page is

currently not available. The Interface then returns a message to the browser informing the user that the page will be available later. The Kioskdaemon then makes a request to retrieve the page from the city. The page request—a file containing the socket dump from the browser—may be sent over any available network (e.g., cellular, KioskNet, etc.). It arrives in the city at the City Fetch Engine which connects to the remote servers as a proxy for the requesting user node back in the village. The City Fetch Engine opens the file and dumps its contents into the socket for the well-connected server to handle. The result of this communication is then placed into a file (another socket dump) and sent back to the requesting user.

The City Fetch Engine's always-on, end-to-end connection also allows the city to return more than was asked for. While the browser has no way of knowing *a priori* what further web requests might be made after this initial request, the City Fetch Engine is well-positioned to perform predictive prefetching. Prefetching can include simple strategies such as downloading all embedded images, following all links [5], or more complicated schemes relying on Markov models [19].

All requested and prefetched pages arrive back at the village via the Notifier. Because all data entering the village passes through the Notifier, it can estimate the amount of storage space currently being used. Using this estimate, the Notifier can decide to accept or reject data prefetched by the City Fetch Engine. Estimating space available in the village and acting accordingly helps maintain tradeoffs in performance and available space. All explicitly-requested (i.e., not prefetched) data is sent to the Kioskdaemon regardless of estimated available space. This ensures correct functioning of the system.

Once a request has returned to the Kioskdaemon, the Kioskdaemon updates its mapping to indicate that the file is now available in the village. The Kioskdaemon then sends the data to the Load Manager of the original requester. The Kioskdaemon also maintains a list of users that have also requested the file and notifies the Load Managers on the appropriate machines. If the original requester can no longer be contacted, the kiosk holds the file in temporary storage and marks itself as the owner of the file. If at some point later, an Interface indicates another request from that node user, the temporarily-stored file is returned to the user at that time.

3.3 Additional Issues

Cachable v. Non-Cachable Pages One design decision concerns the case when a request is made for a page that is in-village, but that the Kioskdaemon knows to be "non-cachable." In such situations, the Kioskdaemon returns the cached (stale) data to the Interface and makes a request for fresh data in the background. The cachability of a page is determined once it has been returned to the village; we simply parse the HTTP response header for fields related to cachability. Similarly, if a page contains a maximum age, this is recorded and when the age has expired, the page is changed in the index map from "cachable" to "non-cachable." By handling non-cachable content in this manner, we ensure that fresh content is always eventually available but that stale data, which may still be of use, can still be returned when the page is first requested.

Cache Eviction Eventually, a node's storage may fill up, either due to its own cached data or due to requests it is storing on behalf of the collaborative cache. The Load Manager sets cache policies to ensure that some nodes do not overuse the collaborative storage on other nodes. Quotas may be fixed or may be dynamically set based on fairness schemes or usage levels. If the local quota is exceeded (i.e., there is too much data stored in the cache), the Load Manager can request space in another user's cache via the Kioskdaemon.

When the Kioskdaemon learns from a Load Manager that cache overflow has occurred, it returns the IP address of the machine to whom the data should be sent (or tells the Load Manager to delete the data). In this way, users themselves need not keep any information about the other users in the village. The Kioskdaemon has flexibility in making its decision (e.g., emptiest, most frequently present, etc.) and can update its mapping appropriately. A Load Manager receiving a page from another Load Manager treats the page as if the page were explicitly requested and returned by the Kioskdaemon.

Node Mobility and Power Failure Nodes are free to move between sets of collaborative caches and power failures are likely to be common. Since users are likely to cluster (such as in libraries, schools, or community centers), multiple disconnected collaborative cache groups may exist within a village—each supported by their own kiosk. Further, nodes may move from one village to another. Because the kiosks serve as DHCP servers, the nearest collaborative cache group can always be found by contacting a known port at the gateway address. When moving from one collaborative cache to another, the data on the node can become integrated into the new collaborative cache in which the node participates.

Whenever a request is made for a page that is found locally (i.e., on the requesting node, rather than elsewhere in the village), the Interface informs the Kioskdaemon that the page is locally stored. This ensures that as nodes move from one collaborative group to another, pages it has stored can be found by other members of the "new" collaborative group. Conversely, if a page is requested from a node that has left the group, the Kioskdaemon behaves as in Section 3.2 and marks the page as pending.

Finally, the Kioskdaemon and Load Managers provide robustness against power failures. The state of the mapping and LRU queue are periodically saved in non-volatile storage (e.g., an SQL database or flat file).

Leader Election Although many nodes may exist in the village, each collaborative group should contain only one kiosk machine at a time. This is done so that only one mapping table needs to be maintained thereby removing consistency issues. Further, the kiosk node serves as a point of contact for the many mobile nodes that may move in and out of that village. Ensuring that there is only one kiosk at a time allows any node entering the village to immediately join the network, since the kiosk can act as a DHCP server. Therefore, should the kiosk machine fail, a new kiosk is elected from reachable nodes.

We assume that the kiosk machine is nearly identical to a generic node in capabilities and storage. Therefore, a kiosk election scheme such a a weighted random back-off is sufficient. In this scheme, when a node detects that there is no kiosk available, it broadcasts a message to all reachable nodes. Upon receiving such a message, each node backs off backs off a random time, weighted by its capabilities (e.g., communication capabilities). We envision the best kiosks to be under the supervision of a village leader, either in a school or community center, but any machine is eligible by default.

During the deployment of the system in Nicaragua, our leader election scheme was put into practice. We experienced a power event and the dedicated kiosk went off-line but users remained able to browse the available pages because a new Kioskdaemon was elected and spawned on one of the user machines. Without such a scheme in place, a power failure would have resulted in complete disruption of service. Instead, however, no time or data was lost.

4 Deployment Experiences and Results

In May 2009, we deployed the C-LINK system in Cinco Pinos, Nicaragua. This section discusses our experiences.

4.1 Overview

Having built and tested a prototype of our system, we brought it to Cinco Pinos, Nicaragua in May 2009 for a full-fledged deployment. Cinco Pinos is a small town (roughly 7000 residents) in northern Nicaragua near Honduras. While portions of the town

have electricity, power failures are frequent; indeed electricity outages occurred every day during our deployment and lasted from moments to hours.

At the time we selected Cinco Pinos (due to personal contacts there) and began planning our deployment, no cyber cafe's were publicly available within a one-hour drive. Subsequently, internet connectivity arrived in town in the form of a small cyber cafe, but with very low-bandwidth connection and high price relative to local incomes. We deployed C-LINK in a library, although we may work with the cyber cafe in the future.

4.2 System Setup and Usage

Our deployment consisted of 5 laptops running C-LINK clients connected to a kiosk computer running C-LINK on top of a KioskNet transport layer [1]. The client computers were Lenovo Netbooks with 1.6GHz processors and 1GB RAM running Hardy Heron Kubuntu. The kiosk computer was a Soekris box configured using the LiveDVD provided by [1]. We stress, once again, that although we used KioskNet for this deployment, our system can and has been written to work with a variety of transport layers.

The well-connected end of our DTN transport layer was a netbook computer and cable modem in a school in Somotillo, Nicaragua about one hour's drive (30 kilometers) away. Five buses per day travel between Somotillo and Cinco Pinos, but to ease the deployment, we drove the route ourselves once per day. Thus, our deployment offered a one-day turnaround time on web requests not serviced in the village.

Users of the system were teenagers and young adults engaged in educational opportunities offered by the two U.S. Peace Corps volunteers in Cinco Pinos. In particular, users were encouraged to browse and search the web on a variety of topics including health and medical issues and small business opportunities (particularly related to cooking and sewing). In addition, users also browsed on topics of their own choosing (with a heavy bias toward song lyrics!). Of the top-20 sites visited, 10% of them represented pages on assigned topics, while the remaining 90% were either casual browsing, ad sites, or pages we cannot easily categorize. Before deployment, we pre-loaded a very modest amount (<10 Mbytes) of web information into the villages netbook caches to allow for a better first-day experience, and we offered a home page with 16 working links to get them started.

The system had 31 unique users over the course of 5 days of use. Because we used a DTN with vehicular backhaul, they were each encouraged to come twice: on their first day, requests would be queued at the kiosk server. By their next visit, requests would have arrived by bus or car, and so their second session was typically about finding results of searches from the first session. There were no weekend sessions, though the 5 day period spanned a weekend. Therefore, for all latency calculations, the delay caused by the two days on which the school and library were closed has been removed, and instead a typical one day latency between days 2 and 3 is assumed. 81% of the users had used the internet before, but only 19% had used it more than 10 times in their lives.

4.3 Basic Results

Figure 2 shows the daily request count as the deployment progressed. The miss rate can been seen in the top stack of Figure 3. There were no user sessions over the weekend as the library was closed, so we plot only the days on which web usage (and data DTN trips to Somotillo) occurred. Miss rate monotonically decreases throughout the deployment, ending at 22%. Days 1, 3, and 5 mark the first visit by a group and Figure 2 shows the correspondingly higher numbers of requests. In contrast, as the groups returned on Days 2 and 4, we see sharp drops in the miss rate of requested pages as well as small numbers of requested pages while students browsed the returned pages form the prior session. Miss rate does not decrease all the way to zero because each day users click new links on the "frontier" of what is locally cached.

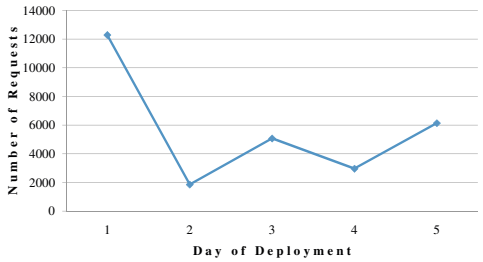

Figure 2. Total Requests per day. On a group's first day, more requests are made. The second day was spent reviewing the results of the previous day and therefore making fewer new requests. This page review then means that fewer total requests are made on the second day, which we see in the downward spikes every other day.

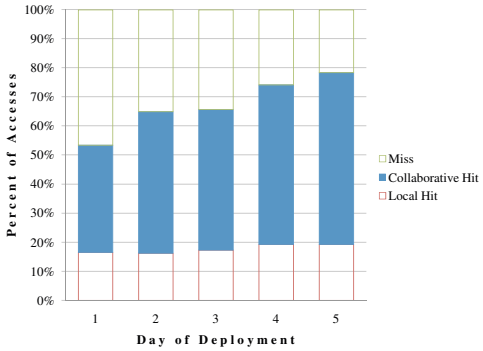

Figure 3. Breakdown of page request results. Day by day the hit rate increases as more information is collaboratively available in the village. Local hit rate remains relatively constant while collaborative hits drive down the miss rate.

Figure 3 shows the benefits of collaborative caching. By Day 5, hits on data stored locally at the requesting machine had reached 19% of all requests. This rate remained relatively unchanged throughout the deployment (rising only 2 percentage points). However hits on data stored collaboratively elsewhere in the village had soared to 59%. The jump in hit rate from 37% to 59% demonstrates the benefits gained by the collaborative portion of the cache.

Figure 4 plots the daily average effective access latency for the deployment. Any misses required a one-day turnaround time. (Recall that we drove to the city once per day; in a more long-term deployment, we could use the 5X per day bus service to reduce this miss latency.) While still high by well-connected standards, this latency is sufficient for web research. Users accepted the wait times and simply moved on to browse other pages that were already available in the system, incurring almost no latency.

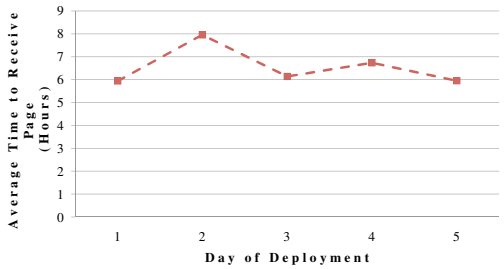

Figure 4. Average page access latency remains relatively constant.

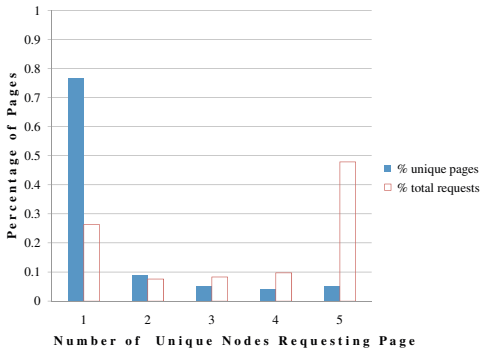

Figure 5. Percentage of unique pages and total requests for any page, broken down by number of requesters. Though only 33% of unique pages are requested by more than one user, 74% of total requests are to pages requested by more than one user.

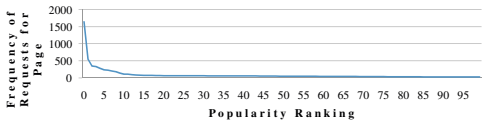

Figure 6. Number of requests to the 100 most popular pages. The first 5 most popular pages account for 11% of all requests.

Figure 5 shows the likelihood that a particular page is requested by more than one user. This data shows the benefits afforded by the collaborative cache. In particular, 33% of unique pages are requested by only one user. This means that in a traditional caching strategy, 33% of pages would have to be replicated more than once with 5% of pages fully replicated 5 times. On the other hand, many requests go to pages already accessed by another user in the system. As such, the collaborative cache requires only 66% of the storage of a traditional cache. This figure also demonstrates that while 33% of pages are requested by more than one user, those pages make up the bulk of all requests made to the system (74%). The large number of shared requests make collaborative caching particularly appropriate to this work load.

Finally, Figure 6 characterizes another aspect of the web referencing patterns during our deployment. The single most popular page is requested 5.8% of the time—nearly 3 times as frequently as the second most popular page—after which the number of requests per page drops slowly. The figure shows that a relatively small number of pages are accessed a large number of times in a correlated manner. This is in part due to the educational scenario in which many users are tasked with looking up similar information, but also holds true here despite the casual web browsing going on as well..

4.4 User Survey Results

At the end of each user's second session, we asked them to fill out a short multiple-choice survey (in Spanish) to gauge their experiences with the system and help us characterize the users. Users were asked to respond to questions about whether they felt the system was useful for education, whether they enjoyed using the system, if they found the information they were looking for, and what features they would like in future versions of the system. On questions such as utility and enjoyment, users could selected between "very", "a little," "not at all," and "no opinion."

Response to the system was very positive. Of users who had never used the internet before, 100% found C-LINK to be "very useful." In general, 77% of all users found C-LINK to be "very useful." All respondents that had only used the internet one or fewer times found the system to be at least "useful." Of the 31 users, 97% reported enjoying using the system at least "a little". Again, considering those that had only used the internet 0-1 times, 60% stated that that enjoyed using the system "a lot."

Finally, we note happily that the most common complaint in the comments section was that there were not enough computers for all users to have as much usage time as they wanted. It was rewarding to hear from students, none of whom had ever used a laptop (the touch pad was quite a shock to them!): "I learned a lot although it was only 2 days. Thank you. I hope you continue bringing projects to continue modernizing our municipality and our youth. Thanks."

5 Discussion and Design Decisions

Through the deployment, we learned things about the use and structure of our system that could be modified to improve performance in the field.

5.1 Cache "Hits"

During the deployment, we adjusted our definition of cache "hits" in order to improve the user experience. Our initial system would simply hash the full URL to obtain a key with which to perform a hit/miss check. Within a URL, some webpages (Google searches in particular) append arguments to a URL to guide their response. While some arguments are central to the information requested (in particular, the search terms themselves) other arguments may not significantly impact the information being returned. Thus, we developed a method to elide such arguments from Google searches before performing the hash and hit check. C-LINK thus returns information as a village cache hit based on the search term, rather than requiring a 100% URL match.

Google requests composed 7.5% of all requests made to the system. This change improved the miss rate on Google requests from 46% to 30%. Users reported that the improvement in browsing was significant.

5.2 Parallelization and Prefetching

Our initial coding for data retrieval was inefficient and relied on serial methods to fetch requested data. After 3 days of deployment, we rewrote this part of the code to parallelize it. Allocating 5 threads to data retrieval results in a 4X speedup. Therefore, from Day 3 onward, the results reflect our ability to prefetch more data while in Somotillo, thereby increasing the likelihood of hits on data never before referenced. Prefetched hits were only 2% of total hits, but provided large increase in user satisfaction when comparing comments from before and after Day 3. Further, throughout the course of the deployment, 48% of prefetched data was eventually used.

The importance of parallelization was demonstrated when power was lost in Somotillo 30 minutes into a data transfer on Day 3. Though previous trips had required an hour to download all of the initial requests, increasing threads allocated to the task meant that, within 15 minutes, all of the initial requests had been fetched.

5.3 Dynamic Kiosk Election

On May 22, a power event caused the kiosk to malfunction and force a restart. During this time, students were able to continue using the system. In the event of power failure, a heartbeat program was designed to elect one of the other machines as kiosk, gather LRU information from other Load Managers, and then serve requests. The logs of May 22 indicate that User 4 serviced 33 requests from 3 other users in the time the kiosk was down. Of these hits, 9 (29%) were for pages located elsewhere in the village.

This power event reiterates the need for a distributed, robust system in this environment. Had the system not been robust to power failures, such an event would have resulted in complete loss of access to data. C-LINK's approach successfully prevented such an occurrence.

5.4 Multiple Villages

In future deployments, multiple villages may share a single vehicle DTN route. In these cases, it is likely that some of the information that is requested by users in one village will be requested by users in other villages. This provides a unique opportunity to perform collaboration and prefetching between the villages as well as between users in a single village.

Collaboration between villages may decrease the latency of a requested page by serving it from a nearby village rather than waiting for the full round trip. In addition, the request trace from one village can be used to inform prefetching in other villages, further decreasing latency. We intend to implement this functionality in future deployments.

6 Conclusions

C-LINK's deployment demonstrates that it is possible to build a flexible, robust system for providing high latency internet connections to developing regions. Further, the usage patterns of students fit well with a collaborative caching model with 74% users interested in the same pages, resulting in miss rates that dropped to 22% in 5 usage days.

Through our deployment in Cinco Pinos, Nicaragua, we saw first-hand the importance of design choices that increased the robustness of the system. Though we were often without power, it was possible for the system to function as long as the netbooks retained their batteries, even without the original kiosk. Had we relied solely on a large, centralized cache, as might be done in a developed region, no pages could have been served during power failures.

We will provide our detailed, anonymized trace to the CRAWDAD repository[6] after publication of this paper.

7 Acknowledgements

We would like to thank Tanushree Dutta Isaacman for acting as a translator during our stay in Cinco Pinos. We also are indebted to Zimo Zheng and Lindsay Gehrig for their efforts in ensuring that there would be students available to make use of the C-LINK system and for designing lesson plans. Finally, we wish to thank all of the students and residence of Cinco Pinos, Nicaragua for welcoming us into their community and for their enthusiastic adoption of this technology. This work was funded in part by National Science Foundation grant number CNS-0614949. In addition, we gratefully acknowledge the support of the Princeton Technology for Developing Regions program, who provided a SEAS/PIIRS deployment grant to fund a portion of our efforts.

8 References

[1] A.Seth et al. Low-cost communication for rural internet kiosks using mechanical backhaul. *12th Intl. Conf. on Mobile Comp. and Networking*, Sept. 2006.

[2] G. Barish and K. Obraczke. World Wide Web caching: Trends and techniques. *IEEE Communications*, 38(5), May 2000.

[3] E. Brewer et al. The case for technology in developing regions. *Computer*, 38(6), May 2005.

[4] J. Chen, L. Subramanian, and J. Li. Ruralcafe: Web search in the rural developing world. *18th Intl. Conference on WWW*, April 2009.

[5] K. Chinen and S. Yamaguchi. An interactive prefetching proxy server for improvement of WWW latency. *7th Conf. of the Internet Society*, June 1997.

[6] A Community Resource for Archiving Wireless Data At Dartmouth. 2009. http://crawdad.cs.dartmouth.edu/.

[7] M. Demmer and K. Fall. DTLSR: Delay tolerant routing for developing regions. *ACM SIGCOMM Workshop on Networked Systems in Developing Regions*, Aug. 2007.

[8] L. Fan et al. Summary cache: A scalable wide-area web cache sharing protocol. *IEEE/ACM Transactions on Networking*, 8(3), June 2000.

[9] S. Isaacman and M. Martonosi. Potential for collaborative caching in largely-disconnected villages. *2008 ACM Workshop on Wireless Networks and Systems for Developing Regions*, September 2008.

[10] A. Joseph et al. Rover: a toolkit for mobile information access. *15th ACM Symp. on OS Principles*, 1995.

[11] P. Juang et al. Energy-efficient computing for wildlife tracking: Design tradeoffs and early experiences with ZebraNet. *Architectural Support for Programming Languages and Operating Systems*, Oct. 2002.

[12] J. J. Kistler and M. Satyanarayanan. Disconnected operation in the coda file system. *ACM Trans. on Comp. Systems*, 10, Feb. 1992.

[13] Miniwatts Marketing Group. World Internet Usage Statistics and World Population Stats, May 2008. `http://www.internetworldstats.com/stats.htm`.

[14] L. Osin. Computers in education in developing countries: Why and how? *Education and Tech. Series*, 3(1), 1998.

[15] T. S. Parikh and E. D. Lazowska. Designing an architecture for delivering mobile information services to the rural developing world. *15th Intl. Conference on World Wide Web*, 2006.

[16] A. S. Pentland, R. Fletcher, and A. Hasson. DakNet: Rethinking connectivity in developing nations. *IEEE Computer 37*, 1, Jan. 2004.

[17] A. Rousskov and D. Wessels. Cache digests. *Computer Networks and ISDN Systems*, 30(22-23), Nov. 1998.

[18] U. Saif et al. Poor man's broadband: Peer-to-peer dialup networking. *ACM SIGCOMM Computer Comm. Rev.*, 37(5), Oct. 2007.

[19] R. R. Sarukkai. Link prediction and path analysis using Markov chains. *9th Intl. WWW Conf. on Computer Networks : The Intl. Journal of Computer and Telecomm. Networking*, 2000.

[20] W. Thies et al. Searching the world wide web in low-connectivity communities. *11th Intl. WWW Conf.*, May 2002.

[21] United Nations. Millennium Development Goals. 2008. `http://www.un.org/millenniumgoals/`.

[22] D. Wessels and K. Claffy. ICP and the Squid web cache. *IEEE Journal on Selected Areas in Communications*, 16(3), April 1998.

[23] A. Wolman et al. On the scale and performance of cooperative web proxy caching. *17th ACM Symp. on OS Principles*, 1999.