Potential for Collaborative Caching and Prefetching in Largely-Disconnected Villages

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

ABSTRACT

In a world becoming ever more reliant on the power of information, bringing data connectivity into developing regions is becoming an important way to lift these regions out of poverty by educating and informing the population. Although many of these regions are not likely to receive the infrastructure to support fully wired (or even wireless) networks, existing cellular and delay tolerant technologies allow limited connectivity. In this paper we show that usability of highly disconnected networks can be increased through collaborative caching and data prefetching techniques. We focus on decreasing the miss rate of pages fetched in both general web access as well as more specialized education applications. We evaluate our schemes by running trace-driven simulations of internet traces from Cambodia and logs from Princeton University's Blackboard courseware web servers. Our caching and prefetching strategies in these environments show improvements in miss rate of up to 90% over more traditional approaches.

Categories and Subject Descriptors

J.1 [**Administrative Data Processing**]: Education; C.2.1 [**Computer-Communications Networks**]: Network Architecture and Design—*Store and forward networks*

General Terms

Design

Keywords

Caching, Prefetching, Delay Tolerant Networking

1. INTRODUCTION

As the internet becomes more pervasive in everyday life in the developed world, those who wish to be competitive in modern markets must have access to its information. Those unable to harness the internet's vast resources will be disadvantaged through a lack of powerful tools for communication, health care, and education [12]. The United Nations has established bringing technology to developing regions as one of its "Millennium Development Goals" to be achieved by 2015 for precisely this reason [19]. Though lack of connectivity is rarely an issue in North America, worldwide internet penetration rates (defined here as the percentage of the population with access to an internet connection point and the knowledge to use it) are just over 20% and in some regions are below 5% [11].

Clearly, the disparity in internet connectivity needs to be addressed, but major stumbling blocks exist to internet penetration in the developing world. The infrastructure to support full connectivity is often non-existent. The cost and logistics involved with laying wires into remote villages in developing regions turns the "last mile" problem into something far greater [4]. Often, a remote village can be a bus ride of many hours away from the nearest feasible internet access point. However, the advent of wireless technology has allowed us to move beyond wired solutions and directly to wireless solutions. Though even wireless solutions suffer from distance and infrastructure limitations (e.g., maximum ranges, tower height restrictions, etc.), they offer hope of bringing the internet to remote villages.

Regardless of the restrictions imposed by wireless technologies, we believe that villages are likely to have devices clustered tightly enough that well connected village networks can be formed; devices are likely to be grouped in schools, hospitals, or community centers. These village networks, however, will likely suffer intermittent, high latency connections to the rest of the world which serve as choke points to usability. Our work demonstrates that it is possible to leverage this well-connected nature of the computers within the *local* network to hide the poor connectivity of the village network as a whole to the outside world.

In this work, we explore the potential of two techniques. In the first, *collaborative caching*, computers within a village are able to access webpages stored on other connected machines. The second technique, *predictive prefetching*, delivers to a client or village webpages that have not yet been requested on the assumption that they will be requested in the near future. These techniques can greatly increase the usability of computer networks in remote villages even under the most naive implementations by decreasing the miss rate of fetched pages over implementations that rely on traditional caching strategies with no prefetching. Because miss penalties may be measured in hours, these reductions in miss rate result in large decreases in overall latency. Our evalua-

WiNS-DR'08, September 19, 2008, San Francisco, California, USA.

tion approach is trace driven, relying on traces obtained from internet kiosks in Cambodia [7] to represent typical developing region usage patterns and from Princeton's Blackboard webserver to represent typical educational usage. To our knowledge we are the first to evaluate the feasibility of these schemes in the context of developing regions.

The rest of the paper is organized as follows. In Section 2 we discuss work related to our own. In Section 3 we discuss a general overview of the uses of our systems. In Section 4 we discuss general characteristics of our trace files and how they were processed. We discuss our results and their implications as well as future improvements to the proposed system in Sections 5 and 6.

2. RELATED WORK

This section discusses the work upon which our own is based. We begin with a discussion of delay tolerant networks (DTNs) and educational applications and shift our focus to work previously done in web caching and predictive prefetching.

2.1 Success of DTNs

In typical networking applications, it is assumed that there exists, at all times, an end-to-end connection from the client to the server. If our connection to the village is via cell phone then end-to-end paths may exist but will be low bandwidth and unreliable. For many areas, however, no such connection exists at all. 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 towards its destination. Though the previous work focused on custom built hardware to forward specifically collected data, the Tetherless Computing Lab [18] and DakNet [14] have each built a network upon which more general internet traffic can move. We envision that our system could be built on top of this or any other framework for networking in developing regions. It is even possible that the underlying network is a hybrid of DTN and traditional end-to-end (though low bandwidth and unreliable) networks using DTLSR [6].

2.2 Distance and Computer Learning

Although we hope our system to be application independent, we are interested in exploring how application characteristics influence the effectiveness of our schemes. In particular, our current focus is on education applications whose correlated web accesses are readily amenable to our proposed scheme and which have the potential to make a big difference in the lives of children in developing regions. The Digital Study Hall project has demonstrated the enormous potential of distance learning applications in developing regions by prerecording instruction and using mail carriers to deliver the lectures to rural classrooms [20]. These lectures are recorded by experienced teachers in city centers and then presented by less experienced instructors in the villages. By using networked computers rather than videos and the postal service, we hope to improve even further on their results by increasing interactivity.

The usefulness of a computer network in learning environments can be easily seen simply by looking at the overwhelming success of Blackboard [2], a web-based utility for course management (including assignment distribution and submission), in university and even high school settings. Other projects, such as the open source KEWL [9] have taken the idea of blackboard and attempted to transfer it to the developing world with some success. Our caching strategies can make programs like KEWL more accessible to those in even more remote areas.

2.3 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 cache) and 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 a server and the proxies can be either explicitly called or transparent [1]. However, when caching in this manner, the network is assumed to have a hierarchical structure that may not exist. For our disconnected village model, a more collaborative cache technique can be employed. In collaborative caching, caches are able to communicate their contents (through message-passing [21] or directory based communication [15, 8]) and make requests to one another for data. 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 move farther up the hierarchy.

We have made no assumptions as to the mechanism by which consistency is maintained. Instead, we extend on cooperative caching because, while these solutions are very successful, they do not directly address the problems of the developing world. Each of the above solutions is assumed to work at the proxy level instead of the client level and thus each proxy itself serves many clients. Instead, we follow the example of DitTorrent [16] in allowing data to be retrieved from client peers. However, since storage space in the village is assumed to be highly constrained, we insist that there exist only one copy of any web page in the village, unlike other caching schemes. The last way in which our approach differs from common collaborative caching schemes is that we ignore the time-to-live field, reasoning that in a highly disconnected environment, stale data is preferable to no data.

2.4 Predictive Prefetching

In developing regions, networks are likely to have very high latencies regardless of whether the system is ultimately built on top of a DTN or a low bandwidth, unreliable cellular link. Therefore, the user should obtain as much data as possible when a connection can be successfully established (or in the case of low bandwidth links, while the user is otherwise occupied). Ideally, by prefetching a page while a connection is present, the page can be displayed instantly once it has been requested.

A number of systems have been proposed to aid in web prefetching. In the Mowgli system [10] the user marks pages of interest while browsing a current website and the system transparently begins to prefetch. Since the user is in control, this results in very few missed predictions although it requires a fairly high level of user sophistication and is not transparent. Alternately, the server can provide hints to proxies regarding what pages users are likely to access given the currently accessed pages [13], thus allowing the client to request those pages before the user knows of their existence. This requires modification on the server side. A

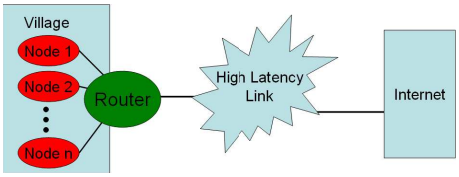

Figure 1: Diagram of the system. Nodes in the village are well connected with zero latency to each other and the router. The router, in turn, is connected over a high latency link to the general internet.

third approach is to simply scan the retrieved page and issue requests for all embedded links [5]. This approach, like the two previous approaches, might would work well over low bandwidth links but are unsuited to DTNs as each approach requires action on the client end resulting in a minimum of two round trips, each with high latencies. Markov models have also been used to aid in prediction [17]. The advantage of these models is that they can be used as the initial page is fetched. For this study of potential benefits, we limit our prefetching to a naive case with no server side support or client side action: making requests for entire folders when a single file is requested. In this way, our results represent a conservative scenario that can be greatly improved upon.

3. OVERVIEW OF APPROACH

This gives the reader an overview of the whole system into which collaborative caching and prefetching schemes fit. Section 3.1 describes the envisioned mechanism while 3.2 discusses general uses of the system. Our collaborative caching and prefetching techniques are described in Sections 3.3 and 3.4 respectively.

3.1 Description of System

The *village* is composed of devices (*nodes*) well-connected to one another each with a modest amount of storage. The nodes can connect to a router, assumed to have only enough storage to hold a few tables and connected via a high latency link to the internet (Figure 3). The latency of this link could be hours or more, particularly if DTN-style data mules are in use.

When a node in the village makes a request to the router for a page that does not currently exist in the collaborative cache, the router returns a signal telling the node to wait. The router then holds the request until it is able to be transmitted either to the unreliable cellular network or to a data mule (as in [18]). At that time it sends all requests for pages received since the last transmission and any associated prefetch requests. Upon receipt of the reply, the router distributes the requested data and pushes prefetched data on to appropriate local nodes. Any subsequent accesses to that page will be rerouted to the local node first to access the pages, thereby avoiding the high latency link. In order to maintain as much generality as possible, we do not make assumptions as to the type of link the data is sent over, though it is assumed to be of high latency. We assume that there are few enough devices close enough together that the cost of acquiring a page cached at a neighbor node is close to zero. This is reasonable compared to the delay involved with traversing the high latency links.

Our work analyzes the potential gains involved in adding collaborative caching and folder-level prefetching strategies. Although prefetching mechanisms may vary depending upon the link technologies employed (high bandwidth data mules v low bandwidth cell networks), generality is maintained by assuming appropriate mechanisms for prefetching exist for the technology. Further, since any technology used is guaranteed to be high latency, collaborative caching is equally general. Any reduction in the number of cache misses results in an increase in usability of the system due to the extreme miss penalty (hours) involved in fetching (or refetching) pages.

3.2 Usage

Such a system could be used for many applications, including education and general web access. Each of these applications would naturally bring with it unique traffic patterns as discussed below. In general web access, the router would act as a distributed performance enhancing proxy [3] to compensate for the high latency link between the village and a connection point. We expect a large number of unique URLs, some of which are accessed frequently (such as the front page of a news website) and others which are accessed much less frequently. The large number of URLs accessed makes prefetching difficult. However, the URLs that are frequently accessed are likely to benefit from collaborative caching.

Education applications (e.g., Blackboard [2] and KEWL [9]) are where we see the most promise for prefetching and collaborative caching. Since all students are likely to need to access the same data (e.g., assignments, study materials, etc.), sending a request once and then servicing it locally (as would happen in the case of a collaborative cache) has a potential to lead to huge savings. Further, since the pages have a structure to them (i.e., they are organized by class) prefetching is easy and effective: all the pages related to a class can be prefetched at once. In fact, if it is known that a village contains students currently enrolled in a given class, a teacher may opt to *push* data to the village rather than wait for it to be requested, thus further mitigating miss penalties.

3.3 Technique 1: Collaborative Cache

Collaborative caching is based on the premise that a web page accessed by one node in the village is likely to be accessed again, either by the same node or another node. If all of the nodes within the village can access data stored in each other's caches, we can reduce misses which would take minutes or even hours to service.

The advantage of this approach over one reliant solely on a proxy sitting at the router node is the scalability of the system. Though participation in the cache is voluntary, allowing others to utilize the storage of your machine gives a user access to the storage of the entire village. As nodes in the village are added to the cache, storage continues to grow. This is in stark contrast with only using a proxy, in which case storage would be limited until the village could afford to add storage without adding compute power. Further, by not relying on a single proxy for storage, the system no longer suffers from a single point of failure and allows power consumption to scale as needed. A proxy must be always on, but as village nodes connect and disconnect, power and storage can constantly be traded off.

When a URL is first accessed, it is stored in the local cache of the node attempting the access. From that point onward, until the data is evicted from the local cache, this node is the *owner* of this URL until a capacity miss occurs. Any node then attempting to access the URL would register a hit, either *local*, if the requester is the owner of the URL, or *collaborative*, if the URL is owned by another node in the village.

In the event that a URL must be evicted from the local cache, a least recently used (LRU) policy is used to choose a URL to evict. The node chooses the URL from its local cache that has been least recently used by *any* member of the village. However, before evicting the URL from the village cache entirely, the node searches for the emptiest cache among all the nodes. If there is room, it pushes the data onto that cache, transferring ownership. If the cache into which the data is placed belongs to a node that has never requested the data, it is put into *village* storage. If, however, the data is placed on a node that has in the past requested the data, it is placed into *hybrid* storage. A URL may move from village to hybrid storage if the URL is accessed locally in the future. A node never evicts data from another node's cache.

3.4 Technique 2: Prefetching

Our prefetching scheme occurs at the folder level. When a URL is requested by a node in the village, the filename is stripped from the URL and the system requests every file in the remaining *folder*. The contents of the folder are placed in the local cache of the node requesting access to the original URL and all the files are marked as owned by said node. Subsequent accesses to any of the files contained within that folder are then considered either local or collaborative hits, as described in Section 3.3. Although folder-level information may not always be available (and if folder information is available, it may not be the correct granularity at which to perform prefetching), we feel that the folder-level prefetching is an appropriate starting point that can be improved upon in subsequent work.

Prefetching at any granularity naturally incurs some bandwidth cost making any prefetching scheme highly dependent on the connection type. In the data mule model, bytes are cheap. Prefetching is done in a well-connected environment, the connection to the data-mule itself is very high-bandwidth, and the data-mule has a large amount of storage. In this case, an aggressive prefetching scheme is almost certainly worth the cost. In contrast, a low-bandwidth cellular connection may be too expensive to justify anything but user directed prefetching.

4. METHODOLOGY

This section starts by giving a high level picture of the traces involved in Section 4.1 and then Section 4.2 describes the behavior of the scripts used to analyze the traces.

4.1 Description Of Traces

First, in Section 4.1.1, we discuss traces taken from internet kiosks in Cambodia and provided by Du et al. [7]. We use these traces to represent typical web access patterns in developing regions. Second, in Section 4.1.2, we discuss traces taken from Princeton University's Blackboard Apache servers and which represent a typical education application. All traces were fully anonymized before we received them. The information available to use in both trace files was: anonymized requester IP address, anonymized URL requested, timestamp, and number of bytes returned from server.

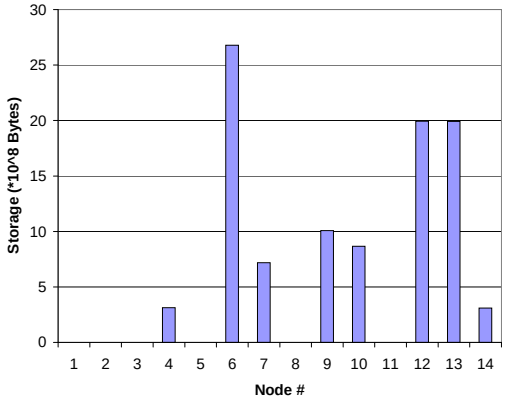

Figure 2: Storage requirements of individual nodes with no collaborative cache. There is a marked imbalance in the amount of data requested by nodes with some accessing more then 2.5GB and others requesting less then 100KB.

4.1.1 Cambodia

The Cambodia traces were taken over a two week period in August of 2005. Over these two weeks, fourteen unique nodes appear with some nodes requesting more then 2.5GB of data and others requesting less then 100KB. Individual storage requirements for the fourteen nodes can be seen in Figure 2. Although over 600,000 unique URLs were requested, the large majority of URLs are requested only once. Nevertheless, nodes that are requested on multiple instances may have been requested up to 29,000 times. This fits very well with our expected usage pattern for general internet usage.

4.1.2 Blackboard

The traces we analyzed from Princeton University's Blackboard servers were collected over a 24 hour period in March 2008. Over the course of 24 hours, 4,470 unique nodes appear in the trace file. The Blackboard users in general requested less data then the Cambodian users, with the heaviest user requesting just over 127MB of unique data. In the Blackboard trace, 700,000 URLs are requested, but only 200,000 of these are unique URLs, implying very high repetition rates and potential for caching. Again, this fits well with our expected usage of an educational application.

4.2 Description Of Script Behavior

As this is a trace driven simulation, nodes only appear in the village *after* they appear in the trace. This is a reasonable assumption, however, as the router does not necessarily know that a node want to participate in the collaborative cache until it has made a request. We must assume that once a node enters the village, it does not leave.

Once a node requests a URL, the number of bytes returned by the server is used to increment the amount of used space on that node. If zero bytes are returned, we assume that the request was a HEAD request or that the return code

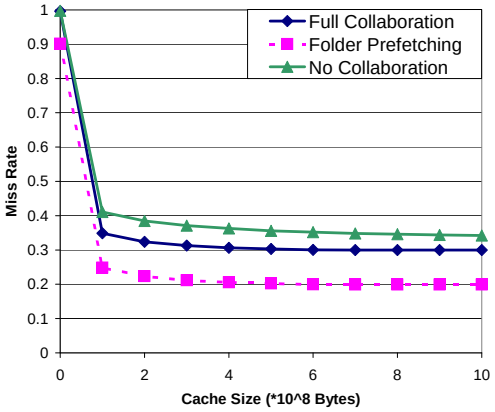

Figure 3: Miss rate v cache size in Cambodia traces. Improvements of about 5% can be seen without prefetching, expanding to 15% when prefetching is taken into consideration. These improvements are large considering the miss penalties.

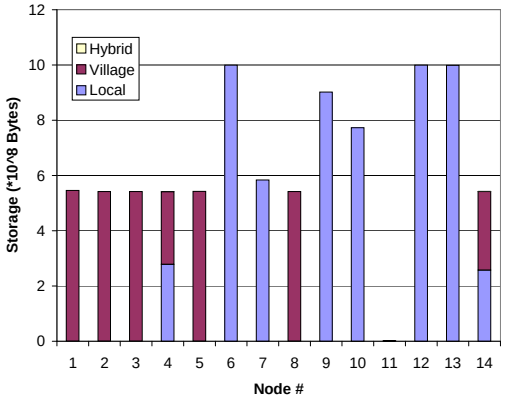

Figure 4: Amount and type of storage among nodes with 1GB local caches. Here we can see that those nodes that access more data have higher cache utilization while those nodes that access little data are evenly distributed at the lower level.

was 304 (unmodified). In either case, we assume the URL was still fetched and that the size is equal to that of the next non-zero appearance of that URL. The script then behaves as described in Section 3.3.

In the "No Collaboration" case, there is no collaborative cache. Each node in this case maintains an individual cache of it's own requests and a website may exists multiple times in the village. In this case, a miss will occur if a site is accessed by a node that has never accessed the site before, even if the site exists in the cache of a different node in the village.

When considering predictive prefetching, script behavior was slightly modified. We always assume that the entire folder can be prefetched. If this URL has not been seen before but is from a folder that is supossededly cached, the used storage space of the owner of the folder is incremented by the size of the new URL and the access is recorded as a hit. This retroactive insertion of URLs into a cache may result in an access that should have been recorded as a miss instead being recorded as a hit (i.e., the data would have been evicted from the cache for space reasons before it was ever accessed). To compensate for this effect, if the most recently accessed URL from a folder is evicted, it is assumed that the entire folder has been evicted and subsequent accesses to the folder are correctly marked as misses.

5. RESULTS

This section is broken down into the cases of general web access (5.1) and an educational application (5.2).

5.1 General Web: Cambodia Trace

Even small caches see a benefit from collaborative caching. Figure 3 shows that even a meager 100MB cache on each node in the village reduces the miss rate from 41% to 35%. Including prefetching reduces the miss rate further to 25%. Although a 6%-16% decrease may seem slight, it is substantial when viewed in context. In a high latency environment (seconds to minutes over a cell network or hours over a DTN) the slightest reduction in miss rate increases usability. Further, if collaborative caches are used, the minimum miss rate (30%) is achieved (i.e., all misses are compulsory) with only 700MB of per-node storage.

Including folder level prefetching, the minimum miss rate drops to 20% with a similar amount of storage required. The minimum hit rate in the non-collaborative case is 33.5% and does not occur until the nodes have 2.5GB of storage. Once 700MB is reached, storage is well divided among the nodes with those nodes storing local data storing slightly more. As cache size increases, this effect become more pronounced as can be seen in Figure 4. Node 11 does not appear until the very end of the trace and thus receives little data. Though collaborative hits remain fairly consistent (around 40% including just the effects of caching and 50% with prefetching), there is a slight maximum around 300MB caches. At this size, the caches are still too small to contain most of the data accessed by a node, but by allowing each node to access the caches of the other nodes, the effective cache size grows large enough that the hit rate is significant. Collaborative hits contribute 50-60% of all hits, regardless of cache size.

5.2 Blackboard Results

Even more than in general web usage, effects of collaborative caching in an educational application are noticeable quickly, as can be seen in Figure 5. With cache sizes of only 10KB, miss rate drops from 98% to 20% and falls even farther to 17% when prefetching effects are included. Unlike general web accesses, however, the miss rate continues to taper off slowly, reaching a minimum only at 10MB. The minimum miss rate without collaborative caching is similar to that of general accesses, 36%, but a collaborative cache brings the minimum miss rate to 3.9% and prefetching lowers it further to a mere 1.1%. Even the smallest storage sizes show a dramatic decrease in miss rate, bringing latencies down to tolerable levels after initial prefetching of the data by a single user. It is likely that the large number of users found in the blackboard trace exaggerated how small the cache could be and still function. Decreasing the number of users 1000x, however, would only necessitate raising the cache size to ĩ0MB, still easily achievable by devices

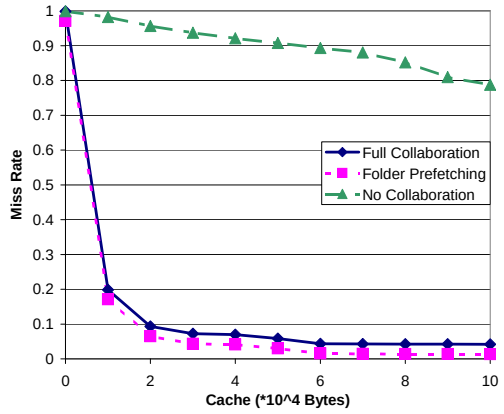

Figure 5: Miss rate v cache size in Blackboard traces. Including effects of folder level prefetching, miss rate is reduced by 89% over the standard case.

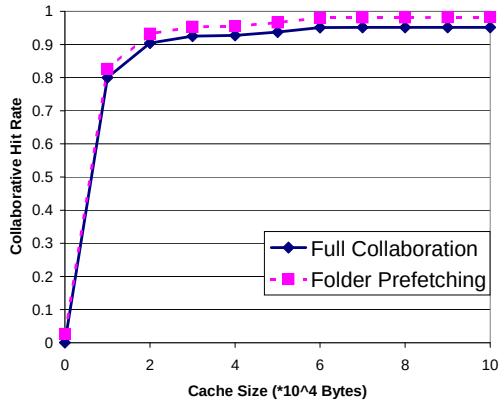

Figure 6: Collaborative hit rate v cache size in Blackboard traces. Collaborative hit rates reach 95% with 30KB caches when prefetching is included, indicating extremely high data reuse rates in the network.

in developing regions. Such drastic reductions in miss rate with the inclusion of collaborative caching are indicative of the fact that 99% of hits are collaborative as can be seen in Figure 6.

6. CONCLUSIONS

It is clear that general web access is difficult to improve. Nevertheless, the observed 6-16% reduction in miss rate improves usability drastically. While caching showed appreciable benefit, prefetching showed even more benefit indicating that there is high spatial locality in the data received and hinting that more sophisticated prefetching mechanisms merit future study.

Our simulated educational application benefited enormously from both of our techniques. 90% improvements in hit rate make an educational system an immediate candidate for real-world deployment. The gains in the educational application are due to the correlated nature of the usage in which multiple members of a class access the same data. Given that in such an applications data is broken down into distinct classes, an obvious further prefetch method would be to immediately download all the data from a class when any one piece is requested. High collaborative hit rates also indicate that more sophisticated "social" or "data-driven" caching techniques may be highly beneficial as well.

The advantage collaborative caching has over placing a proxy with large storage at the router in the village is primarily economic. Allowing the cache to be spread across all of the users in the system obviates the near for a large dedicated storage. Further, collaborative caching scales well. As more computers are purchased or donated to the school, the cache size increases. If the power costs of keeping a node running force the node to leave the village, the router will fail to retrieve the data and can behave as if the data has been evicted, thus scaling gracefully as nodes enter and leave. A proxy at the router, in contrast, would have to be "always on."

Overall, this work is intended as a proof of concept; these techniques have great potential for improving web access in disconnected villages. Though the approaches are naive, there are clearly appreciable gains; these techniques warrant a more in depth investigation. One key in proceeding further is understanding the differences in low bandwidth (such as over a cell phone network) and high bandwidth (such as via a data mule) connections, particularly with respect to prefetching, which may or may not be benificial. Our results show that collaborative caching and prefetching strategies can help bring the internet into disconnected regions and their classrooms thus help empower those children to learn and succeed.

7. ACKNOWLEDGMENTS

We would like to express a deep appreciation to Bowei Du for allowing us use of the traces from his team's research in Cambodia and to Thomas True and Kevin Perry for their hard work in creating the Blackboard logs. Without either of these traces, none of this would have been possible. This research was funded in part by the National Science Foundation through NSF grant number CNS-0614949.

8. REFERENCES

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

[2] Blackboard Inc. 2008. `http://www.blackboard.com`.

[3] J. Border. Performance enhancing proxies intended to mitigate link-related degradations. *RFC 3135*, June 2001.

[4] E. Brewer, M. Demmer, B. Du, M. Ho, M. Kam, S. Nedevschi, J. Pal, R. Patra, S. Surana, and K. Fall. The case for technology in developing regions. *Computer*, 38(6), May 2005.

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

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

[7] B. Du, M. Demmer, and E. Brewer. Analysis of WWW traffic in Cambodia and Ghana. *15th International WWW Conference*, May 2006.

[8] L. Fan, P. Cao, J. Almeida, and A. Z. Broder. Summary cache: A scalable wide-area web cache sharing protocol. *IEEE/ACM Transactions on Networking*, 8(3), June 2000.

[9] D. Keats. Knowledge Environment for Web-based Learning (KEWL): An open source learning management system suited for the developing world. *The Technology Source*, January/February 2003.

[10] M. Liljeberg, H. Helin, M. Kojo, and K. Raatikainen. Mowgli WWW software: Improved usability of WWW in mobile WAN environments. *GLOBECOM '96. Communications: The Key to Global Prosperity*, November 1996.

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

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

[13] V. N. Padmanabhan and J. C. Mogul. Using predictive prefetching to improve World Wide Web latency. *ACM SIGCOMM Computer Comm. Rev.*, 26(3), July 1996.

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

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

[16] U. Saif, A. L. Chudhary, S. Butt, and N. F. Butt. Poor man's broadband: Peer-to-peer dialup networking. *ACM SIGCOMM Computer Comm. Rev.*, 37(5), October 2007.

[17] R. R. Sarukkai. Link prediction and path analysis using Markov chains. *Proc. of the 9th International WWW Conference on Computer Networks : The International Journal of Computer and Telecomm. Networking*, 2000.

[18] A. Seth, D. Kroeker, M. Zaharia, S. Guo, and S. Keshav. Low-cost communication for rural internet kiosks using mechanical backhaul. *Proc. of the 12th Annual International Conf. on Mobile Comp. and Networking*, 2006.

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

[20] R. Wang, K. Li, M. Martonosi, and A. Krishnamurthy. Distance learning technologies for basic education in disadvantaged areas. *Technical Report TR-685-03, CS Dept., Princeton Univ.*, November 2003.

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