7DS - Node Cooperation and Information Exchange in Mostly Disconnected Networks

Suman Srinivasan, Arezu Moghadam, Se Gi Hong and Henning Schulzrinne
Columbia University, New York, NY
{sumans,arezu,segihong,hgs}@cs.columbia.edu

Abstract— **When the node density in a network decreases below the level necessary to sustain ad-hoc and mesh networks, communication can succeed only by leveraging node mobility and transitioning to message-based communications. In the 7DS (Seven Degrees of Separation) project, we have been investigating how to emulate two core Internet services, namely web access for information retrieval and email for delivering messages from mobile nodes to the Internet. We have implemented and evaluated a 7DS prototype system that leverages search, feedback and propagation limits to build a scalable system that can deliver data to and from mobile nodes.**

7DS makes data exchange in disconnected networks possible by providing an application-level set of protocol services that will enable exchange of information between peer devices. It enables dynamic information exchange by using a proxy server, a multicast query system, a search engine, and a transport entity. With these entities, 7DS can perform efficient and transparent data exchange among peers in the absence of a network connection. Data exchange with the larger Internet occurs when peers enter or exit the peer network.

I. Introduction

Wireless networks are widespread today. But despite the ubiquity of these networks at businesses and homes, users who need them the most, people who are often on the move, often lack cheap and easy access to the Internet.

7DS [1] is a unique system that allows wireless users to exchange data in a local disconnected network. This technique allows for successful exchange of relevant information based on two realistic assumptions - (1) that devices move in and out of the network, eventually connecting to the Internet and sending out information and bringing in new information and (2) there is a high probability that the information that is being searched for exists on a device in the near vicinity. (The second assumption is based on the fact that there are globally popular data items that most users would be interested in.)

7DS uses a proxy server that facilitates exchange of requested information among peers in the absence of a connection to the Internet. The proxy server uses the search engine and the multicast query engine to search the shared files on the device to see if there is a match. The system would thus transparently allow the user to exchange information by querying other nodes and searching their shared folders.

7DS also enables e-mail transport by implementing a Mail Transport Agent (MTA) that receives the e-mail and broadcasts it to the peers. When the peers reach the Internet, they forward the e-mail to an SMTP server that delivers the e-mail to the destination.

The 7DS system is described in more detail in the rest of the paper, which is organized as follows. In Section II, we introduce the problem of disconnected networks. Section III describes the approach to this problem and how 7DS is implemented. Section IV details the architecture of the 7DS system. Section V and VI describe the transport engine and community extensions to 7DS. In Section VII, we describe how the 7DS system was implemented and ported to embedded devices running Linux. We evaluate the performance of the system in Section VIII, and present related work in Section IX. Section X summarizes our paper and provides our ideas for future work on 7DS.

II. Problem

To facilitate information flow in a disconnected network described earlier, devices need to run a platform that enables data exchange within this disconnected network. In order to enable data exchange among peers in a disconnected network, a peer-to-peer (P2P) data sharing system is needed. Current P2P file-sharing protocols such as Gnutella [2] and BitTorrent [3] are built to run on connected networks with high-bandwidth. They are too heavy for mobile ad-hoc networks that are constrained by bandwidth and connectivity issues.

The 7DS system should be capable of setting up a P2P network that uses very little bandwidth and is also very robust. It should be able to work seamlessly in a highly mobile scenario where users are moving in and out of the ad-hoc network. It also has to be as interoperable, platform-independent and use resources sparingly to enable it to run on a variety of devices, from embedded systems to laptop computers.

III. Approach

The 7DS platform that we have implemented uses a very lightweight protocol involving simple XML messages for exchanging queries and responses with peers. It works seamlessly and transparently: in the absence of an Internet connection, the 7DS platform automatically queries its peers, retrieves the requests and presents the user with the data he has requested. Finally, the 7DS discovery service handles service discovery as well as discovery of neighboring nodes very efficiently, enabling the system to work robustly in dynamic scenarios.

The 7DS platform was designed as an application-layer solution running as daemons. All the components of the 7DS

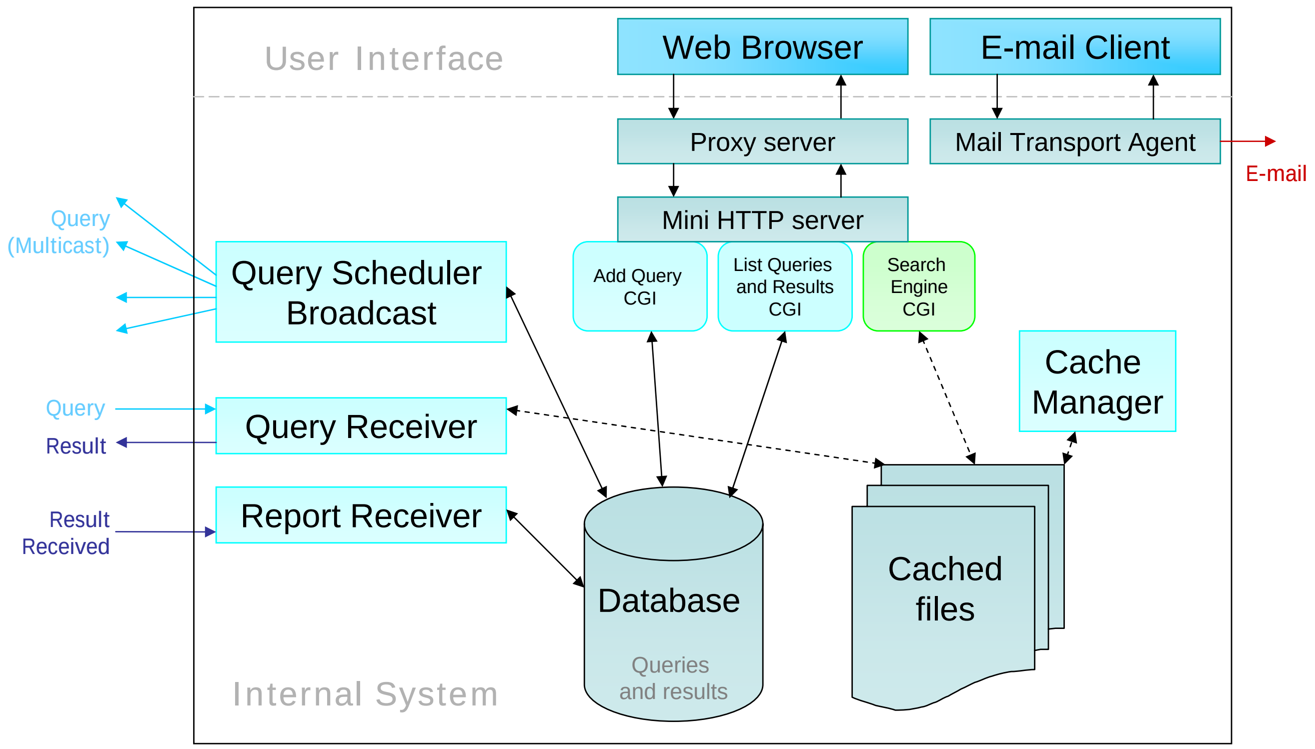


Fig. 1. The overall structure of the 7DS system, showing the search, multicast and transport engines

system can be fully utilized by popular existing software by simply changing a few settings, such as the proxy server setting on the browser for HTTP traffic, or the SMTP mail server on the e-mail client for e-mail traffic. Further, the 7DS components may, if required, be implemented as transparent proxies so that no reconfiguration of the client software will be needed at all.

IV. Architecture of the 7DS System

The 7DS system consists of application-level services and CGI executables. The components of the 7DS system consist of a proxy server, web server, search engine, multicast engine and a transport engine.

The proxy server provides the intelligence to the 7DS system, routing requests to the Internet or peers depending on whether connectivity is present or not. The web server provides the user interface and also allows files to be exchanged using the HTTP protocol. The search engine enables local searches, while the multicast engine enables searches across peers.

These binaries were developed at Columbia University's Internet Real-Time Lab and tested on several platforms. The components are described in this section. The details of the implementation and experiments are discussed in Section V.

A. Zero Configuration Networking setup

The discovery protocol of the 7DS system is mainly based on the Zero Configuration Networking specification (ZeroConf) [4]. ZeroConf enables devices to obtain IP addresses for network connectivity without a central DHCP server. It uses multicast DNS (mDNS) for name resolution, and either DNS Service Discovery (DNS-SD), Simple Service Discovery Protocol (SSDP) or Service Location Protocol (SLP) for service discovery.

In our implementation, IP addresses are allocated through a discovery protocol with a cross-platform implementation of Zeroconf called Howl [5]. (We will be using Apple's Bonjour [6] in future versions.) The 7DS discovery program uses Howl to publish a service description using ZeroConf publishing services to all the clients that are listening for the publish message. The program also acts as a ZeroConf subscriber so that it can receive messages that are being published. As services are removed and added, the discovered services are stored in memory. This enables the system to find services and their locations without a discovery server such as DNS Service Discovery.

B. Proxy Server

The proxy server listens to incoming HTTP requests. Based on the type of request and whether the device is connected to the Internet or not, the proxy server decides to serve the request from the cache, the Internet or through querying other 7DS system nodes via the 7DS multicast engine. The proxy server serves as the interface between the user, the Internet and other 7DS peers.

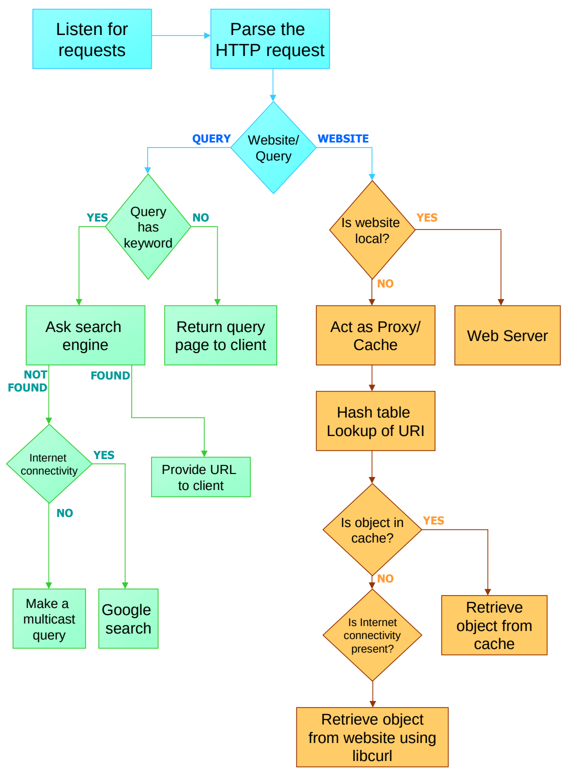


Fig. 2. The algorithm of the 7DS proxy server for handling HTTP requests

The proxy server, based on the incoming query, retrieves the data object most relevant to the user's request from the local cache or the Internet, in that order. The proxy server uses the libcurl [7] library in order to retrieve files over the network.

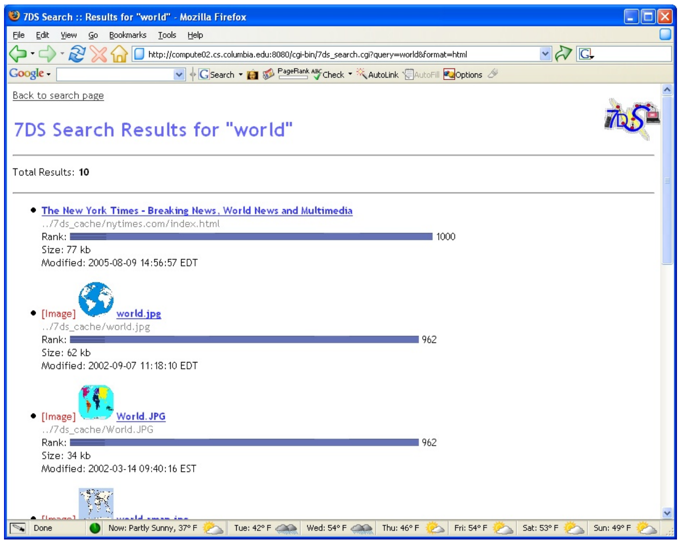
The algorithm used by the 7DS server to decide how to serve the client's request is outlined in Fig. 2. A separate service thread is created to handle each client.

C. Local Web Server

The web server on the 7DS system serves two functions. First, it runs the web-based user interface to the 7DS system. Secondly, it works together with the proxy server to display local cached results in the absence of Internet connectivity.

The web server should be capable of running on embedded devices. One such small open-source web server is thttpd [8]. The thttpd binary is only 49 KB in size, making it suitable for embedded devices. Another web server that is slightly larger in size but has more features is called lighttpd [9]. In addition, any web server that supports CGI and PHP can be used in conjunction with the 7DS system.

The 7DS system uses a folder where shared files are placed. This directory can be searched and indexed by the 7DS system components.


7DS Search :: Results for "world" - Mozilla Firefox
File Edit View Go Bookmarks Tools Help
Google
Search
PageRank
Check
AutoLink
AutoFill
Options

http://compute02.cs.columbia.edu:8080/cgi-bin/7ds_search.cgi?query=world&format=html

Back to search page

7DS Search Results for "world"

Total Results: **10**

- The New York Times - Breaking News, World News and Multimedia
  ../7ds_cache/nytimes.com/index.html
  Rank: 1000
  Size: 77 kb
  Modified: 2005-08-09 14:56:57 EDT
- [Image] world.jpg
  ../7ds_cache/world.jpg
  Rank: 962
  Size: 62 kb
  Modified: 2002-09-07 11:18:10 EDT
- [Image] World.JPG
  ../7ds_cache/World.JPG
  Rank: 962
  Size: 34 kb
  Modified: 2002-03-14 09:40:16 EST

Done | Now: Partly Sunny, 37° F | Tue: 42° F | Wed: 54° F | Thu: 46° F | Fri: 54° F | Sat: 53° F | Sun: 49° F

Fig. 3. The 7DS search page shows results for a keyword search. The results correspond to matching files in the local cache.

D. Search Engine

The working of the search engine and the multicast engine are shown in Fig. 1.

The search engine is built using the Swish-e library [10] that indexes HTML and XML files for keyword searches. Other file formats, such as Microsoft Word, Adobe PDF documents, and popular image formats such as JPEG, PNG and GIF, can also be indexed through suitable plugins. The Swish-e plugin that indexes the images by filename (JPG, PNG and GIF) was built at the Internet Real-Time Lab.

The search engine is a CGI binary that runs on the local web server. It provides the user the ability to find files corresponding to the requested keyword that exist in the device's internal database/cache.

A screenshot of the 7DS search engine in operation is shown in Fig. 3.

E. Cache Manager

The cache manager is a daemon that runs in the background at frequent intervals. The default interval is 20 seconds, but this can be modified through the configuration files. The cache manager checks if there have been any updates to the cache where the shared files reside and updates the indexes used to search the cache if necessary. If there have been no updates to files in that directory, then it just goes to sleep without taking any action.

F. Multicast Query Engine

The multicast query engine is used to exchange information among peers in the network. The user first enters a query through the 7DS web-based user interface. This query is added to the device's internal database. The user is then presented with a dynamic page that lists the results corresponding to the user's query. This dynamic page, which is generated by a CGI binary, refreshes every 10 seconds and provides the user with an updated result list.

For the multicast system to work seamlessly, the following components are needed.

The queries, results and corresponding peers are stored in a **SQLite query database**, which is a small-footprint, open-source database engine [11]. Unlike the larger and more popular database engines, SQLite does not require a daemon to handle SQL requests and is hence very suitable for our project.

The **query scheduler broadcast** engine broadcasts the query list in an XML-encoded string to the network. It reads the list of queries, encodes them in an XML-formatted string and broadcasts the string on a multicast packet. It sleeps for a small interval (20 seconds by default) and then resumes and broadcasts again.

The **query receiver** listens for incoming packets. Upon receiving a query list, it runs a local search on the device using the search engine. If related information is present, it encodes it in a RSS-based XML format [12] and sends the XML as a response in UDP packets to the requesting peer.

The **report receiver** listens on a UDP port for packets sent by the query receiver. Upon receiving the XML packet containing the response, it decodes and parses the XML. It adds the information about the queries, corresponding results and peers to the database table while avoiding duplication.

In addition to the daemon components that are running on the device, several CGI programs invoked by the web server provide the user interface to allow the user to add queries and to view the results. The CGI query page allows a user to add a query to the database. The CGI results page lists the queries that were made and also shows the results corresponding to each query. The results page automatically refreshes at regular intervals to return the latest results to the user.

V. Transport Engine

In addition to functioning as a query/response system, 7DS is also designed to perform data gathering and data delivery. The core communication protocol for this part is SMTP [13].

The SMTP server listens to incoming messages and dumps those which should be propagated through the network to the local Message Transfer Agent (MTA). The MTA unit later relays them to its neighboring MTAs. The SMTP server also takes care of managing and storing all the received e-mails in each 7DS mobile node.

The SMTP server receives the emails from the clients and creates a SHA1 hash of the email and recipient information. When the 7DS node meets another node, its MTA goes through the hash-table and the email directory, reads all the stored emails and sends them to the peer's MTA. When the node is connected to the Internet, the Transport Engine sends the e-mails to the intended recipient. Because of problems with e-mail duplication, we will explore the possibility of filtering the e-mails through a single server in future versions.

The library used to implement SMTP functionality is libESMTP [14].

VI. Community Extensions

We are currently working on extensions to the basic 7DS system to add user community functionality. While the services detailed above enable a user to share popular information that originates from centralized web servers, the community extensions will allow users to create their own objects such as events, calendars, maps and recipes and share them with other users..

The community extensions provide the user with a web-based interface through which the user can enter information for an event or another type of object. The data is stored in an XML file whose schema corresponds to a RELAX NG schema [15] for that object. Users can also define their own object types using another web-based interface that provides them with an easy-to-use interface to generate new RELAX NG schemas without having to understand the format.

The community extensions are currently under development.

VII. Implementation

The first version of the 7DS system written in C has been completed and tested on several platforms. A running 7DS system can be downloaded from the 7DS project web page [16]. 7DS has been compiled and tested on regular desktop versions of Linux, a small-footprint Linux running on an embedded platform called WRAP [17], as well as Windows and Mac OS X.

A. Experimental Setup

We ran the 7DS code on several computers in order to test the different parts of the system and see whether they performed as expected in providing Internet services in a disconnected network. Files from a few test websites were placed in 7DS caches of the different computers. Searches were performed to check whether the searches were successful and if files were found and transferred successfully.

The computers and devices that ran the 7DS code were all set up and tested in two wired networks (Computer Science department and Electrical Engineering department), one wireless network (Columbia University's Engineering School) and one ad-hoc network (in Columbia's COMET Lab). Our pilot test ran on Red Hat Linux, Windows, Mac OS X as well as an embedded system running a small-footprint Linux called LEAF.

B. Porting 7DS to WRAP

This section has details about how 7DS was packaged to run on PC Engines' WRAP (Wireless Router Application Platform). The WRAP platform presented several challenges.

The WRAP system is a board that is approximately the size of an adult's palm. The processor is a National Semiconductor Geode x86. It also has a network interface card and a wireless card. Our WRAP board boots off a 32 MB Compact Flash (CF) card attached to the board's card reader.

We installed a popular embedded version of Linux called Linux Embedded Application Firewall (LEAF) [18] on the CF card and configured it to boot from the CF card on the WRAP

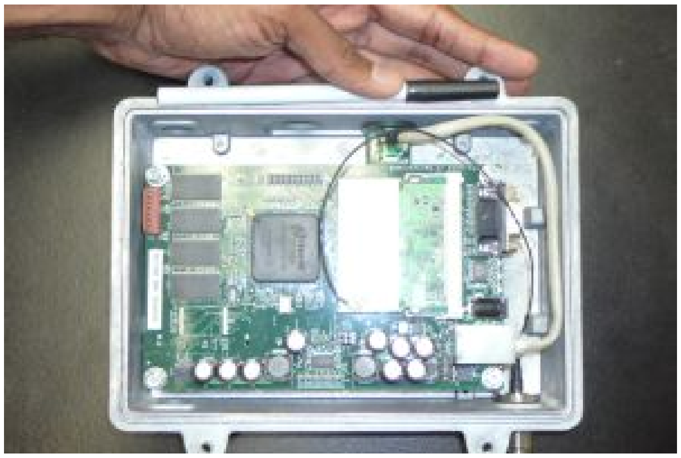
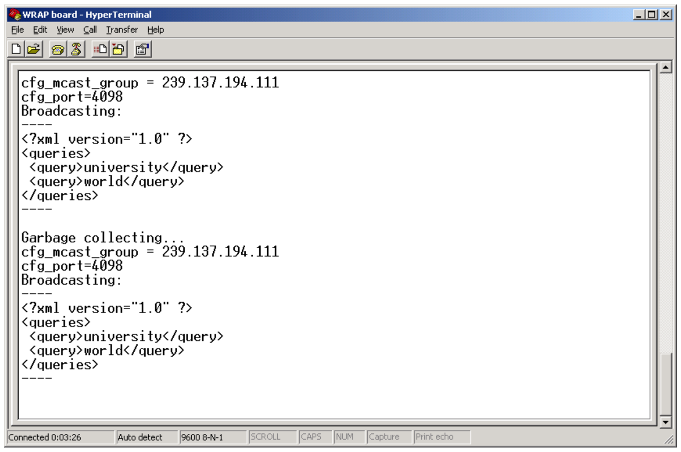


Fig. 4. The inside of the WRAP platform. The hand and the pen in the picture show how small the WRAP board is.



```
cfg_mcast_group = 239.137.194.111
cfg_port=4098
Broadcasting:
----
<?xml version="1.0" ?>
<queries>
 <query>university</query>
 <query>world</query>
</queries>
----

Garbage collecting...
cfg_mcast_group = 239.137.194.111
cfg_port=4098
Broadcasting:
----
<?xml version="1.0" ?>
<queries>
 <query>university</query>
 <query>world</query>
</queries>
----
```

Fig. 5. 7DS running on the WRAP platform. The screenshot shows a Windows HyperTerminal terminal interface with the WRAP board through a serial port connection. The WRAP board runs the *query scheduler broadcast* which broadcasts packets with the queries.

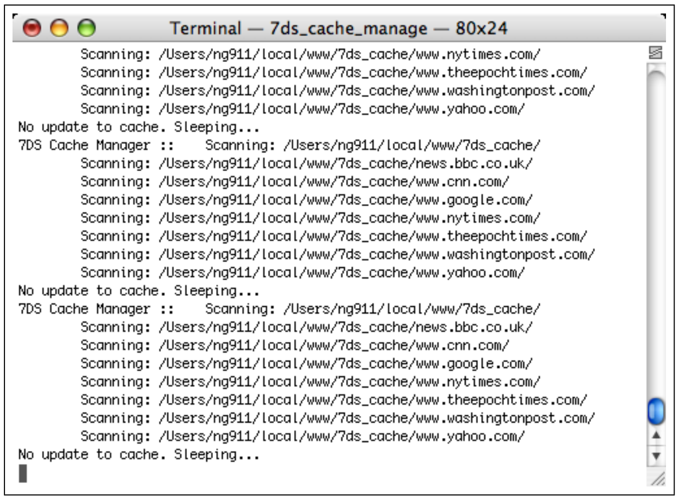
board. LEAF is a stripped down version of Linux that boots off most IDE, memory or other devices with a small-footprint kernel. Packages are added via LRP files which contain the binaries and metadata for running them.

The 7DS system was repackaged for LEAF by packaging the binaries in the LRP format. Also, because of the absence of libraries necessary for running 7DS on the LEAF platform, we also repackaged glibc 2.3, libcurl, swish-e, Howl and other Linux libraries in the LRP format.

C. 7DS on Linux, Mac OS and Windows

7DS was initially built on the Linux platform, and now exists as a GNU-style source distribution that can be built with configure/make commands. We are currently porting the 7DS system to run on the Mac OS X and Windows platforms as well.

7DS has now been compiled and tested on the Mac OS X 10.4 platform as well. In addition to the development utilities that come with Mac OS X, a packaging system called



```
        Scanning: /Users/ng911/local/www/7ds_cache/www.nytimes.com/
        Scanning: /Users/ng911/local/www/7ds_cache/www.theepochtimes.com/
        Scanning: /Users/ng911/local/www/7ds_cache/www.washingtonpost.com/
        Scanning: /Users/ng911/local/www/7ds_cache/www.yahoo.com/
No update to cache. Sleeping...
7DS Cache Manager ::    Scanning: /Users/ng911/local/www/7ds_cache/
        Scanning: /Users/ng911/local/www/7ds_cache/news.bbc.co.uk/
        Scanning: /Users/ng911/local/www/7ds_cache/www.cnn.com/
        Scanning: /Users/ng911/local/www/7ds_cache/www.google.com/
        Scanning: /Users/ng911/local/www/7ds_cache/www.nytimes.com/
        Scanning: /Users/ng911/local/www/7ds_cache/www.theepochtimes.com/
        Scanning: /Users/ng911/local/www/7ds_cache/www.washingtonpost.com/
        Scanning: /Users/ng911/local/www/7ds_cache/www.yahoo.com/
No update to cache. Sleeping...
7DS Cache Manager ::    Scanning: /Users/ng911/local/www/7ds_cache/
        Scanning: /Users/ng911/local/www/7ds_cache/news.bbc.co.uk/
        Scanning: /Users/ng911/local/www/7ds_cache/www.cnn.com/
        Scanning: /Users/ng911/local/www/7ds_cache/www.google.com/
        Scanning: /Users/ng911/local/www/7ds_cache/www.nytimes.com/
        Scanning: /Users/ng911/local/www/7ds_cache/www.theepochtimes.com/
        Scanning: /Users/ng911/local/www/7ds_cache/www.washingtonpost.com/
        Scanning: /Users/ng911/local/www/7ds_cache/www.yahoo.com/
No update to cache. Sleeping...
```

Fig. 6. The cache manager component running on a Mac OS X system

DarwinPorts [19] needs to be installed. DarwinPorts provides an open source packagement management system for the Mac OS platform. A screenshot of 7DS components running under Mac OS X is shown in Fig. 6. 7DS has also been ported to the Windows platform using the Cygwin [20] shell and GNU utilities that come with Cygwin. We are currently working on packaging the software binaries for easy installation on the Mac OS and Windows platforms.

D. Porting 7DS to Windows Mobile

While the original 7DS code was implemented in Linux, 7DS clearly benefits from having a Windows Mobile version as well. This would be a first step towards porting 7DS to a PDA or Smartphone platform, since the major Smartphone platforms almost exclusively support Windows Mobile as a development and runtime environment. A version of 7DS for Windows Mobile is currently under development.

VIII. Performance Evaluation

Yuen and Schulzrinne [21] have carried out an analytical study of the feasibility and performance of the 7DS system in disconnected networks. In particular, they have compared time-based and hop-based Time to Live (TTL) schemes. Some of their results are summarized below in the subsection analyzing performance evaluation of the 7DS e-mail system.

An important aspect of performance improvement measurements depends on the presence and density of wireless Access Points (AP). 7DS itself will benefit nodes in disconnected networks, allowing them to retrieve or send information through peers to the Internet, but absolute performance improvements in some applications - like e-mail sending - will depend on the AP density. If performance measurement is done in a state like North Dakota, where the population density - and hence wireless AP density - is low, the delay in e-mails reaching the server is much larger. However, in a densely populated area like Manhattan, 7DS will be of more use and help in reducing delays dramatically. A study performed in February 2002 [22] shows that there are over 13,000 wireless APs deployed in most areas of Manhattan.

A. E-Mail Performance

We will look at e-mail performance in detail. The most critical part of the e-mail delivery process is the amount of time the e-mail spends in the 7DS network itself. Once the e-mail reaches the Internet, delays are extremely minimal (in the order of seconds). Hence, we will attempt to quantify the performance boost due to 7DS in terms of improvement in delay while the node is in the 7DS network.

Yuen and Schulzrinne [21] find that message delivery in most of their target scenarios is of the order of 100 seconds, which is quite reasonable. They also find a e-mail queue storage size of 50 messages when the wireless AP is seventeen minutes away, 65 messages when the AP is thirty-four minutes away and 127 messages when the AP is eighty-three minutes away. Given the moderate size of e-mail messages, we believe that the storage-delay tradeoff is quite worthwhile.

Without 7DS, each node would have had to wait to get to the AP itself, and the delay would have been five to ten times as large. Even though more e-mails are stored on behalf of peers, the storage costs are a small price to pay for the reduction in delay.

B. Webpage Sharing

Unlike e-mail, sharing of webpages or websites is much more dependent on the data present in the local network. Hence, 7DS will highly improve performance when websites are requested in a disconnected network. Studies have shown that distribution of webpages in terms of popularity follows Zipf's law [23] [24]. Since the most popular pages will be requested by most of the nodes in the network, several of the nodes would have an updated version of the requested pages in their cache and could return them to the node that requests the webpage. Even though the page is slightly outdated, the retrieval of this information is still more useful than having to wait to get Internet connectivity.

IX. Related Work

It should be noted that several of the projects presented here, while similar in design to 7DS, are targeted at solving problems with mobile ad-hoc networks. While 7DS can be applied in a mobile ad-hoc network, it is meant to run on all disconnected networks where peers can interact.

In terms of file sharing, Gnutella [2] and BitTorrent [3] are the first two applications that come to mind. However, these protocols are designed to work with always-connected clients. Further, the base protocols for peer-to-peer file sharing applications are very inefficient and use a lot of packets to communicate and exchange files. These involve too much overhead for a mobile network.

JXTA [25] is a library that enables development of XML-based P2P protocols to allow peers in a network to interact with each other. However, just like the Gnutella and BitTorrent networks, JXTA is suitable for devices that are often connected to the Internet.

Hayes and Wilson [26] have built a platform based on Gnutella for sharing files on a peer-to-peer mobile ad-hoc network. However, they use the Gnutella protocol which includes routing capabilities that are not needed in the 7DS system. The 7DS protocol is much lighter and requires very little data to be exchanged. Further, by virtue of being an application level service, it is abstracted from the underlying network. Hence, in contrast to Hayes' work that runs only on Bluetooth, 7DS is capable of running on any network, be it Bluetooth, Ethernet, Wi-Fi or other networks.

Klemm, Lindemann and Waldhorst [27] have built a P2P file-sharing system called ORION (Optimized Routing Independent Overlay Network) for mobile ad-hoc networks. It uses an overlay network that combines application level query processing with network layer route discovery for file sharing. 7DS' multicast system works similarly, but without requiring a routing system. Further, 7DS enables a whole set of network applications, not just file sharing.

iClouds [28] is another P2P application that enables information sharing in mobile environments. iClouds is built on the J2ME platform. iClouds uses a UDP ping/pong mechanism to discover nearby services. In contrast, 7DS uses ZeroConf for service discovery. The iClouds' "virtual notice board" concept using information exchange of iHave and iWant lists is similar to 7DS' forthcoming community extensions, even though they are implemented differently. The 7DS community extensions allow a user to define his own class and object to share with the community.

Proem [29] is a platform similar to the 7DS system. Like 7DS, it is meant for P2P sharing on disconnected mobile ad-hoc networks. Proem is a protocol stack that allows other developers to build on top of it, but is not an application itself that can be deployed like 7DS. Again, in contrast with 7DS, Proem is also built on Java.

Earlier versions of the 7DS system were developed several years ago [1] [30], but they were written in Java. Our current implementation was built from ground-up in C, and it is hence smaller and faster than the previous version. Further, our implementation is small enough to be run on embedded systems with limited computing power.

X. Future Work

As future work, we would like to complete work on the 7DS community extensions and port 7DS to Windows and Windows Mobile platforms. We would also like to make the system more modular and open, so users and developers will be able to develop applications or tools for disconnected IP networks that run on top of the 7DS system.

XI. Conclusion

The 7DS system developed so far appears to fulfill its role in serving as a platform for exchanging information in a disconnected network. The components we have built so far enable webpages and e-mails to be exchanged within the disconnected network easily.

In the absence of ubiquitous connectivity, the 7DS system presents a good solution for implementing transparent data exchange in a disconnected network without the presence of the Internet. As devices join and leave the network, they bring in new information or carry out internal information that needs to be sent to the outside network.

Setting up 7DS on any device or computer is fairly easy. Once 7DS-enabled, devices can automatically exchange information to overcome the lack of Internet connectivity.

XII. Acknowledgment

This work was supported by the National Science Foundation (NSF) under Grant 04-12025.

References

[1] M. Papadopouli and H. Schulzrinne, "Design and implementation of a peer-to-peer data dissemination and prefetching tool for mobile users," in *First NY Metro Area Networking Workshop, IBM TJ Watson Research Center, Hawthorne, New York*, March 2001.

[2] "Gnutella." [Online]. Available: http://www.the-gdf.org/

[3] "Bittorrent protocol." [Online]. Available: http://wiki.theory.org/BitTorrentSpecification

[4] Zeroconf working group. [Online]. Available: http://www.zeroconf.org/

[5] "Porchdog software's howl." [Online]. Available: http://www.porchdogsoft.com/products/howl/

[6] "Apple computer's bonjour." [Online]. Available: http://developer.apple.com/networking/bonjour/

[7] "libcurl." [Online]. Available: http://curl.haxx.se/

[8] "thttpd, a small-footprint web server." [Online]. Available: http://www.acme.com/software/thttpd/

[9] "lighttpd, a small-footprint web server." [Online]. Available: http://www.lighttpd.net/

[10] "Swish-e search and indexing library." [Online]. Available: http://www.swish-e.org/

[11] "Sqlite database." [Online]. Available: http://www.sqlite.org/

[12] "The rss 2.0 specification." [Online]. Available: http://blogs.law.harvard.edu/tech/rss

[13] "Rfc 2821: Simple mail transfer protocol." [Online]. Available: http://www.ietf.org/rfc/rfc2821.txt

[14] "libesmtp." [Online]. Available: http://www.stafford.uklinux.net/libesmtp

[15] "Relax ng." [Online]. Available: http://www.relaxng.org/

[16] "7ds project home page." [Online]. Available: http://www.cs.columbia.edu/IRT/

[17] "Wrap board." [Online]. Available: http://www.pcengines.ch/wrap.htm

[18] "Linux embedded application firewall (leaf)." [Online]. Available: http://www.leaf-project.org/

[19] "Darwinports." [Online]. Available: http://www.darwinports.org/

[20] "Cygwin." [Online]. Available: http://www.cygwin.com/

[21] W. Yuen and H. Schulzrinne, "Performance evaluation of time-based and hop-based TTL schemes in partially connected ad hoc networks," in *Proc. IEEE ICC '06*, June 2006.

[22] "Public internet project - wireless ap density in manhattan." [Online]. Available: http://publicinternetproject.org/research/research_details.html

[23] P. Jelenkovic and A. Radovanovic, "Asymptotic insensitivity of least-recently-used caching to statistical dependency," 2003. [Online]. Available: citeseer.ist.psu.edu/jelenkovic03asymptotic.html

[24] V. Almeida, A. Bestavros, M. Crovella, and A. de Oliveira, "Characterizing reference locality in the WWW," in *Proceedings of the IEEE Conference on Parallel and Distributed Information Systems (PDIS)*, Miami Beach, FL, 1996. [Online]. Available: citeseer.ist.psu.edu/almeida96characterizing.html

[25] "Project jxta." [Online]. Available: http://jxta.org/

[26] A. Hayes and D. Wilson, "Peer-to-peer information sharing in a mobile ad hoc environment," in *Proceedings of the Sixth IEEE Workshop on Mobile Computing Systems and Applications*, 2004.

[27] C. L. Alexander Klemm and O. P. Waldhorst, "A special-purpose peer-to-peer file sharing system for mobile ad hoc networks," in *Proc. IEEE Semiannual Vehicular Technology Conference (VTC2003-Fall)*. Orlando, FL: IEEE, October 2003.

[28] A. Heinemann, J. Kangasharju, F. Lyardet, and M. Mühlhäuser, "iClouds – Peer-to-Peer Information Sharing in Mobile Environments," in *Euro-Par 2003. Parallel Processing, 9th International Euro-Par Conference*, ser. Lecture Notes in Computer Science, H. Kosch, L. Böszörményi, and H. Hellwagner, Eds., vol. 2790. Klagenfurt, Austria: Springer, 2003, pp. 1038–1045.

[29] G. Kortuem, J. Schneider, D. Preuitt, T. G. C. Thompson, S. Fickas, and Z. Segall, "When peer-to-peer comes face-to-face: Collaborative peer-to-peer computing in mobile ad hoc networks," in *P2P '01: Proceedings of the First International Conference on Peer-to-Peer Computing (P2P'01)*. Washington, DC, USA: IEEE Computer Society, 2001, p. 75.

[30] M. Papadopouli and H. Schulzrinne, "Seven degrees of separation in mobile ad hoc networks," in *IEEE GLOBECOM, San Fransisco*, November 2000.