

# 7DS - A Modular Platform to Develop Mobile Disruption-tolerant Applications

Arezu Moghadam, Suman Srinivasan and Henning Schulzrinne  
Columbia University, New York, NY  
{arezu,suman,hgs}@cs.columbia.edu

**Abstract**—Due to the low density of mobile devices or their limits in wireless radio range, a continuous network connectivity cannot be sustained in mobile disruption-tolerant networks. In these networks traditional networking models relying on end-to-end communication cease to work. These type of mobile disconnected networks can only support opportunistic, store-carry-forward communication. Furthermore, due to mobile nodes that dynamically join and leave the network the concept of a single, always-on server is unreasonable. In these scenarios, mobile users have to rely on node and service discovery and applications that cope with network disruptions.

7DS introduces a new platform to develop mobile applications for disruption-tolerant mobile networks. 7DS provides the necessary transport and application layer functionalities for mobile nodes to exchange information in store-carry-forward communication. 7DS has been developed as a modular platform that provides application developers with core underlying functionalities required for mobile disruption-tolerant communication.

We discuss the primary requirements of mobile applications and introduce two new applications implemented on top of the 7DS platform. These applications assist mobile users in data and event sharing in a server-less and disconnected networking environment.

## I. INTRODUCTION

With the growing number of mobile devices equipped with wireless interfaces, mobile users increasingly find themselves in different types of networking environments. These environments, spanning from globally connected networks such as cellular networks or the Internet to the entirely disconnected networks of stand-alone mobile devices, impose different forms of connectivity. This causes users to experience disruption in their service as they migrate from globally connected environments to isolated ad-hoc networks. Due to the high mobility, communication links in these remote islands of mobile nodes are transient and temporarily connected. Therefore, a continuous end-to-end path between a source and a destination can not be sustained. The lack of infrastructure in these self-organized mobile environments introduces another challenge in service provisioning. We classify these type of networks as disconnected or disruption-tolerant vs. the connection based model which is built on the assumption that the source and destination are connected for the duration of the communication session [1].

Traditional "connected" applications rely on interactive protocols in which a complete one-way message involves many source-to-destination signaling roundtrips. Such applications cease to work in disruption-tolerant environments due to the abrupt disconnections typical of such networks. The absence of

the end-to-end connectivity in these type of networks requires a different set of transport and application layer functionalities. Therefore, a new class of mobile applications has to be defined.

7DS [2], [3] was developed as a platform to address the previously mentioned connectivity problem in mobile networks by providing store-carry-forward communication [4]. 7DS [3] was composed of a bundle of applications for web query and email exchange in these networks. Implementing these applications revealed the need for a more standard software development environment. Such environment should facilitate application development by providing the underlying functions required for communication in disconnected networks [1] such as device discovery and connection set-up.

In this paper, we enumerate a class of disruption-tolerant applications that includes web query, email, file synchronization and bulletin board system. By considering these applications we determine a common set of primary functionalities to be provided by the software platform. These applications cover a broad range of core requirements in device and resource discovery, search engine, messaging and data sharing. We explain how we are evolving 7DS toward a generic software platform that provides application developers with these underlying functionalities. In addition, we explain how to use our new software platform to develop two new applications for mobile disconnected networks. These applications, named data synchronization and bulletin board system, are developed using tools and protocols offered by the new 7DS platform.

In Section II we motivate the necessity of data sharing and synchronization applications in mobile disruption-tolerant networks. In Section III we summarize the architecture of the system. Section IV explains the implementation issues we have encountered. Section V discusses the scalability of the system. Section VI briefly talks about the related works in the area. In Section VII we review our future direction and section VIII concludes the paper.

## II. MOTIVATION

The new 7DS platform is motivated and inferred by our earlier work on developing mobile applications for disconnected networks [3]. In [3], we developed web query and email exchange applications and discussed their necessity in disruption-tolerant environments. The choice of applications is important in determining the required core components for the platform. Here, we introduce two new applications to cover a

wider range of requirements. Later, in Section III, we illustrate the components that have been added to the 7DS architecture to realize these applications.

The ultimate goal of mobile disruption-tolerant applications is to maximize information propagation to the global Internet. Therefore all of these applications require the same common capability in data sharing and distribution that has to be provided by the platform. The "data sharing" concept can be modeled from different perspectives as described below. We further motivate the need for each model by some example scenarios.

### A. File Sharing and Synchronization

File synchronization application assists mobile disconnected users in different scenarios. For instance, mobile team members developing documents in a collaborative environment are able to share their modifications with others. In another scenario, using a mobile file sharing application facilitates schools to organize portable classrooms outside the campus. Using this application teacher and students are able to share the updated course materials with each other without any central server or infrastructure.

Mobile hosts might produce new files such as documents and pictures while disconnected from the global Internet. They might also download and cache some web content when they go back online. To maximize data availability in the disconnected network, mobile users should share these data objects with other mobile peers. Epidemic replication [5], has been proposed as a technique to distribute data among mobile nodes in disruption-tolerant networks. However, due to possible modification of these replicated files in each host, inconsistency results among the distributed "copies" over time. Therefore, a file synchronization mechanism is required to reconcile the inconsistent files when mobile nodes are in physical proximity. All hosts ideally should have the most up to date copy of the file.

The file sharing problem has been studied in maintaining large replicated collections of files or documents in connected distributed environments [6], [7]. Systems like CVS [6] manage multiple revisions of the same file by storing the current versions on a central server. Lacking central servers in infrastructure-less mobile environments turns file sharing into a challenging problem. In contrast with an asymmetric server-based model, in which files are added to and downloaded from a central repository, here versioning and synchronization should be managed in an entirely peer-to-peer fashion. Every host should be able to add files to other hosts or download from them directly through symmetrical uploads and downloads. Furthermore, due to intermittent connections with unpredictable lifetime, communication costs to exchange files should be minimized to save link bandwidth. For example, in synchronizing large files, transmitting the entire file for each incremental update is a waste of bandwidth and time.

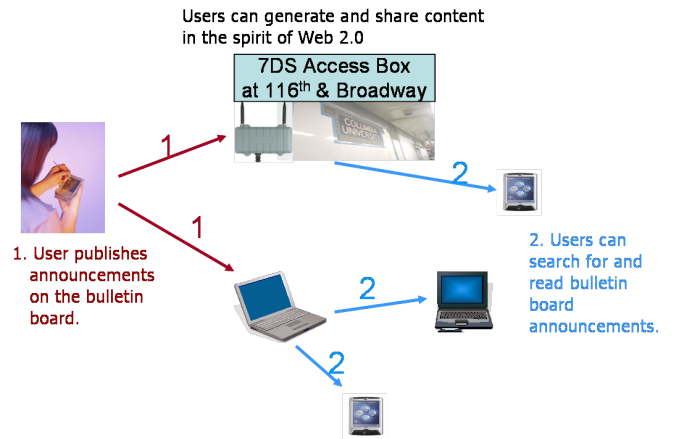


Fig. 1. Using BBS, mobile users after publishing announcements at step 1 can review and download the related information at step 2.

### B. News Sharing in a Bulletin Board System (BBS)

File synchronization, as explained in II-A, is classified as a "pull-based" data sharing model. In the pull-based file synchronization model, mobile applications are able to download new files from newly discovered mobile peers automatically. These uploads and downloads are transparent to the application. In a "push-based" model however, information is shared based on applications preference. The bulletin board system for news sharing falls into this category.

The necessity of sharing news through an ad-hoc bulletin board system could be elucidated considering some example scenarios. For instance, students walking into a classroom might have visited various locations on the campus and have collected advertisements about upcoming events. In another scenario, customers strolling in a shopping mall might collect product advertisements from different stores in their handheld devices. They are able to distribute these advertisements from device to device among other shoppers using 7DS BBS.

Bulletin board systems in connected environments are implemented by dedicating central message centers. Using central servers to post and review messages is not feasible in self-organized disconnected mobile networks. 7DS BBS enables mobile users to post and review community news in these type of networks in a peer-to-peer fashion.

Most bulletin boards serve specific interest groups. Mobile users choose to download information based on their significance after reviewing the advertisements. Unlike automatic updates in the pull-based file sharing model, BBS gives more freedom to the users to just download information they are interested in. In the BBS shown in the Figure 1 mobile users can review events advertisements at step 2, after their publication by the publisher at step 1 without any pre-configured infrastructure.

An alternative way of design is to build a BBS by installing stationary message repositories equipped with wireless interface in some specific areas. These infostation based model repositories [8] provide high bandwidth wireless connectivity in isolated coverage areas. They advertise information to the

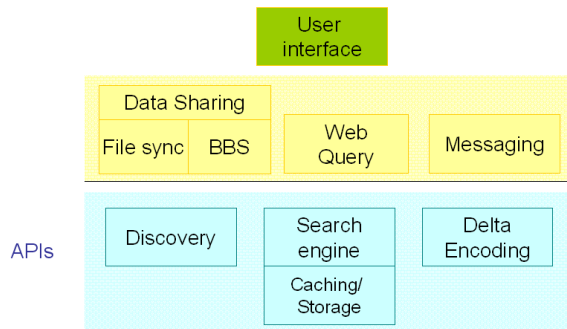


Fig. 2. 7DS modular architecture.

mobile users as they pass through their proximities. Mobile users communicate with these infostations to review or post messages. However, in the 7DS BBS model, mobile hosts communicate directly with each other in an entirely peer-to-peer fashion.

### III. ARCHITECTURE OF THE SYSTEM

7DS has been developed as a platform to provide necessary core components for disruption-tolerant communication. This platform constructs an abstract layer to conceal networking details and disconnections from the applications. We are building the 7DS platform as shown in Figure 2 as a modular architecture. This will allow us to expand the system by developing necessary plugins or modules for the future applications. The current components will be reusable without having the developers to re-implement them. In our previous work [3] we discussed and developed necessary modules for messaging and web query such as caching and search engine. Here, we extend the architecture by adding new components for data synchronization and delta compression and improve the discovery module as explained below.

#### A. Discovery Module

The discovery module is built on top of the mDNS [9] protocol. It enables mobile users to create an IP network automatically without any server such as DHCP [10] or DNS [11]. Furthermore, it helps 7DS users in the physical proximity to announce and locate the presence of services. In the 7DS platform file versions and events to be posted on the bulletin board, and all other services each node offers are registered and advertised through the discovery module. After learning about these services, applications willing to exploit them resolve each service to its corresponding location. Discovery module provides applications with mDNS protocol capabilities to announce, discover and resolve services. Discovery module is one of the core components of the 7DS platform.

#### B. Data Sharing

Protocols that enable mobile peers to share and synchronize data have been bundled in the data sharing module. This component determines inconsistencies in shared data among mobile applications and reconcile them by applying a predefined policy. In the pull-based model peers in physical

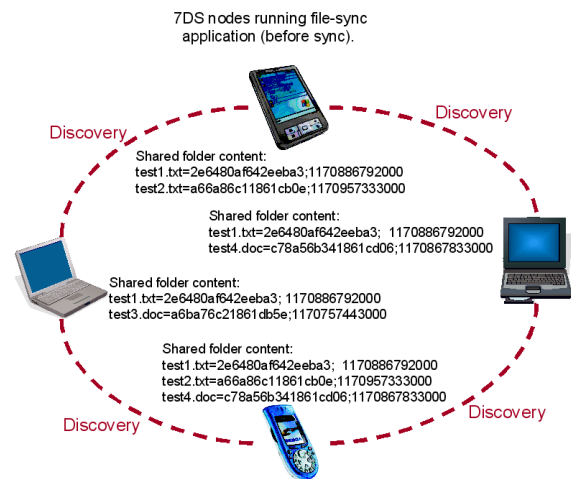


Fig. 3. Mobile nodes discovering other peers and file information lists offered by them. Each file information embodies a (hash value, date) tuple.

proximity synchronize their shared objects automatically and transparently from the application. The reconciliation policy is to update the shared directory in all peers to the latest version of data objects through automatic uploads and downloads. On the other hand, in the push-based model data is shared after reviewing the advertised metadata by the user. The goal here is to distribute data based on group interests rather than reaching a consistent state in all hosts.

1) *File Synchronization*: Mobile hosts willing to share data objects with others define a shared directory in which all shared objects are placed. Applications commit their modifications into this directory. This component contains a version controller which is devised to keep track of multiple versions, created over the network. The version controller scans the shared directory upon any insertion or deletion to determine and annotate all modifications. After this step, the version controller announces these versions by multicasting tuples of hash values and modification dates through the discovery module. Upon discovery of new nodes, file information lists are exchanged among nodes as shown in Figure 3. The version controller residing on each mobile host compares file information lists in pairs. The shared directories missing a file or containing some older versions are reconciled automatically to the most up-to-date versions. The synchronization request is issued by the file-sync module on behalf of the application. After reconciliation all participating nodes have a global consistent view of the shared folder as shown in Figure 4.

2) *Bulletin Board System*: As part of the data sharing system the BBS module is responsible to exchange data in push-based mode. Unlike the file-sync module data exchange does not take place automatically and transparently to the application. This module is suitable to distribute information based on group members' interests. Metadata is exchanged after discovery of new mobile users. After reviewing the metadata based on the category and type of information advertised by the peer, a download request might be generated. Similar to

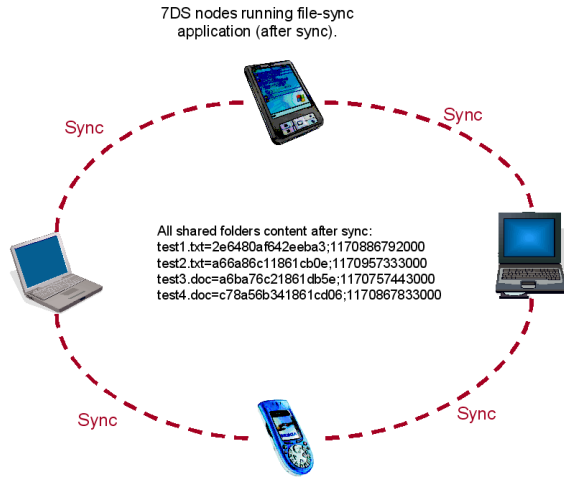


Fig. 4. Mobile nodes' shared folders after file synchronization.

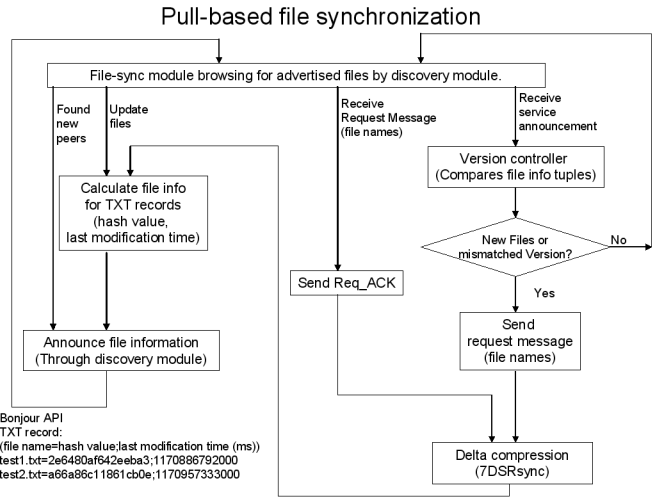


Fig. 6. File synchronization algorithm

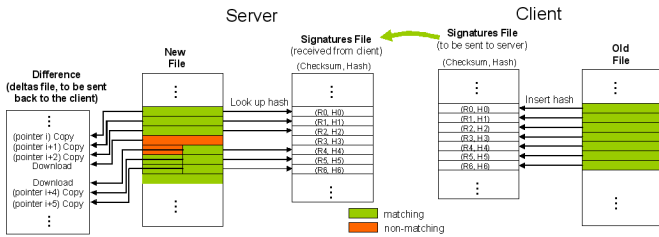


Fig. 5. Client generates signature file and server creates delta file according to the Rsync delta encoding algorithm.

the pull-based data sharing, all advertisements are announced and discovered through the discovery module. Information is exchanged in an XML-based data structure format for describing events and advertisements.

### C. Delta Encoding

In disconnected environments, contacts are intermittent with unpredictable lifetime. So a cost efficient file synchronization mechanism should be devised to save the connection bandwidth. Usually most of the update requests cause the retrieval of slightly modified instances of a resource for which the client already has a stale entry. Rsync delta encoding algorithm [7] is a way of efficiently transmitting a file across a communications link when the receiving host already has a stale version of the same file. This makes a more efficient use of network bandwidth by transferring a minimal description of changes, instead of the entire new instance of the resource.

In the Rsync delta encoding algorithm, the requesting peer acts as the client in the protocol and splits its old version of the file into a series of non-overlapping blocks of size  $s$  as shown in Figure 5. For each of these blocks the client with the old file calculates two checksums: a weak *rolling* 32-bit checksum and a strong 128-bit *MD4* checksum. The client sends these checksums to the host with the new version of the file. The host with the new version operates as the server in the Rsync algorithm. The server scans through its own new

version of the file looking for all matching blocks of size  $s$  with the same weak and strong checksums. The scan starts at different offsets (not just multiple of  $s$ ) so a modification in the middle of the file won't be overlooked. After this step the server sends the client a sequence of instructions, called deltas, for constructing a copy of the new file. Each instruction is either a reference to a block of the old file residing on the client or literal data. Literal data is sent only for those sections of the new file which did not match any of the blocks of the old file. The end result is the peer with the old version of the file constructs a copy of the new file.

Only pieces of the new file that are missing in the client plus a small amount of data for checksums and block indices are sent over the link. This saves link bandwidth by avoiding unnecessary transmissions especially when the file is large. Moreover, the algorithm only requires one roundtrip.

## IV. IMPLEMENTATION

As explained in Section III, the data sharing module of the 7DS platform is composed of two sub modules, BBS and pull-based file synchronization. Pull-based file synchronization module as represented in Figure 6, employs discovery module to periodically announce file information lists as tuples of hash values and modification dates of the files residing in the shared folder. After discovering new data objects by the version controller, again through discovery module, this component exploits rsync delta encoding protocol to retrieve the most up-to-date versions in a bandwidth efficient way. 7DSRsync is the implementation of the rsync [7] delta encoding algorithm for the 7DS platform. 7DSRsync implements the rsync protocol steps required for communication between the client and the server. We used Jarsync [12] which offers primary libraries for the rolling and MD4 checksums of the rsync algorithm. Jarsync [12] is an effort to implement an equivalent of librsync [13] for Java platform. It also provides necessary APIs to create deltas by scanning the new file residing on the server.

But unlike librsync [13], the client and server of the Jarsync library have not been implemented [12].

We have used Apple Bonjour [14] as an implementation of the mDNS [9] protocol. We have tailored a thin wrapper around DNS-SD [14] service advertisement and discovery APIs to hide its complexities from the application developer. Our Java discovery APIs, called BonAHA [15], are simpler but provide sufficient service resolution functionality for disconnected environments. In 7DS mobile applications, file versions and every type of information that should be announced on the network are treated as services. Applications developed on top of the 7DS platform are able to announce their own services or resolve the ones advertised by the others through BonAHA APIs.

The 7DS file synchronization application uses the JNotify library [16] to listen to file system events such as file creation, modification or deletion. JNotify works on both Windows (Windows 2000, XP, Vista) and Linux with INotify support (Kernel 2.6.14 and above). The JNotify Linux API is a thin wrapper around the Linux INotify API. Due to the complexity of the Windows APIs for file system events, most of event handling logic has been put into the C++ code for Windows JNotify. After every insertion, deletion or file modification in the shared folder the new file info list is announced through BonAHA APIs.

## V. DISCUSSION

All 7DS applications, introduced so far, are responsible for data propagation in different mobile scenarios. Among all, file synchronization module is in charge of transferring sometimes large amounts of data. So, scalability of this module of the 7DS platform should be investigated in more details.

The mobile nature of the network makes duration of links connectivity and number of contemporaneous hosts completely unpredictable. In dense and highly mobile networking scenarios, a large number of mobile nodes meet simultaneously for short periods of time. File exchange among these hosts imposes a large amount of traffic to the network for this time interval especially when all nodes have some new files to share. Therefore, the demand on each host increases in proportion to the total number of hosts, quickly overrunning the network's limited capacity.

Suppose there are  $N$  mobile nodes in the network each of which has a new file with size  $M$  MB to share. Distributing a totally new file is expensive in terms of bandwidth consumption. Suppose the links bandwidth is  $B$  Mb/s. After discovery phase, which takes place in multicast, all  $N$  nodes contact each other in unicast mode to download the new file. Therefore, the transmission bandwidth to all these nodes becomes  $\frac{B}{N}$ . As a result, all nodes requesting the file receive it after  $\frac{8MN}{B}$  sec. So, connections between the file owner and other mobile nodes should at least last for  $\tau = \frac{8MN}{B}$  sec. If the link lifetime, before mobile nodes go away, is  $T$ , we should have:  $\frac{8MN}{B} < T$  for a successful file transfer. From this equation we get:  $N < \frac{TB}{8M}$ . Therefore, scalability of the system is bounded by speed of nodes, size of data objects and bandwidth of the links.

## VI. RELATED WORK

Several systems have been proposed to address the file sharing problems. Some of them specifically target always connected networks, while the others that are designed for mobile ad-hoc networks suffer from too much overhead traffic. 7DS has been designed and developed considering limitations of wireless disconnected networks.

Applications such as Gnutella [17] and BitTorrent [18] have been designed to search for data objects in always-connected environments. Some systems like Hayes' [19] have been developed based on Gnutella protocol to share files on a peer-to-peer mobile ad-hoc network. But they suffer from extra overhead that Gnutella's routing and file exchange protocols create for mobile networks. Furthermore, in contrast to Hayes' system [19] that runs only on Bluetooth, 7DS is capable of operating on any type of network.

Microsoft Groove [20] is a commercial shared workspace in which groups are built via invitations. So, unlike 7DS, there is no notion of node and service discovery. All changes are sent to all users or a dedicated server after file modifications. Client-client or client-server communications are based on simple symmetric transmission protocol.

The goal of the OLPC [21] networking project is to connect remote wireless users with no infrastructure to the Internet. Connectivity is achieved using mesh networking technology. OLPC utilizes conventional ad-hoc protocols such as AODV [22] or OLSR [23] for routing. In contrast, 7DS is designed to operate in sparse mobile disconnected networks via store-carry-forward routing.

iClouds [24] and Clique [25] were developed for information sharing in mobile environments. Clique's file sharing mechanism is based on an optimistic replication algorithm. Its inter-node communication is based on broadcasting hash values of fixed size data chunks to all participating nodes. Considering just fixed size file chunks might result in sending the entire file over in case of a minor insertion in the middle of the file. This overhead plus the overhead caused by unnecessary broadcasts creates extra traffic and utilizes a large portion of the of the mobile network's capacity. Unlike 7DS that uses mDNS as its core discovery protocol, iClouds [24] uses a UDP ping/pong mechanism to discover nearby services. Data versions in iCloud are matched by comparing the actual data represented in XML format through string matching. However, in 7DS comparing hash values instead of the real data results in a more efficient version control system.

Proem [26] and Peer2Me [27] are two platforms to develop mobile ad-hoc applications. Proem [26] offers a protocol stack to provide naming, discovery, communication and security in ad-hoc networks. They both use Bluetooth device discovery protocol to search for all nearby Bluetooth devices. After device discovery phase, mobile nodes submit their profiles to announce their presence. Neither Proem nor Peer2Me supports the notion of the automatic service discovery as in the 7DS platform. Peer2Me uses OBEX protocol for communication between peers. Peer2Me is not being developed any further by

its developers. Proem offers an MP3 file sharing application developed using the Proem middleware. However, applications offered by the 7DS platform are more general and cover a broader range of communication requirements in mobile disruption-tolerant networks.

## VII. FUTURE WORK

We are evolving 7DS toward a generic software platform to develop applications for mobile disruption-tolerant networks. A standard platform should provide a set of standard protocols for communication. Syntax and semantics of protocol messages should be defined and standardized. We are planning to specify a set of protocols, possibly standardized, and APIs to facilitate application development.

Most of the routing protocols designed for mobile disruption tolerant networks use some form of controlled flooding to increase delivery probability. Flooding all users with all data objects wastes bandwidth and communication resources and is not energy efficient. Context based routing protocols [28] try to select message relays based on their context similarity to the destination. The context they mostly consider is in the form of mobile nodes' geographical information and history of their contacts. We are working on how to enrich this context by considering mobile nodes' content access behavior in the past. The goal is to infer mobile users' interests from the history of their downloaded objects. This enriched context is exploited to design smarter forwarding algorithms to provide users with closer data objects to their interests.

## VIII. CONCLUSION

7DS provides a platform that serves as an environment to develop mobile disruption-tolerant applications. Modular design of 7DS facilitates future expansion of the platform independently from the underlying core system. We introduced a new class of applications and specified their fundamental requirements in disruption-tolerant networks. The main significance of these applications is in enfolding a broad range of core functionalities. Considering these applications we designed and implemented 7DS primary APIs for service resolution, search engine, delta encoding and file synchronization. We further implemented two new applications for data sharing in disconnected networks using this platform. Data sharing components of the 7DS system offer both push-based and pull-based data sharing models. Setting up 7DS on any device or computer is fairly easy. 7DS-enabled devices can automatically exchange information to overcome the lack of the global connectivity.

## IX. ACKNOWLEDGMENT

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

## REFERENCES

[1] Rfc 4838: Delay-tolerant networking architecture. [Online]. Available: <http://www.ietf.org/rfc/rfc4838.txt>

- [2] M. Papadopoulou 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.
- [3] S. Srinivasan, A. Moghadam, S. G. Hong, and H. Schulzrinne, "7ds - node cooperation and information exchange in mostly disconnected networks," in *IEEE International Conference on Communication, Glasgow, Scotland.*, June 2007.
- [4] K. Fall, "Messaging in difficult environments," in *Intel Research Berkeley, IRB-TR-04-019.*, December 2004.
- [5] A. Vahdat and D. Becker, "Epidemic routing for partially connected ad hoc networks," in *Technical Report CS-200006, Duke University.*, April 2000.
- [6] Cvs - concurrent versions system. [Online]. Available: <http://www.nongnu.org/cvs/>
- [7] Rsync web page. [Online]. Available: <http://samba.anu.edu.au/rsync/>
- [8] D. J. Goodman, J. Borrs, N. B. Mandayam, and R. D. Yates, "Infos-tations: a new system for data and messaging services," in *Proc. IEEE VTC'97, vol. 2*, 1997, pp. 969-973.
- [9] Multicast dns. [Online]. Available: <http://files.multicastdns.org/draft-cheshire-dnsext-multicastdns.txt>
- [10] R. Droms, "Dhcp - dynamic host configuration protocol," in *RFC 2131.*, March 1997.
- [11] P. Mockapetris, "Domain names - implementation and specification," in *RFC 1035.*, November 1987.
- [12] Java rsync implementation. [Online]. Available: <http://jarsync.sourceforge.net/>
- [13] Rsync protocol library. [Online]. Available: <http://librsync.sourceforge.net/>
- [14] Apple computer's bonjour. [Online]. Available: <http://developer.apple.com/networking/bonjour/>
- [15] Bonjour for ad-hoc applications. [Online]. Available: <http://bonaha.cvs.sourceforge.net/bonaha>
- [16] Jnotify java api. [Online]. Available: <http://jnotify.sourceforge.net/>
- [17] Gnutella. [Online]. Available: <http://www.the-gdf.org/>
- [18] Bittorrent protocol. [Online]. Available: <http://wiki.theory.org/BitTorrentSpecification>
- [19] 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.
- [20] Microsoft office groove 2007. [Online]. Available: <http://office.microsoft.com/en-us/groove/>
- [21] One laptop per child. [Online]. Available: <http://wiki.laptop.org/go/Home>
- [22] E. B.-R. C. Perkins and S. Das, "Ad hoc on-demand distance vector (aodv) routing," in *RFC 3561.*, July 2003.
- [23] Rfc 3626: Optimized link state routing protocol (olsr). [Online]. Available: <http://www.ietf.org/rfc/rfc3626.txt>
- [24] 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. [Online]. Available: <http://iclouds.tk.informatik.tu-darmstadt.de/iClouds/pdf/europar2003.pdf>
- [25] B. Richard, D. M. Nioclais, and D. Chalon, "Clique: A transparent, peer-to-peer collaborative file sharing system." [Online]. Available: [citeseer.ist.psu.edu/richard02clique.html](http://citeseer.ist.psu.edu/richard02clique.html)
- [26] 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.
- [27] T. B. Alf Inge Wang and K. Saxlund, "Peer2me - rapid application framework for mobile peer-to-peer applications," in *The 2007 International Symposium on Collaborative Technologies and Systems (CTS 2007)*, Orlando, Florida, USA., May 2007.
- [28] C. Boldrini, M. Conti, and A. Passarella, "Hibop: a history based routing protocol for opportunistic networks," in *In Proc. of The IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM 2007)*, Helsinki, Finland., June 2007.