AdHocFS: Sharing Files in WLANs

Malika Boulkenafed, Valérie Issarny

AdHocFS: Sharing Files in WLANs

Malika Boulkenafed, Valérie Issarny
INRIA-Rocquencourt
Domaine de Voluceau, Rocquencourt, BP 105,
78153 Le Chesnay Cédex, FRANCE
Email: {Malika.Boulkenafed, Valerie.Issarny}@inria.fr

ABSTRACT

This paper presents the ADHOCFS *file system for mobile users, which realizes transparent, adaptive file access according to the users' specific situations (e.g., device in use, network connectivity, etc). The paper concentrates more specifically on the support of* ADHOCFS *for collaborative file sharing within ad hoc groups of trusted nodes that are in the local communication of each other using the underlying ad hoc network, which has not been addressed in the past.*

I. INTRODUCTION

A number of distributed systems deals with mobility (e.g., [1], [2], [3], [4], [5], [6]). In these systems, data copies on a given mobile node are updated locally and are subsequently loosely synchronized with copies on either other mobile nodes (e.g., [3], [4], [7], [8]) or stationary servers (e.g., [2], [1]) through propagation of updates, which is also referred to as *optimistic replication*. Copy synchronization is then handled through the provision of protocols for the management of conflict detection and resolution. Optimistic replication has proven successful for dealing with data access from wireless terminals, while accounting for network disconnection. However, this comes at a high cost in terms of communication and hence energy, due to the underlying scheme of update propagation. This is even more true in the case of collaborative work where updates are propagated to all nodes storing a copy of updated files.

In general, advances in WLANs and in service discovery protocols call for revising the handling of data sharing on mobile nodes. Specifically, nodes that are in the communication range of each other dynamically form a LAN system, which may be seen as a temporary wired LAN system. However, the resulting LAN system has the following intrinsic requirements that must be dealt with: resource saving and in particular energy saving for (unplugged) mobile nodes, adaptation according to the network's dynamics, and security. This paper introduces the ADHOCFS distributed file system that addresses the above requirements. ADHOCFS manages ad hoc groups, i.e., dynamic groups of trusted peers that are in the communication range of each other, so as to further support secure collaborative sharing of files among the group's members. In addition, ad hoc group management is realized so as to minimize resource consumption, and in particular energy consumption on the mobile nodes.

The remainder is organized as follows. Section II provides an overview of ADHOCFS, discussing resolution of file access from any mobile terminal. Sections III to V detail the core functions of ADHOCFS that together serve realizing secure cooperative caching within ad hoc groups through support for security, ad hoc group management and coherency management. Section VI then assesses ADHOCFS, evaluating response time of core functionalities through experiment. Finally, Section VII summarizes our contribution compared to related work.

II. ADHOCFS OVERVIEW

ADHOCFS is organized around a traditional file system hierarchy and the local file systems of the mobile terminals act as caches of the distributed file system. Then, file names are resolved into the various locations from which the files may be retrieved. For a given file, its locations include at least the address of the file's home server (see Fig. 1-[a]) and may further extend to both a local address should the file be cached on the terminal (see Fig. 1-[b]), and to terminals in the local communication range (see Fig. 1-[c]) that cache the file, as identified using the ADHOCFS location service. The location service serves identifying *peer (mobile) terminals* that are accessible by the given mobile terminal and with which files may safely be shared (see Fig. 1-[d]), i.e., terminals belonging to the embedding ad hoc group. Such an identification relies on a traditional service discovery protocol combined with service identification using *security domains*. Each security domain is uniquely identified through the address of the home server storing the file system appertained to the given domain (see Fig. 1-[e]). An example of security domain is the Web server (identified by its URL) hosting the information relating to a given project, which is used by the project's members to share data. Seamless access to files is then realized by making sure that at least the address of the user's file system stored on his home server

Proceedings of the Second IEEE International Symposium on Network Computing and Applications (NCA'03)
0-7695-1938-5/03 $17.00 © 2003 IEEE

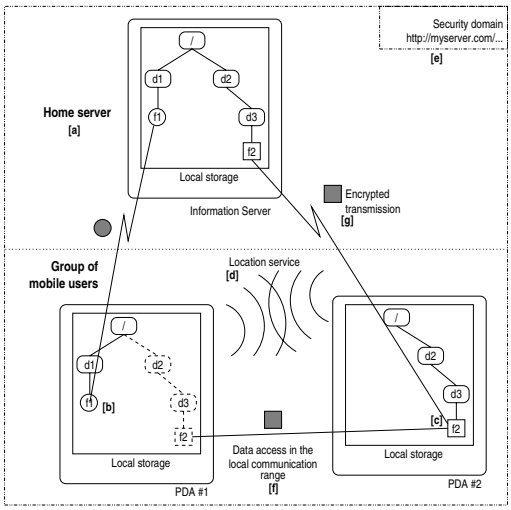

Fig. 1. File access in AdHocFS

and the ones of accessible security domains are available on all his mobile terminals, and through a component for both resolving file names according to the terminal's specifics (i.e., cached files, terminals discovered in the local communication range) and getting the files upon request. The local file system of a mobile terminal is then updated as follows. When access to a file whose name resolves only into the file's home server, is requested, a connection is established using either the wireless LAN or wide-area wireless network, depending on the terminal location. The hierarchy of the local file system gets extended with the path that leads to the requested file (if missing) and the file is copied locally. In addition, upon discovery of peer terminals, the directory hierarchy of the terminals' local file systems are merged so that any file accessible in the local communication range gets identified[1]. Access to such a file from any of the mobile terminals then leads to copy the file locally -if not already cached.

In our context where the files get copied and updated in various locations as users move, it is crucial that each user always sees coherent data (i.e., at least the last version he accessed or, possibly, later versions that got subsequently modified by authorized users). It is further mandatory to integrate adequate cryptographic techniques within the system so that the user's data can only be accessed by authorized users. Such techniques have to be used upon access and transfer of data both in the local (see Fig. 1-[f]) and in the wide area (see Fig. 1-[g]). The next sections detail ADHOCFS support for meeting the above requirements, concentrating on collaborative file sharing within ad hoc groups, which lies in the following three core functionalities: (i) data security, which guarantees that files get accessed and replicated only from/on trusted (mobile) nodes; (ii) ad hoc group management that deals with the creation of ad hoc groups and their dynamics in terms of addition and removal of peer nodes; and (iii) coherency management within groups.

Access to files on the home server from mobile terminals is no longer discussed, as it is managed in a way similar to existing mobile systems. Briefly stated, secure file access from a mobile terminal to the file's home server is implemented via public key cryptography. Coherency management among distinct groups is further based on optimistic replication; file copies on mobile terminals are then reconciled with copies at the server upon synchronization with the server, using the same reconciliation scheme as the one discussed in Section V. Finally, it is considered that any mobile terminal regularly synchronizes with the files' home servers to which it is granted access to, using either infrastructure-based wireless networking or ad hoc networking, depending on the specific location of the terminal and servers.

III. SECURITY MANAGEMENT

Security is of crucial importance in our context since we are using both wireless LAN and global wireless networks. It is therefore mandatory to ensure end-to-end privacy and integrity of the user's data. However, as our platform aims at running on resource constrained terminals (e.g., wireless PDA), it is necessary to balance strong security enforcement with resource consumption, and in particular energy. ADHOCFS uses both asymmetric and symmetric cryptography. The former is used for securing communication links between any mobile terminal and a file server, and is not further detailed due to the paper focus on file sharing among mobile terminals. The latter is used for securing communication links between any two peer mobile terminals within an ad hoc group. Symmetric cryptography enables ensuring privacy of data between two terminals using a well-known algorithm and a shared secret, i.e., a secret key. Moreover, by using symmetric cryptography in conjunction with a cryptographically secure hash function, also called one way function, it is possible to ensure data integrity [9].

In ADHOCFS, files of related interests are grouped into a *security domain*. A secret *domain key* is then associated with each domain, and it is shared by the domain's server and all the terminals whose users are granted access to this domain. More specifically, the secret key is regularly computed at the domain's server and securely propagated to trusted mobile terminals when they synchronize with the server. The domain key is used for both authentication and secure message exchanges among peer mobile terminals.

The use of the domain key enables minimizing the computation cost associated with cryptography. However,

[1]Note that the file data is not merged.

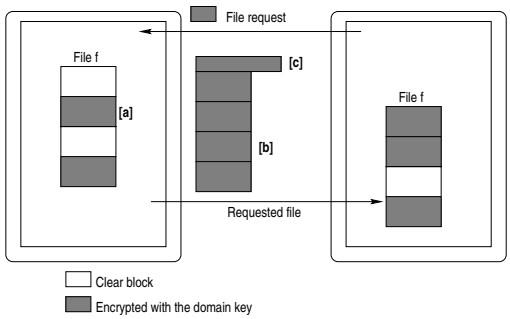

Fig. 2. File encryption

the security enforced by ADHOCFS is dependent upon the avoidance of the domain key forgery, which requires regular revocation of the key. Protection against forgery of the domain key on the terminal itself is enforced by storing the key encrypted on the terminal, and by decrypting it using the user's password (e.g., using a PAM module in Linux)[2]. An alternative solution to the setting of the group key would have been to integrate a protocol for key agreement within dynamic collaborative groups (e.g., [10]). However, such protocols are costly, both in terms of computation and communication costs. This is why we have preferred to undertake a simpler solution in a first step. Finally, access control to files relies on the access control of the underlying local file system, given that file owners do not change.

Regarding the encryption of files transferred from one mobile node to another, the file blocks are encrypted and decrypted independently of the others (see Fig. 2-[a]). However, to protect against attacks that would replace blocks within messages, the header of each such message includes a secure checksum (using MD5) of the transferred file blocks (see Fig. 2-[c]). In addition, we use a nonce to ensure that any message received in reply to a request is indeed the reply. Using per-block encryption allows the decryption of blocks upon actual access, instead of decryption at the time the blocks are received. In the same way, a block that is decrypted will be encrypted only if a request for the block (e.g., request for a file copy from a peer mobile terminal) is received. In this way, the computation cost associated with cryptography is minimized, and hence energy consumption due to security management is reduced. Note that reduced energy consumption based on on-demand decryption is obtained due to the fact that transferred files are encrypted using the domain key (see Fig. 2-[b]). If the files were encrypted using a temporary secret key that would be set up upon the establishment of a connection between any two peer mobile terminals, then upon every subsequent transmission of a file, the file would have to be decrypted (using the secret key shared with the terminal from which the file was obtained) -if not already done- and then encrypted (using the secret key shared with the requesting terminal).

[2]Note that this could alternatively be realized using a smartcard, for mobile terminals equipped with a smartcard device driver.

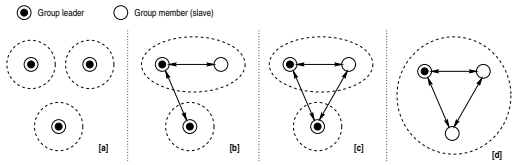

Fig. 3. Merging groups

IV. AD HOC GROUP MANAGEMENT

Ad hoc group management in ADHOCFS builds upon an existing service discovery protocol, i.e., the IETF Service Location Protocol (SLP) [11]. We use the configuration of SLP that does not rely on a directory agent for service discovery. In ADHOCFS, SLP serves locating peer mobile nodes, i.e., nodes in the communication range of the WLAN that have access to common security domains. Every mobile node acts as a *Service Agent* that handles lookup queries for peer nodes of ADHOCFS. Upon initialization, each terminal registers itself for each security domain of ADHOCFS it has access to, using the following format (i.e., SLP's Universal Resource Locator -URL- format): service:AdHocFS:///*IP_address*:*port_number*/*domain*[3].

Periodically, every mobile terminal looks for peer terminals by issuing, as a *User Agent*, the following service request: service:AdHocFS. Every mobile terminal in the local range sends back as many messages as security domains it belongs to; each message carries the URL associated with the given security domain, which embeds the terminal's IP address, port number, domain and unique identifier (UID). Using received URLs, terminals are able to join the ad hoc groups they belong to.

Consider first that all peer nodes are isolated (see Fig. 3-[a]), which is known through the *group* variable that has the value *nil* on every peer. Then, the embedding ad hoc group is created by one of the peers, called *leader*, that is the node that has the smallest UID. The leader concurrently interacts with all its peers as follows: it first establishes a secure connection with the peer, it then requests the peer for its local directory hierarchy that it will merge with its own. The request additionally embeds the list of group members. The leader finally broadcasts the directory hierarchy of the ad hoc group to all the peers and cooperative caching within the group may proceed. In addition, all the peers but the leaders concurrently establish secure connections with all their peers that have a larger UID than theirs. On

[3]Mobile terminals that do not want to cooperate (e.g., for the sake of energy saving) de-register for the security domains they belong to.

every peer, the value of the *group* variable is now equal to the list of peers, and the *leader* variable is equal to the UID of the leader. Note that for the case where a node is isolated and hence the single member of a group, the value of the *leader* variable is the UID of the node.

Dynamics of the group is handled by periodically looking for peers. For a given period, the leader becomes the peer with the next higher UID than the value of *leader*. The leader then checks the value of *group* with the list of nodes returned by SLP. If these two values differ, the ad hoc group must integrate the (potential) new comers (which may themselves be part of a distinct ad hoc group), and discard the (potential) nodes that left. Addition of members is handled by merging groups as follows, considering that an isolated node forms a singleton group (see Fig. 3-[b]). In the reply to SLP requests for the discovery of peer nodes, the node embeds its local value of *group* and *leader*. Then, the leader of the new group becomes the node that has the smallest UID among all the leaders if all the merged groups have more than one member. If some groups are singleton, the leader is taken from the largest groups. The new leader then performs a merge protocol similar to the above one with all the leaders of the merged groups. All the leaders of merged groups then forward the updated directory hierarchy[4] to the peers that were led by them, together with the new value of *group* and *leader*, leading nodes to concurrently establish missing connections with peer nodes (see Fig. 3-[c]) and to form the new group (see Fig. 3-[d]). Leaving nodes are straightforward to handle; the leader broadcasts the new value of *group* to peer nodes. Note that the case of leader removal is handled by changing the leader on each period. Hence, in the worst case, a newly formed group stabilizes in at most two periods after the change occurrence.

The main cost of ad hoc group management lies in message exchanges for discovering peers and then merging groups. The overall cost is further proportional to the period that is set for managing the group's dynamics. The period is initially set to a given value T and is then dynamically adapted according to the past behavior of the embedding group (which may be the node itself in the case of singleton group), using *statistical modal class* [12].

Peers interact through the WLAN that may be in either the infrastructure-base mode if a base station is nearby or the ad hoc mode in the absence of base station. ADHOCFS does not impose any specific networking mode and it is currently up to the users to set the preferred mode of operation, assuming that nodes in the local communication range of each other select the same networking mode. Note that ongoing work in the area of mobile networking will further allow for the transparent setting of the most adequate networking mode according to the environment [13]. In the specific case of ad hoc networking, it should be accounted for the fact that groups may only be partially connected if the network protocol does not support ad hoc routing among nodes. In this case, two groups merge only if all their members can communicate, which is identified by comparing the lists of peer nodes obtained through the slptool function of SLP.

[4]This update is the difference with the previous value of the hierarchy.

V. COHERENCY MANAGEMENT

Existing network file systems aimed at mobile nodes implement optimistic replication so as to enable file access in disconnected mode. However, in the case of file sharing within an ad hoc group, file copies cached on peers may be strongly synchronized so as to prevent the occurrence of conflicting updates while in the group (apart from those created due to concurrent file updates before joining the group) and hence better support collaborative work groups. This further leads to minimize the communication cost (and hence energy cost) of update propagation since the complexity of update propagation among peer nodes under optimistic replication is in O(N) for every concurrent update, with N being the number of peers, while it can be significantly reduced under conservative replication as detailed hereafter. In other words, base techniques that have been proven successful in wired LANs are of direct relevance for coherency management within ad hoc groups. Hence, ADHOCFS implements optimistic replication over nodes that belong to distinct groups, and conservative replication over peer nodes[5]. The following details our conservative coherency protocol within ad hoc groups.

Our coherency management protocol is based on a *exclusive writer* protocol within an ad hoc group. Using read/write locking within a group, all write operations are exclusive within the group while read operations are shared. However, local files can be manipulated (read/write) independently within disjoint groups, provided that data are synchronized when integrating a group[6]. A peer can be in five modes regarding a given data file: Fresh, the peer holds a fresh data copy after synchronizing with the reference copy; Read, the peer can read the data; UpdateRead, the peer is allowed to read the data but has to update its copy first; ReadWrite, the peer can read and write the data; and Invalid, the peer has no access to the data. With this protocol, either a peer modifies data in the ReadWrite state and all the other peers are in the Invalid

[5]Note that the granularity of coherency management is left upon the users who choose the most effective decomposition of their shared data into files. Taking for instance the case of document editing, the document may be structured into a number of files according to the degree of concurrency updates that is targeted.

[6]Note that a user who is being involved in a group can still choose to read his local file copy, which may possibly not be up to date, while waiting for acquiring a lock for accessing the shared file.

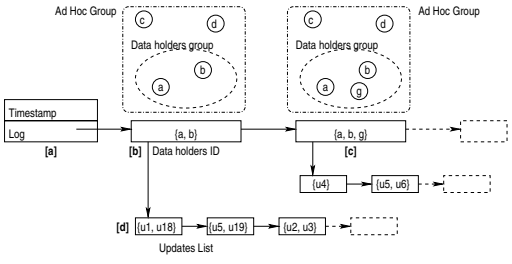

Fig. 4. Coherency Control List

state; or all the peers are in the Read or UpdateRead state. This protocol ensures that all reads access the most recent file version within the group. This guarantees data coherency within a group, given that data are reconciliated upon their first access.

Within a group, concurrent and diverging file copies are detected using the *CCL* (Coherency Control List), which is attached to each file. The CCL logs in a data structure how and when the given file copy was modified since it was copied from its home server. An example of CCL is given in Figure 4. The CCL comprises the timestamp value of the reference copy, i.e., the copy stored on the file's home server[7], at the time it was copied/synchronized with, and the log of subsequent updates on the local copy (see Fig. 4-[a]). The log is the list of all successive data holders, as seen by this specific data file copy (see Fig. 4-[b]). Data holders give the set of peers belonging to the same ad hoc group and locally caching the data. Every time the group members composition is modified, through addition or deletion of a peer, and upon update, a new item is added to the list, which contains the new set of group members storing the data (see Fig. 4-[c]). This enables tracking the dynamics of the ad hoc group due to peers mobility, and to distinguish updates within different ad hoc groups in order to enhance conflict detection. If a disconnected peer updates its local copy of the data, a new item is added to its local list, which contains only its UID. For every group composition, the list of modified file blocks is maintained (see Fig. 4-[d]).

Given local CCLs, a mobile node is able to determine whether its local file copies are coherent and/or can be reconciled with the file copies stored on peer nodes of the embedding ad hoc group[8]. When a node n joins the ad hoc group, its directory hierarchy and the group's one are merged (see § IV). The merging is further realized so that the *a priori* latest version (i.e., greatest timestamp and largest update list) of a given file copy is assigned the ReadWrite mode while others are assigned the Invalid mode if none of the file copies are being accessed. The same scheme applies if either the only copy that is being accessed is the latest version or all the copies being accessed may be reconciled. On the other hand, if diverging copies are being accessed, exceptions are signaled to all the applications that are accessing the file, leading to application-specific conflict resolution. In general, at the time of group merging, diverging copies are detected and are made known to users by associating a tag to local file copies that have diverged, which appears upon browsing.

File updates within the group are then lazily propagated from the latest writer as follows. If the file is requested for write access by another node, then the node acquires the ReadWrite lock and gets the list of updates. If the file is requested for read access the node acquires the Read lock and gets the list of updates. However, all the other nodes caching a copy of the file acquire only the UpdateRead lock. Note that the sent list of updates may not be sufficient to reconcile -if possible- the local copy since copies are not synchronized as nodes enter an ad hoc group. Hence, upon actual access, missing updates are possibly requested to the latest writer for reconciliation. In the case where the latest writer of a given file leaves the group, the protocol guarantees that the copy that will be accessed is the latest version present in the group. Lazy update propagation follows from our concern of minimizing energy consumption and hence computation and communication.

VI. EXPERIMENT

A first prototype of ADHOCFS has been implemented in Objective Caml 3[9]. The ADHOCFS prototype builds upon the Extended 2 FS (Ext2) local file system, the Blowfish symmetric encryption algorithm [14] using 128 bits keys, and the OpenSLP[10] implementation of SLP. Mobile terminals of the platform are laptops with a 500 MHz Pentium III CPU, 256 KB of cache, 200 MB of RAM and a 10 GB hard disk running under Linux operating system. The wireless LAN is IEEE 802.11b in the ad hoc mode (Lucent 11Mb WaveLAN PC Card).

Figure 5 gives the time taken for creating a group, leaving and joining a group, which are linear with the group size. The main cost of group creation lies in detecting trusted peers and sending the directory hierarchies by group leader so as to compute the directory hierarchy of the overall group, which is mandatory to allow peers belonging to the group to have a global view of shared data and to enable collaborative sharing.

Table I gives the time taken to locally access a coherent data file in the write mode when the peer is in the Read state. This time is constant for a given group size: it

[7]If a file is created locally on a peer, the corresponding timestamp has a special value (*NIL*) to indicate that this file has no reference copy yet.

[8]Basically, two copies can be reconciled if the CCL associated to one is a sub-list of the CCL associated with the other.

[9]`caml.inria.fr`

[10]`openslp.org`

TABLE I

FROM READ TO READWRITE STATE

Group size (*peers*)	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
Time (*sec*)	0.00949	0.01281	0.01982	0.02129	0.02645	0.02977	0.03493	0.03825	0.04341
Energy (*mW.sec*)	1.58	2.91	5.4	6.67	9.08	11.87	15	18. 51	22.38

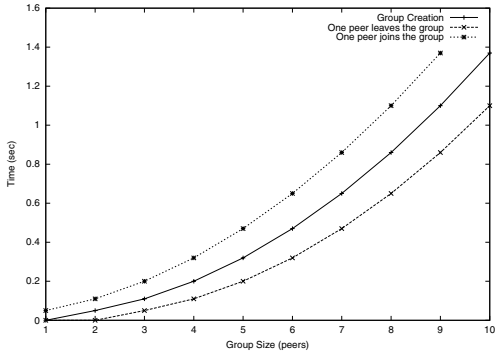

Fig. 5. Time taken for group creation

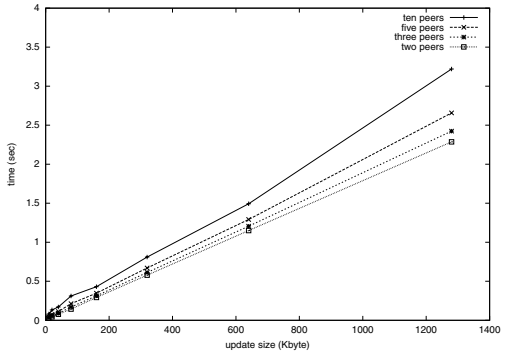

Fig. 7. From Invalid to ReadWrite state after successive writes

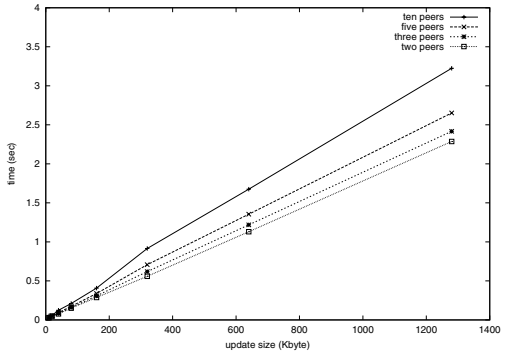

Fig. 6. From UpdateRead to ReadWrite state.

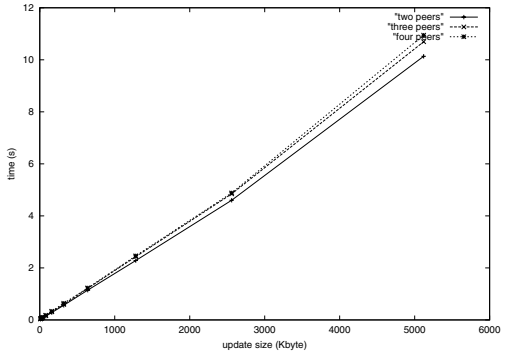

Fig. 8. Write time when not locally caching the file

corresponds to the cost of getting the token and notifying the peers to set their state to Invalid. Thus, messages sent to realize the coherent access are only control messages, and do not contain any updates. Note that the cost of additional peers in terms of response time is only of about 0.005 sec. Table I further shows the associated cost in terms of energy consumption, based on the formulas given in [15]. Note that energy consumption is highly dependent upon the size of the group since non destination peers consume energy when messages are being sent under the ad hoc mode.

Figure 6 gives the time taken to access a coherent data file when the peer is in the UpdateRead state, which corresponds to the cost of update propagation. Using our coherency protocol, getting locally access to a coherent data depends on the update size, while group size affects slightly the performance of our protocol. However, the energy consumption depends on both the group size and the update size. It is given by: $\varepsilon_{msg} = (0.1 + 0.25 * n) * size + 221 + 163 * n$ (based on equations from [15]), where n is the ad hoc group size and $size$ the update size.

Figure 7 gives the time taken to access a coherent data file when the peer a is in the Invalid state, after successive write operations performed by different peers. This time corresponds to the cost of getting the token and the updates, which are lazily propagated. More precisely, the peer a requesting for file access knows only the identity of the first writer b (i.e., the peer that set a to the Invalid state), and thus sends its request to b. However, b is not the actual token owner (last writer); it thus forwards a's request to the peer that it considers as being the token owner (i.e., the peer that set b to the Invalid state). The process is then repeated until the request reaches the actual token owner (the last writer), which replies with the updates and the appropriate meta-data. Compared to the results shown

in Figure 6, the cost of propagating the request among successive writers amounts to the cost of propagating an update from one peer to another.

Figure 8 gives the cost of getting a copy of a data file for writing when the peer does not cache it locally, while the peers caching the file within the ad hoc group, are in the Read/UpdateRead state. The resulting cost is linearly dependent on the file size. Similarly, the cost of opening a data file in the writing/reading mode increases linearly with update size. Hence, using our coherency protocol, getting locally access to a coherent file depends on the updates size, while group size affects only very slightly the performance of our protocol in terms of response time.

The above results show that the overhead introduced by the core functions of ADHOCFS is kept to a minimum, since response times offered by ADHOCFS remain comparable to the exchange of a data file between two nodes through the WLAN. To further show the benefits of lazy update propagation within ad hoc groups in terms of response time and energy consumption, we compare it to a version of our coherency protocol where updates are propagated whenever they occurs as realized by optimistic replication protocols supporting collaborative, peer-to-peer file sharing. We consider an ad hoc group of four peers, all caching the data, and being in the Read state. We further consider four successive write operations by different peers, each leading to an update size of 640 Kbytes. Under our protocol, the times taken to write access a coherent data file from the first write to the fourth one are respectively equal to: 0.019 sec, 1.20 sec, 1.22 sec, and 1.27 sec (see Figure 7). On the other hand, when the updates are propagated whenever they occur, the time taken to write access a coherent data file is the same for all four write operations in the above scenario, and is equal to 3.639 sec, which leads to 7218 mW of extra energy consumption. Thus, lazy update propagation allows for better response time but also for a lower communication cost, and hence lower energy consumption.

VII. CONCLUSION

In ADHOCFS, the file systems of mobile terminals act as local caches, and mobile terminals that have access to common files and are able to communicate through the wireless LAN, cooperate to form an ad hoc distributed file system. Core components of ADHOCFS lies in:

- A naming service that resolves file names into the various locations from which files may be retrieved (i.e., at least the address of the file's home server, local copy if cached, peer terminals in the communication range that stores a file copy).
- A location service, which enables setting up ad hoc groups of trusted mobile terminals that are connected using the wireless LAN. In addition, secure links are established between peer nodes so as to guarantee data integrity and privacy, while minimizing the computation cost and hence energy consumption associated with cryptography.
- A coherency management service that reconciles copies cached on mobile terminals that belong to the same group and enables collaborative file sharing among peer nodes.

We have presented an evaluation of ADHOCFS in terms of response times offered by the current ADHOCFS prototype. Results show that the response time of ADHOCFS operations remain comparable to the time taken for exchanging a file over the WLAN.

There has been a large amount of work on supporting access to shared files on mobile terminals. Early work in the area has been concentrating on enabling access to local files, independent of network connectivity. Hence, most proposals have been oriented towards ensuring availability of a local copy so as to cope with temporary disconnection. Relevant work in the area includes solutions to file prefetching (e.g., [16]) and to optimistic replication (e.g., [2], [3], [1], [17], [4], [7], [8], [6]). ADHOCFS does not currently include any support for managing the local cache according to future accesses, and in particular support for prefetching. This is part of our future work, which will benefit from existing solutions such as the one of [16]. On the other hand, ADHOCFS resembles and benefits from past work on optimistic replication since we have adopted a log-based solution for update propagation, which has been proven successful in this area. ADHOCFS further uses a conventional log-based optimistic coherency management for reconciliating copies that have been accessed within distinct ad hoc groups. It is further part of our future work to enhance support for automatic reconciliation based on latest research results (e.g., [18]). However, ADHOCFS differs from the aforementioned references by accounting for the specifics of today's WLANs that allows creating dynamic networks of mobile nodes when they are in the local communication range of each other. Such dynamic networks may then be exploited for supporting collaborative group works. ADHOCFS offers such a capability through the management of dynamic, ad hoc groups of trusted mobile terminals, among which files may safely be shared. This has further led us to use a conservative replication scheme within ad hoc groups as it is more suited to collaborative applications, and further allows reducing the communication and energy cost associated with coherency management.

An ad hoc group of ADHOCFS realizes cooperative sharing of nomadic data, which has been an active area of research over the last few years given the increasing interest for supporting pervasive computing. Work in the area that is the closest to our concern relates to enabling

data caching in various locations, and subsequently retrieving them in another location. However, existing proposals concentrate on leaving data on untrusted, stationary local storage servers (e.g., [5], [19]), and are thus complementary to ours since we are addressing collaborative caching among trusted mobile nodes.

Among enhancements of ADHOCFS that we are working on, we are going to further elaborate synchronization with the files' home servers so as to both minimize the occurrence of diverging copies, and regularly revoke the domain key used within groups to securely share files. We are further interested in improving the quality of service of ADHOCFS, regarding in particular availability, examining more specifically replication of cached files within a group to ensure availability of the files' latest version despite the groups' dynamics. Exploitation of ad hoc networking raises the issue of further exploiting ad hoc routing protocols to allow accessing files stored on a terminal that is reachable in a number of hops. We have not integrated such a facility in ADHOCFS due to our focus on collaborative work where we consider that the communication range of WLANs like IEEE 802.11 allows for sufficient coverage of collaborative work groups. However, ad hoc routing protocols may conveniently be exploited to access a file that is not available in a group in the absence of a base station, which prevents accessing the file's home server. A base solution to this issue is presented in [20] for the specific case of Web data, which can be easily adapted to the context of ADHOCFS. Finally, we are going to experiment the use of ADHOCFS with various types of mobile terminals (e.g., iPAQ), which is quite direct given the availability of Linux for embedded platforms.

ACKNOWLEDGMENT

The authors would like to acknowledge the contribution of David Mentré, Anis Ben-Arbia, and Animesh Pathak to the design and the implementation of the ad hoc group management protocol and the coherency management protocol. This Project has been partially funded by ITEA VIVIAN Project (ITEA 99040).

REFERENCES

[1] P. Kumar and M. Satyanarayanan, "Log-based directory resolution in the coda file system," in *Proc. of the 2nd International Conference on Parallel and Distributed Information Systems*, 1993, pp. 202–213.

[2] M. Satyanarayanan, J. J. Kistler, P. Kumar, M. E. Okasaki, E. H. Siegel, and D. C. Steere, "Coda: A highly available file system for a distributed workstation environment," *IEEE Transactions on Computers*, vol. 39, no. 4, pp. 447–459, 1990. [Online]. Available: citeseer.nj.nec.com/satyanarayanan90coda.html

[3] R. G. Guy, "Ficus: A very large scale reliable distributed file system," Los Angeles, CA (USA), Tech. Rep. CSD-910018, 1991. [Online]. Available: citeseer.nj.nec.com/guy91ficu.html

[4] K. Petersen, M. Spreitzer, D. Terry, M. Theimer, and A. Demers, "Flexible update propagation for weakly consistent replication," in *Proceedings of the 16th Symposium on Operating Systems Principles*, 1997.

[5] J. Kubiatowicz, D. Bindel, Y. Chen, P. Eaton, D. Geels, R. Gummadi, S. Rhea, H. Weatherspoon, W. Weimer, C. Wells, and B. Zhao, "Oceanstore: An extremely wide-area storage system," in *Proceedings of the 9th International Conference on Architectural Support for Programming Languages and Operating Systems*, Boston, MA, November 2000. [Online]. Available: citeseer.nj.nec.com/bindel00oceanstore.html

[6] C. Mascolo, L. Capra, S. Zachariadis, and W. Emmerich, "XMIDDLE: a data-sharing middleware for mobile computing," *Wireless Personal Communications*, vol. 21, 2001.

[7] H. Yu and A. Vahdat, "Design and evaluation of a continuous consistency model for replicated services," in *Proc. of the 4rd Symposium on Operating Systems Design and Implementation*, 2000. [Online]. Available: citeseer.nj.nec.com/yu00design.html

[8] T. Ekenstam, C. Matheny, P. Reiher, and G. Popek, "The bengal database replication system," *Distributed and Parallel Databases*, vol. 9, no. 3, 2001. [Online]. Available: citeseer.nj.nec.com/449358.html

[9] A. J. Menezes, P. C. van Oorschot, and S. A. Vanstone, *Handbook of Applied Cryptography*, 4th ed. CRC Press, 1996, iSBN: 0-8493-8523-7. [Online]. Available: http://www.cacr.math.uwaterloo.ca/hac/

[10] Y. Kim, A. Perrig, and G. Tsudik, "Simple and fault-tolerant key agreement for dynamic collaborative groups," in *ACM CCS-7*, 2000, pp. 235–244.

[11] E. Guttman, C. Perkins, J. Veizades, and M. Day, "Service location protocol, version 2," June 1999, rFC 2608.

[12] M. H. Degroot, *Probability and Statistics, 2nd Edition*. Addison-Wesley, 1986.

[13] K. Al Agha and G. Pujolle and D. Zeghlache, "VCB: an efficient resource sharing scheme for cellular mobile systems," *Telecommunication Systems, Kluwer*, vol. 19(1), pp. 101 – 110, January 2002.

[14] B. Schneier, "Description of a new variable-length key, 64-bit block cipher (blowfish)," in *Fast Software Encryption, Cambridge Security Workshop Proceedings (December 1993)*. Springer-Verlag, 1994, pp. 191–204.

[15] L. Feeney and M. Nilsson, "Investigating the energy consumption of a wireless network interface in an ad hoc networking environment," *In proc. of the IEEE Infocom*, vol. 5, no. 8, 2001. [Online]. Available: www.sics.se/~feeney

[16] G. Kuenning and G. Popek, "Automated hoarding for mobile computers," in *Proceedings of the 16th ACM Symposium on Operating Systems Principles*, 1997.

[17] A. Joseph, A. De Lespinasse, J. Tauber, D. Gifford, and M. Kaashoek, "Rover: A toolkit for mobile information access," in *Proceedings of the 15th ACM Symposium on Operating Systems Principles*, 1995.

[18] A.-M. Kermarrec, A. Rowstron, M. Shapiro, and P. Druschel, "The IceCube approach to the reconciliation of divergent replicas," *Proc. of the 20th ACM Symposium on Principles of Distributed Computing (PODC 2001)*, August 2001.

[19] P. Castro, B. Greenstein, R. Muntz, C. Bisdikian, P. Kermani, and M. Papadopouli, "Locating application data across service discovery domains," in *Proceedings of ACM SIGMOBILE'01*, 2001.

[20] F. Sailhan and V. Issarny, "Energy-aware web caching for mobile terminals," in *Proceedings of the CDCS Workshop on Web Caching Systems*, 2002, http://www-rocq.inria.fr/arles/doc/doc.html.