Partitionable Mobile File System over Ad-Hoc Networks

Weider D. Yu *Yan Chen*

San Jose State University, San Jose (Silicon Valley), California, USA

Summary
Most peer-to-peer file storage sharing systems work on wired networks. Due to the multi-hop wireless communication nature in ad-hoc networks, the development of mobile ad-hoc network file storage sharing system faces many challenges. In this paper, a hypothetical partitionable mobile file system (PMFS) is proposed. This new PMFS is a distributed file storing/sharing system built on partitionable mobile wireless ad-hoc networks. The main proposal of PMFS is focused on the system performance and service availability over wireless ad-hoc networks. To enhance performance and achieve efficient file storage and retrieval, various techniques have been proposed on the system. Overall, PMFS is aimed to be a more flexible and efficient file storage system over partitionable mobile wireless ad-hoc networks.

Key words
Wireless ad-hoc networks, mobile networks, peer-to-peer file storage, mobile file systems, file storing/retrieval, file sharing, mobile network partition.

1. Introduction

The ability to communicate with the rest of the world instantaneously has been the ultimate goal for the design of network communication system. For such a large coverage, it seems only realistic and achievable through wireless networks. This becomes the driving force of all the wireless network research done all over the world. Moreover, due to the popular growing demand of file sharing through Internet, such as Napster [1], Morpheus [2], Gnutella [3] and Freenet [4], this idea of file storage and sharing through networks has been extensively studied recently in peer-to-peer system. However, to the best of our knowledge, no such file sharing system has been studied in mobile wireless ad-hoc networks. The combination of file storage management built on mobile wireless ad-hoc networks increases the difficulty of such design.

Unlike the conventional infrastructure-based wireless network, ad-hoc network, as a distributed wireless network, is a set of mobile wireless terminals communicating with each other without any pre-existing fixed infrastructure. The mobile wireless ad-hoc network has several unique features that challenge the network operation, such as the routing algorithm, Quality of Service (QoS), resource utilization, etc. Figure 1 depicts a small-scaled model of a wireless ad-hoc network. All terminals, also referred to as mobile nodes, exchange information among one another in a fully distributed manner through wireless connections within the ad-hoc network. Due to the mobility of

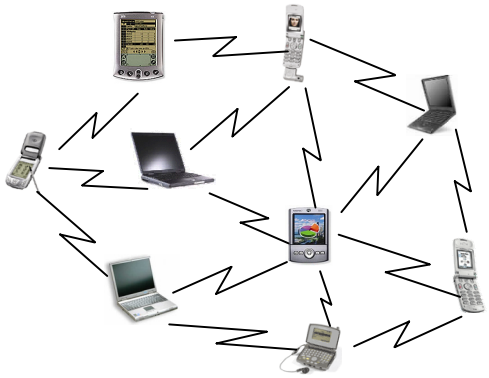


Fig. 1 A Small-scaled Model of a Wireless Model of a Wireless Ad-hoc Network

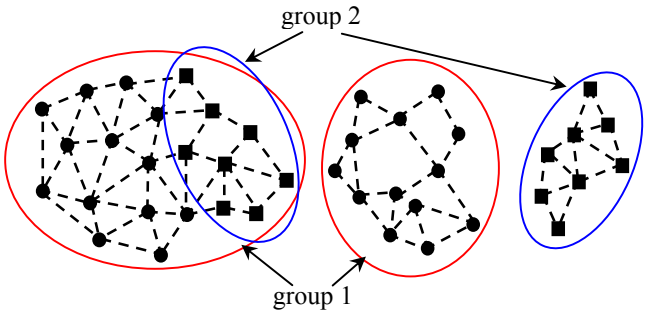
these nodes, the network topology is under constant changes without any centralized control in the system. These are several main concerns that need to be considered when designing a specific application-layer protocol based on wireless ad-hoc networks.

In a wireless ad-hoc network, all the nodes are interconnected by single-hop or multi-hop wireless connections. There is no centralized administration or base station to coordinate the behavior of each node in the network. As a result, each node must be self-configurable in order to adapt to various network topologies. And each node may function as a packet-level router for other nodes in the same network [5]. They can assist transmitting packets from a source to a destination through wireless connections in a fully peer-to-peer fashion. At the same time, because of the wireless connections, the service coverage and bandwidth availability become critical issues in the wireless channels. The limited transmission coverage restricts the design of such network, because wireless signals die out exponentially as the transmission distance stretches. On the other hand, how to save the transmission bandwidth and maximize its usage is also a major concern in ad-hoc network, because it directly results the achievement of the effective and efficient information access in such network. Both of these challenges have attracted attentions from various research groups in ad-hoc network laboratories around the world.

A main feature of ad-hoc networks is that all the nodes within the network have the freedom to move around, which causes the network topology to change dynamically and unpredictably. This means any node may join or leave the network at any time. Due to the lack of centralized control in the network, nodes rely

only on the neighbors in a self-organizing manner. Thus, it is not an easy task to maintain the network connectivity and service availability while the nodes behaviors are unknown. However, quite a few researchers have proposed many prediction schemes [6,7,8,9] to predict and model the nodes mobility, and thus, to improve the network connectivity and routing algorithms. Furthermore, in ad-hoc networks, this topological dynamics can be described as frequent network partitioning, due to the natural grouping behavior in mobile user's movement. With some further study, several grouping methods have been proposed, such as velocity grouping [6] and physical location grouping [10]. These grouping models have been introduced to provide a better prediction on the behavior of ad-hoc networks. And at the same time, these models also provide an easier solution to manage the mobile nodes in different groups rather than single individual nodes. Figure 2 describes the ad-hoc network before and after partitioning visually. As shown, group 2 is separated from group 1, and forms another isolated ad-hoc network that is out of reach of the original group 1. When network partition happens, all the services in the previous network need to be maintained for all the nodes in both partitions, so that the nodes should not be aware of the network partitioning at all. Therefore, in Figure 2, all the nodes in both group 1 and group 2 should have the same services available in both before (Figure 2a) and after (Figure 2b) partitions.

Fig. 2 An Ad-hoc Network Model before and after



(a) before partition (b) after partition

The focus of this paper is to improve the current file storage sharing system over mobile wireless ad-hoc networks. An advanced hypothetical system called Partitionable Mobile File System (PMFS) is proposed here. As previously mentioned, there are a number of existing peer-to-peer storage systems that have been developed in the past, such as CFS [11], PAST [12], Freenet [4], etc. Together, they address a wide range of interesting technical aspects regarding peer-to-peer networks, such as the decentralized control with distributed algorithm, the efficiency of resource management, the delay in file storing and retrieval, the robustness of the system against network failures, the load balance among all the machines, and the scalability of the system. All of these characteristics mentioned above in peer-to-peer networks would also be our concerns for designing the PMFS based on wireless ad-hoc networks. In other words, the PMFS is targeted to adapt the existing peer-to-peer file storage system into ad-hoc networks. Besides the smooth adaptation, another main task in this paper is also to maximize the performance of this storage system. Because of the mobile and partitioning nature of the ad-hoc networks mentioned earlier, the task is not as trivial as it may seem to be. Overall, PMFS needs to adapt the existing peer-to-peer storage utility into the wireless ad-hoc networks. The system stores and retrieves files in a group-oriented manner with individual file blocks. It modifies and extends an existing simple lookup algorithm to allocate mobile nodes. And it also maintains the storage service for lookup within mobile nodes, even after network partitions.

The remainder of this paper is organized as follows. Section II discusses related technologies, existing issues and basic background concepts and definitions. Section III gives a detail overview of PMFS system design, which explains each newly proposed technique in more depth. Section IV presents the results and analysis of PMFS. Finally, section V concludes this paper.

2. Background Concepts and Definitions

In the mobile wireless ad-hoc networks, the network topology dynamically changes due to the node mobility, unpredictable partitioning and decentralized system. This network characteristic directly increases the design complexity and difficulty of any storage system over such mobile wireless networks. For such storage system, it needs to work with the decentralized system, and meanwhile, it also needs to provide the storage reliability and persistence. Here, a few important questions arise: How should this system manage its nodes without centralized control, but yet, it can recover from network partition? How should this system keep track of all the file blocks among all the mobile nodes? How could this system possibly save all the blocks reliably so that the files will not be lost after network partitions? In order to have a complete and functioning Partitionable Mobile File System, all the above questions are the keys to the success of designing such file system. Therefore, they will all be answered throughout this paper.

2.1 Velocity Grouping

In a partitionable mobile file storage system, all the mobile nodes need to be managed in the way such that all the file blocks are stored reliably even with the network partitions. The approach we use in PMFS is to group all the nodes according to their similar velocities, called the Reference Velocity Group Mobility (RVGM) model [6]. For simplicity and clarity, Table 1 lists a few important notations used throughout the paper.

Table 1: Notations used for Velocity Grouping

Notation	Definition
m	maximum number of nodes allowed on one velocity group
M	total number of dummy servers in one velocity group
v_i	velocity of each node
v_m	mean velocity of a velocity group
(v_x, v_y)	Cartesian representation of node velocity in x and y axes

The basic idea behind RVGM model is the observation of how *"mobile users exhibit correlated mobility patterns in their movements"*. In other words, frequent network partitions are not totally untraceable in mobile wireless ad-hoc networks. All the mobiles nodes tend to travel in some kind of group-based movements. Thus, PMFS proposes the use of such velocity grouping, RVGM.

2.2 Lookup Algorithm

The lookup algorithm in any file storage system is the key to the system performance. There are many researches done on the most efficient lookup algorithm. But to work with mobile wireless ad-hoc network, the lookup algorithm in PMFS is chosen to adapt an efficient and scalable lookup algorithm proposed in [13]. The idea behind this algorithm is to minimize the protocol overhead during the block lookup process in a large scale mobile wireless ad-hoc network. It aims to reduce the number of messages required for one lookup request, as well as to minimize the total distance traveled by the lookup message. However, because of the group-oriented design in PMFS, the lookup algorithm also needs to be slightly modified in order to fit PMFS needs.

The basic idea behind this lookup algorithm is based on Chord [14]. Each file block is associated with a specific *key* and stored in a particular node in the mobile wireless ad-hoc network. All the nodes in the network are labeled with a unique node ID as its identifier. All the node IDs and file block keys are mapped into an m-bit identifier space using consistent hashing algorithm. The identifier space may be viewed as a circular space with operation based on module 2^k. Each node is required to store the file block whose key is mapped to the same identifier as that of the identifier of the node. The node responsible for a block with a certain key is called the *successor* node of that key. The *successor* node is defined as the next node on the identifier circle. Each node maintains a list of its *successors*, as well as a figure table to speed up the block lookup process. The only operation this algorithm achieves is that: given a file block key, it will find the node responsible for this block.
The algorithm proposed by Li *et al.* in [13] is quite simple. Based on their observation about the behavior of lookup process in wireless ad-hoc network, they suggest allocating the node identifiers based on the physical locations, rather than the consistent hashing as in Chord. So if two nodes are close on the identifier circles, they are physically close to each other as well. Intuitively, this improvement directly reduces the average path one lookup message need to travel geographically; meanwhile, it also increases the possibility of nodes with more relevant lookup routing information to provide a shortcut on the lookup path, thus the number of lookup message required for one request may be reduced.

2.3 Storage Management

Ideally, when one node is requesting to store a file within a storage system, the obvious solution is to keep the file blocks as physically close as possible. So the network bandwidth can be minimized for this storing process. However, this solution is impossible due to two issues. First, all the nodes within the storage system can request these file blocks, from anywhere at any time. Even though the storing process is optimized locally, the retrieval process will be hectic in the system. Second, all the nodes are constantly moving in wireless ad-hoc network. The temporary physical neighbors can be farther apart from each other a while later. Then it is rather complicated to keep track of all the file blocks within this mobile network; the retrieval process is not straight forward either.

One obvious observation worth mentioning here is that, in a file sharing storage system like PMFS, the ratio of file being retrieved is much higher than it being stored (which is only once). Therefore we are willing to use more network bandwidth and capacity in the block storing process, than it being consumed in the retrieval process. Hence, our performance optimization is mainly focused on the block retrieval.

3. PMFS System Design

PMFS provides a read-only file storing/sharing system based on mobile wireless ad-hoc networks. It consists of a large number of mobile wireless nodes, which may serve as both server and client at the same time. Each individual node has PMFS software installed, while it also contributes its available storage space for the system. If any node is unwilling to participate in the storage contribution, it has the flexibility to behave as an access point only to the PMFS storage system. In addition, any node may change its contribution of storage space at any time. Overall, regardless of the node functionality (client or/and server) within the mobile wireless ad-hoc networks, users may access any node at any time through the application interface, to store or retrieve a specific file, providing the file name and corresponding password.

The following are the user operations that PMFS supports in application interface:

- **Store**: Boolean = store (filename, private password, public password, time to keep). The user inserts a specific file into PMFS, by providing the file name, his/her private password and a public password for

the file, and also specifying how long the file is expected to be kept in the system. In return, the user receives a success or failure notice.

- **Retrieve**: file = retrieve (filename, public password). This function is made for public users to share the stored files in PMFS. By providing the public password, any users may view the files from any PMFS participating node that is connected to the mobile wireless ad-hoc network.
- **Check**: filename = check (public password). If a user only knows the public password for a file without knowing its exact filename, this function helps user to search for the corresponding filename.
- **Update**: boolean = update (filename, private password). PMFS reserves the flexibility for the original file publisher to update his/her file stored in the system. PMFS will update the original file only if the user provides the correct private password for the corresponding filename.

Because of the node mobility and frequent network partition, PMFS manages its mobile nodes in different groups according to its similar velocities [6]. Figure 3 illustrates the basic group-oriented structure of the PMFS system.

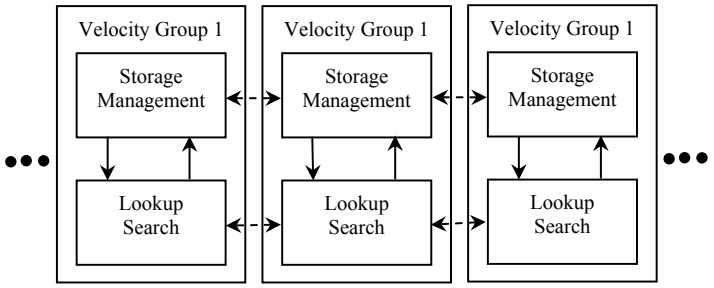
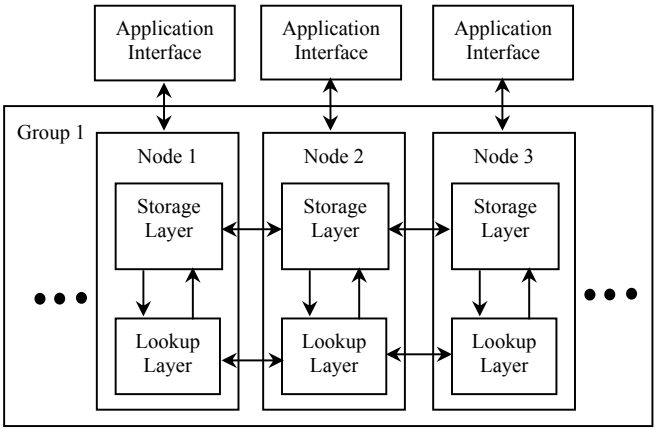


Fig. 3 Group-oriented PMFS System Structure

As shown in Figure 3, all the nodes are categorized in different groups, and both the storage management and lookup search are done independently within each group. Although the communications between groups still exist (as shown in dashed lines), since all the nodes are still free to talk to its neighbors, PMFS tends to make the storage arrangement within its group first, then it will contact the other groups for replications or storage supports. However, faster file retrievals can be achieved by communications between different groups through local broadcasting.

Because the storage management and lookup search are identical in all PMFS velocity groups, we may focus on one group structure in detail for now. Within one PMFS group, all nodes have two software layers: the lookup layer and storage layer, as shown in Figure 4.

Fig. 4 Layered Structure in one PMFS Velocity Group



The PMFS individual group structure is similar to CFS, except the lookup algorithm used in the lookup layer is adapted from [13], in order to work with mobile ad-hoc networks more efficiently. Instead of randomly mapping the nodes into the identifier ring like CFS, PMFS assigns the node identifiers within each group, based on the approximation of their physical locations associated with the group creation time (group ID). The objective of such algorithm is to reduce the number of lookup messages required for one lookup request, and minimize the distance the request message needs to travel. When users access PMFS through application interface with the user operation commands mentioned previously, the storage layer receives the request from the interface. It passes the command over to the lookup layer to find the corresponding nodes. Meanwhile, the storage layer takes care of the main block storage features, such as file block division, blocks distribution, etc. For one PMFS velocity group, the functions of the two software layers are listed as the following:

1. Storage Layer:

- Map filename with private password to an unique private key, and map filename with public password to a public key;
- Divide each file into blocks;
- Sign the root-block with the private key and use the public key as its block ID;
- Provide all blocks with a unique block ID using content-hash;
- Insert each block into PMFS storage layer using the given public key and its block ID;
- Interact with lookup layer during node allocation for storing or retrieving file blocks;
- Define and maintain the velocity groups;
- Provide reliable storage of individual blocks by erasure coding within the group;
- Prevent data loss from network partitions;
- Responsible for the system performance: data availability, bandwidth usage efficiency, load balancing, and delay time.

2. Lookup Layer:

- Assign node identifiers so the lookup algorithm is maximized among mobile nodes in wireless mobile ad-hoc networks;
- Decide nodes responsible for specific blocks within a group, by consistent hashing to map both the node identifier and the file block keys into identifier space;
- Maintain the routing table for lookups.

3.1 Operations of the System

A. Velocity Grouping Technique

In PMFS, each node is assigned a unique identifier, and is capable of monitoring its current position through either GPS devices or any other signal sources. The position will be presented as a two-dimensional Cartesian coordinate. Through the history of its successive locations, each node can then calculate its velocity, $v_i = (v_x,v_y)$. Then by exchanging the velocity information between nodes, all the nodes will eventually be categorized into different velocity groups. The detailed mobility group identification is explained later on.

To identify the group patterns in the decentralized PMFS system, we propose to adapt the same Sequential Clustering (SC) algorithm for pattern recognition mentioned in [6]. In order to find its correct mobility group upon the first joint in PMFS, each node needs to perform a few operations following the SC algorithm. First, each node needs to compare its own velocity with all other mean group velocities v_m, obtained from its neighbors. The comparisons are done through the velocities in the Cartesian domain. If any Cartesian distance between the node velocity v_i and one mean group velocity v_m is within certain threshold, the node is then classified and labeled as a member of that group. Otherwise, a new group will be created, with this node as the first member in the group. Similarly, all the nodes will join PMFS classified as different velocity groups.

At the beginning of the PMFS setup, the first node to join the system will be categorized as the first member of the first group. As the number of nodes increases in the system, they all follow the SC algorithm to identify its group numbers, with the information provided by their neighbors, and with assistances from their group dummy servers.

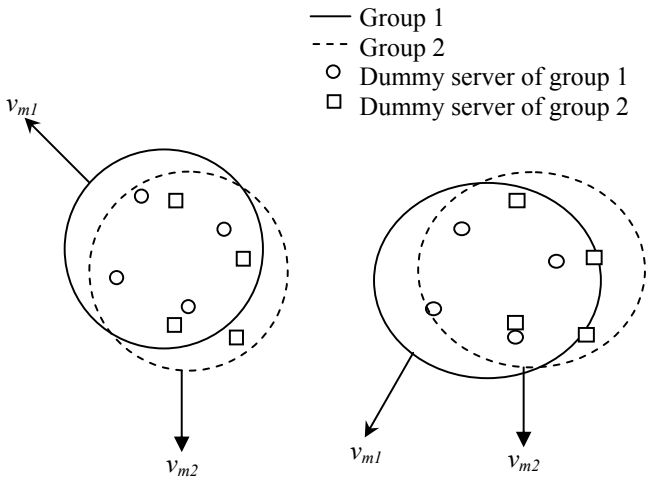
All the dummy servers are dynamically generated among each individual group in PMFS, and maintained in a distributed manner. Because of this dynamic generation of dummy servers, each server is free to transfer its service instance to another group member at any time. The reason we call them dummy servers, is that no mobile node relies on them for either block storing or retrieving. While all the nodes are moving in a group-based velocity, the servers are only intended to help with the group setup and to keep the group maintenance. And at the same time, dummy servers exchange information among themselves in order to keep each other updated. Another service that dummy servers support is that they collect storage information from all the mobile nodes within the group. Every time when dummy server passes by its own group member, it will ask the neighboring member for its most recent storage list. However, it is almost impossible for one dummy server to meet all its group members in the system. The storage lists only remain as rough estimations for reference in dummy servers. When two dummy servers within the same group pass by each other, they update each other's group information accordingly, such as the most recent v_m and the most recent group storage list, etc. This information exchange will improve the estimations, but they may still not reflect the current group status in PMFS.

During the PMFS initial setup, the group joining process occurs. The first node in the system is automatically selected as the dummy server for the first group in the system. As the only member in the group, this node stores its own velocity as the mean group velocity. Also, it uses the current group creation time as its unique group ID. Unless two groups are created at the exact same time, this avoids the duplicated group IDs. When the second node is added to the system, it will follow the SC algorithm explained previously and find out its group identity. If it belongs to the first group, it will then take the group ID from the first group and piggyback its own velocity to the dummy server, so that the mean group velocity can be recalculated and broadcasted back; otherwise, it will create another new group with different v_m and group ID. Thus all the nodes will eventually join the system using the mean group velocity information obtained from its neighbors. During the group joining process in PMFS, the founder of each group is also referred to as the *root server*, since it becomes a dummy server for the group automatically. Suppose each group has M number of dummy servers. This is similar to the virtual backbone selection in [16]. In the bootstrapping procedure, all the nodes broadcast their existences to the network. Then each rooter server from individual group will select M-1 group members to form the dummy servers. Once the selection is done, the root server will just become another regular dummy server moving around freely like every other node.

In the group maintenance phase of PMFS, all the mean group velocities are recalculated in dummy servers, with a record of update *time stamps*. Each time stamp reflects the time of the last update done on the v_m. When a node joins the system, very unlikely it will happen to have a group dummy server in the neighborhood, because the ratio of M/m (number of dummy server to the number of nodes in one group) is quite small. Thus, every time after a new node joins a group, it needs to inform its dummy server about its velocity v_i, so that the new v_m can be calculated for the group. So it will keep on searching for any of its group dummy servers until it actually finds one and sends in its v_i. Then the dummy server updates the mean group velocity, and broadcasts periodically to its neighbors. Due to the broadcast nature of mobile wireless ad-hoc network, each node also periodically broadcasts its group ID and the mean group velocity to other neighbors. Because every node, including the dummy servers, stores v_m associated with the time stamp, all the

group nodes are free to exchange their most updated mean group velocity among all the group neighbors.

Because of the node mobility in wireless ad-hoc network, each node periodically broadcasts its group ID and v_m, group mean velocity, to its neighbors. If two nodes with different group IDs happen to have similar velocities, the node with later group ID (later creation time) will be merged into the other group. This prevents the situation when two nodes with similar velocities join the system farther apart, thus two different groups are formed around them separately. But as they move closer eventually, we want to minimize the total number of velocity groups in the system. Thus the group created later will be merged into the earlier group with similar group velocity. Another important role of dummy servers is to predict the network partition. From periodic broadcast, all the dummy servers are aware of other neighbors' v_m in other velocity groups. Thus, similar to the velocity comparison done for group classification in every node, each dummy server is constantly comparing its v_m with others. If the Cartesian comparison result reaches certain threshold, it will broadcast a *partition alert* to all the nodes in PMFS. Figure 5 depicts two scenarios of whether or not dummy server will send out the partition alert.



(a) Partition Alert (b) No Partition Alert

Fig. 5 Partition Prediction

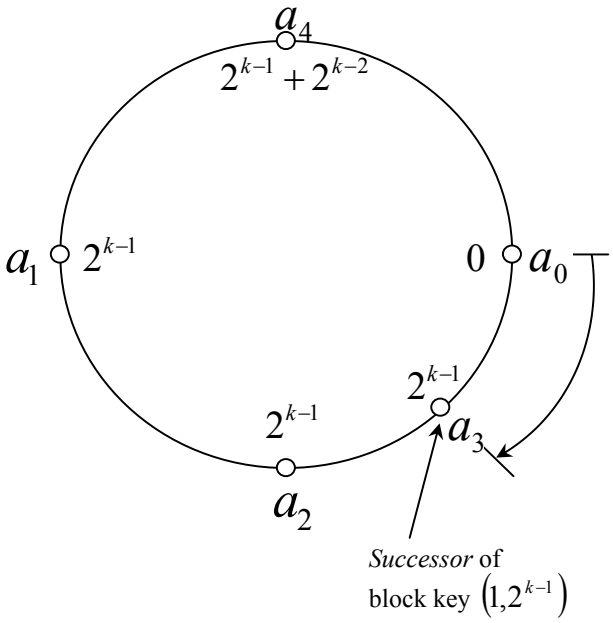
In Figure 5(a), the velocity difference between group 1 and group 2 exceeds the threshold, thus the dummy server from each group, whichever detects the partition first, will send out the partition alert. On the other hand, the dummy servers in Figure 5(b) will not detect the partition, because the mean group velocities are relatively close to each other. However, the exact threshold for such partition alert is beyond the scope of this paper.

Upon the partition alert, PMFS needs to check all the different file blocks stored in both partitions, so that it will transfer and store the missing blocks accordingly.

B. Group-Oriented Lookup Algorithm

In order to manage file blocks more reliably, PMFS is group-oriented, mainly to reduce the impact of the frequent network partitions on PMFS in mobile wireless ad-hoc networks. So the lookup layer in each node needs to recognize its own group members first, before it assigns a unique node identifier to another node. So, instead of assigning group ID globally in the entire wireless ad-hoc network, PMFS manages all the mobile nodes in velocity groups, and all the node identifiers are assigned based on each group. In other words, we apply the algorithm proposed in [13] to smaller units of PMFS wireless mobile nodes in their velocity groups. And the maximum number of nodes in each group is $m = 2^k$, where k is the number of bits assigned for node ID. Figure 6 illustrates the node ID allocation algorithm applied to one PMFS group.

Fig. 6 Allocation of Node Identifiers in PMFS Velocity Group



As explained earlier, when a node joins PMFS, it first contacts all its neighbors to determine its group ID. Due to the node mobility in mobile wireless ad-hoc network, the node a_1 joins the group of node a_0. The node a_1 will be located farther apart from a_0 geographically, comparing to a_2 and a_0. a_2 is another node joins the same group through a_0 after a_1. This is a key insight briefly mentioned in [13]. Therefore, each node in PMFS always assigns the furthest ID available on the identifier circle to its group member. After each assignment, the previous identifier space is divided up into two regions, and each node is then responsible for one region.

For example, as show in Figure 6, when the node a_0 first creates the group, it starts with first node ID 0. It is in charge of the entire identifier circle from 0 to 2^k. After the node a_1 decides to join the group based on the mean group velocity provided by a_0, a_0 then assigns a_1 as 2^{k-1}. And now a_1 is in charge of the upper half of the identifier circle and a_0 only has the bottom half left in

charge. And node a_2 obtained its group ID from a_0; then it took over another half of a_0's bottom half circle. Then all the nodes follow the same pattern to join the group. And the identifier circle responsibility is assigned in a clockwise order. Of course, any node within one group will not assign node ID to nodes from other groups. Every group in PMFS will use this algorithm to assign its nodes within the velocity group. So each node is recognized by node ID within the group, and uniquely identified by both the group ID and node ID in the entire system.

Mobile wireless ad-hoc network is always under frequent network partitions. When a node leaves the group, its node ID is returned to its predecessor. There is no extra overhead introduced in the lookup layer. And when network partition happens in PMFS, it does not have major impact on the lookup layer either, because of the group-oriented design. We assume most partition happens based on the velocity groups. If every node in one group partitions from the previous network, its routing table remains the same in the lookup layer within the group.

C. Group-Oriented Storage Management

As depicted earlier in Figure 4, the basic software structure of each PMFS mobile wireless node consists of two layers: storage layer on top of lookup layer. The PMFS storage layer is responsible for storing and retrieving individual file blocks within its group, providing the corresponding unique block ID, also referred to as the *block key*. The storage layer communicates with the lookup layer to locate these blocks. This structure is similar to Chord [14]. In fact, each group in PMFS can be looked on as a mini storage system, which has independent storing and retrieving capabilities on its own. However, a single group will have poor performance in mobile wireless ad-hoc network, due to the lack of the optimal solutions for bandwidth efficiency and network partition issues.

On the other hand, Figure 3 illustrates the group correlations among various groups. Because of the frequent network partitions in mobile wireless ad-hoc network, it is crucial for PMFS to keep enough replicas among its mobile nodes; hence the file will not be lost after network partitions. Besides the concern of data availability in ad-hoc networks, the efficient usage of network bandwidth is another key factor that directly affects the performance of PMFS. As the result, PMFS proposes this group-oriented system structure.

PMFS attempts to optimize its performance in the storage layer. Erasure coded replication [17] is used in a group base, in order to improve data availability within individual groups. Block replication is also provided between groups, to prevent data loss through network partitions. Simple caching techniques are implemented in PMFS to speed up the lookup process as well as to minimize the multi-hop communications required. Group communications through local broadcasting operations are permitted to reduce the bandwidth consumption in block lookup process as well. In other words, PMFS takes the advantage of its group oriented structure and achieves its best performance through group cooperation.

(a) Data Availability

In order to achieve higher data availability in mobile wireless ad-hoc networks, PMFS storage layers deploy three different techniques to ensure its reliable storage service.

Erasure Coded Replication

Erasure coding (EC) has been carefully studied in [17, 18], where it has been proved to be highly efficient and durable in terms of network bandwidth, storage capacity and fault tolerance. Here, we propose the use of such erasure coded replication scheme in each group of PMFS. The concept of this scheme is simple. In PMFS, the storage layer of every node divides each file in i blocks, and then encodes them into j blocks, where $i <= j$. The ratio of j/i is called the *stretch factor* of the erasure code, which reflects the amount of redundancy added to the original file. After the encoding, the storage layer distributes them to their key successors within the group for block storage, just like other regular blocks. But the benefit from this process is that the original file can be reconstructed from any i of out of those j encoded blocks.

The benefits of this scheme applied to every PMFS group are summarized as the following:

- Intuitively, the data availability is proportionally improved through this replication. As the *stretch factor* increases, the number of possible block combinations, through which the file can be reconstructed, has been increased. Thus, each PMFS group can tolerate a certain degree of network failures without affecting the file retrieval service to the end users.
- At the same time, this flexible choice in our block retrieving process also helps to save network bandwidth. When the lookup layer searches for blocks to reconstruct a requested file, it will first search for the necessary number of blocks that are geographically closer to itself[1]. This reduces the total lookup path for one file retrieval, and thus saves the bandwidth in the network traffic.
- And studies also show, for the same level of data availability, this scheme significantly reduces the storage required, comparing to direct block replications.
- One last important reason, why erasure coded replication is applied to PMFS in such a group-based fashion, is to improve the data availability after network partitions happen in mobile wireless ad-hoc network. When one PMFS group is partitioned away from the original network, erasure coding increases the data availability within both partitions, because it does so independently for each group.

[1] This means the lookup layer will find the block keys that are closer to its node ID on the identifier circle first.

File Replication - Best Effort Replication

Comparing to erasure coding, file replication uses much more network bandwidth and storage space. However, it is unavoidable between PMFS groups, which are all built upon wireless ad-hoc networks with frequent network partitions. When a partition happens, erasure coding can increase the file availability in every group, only if the file was actually stored in the group previously. In other words, erasure coding does not create file when it is missing. It only gives more possibility for each PMFS group to reconstruct their files, even with some missing or damaged blocks, suffering from either the network partition or other network failures. Therefore, the straightforward solution to this problem is to store one erasure encoded copy of each file in every PMFS group within the mobile wireless ad-hoc network.

However, keeping a copy of each file in every group within PMFS is rather ideal. Due to the decentralized control, no node is aware of how many groups currently existing in PMFS. Thus each node will try individually to keep estimation on the number of existing groups. When a node first joins PMFS and obtains a group ID, it automatically starts a *group list* starting from an empty list. Every time it receives a new group ID from neighbor broadcasting, it saves the group ID in its *group list*. And periodically, every node also compares its own group list with neighbors' group lists, and updates its own list if any new group ID is found. However, this only gives each node a rough estimation on the number of currently existing PMFS groups. PMFS neither has control in the group lists when a group leaves the system, nor guarantees for every node to have an up-to-date list when a new node joins.

In each file replication process, PMFS storage layer takes the advantage of local broadcasting nature of wireless networks. When storage layer in the node *a* receives a storing request from application layer, it first broadcasts this request and its group ID to all the neighbors, all the nodes from different groups will reply by sending back their group IDs. Then the node *a* will send a copy of the entire file to the node which replies back first from each group. After all the neighbors receive a copy of the file, they will apply erasure coding and distribute all the file blocks following the same algorithm introduced earlier. This results in the similar storage mapping structure in every group, which becomes beneficial later. On the other hand, most likely the node *a* will not be neighboring with all other group members when the request arrives. Thus the node *a* will be set on *replication alert*, and keep track of which groups have received the file replicas and which groups have not. As the node *a* moves around and network topology constantly changes in mobile wireless ad-hoc network, the node *a* will pay attention to all its neighbors. If the node *a* recognizes any neighbor, who is from the group that does not have the file replica yet, it will the send the file to that neighbor through local broadcasting; After looking around for a period of time *T*, the *file replication alert* expires and the node *a* stops its search, even if some groups in PMFS have not yet received a replica of the file. This is why this file replication scheme is called the *best effort replication*.

Partition Prediction - Best Effort Block Transfer

Similar to the *best effort replication*, each PMFS group also attempts to provide possible network partition prediction through a set of its *dummy servers*. Dummy servers are dynamically generated among groups from each *root server*, and serve its group voluntarily. But upon the assignment of such service, one important role that dummy servers must fulfill for its group is to collect the current up-to-date *block storage list* from its group members through periodic local broadcasting, as they pass by the neighborhoods. And when two dummy servers from the same group become in contact with each other locally, they will also exchange and update each other's collection list. Therefore, each dummy server tries to maintain a rough idea of what files blocks are stored within the group. As one can tell, the completeness of each storage list stored in each dummy server is rather random, depending on the number of neighboring group members it meets during its mobile service. The service provided through such information process is labeled as PMFS *best effort*.

As illustrated in Figure 5 earlier, dummy servers watch out for the possibility of network partitions during its mobile service. When any dummy server detects a network partition in its group, through the comparisons of v_m with other groups, it will broadcast a *partition alert* to the entire network through either single- or multi-hop communication. This method is proposed based on an assumption that network partitions do not happen constantly in PMFS. Upon the receiving of the *network partition alert*, all dummy servers from other groups will look for the dummy servers that are partitioning away from the current network. Once they succeed, the estimated block storage lists are exchanged and compared between two groups. Any block missing from either group will be copied and stored according to block key. However, an improved mechanism is necessary for dummy servers to avoid redundant block transfers, because all the dummy servers from other groups can request the storage list from partitioning dummy servers concurrently.

This *best effort block transfer* is the last mechanism we propose for PMFS to improve its data availability even after network partitions. Its intention is to ensure the number of block replicas in each PMFS group, one last time before network partitions. Through partition prediction, together with erasure coding within every group and file replication among groups, PMFS attempts to achieve higher data availabilities for its users regardless of their accessing points in the mobile wireless ad-hoc network. These weaker semantics of *best effort* avoids the complex agreements protocols among the nodes and groups in PMFS.

b). Bandwidth Efficiency

While PMFS tries to keep up with the level of data availability by using three different types of replication and recovery mechanism in previous subsection, it requires a significant amount of network bandwidth to complete the task. But upon

the assumption that the PMFS network is relatively stable (as for network partitioning does not happen constantly), we are willing to afford such usage to keep reliable block storage. However, on the other hand, as files are shared in the entire PMFS system with constant retrieval request, PMFS tries to reduce the bandwidth consumption for such purpose. It achieves so by providing group communications for faster retrieval path and local caching.

Group Communications - Alternative Lookup Path in Block Retrieval

To take advantages of local broadcasting nature in mobile wireless ad-hoc network and the group-oriented structure of PMFS, PMFS allows group communications for an alternative lookup path through local broadcasting in block retrieval process. In each PMFS group, the identical mapping strategy is applied between block keys and individual circular identifier space. When a copy of block is replicated and stored in different groups, this can result in the similar block location among different groups. Conveniently, all the block keys are mapped to the same identifier space as all the nodes, which are also allocated based on their physical locations. Thus when a node ID is closer to a block key on the identifier space, the node is also physically closer to the block. And due to the mobility of each PMFS node, a node received a lookup request for a certain block key may find a closer location for this block in other group through local broadcasting.

With this key observation in PMFS, our group communication method for faster block retrieval is clearly illustrated in the Figure 7.

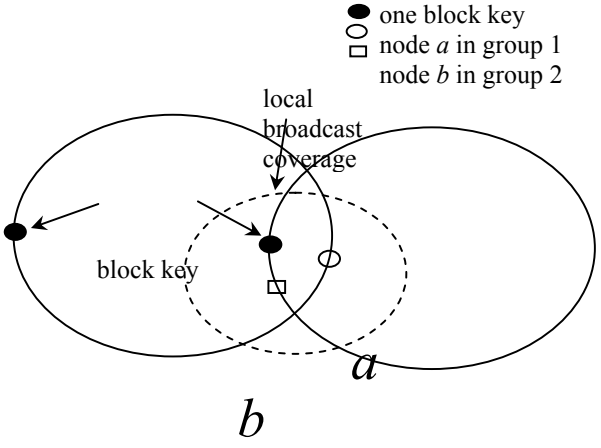


Fig. 7 Group Communication for more Efficient Lookup

Both group 1 and 2 have the same block key mapped to each individual identifier circle. When the node *a* receives a lookup request for the block key, it will first broadcast its node ID, group ID and this block key to its neighbors for an alternative lookup path. All the neighbors will compare the difference between its node ID and the block key, to the difference from the node *a*. If a closer distance is found, most likely the neighbor have a closer copy of this block than the node *a*, thus it will send its calculated difference back to *a*. The node *a* will pick the smallest difference among all the replies, and ask the corresponding neighbor to fetch the file, such as the node *b* as shown in Figure 7.

However, the group communication is based on a strong assumption that all the files will be replicated to each PMFS group. Otherwise, the node *b* may have the smallest difference to the block key on the identifier space, but it may not have the block stored in the group at all. Therefore, reliable storage of each file in every PMFS group is the key to this alternative lookup shortcut.

Local Caching

Lacking a full guarantee of file replications in each PMFS group, a simple local cache may be helpful for a quicker and shorter lookup request. With periodic local broadcasting to its neighbor, every node also sends its block storage list for caching on the neighbors together with its group ID. All the neighbors store these cached locations of block keys for a short period of time, so all the caches are stored in a FIFO manner. When a lookup request is received by the node *a* from its neighbor, it will first check its local cache directory for this block key. Luckily, it will send the cached block successor directly back to the node *a*; otherwise, it will activate the group lookup mechanism introduced previously to compare the block key with its own node ID. The purpose of alternative lookup path and local caching in PMFS is to minimize the lookup delay and reduce the network bandwidth usage, for constant block retrieval requests in PMFS. They together achieve so by providing a simple local cached directory for immediate lookup or a closer physical location for more efficient block retrieval.

4. Results and Analysis

To develop such a large system like PMFS over mobile wireless ad-hoc network, a lot of design details need to be considered and resolved as the development proceeds. In our work, we are focusing on the most fundamental design requirements for such system: data availability, bandwidth efficiency, scalability, durability and persistence of PMFS, under environment of mobile nodes and frequent network partitions in wireless ad-hoc networks. However, in reality, load balance, storage utilization and system security are also design requirements for PMFS, which will significantly affect PMFS performance in the real world application.

4.1 Environment Assumptions

Due to the complexity in partitionable mobile wireless ad-hoc network, the following assumptions are made for PMFS environment in order to simplify our design protocols:

- *Group Mobility:* All nodes, network participants within PMFS, travel and partition in velocity groups. The velocity group orientation is the design foundation of PMFS. Each group communicates and cooperates with one another to achieve desired system performance in PMFS.

- *Physical Location Mapping:* Each node that joins a PMFS group earlier through a group neighbor will be farther apart from that neighbor, comparing to other node that joins later through the same neighbor. This assumption supports the lookup algorithm in PMFS lookup layer, when all node IDs are assigned based on theirs physical locations within each group.
- *Neighbor Awareness:* All nodes are aware of its neighbors through local broadcasting. To take the advantage of wireless broadcasting nature, each PMFS node broadcasts its information locally and periodically to keep communications in a decentralized manner among all neighbors.
- *Similar Storage Capacity:* In order to replicate file blocks among all the groups, each group in PMFS requires sufficient storage space. Thus, similar storage capacity is assumed in each group to keep up with the level of data availability and load balance.
- *Relatively Stable Groups:* To achieve efficient performance, PMFS does not support network with constant partitions, which will cause the system to enter *partition alert* mode constantly and replicate files redundantly. So PMFS only supports mobile wireless ad-hoc networks with considerably stable groups, which means they will have network partitions frequently, but not constantly.

These assumptions simplify our design protocols and provide more workable system environment in wireless mobile ad-hoc networks. The first two assumptions are made based on general observation of ad-hoc networks, while the rest are proposed for simplicity purpose. Among all the above assumptions, similar storage capacities among groups are the hardest to achieve in reality with PMFS decentralized control. Even if this is satisfied, this strong assumption still restricts the flexibility of PMFS participants.

4.2 PMFS Node Functionalities

When a node joins PMFS, it automatically operates in a decentralized manner. But the necessity and contribution of dummy servers cannot be neglected. They not only assist group join process in each PMFS groups, but also monitor the group behaviors and watch out for network partitions. Beside its server duty, each dummy server also servers the group just like every other nodes. Table 2 and Table 3 summarize the possible functions each node are required (as regular participant) or asked (as dummy server) to perform upon the joining of PMFS.

4.3 System Characters Comparison

Since currently there is no file storage system available on mobile wireless ad-hoc networks, we compare the major similarities between PMFS and DFS in the Table 4. From Table 4, the advantages and disadvantages of PMFS are clearly shown. To optimize the PMFS performance, we need to further improve its current advantages, and continuously work on the disadvantages.

4.4 Advantages

The aim of PMFS is focused on data availability and bandwidth efficiency over a large scale partitionable mobile wireless ad-hoc networks. PMFS uses group-oriented storage management to achieve better system performance. It provides reliable file storage service even with frequent network partitions. An optimized lookup algorithm is also adapted in the lookup layer to provide efficient and scalable file block allocation. The main advantages in PMFS are listed as follows:

- *Data availability:* Comparing to CFS, PMFS needs to put a lot more effort in keeping up the level of data availability due to the node mobility in the system and network partitions. Erasure coded replications raises the file reconstruction ability within individual PMFS groups; file block replication between groups and partition predictions are also applied in order to avoid data loss during network partitions. PMFS aims to reduce the impact of node mobility on data storage, so the data loss can be minimized among mobile wireless ad-hoc networks.

Table 2: Function of Regular Node in a PMFS Velocity Group

Function	**Regular Node**
Information Stored:	- Node ID - Node velocity: v_i - Group ID - Group mean velocity with time stamp: v_m - Group list - Successor list and Finger table - Block storage list
Operation Performed:	- Locates file blocks with given block key - Stores blocks within group and retrieves from the closest location found - Sends out periodic local broadcast to neighbors - Exchanges stored information between neighbors for new update - Assigns node ID to newly joined neighbor member - Keeps an estimated list of groups in PMFS - Files *replication alert* for stored files within timeout period - Replies to *Alternative Lookup* request - Sends block storage list for caching purpose

Table 3: Function of Dummy Server in a PMFS Velocity Group

Function	Regular Node
Information Stored:	In addition to what a regular participant stores: - Block storage list in the group - Member list
Operation Performed:	In addition to what a regular participant performs: - Calculates current v_m after every update for its group - Broadcasts new v_m to its group neighbors - Keeps block storage list for the group - Exchanges v_m and block storage list between each dummy servers with the group - Files *partition alert* upon partition prediction for the group - Keep an estimated number of its group members

Table 4: System Comparison between CFS and PMFS

Characters	PMFS	CFS
wireless	Y	Y
decentralized control	Y	Y
partitionable	Y	X
mobile	Y	X
group-oriented management	Y	X
block-oriented storage	Y	Y
coded replication	Y	X
block replication	Y	Y
caching	Y	Y
efficiency	Y	Y
scalability	Y	Y
data availability	Y	Y
quotas	X	Y
load balance	X	Y
security	X	Y
persistence	Y	Y
pre-fetching	X	Y

- *Efficiency:* Bandwidth limitation has always been a major concern in wireless networks. In PMFS, all node IDs are assigned physically in each group to achieve better lookup efficiency, both in terms of delay time and bandwidth usage. Caching and alternative lookup request among groups save network bandwidth from unnecessary and longer lookup path as well as decreases the lookup delay time.
- *Decentralized control:* There is no centralized control in PMFS. Although *dummy servers* server its group for the v_m calculation and partition prediction, no node members will request any specific information from them. All information is still passed around through local broadcasting. All nodes exchange and update information through each other in a decentralized manner.
- *Scalability:* Due to the group-oriented management and optimized lookup search, PMFS is capable of supporting a large number of participants. Although the total number is still restricted by the number of groups allowed in the system, the system itself is still quite scalable in each group.
- *Flexibility:* All nodes in PMFS can be used as access points to all the files stored in the system. All nodes provide equal service from users' point of view. Even though some nodes are assigned as *dummy servers*, this is transparent from their users. Thus all the files can be retrieved concurrently through different access points.
- *Durability:* Because of all the efforts in keeping up with the satisfactory level of data availability in PMFS, its durability is also significantly increased. It can survive certain levels of data loss through either network failures or network partitions, by keeping a number of duplicated blocks stored in different groups.
- *Best Effort:* Best effort simplifies the design protocol and reduces the protocol overhead. It keeps the decentralized control in the system and self-configurable nature in each PMFS group. However, it also decreases the service reliability in PMFS at the same time, which is a disadvantage of such technique.
- *Persistence:* When user stores a file in PMFS, he/she needs to specify how long the file is expected to be stored in the system. PMFS maps all the information together with filename and password into a unique private key, which later on becomes the signature on the root block. Then each node in charge of the root block will check the expiration date on a daily base. Once it expires, the node will remove the block from its storage space and inform other members within its velocity group. So others will remove the stored file blocks corresponding to the same root block.

4.5 Disadvantages

So far, we are only focusing on how to reach the level of data availability and efficiency in PMFS. In order to provide high data availability in PMFS, the number of replicas increases as the number of groups increases in the system. This results in a number of disadvantages in PMFS.

- *Storage Capacity:* However, due to the high demand in storing replicas among different groups, storage capacity in PMFS is another major issue. Both file replications and erasure coded replications need to be stored somewhere, requiring sufficient system storage space. And also because of the file replications between each groups, similar storage space required for PMFS as well. This is rather impractical in reality.

- *High Redundancy:* Because of the number of replicas PMFS needs to produce and store among groups, the protocol overhead is highly redundant.
- *Best Effort:* For efficient lookup search through alternative path in other groups, a reliable file replication between each group is a requirement. But through best effort replication technique, this requirement is not satisfied, thus alternative path lookup search may become very inefficient if file block is missing from the group.

These disadvantages introduced above may be reduced if the maximum number of PMFS groups is restricted. Thus, the number of replicas produced in PMFS will be constrained, and PMFS will have better controls managing its participants. While group numbers are large, more communications between each groups are needed, thus more protocol overhead and less chance for best effort to be successful.

Meanwhile, from the comparison with CFS in Table 4, there are a few system characteristics PMFS needs to improve on:

- *Load Balance:* PMFS distributes the storage load for large files by dividing each file into many different file blocks. Then PMFS maps/stores these blocks among different nodes within groups, based on the block IDs and node IDs. However, when each PMFS node is mapped into its corresponding identifier circle, the distribution of node allocation on the circle is rather random. This results in the possibility that certain nodes are running out of storage space, while others may be quite empty.
- *Security:* Security policy and mechanisms are important for the PMFS to prevent any potential service attacks through networks and other channels. To provide mechanisms in performing authentication, authorization and cryptography on user data and files are necessary efforts to ensure the correctness and integrity of files.

4.6 Simulation Experiments

Simulations would validate PMFS operations and functionality. Due to the time frame for this special topic, only expected simulation results are discussed here.

Before running these simulations, we would need an environment model that reflects the nature of mobile wireless ad-hoc networks: mobile nodes with frequent network partitions, where each node communicates with one another in a decentralized manner through local broadcasting within certain range. We need to pre-configure the number of nodes in the system, the mobile space, and each node transmission range.

Data Availability

One of the main goals in PMFS is to achieve high data availability in mobile wireless ad-hoc network. Three different techniques are proposed to prevent data loss: erasure coded replications, file replication among all groups and block recovery upon partition prediction. Simulations would demonstrate how much each technique can improve data availability in PMFS.

With the environment model we have, we would insert a large number of uniquely identified files (each divided into different number of blocks) among all the nodes. Then we need to define the number of partitioning velocity groups that will be formed during the process. For simplicity, we will start with two. Once the system is ready, we would run five sets of different implementations to see the block availability on each partition. All these simulations will run based on "Data Availability vs. Time" for each partition. The data availability will be calculated based on the number of files that can be reconstructed over the original file number.

- No replication. No replication effort at all.
- Erasure Coded Replication. When partition happens, certain group members will choose not to follow their own groups based on a random decision. File can be reconstructed with certain percentage of missing blocks (depends on the stretch factor in erasure coding).
- File Replication. File are replicated and distributed among all the groups with random failures. This is to model the *best effort* file replication. A file can only be reconstructed if all of its blocks are presented in one partition.
- Partition Prediction Replication. Similar to file replications, but instead of random failure on the entire file, random failures are applied to block replication among groups.
- All. Combine all three above in one implementation.

We expect the last simulation would result in best data availability over time domain.

Efficiency

In order to improve lookup efficiency in PMFS, caching and group communications for alternative path are proposed in the system. Simulations would show how effective these techniques will be in mobile wireless ad-hoc network.

Upon our simulated network environment model, we assign certain number of velocity groups. In each group, we assign all node IDs based on the initial location, closer pair with more different IDs. Then a large number of file blocks are assigned in each group, based on their node IDs. And each node has a successor list for block storage within its group. Similarly, four sets of simulations would be run for this efficiency test. All simulations would have results as "Number of Lookup Message vs. Trial number". The averages will be compared to show the improvement for each technique.

- None. No optimization is done. Each node looks up for blocks within its own group independently.
- Cache. Each node sends its storage info to neighbor periodically. And lookup starts with cache in each node first.

- Alternative Path. Each node is allowed to ask for alternative closer path from other groups through local broadcasting.
- Both. Combine all three above in one implementation.

The last run is expected to show us the best result as it is the closest simulation to PMFS.

Scalability

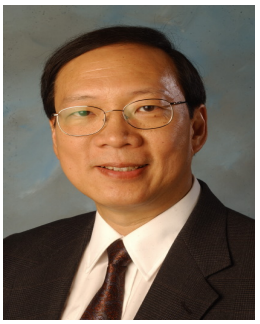
Both efficiency and data availability simulations would be run separately again using the optimal implementations (best result from each simulation). They would both run on the environment model with an increase in the total number of nodes and in the total mobile space. The result would be shown as "Data Availability vs. Number of Nodes" and "Number of Lookup Message vs. Number of Nodes". For the former, we would hope for a gradual decrease, rather than linear decrease; as for the latter, we would expect to see a logarithmic increase, rather than linear increase.

5. Conclusions

PMFS, Paritionable Mobile File System, is an advanced file storing/sharing system over mobile wireless ad-hoc networks handling frequent network partitions. The main design focus has been to improve system performance and provide efficient and scalable file storage and retrieval services. The system has been compared to CFS, a peer-to-peer storage system. The advantages/disadvantages of PMFS have been discussed theoretically. Expected simulations have been presented in detail. The proposed simulations are expected to show the effectiveness of various techniques applied in PMFS for the improvements of data availability and bandwidth efficiency.

References

[1] "Napster website. http://www.napster.com".
[2] "Morpheus website. http:// www.morpheus.com".
[3] "Gnutella website. http://www.gnutella.com".
[4] I. Clarke, O. Sandberg, B. Wiley, and T. Hong, "Freenet: A distributed anonymous information storage and retrieval system", in *Proceedings of the Workshop on Design Issue in Anonymity and Unobservability*, 2000.
[5] E.M. Royer and C.K.Toh, "A Review of Current Routing Protocols for Ad-hoc Mobile Wireless Networks", *IEEE Personal Communications*, pp. 46-55, April 1999.
[6] K.H. Wang and B.Li, "Efficient and Guaranteed Service Coverage in Partitionable Mobile Ad-hoc Networks", in *Proceedings of IEEE INFOCOM*, 2001.
[7] S. Jiang, D. He, and J. Rao, " A Prediction-based Link Availability Estimation for Mobile Ad-hoc Networks", in *Proceedings of IEEE INFOCOM*, 2001.
[8] A. McDonald and T. Znati, "A Mobility-Based Framework for Adaptive Clustering in Wireless Ad-hoc Networks", *IEEE Journal on Selected Areas in Communications*, vol. 17, no. 8, pp. 1466-1486, August, 1999.
[9] W. Su, S.J. Lee, and M. Gerla, "Mobility Prediction and Routing in Ad-hoc Wireless Networks", *International Journal of Network Management*, 2000.
[10] X. Hong, M. Gerla, and G. Pei and C. Chiang, "A Group Mobility Model for Ad-hoc Wireless Networks", in *Proceedings of ACM/IEEE MSWiM*, 1999.
[11] F. Dabek, M.F. Kaashoek, D. Karger, R. Morris, and I. Stoica, "Wide-Area Cooperative Storage with CFS", in *Proceedings of ACM SOSP*, 2001.
[12] P. Druschel and A. Rowstron, " Past: A Large-scale Persistent Peer-to-Peer storage Utility", in *Proceedings of HotOS VIII*, 2001.
[13] B. Li., Efficient and Scalable Lookup in Mobile Wireless Ad-hoc Networks", in *Proceedings of the 12th IEEE International Conference on Computer Communications and Networks (ICCCN 2003)*, 2003.
[14] I. Stoica, R. Morris, D. Karger, M.F. Kaashoek, and H. Balakrishnan, "Chord: A Scalable Peer-to-Peer Lookup Service for Internet Applications", in *Proceedings of IFIF/ACM International Conference on Distributed Systems Platforms (Middleware 2001)*, 2001.
[15] M. Papadopouli and H. Schulzrinne, "Network Connection Sharing in an Ad-hoc Wireless Network among Collaborative Hosts", in *Proceedings of NoSSDAV*, 1999.
[16] J. Liu, Q. Zhang, W. Zhu and B. Li, "Service Locating for Large-Scale Mobile Ad-hoc Networks", *Kluwer International Journal of Wireless Information Networks*, 2000.
[17] R. Bhagwan, D. Moore, S. Savage, and G.M. Voelker, "Replication Strategies for Highly Available Peer-to-Peer Storage", in *Proceedings of FuDiCo: Future Directions in Distributed Computing*, 2002.
[18] H. Weatherspoon and J.D. Kubiatowicz, "Erasure Coding vs. Replication: A Quantitative Comparison", in *Proceedings of the First International Workshop on Peer-to-Peer Systems (IPTPS 2002)*, 2002.



Weider D. Yu received an M.S. in Computer Science from the State University at Albany, New York, and a Ph.D. from Northwestern University, Evanston, Illinois, in Electrical Engineering and Computer Science. He also attended the MBA program at University of Chicago and completed a security engineering certificate program at Carnegie Mellon University.

As an associate professor in the Computer Engineering Department of College of Engineering at San Jose University, San Jose (Silicon Valley), California, USA, Dr. Yu performs his research and teaching in the areas of Web based software engineering, Web services, Web services security, software security, distributed software engineering, distributed systems, wireless mobile system design, real-time and embedded software systems, systems software, software engineering process, and quality, reliability and performance factors in computer and communication network systems.

Dr. Yu was a Distinguished Member of Technical Staff and Senior Manager at Bell Laboratories, where he did an extensive research and forward looking work in the broad software engineering area for advanced communication software. He was also an adjunct associate professor in the department of Electrical Engineering and Computer Science, University of Illinois at Chicago for a number of years. Dr. Yu has publications on Bell Labs Technical Journal, AT&T Technical Journal, IEEE Journal of Selected Areas in Communications, and various international IEEE and other conferences. He is a senior

member of IEEE and an active member in the professional technical society. He has been a technical program representative or symposium chair for various IEEE international conferences, symposiums and IEEE/ISO committees.

Yan Chen received the B.S. degree in Engineering Science Computer Option from University of Toronto, Toronto, Canada in 2003. She now works for Actel Corporation located at Mountain View, California. She is currently completing her M.S degree in Computer Engineering at San Jose State University, San Jose, California.