A Peer-to-Peer File Sharing System for Wireless Ad-Hoc Networks

Hasan Sözer, Metin Tekkalmaz, and İbrahim Körpeoğlu

***Abstract*—File sharing in wireless ad-hoc networks in a peer-to-peer manner imposes many challenges that make conventional peer-to-peer systems operating on wire-line networks inapplicable for this case. Information and workload distribution as well as routing are major problems for members of a wireless ad-hoc network, which are only aware of their neighborhood. In this paper we propose a system that solves peer-to-peer file-sharing problem for wireless ad-hoc networks. Our system works according to peer-to-peer principles, without requiring a central server, and distributes information regarding the location of shared files among members of the network. By means of a "hashline" and forming a tree-structure based on the topology of the network, the system is able to answer location queries, and also discover and maintain routing information that is used to transfer files from a source-peer to another peer.**

***Index Terms*—Wireless ad-hoc networks, file sharing, peer-to-peer networks**

I. INTRODUCTION

Peer-to-Peer networks have been very popular since their first emergence. Some systems have already been deployed to be functional on the Internet, like Napster [10], Gnutella [7] and Fasttrack [6]. Many peer-to-peer systems currently serve users who are able to share files located at their PCs without requiring information to be used at central servers. Together with the new users of the Internet and the emergence of different types of files to be shared (documents, audio files, etc.), number of users of peer-to-peer systems increases every day.

At the mean time, mobile devices and wireless communication technologies are evolving and becoming very popular. Both areas have experienced rapid improvements during last few years, which led to development of high-performance products. Today, PDAs have almost the same abilities that of ordinary PCs despite their small size and weight. On the other hand, new wireless technologies enable PDAs and other handheld devices to communicate and form ad-hoc networks in an easy and automated way. Bluetooth [2], for instance, is such a technology that uses short-range radio communication and that interconnects handheld electronic devices ranging from cellular phones to PDAs.

Although high-performance handheld devices that communicate with each other through ad-hoc wireless communication technologies are available today, peer-to-peer file sharing in such an environment imposes many challenges that make conventional peer-to-peer systems operating on wire-line networks inapplicable for this case. Peer-to-peer systems were developed as opposed to central approaches in order to increase availability and reliability. In that respect, they are very suitable for wireless ad-hoc networks (WANETs) where spontaneous connections occur and users have relatively higher degree of mobility. However, traditional peer-to-peer systems are not sufficient for providing file sharing in such an environment since:

- Such networks can be formed anytime and anywhere without requiring an infrastructure,
- Nodes in the network may tend to change their locations frequently,
- There is lack of widely accepted and used standards for routing data in mobile ad-hoc networks.

A peer-to-peer file sharing system that is running on Internet may find a desired file at a member node, which is identified by a unique ID. This can be achieved by using centralized or distributed indices that maps the name of the file to the member node's IP address through which the node can be reached. After knowing the IP address of the node from where a file can be downloaded, the network layer of the Internet (IP) would handle all intermediate steps and forwarding needed in order to reach to the node and to perform the download. However, this is not possible on a WANET that does not run an ad-hoc routing algorithm. A WANET may be composed of heterogeneous mobile systems in which a

This work is supported by The Scientific and Research Council of Turkey (TÜBİTAK), grant number 103E014.

H. Sözer is with the Computer Engineering Department, Bilkent University, Ankara, 06533, Turkey (e-mail: hsozer@cs.bilkent.edu.tr).
M. Tekkalmaz is with the Computer Engineering Department, Bilkent University, Ankara, 06533, Turkey (e-mail: metint@cs.bilkent.edu.tr).
İ. Körpeoğlu is with the Computer Engineering Department, Bilkent University, Ankara, 06533, Turkey (e-mail: korpe@cs.bilkent.edu.tr).

standard routing algorithm is not supported at all nodes. Currently, although there are various efforts [4, 5] that propose protocols to route packets in a WANET, we still lack a common and widely used standard routing protocol for this environment. And it seems that it will take some more time before we have a widely accepted common routing protocol and its implementation available and deployed.

Therefore, to support peer-to-peer file sharing in a WANET, we believe that a peer-to-peer system should also provide routing functionality besides providing lookup functionality. In this way, the peer-to-peer system should be able to determine both from *where* and *how* to obtain a file.

In this paper we propose a system that solves peer-to-peer file-sharing problem in wireless ad-hoc networks. Our system works in a peer-to-peer manner and distributes information regarding the location of files that are shared among members of the network. Besides location information, the system also stores routing information as part of a distributed index maintained in the system. While designing the system, we have adapted some techniques from source routing and peer-to-peer location lookup that were previously proposed for wire-line networks.

The remainder of this paper is organized as follows. In the next section, related previous studies are summarized. In section 3, an overview of the system is given, which is followed in section 4 by a detailed description of each operation supported by the system. In section 5, we present a working scenario of the system to show how each operation updates and maintains the distributed location and routing information stored in the system. Finally, in section 6 we give our conclusions and discuss some future work issues.

II. Related Work

Network environments can be grouped in three categories according to their impacts on peer-to-peer file sharing. The first one is wired Local Area Networks (LANs). Handling file sharing is rather easy in wired LANs since they are built-up by relatively low number of computers, which are well known to each other and each of which can communicate to all other nodes directly. Wired LANs are out of interest of this paper since they do not have much common characteristics with the wireless ad-hoc networks, which is the network environment that the paper offers a solution for file sharing. The second network environment is the Internet. It connects huge number of computers and is a transport infrastructure that enables peer-to-peer file sharing. But unlike wired LANs the computers willing to share files are usually have little or no awareness of others. Several works have been carried out in the recent years to cope with problems posed for peer-to-peer file sharing by the dense, highly dynamic and lowly aware nature of the Internet. Napster [10] is one of the earliest and most popular applications, which enables file sharing on the Internet among computers that are hard to predict when to connect and disconnect. The main idea behind Napster is a central server that stores index information (filename and address pairs), which is used to answer queries about where files are stored on the Internet. Once the location of a file is determined, file transfers are carried on peer-to-peer (P2P) manner. Although the actual file transfers are P2P, storage of and accessing index information is done using client-server paradigm. Napster enables easy location lookup by using a central server, but it is affected by the typical weaknesses of centralized systems. More recent works aim fully distributed peer-to-peer systems that store index information in a distributed manner. CAN (Content-Addressable Network) [11], being one of them, is based on a fully distributed hash table. In CAN, filenames are hashed and mapped to points on a d-dimensional space. The d-dimensional space is divided into chunks and distributed among the members of the network where each member is responsible from one portion of the space (i.e. a chunk). Along with a chunk each node stores some information about the neighboring nodes, which makes searching of files possible by providing the location information for files and an overlay network-level routing. Chord [8] is another well known fully distributed peer-to-peer system in which a ring shaped overlay network is applied. Each node on this ring maintains pointers to other nodes at various distances. To gather the location information of a file, these pointers are followed in a manner that shortens the access path as much as possible.

Last type of networks is Wireless Ad-hoc Networks (WANETs), which have dynamic nature causing many difficulties for file sharing as stated before. First work on P2P file sharing on WANETs is 7DS [9] and used to enable nodes browse the web with an intermittent Internet connection, in which, whenever a node fails to connect to the Internet, it can search the required data among peers. Other works on file sharing in WANETs like [1] and [3] are based on partial flooding where the searches are carried basically by queries broadcasted several hops ahead, and where flooding the entire network is prevented by mechanisms like caching and selective routing. Such approaches work fine for small-size WANETs but as the network gets bigger they cause traffic overhead and the probability of finding a file in the network reduces. Our system provides a deterministic way to locate and access files, hence if a file is shared in the WANET, its location can be determined and it can be accessed.

III. System Overview

The system expects three basic functionalities from the underlying network layers:

- Device discovery
- Communication with nodes in the range
- Notification of link failure

Together with these functionalities the system makes use of a fully distributed hash table where keys are the names of the files to be shared and the values are the globally unique locations of these files (MAC address of the device together with the full path of the file on the device may provide this uniqueness) together with necessary routing information which will be described soon. The basic dynamics of the system is as follows. A one-dimensional space (i.e. a line) is used to store (key, value) pairs by mapping each key to a point P on the ”hashline” using a uniform hash function. In fact, any hash function that can map a file name to a real number between 0 and 1 may be used for this purpose. However, uniformity would lead to a more balanced information distribution among the nodes. Each node in the WANET is responsible for storing a segment of the hashline (i.e. the hash table entries which correspond to points that are included in this hashline segment).

We call the node which is responsible for the segment of hashline containing a point P as *P-Node,* and the node which stores a file with name F as *F-Node*. Hence a P-Node stores index information along with location information and an F-Node stores the actual file.

At the highest abstraction level, a file is accessed following the steps listed below:

1. Name of the file to be searched is hashed to determine a point P on the hashline.
2. P-Node is accessed.
3. The location of the searched file, F-node, and the route to that location is determined from P-Node.
4. F-Node is accessed, and the file is downloaded.

These steps seem simple but determining routes between nodes are the heart and distinguishing part of the system. System is designed to cope with this problem using a logical tree structure that is imposed on the nodes of a WANET. The tree-structure helps in accessing to P-Node, and the information obtained from the P-Node helps in determining the route to F-Node from where the file will be downloaded. Hence, although the network may include loops at the link layer, loops are not allowed in the layer at which P2P system is implemented, which is usually the application layer. While the network grows with the addition of new members, a new member node is not permitted to join the same file sharing enabled WANET via more than one link (i.e. via more than one neighboring node). A loop-free network can be achieved by providing a unique network ID (e.g. MAC address of the root node) for each file sharing enabled MANET and not allowing a node to have more than one parent with the same network ID.

The next section describes the details of the design along with the operations in the system.

IV. Operations of the System

There are several basic operations supported by the system to locate files and route the download to enable file sharing. *Node-Join* operation is carried on when a node is connected to a file sharing enabled WANET and *Network-Join* is carried on when two file sharing enabled WANETs are merged. *Access2P-Node* operation is used to find and access the node which stores segment of the hashline including a desired point P. *Access2F-Node* operation is used to find and access the node which stores a desired file with name F. *Insert* and *Delete* operations are used to add a file to the network (i.e. enable sharing) or remove a file from the network. *Recover* operation is carried on to preserve the consistency between the actual location of shared files and the hash table storing the routing information when a disconnection with an adjacent node is detected. Finally *Leave* operation is carried on when a node decides to leave the file sharing enabled WANET.

Some of the operations mentioned above include other operations (e.g. *Join* includes *Access2P-Node* and *Insert*). Detailed information about each operation is given in the following subsections.

A. Node-Join

Whenever a node **N** decides to join a file sharing enabled WANET, the following steps are executed:

1. **N** connects to an already existent node **K** of the network, which is accomplished by the underlying protocols specific to the WANET.
2. **K** assigns a portion of its segment of hashline to **N** and passes related hash table entries to it.
3. **N** adds **K** to the routing path information

maintained at each hash table entry for files indexed at **N** before saving the hash table entries.

4. **N** assigns **K** as its parent and **K** adds **N** to its children list in the logical tree-structure.
5. **N** calls Insert operation for each file it wants to share and whose hashed value is out of its responsibility.

As it can be noticed, the hashline segment assigned to the new node is not randomly determined. Instead, the node, to which the new node directly connects, shares some portion of its responsibility on the hashline (e.g. half of it). This simple design is crucial for easy and efficient routing of location queries to the nodes that can answer them. The corresponding operation is described in detail in section 4.3.

B. Network-Join

Let the nodes **N** and **K** be the members of two distinct file sharing enabled WANETs, **N-Net** and **K-Net** respectively, which are going to merge by **N-K** connection. To obtain a larger file sharing enabled WANET from two smaller ones, the following steps are executed:

1. **N** and **K** decide on which one is going to share its responsibility on hashline, equivalently which one is going to be the parent of other. (Say **N** is chosen as the one to share its area of responsibility using some decision criteria).
2. Every node of **K-Net** on the path from node **K** to the root of **K-Net** (node with no parent), exchanges the parent-child role with its parent, including node **K** and the root of **K-Net**. That is every node on the specified path adds its former parent to its children list and it becomes the parent of its former parent. In this way, **K** becomes the new root of **K-Net**.
3. **K** is connected to **N**, hence **N** becomes the parent of **K**.
4. Based on the new parent-child relationships among **K-Net** nodes and **N**, starting from node **N** each parent shares some portion of its responsibility on the hashline with its children, in an iterative manner.
5. Each node in **K-Net** calls *Insert* operation for each file it wants to share.

C. Access2P-Node

Whenever a node **N** wants to access *P-Node* (i.e. the node which is responsible for the segment of the hashline containing point P), it invokes the *Access2P-Node* operation. A node **K** receiving *Access2P-Node* request follows these rules:

1. If point P is included by the segment of hashline that **K** is responsible for: *P-Node* is found and is **K**.
2. If point P is included by the segment of hashline that one of the children of **K** is responsible for: **K** adds itself to the route list and forwards *Access2P-Node* request to the relevant child node.
3. Otherwise: **K** adds itself to the route list and forwards *Access2P-Node* request to its parent.

Note that initially **N** = **K**. Also note that the *P-Node* finally has the routing information between the node issuing *Access2P-Node* request (i.e. node **N**) and itself, since each node on the path from **N** to *P-Node* adds itself to the routing information carried inside the *Access2P-Node* request.

D. Access2F-Node

Whenever a node **N** wants to access *F-Node* (i.e. the node which contains the file with name F), it invokes the *Access2F-Node* operation, which consists of the following steps:

1. **N** hashes F and determines P, that is P = hash(F).
2. Having point P, **N** invokes the *Access2P-Node* operation with *F-Node* location request, that is **N** asks *P-Node* the routing information from *P-Node* to *F-Node*.
3. Having the route information back to **N**, due to the feature of *Access2P-Node*, the *P-Node* sends to **N** the route from itself to *F-Node* (Remember that the route from *P-Node* to *F-Node* is stored as part of the hash table entry corresponding to point P).
4. **N** combines the route information from itself to *P-Node* and from *P-Node* to *F-Node* and constructs the route necessary to access the *F-Node*.

E. Insert

Whenever a node **N** wants to share a file with name F, it invokes the *Insert* operation, which consists of the following steps:

1. **N** hashes F and determines P, that is P = hash(F).
2. Having point P, **N** invokes the *Access2P-Node* request with insertion as the request type and F as the filename.
3. Upon receiving the request, the *P-Node* stores the filename F and the route information back

to **N**, which is obtained during *Access2P-Node* operation, as part of the hash table entry created.

F. Delete

Whenever a node **N** wants to stop sharing a file with name F, it invokes the *Delete* operation, which consists of the following steps:

1. **N** hashes F and determines P, that is P = hash(F).
2. Having point P, **N** invokes the *Access2P-Node* operation with deletion as the request type and F as the filename.
3. Upon receiving the request, the *P-Node* removes the entry for the file with name F from the hash table.

G. Recover

Whenever a node **N** determines a disconnection with one of its child nodes **K**:

1. **N** regains the responsibility of the segment of hashline that **K** was responsible for.
2. **N** broadcasts to the WANET a message that includes information about the regained segment to force all the nodes to invoke *Insert* operation again for the files whose hashed names are included by the segment that **K** used to be responsible for. In this way, node **N** will have the hash table entries created for these files.

Whenever a node **K** determines a disconnection with its parent node **N**:

1. **K** takes full hashline as the area of responsibility.
2. Starting from **K** each parent shares some portion of its responsibility on the hashline with its children.
3. Each node calls *Insert* operation for each file it wants to share.

H. Leave

When a node **N** wants to leave the file sharing enabled WANET, it invokes the *Leave* operation, which consists of the following steps:

1. **N** invokes the *Delete* operation for each file it shares after which all index information about the files stored in **N** is removed from the WANET.
2. **N** gives its responsibility on its segment of the hashline to its parent.
3. **N** informs its parent $\mathbf{P_N}$ and children $\mathbf{C_1}$, $\mathbf{C_2}$, …, $\mathbf{C_n}$ about its departure to make sure $\mathbf{P_N}$ adds $\mathbf{C_1}$, $\mathbf{C_2}$, …, $\mathbf{C_n}$ to its children list and $\mathbf{C_1}$, $\mathbf{C_2}$, …, $\mathbf{C_n}$ assign $\mathbf{P_N}$ as their parent.

Note that the third step is possible only if all the children of node **N** are in the communication range of $\mathbf{P_N}$. For the children that are not in the communication range of P_N, *Recover* operation is executed.

Due to the nature of ad-hoc networks, nodes are not expected to leave the network with notification. But it may still be the case where *Leave* operation is beneficial. Otherwise, *Recover* operation still handles the situation despite its higher communication cost.

V. A Sample Scenario

After specifying each operation supported by the system, this part of the paper presents a sample scenario in which the way that system works can be observed. Suppose that initially two nodes called A and B meet. A includes files A1, A2, while B has B1, B2, B3. B discovers A, in other words, B joins the network, which is only composed of A. Previously, A was responsible of all hashline and files A1 and A2 were mapped on to this line as depicted in Figure 1.a. As explained in 4.1, when B is connected to A, A divides the entire hashline into two and gives one of them to B. Since, A2 falls within the segment that B is now responsible for, A sends the location information (index information) for file A2 to B. Previous location information for A2 was null, meaning that the file was stored at the same node where the location information is kept. But, from now on, B stores an index entry for A2 with location information like [A2, A]. Then B executes *Insert* operation for files B1 and B2, since these are the files owned by B but they are not mapped to the part of the hashline that B is responsible for. Now, A stores location information, [B1, B] and [B2, B], for these files as depicted in Figure 2.b.

Suppose that a new node C discovers B and connects to it. Again a *Node-Join* operation will be invoked and the hashline segment that B is responsible for will be divided into two parts, as depicted in Figure 1.c. C stores and shares files C1 and C2, which map to the points on the hashline as shown in the figure. First of all, B sends information about A2 to C, since A2 falls now in C's segment of responsibility. C should not only keep information about the node where file A2 can be found, but also keep path information about how it can be reached from C to that node. Therefore, C adds also B to the path information and stores an index entry like [A2, BA]. This indicates that file A2 is stored at node A (right-most node in the path) and the path from C to that node is "AB". Next, C invokes the *Insert* operation both for C1 and C2. C1 maps to the segment controlled by A and C2 maps to the segment controlled by B. Therefore, *Access2P-Node* request reaches to B for file C2, and to A for file C1. So, corresponding nodes stores file names together with their route information to the node where files are actually stored. The route information is obtained during the path traversals of the *Access2P-Node* requests. The current state of location and routing information that is maintained in the network can be observed in Figure 2.c. As the last member of the network, D discovers B and connects to it. B, again divides the segment of hashline it is responsible for into two parts and sends information about B3 to D. After that, D sends information about a single file it owns, D1 to A using *Insert* operation. Final view of the hashline and the network topology together with distributed index information can be observed in figures 1.d and 2.d, respectively.

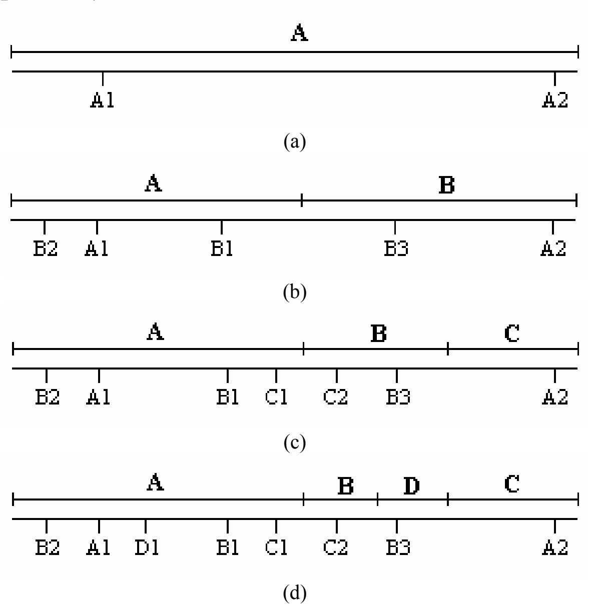
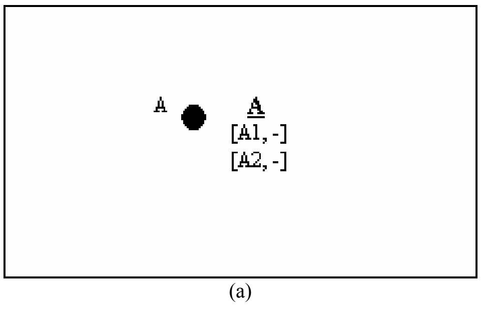


Fig. 1. Hashline states during network formation



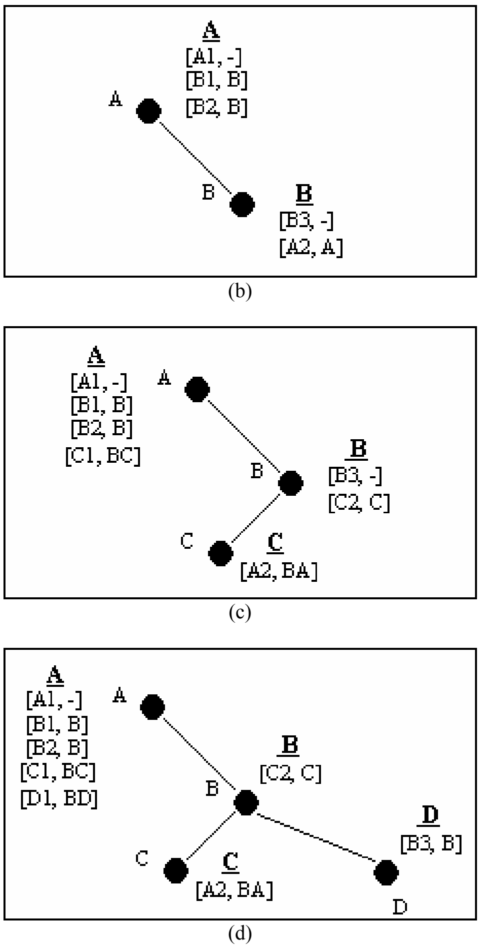
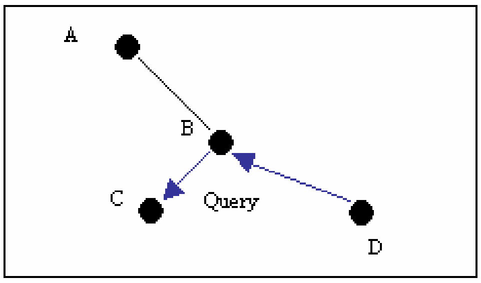
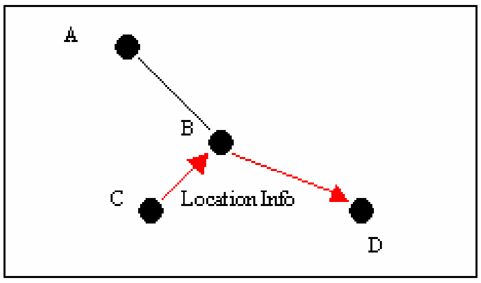


Fig. 2. Network Topology & Information Distribution

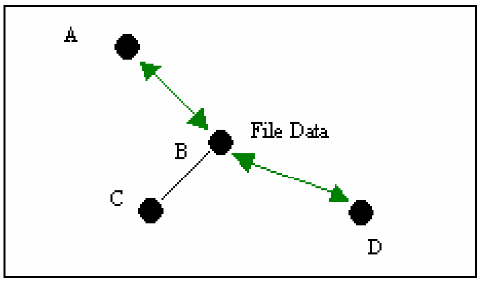
Now, assume that D needs file A2. D does not know where the file A2 resides or even whether such a file exists or not. However, according to the hash value of the filename, it is known that this information is held by another node. D has only one neighbor, B (as its parent) to which the query is forwarded. So, B receives the query, expressed as [A2, D], meaning that file A2 is requested by D. B has two neighbors, A and C. According to the hash value of the filename and the current state of the hashline, B decides to forward the query to C. This is because B knows that one of its children, C in this case, is responsible for the segment of the hashline that includes the point that represents the hash value of the name of requested file. Otherwise, B was going to forward the query to its parent, A. When query is forwarded to C, it is not guarantied that it will be answered by C. C may have some other nodes connected to it meanwhile, so it may further forward the query to one of its children again by looking within which segment the point lies. However, it does not matter for B whether C or some descendant of it answers the query. B only knows that query should be forwarded towards C in order to be resolved. For this particular case, C does not have any children and C holds the location information for A2. The path to source at which the query is initiated is also attached to the query. In this way, C receives a query [A2, BD], which means that node D requested file A2 and its request reached through node B. This path is used in order to send the query response (location information), [A2, BA], back to node D. C generates a query response message, [A2, BA], targeted to D and including the source route information “CBD” that gives the path to be followed. C passes the response to the next node on the path which is B. Again by looking to the path information in the response message, B passes the message to the next node on the path, which is D. D is the originator of the query to locate file A2. D receives the query response message and the message includes the location information [A2, BA]. Now, D knows that the file A2 is located at node A and D also knows two paths: the path from D to C (the node which holds the location information) and the path from C to A (the node which stores the file). Node D concatenates those paths (D-B-C-B-A) and then eliminates the unnecessary loop B-C-B. The result is “D-B-A”, the path from D to A. This is the path from query originator D to the node A that stores and shares the file A2. By means of this path, file A2 can now be directly reached and downloaded from A. These steps are depicted in figures 3.a through 3.c.



(a)



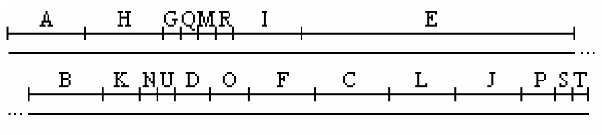
(b)



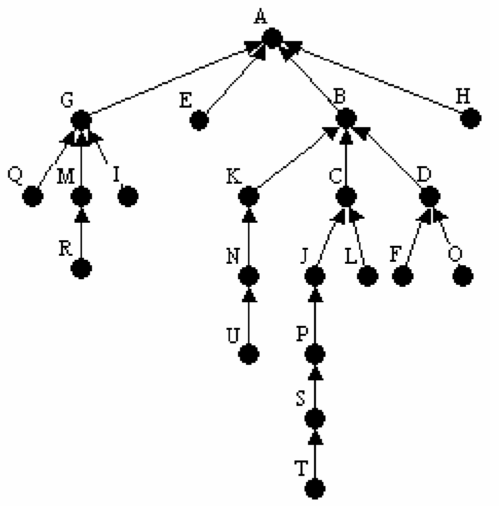
(c)

Fig. 3. File Search & Retrieval

As more nodes join to the file sharing enabled network as explained in section IV.A, the tree-structure become more involved. In Figure 4 a later phase of the WANET shown in Figure 1 and Figure 2 is given. New nodes have joined to the WANET in alphabetical order. Connections between nodes can be inferred from the tree-structure given in Figure 4.b. The state of the hashline is shown in Figure 4.a.



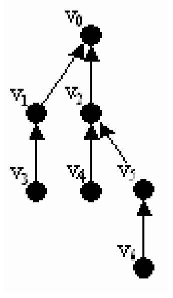
(a)



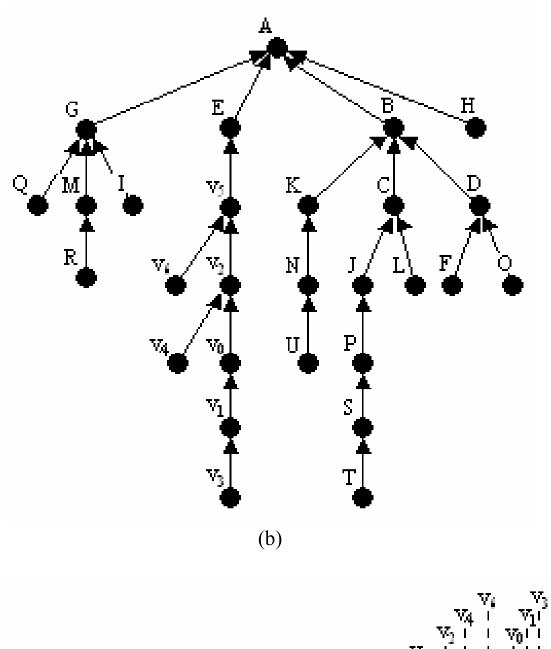
(b)

Fig. 4. Before Network Join

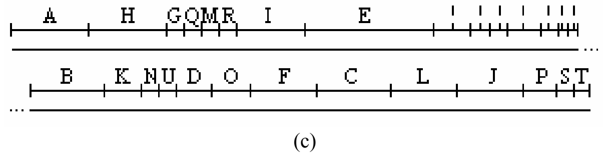
Now, consider the case where the file sharing enabled WANET in Figure 4 (WANET-1) merges with another file sharing enabled MANET shown in Figure 5.a (WANET-2) and assume that the connecting nodes are E of WANET-1 and V_5 of WANET-2. For such a merge operation, Network-Join procedure, which is explained in IV.B, is executed where nodes N and K in the procedure correspond to the nodes E and V_5 in this sample scenario, respectively. Due to the Step 2 of *Network-Join* all nodes on the path from node V_5 to the root node V_0, (i.e. V_5, V_2, V_0) exchange their parent-child relationships. The resulting parent-child relationships are depicted in the subtree, rooted at V_5, of the combined network shown in Figure 5.b. Once the subtree rooted at V_5 is built, E shares a portion of its responsibility on the hashline with V_5. All descendants of V_5 share their responsibility on hashline in a similar manner, iteratively. One possible distribution of responsibilities on the hashline among the nodes of the new combined tree is depicted in Figure 5.c. Note that the resulting distribution may differ due to the order of children that a parent shares its responsibility with.



(a)



(b)



(c)

Fig. 5. Network-Join

VI. Future Work

As a future work, first of all, we would like to implement the system and make it practically work in a real environment, a Bluetooth scatternet of a set of pocket PCs, for instance. Secondly, some improvements would be beneficial in order to overcome some deficiencies that currently exist in our system. The most crucial of all is the scenario in which a node at the core of the network looses connection. In such a case, all information located at the nodes that are connected later to it, must be updated. Since interconnections are formed based on parent-child relationships leading to a tree-structure, the scenario is just like a case of a tree-shape topology in which a node close to the root loses connection and all its children must be updated in reconnection.

VII. Conclusion

In this paper, we proposed a peer-to-peer system that would enable file sharing in wireless ad-hoc networks.

Compared with central approaches, peer-to-peer systems are much more suitable for wireless networks for reliability and availability reasons. However, peer-to-peer systems that are used today are specialized for wire-line networks, namely the Internet. Although they introduce neat solutions for the file sharing problem in general, they make use of transport and network layers of the Internet. In other words, by default, they have the ability to look for a file at a specified address directly. Besides, each node can receive or send files from/to another one, directly. No doubt, the files do not reach directly to the end system but intermediate routing functionalities are handled by the underlying networking layers.

In wireless networks, this is not the case. Each node is only aware of its surrounding, i.e. the nodes in its range of communication. In this paper, we propose a system, which is able to find the location of a file in a WANET, if such a file exists in any node of the network, and which finds a way to bring the file from where it is stored to where it is needed. The only functionality that should be supported by the underlying WANET protocols is to handle communication between any two nodes that are in the range of each other. Routing of the files is handled by the peer-to-peer system itself. Our system also handles disconnections and reconnections that may happen as a result of mobility or due to problems in the wireless channel. As mentioned earlier, in reaching to a solution, we adapt techniques from peer-to-peer systems developed for wire-line networks as well as source routing techniques. By means of a "hashline" and forming a tree- structure based on the topology of the network, we are able both to distribute the index information to the nodes of the system and maintain the routing information between sources and destinations of files, which make file sharing possible in WANETs.

REFERENCES

[1] A. Klemm, C. Lindemann, and O. Waldhorst, "A Special-Purpose Peer-to-Peer File Sharing System for Mobile Ad Hoc Networks", *Proc. IEEE Semiannual Vehicular Technology Conference (VTC2003-Fall)*, Orlando, FL, October 2003.

[2] Bluetooth Special Interest Group. http://www.bluetooth.com.

[3] C. Lindemann, O. Waldhorst, "A Distributed Search Service for Peer-to-Peer File Sharing in Mobile Applications", *Proc. 2nd IEEE Conf. on Peer-to-Peer Computing (P2P 2002)*, Linköping, Schweden, 73-81, September 2002.

[4] D.B. Johnson and D.A. Maltz, "Dynamic Source Routing in Ad-Hoc Wireless Networks", *Proceedings of SIGCOMM'96*, ACM, California, USA, Aug., 1996.

[5] Elizabeth Royer, C.-K. Toh, A Review of Current Routing Protocols for Ad Hoc Mobile Wireless Networks, *IEEE Personal Communications*, April 1999.

[6] Fasttrack. http://www.fasttrack.nu.

[7] Gnutella. http://gnutella.wega.com.

[8] 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 the ACM SIGCOMM '01 Conference*, San Diego, California, Aug. 2001.

[9] M. Papadopouli and H. Schulzrinne, "Effects of Power Conservation, Wireless Coverage and Cooperation on Data Dissemination among Mobile Devices", *Proc. ACM Symp. On Mobile Ad Hoc Networking & Computing* (*MobiHoc 2001*), ACM, Long Beach, CA, 2001.

[10] Napster. http://www.napster.com.

[11] S. Ratnasamy, P. Francis, M. Handley, R. Karp and S. Shenker, "A Scalable Content-Addressable Network", *Proceedings of SIGCOMM'01*, ACM, California, USA, Aug., 2001.