A Cloud-based Wireless Mesh Network with Adaptive Data Storage Functions

S. T. Yang, Henry C. B. Chan, Patrick P. Lam, and Peter H. J. Chong

Abstract — **In recent years, there has been considerable interest in developing and using wireless mesh networks. Compared to wireless local area networks, wireless mesh networks are more flexible because mesh routers are interconnected by wireless links. In addition, they can be much easier to install and maintain, especially in environments where cables are difficult to install. Inspired by cloud computing, this paper presents an innovative cloud-based wireless mesh network with adaptive data storage functions for storing data dynamically and flexibly in a wireless environment. In particular, a person-based adaptive data management mechanism is presented.**

Index Terms — **wireless mesh network, cloud computing, data management**

I. Introduction

In a Wireless Mesh Network (WMN), Mesh Access Points (MAPs) provide wireless access to mobile terminals such as mobile phones and notebooks (i.e., similar to a wireless local area network). In addition, MAPs are interconnected wirelessly and function as routers for forwarding packets between each other in a collaborative manner [1]. Through a gateway, a WMN can also be connected to the Internet (i.e., providing Internet connectivity). This project focuses on IEEE 802.11-based WMNs [2]. IEEE 802.11, the widely used wireless LAN standard, can operate over different frequency bands such as 2.4, 5 and 60 GHz. Following the introduction of the first IEEE 802.11 standard, there are now a series of standards for different requirements. In particular, IEEE 802.11n operates over both 2.4 GHz and 5 GHz bands and supports a maximum data rate of 600 Mbit/s based on Multi-Input & Multi-Output (MIMO) antenna technology [3][4]. IEEE 802.11ac, the new generation of the IEEE 802.11 standard, can operate in the Gbit/s range using eight MIMO channels [5].

This research project is conducted through university-industry collaboration and is inspired by cloud computing. The aim is to study how to develop an innovative cloud-based WMN, particularly for adaptive data management purposes. Generally speaking, the aim of cloud computing is to allow the sharing of computing resources (e.g. storage, applications, and servers) based on a utility-based model [6]. There are three major kinds of cloud computing models: software as a service (SaaS), platform as a service (PaaS), and infrastructure as a service (IaaS). For deployment, four major methods can be employed namely: public cloud, private cloud, community cloud, and hybrid cloud. In our proposed cloud-based WMN, the focus is on providing dynamic data storage and adaptive data management services. Furthermore, consideration is given to user mobility or movement over a WMN (e.g., see [7]).

While research on cloud-based wireless mesh networks together with adaptive data management is relatively new, there are other related works on data caching for WMN. The following are some examples. In [8], Das, Pucha, and Hu presented a MeshCache system, a hop-by-hop cooperative caching system for WMN. The caching and fetching method of MeshCache enhances network throughput by taking into consideration user locations (e.g., to fetch data from the closest cache). In [9], Denko and Nkwe proposed a caching scheme for WMNs called CacheRescue. With CacheRescue, valid but evicted data can be stored in mesh routers based on expandable storage. In [10], Wu and Huang presented a cooperative caching system for WMNs. In that system, MAPs serve as user coordinators, and cell-based and network-based schemes are employed for cache placement purposes. In [11], Xu, Wu, Wu, and Cao studied a cooperative caching scheme for WMNs, in which the focus is on maintaining data consistency. With the aim of reducing the cost of communications and the number of messages lost, a hierarchical architecture is used and push-based and pull-based mechanisms are employed. There are also other related studies, such as the work of [12] and [13], which addresses routing issues in a WMN, and that of [14], which examines a scheduling problem for WMNs. [15]-[17] studied more fundamental and architectural issues related to WMNs. The aforementioned works provide useful references and valuable insights into the design of the proposed cloud-based WMNs.

The aim of this research project is to investigate adaptive data management mechanisms for a cloud-based WMN (i.e., how to store and manage data effectively and efficiently over a WMN to provide cloud computing-like services). This study is inspired by and/or based on caching mechanisms for WMNs (e.g., [8]-[11]), routing issues (e.g., [12]-[13]), scheduling methods (e.g., [14]), and architectural issues relating to WMNs (e.g., [15]-[17]).

Manuscript received January 8, 2015; revised January 28, 2015.

This work was supported by the Innovation and Technology Fund of the Hong Kong Special Administrative Region Government and by P2 Mobile Technologies (P2MT) Limited, Hong Kong.

S. T. Yang and Henry C. B. Chan are with the Department of Computing, The Hong Kong Polytechnic University, Hong Kong (e-mail: {cssyang, cshchan}@comp.polyu.edu.hk).

Patrick P. Lam is with P2 Mobile Technologies Limited, Hong Kong (e-mail: patricklam@p2mt.com).

Peter H. J. Chong is with the School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore (e-mail: ehjchong@ntu.edu.sg).

II. SYSTEM ARCHITECTURE

Fig. 1 shows the basic system architecture. Basically a wireless mesh network (WMN) consists of the following network components: mesh access points (MAPs), mesh clients and, possibly, network gateways. MAPs, also known as mesh routers, are responsible for communicating with mesh clients, and for routing and forwarding data over the WMN. Mesh clients are mobile devices for providing connections between users and the network. A mesh client usually connects to one MAP. When a mesh client moves from one MAP to another MAP, the connection should still be maintained. Network gateways are optional for a WMN. They provide connection to the Internet. To provide cloud-based services, each MAP can provide storage capability. The storage functionality can be implemented either through a general Linux virtual file system (VFS) on mesh routers, or through a light-weight distributed file system (e.g. GlusterFS, Ceph, etc.) to form a virtual file system on the cloud. Furthermore, MAPs can communicate with each other through conveying messages for data management purposes. Note that data can be forwarded and replicated over the network for adaptive data management purposes. Mesh clients can upload and download data using an effective mechanism.

Besides network communications service, the proposed cloud-based WMN can also support uploading/downloading functions (e.g., the uploading and downloading of personal data files to/from a WMN). Data can be stored in a MAP under each user account with proper access rights. For example, a user can store frequently used data files in a cloud-based WMN. As users can move around, the WMN should keep track of the users' locations so that data can be moved based on user locations (i.e., certain data can follow a user). Basically, when a user moves from one MAP to another MAP, some data files can be moved from the previous MAP to the current MAP, depending on certain requirements. This mechanism seeks to enhance the access or retrieval time for the user data (e.g., important files).

However moving data frequently, especially large data files, can be costly in terms of the consumption of network resources. Therefore it should be desirable to move certain data files only, taking into consideration various factors, such as user mobility, MAP storage, access probabilities, and so forth.

III. HARDWARE AND PROTOTYPE

We have been developing a prototype based on WMN routers provided by P2MT. Each WMN router consists of three major components: one mother board, up to four wireless network interface controllers, and antennas. The mother board is a single board computer equipped with a 600MHz dual core CPU, 256Mbytes of SDRAM memory, and 16Mbytes of flash system memory [18]. It has four mini-PCI sockets, which can support up to four wireless network interface controllers (either 802.11n 2.4GHz or 5GHz, or dual-band) to implement mesh router functionalities, and two Ethernet Ports for wired connections (for configuration or integration with other wired networks). There are also two RS232 serial ports, a micro SD flash expansion socket, and USB ports for expansion and other purposes.

For the operating system on WMN routers, a customized version of OpenWrt for the Laguna family (cns3xxx) is used. OpenWrt is a light-weight Linux distribution for embedded devices [19]. It is free, open-source, and community driven. Nowadays, OpenWrt is widely used by industrial vendors to facilitate product development and is also used by customers to customize device firmware in order to obtain more functions or enhance performance.

Openwrt provides package management functions so that any packages can be chosen to suit different requirements and development purposes [19]. Programs are cross-compiled through SDK or Toolchains built from the Laguna OpenWrt source together with all other necessary libraries needed for the development. With the package management features of OpenWrt, additional functionalities, such as routing management, a web server with a common gateway interface (CGI), databases, and so on, can easily be added to expand and customize the mesh routers.

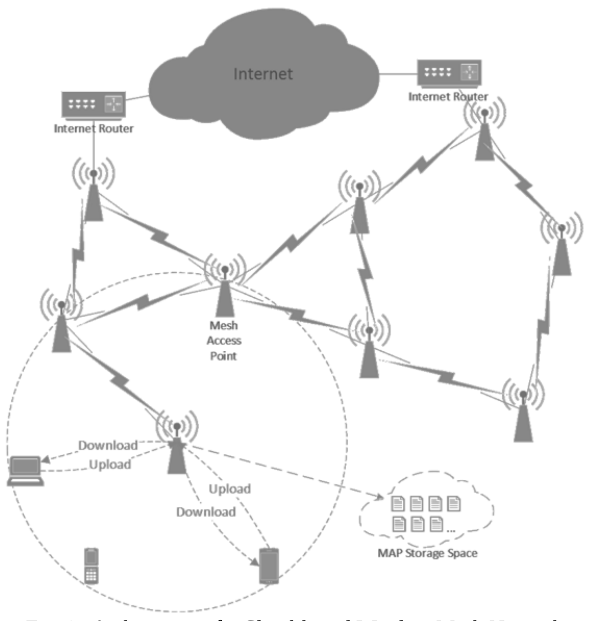
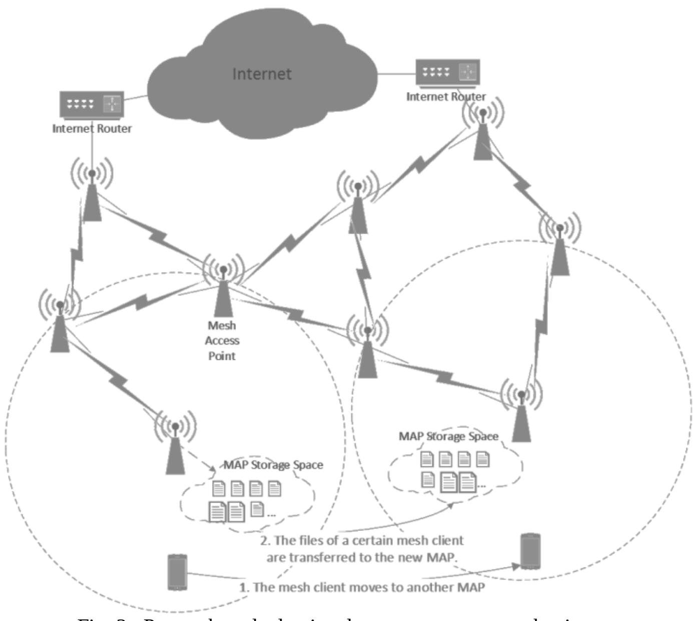


Fig. 1. Architecture of a Cloud-based Wireless Mesh Network


Internet
Internet Router
Internet Router
Mesh Access Point
MAP Storage Space
MAP Storage Space
2. The files of a certain mesh client are transferred to the new MAP.
1. The mesh client moves to another MAP
Fig. 2. Person-based adaptive data management mechanism

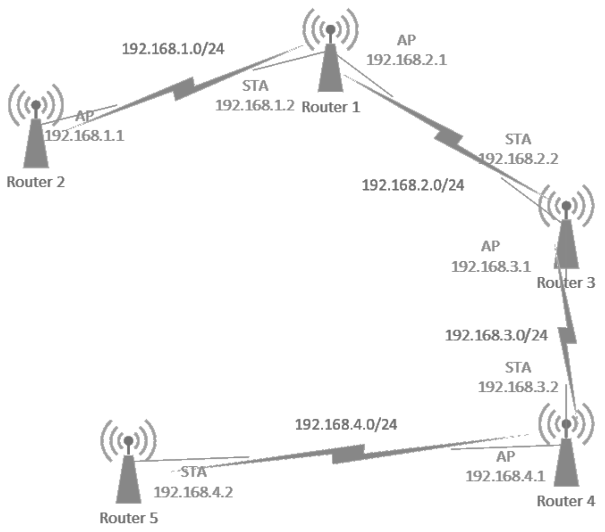


Fig. 3. One of the possible topologies of the WMN backbone

Every WMN router used to build the wireless mesh network is loaded with at least one 2.4GHz and two 5GHz wireless network interface controllers. The 2.4GHz interface is used to access wireless networks for mesh clients, and 5GHz interfaces are used to build the backbone of the mesh network.

There are three ways to establish interconnections among mesh routers: Access Points (AP) to Station (STA) mode, Ad-Hoc mode, and Wireless Distribution System (WDS) mode. In this study, the AP-STA mode is adopted, where the backbone interfaces (the two 5GHz ones) act as either AP or STA, and are connected as pairs. Because different subnet IP addresses are assigned to these paired connections, the Optimized Link State Routing (OLSR) protocol is used to rout packets among these interfaces to make different IP subnets accessible. Fig. 3 shows one of the possible topological configurations for the wireless mesh network. Five mesh routers are connected by four links in four different subnets (192.168.1-4.0/24). For mesh routers to be in two subnets simultaneously, two wireless network interfaces are used as mentioned above, and the routing tables are built with the tool orsld (OLSR daemon).

IV. FILE SYSTEM

A. *Storage function with Linux file system*

In the prototype, the normal Linux file system is used for the purpose of storing data. All data are stored in a particular directory on the mesh router. This section focuses on describing how the data of mesh clients are stored on the entire mesh network (i.e., in a distributed manner), and how the data management system can locate particular data. Since every mesh router in the mesh network is capable of storing data for mesh clients, the data management system needs to know where a particular file is stored. One method of locating it is to keep the index of the data stored so that the data management system can make a direct search of the index to obtain the location of the data. A database is used for keeping these indices. Fig. 4 is a Unified Modeling Language (UML) diagram of the database to illustrate the schema.

There are three types of main entities in the schema:

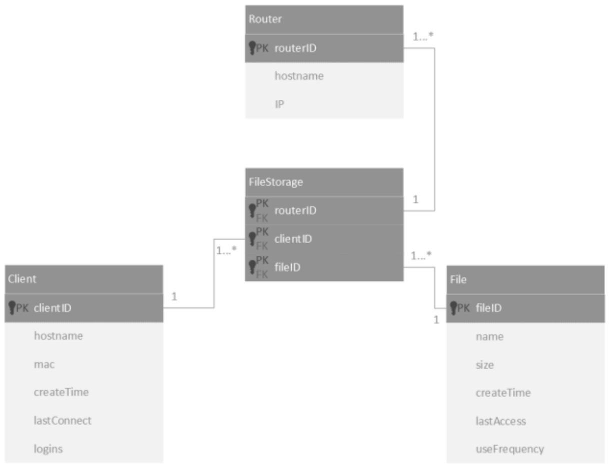
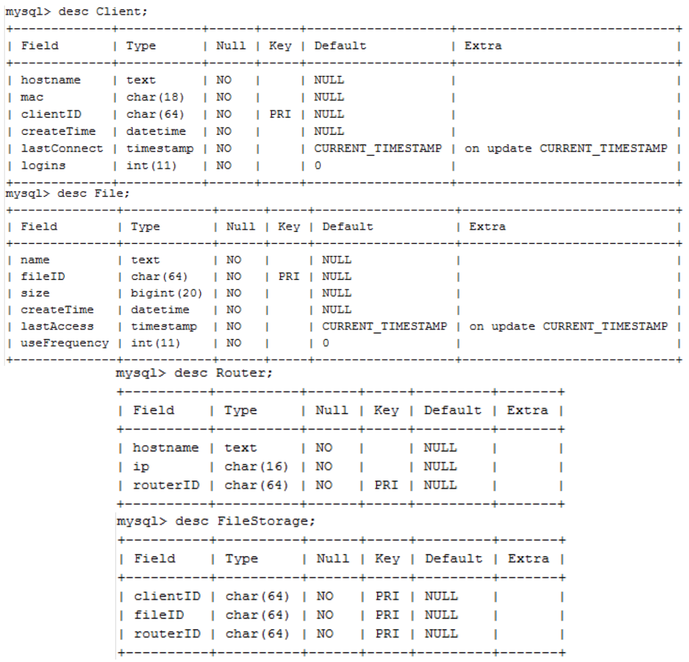


Fig. 4. UML diagram for data location index

Client, File, and Router. A unique ID is assigned to each entity to maintain its uniqueness and prevent unnecessary redundancies. The schema of this database follows the principles of "who, what, and where". We combine these three kinds of unique ID under a fourth table named *FileStorage*, which is the index that records which file (what) of the mesh client (who) is stored on which router (where).

Table Client records the basic information of mesh clients, such as the hostname, MAC address, and time of creation. Information such as the time of the last connection and the number of logins is also recorded for other management purposes. A mesh client may own multiple file storage indices because a client can own multiple files, and these files may also be duplicated on different routers. However, a file storage index can only belong to one client because it is uniquely identified by a foreign key *clientID*.

Table File records the basic information of a file, such as its name, size, and time of creation. Information such as the time of the last access and the frequency of use is also recorded for other management purposes. Similar to the relationship between mesh clients and the file storage index, a file may also own multiple file storage indices because a



```
mysql> desc Client;
+-------------+-----------+------+-----+-------------------+-----------------------------+
| Field       | Type      | Null | Key | Default           | Extra                       |
+-------------+-----------+------+-----+-------------------+-----------------------------+
| hostname    | text      | NO   |     | NULL              |                             |
| mac         | char(18)  | NO   |     | NULL              |                             |
| clientID    | char(64)  | NO   | PRI | NULL              |                             |
| createTime  | datetime  | NO   |     | NULL              |                             |
| lastConnect | timestamp | NO   |     | CURRENT_TIMESTAMP | on update CURRENT_TIMESTAMP |
| logins      | int(11)   | NO   |     | 0                 |                             |
+-------------+-----------+------+-----+-------------------+-----------------------------+
mysql> desc File;
+--------------+------------+------+-----+-------------------+-----------------------------+
| Field        | Type       | Null | Key | Default           | Extra                       |
+--------------+------------+------+-----+-------------------+-----------------------------+
| name         | text       | NO   |     | NULL              |                             |
| fileID       | char(64)   | NO   | PRI | NULL              |                             |
| size         | bigint(20) | NO   |     | NULL              |                             |
| createTime   | datetime   | NO   |     | NULL              |                             |
| lastAccess   | timestamp  | NO   |     | CURRENT_TIMESTAMP | on update CURRENT_TIMESTAMP |
| useFrequency | int(11)    | NO   |     | 0                 |                             |
+--------------+------------+------+-----+-------------------+-----------------------------+
mysql> desc Router;
+----------+----------+------+-----+---------+-------+
| Field    | Type     | Null | Key | Default | Extra |
+----------+----------+------+-----+---------+-------+
| hostname | text     | NO   |     | NULL    |       |
| ip       | char(16) | NO   |     | NULL    |       |
| routerID | char(64) | NO   | PRI | NULL    |       |
+----------+----------+------+-----+---------+-------+
mysql> desc FileStorage;
+----------+----------+------+-----+---------+-------+
| Field    | Type     | Null | Key | Default | Extra |
+----------+----------+------+-----+---------+-------+
| clientID | char(64) | NO   | PRI | NULL    |       |
| fileID   | char(64) | NO   | PRI | NULL    |       |
| routerID | char(64) | NO   | PRI | NULL    |       |
+----------+----------+------+-----+---------+-------+
```

Fig. 5. Schema of tables implemented for storage functionality

file can be shared by multiple mesh clients, or be duplicated on different routers. A file storage index only belongs to one file because it is uniquely identified by a foreign key *fileID*.

Table Router records the basic information of a mesh router, such as its hostname and IP address. A mesh router may own multiple file storage indices because it can accommodate multiple files stored on it. However a file storage index only belongs to one mesh router because it is uniquely identified by a foreign key *routerID*.

The file storage index links mesh clients, files, and mesh routers together and uniquely identifies the ownerships and location information. Hence, from any side of this triangular relationship, the other two sides can easily be accessed when an enquiry is made for information.

B. Storage function with a distributed file system

Basic requirements can be met by using a Linux file system and by ensuring that the system can operate properly in an experimental environment; however, some problems may be encountered in the practical use of the system. A critical problem is the Single Point of Failure (SPoF). For example, if one of mesh routers encounters network problems or experiences system failure, then the data of the mesh clients that were stored on this router can no longer be accessed; likewise, if the mesh router hosting the file storage index database encounters network problems or experiences system failure, then the whole data management system will fail. To avoid such fatal situations, it is necessary to have a replicate copy of the data and file storage indices of the mesh clients. This will improve the reliability, accessibility, and fault-tolerance of the data management system.

Adopting a distributed file system is a good solution for such requirements. Two important features of a distributed file system are access transparency and replication transparency, which means that what mesh clients see in the distributed file system is exactly the same as what they see in a local file system, so they will not regard the replications made by the distributed file system as fail-safes.

Currently, there are various distributed file systems. They are designed for different goals, different system scales, and different considerations. In the future, we shall study the use of a distributed file system for an effective adaptive data management system.

V. Person-based Adaptive Data Management Mechanism

In this section, we present a person-based adaptive data management mechanism, which provides upload/download functions for mesh clients, and adaptively moves files along with the movements of the owner to enhance access speed. Hence, the development of this section has been divided into two parts: (i) a web User Interface (UI) to let mesh clients upload and manage personal files; (ii) an agent program running on mesh routers to monitor the mobility of mesh clients, and to adaptively decide whether to move the files and on which mesh router the files should be placed.

A. Web UI for file uploading/management

The web UI is responsible for handling file uploading/ management operations, similar to current commercially available online drive/sync services. The following is a brief description of some of the ways in which the web UI can be used, as examined in this study:

1. A mesh client accesses the web UI, and the web UI prompts the client by asking for the client to log in, or to go directly to the main page if the client is already logged in;
2. The mesh client logs in through an account/device, and the web UI displays the main page containing all of the files belonging to the logged on mesh client;
3. The mesh client uploads new file(s), and the web UI stores the uploaded files in file system, inserts information on the file, and builds a file storage index in the database;
4. The mesh client does management operations (e.g., renames, deletes, downloads, etc.) on existing file(s); the web UI responds accordingly, and updates/deletes the file information/file storage index to/from the database.

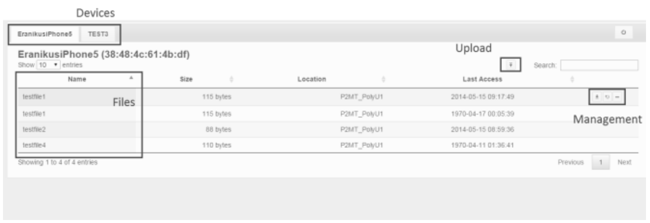
Note that the web UI does not need to be concerned about which type of storage file system is used (the normal Linux file system or the distributed file system) because they are separate. If the normal Linux file system is used, the placement of the file will be handled by the agent program, which will be described in next section; if the altered distributed file system is used, the storage location will be determined by the altered distributed file system.

The front end of this web UI is developed using the *JQuery* library with multiple plugins, such as the *JQuery-UI*, *JQuery DataTables*, *JQuery Download*, and *JQuery Fileupload*. The back end of the web UI is written using *PHP*, and an *UploadHandler* plugin is adopted to achieve the drag-and-upload function. *MySQL* is used as the database to keep the file information and file storage indices.

The web UI is deployed on one of the mesh routers, normally the network gateway of the wireless mesh network. The index database is placed on another mesh router that, for reasons of security, cannot be directly accessed from an external network environment.

B. Agent program for monitoring the mobility of clients and making decisions on the movement of files

The agent program runs on all mesh routers with storage functionality inside the mesh network. This agent program is responsible for three major tasks: monitoring the movements of mesh clients, making decisions on the movement of files, and transferring files to a target router if needed. Instead of having a central router conduct all computations, agent programs running on all mesh routers work in a decentralized way to share the work load. Each mesh router retrieves and computes information individually, and exchanges information with fellow routers if needed. Communications among routers are carried out through UDP messaging. The agent program is written in C language, with



EranikusiPhone5 (38:48:4c:61:4b:df)

Name	Size	Location	Last Access
testfile1	115 bytes	P2MT_PolyU1	2014-05-15 09:17:49
testfile1	115 bytes	P2MT_PolyU1	1970-04-17 00:05:39
testfile2	88 bytes	P2MT_PolyU1	2014-05-15 08:59:36
testfile4	110 bytes	P2MT_PolyU1	1970-04-11 01:36:41

Showing 1 to 4 of 4 entries

Fig. 6. Web UI for a person-based adaptive data management system

assistance from some shell scripts and external libraries (e.g., libmysqlclient, libssh2, libgcrypt, etc.)

In order to make files follow along with a mesh client's movements, the data management system needs to be aware of the mobility behaviour of the mesh client. A mesh router should make a file movement decision when a handoff operation happens. Hence, the agent needs to monitor connections from mesh clients, and trigger a decision on the movement of a file when a new connection is built. Since OpenWrt is a Linux distribution for embedded devices, especially for network routers, shell scripts can be used to help retrieve information on connected devices instead of collecting information from scratch. The following piece of shell script code returns the IP address, hostname, and MAC address of the current connected devices:

```
for interface in `iw dev | grep Interface | cut -f 2 -s -d' '`
do
maclist=`iw dev $interface station dump | grep Station | cut -f 2 -s -d' '`
  for mac in $maclist
  do
    ip='UNKN'
    host=''
    ip=`cat /tmp/dhcp.leases | cut -f 2,3,4 -s -d' ' | grep $mac | cut -f 2 -s -
d' '`
    host=`cat /tmp/dhcp.leases | cut -f 2,3,4 -s -d' ' | grep $mac | cut -f 3 -s
-d' '`
    echo  $ip,$host,$mac
  done
done
```

The agent program runs this piece of shell script periodically to retrieve information on currently connected mesh clients, and compare this information to the past client list kept from the last timeslot. If the current client list contains a new member that does not appear in the past client list, then this client is a newly connected mesh client, and the process of making a file movement decision should be triggered. If a member from the past client list does not appear in the new client list, this means that this mesh client has disconnected from this mesh router. The client may have been disconnected from the wireless mesh network, or have moved to another mesh router.

The following is the pseudo-code of the algorithm, which monitors movements of mesh clients. The algorithm is run periodically:

```
newClientList := getClientList shell script

for each client in pastClientList
    // do nothing for stationary clients
    if client exists in newClientList
        remove client from newClientList
    end if

    // remove clients if they leave
    if client not exists in newClientList
        call clientDisconnected(client)
        remove client from pastClientList
    end if
end for

// the remaining clients in newClientList are new clients
for each client in newClientList
    add client to pastClientList
    call clientConnected(client)     // do file movement decision making
end for
```

The decision to move a file is made by calculating a score called the file transfer priority. During the process of making a decision to transfer a file, the file transfer priority value of all of the files belonging to a mesh client will be calculated and sorted in order to determine whether or not to transfer a file. As an example, we consider that the file transfer priority *p* is related to:

f = use frequency
s = file size
t = time of last access

For this illustrative example, we assume that frequency *f* is directly proportional to *p*, and that file size *s* and time to last access *t* are inversely proportional to *p*:

$$p \propto \frac{f}{st}$$

However, the range of file size *f* and time to last access *t* may have large values, so they need to be normalized within a reasonable range. According to the properties of the logarithm function, the log value will increase much more slowly when the actual value becomes larger. Therefore, we use the logarithm function for normalization purposes.

Since the file size is represented by bytes, the value of $\log_2 s$ will be 10, 20, 30, and 40 for 1KB (2^{10} bytes), 1MB (2^{20} bytes), 1GB (2^{30} bytes), and 1TB (2^{40} bytes), respectively. A UNIX timestamp is used to represent the time to the last access *t*, with *t* being equal to the current timestamp minus the last access timestamp. We can also use a logarithm to control the range of value *t* (e.g., log *t* will be around 16, 21, or 25, if the file has not been accessed for one day, one month, or one year, respectively);

Now the equation becomes:

$$p = \frac{f}{\log_2 st}$$

In most situations, the value of $\log_2 st$ is less than a hundred, so the scale is much more acceptable for a priority value. Note that the above is just an example – other policies can also be used.

After calculating the priority values of all of the files, the agent program will decide whether or not to transfer files to the target mesh router. Other information that may be involved into this decision-making process (e.g., the possible transfer cost, the traffic of the mesh network, etc.) will be retrieved through messaging communications among mesh routers.

If the agent program decides to transfer a particular file from another mesh router to itself, the file transfer operation will be performed. The method of transferring a file depends on the type of storage system. If a normal Linux file system

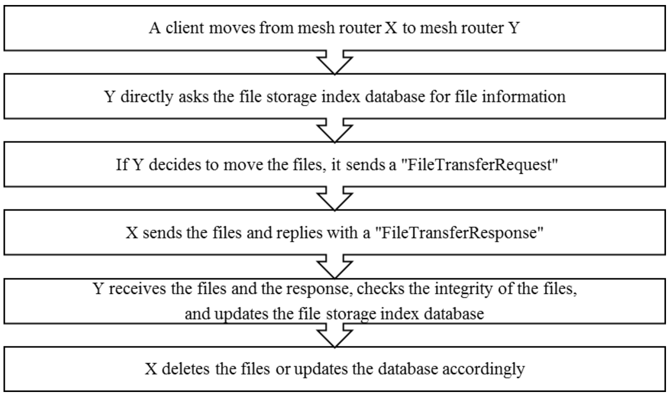

A client moves from mesh router X to mesh router Y
Y directly asks the file storage index database for file information
If Y decides to move the files, it sends a "FileTransferRequest"
X sends the files and replies with a "FileTransferResponse"
Y receives the files and the response, checks the integrity of the files, and updates the file storage index database
X deletes the files or updates the database accordingly

Fig. 7. Work flow of the file transfer operation

is used for storage functionality, the current mesh router will send a *FileTransferRequest* to the mesh router that holds the desired file. After the target router receives the request, the file will be directly sent to the requesting mesh router through Secure Copy (SCP) with a *FileTransferResponse*. Next, the requesting mesh router receives the file and the response, checks the integrity of the received file, updates information in the file storage index database, and sends an acknowledgement to the sender with a *FileTransferSucceed*. The sender will decide whether to remove the file from its local storage or to keep it, depending on whether there are other ownership relations existing between other mesh clients and this file. If the file is to be kept, the respective information in the file storage index database will be updated accordingly. Fig. 7 shows the work flow of the file transfer operation.

VI. Conclusions and Future Work

In conclusion, we have presented a cloud-based wireless mesh network with cloud storage functionality, with an accessible web UI for uploading/managing files, and an agent program to monitor the movements of mesh clients and adaptively move files among mesh routers to reduce access costs. The person-based adaptive data management mechanism can improve data access efficiency for mesh clients. In the future, apart from further developing the person-based data management mechanism, we also plan to investigate a group-based data management mechanism, which considers data to be shared by multiple mesh clients.

References

[1] I. F. Akyildiz and X. Wang, "A survey on wireless mesh networks", *IEEE Communications Magazine*, vol. 43, no. 9, pp. S23-S30, Sep. 2005.

[2] A. Raniwala and T. Chiueh, "Architecture and algorithms for an IEEE 802.11-based multi-channel wireless mesh network", *Proc. 24th Annual Joint Conference of the IEEE Computer and Communications Societies*, vol. 3, pp. 2223-2234, Mar. 2005.

[3] E. Perahia and R. Stacey, *Next Generation Wireless LANs: 802.11n and 802.11ac*, 2nd edition, Cambridge University Press, 2013.

[4] I. Ho, P. Lam, and P. Chong, "Harnessing the high bandwidth of multi-radio multi-channel 802.11n mesh networks", *IEEE Transactions on Mobile Computing*, pp. 1-10, Jan. 2013.

[5] L. Ward, *802.11ac Technology Introduction White Paper*, Mar. 2012. Available: http://cdn.rohde-schwarz.com/dl_downloads/ dl_application/application_notes/1ma192/1MA192_7e.pdf

[6] P. Mell and T. Grance, "The NIST definition of cloud computing - recommendations of the National Institute of Standards and Technology", *National Institute of Standards and Technology, U.S. Department of Commerce*, Sep. 2011. Available: http://csrc.nist.gov/ publications/nistpubs/800-145/SP800-145.pdf

[7] F. Bai and A. Helmy, "A survey on mobility models in wireless ad-hoc networks", *Wireless Ad-Hoc Networks, Kluwer Academic*, vol. 1, pp. 1-30, 2006.

[8] S. M. Das, H. Pucha, and Y. C. Hu, "Hop-by-hop cooperative caching in wireless mesh networks: a design and implementation study", 2006. Available: http://www.cl.cam.ac.uk/research/srg/netos/sla/ mobileman/d15/papers06/266_2_realman.pdf

[9] M. K. Denko, T. Nkwe, and M. S. Obaidat, "Efficient cooperative caching with improved performance in wireless mesh networks", *Proc. 2010 IEEE International Conference on Communications*, pp. 1-5, May 2010.

[10] W. Wu and Y. Huang, "Hierarchical cooperative data caching for wireless mesh networks", *Proc. 2010 IEEE/IFIP International Conference on Embedded and Ubiquitous Computing*, pp. 289-296, Dec. 2010.

[11] W. Xu, W. Wu, H. Wu, and J. Cao, "A cooperative approach to cache consistency maintenance in wireless mesh networks", *Proc. 2011 IEEE 17th International Conference on Parallel and Distributed Systems*, pp. 512-519, Dec. 2011.

[12] R. Draves, J. Padhye, and B. Zill, "Routing in multi-radio, multi-hop wireless mesh networks", *Proc. 10th annual international conference on Mobile computing and networking*, pp. 114-128, Jul. 2004.

[13] S. Miskovic and E. W. Knightly, "Routing primitives for wireless mesh networks: design, analysis and experiments", *Proc. IEEE Infocom*, pp. 1-9, Mar. 2010.

[14] S. Cheng, P. Lin, D. Huang, and S. Yang, "A study on distributed/centralized scheduling for wireless mesh network", *Proc. 2006 International Conference on Wireless Communications and Mobile Computing*, pp. 599-604, Jul. 2006.

[15] S. Sen and B. Raman, "Long distance wireless mesh network planning: problem formulation and solution", *Proc. 16th International Conference on World Wide Web*, pp. 893-902, May. 2007.

[16] J. Jun and M. L. Sichitiu, "The nominal capacity of wireless mesh networks", *IEEE Wireless Communications*, vol. 10, no. 5, pp. 8-14, Oct. 2003.

[17] V. Navda, A. Kashyap, and S. R. Das, "Design and evaluation of iMesh: an infrastructure-mode wireless mesh network", *Sixth IEEE International Symposium on a World of Wireless Mobile and Multimedia Networks*, pp. 164-179, Jun. 2005.

[18] Laguna GW2388-4 Single Board Computer, Available: http://www.gateworks.com/prod uct/item/laguna-gw2388-4-network-processor.

[19] OpenWrt, https://openwrt.org/.