PosCFS+: A Self-Managed File Service in Personal Area Network

Woojoong Lee, Shine Kim, and Chanik Park

Wearable computers consisting of various small devices such as smart phones, digital cameras, MP3 players and specialized I/O devices in personal area networks will play an important role in future ubiquitous computing. In this environment, accessing user data is quite complex due to the dynamic and heterogeneous characteristics of the underlying networks. Moreover, since the amount of user data increases rapidly, automatic data backup management is also critical. To overcome these challenges, several studies have been conducted including our previously proposed file service system, PosCFS, which could be adapted to the requirements with a virtualization technique allowing per-user global namespace for managing and accessing data stored on physical storage spaces detected in PAN. In this paper, we present a smart file service framework, PosCFS+ which is an improved and extended version of our previous work. Performance improvement is made possible by redesigning the metadata management scheme based on database and keywords rather than ontology. In addition, the automatic data replication management is newly designed based on the OSD protocol.

Keywords: Ubiquitous computing, file service, semantic metadata, data replication, UPnP, WebDAV, OSD.

Manuscript received Oct. 01, 2006; revised Mar. 22, 2007.

This research was supported by the MIC (Ministry of Information and Communication), Korea, under the ITRC (Information Technology Research Center) support program supervised by the IITA (Institute of Information Technology Assessment), grant number IITA-2006-C1090-0603-0045. This research was also supported in part by the wearable personal station project from the MIC and the BK21 program of the Ministry of Education of Korea.

Woojoong Lee (phone: + 82 54 279 5668, email: wjlee@postech.ac.kr), Shine Kim (email: postshin@postech.ac.kr), and Chanik Park (email: cipark@postech.ac.kr) are with the Department of Computer Science and Engineering, POSTECH, Pohang, Korea.

I. Introduction

Nowadays, users may carry several personal devices, such as PDAs, notebooks, MP3 players, digital cameras, and smart phones, which are each equipped with storage space well above 2 GB. With the recent advance of flash memory and small form factor hard disk technologies, each personal device will be expected to carry up to 1 TB in 2010 [1].

Wearable computers are gaining importance in embedded ubiquitous computing, where all the personal devices are connected by personal area networks (PANs). In order for a user to manage and access all the data stored on personal devices, the underlying file service layer has to deal with the dynamic routing in the underlying PAN and build up a new directory hierarchy for all the gathered data. Moreover, since wireless networks such as Bluetooth, Zigbee, UWB, and 802.11 are used for PANs, the supporting dynamic configuration of wearable computers is also important.

In our previous work regarding PosCFS [2], [3], we addressed the main functionalities required for file services in ubiquitous computing and presented a new smart file service which could be adapted to the requirements with a virtualization technique which provides per-user global namespace for managing and accessing data stored on physical storage spaces detected in PAN. As a by-product of virtualization, we could make the system include a basic context-awareness concept in the file service. That is, it could provide a special ability, retrieving files which correspond to the current context for context-aware applications. The file service was implemented using the UPnP protocol [4] to automatically build up a virtualized space over all personal devices in a PAN and also by using WebDAV [5] for file I/O.

Storage virtualization in PosCFS was represented by two interfaces. One was a WebDAV-based storage interface and the

other was the virtual directory, which is the key concept for per-user global namespace and supporting context-awareness. It is dynamically generated by matching file metadata maintained by the file service with some conditions, such as the user's profile and context information. For more details, please refer to [1], [2] and section III. However, in our previous implementation, the user's profile and file metadata are organized to the ontology language [6], [7], but this turns out to be inefficient on small embedded devices. Moreover, the system needs to be extended to support automatic data backup management.

To overcome these challenges, we present a new file service framework, PosCFS+, which is an improved and extended version of our previous work in terms of self-manageability. Using database management like SQLite [8] instead of ontology technology, we redesigned the metadata management scheme to improve performance. In addition, automatic data backup management is newly designed with a simple replica placement algorithm based on the object-based storage device (OSD) protocol [9]. Therefore, we have two separate software layers in PosCFS+: the data access layer based on the WebDAV protocol and the data replication layer based on the OSD protocol. Having the data replication layer separate from the data access layer enables PosCFS+ to easily interoperate with other non-PosCFS+ based devices such as a backup server or a home server gateway.

The remainder of this paper is organized as follows. Section II gives a brief outline of related works. In section III, we present details of the PosCFS+ architecture for both the data access layer for better performance and the data replication framework based on OSD protocol. Experimental evaluations are given in section IV. Finally, section V presents our conclusions and future works.

II. Related Works

In this section, we describe the state of the art of smart file services which provide an intelligent and integrated framework for file management in PANs.

The GAIA context-aware file system [10], proposed by the System Software Research Group of the University of Illinois, was the first approach which tried to adapt a context-aware concept to a file system in Active Space, an intelligent PAN. It provides a novel concept as a well-defined middleware component and is applicable to diverse computing environments. However, it is not suitable for the wearable computing environment due to its centralized file system construction mechanism; there must be one mount server for constructing a shared space between devices, and there is a lack of representations for describing file metadata.

OmniStore [11], proposed by the University of Thessaly, not only tries to integrate portable and backend storage in a PAN, but also exhibits self-organizing behavior through spontaneous device collaboration. Moreover, the system provides transparent remote file access, automated file metadata annotation, and a simple data replication framework. Despite the innovative features, the system is limited in terms of interoperability because it is implemented with its own defined discovery and file I/O protocols rather than standard protocols.

EnsemBlue [12], proposed by the University of Michigan, provides a global namespace shared by all devices in a PAN, which is maintained by a centralized file server. It also utilizes energy efficiency and file I/O performance. These features, inherited from BlueFS [13], were also developed by the same authors. However, it only provides a static global shared space, a global file tree, among devices which belong to users of the same group, such as a family or an organization.

We briefly summarize the characteristics of the PosCFS+ and these systems in Table 1 with some criteria, such as file service construction mechanisms, file metadata management schemes for intelligent file browsing or accessing, namespace management for shared space, automatic data replication or backup, and so on.

Regarding data replication frameworks in mobile ad-hoc networks or PANs, several studies have been conducted [14], [15], [16]. There are various issues related to replica relocation, consistency management, location management, and so on. Oasis [17], developed by Intel Research, provides an asymmetric peer-to-peer data replication framework tailored to the following requirements: availability, manageability, and programmability in a PAN. Oasis addresses these requirements by employing a peer-to-peer network of weighted replicas and performing background self-tuning. OmniStore [11] also provides a simple replication framework for PANs as we mentioned. It was implemented based on a simple backup policy with a base station.

In our work, we present a concept called per-user global name space which is supported by virtual directories. It provides a semantic namespace inspired by previous studies. In order to support semantic namespace in a file service, files can be indexed by their semantic metadata and accessed by the information. SFS [18], LISFS [19], CONNECTIONS [20], and LiFS [21] address the issues of how to generate semantic information and how to index and access files with the information.

III. PosCFS+ Architecture

In this section, we present the overall architecture of PosCFS+, which is shown in Fig. 1. PosCFS+ has two separate management layers: the data access layer and the data replication layer. By having two separate layers, greater extensibility and interoperability in data management, namely,

Table 1. Characteristic of GAIA, OmniStore, EnsemBlue and PosCFS+.

	GAIA	OmniStore	EnsemBlue	PosCFS+
File system construction method	Centralized mount server	Distributed, P2P	Centralized server	Distributed, P2P
File metadata management	Keyword based	Keyword based	Not support	Keyword based
Automated metadata annotation	Restricted	Support	Not support	Restricted (extracting from file itself)
Namespace management for shared space	Static and global namespace in PAN (directory-based)	Flat model	Static and global namespace (directory based)	Per-user global namespace-based user profile in PAN
Context-awareness support	Can provide files corresponding to context information	Flat model, Can provide files corresponding to context information	Not support	Virtual directory corresponding to context information
Replication framework	Not support	Backup policy with base station	Not support	Adaptive, Considering device and target availability of data
Energy efficiency and performance	Not considered	Not considered	Considered	Not considered

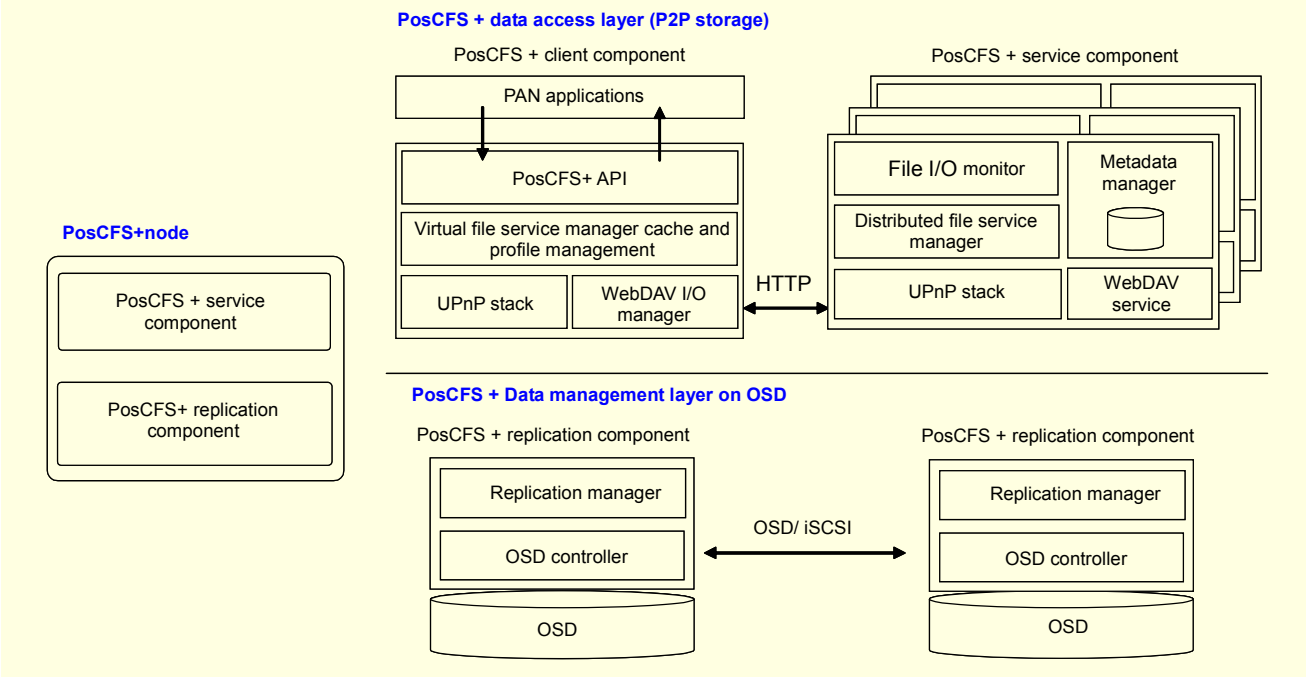

Fig. 1. Two-layered architecture of PosCFS+.

automated backup and replication, are achieved.

The data access layer is constructed using a peer-to-peer structure with UPnP and WebDAV protocols. The role of this layer is to provide easy access to user data based on semantic metadata of files in the PAN. Easy access to user data is supported by storage virtualization, which includes the concepts of semantic file addressing and the virtual directory.

The data replication layer is based on the object-based storage device (OSD) protocol and is in charge of automated data backup and replication considering the availability parameter of each device and the target availability pre-assigned to replication units by users. More details of this layer are discussed in the following subsection.

1. Data Access Layer in PosCFS+

A. Semantic File Addressing

Most of the existing file systems have a namespace, such as

a directory structure which represents file addresses based on their own internal logic. However, the structure is rigid and implicitly assigned by users. Moreover, since the amount of user data increases rapidly, users have difficulty managing and accessing their files.

Some studies [18], [19], [21] have been conducted to overcome these challenges. They define not only the traditional directory structure but also another namespace for accessing files using semantic information or the metadata on a local file system. However, they are limited in that they cannot be expanded to PANs. Thus, there is a need for a new namespace management technique with a virtualization technique which can be applied to the dynamic and heterogeneous network.

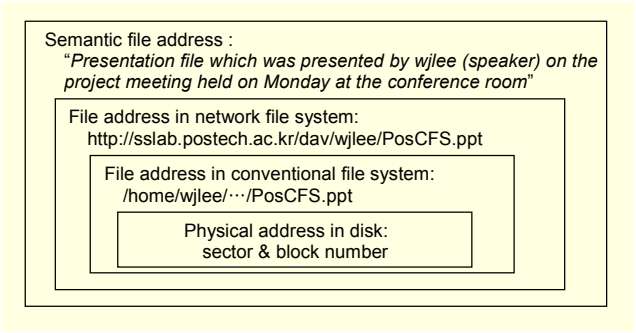

Fig. 2. Concept of semantic file address.

B. Conceptual View of Virtual Storage Space in PosCFS+ and WebDAV-Protocol-Based File I/O

PosCFS+ nodes use UPnP [4] to discover and control one another in a peer-to-peer manner. The WebDAV [5] protocol is used for file I/O in the system. This protocol is an extended version of HTTP, which defines some extended methods for supporting file I/O on a traditional network file system, such as file writing, directory and file property management and locking, as well as the basic methods defined as HTTP, GET, and POST, which are methods for file reading. By using these global standards, we have been able to implement a platform-independent and self-constructible file service.

In PosCFS+, there are two concepts of storage space: a physical view, based on store and storage concepts; and a logical view based on the virtual directory. (See Figs. 3 and 4.)

The store interface, an instance of virtualization in PAN, is dynamically and automatically constructed and managed by the UPnP protocol. The storage interface in the store provides an abstraction of WebDAV-based storage.

The virtual directory is a basic unit of semantic file addressing in our system. As shown in Fig. 4, the virtual directory is dynamically constructed by matching file metadata maintained by the file service with some conditions, such as user query, profile, and context information. The details of the construction mechanism are described in the next section.

C. Metadata Management and Querying in PosCFS+

Instead of using the ontology technique to describe a semantic file address, we use a simple keyword-based query and an SQLite [8] based metadata repository in PosCFS+ to enhance the query performance and alleviate metadata management overhead. We will describe in detail a new design for semantic file addressing in the following section.

1) Metadata repository

The roles of the metadata repository managed by the metadata management module in the PosCFS+ service component are to store and manage the semantic metadata of files. For that purpose, the repository stores two kinds of databases whose schema are shown in Fig. 5. One is for file metadata and the other is for user profiles.

The metadata database is managed by a background process called the file I/O monitor (see Fig. 1). The file I/O monitor process carries out the monitoring and logging of file I/Os and then updates the metadata database with extracted information from the file tags and the underlying file system. Most file

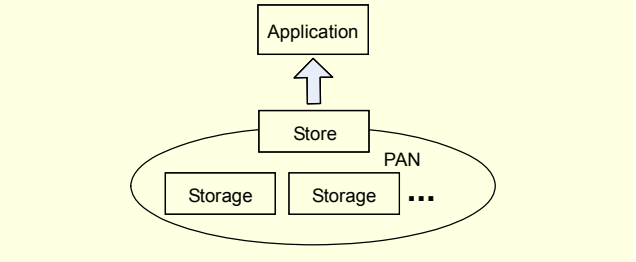

Fig. 3. Store and storage spaces.

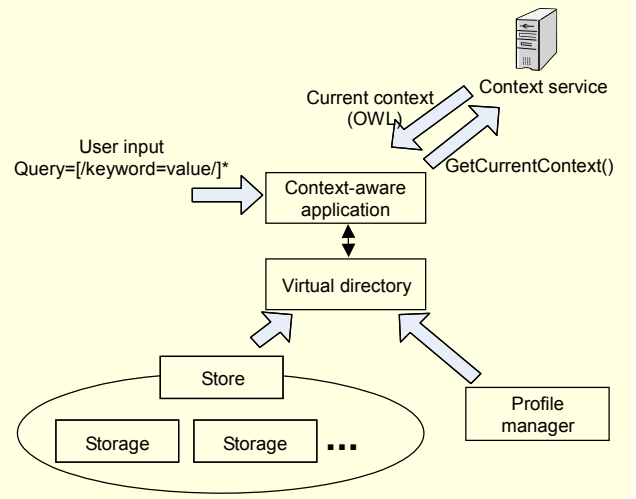

Fig. 4. Virtual directory.

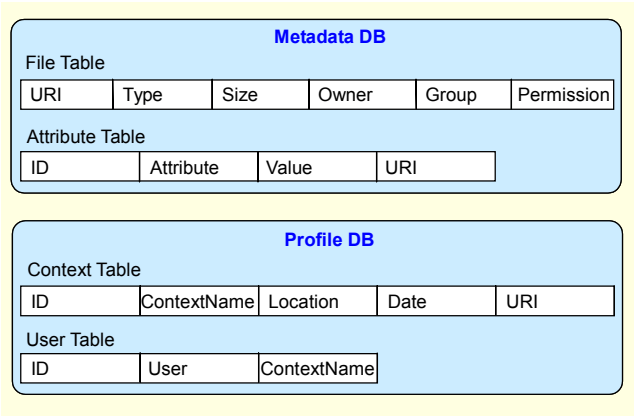

Fig. 5. DBs and tables in metadata repository. URI (universal resource identifier) represents a file reference used in WebDAV protocol.

formats have their own metadata fields. For instance, in the case of the MP3 format, the file I/O monitor extracts some semantic information, such as "Artist" and "Genre" from the ID3 tag of the format. The pdf or ps format has some tags for "Author," "Title," "Subject," and so on. For extensibility, an attribute table has been designed, whose internal representation is similar to the RDF triple structure [6], resulting in no limitation of the number of attribute-value pairs attached to a file resource.

The Profile database stores the user profile which consists of two types: named context and unnamed context. The named context represents explicit details of the user's schedule or events. For instance, "Project Meeting," "Room 423 in PIRL building," and "2007-02-01" can be used as field values representing a named context in the context table. On the other hand, the unnamed context represents a situation defined by a location and a point of time. This information may be useful for maintaining the user's preference of files in a given situation. For example, we can maintain information such as which music files have been played at home by a user.

2) Virtual directory construction

Virtual directories are dynamically created at each service node, and then merged into a file tree at a client node requesting with a query that is generated from a user input profile or current context information. How to create a virtual directory can be specified using relational algebra. A virtual directory, its sub-virtual directory, and files included in the directory can be obtained as shown in the algebra, represented by V and F_v, respectively. This process tries to match a keyword either explicitly given by the user or by a special type of user profile (view preference) to build a per-user global namespace in the PAN and file metadata which is maintained by the file and attribute tables. Due to the RDF-like structure of the attribute table, we can obtain the results, V and F_v, from join operations with tables that are obtained by selection with each keyword; V_{ctx} and F_{Vctx} can be simply obtained by selection of each keyword using the context table.

The namespace, in other words, is a virtual directory tree constructed using view preference. For example, if a user wants to browse files on his/her handheld devices using a view preference, (for example, a categorization sequence represented by "Genre-Artist-Album") then it is dynamically organized accordingly.

In our file service, view preference is maintained by the virtual file service manager module in the PosCFS+ client component (see Fig. 1). It contains categorization rules for each file type, such as documents, presentations, and types of images.

$$\mathbb{V} := \pi_{value}((\sigma_{A.attr=k}(A) \bowtie_{uri} \sigma_{A_{cond}(0)}(A) \bowtie_{uri} \ldots \sigma_{A_{cond}(n-1)}(A)) \bowtie_{uri} \sigma_{F_{cond}}(F)),$$

$$\mathbb{F}_V := \pi_{uri}((\sigma_{A.attr=k \wedge A.value=value}(A) \bowtie_{uri} \sigma_{A_{cond}(0)}(A) \bowtie_{uri} \ldots \sigma_{A_{cond}(n-1)}(A)) \bowtie_{uri} \sigma_{F_{cond}}(F)),$$

$$\mathbb{V}_{ctx} := \pi_{k_{ctx}}(\sigma_{C_{cond}(0) \wedge C_{cond}(1) \wedge \ldots C_{cond}(n-1)}(C)),$$

$$\mathbb{F}_{Vctx} := \pi_{uri}(\sigma_{C_{cond}(0) \wedge C_{cond}(1) \wedge \ldots C_{cond}(n-1)}(C)),$$

where n : size of list,
F: File table,
A: Attribute table,
C: Context table,
F_{cond}: List of field name and value pairs in F,
A_{cond}: List of attribute-value pairs in A,
C_{cond}: List of field name and value pairs in C,
k: Keyword for virtual directory,
k_{ctx}: Context keyword for virtual directory,
$\mathbb{V}$: a set of virtual directories,
$\mathbb{V}_{ctx}$: a set of virtual directories corresponding to a context query,
$\mathbb{F}_V$: a set of files in $\mathbb{V}$,
$\mathbb{F}_{Vctx}$: a set of files in $\mathbb{V}_{ctx}$.

2. Data Management Layer in PosCFS+

A. Overall Architecture of Data Replication Layer

In our system, the data management layer is implemented using the OSD protocol for data management and replication and the UPnP protocol to discover each replication component. This layer is a perfectly separated module with an upper layer, the data access layer. It is designed for a private PAN (P-PAN), which is a private network between devices belonging to a user or a group of users. The separate design of the data access and replication layers enables the extensibility and interoperability of PosCFS+ with other non-

PosCFS+ based devices such as a backup server or a home server system in a PAN. However, for cooperation with the upper layer, we apply a "home node" concept for each replication unit in our system; a home node contains original data and replication policies. In our implementation, every file write request from the upper layer can be delivered to the home node only, and if the home node fails, then a new home node will be elected from its replicas of the replication unit while the read requests for data can be performed with any replicated data. We will present an in-depth description of this mechanism later.

Since PosCFS+ uses the OSD protocol for data replication and replica management, it is possible to take advantage of the main features of an OSD-based device. An OSD-based device has the following advantageous characteristics [22]:

- Objects contain both data and meta-data.
- It allows fine-grained object-level security.
- It allows non-mediated access to networked storage devices.
- It is possible to support efficient storage management, namely, controller QoS guarantees, object placement, and so on.

This data replication layer consists of three components: an OSD controller, an OSD target, and a replication manager. Figure 6 shows the internal structure of the data replication layer.

The OSD controller enables a personal device to behave as an OSD initiator which can communicate with other OSD target devices. Each OSD target device detected in a PAN is recognized as a general SCSI device to the upper level data access layer.

Since a personal device cannot always be connected to the network due to its resource-restricted environment, the data replication manager must create and manage the replica nodes with replication metadata. We consider the lightweight and decentralized protocol for low overhead in the resource-restricted environment of a personal device. The main functions of a replication manager include: replica creation, replication placement, replica access, and management of replica metadata.

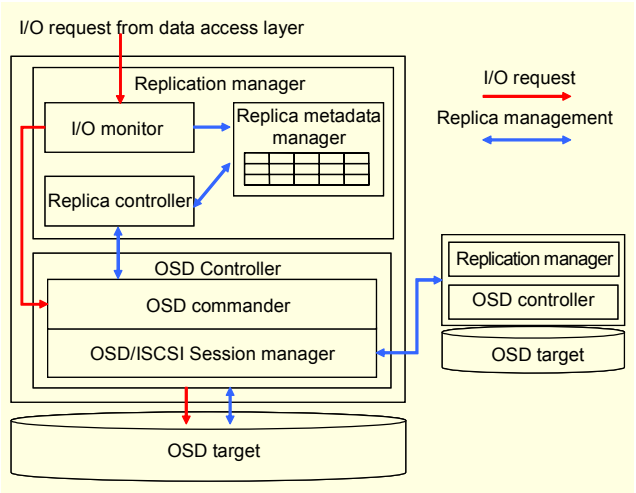

Fig. 6. Data replication layer.

The I/O monitor investigates every read/write operation and maintains the read/write ratio of each object. In our data replication layer, creating or deleting replicas is triggered by using the read/write ratio or the pre-defined target availability of each replication unit. The replica manager maintains replica related metadata and creates and deletes replicas.

B. Replication Unit

The data replication layer assumes the replication unit which is a basic unit for replication. Each replication unit is an object in the object-based storage device (OSD), which is a container for real file objects, depending on the system configuration. Figure 7 shows how a replication unit is structured in our framework. The replication unit includes two fields: the object data field and the object attribute field. The object data field actually stores the object pointers to actual data objects to be replicated. The object attribute field contains the replica metadata on how objects are replicated. The replica metadata includes reference availability, the original owner device ID of the data, the replica node IDs, and so on.

The home_node information represents the device in charge of replica creation and deletion as well as replica metadata management, whereas the replica_placements information describes the replicas of the replication unit and the failure probability of each replica. The version information for replicas themselves and replica metadata is also maintained for consistency.

Figure 9 illustrates how the operation of replica metadata management works. Assume that node A has the home_node (HN) of the replication unit (RU), and the replica can be found in node B. Then, the replica manager of node A tries to create a new replica in node E via the OSD protocol when it detects that the current estimated availability (0.6) of the RU is lower than

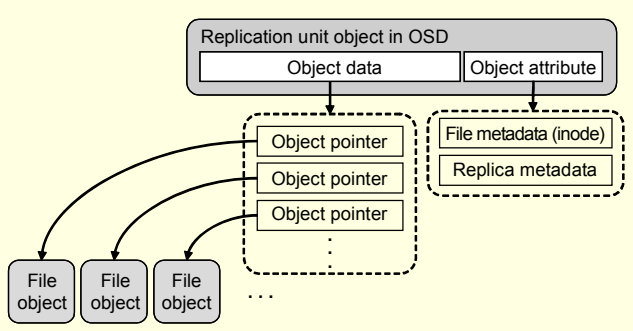

Fig. 7. Replication Unit.

```
Replica Metadata structure for a Replication Unit

struct replica_metadata {
    /* Home node */
    struct osd_node         home_node
    /* Replica information */
    struct osd_node_list    replica_placements
    /* Version of Replica Metadata */
    unsigned int            ru_md_version
    /* Version of Replication unit */
    unsigned int            ru_version
    /* Pointer to data object list in Replication Unit */
    struct object_list      objects
    /* Current availability of Replication Unit */
    unsigned int            current_availability
    /* Target availability of Replication Unit */
    unsigned int            target_availability
}
```

Fig. 8. Replication metadata.

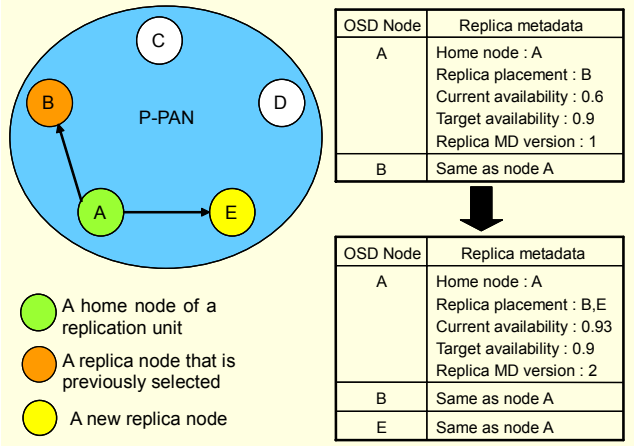

Fig. 9. Replica metadata update example.

the required reference availability (0.9), the target availability, specified in its replica metadata. After successfully creating a data replica in node E, the replica manager of node A re-estimates the current availability (0.93) of the RU. If the current availability is greater than or equal to the target availability, it sends the updated replica metadata of previously and newly created replicas of the RU to nodes B and E.

C. Replica Management Policy

At first, the data replication layer tries to discover all the accessible OSD devices. Figure 10 describes in detail the main steps of OSD device discovery via UPnP. First, a new OSD node managed by the PosCFS+ replication manager is discovered. Next, the replication management service on the home node shown at the left side of the figure writes the information of the newly discovered node to the proc file system in Linux, including the IP address, the failure probability of the node, and the node status. Then the replication manager obtains the information from the proc file

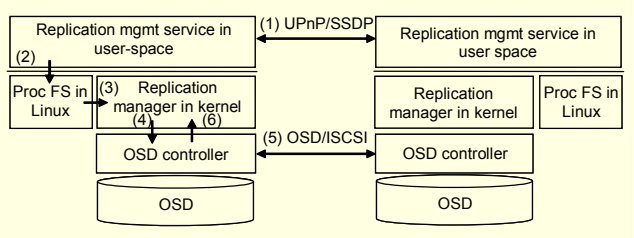

Fig. 10. Replica discovery steps.

system and updates the OSD node list which is maintained for a future replica selection. If a new replica of the RU is required, then the replication manager notifies the OSD controller to create a new replica node. The OSD controllers negotiate with each other in order to create an OSD/iSCSI session. Finally, the OSD controller reports the operation result to the replication manager.

The data replication layer has to deal with the following issues for replica placement.

- Data availability estimation for an RU and replica management
- Consistency management
- Home node election

1) Data availability estimation and replica management

The home node of an RU takes charge of creating and deleting replicas and updating the replica metadata. The replication manager in the home node continually estimates the failure probabilities of all the replicas under its supervision. When it finds that an RU does not satisfy the availability requirement as we mentioned before, that is, the currently estimated availability of the RU is less than the desired reference availability (the target availability specified in the replica metadata of an RU), the replication manager of the home node selects a candidate node for a new replica from the OSD node list. Estimating the current availability of an RU is based on the following formula:

$$\text{Current availability of an RU} = 1 - \prod_{i=1}^{n} p_i$$

where

n : the number of replicas,

p_i : the failure probability of node i.

We assume that the failure probability of all the OSD devices is known in advance. The replication manager of the home node selects the device with the highest availability among the OSD devices as a new replica.

2) Consistency management

We use a simple read-one/write-all (ROWA) method [23] for consistency management among replicas. In our replication

framework, read requests for data objects are allowed from any replica, while write requests for data objects should be propagated from home nodes to all of its replicas currently available after the write requests from the upper layer. As previously mentioned, writes can be permitted only to objects maintained by home nodes.

Figure 11 illustrates how the ROWA method works and how the replica metadata is updated. Assume that node A is the home node of a data object whose replicas are found in nodes B, D, and E. When a client sends a read request to node B, node B first retrieves the replica version information by sending a request message to home node A. Then, node B can carry out the read operation requested by the client as long as it finds that the received replica version is identical to the replica version found in its local replica metadata. Otherwise, the read request will be forwarded to the home node.

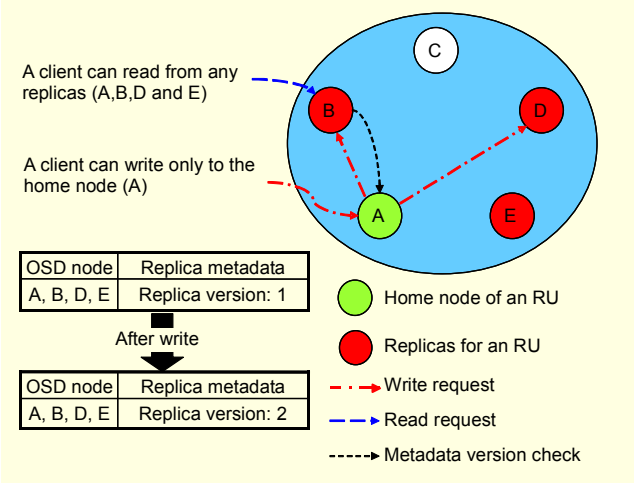

Fig. 11. Replica consistency management.

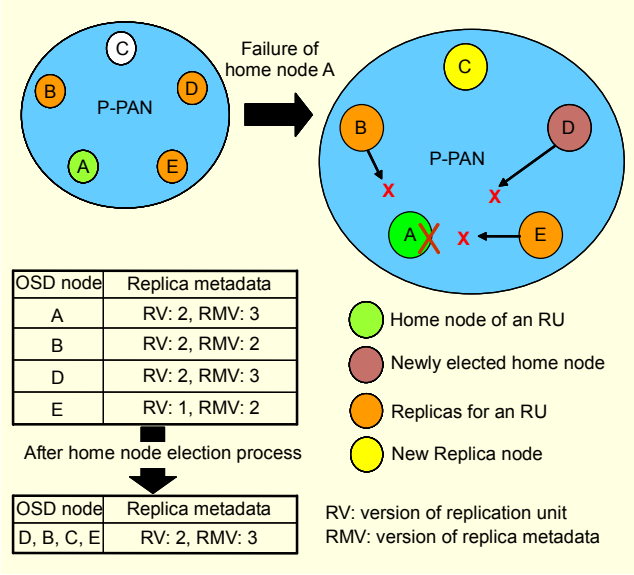

Fig. 12. Example of home node election process.

Regarding the write operation, the home node increases the replica version by one before processing a write request received from a client. After fulfilling the write request, the home node sends the updated replica metadata information to nodes B, D, and E, where the corresponding replica metadata is stored.

3) Home node election

It is important to note that the operation of creating and deleting replicas can be performed only by the home node. Since all the nodes are weakly connected by wireless connection in a PAN, we face the situation where the HN is no longer accessible in the current configuration of a PAN. In order to ensure the correct replica operation even when the original home node is not available, the replication manager elects a new home node.

To detect the failure of a home node, every replica node has to check the status of its home node periodically. When the break-down of the original home node is detected on a replica node, they negotiate with each other for election. If a replication manager on the firstly noticed node recognizes that it has the most recently updated RU, then it becomes the new home node itself and then propagates the event for the new node election. If not, it relinquishes its right as a candidate. In that case, the secondly noticed node performs the same process. This process is repeatedly propagated to all the replica nodes in consecutive order.

D. Automatic Backup in PAN

Usually, there are highly stable nodes in a PAN. A home server or a desktop are typical examples for this. In such an environment, all personal data on various devices in the PAN can be automatically backed up to the most reliable node, such as a home server or a desktop.

IV. Experimental Evaluation

In order to evaluate PosCFS+, we conducted several experiments, particularly on the device discovery time including the initialization time of the data access layer and the query response time for actual data retrieval. For that purpose, a prototype of PosCFS+ was developed in Linux 2.6.12 with Intel UPnP SDK v1.4, Intel OSD/ISCSI reference implementation v2.0.16, SQLite v3.2.7, libneon 0.25.3, and libextrator 0.5.14. The prototype implementation also includes a VFS support module in Linux. We implemented the module using the file system in user-space (FUSE) functionality provided by the Linux kernel and libfuse 2.6.0.

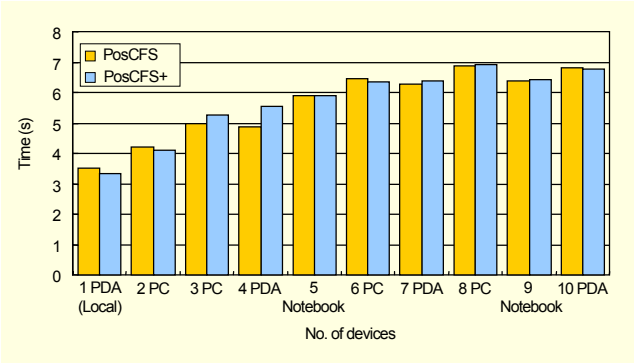

Fig. 13. Device discovery time in PosCFS+.

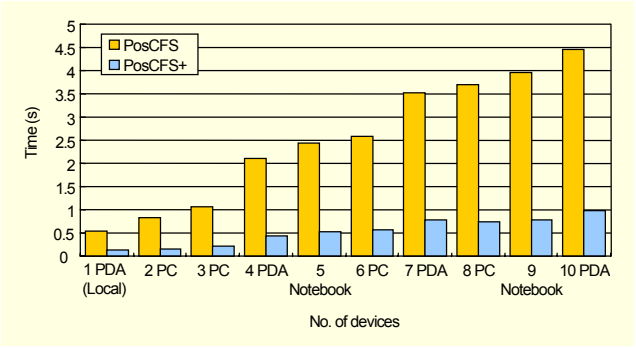

Fig. 14. Query processing time in PosCFS+.

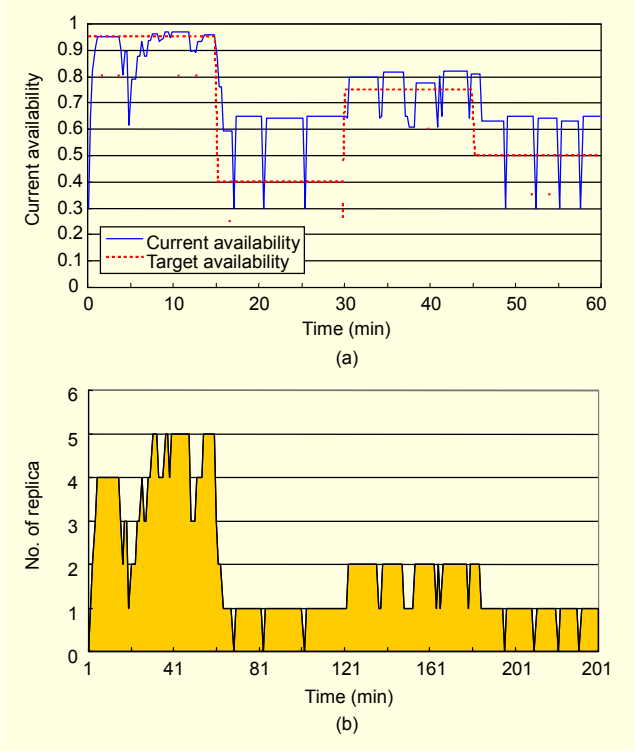

Fig. 15. Simulation results in case of 10 devices: (a) availability and (b) number of replicas.

The testing platforms we used include four Jaurus-SL5500 PDAs with Strong ARM 266 MHz, 32 MB memory, and 802.11b wireless networks; three PCs with 0.8 GHz to 2.8 GHz Pentium, 512 to 1000 MB RAM, and 100 MB wired Ethernet; and two laptop computers with 1.8 GHz P4, 512 MB RAM, and 802.11g wireless Ethernet.

Figure 13 shows the results for device discovery time. Note that the device discovery time is not dependent on the number of devices; it gradually converges to 7 seconds. This is because the UPnP discovery process depends on the search time of the simple service Discovery Protocol (SSDP) [4].

The response time to a request for virtual directories by a client is referred to as query processing time which is one of important criteria for scalability. Figure 14 shows how query processing time varies as the number of devices increases. In this evaluation, 1000 files are uniformly distributed to all nodes. The results demonstrate that the query processing time is greatly reduced in PosCFS+. This enhancement is made possible by the complete redesign of the file metadata representation framework from an ontology-based scheme to one based on the database and keyword table, thus eliminating the parsing overhead for ontology-based documents. Moreover, we have redesigned the metadata management scheme to separate the user profile from the file metadata and optimized the query processing code. Moreover, query processing time is less sensitive to the number of devices in a PAN with our new design.

To evaluate our newly designed data replication layer under various circumstances, we conducted several experiments by simulation. We assumed that the underlying wireless network is 54 Mbps 802.11g and the failure probabilities of the various devices are uniformly distributed between 0.4 and 0.5. The probability of the home node was fixed to 0.3 which is a little bit less than those of the other devices.

Initially, a data object of 100 MB was stored in its home node and the reference data availability was set to 0.3. For simplicity, no write operations were assumed during the simulation process. Simulations were conducted under system configurations with 5, 10, and 20 devices. The results are shown in Figs. 15 and 16. We changed the target availability of the replication unit dynamically to show how our data replication layer would react to the current system configuration. Initially, the reference availability was set to 0.95. It was changed to 0.4 at 15 minutes, set to 0.75 at 30 minutes, and finally changed to 0.5 at 45 minutes. Figure 15 shows how the current availability of the data object was adjusted dynamically by data replication and how many data replicas were currently managed in the system of 10 devices. The results demonstrate that our data replication layer manages the data replication correctly in the current system configuration.

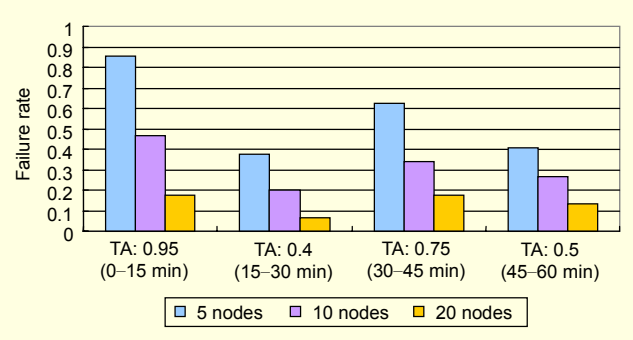

Fig. 16. The failure rates showing how many times the resulting data replication processing cannot meet the given target availability when the number of devices is 5, 10, and 20.

When the number of devices in the system is small, the effect on data availability of losing a single device may be much more significant than when there are many devices. Therefore, our data replication layer may not find an appropriate solution that can satisfy the target availability of the data object due to the small number of devices operating. This may trigger more frequent execution of the data replication layer. Figure 16 shows the failure rates, namely, the number of times the resulting data replication process cannot meet the given target availability. Figure 16 demonstrates that our data replication layer produces more reliable results as the number of devices in a system increases.

V. Conclusion and Future Work

Core functionalities required for file service in ubiquitous computing were designed and implemented in a system we previously proposed, PosCFS. The key feature of the file service is a virtualization technique which allows per-user namespace for managing and accessing data stored on physical storage spaces detected in PAN via UPnP protocol.

In this paper, we proposed a new smart file service framework, PosCFS+, which is an improved and extended version of PosCFS in terms of self-manageability. Performance has been improved by redesigning the metadata management scheme using database and keyword based query rather than ontology. In addition, the data replication layer is newly designed separately from the existing data access layer for automatic data management based on the OSD protocol. The separate design of the data access and replication layers enables extension and interoperation with other non-PosCFS-based devices such as a backup server or a home server system.

We need to conduct more extensive evaluation with realistic workloads on the data access layer and data replication layer. Moreover, supporting context-awareness in file systems should be considered in more details. In this paper, we presented a virtual directory concept and per-user global namespace using user profile and file metadata management techniques. However, there are some remaining unresolved issues related to the management framework. We believe that the framework should be improved with automated metadata tagging using external sensing information and file relation extracted from I/O monitoring. Moreover, in-depth research into context-awareness is needed to enable context-aware applications to easily access relevant data corresponding to the current context managed by a context server in a PAN.

Privacy and security should be also considered as well as data searching in PAN. Currently, we are considering a data searching framework utilizing existing desktop search engines, such as Google Desktop Search [24] or Beagle [25] with our file service.

References

[1] Jim Gray, "Storage Bricks Have Arrived," *Keynote presentation at the USENIX Annual Conference on File and Storage Technologies (FAST)*, 2002.

[2] W. Lee, S. Kim, J. Shin, and C. Park, "PosCFS: An Advanced File Management Technique for the Wearable Computing Environment," *LNCS 4096 - Proc. EUC'06, IFIP,* 2006, pp. 965-975.

[3] J. Shin, W. Lee, S. Kim, and C. Park, "PosCFS: A Context-Aware File Service for the Wearable Computing Environment," *Proc. Next Generation PC Int'l Conf.*, 2005.

[4] UPnP Forum, "UPnP: Universal Plug-and-Play," http://www.upnp.org

[5] IETF, "WebDAV: Web-Based Distributed Authoring and Versioning," RFC 2518.

[6] W3C, "RDF: Resource Description Framework," http://www.w3c.org/RDF

[7] W3C, "OWL Web Ontology Language," http://www.w3.org/TR/owl-features

[8] SQLite, http://www.swlite.org

[9] T10, "SCSI Object-Based Storage Device Commands (OSD)," http://www.t10.org/ftp/t10/drafts/osd

[10] C.K. Hess and R.H. Campbell, "A Context-Aware Data Management System for Ubiquitous Computing Applications," *Proc. Int'l Conf. Distributed Computing Systems*, 2003.

[11] A. Karypidis and S. Lalis, "OmniStore : A System for Ubiquitous Personal Storage Management," *Proc. Fourth Annual IEEE Int'l Conf. Pervasive Computing and Communications (PERCOM'06)*, 2006.

[12] D. Peek and J. Flinn, "EnsemBlue: Integrating Distributed Storage and Consumer Electronics," *7th Symp. Operating Systems Design and Implementation (OSDI)*, 2006.

[13] E.B. Nightingale and J. Flinn, "Energy-Efficiency and Storage

Flexibility in the Blue File System," *6th Symp. Operating Systems Design and Implementation (OSDI)*, 2004.
[14] T. Hara, "Data Replication Issues in Mobile Ad Hoc Networks," *6th Int'l Workshop on Database and Expert Systems Applications*, 2005.
[15] T. Hara and S. Madria: "Consistency Management among Replicas in Peer-to-Peer Mobile Ad Hoc Networks," *Proc. of Int'l Symp. Reliable Distributed Systems,* 2005.
[16] T. Hara and S. Madria, "Location Management of Replicas Considering Data Update in Ad Hoc Networks," *Proc. 20th Int'l Conf. Advanced Information Networking and Applications*, 2006.
[17] M. Rodrig, A. LaMarca, "Oasis: An Architecture for Simplified Data Management and Disconnected Operation," *Personal and Ubiquitous Computing Journal*, vol . 9, no. 2, 2005.
[18] D.K. Gifford, P. Jouvelot, M.A. Sheldon, J.W. O'Toole, Jr., "Semantic File Systems," *13th ACM Symp. Operating Systems Principles,* 1991.
[19] Y. Padioleau, O.Ridoux, B. Sigonneau, S. Ferre, M. Ducasse, O. Bedel, and P. Cellier, "LISFS: A Logical Information System as a File System," *28th Int'l Conf. Software Engineering,* 2006.
[20] C.A. Soules and G.R. Ganger, "Connections: Using Context to Enhance File Search," *20th ACM Symp. Operating Systems Principles,* ACM Press, 2005, pp. 119-132.
[21] A. Ames, N. Bobb, S.A. Brandt, A. Hiatt, C. Maltzahn, E.L. Miller, A. Neeman, and D. Tuteja, "Richer File System Metadata Using Links and Attributes," *Proc. the 22nd IEEE / 13th NASA Goddard Conf. Mass Storage Systems and Technologies,* Monterey, CA, April 2005.
[22] IBM, "Object Storage: The Future Building Block for Storage Systems," http://dl.alphaworks.ibm.com/technologies/osdsim/osdsim2.pdf
[23] R. Budiarto, S. Noshio, and M. Tsukamoto, "Data Management Issues in Mobile and Peer-to-Peer Environments," *Data and Knowledge Engineering*, vol. 41, 2002, pp. 183-204.
[24] Google Desktop Search, http://desktop.google.com
[25] Beagle, http://beable-project.org

Woojoong Lee received the BE degree in chemical engineering and the MS degree in computer science from Hanyang University, Korea, in 2002 and 2004, respectively. He is currently a PhD candidate in the Department of Computer Science and Engineering, POSTECH, Korea. His research interests include pervasive computing, storage systems, and embedded systems.

Shine Kim received the BS degree in earth and environment sciences from Chonbuk National University, Korea, in 2004. He also received the MS degree from the Graduate School for Information Technology of POSTECH, Korea, in 2007. After graduation, he joined Samsung Electronics Co., Ltd. His research interests include pervasive computing and storage systems.

Chanik Park received the BE degree in 1983 from Seoul National University, Seoul, Korea, the MS degree in 1985, and the PhD degree in 1988, both from Korea Advanced Institute of Science and Technology, Korea. Since 1989, he has been working for POSTECH, where he is currently a professor in the Department of Computer Science and Engineering. He was a visiting scholar with Parallel Systems group in the IBM Thomas J. Watson Research Center in 1991, and was a visiting professor with Storage Systems group in the IBM Almaden Research Center in 1999. He has served a number of international conferences as a program committee member. His research interests include storage systems, embedded systems, and pervasive computing.