VMDFS: Virtual Memory based Mobile Distributed File System

Susmit Bagchi
ATD/CS Research Laboratory, Samsung India Software Operations
Bagmane Lake View, C.V. Raman Nagar, Bangalore, India
susmitbagchi@yahoo.co.uk

Abstract

The mobile computing paradigm is becoming a reality due to the advent of mobile devices such as, PDA and Smart Phones having wireless interface to WWW. As a result, the high-end applications related to mobile multimedia and distributed digital contents have gained much attention. However, the existing distributed and mobile file systems are not adequately suitable to support such applications handling the challenges offered by mobile computing systems. In this paper, the concept, design architecture and consistency model of Virtual memory based Mobile Distributed File System (VMDFS) are proposed based on thin-mobile-client/fat-server model. VMDFS creates the migrate-able virtual memory based mobile file system on top of the disk file systems of remote servers employing dynamic frame-lock to reduce network latency and to attain high-performance. An abstract mathematical model of VMDFS is constructed and associated properties are evaluated. The kernel-level implementation framework of VMDFS is illustrated based on Linux 2.4.22. The network paging latencies for LAN, Wireless VPN and 2.5G GPRS are measured experimentally. The results indicate that VMDFS is realizable.

Keywords: *Virtual Memory, Mobile Computing, MDVM, File System, Kernel, Network Latency.*

1. Introduction

Due to the advent of mobile computing devices having wireless communication interface and access to internet such as, PDA, Smart-Phone, the mobile computing paradigm is becoming a reality [28][29][30]. As a result, a set of high-end mobile applications, for example mobile multimedia systems and digital contents, has emerged [1]. In general, the distributed multimedia applications rely on the static client-server computing model based on high-speed wired network and fail to handle the challenges imposed by mobile computing paradigm. The technical challenges associated to mobile computing system are the resource constraints of mobile devices and the intermittent low-bandwidth wireless network [28][29]. Researchers have directed to utilize the resources of remote servers by mobile clients through the wireless communication interface in order to realize the high-end mobile distributed applications [16][29][30]. The resources and storage offered by servers enable the mobile devices to create, store and distribute/share the digital contents under mobility. In order to realize the location transparent and high-available mobile digital contents, a mobility-aware, distributed and high-performance file system is required. The traditional file system interfaces need to be extended to achieve high-performance coping with the larger file sizes and mobility [1]. The existing distributed file systems such as, NFS, DAFS and DFS, perform poorly in the mobile computing environment [6][24][25]. Researchers have directed to design the kernel-level NFS [27] and kernel-level resource management architecture to achieve QoS

and better resource utilization [2][3][5]. The traditional RAM-disk file system tries to achieve comparatively lower data-access latency, however, it suffers from the poor performance, volatility of data and the duplicate buffer copy [10]. In addition, the multimedia data must not be paged-out of main memory to maintain high-performance [21]. On the other hand, the proposed file system for mobile computing suffers from the complexity and lack of location transparency [31]. In this paper, the design architecture, a model and an implementation framework of Virtual memory based Mobile Distributed File System (VMDFS) are proposed. In addition, the consistency model of VMDFS is described. The proposed VMDFS architecture may serve as a platform to develop a set of high-end mobile applications such as, mobile multimedia system and the electronic briefcase of digital contents. The VMDFS model incorporates location transparency by employing the concept of Server-Group (SG) embedded into the mobile communication infrastructure. An abstract mathematical model of VMDFS is constructed to establish detail analysis and verification considering the mobility of clients. In order to design and implement VMDFS, the monolithic kernel of Linux 2.4.22 is chosen. The round-trip network paging latencies are measured experimentally for the wired LAN, wireless VPN and 2.5G GPRS considering variable page-sized data blocks. The experimental results demonstrate that VMDFS would be realizable based on 2.5G, 3G or higher mobile communication systems. The distinguishing features of the proposed VMDFS model are as followings.

● *The VMDFS design architecture implements the virtual memory based mobile file system on top of the local disk file system to achieve better resource monitoring and utilization along with the reduction of complexity of developing a new mobile file system.*
● *The VMDFS design incorporates location transparency along with the reduction of network-latency using the concept of Server-Group (SG) residing in the cells allowing inter-SG page migration.*
● *The VMDFS combines the file system and virtual memory system employing the selective page-locks to reduce file access latency and to eliminate duplicate buffer copy.*
● *Unlike the NFS* [6]*, the VMDFS does not require the high-speed reliable network between clients and servers.*
● *Contrasting the DAFS and RDMA based designs* [8]*, the VMDFS does not require any specific hardware support.*
● *In contrast to Segank* [23]*, the VMDFS does not require the mobile-clients to be always power-on.*
● *Contrasting the proposed client-centric file system for mobile computing* [31]*, the mobile clients of VMDFS are free from the task of distributed resource management under mobility.*
● *Unlike Sprite* [36]*, VMDFS does not rely on large client cache and entire cache eviction on client side to maintain write-consistency of files.*

The rest of the paper is organized as followings. Section 2 describes the related works. The architecture of VMDFS is illustrated in section 3. An abstract mathematical model and consistency model of VMDFS are illustrated in section 4. Section 5 describes the implementation framework of VMDFS along with the experimental evaluation of network-paging latency. Section 6 concludes the paper.

2. Related Works

In order to realize the mobile computing paradigm, the MDVM system is proposed aiming to reduce the computational resource constraints of mobile devices [29]. The MDVM system

advocates utilizing the resources of remote server by the mobile clients through wireless communication interface. The existing Distributed Virtual Memory (DVM) systems do not provide an easy way to dynamically use and share DVM resources preserving transparency, adaptability and extensibility [29][30]. The DVM systems become non-scalable under the condition of mobility of clients [29]. The present design architectures of operating systems make it difficult to establish a low latency communication path between the application or middleware and the kernel to access the resources [1][2]. The user-level and middleware-level management of resources to maintain QoS have restrictions due to the granularity of resource management and insufficient fidelity [2]. The KECho [5], Q-channels and ELinux [2][3][4] are the kernel-level architectures to implement the distributed resource management based on event-oriented communication and QoS characteristics. The distributed multimedia file systems have two broad classifications namely, *(1) Partitioned FS* such as, FFS, RIO, Shark and a combination of UFS and CMFS [1] and, *(2) Integrated FS* based on multiplexing the disk bandwidth, storage space and the buffer cache among all the multimedia data [1]. The examples of integrated FS are Nemesis FS, Fellini and Symphony [1]. Apart from these, MMFS handles the interactive multimedia applications by extending the UNIX file system [1]. The file systems designed for high-bandwidth and low-latency client-server model are unable to perform in the low-bandwidth and high-latency mobile computing environment [6]. Researchers have suggested implementing the disconnected file operations using cache, for example the Coda file system, although this technique fails to take advantage of the network opportunities and is inappropriate for fully connected systems [6]. On the other hand, a data staging architecture is proposed to improve the performance of DFS running on the storage limited mobile devices [7]. This technique opportunistically prefetches the files and caches them on nearby un-trusted surrogate machines [7]. However, such data staging mechanism is complex and requires multi-path communication between the mobile clients and servers. The DAFS design relies on the kernel-server of FreeBSD operating system, which is a direct-access file system based on the supports provided by the networking hardware (NIC) for user-level networking [8]. It suffers from the hardware dependency and permanent page locking to employ RDMA [8]. In a different approach, the Federated DAFS combines user-space DAFS with a low overhead clustering layer to prepare a scalable DAFS cluster [9]. However, the Federated DAFS does not consider the mobility of the clients. The Sprite uses large main-memory to cache disk blocks providing non-write-through file caching [36]. The difficulties associated to Sprite are that it does not consider the mobility of the clients and requires a large cache size in the main memory of the clients. The Sprite bases on entire cache eviction on each client side on the event of file update by any client resulting in repeated file re-caching consuming network bandwidth and time. Inspired by RAM-disk system, a page-able memory based file system is developed [10]. The proposed design architecture aims to reduce the poor utilization of available main memory. However, the proposed solution employs duplicate data copy between file system cache and the system buffers [10]. It is reported that when the variable sized file cache and the virtual memory are coexisting in a system, the concurrent execution of file-intensive processes degrades the overall system performance if the cache pages and the virtual memory system are not treated differently [11]. Instead of permanently locking the virtual memory pages, an adaptive page-locking policy is proposed to maintain the QoS for the large-file-intensive applications [12]. The proposed solution performs adequately to maintain the QoS, however, could not reduce the page faults considerably and its performance is dependent on the behaviour of the applications. The architecture of a virtual memory based distributed file system in high-speed wired network environment is proposed in [13]. It is noted that establishing a large file cache in the virtual memory is possible without introducing any performance degradation [14]. Apart from designing the file system, an intelligent file hoarding for mobile computers is proposed, which heuristically

creates the file set based on file access patterns [15]. The network file system such as, NFS, does not utilize the warm cache on the face of disconnection between a client and the server and hence, is not suitable for mobile computing systems [24][25]. Researchers have directed to design the kernel-level NFS client architecture bypassing the kernel buffer cache and thus avoiding data copy [26][27]. A mobile distributed file system using client-centric update and caching along with lazy propagation has been proposed [31]. However, the main difficulties of this design are the client initiated primary-to-primary lazy copy of entire file set and the client initiated localization of server, which degrades the criteria of location transparency. In addition, due to the lazy update propagation and the client resident cache, the updates are vulnerable to the client crash. The other works related to the clusters, mobile storage systems and data partitioning include DATOM [16], e-vault [17], COMA [19], PersonalRAID [22] and Segank [23]. In order to enhance performance, a dynamic set model is developed to overlap the file data fetch and processing [18][20]. This approach is useful for weakly connected mobile clients. However, the fetching policy of dynamic set model may degrade the performance by stalling the applications and it performs poorly in the high-speed network environment [18][20].

3. The VMDFS Architecture

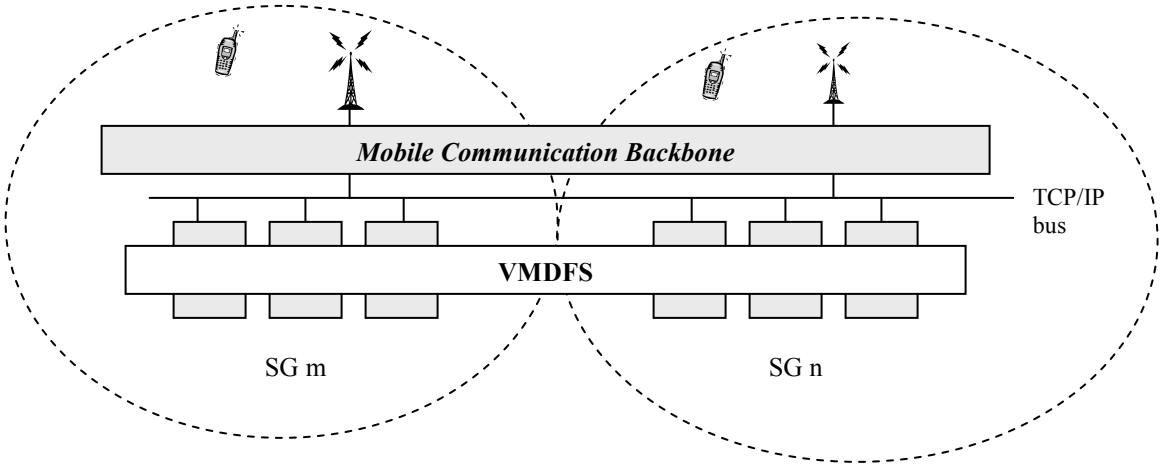

Figure 1. Conceptual Architecture of VMDFS.

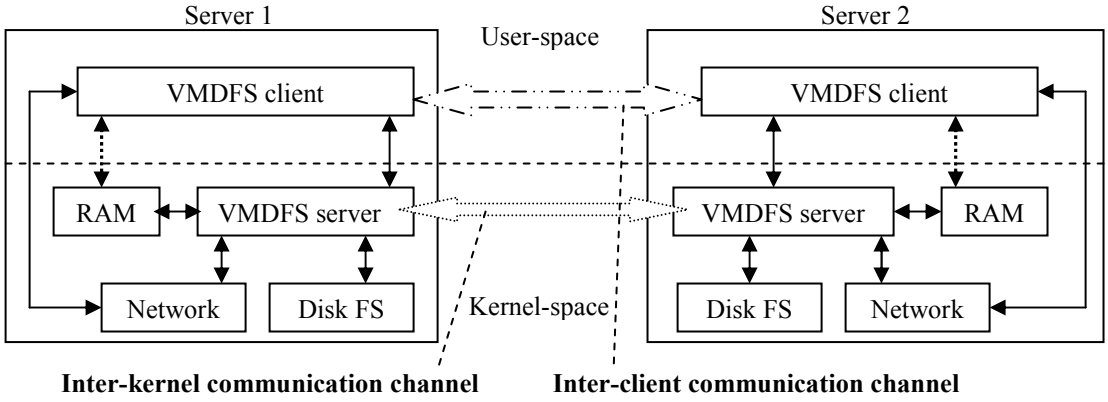

Figure 2. Architectural Components of VMDFS.

The design architecture of VMDFS considers the Server-Group (SG) in a cell and SGs of the other cells of the mobile communication architecture. A SG is comprised of a set of resource-fat servers residing in the cells [29]. The servers of a SG in a cell as well as different SGs are connected by high-speed wired TCP/IP network.

The VMDFS architecture can be viewed as a distributed file system layered across servers in the SGs accessible by mobile clients with location transparency. The conceptual architecture of VMDFS is shown in Figure 1. The servers in a SG are connected by TCP/IP bus and are attached to the mobile communication backbone. The VMDFS is constructed in the kernel of the operating system of the individual servers in order to achieve better monitoring and utilization of available resources at servers and to establish faster inter-kernel communication system. The detailed architecture of VMDFS residing in the kernel of two servers is shown in Figure 2. The VMDFS architecture can be decomposed into two functional entities namely, VMDFS client and VMDFS server. The VMDFS client has dual functionalities. The VMDFS client residing in the user-space of a server of a SG acts as a server to the mobile clients. On the other hand, the same VMDFS client acts as a client to its local VMDFS server residing in the kernel-space. The user-space resident VMDFS clients in a SG communicate through TCP/IP bus. The kernel-space resident VMDFS servers in a SG communicate using a separate inter-kernel communication channel based on events. The VMDFS servers resident in SGs construct a distributed file system on top of individual local disk file systems. The VMDFS servers create the virtual memory resident page-able file system utilizing available free main memory. As the disk access latency is an order higher than the memory access latency, this enables to reduce data-access latency by decreasing the disk access frequency. Due to the mobility of clients, the resource-load on servers may vary randomly in short time. In order to handle the unpredictable variation of resource-load, the VMDFS servers use a dynamically varying set of virtual memory pages to establish the file system. In order to avoid swapping of file system pages to the high latency disk drive, the page frames are locked temporarily. Based on the resource-load of the system, the page frames are unlocked and flushed to disk if the pages are dirty. Otherwise, the page frames are released to the kernel to balance the resource-load. On the other hand, in case of the long time stability of the system in terms of resource-load, a periodic flushing of dirty pages to the disk storage is employed to attain durability property of the file systems without releasing the page frames to the kernel.

3.1. File System Structure

This section describes various components of the file system structure associated to VMDFS. The file system design of VMDFS uses data-caching model based on page-level transfer. The advantages of such design are the reduction of network traffic and the suitability to the diskless mobile thin-clients. Conceptually, the VMDFS residing in the virtual memory pages of a server can be segmented into the metadata structure and the list of data blocks. The file metadata structure of VMDFS maintains per client file status and IO information along with the page update information. The file metadata structure can be marshaled and stored on disk as a file for future reconstruction. The components of the file system structure of VMDFS are depicted in Figure 3. The file structure is fundamentally composed of the sequential stream of File System Blocks (FSB). Each FSB is comprised of File Metadata Structure (FMS), Page Frame List (PFL) and Migrated Page List (MPL). The FMS contains the file metadata information, whereas, the PFL contains the information related to page frames residing in virtual memory area holding the file data blocks. The MPL holds the globally resolvable addresses of pages those are migrated to another server. The List Index (LI) is used to locate a page list of an open file at a VMDFS server in a SG. The LI of each

open file at each server is distinct and is formed by concatenating the File Home Address (FHA) and a monotonically increasing integer Sequence Number (SN) assigned to the open files. The File Path Name (FPN) gives the complete path and name of an open file. The File Update Flag (FUF) denotes the consistency states of the file in the file system, whereas, the mobile Client List (CL) denotes the list of client profiles accessing a file. The Number of Disk Blocks (NDB) of a file contains the file size as a multiple of page size and the Number of File Blocks (NFB) depicts the number of file blocks present in virtual memory at any time. The File Security and Sharing Option (FSSO) states the file sharing and security parameters as set by the file owner. The FSSO allows realizing a set of specialized policies such as, selective group-sharing, time-limited sharing, file-fragment sharing and, many-read/once-write sharing of files. The Page Rank (PR $\geq$ 0) is a monotonically increasing integer associated to each page frame holding the file data block and is immutable as long as the file blocks are memory resident. The Page Frame Descriptor (PFD) holds the mutable physical address of the page frame containing file data and the other information related to the page frame. The Write Limit (WL) of each page denotes the length of valid data in a frame from the starting address of the corresponding page frame. The Page Write Log (PWL) contains the updated data of a page in a newly attached page frame having null PR and results in the formation of a tree of page frames. It is to be noted that the page frame in a PWL will have the PR equal to null until the update is merged. Update merging is done by simply readjusting the page frame addresses of the parent PFD and the PFD in PWL. This avoids the memory copy operation and facilitates the roll-back mechanism to recover from the failure. The tree of page frames containing the update log is depicted in Figure 4. The Disk Block Pointer (DBP) of each PFD contains the mutable disk block pointer of the underlying file system for the pages in PFD. During the flush of the updated pages, the pages of each PR are updated based on the update log given in PWL and are sequentially written to the disk starting from the DBP of the page having lowest PR in the PFL. Hence, the future reconstruction of PFL may have different PFD, DBP but the PR remains the same as earlier denoting the file segment marks in the stream of data. The DBP of a page may change due to the file append and truncation operations. Due to the mobility of clients, the network-distance between a server in SG and a mobile client may increase. On the event of hand-off, a mobile client may cross the cell boundary entering in the domain of another SG. In order to reduce the network-cost and data-access latency, VMDFS servers employ inter-SG page migration while maintaining consistency of the data blocks. Hence, instead of employing a static file replication among SGs, a page-set replication of a file is created dynamically through page migration. Any update of the migrated page is propagated periodically to the home-location server of the corresponding file. The MPL holds the PR and Page Remote Address (PRA) of the migrated pages where, PRA is a globally resolvable page address of a migrated page given as ⟨Remote_Server_Address, LI⟩. The mobile clients accessing a file are denoted by individual Client ID (CID), File Mode (FM) of operations such as, read/write and, File Handle (FH) where, FH is given as ⟨LI, PR, Offset⟩. Each PFD contains the Frame Address (FA) giving the physical address of a page frame and the Frame Lock Bit (FLB) to control the locking of a frame in the main memory. The page migration involves the transaction of ⟨FMS, PR, ⟨PFL-PFD, WL⟩, ⟨PWL-PFD, WL⟩⟩ between the source and destination VMDFS servers. The VMDFS server incorporates page-based file IO using memory mapping approach. The duplicate data copy is avoided by mapping the set of memory-locked page frames into the address space of VMDFS client. As a trusted entity, the VMDFS client releases the page frames on demand due to the increase of memory-load above a predefined threshold in a server.

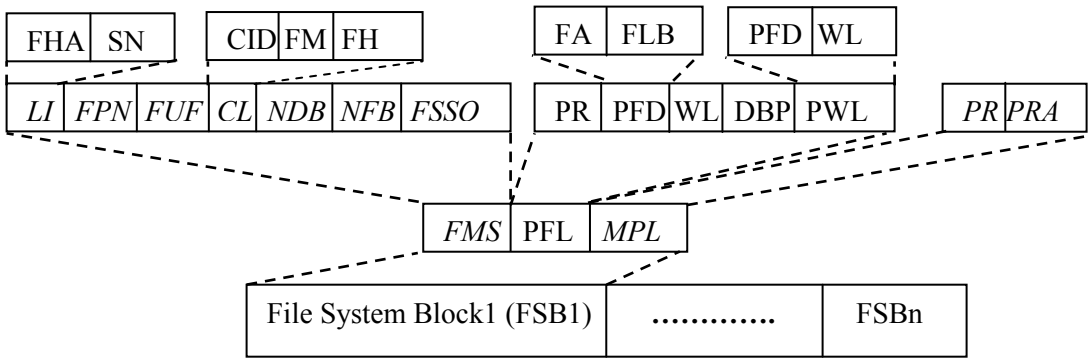

Figure 3. The File System Structure of VMDFS.

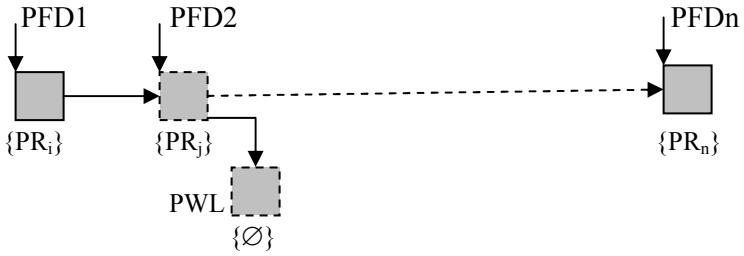

Figure 4. The Tree of Page Frames and Update Log, $PR_n > PR_j > PR_i$.

4. The VMDFS Model

The memory management system of the operating systems can be modeled with abstract and precise formalism in order to represent the design architecture [33][34]. The abstract modeling of the system architecture and memory management mechanisms allow the generalization of the concept and the easiness of understanding without tying up to a particular kind of implementation [33][34][35]. In this section, the architectural model of VMDFS is constructed in abstract mathematical formulation for detail analysis and verification considering the mobility of clients. A set of properties of VMDFS architecture is explained as theorems. Let, a set of cells is denoted by $C_\alpha = \{\alpha_i : i \geq 0, i \in \Gamma^+\}$ and a set of servers $S = \{ s_j : j > 0, j \in \Gamma^+\}$. Let, the binary relation η is defined as η: $S \rightarrow C_\alpha$ such that, $\forall\alpha_i,\alpha_j \in C_\alpha$, $\exists s_n \in S$, $(s_n , \alpha_i) \in \eta \Rightarrow (s_n , \alpha_j) \notin \eta$. Hence, S will have $m = | C_\alpha |$ number of partitions denoted by $\pi_1, \pi_2 \ldots\ldots \pi_m$ such that, $\cup_{i=1, m} \pi_i = S$ and $\pi_i \cap \pi_j = \varnothing$, $i,j \leq m$ and $i \neq j$. A SG is defined as $\pi_i \subset S$ for a $\alpha_i \in C_\alpha$ such that, $\forall s_i , s_j \in S$, $(s_i , \alpha_i) \in \eta \wedge (s_j , \alpha_i) \in \eta \Rightarrow s_i , s_j \in \pi_i$. Again, $\forall s_j \in \pi_i$, F_j denotes a set of home-located files at s_j and $F_j = \{f_{j1}, f_{j2}, \ldots f_{ja}\}$, $a \in \Gamma^+$, $a > 0$. Now, $\forall f_{jx} \in F_j$, the set of file blocks is denoted by $f_{jx} = \langle b_{jxn} : n \in \Gamma^+ \rangle$. Similarly, a set of virtual memory pages resident in main memory at s_j is denoted by $\rho_j = \{p_{jm} : m \in \Gamma^+, m > 0\}$. For all $s_j \in S$, a binary relation Γ_j is defined as, $\Gamma_j : \pi^{\rho}_{jx} \rightarrow \rho_j$, where $\pi^{\rho}_{jx} \subseteq \cup_{k=1,a} f_{jk}$, $f_{jk} \in F_j$. The property of relation Γ_j is that, $\exists b_{jxl} \in \pi^{\rho}_{jx}$ and $\forall p_{jy}, p_{jz} \in \rho_j$, such that, $(b_{jxl}, p_{jy}) \in \Gamma_j \Rightarrow (b_{jxl}, p_{jz}) \notin \Gamma_j$. In addition, in a π_i, the total number of virtual memory pages distributed globally can be computed as $V_i = \cup_{h=1, x} \rho_h$, where $x = |\pi_i|$.

4.1. Definition of ◊

Let, the symbol ◊ denotes a relation between C_α and C, where C is a set of mobile clients, C = $\{c_1, c_2, \ldots c_n\}$, n >> 0 such that, at time t, if $c_i \in C$ and c_i is in the cell $\alpha_i \in C_\alpha$ then, $(c_i \Diamond \alpha_i)|_t$.

4.2. Definition of ⊥

Let, the symbol ⊥ denotes a precedence relation on ◊ such that, $\exists\ \alpha_i , \alpha_j \in C_\alpha$ and $\forall c_i \in C$, the following holds, $(c_i \Diamond \alpha_i)|_{t-1} \wedge (c_i \Diamond \alpha_j)|_t \Rightarrow (c_i \perp \alpha_i)|_t$, $\alpha_i \neq \alpha_j$.

4.3. Definition of ||

Let, a symbol || denotes the association relation between π^V_i and the $c_i \in C$ and is presented as $c_i \parallel \pi^V_i$, where $\pi^V_i \subset V_i$, such that $(c_i \Diamond \alpha_i)|_t \Rightarrow c_i \parallel \pi^V_i$, $\alpha_i \in C_\alpha$.

4.4. Definition of $c_i \parallel \pi^V_{ij}$

If, at time instant t-1 and at the next time instant t, $\forall\ \alpha_i , \alpha_j \in C_\alpha$, $\exists\ c_i \in C$ then, $(c_i \Diamond \alpha_i)|_{t-1} \wedge (c_i \Diamond \alpha_j)|_t \Leftrightarrow\ c_i \parallel \pi^V_{ij}$, where $\pi^V_{ij} = \pi^V_j \cup (\beta_j \circ \Gamma_i)$, $\beta_j : \Gamma_i \rightarrow V_j$, $\pi^V_j \subset V_j$ and $V_j = \cup_{h=1, x}\ \rho_h$, where $x = |\pi_j|$.

4.5. Definition of VMDFS

For all $c_i \in C$ and an integer n > 1, if $(c_i \Diamond \alpha_i)|_{t-n} \wedge (c_i \Diamond \alpha_j)|_{t-n+1} \ldots\ldots \wedge (c_i \Diamond \alpha_m)|_t$ then, the VMDFS of c_i is defined as, $c_i \parallel \pi^V_{ij\ldots m}$ where, $\pi^V_i \subset V_i$, $\pi^V_j \subset V_j$,……$\pi^V_m \subset V_m$ and $\alpha_i , \alpha_j , \ldots \alpha_m \in C_\alpha$.

4.6. Properties of VMDFS Model

Theorem 1: ***Conjunctive predicate property***
Statement: $(c_i \Diamond \alpha_i)|_{t-1} \wedge \neg (c_i \Diamond \alpha_i)|_t \Rightarrow (c_i \perp \alpha_i)|_t$, where $\alpha_i \in C_\alpha$, $c_i \in C$.
Proof: Let, $\alpha_j \in C_\alpha$ and $\alpha_i \neq \alpha_j$ such that $(c_i \Diamond \alpha_j)|_t$. Then, $(c_i \Diamond \alpha_i)|_{t-1} \wedge (c_i \Diamond \alpha_j)|_t \Rightarrow (c_i \Diamond \alpha_i)|_{t-1} \wedge \neg (c_i \Diamond \alpha_i)|_t$. Again, at time t, $(c_i \Diamond \alpha_j)|_t$ and at time t-1, $(c_i \Diamond \alpha_i)|_{t-1}$. Hence, $\neg (c_i \Diamond \alpha_i)|_t \Leftrightarrow (c_i \Diamond \alpha_j)|_t$. So, $(c_i \Diamond \alpha_i)|_{t-1} \wedge \neg (c_i \Diamond \alpha_i)|_t \Rightarrow (c_i \Diamond \alpha_i)|_{t-1} \wedge (c_i \Diamond \alpha_j)|_t$. However, from the definition of ⊥, $(c_i \Diamond \alpha_i)|_{t-1} \wedge (c_i \Diamond \alpha_j)|_t \Rightarrow (c_i \perp \alpha_i)|_t$. Hence, $(c_i \Diamond \alpha_i)|_{t-1} \wedge \neg (c_i \Diamond \alpha_i)|_t \Rightarrow (c_i \perp \alpha_i)|_t$.
Description: The conjunctive predicate property of VMDFS constructs the mobility-graph of a mobile client. Let, at time t, $(c_i \Diamond \alpha_i)|_t$. If the mobile client notifies to a VMDFS server in α_i that $(c_i \perp \alpha_j)|_t$ then, the server in α_i may contact another VMDFS server in $\alpha_j \in C_\alpha$ to discover that $(c_i \perp \alpha_k)|_{t-1}$, $\alpha_k \in C_\alpha$. Hence, the VMDFS server in α_i will construct a conjunctive predicate denoted by $(c_i \Diamond \alpha_k)|_{t-2} \wedge (c_i \Diamond \alpha_j)|_{t-1} \wedge (c_i \Diamond \alpha_i)|_t$, which is a mobility-graph of $c_i \in C$ in this case. This enables a VMDFS server to construct and maintain the page-set of a mobile client by dynamically constructing the mobility-graph.

Theorem 2: ***Distribution property***
Statement: $(c_i \Diamond \alpha_j)|_t \wedge (c_i \perp \alpha_i)|_t \Rightarrow c_i \parallel \pi^V_{ij}$, where $\alpha_i , \alpha_j \in C_\alpha$, $c_i \in C$.
Proof: As, $(c_i \perp \alpha_i)|_t \Rightarrow \neg (c_i \Diamond \alpha_i)|_t$, hence, $(c_i \Diamond \alpha_j)|_t \wedge (c_i \perp \alpha_i)|_t \Rightarrow\ (c_i \Diamond \alpha_j)|_t \wedge \neg (c_i \Diamond \alpha_i)|_t$. However, according to the statement $(c_i \Diamond \alpha_j)|_t$ is true. Hence, in the two consecutive time

intervals t-1 and t, $(c_i \Diamond \alpha_j)|_t \wedge \neg (c_i \Diamond \alpha_i)|_t \Rightarrow (c_i \Diamond \alpha_i)|_{t-1} \wedge (c_i \Diamond \alpha_j)|_t \Rightarrow c_i \,||\pi^V_{ij}$ (from the definition 4.4).

Description: The distribution property of VMDFS explains that pages of a file in VMDFS may be located at a set of servers residing in different SGs.

Theorem 3: ***Vector property***

Statement: $c_i \,||\pi^V_{ij} \neq c_i \,||\pi^V_{ji}$.

Proof: Let, $\pi^V_i \subset V_i$, $\pi^V_j \subset V_j$, $\pi^\rho_{ix} \subseteq \cup_{k=1,a} f_{ik}$, $\pi^\rho_{jx} \subseteq \cup_{k=1,b} f_{jk}$, $p_{ix} \in \rho_i$ and $p_{jy} \in \rho_j$.
Now, from the definition 4.4, $c_i \,||\pi^V_{ij} \Rightarrow c_i \,||\pi^V_j \cup \beta_j(\Gamma_i (\pi^\rho_{ix})) \Rightarrow c_i \,||\pi^V_j \cup \beta_j(e_i) \Rightarrow c_i \,||\pi^V_j \cup v_j$, where $v_j \subset V_j$ and $e_i \subset \rho_i$. Again, $c_i \,||\pi^V_{ji} \Rightarrow c_i \,||\pi^V_i \cup \beta_i(\Gamma_j (\pi^\rho_{jx})) \Rightarrow c_i \,||\pi^V_i \cup \beta_i(e_j) \Rightarrow c_i \,||\pi^V_i \cup v_i$, where $v_i \subset V_i$ and $e_j \subset \rho_j$. Hence, $c_i \,||\pi^V_{ij} \neq c_i \,||\pi^V_{ji}$.

Description: The vector property captures the location dependent page migration concept of the VMDFS model. The mobility of a client from cell to cell incorporates the corresponding page-set migration from the source location to the destination, where the characteristics of the page-set are vectors.

Theorem 4: ***Convergence property***

Statement: $c_i \,||\pi^V_{iji} \subset \rho_i$.

Proof: Let, $\pi^V_i \subset V_i$, $\pi^V_j \subset V_j$. Following the definition of $c_i \,||\pi^V_{ij}$, $c_i \,||\pi^V_{iji} \Rightarrow (c_i \Diamond \alpha_i)|_{t-2} \wedge (c_i \Diamond \alpha_j)|_{t-1} \wedge (c_i \Diamond \alpha_i)|_t$. Let, t-1 = x. So, $(c_i \Diamond \alpha_i)|_{t-2} \wedge (c_i \Diamond \alpha_j)|_{t-1} = (c_i \Diamond \alpha_i)|_{x-1} \wedge (c_i \Diamond \alpha_j)|_x$. However, from the definition 4.2, $(c_i \Diamond \alpha_i)|_{x-1} \wedge (c_i \Diamond \alpha_j)|_x \Rightarrow (c_i \perp \alpha_i)|_x$. Hence, $c_i \,||\pi^V_{iji} \Rightarrow (c_i \Diamond \alpha_i)|_t \wedge (c_i \perp \alpha_i)|_{t-1} \Rightarrow (c_i \Diamond \alpha_i)|_t \wedge \neg (c_i \Diamond \alpha_i)|_{t-1}$. But, at time instant t-1, $\neg (c_i \Diamond \alpha_i)|_{t-1} \Rightarrow (c_i \Diamond \alpha_j)|_{t-1}$. Hence, $c_i \,||\pi^V_{iji} \Rightarrow (c_i \Diamond \alpha_j)|_{t-1} \wedge (c_i \Diamond \alpha_i)|_t \Rightarrow c_i \,||\pi^V_{ji}$ (from definition 4.4). Again, $c_i \,||\pi^V_{ji} \Rightarrow c_i \,||\pi^V_i \cup v_i$ where, $v_i \subset V_i$. As, $(\pi^V_i \cup v_i) \subset V_i$, hence, $c_i \,||\pi^V_{iji} \subset \rho_i$.

Description: The convergence property of VMDFS denotes that based on the circuit in mobility-graph of a client, the migration of distributed pages of a file converges to a vertex in the set of vertices of the corresponding circuit.

4.7. VMDFS Consistency Model

The VMDFS model employs dynamic page-set replication of the files among the servers of SGs based on the access pattern of the files and the hand-off of the mobile clients. A complete file may get replicated dynamically if all the pages of that file are hot for the mobile clients while initiating a hand-off. Instead of employing directory-level lock granularity, the VMDFS design incorporated file-level lock granularity to maintain consistency. The VMDFS follows the concept of Pipeline RAM (PRAM) consistency model with memory coherence where, the write operations of a mobile client on pages are seen in a pipeline by all the other mobile clients accessing the file. For example, if a mobile client updates two pages p1 and p2 of a file in that order then, all the other mobile clients will see either (w1, w2) or (w2, w1). A file in VMDFS may have three read/write consistency states namely, read-consistent state (RC), write-wait-consistent state (WC) and read-write-consistent state (RWC). The read-consistent state denotes that the file is opened in read mode by mobile clients and hence, the cached data is read consistent. The write-wait consistent state of a file indicates that at most one mobile client has opened the file in write mode, however, the update is not propagated to the other mobile clients. In this state, a portion of the file data cached as pages at mobile clients may become stale and needs update propagation. The read-write-consistent state indicates that a mobile client, who is in read-consistent or write-consistent states, has failed or disconnected and hence, the data cache held by such client is stale needing entire cache update. According to the PRAM consistency model, the write updates by a mobile client are

propagated in a pipeline to all the other mobile clients holding the data page in the cache. Whenever a mobile client requests and gets a write-lock of a file from the home server of the file, all the other mobile clients holding the file pages of that file in read mode are notified by the server about the transition of the file from read-consistent state to write-wait-consistent state. The file home server periodically pulls the updated pages from the mobile client holding the write-lock and pushes the updated pages to the other mobile clients. When the mobile client releases the write-lock, the file home server notifies to all other mobile clients about the change of file state from write-wait-consistent state to read-consistent state and the file home server marks internally the file state as read-consistent state. At any time, at most one mobile client can acquire the write-lock of a file and the other concurrent requests to acquire a write-lock will be in the queue. The write-lock requests in the queue are processed in the FIFO order. On the event of detection of failure of a mobile client, the VMDFS server enforces the release of the write lock held by the failed mobile client. This prevents the starvation of the other mobile clients. A failed mobile client, after its recovery, puts itself in the read-write-consistent state and updates the entire cache as well as the current file state. The VMDFS server can detect a potential failure of a mobile client holding a write lock through the non-responding periodic pull of the page updates. The server-side and mobile client-side consistency models are illustrated in Figure 5.

5. Implementation

5.1. Components of VMDFS

The entire implementation of VMDFS is segmented into five parts namely, *(1) Establishing the kernel framework, (2) Realizing the file system structures and handling page-block oriented update logging and data consistency, (3) Handling resource-load balancing, page migration and global page addressing, (4) Periodic update propagation to home server based on periodic-pull model* and, *(5) Realizing security and sharing policies*. This section briefly describes the implementation framework of VMDFS in the Linux kernel 2.4.22. The entire VMDFS system architecture is composed of two components namely, VMDFS client and VMDFS server. The VMDFS client resides in the user-space and corresponding VMDFS server resides in the kernel-space. The VMDFS client communicates with the VMDFS server using the device interface exported by the VMDFS server. The designing of the VMDFS architecture considers the allocation of VMA (Virtual Memory Areas) such that each mobile client will have individual VMA. However, the totally allocated VMA to a mobile client consists of the sum of virtual memory areas allocated separately. Each of the allocated virtual memory area is represented by a file descriptor. Each mobile client is distinguished by the individual client IDs. There are two types of file descriptors allocated to a mobile client. One file descriptor is allocated to a mobile client for accessing the device file (fd_D) and another one is assigned to access the disk file system (fd_F). The fd_D descriptor is used to map page frames in the VMDFS client address space through VMDFS server interface. Based on the mobility, the state of mobile clients and resource-load on server, the page frames are selectively locked forcing the virtual memory system of kernel to make the hot pages memory resident. On the other hand, the fd_F descriptor is used to swap a part of the mapped page frames to local disk file system by the VMDFS client. The components of VMDFS implementation showing the organizational structure of VMA allocation is shown in Figure 6. The advantages of such design are: *(1) selective page-frame locking is employable for active mobile clients achieving load-balancing and reduction in access latency, (2) selective page swap is possible for a client depending on the size of the VMA* and, *(3) the freeing of memory*

can be done by spanning the VMA of multiple clients. Towards the lower end of the implementation, the VMDFS client maintains all the VMA in a list having a root VMA. The root VMA does not allocate any virtual memory area descriptor but points to the list of VMA devices. Each of the VMA devices contains a list of page frame descriptors. Each of the page frame descriptors associated to a VMA device contains a number of page frames and associated data members. In addition to the list of the page frame descriptors, each VMA device contains a reference to the virtual_memory_area data structure of kernel associated to each VMA device. The page fault handler routine of VMDFS server is same for all VMA devices and is installed in the kernel through the virtual_memory_operations data structure of the kernel.

5.2. Network Paging Latency

The network paging latencies of the existing networking technologies are experimentally measured to compare and understand the impact of page migration on VMDFS. The communication technologies chosen for the experimentations are, *1. Wired LAN (100Mbps), 2. Wireless Virtual Private Network (10Mbps)* and *3. Cellular network 2.5G (57.6Kbps GPRS)*. The round-trip latencies are depicted in Table 1. The results indicate that VMDFS is a promising reality considering 2.5G and higher systems having substantially higher bandwidth and reliability as compared to the 2G.

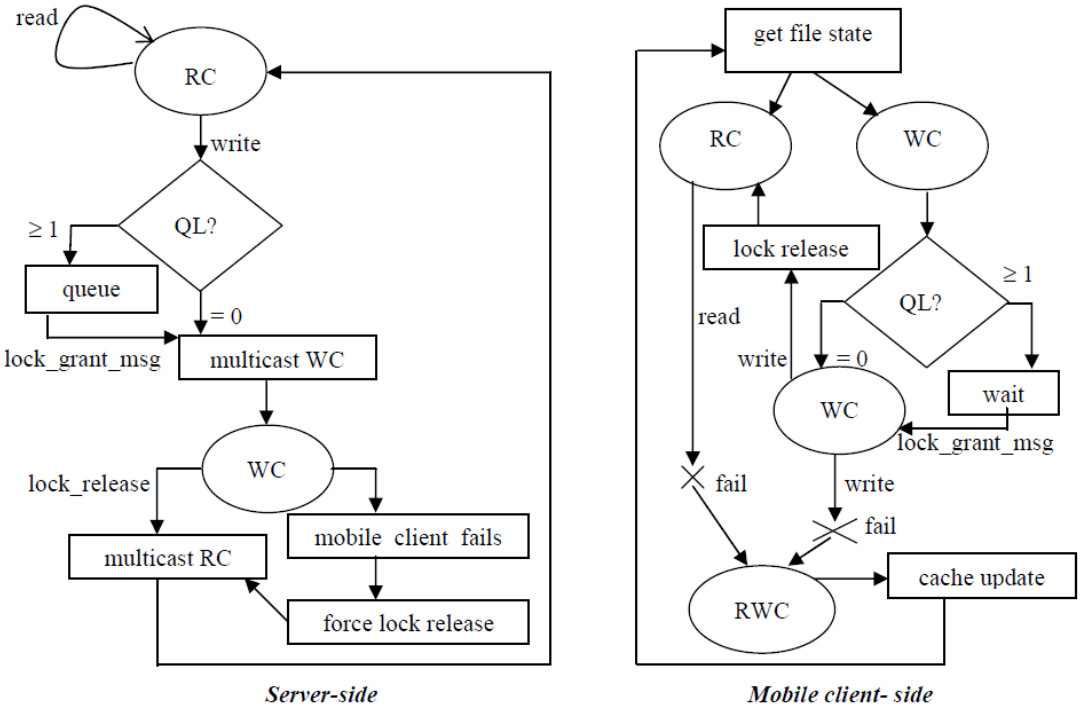

Figure 5. The Server-side and Mobile-client-side Consistency Models, QL : Queue Length.

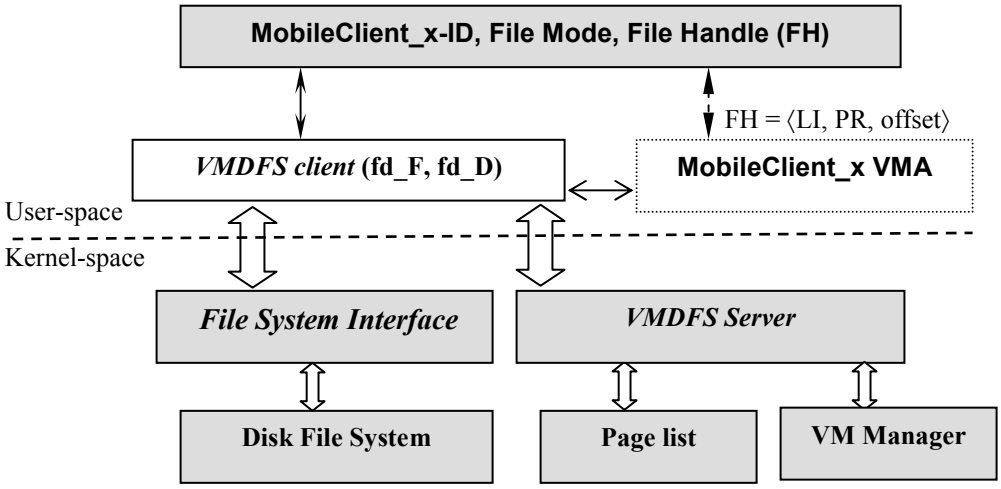

Figure 6. Implementation Framework of VMDFS.

Table 1. Round-trip Network Latency of Page Transfers.

Networks	4KB	8KB	16KB	32KB
Wired LAN	7.06ms	106.7ms	108.4ms	207.35ms
Wireless VPN	290.25ms	312.75ms	354.5ms	346.25ms
2.5G Mobile Network	6.4sec	9.52sec	17.26sec	33.45sec

6. Conclusions

This paper proposes the concept, design architecture, model and an implementation framework of Virtual memory-based Mobile Distributed File System (VMDFS). The VMDFS will allow the resource constrained mobile devices to access remote resources to create, distribute/share and access the digital contents comprised of heterogeneous data under the mobility of clients. The VMDFS may serve as a platform to develop the high-end mobile applications. An abstract mathematical model of VMDFS is constructed and a set of properties is evaluated for detail analysis and verification considering the mobility of clients. The consistency model of VMDFS is illustrated. The VMDFS design incorporates dynamic page-locks and page migration in order to achieve better resource utilization providing high-performance and reduction of network-cost along with data access latency. The monolithic Linux kernel is chosen for experimental prototype development. The round-trip network paging latencies for page-sized file blocks are experimentally evaluated in wired LAN, wireless VPN and 2.5G GPRS. The round-trip network paging latency values illustrate that VMDFS would be a promising reality utilizing the 2.5G, 3G and higher mobile communication systems.

7. References

[1] Plagemann T., Goebel V., Vorsen P. H., *Operating System Support for Multimedia Systems*, Computer Communication Journal, Elsevier Science, 1999.
[2] Poellabauer C. et al., *Kernel Support for the Event- Based Cooperation of Distributed Resource Managers*, Proc. of 8th IEEE Real-Time and Embedded Technology and Applications Symposium, IEEE CS, 2002.
[3] Poellabauer C. et al., *Flexible User/Kernel Communication for Real-Time Applications in ELinux*, Proc. of Workshop on Real Time Operating Systems and Applications, 2000.
[4] Poellabauer C., Schwan K., West R., *Lightweight Kernel/User Communication for Real-Time and Multimedia Applications*, ACM NOSSDAV, 2001.
[5] Poellabauer C. et al., *KECho- Event Communication for Distributed Kernel Services*, In the Intl. Conference on Architecture of Computing Systems: Trends in Network and Pervasive Computing, Springer LNCS, Vol. 2299, 2002.
[6] Honeyman P., Huston L. B., *Communications and Consistency in Mobile File Systems*, CITI technical Report 95-11, Center for Information Technology Integration, University of Michigan, USA, 1995.
[7] Flinn J. et al., *Data Staging on Untrusted Surrogates*, In the Proc. of 2nd USENIX Conference on File and Storage Technologies, USA, 2003.
[8] Magoutis K., *Design and Implementation of a Direct Access File System (DAFS) Kernel Server for FreeBSD*, Proc. of USENIX BSDCon, USA, 2002.
[9] Rangarajan M. et al., *Federated DAFS: Scalable Cluster-Based Direct Access File Servers*, Proc. of 2nd Workshop on Novel Uses of System Area Networks (SAN-2), 2003.
[10] McKusick M., Karels M., Bostic K., *A Pageable Memory Based File System*, Technical Report, Computer Systems Research Group, Computer Science Division, University of California, Berkeley,www.docs.freebsd.org/44doc/papers/memfs.pdf.
[11] Nelson M., *Virtual Memory vs. The File System*, WRL Research Report 90/4, Digital Western Research Lab., USA, 1990.
[12] Nakajima T., Tezuka H., *Virtual Memory Management for Interactive Continuous Media Applications*, Proc. of International Conference on Multimedia Computing and Systems, IEEE CS Press, USA, 1997.
[13] Murphy D., *A Virtual Memory Distributed File System*, Technical Report, Digital Equipment Corp., USA, 1989.
[14] Park Y., Scott R., Sechrest S., *Virtual Memory versus File Interfaces for Large, Memory-Intensive Scientific Applications*, Proc. of ACM/IEEE Conference on Supercomputing, IEEE CS Press, 1996.
[15] Tait C. et al., *Intelligent File Hoarding for Mobile Computers*, Proc. of ACM MOBICOM, 1995.
[16] Policroniades C. et al., *A Data Repository for Fine-Grained Adaptation in Heterogeneous Environments*, Proc. of 3rd ACM International Workshop on Data Engineering for Wireless and Mobile Access (MobiDE), USA, 2003.
[17] Iyengar A. et al., *Design and Implementation of a Secure Distributed Data Repository*, Proc. of 14th IFIP Internet, Information Security Conference, 1998.
[18] Steere D. et al., *A Case for Dynamic Sets in Operating Systems*, Technical Report, CMU-CS-94-216, CMU, 1994.
[19] Renaud L., Christine M., *A Cluster Operating System Based on Software COMA Memory Management*, Proc. of 2nd Workshop on Software Distributed Shared Memory (WSDSM), Santa Fe, 2000.
[20] Satyanarayanan M., *An Agenda for Research in Large-Scale Distributed Data Repositories*, Proc. of International Workshop on Operating Systems of the 90s and Beyond, Springer-verlag LNCS, 1991.
[21] Steinmetz R., *Analyzing the Multimedia Operating System*, Proc. of IEEE Multimedia, Vol. 2, Issue 1, IEEE CS Press, 1995.
[22] Sobti S. et al., *PersonalRAID: Mobile Storage for Distributed and Disconnected Computers*, In the Proc. of 1st USENIX Conference on File and Storage Technologies, Monterey, 2002.
[23] Sobti S. et al., *Segank: A Distributed Mobile Storage System*, In the Proc. of 3rd USENIX Conference on File and Storage Technologies, San Francisco, 2004.
[24] Michalakis N. et al., *Designing an NFS-based Mobile Distributed File System for Ephemeral Sharing in Proximity Networks,* Proc. of 4th Workshop on Applications and Services in Wireless Networks, IEEE CS Press, 2004.
[25] Anderson A. et al., *Serverless Network File Systems*, ACM Transactions on Computer Systems, 1996.
[26] Agarwala S. et al., *System-Level Resource Monitoring in High-Performance Computing Environments*, Journal of Grid Computing, Kluwer Academic Publishers, 2003.
[27] Magoutis K. et al., *Making the Most out of Direct-Access Network Attached Storage*, Proc. of 2nd USENIX Conference ob File and Storage Technologies, San Francisco, 2003.

[28] Bagchi S. et al., *MDVM System Concept, Paging Latency and Round-2 randomized Leader Election Algorithm in SG*, Journal of Advanced Computational Intelligence and Intelligent Informatics, Vol. 10, No. 5, Japan, 2006.
[29] Bagchi S., *Design Architecture and Model of MDVM System*, IFIP International Conf. on Intelligence in Comm. Systems, Springer LNCS, 3283, 2004.
[30] Bagchi S., *MDVM: Architecture, Design Goals and Requirements*, In the Proc. of 2nd International Conference on Advances in Mobile Multimedia, Austrian CS Press (OCG), 2004.
[31] Tait C. D., *A File System for Mobile Computing*, PhD Thesis, Columbia University, 1993.
[32] Moertiyoso N., Yow K., *Designing Wireless Enterprise Applications on Mobile Devices*, In Proc. of International Conference on Information Technology and Applications, Australia, November, 2002.
[33] Liedtke J., *On μ-kernel Construction*, 15th ACM SOSP, Colorado, December 1995.
[34] Morrisett G. et al., *Abstract Models of Memory Management*, Proc. of 7th International Conf. on Functional Programming Languages and Computer Architecture, La Jolla, 1995.
[35] Nutt G., *Operating Systems*, 3rd Edition, Addison Wesley, 2004, pp. 152-154, 773.
[36] Nelson M. et al., *Caching in the Sprite Network File System*, ACM Transactions on Computer Systems, Vol. 6, No. 1, 1988.

Author

Susmit Bagchi

Received B.Sc.(Hons.) from Calcutta University in 1993. He received B.E. and M.E. in Electronics & Telecommunication engineering in 1997 and 1999 from Nagpur University and Bengal Engineering and Science University, respectively. His research interests are in the domain of Operating Systems, Distributed OS/ Systems and Mobile Computing. Currently, he is holding the Technical Lead position at ATD/CS Research Laboratory, Samsung Electronics Ltd. (SISO), India.