Design Architecture and Model of MDVM System

Susmit Bagchi

Department of Computer and Information Science,
Norwegian University of Science and Technology (NTNU)
Trondheim, Norway
susmit@idi.ntnu.no

Abstract. The realization of high-end mobile applications, such as virtual organization (VO), are becoming reality due to the enhanced wireless communication bandwidth, reliability and services offered by WWW. However, the general-purpose operating systems are not completely capable to handle the challenges of mobile computing paradigm. The novel concept of mobile distributed virtual memory (MDVM) extends the server CPU and memory resources to the mobile clients over the mobile communication interface in the VO framework. In this paper, the architecture and associated model of MDVM system are described.

Keywords: Mobile computing, Operating Systems, Virtual Memory, Distributed Computing.

1 Introduction

Mobile computing is becoming a reality due to the availability of portable computing devices having access to WWW [9]. The mobile computing paradigm has created a set of high-end mobile applications such as, m-commerce, SES [6] and virtual organization [7], which need the support of operating system to meet the challenges offered by mobile computing paradigm. Due to the limitation in battery power, mobile devices are limited in hardware resources [11] and operate in doze mode to reduce power requirement [2][10]. The existing wireless communication technology is restricted in terms of bandwidth and reliability [10]. Because of these restrictions, mobile computers limit the nature of user applications [2][10]. The existing operating systems offer very little support for managing and adapting to the mobile computation paradigm [1][3]. The trends in future technological advancements indicate that the wireless communication bandwidth and reliability will be enhanced [6]. Researchers argue that resource-thin mobile computers should utilize remote server based resources in order to support mobile applications [4][5]. It is reported that the remote memory on servers could be utilized in mobile computing system [8]. As a novel approach, the concept of mobile distributed virtual memory (MDVM) is introduced to enable mobile clients exploiting server resources using mobile communication network [13]. The MDVM system will reduce the resource constraints of mobile

© IFIP International Federation for Information Processing 2004

devices to a great extent, and will help to realize a set of mobile applications in virtual organization framework [13]. The MDVM system allows mobile clients to utilize server CPU and memory for data cache and process execution purposes [13]. In virtual organization architecture, mobile clients can utilize server memory as a large and reliable data cache. On the other hand, mobile clients can use server CPU, virtual memory and other devices by initiating remote processes on the server. The prior works on remote paging to realize distributed virtual memory (DVM) aim to utilize the high-speed network for paging rather than the local disk space assuming static network topology. Such assumptions are not applicable to MDVM system. Unlike the monolithic kernel-based MDVM system, the DVM system designs base on the user-space pager server of microkernel architecture. Researchers have directed to modify the kernel of operating system to handle the challenges of mobile computing in an efficient manner [12]. The MDVM design considers the monolithic kernel of a general-purpose operating system. The monolithic kernel architecture offers enhanced performance as compared to the microkernel by reducing context switch frequency, TLB misses and instruction count [15]. The inclusion of MDVM within the kernel offers the benefits of user transparency, performance, flexibility and greater control on system resources. This paper describes the architecture and an abstract model of MDVM system. We are currently designing and implementing the MDVM system in Linux kernel 2.4.22. The paper is organized as followings. The trends in future technological directions in mobile communication and devices along with their limitations are described in section 2. Section 3 introduces the MDVM system architecture. Section 4 describes the model of the MDVM system. Section 5 and 6 state the background work and conclusion respectively.

2 Technological Trends and Limits

The trend in the modern computer industry is to produce mobile computers [14]. Due to rapid development in hardware technology, the mobile computers are getting equipped with powerful microprocessors, high-speed network capabilities and other resources. The next generation mobile communication technology will be 3G and 4G as proposed in industry such as NTT DoCoMo [6]. The proposed capabilities of 3G and 4G mobile communication technologies are described in Table 1 [13]. The 3G and 4G mobile communication technology will offer significantly higher wireless communication bandwidth and reliability as compared to 2.5G [6]. A set of restrictions that would exist in future is: *(1) Limitation of computing time and peripheral devices due to power consumption [8], (2) Unavailability of secondary storage as a peripheral of mobile devices, (3) Performance limitation of FLASH memory, (4) Physical specifications of mobile devices are to be lightweight and small in size,* and *(5) The cost of wireless communication bandwidth is high.* The MDVM concept extends the server resources to mobile clients using mobile communication interface to reduce the resource constraints of mobile devices.

Table 1. 3G and 4G Mobile Communication Technology Capabilities

Technology	3rd Generation (3G)	4th Generation (4G)
Speed	384Kbps – 2Mbps	100Mbps
Applications	Voice, email, video-on-demand, multimedia	Voice, high-speed internet access, high-resolution video, IPv6

3 MDVM System Architecture

3.1 Dynamic Server-Group

The MDVM design concept considers a group of servers connected by high-speed wired network residing within the cells. The servers offer MDVM as services to mobile clients through the wireless network. The MDVM servers form a server-group (SG) as shown in Figure 1. Individual servers in a SG serve the mobile clients in the corresponding cell. The members of a SG elect a leader and co-leader to perform the periodic tasks in a SG. Such tasks are comprised of keeping the overall status of the SG-members, resource allocation history and to keep the replica of storage segments of MDVM located on other members of SG as a fault tolerance mechanism. The co-leader assumes the responsibility of the leader if the leader crashes. This eliminates the necessity of "*stop and leader-reelection*" in the system at any point of time due to leader crash. On the other hand, the co-leader shares the load on a leader eliminating the problem of central point of load concentration along with the saturation of network bandwidth around the leader. The members of a SG are required to handle the memory migration from another SG requiring inter-SG communication system. The 4G mobile networks standard supports protocol conversion techniques. This enables the two SGs to communicate using TCP/IP protocol stack over the high bandwidth wired link.

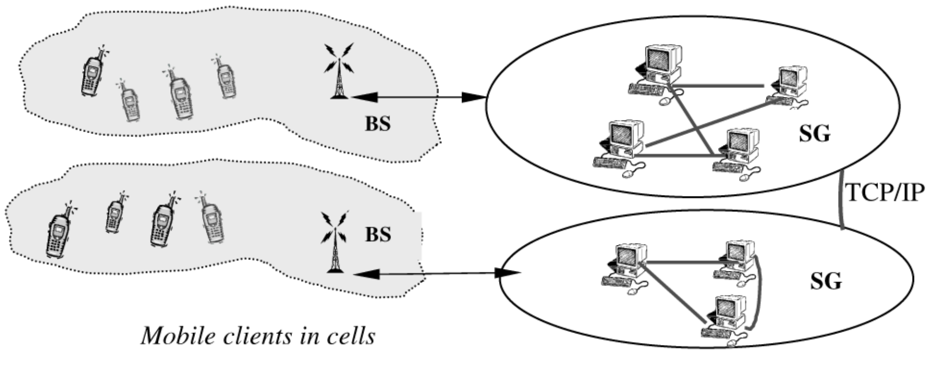
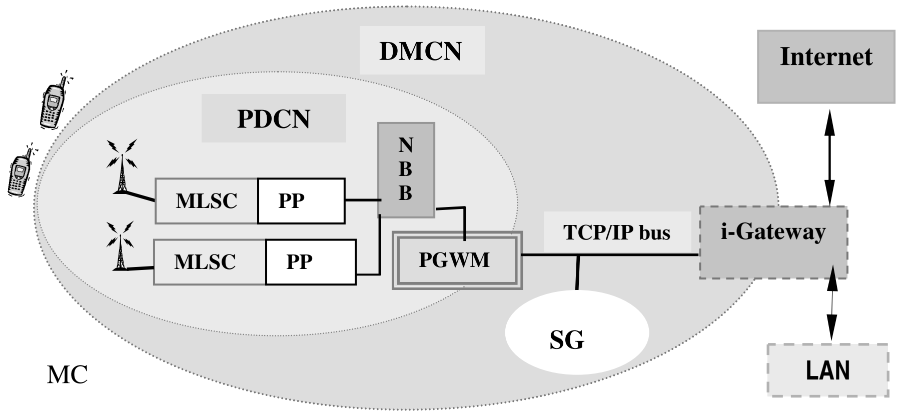


Fig. 1. The Server Groups and Mobile Clients



PDCN: *Packet Data Communication Network*, MLSC: *Mobile Local Switch Center*, PP: *Packet Processor,* NBB: *Network Backbone*, PGWM: *Packet Gateway Module.*

Fig. 2. The SG Embedded into Cellular Network

3.2 SG in Cellular Network

The architecture of mobile communication network containing MDVM servers is shown in Figure 2. The system architecture is consisting of mobile clients (MC) and stationary servers. The stationary servers of SG are placed in the Digital Mobile Communication Network (DMCN) on the TCP/IP bus. According to the IMT-2000 standard [6] the DMCN has capability to support TCP/IP network stack and protocol conversions. The servers in SG are interconnected by high-speed wired TCP/IP network and reside in between i-Gateway and PGWM. The PGWM interfaces TCP/IP bus and NBB. The job of NBB is to support the basic cellular communication system backbone. The data packet processing in the mobile communication system is handled by PP and MLSC. The BS in a cell is connected to NBB via MLSC. In 3G and higher mobile communication standard, the DMCN has capability to interface the Internet servers using special server i-Gateways using TCP/IP protocol. Hence, the MCs may communicate to the MDVM servers, corporate LAN and web servers via DMCN using the TCP/IP interface controlled by PGWM. Similarly, the servers in SG can reach corporate LAN and Internet via i-Gateway interface. The inter-SG communication is based on TCP/IP interface controlled by PGWM.

4 Placing MDVM in Monolithic Kernel

The MDVM is designed as a dynamically loadable kernel-module functioning as a kernel subsystem. No special hardware is involved with the design of MDVM system. MDVM device architecture creates the infrastructure within the monolithic kernel to exploit CPU and memory resources by offering them to the mobile clients. This indicates that MDVM, as a software subsystem, can be treated as a capability enhancement of a monolithic kernel to suit it in the mobile computing paradigm. The architecture of a typical monolithic kernel containing MDVM system is shown in Figure 3.

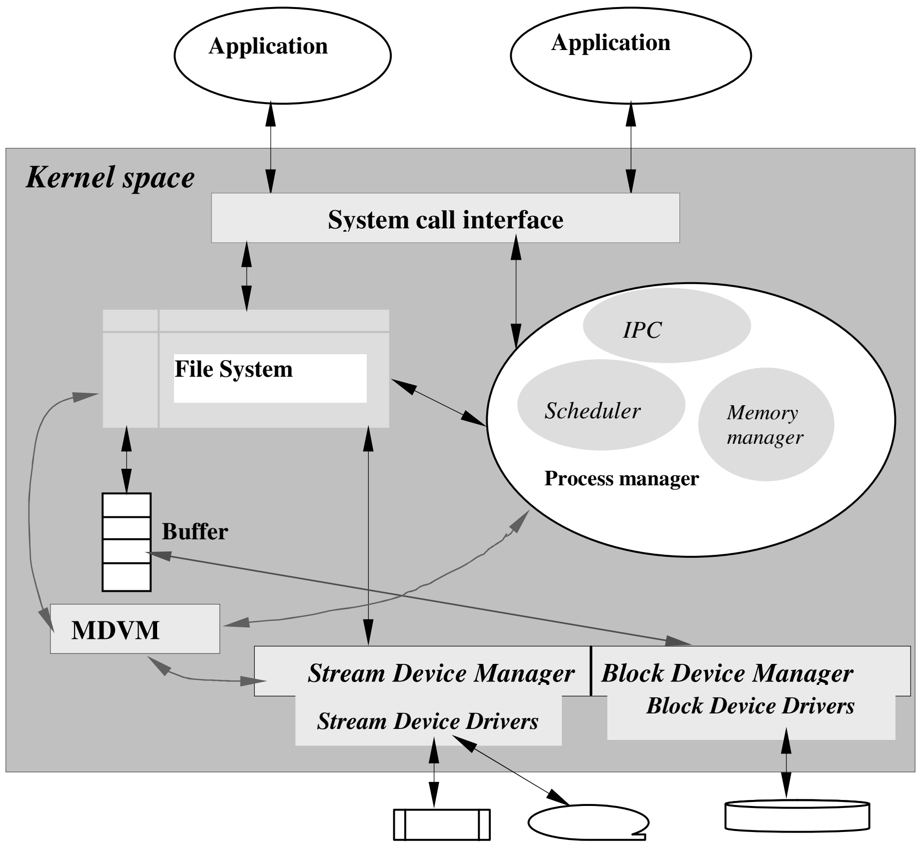


Fig. 3. MDVM System within a Monolithic Kernel

The mobile clients may need small amount of data access such as a byte or they may need to access larger chunk of data such as multiple pages. This requires MDVM system having capability of byte addressing and page addressing. A byte is considered as 8bit data and the memory page size is typically considered as 4KB. According to the concept of server-group, a set of resource-fat servers resides within each cell of mobile communication architecture connected by high-speed wired network supporting TCP/IP protocol stack. The MDVM system software module is placed within the kernel of all the servers offering MDVM services. The MDVM modules, as active entities in a cell, form the logical ring topology for communication and resource management purposes as shown in Figure 4. The mobile clients may reach the MDVM modules utilizing interface of protocol conversion supported by 3G and 4G communications standard. On the other hand, a MDVM module can communicate to other MDVM module using TCP/IP protocol over wired network. The messages among MDVM modules and mobile clients are mainly comprised of two types. These are the short message data packets and relatively large memory pages. The MDVM system servers consider the page size of 4KB. This is because the maximum transmission unit (MTU) of the packets over TCP/IP Ethernet link is in the order of 1.5KB. The larger the page size, the more are the fragmentation and joining of packets at source and destination respectively. This will lead to the increased network traffic, bandwidth saturation and computation overhead at source and destination. The MDVM modules residing in the kernel of servers elect a leader and a co-leader. This indicates that the MDVM modules in a server-group create three classes: ***leader MDVM, co-leader MDVM*** and ***member MDVM*** as shown in Figure 4.

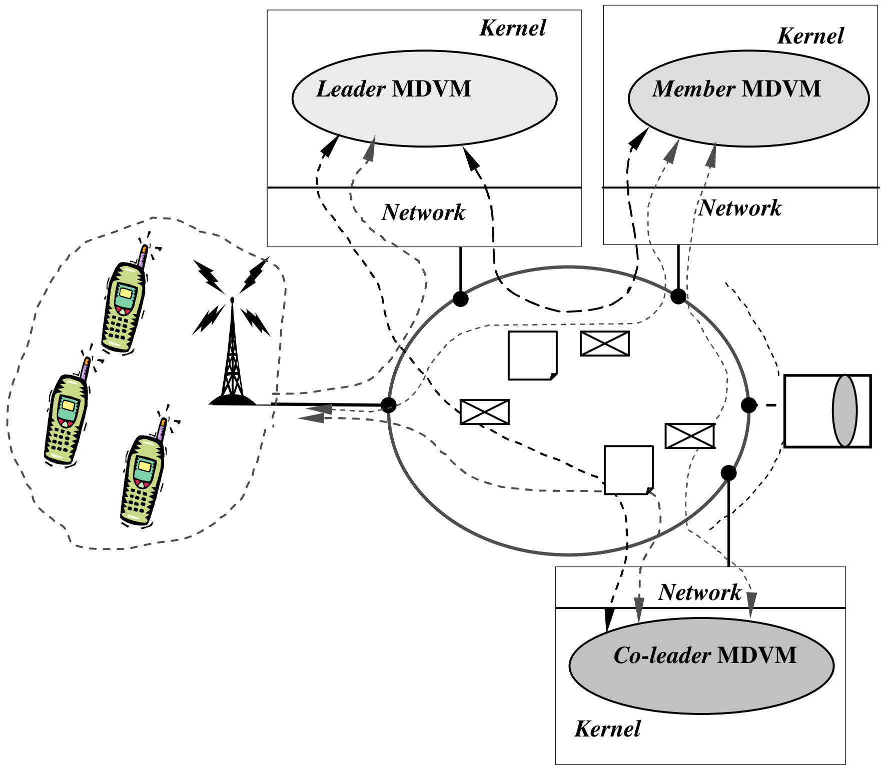


Fig. 4. MDVM System Network

The member MDVM modules offer the memory and CPU resources of the local servers to the mobile clients. The leader and co-leader MDVM modules maintain the replica of storage segment pages of MDVM offered by the member modules on the local disk. In addition, the leader module and co-leader module maintain the information about the operational status of the ring such as, number of active servers in a ring and the resource-loads on the servers. The update of such information is done periodically by *members-initiated* manner. The co-leader MDVM module of a ring automatically assumes the responsibility of the leader on the detection that the leader MDVM server is crashed. This will eliminate the necessity of complete system halt in order to elect a new leader in the ring on the face of leader failure. The election of the new co-leader will be started as soon as the former co-leader will assume the job of the leader. The election of a new co-leader can be done concurrently without halting all the members in the ring. In addition, the leader module diverts a part of the traffic to the co-leader if the load on the leader crosses a certain pre-computed threshold. This in a way increases the data cache availability due to the fact that the co-leaders also maintain a copy of the replica of cache pages handed over by the leader periodically. The communication paths involved in the MDVM system architecture can be between the two MDVM modules in the ring or between a module and the mobile clients as shown in Figure 4.

4.1 Required Kernel Subsystems

The MDVM system offers CPU and memory resources of the server to the mobile clients. In order to utilize memory efficiently, the MDVM system swaps cache pages to the local disk according to the page access patterns of mobile clients. It is better to swap the cache pages held in RAM on the local disk in order to free the page frames if the mobile clients holding such cache are inactive for sufficiently long time. The MDVM module uses RAM, CPU, network device and disk drive of a system in order to perform the required tasks. The kernel of the operating systems running on mobile devices map a part of the virtual memory as remote pages accessible through wireless communication network. The monolithic kernel of the server operating system can be subdivided in a number of subsystems: memory manager (MM), file system (FS), scheduler, process dispatcher (Exec), block device manager and stream device manager. The MDVM system module is placed within the kernel of the server operating system having interfaces to MM, FS, scheduler and network device. The access to the network device is designed by using the stream device manager interface because the network device is a class of stream device. The diagram showing interfaces between MDVM subsystem and the other subsystems of a monolithic kernel is depicted in Figure 5. The MDVM subsystem maintains the interface with the process scheduler in order to estimate CPU load, schedule a remote process execution and control or monitor the state of execution of remote processes started by mobile clients. The estimation of memory-load is done periodically. The interface with memory manager is used to allocate page frames to the mobile clients for the data cache. Because of the higher disk access latency, the cache pages of active clients will be prevented from being swapped out by the virtual memory manager of the kernel. The swapping of cache pages are done by the MDVM system based on the page access pattern of mobile clients. Hence, the interface to the local FS is required by the MDVM system in order to manage the swap space on local disk separately from that of the swap space managed by the virtual memory manager of kernel. The interface between the network device and MDVM will be required to maintain network communication activities using TCP/IP on wired network. The wired network to the wireless network protocol conversion is handled by the cellular communication infrastructure.

4.2 Modelling MDVM Architecture

The memory management system of the operating systems can be modelled with abstract and precise formalism in order to represent the design architecture [39][40]. 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 [38][39][40]. In this section, the abstract model of the MDVM system architecture is formulated. One of the main components of the MDVM system architecture is the server group (SG). Each server in a SG offers a set of pages from its virtual address space to the mobile clients in order to fulfil the storage and execution space needs of the clients. Let, the set of server groups in a MDVM system architecture is represented by S_g such that, $S_g = \{g_1, g_2, \ldots g_b\}$. There is no constraint on the $|g_i|$ meaning that $\forall g_i, g_j \in S_g$, either $|g_i| = |g_j|$ or $|g_i| \neq |g_j|$. Let z be a MDVM server and $z \in g_i$. Let $V_z = \langle 0, 1, 2, \ldots, G\text{-}1 \rangle$ is the set of virtual addresses available at z. The virtual address spaces can be segmented or paged. The MDVM system design model considers the paged virtual memory management system. A page p_z is consisting of a set of addresses $E_z = \{e_j \mid 0 \leq j < |p_z|\}$ residing at z such that $p_z \subset V_z$. Let n_z represents the total number of virtual memory pages available at z. The page frames of a system are generally numbered and indexed in the page frame table. If all the page frames of z are numbed by f then, $\forall e_j \in E_z$, $e_j = f.\,|p_z|+q$, $0 \leq q < |p_z|$ and $f = 0, 1, 2, \ldots, h$. Hence, a page p_z residing at z can be given by the ordered pair $\langle f, q \rangle$. The address map is the function to translate the virtual addresses into the physical memory addresses. Let β_t is such a function translating the memory addresses for the entire time space t. Then, β_t can be computed as, β_t: $v_z \rightarrow \langle f, q \rangle \cup \{\phi\}$ where $v_z \subseteq V_z$. The set of page frames residing at z can be given as, $P^z_M = \{p_w \mid 0 \leq w \leq m\text{-}1, m>0\}$. According to the definition of the virtual memory, the set of virtual address pages available at z can be computed as, $P^z_V = P^z_M \cup S_z$ and $|P^z_V| = n_z$. The set of pages $S_z = \{d_0, d_1, \ldots, d_{u-1}\}$ represents the swap space at MDVM server z, where $u = n_z - |P^z_M| + 1$. If the function $r(g_i)$ computes the total amount of virtual memory resource of the server group $g_i \in S_g$ then, $r(g_i) = \cup_{i=1,c} P^i_V$ such that $c = |g_i|$.

A MDVM server of a SG residing in a cell is capable to handle multiple mobile clients in the corresponding cell. Let C_z represents a set of mobile clients that are using MDVM at the server z and k is such a mobile client, $k \in C_z$. The MDVM request from k consists of process execution request and the request for data cache pages. The MDVM request from k is composed of $\langle \alpha^r_k, v^c_k \rangle$, where α^r_k is the set of virtual address spaces of the requested execution of the remote process r and v^c_k is the

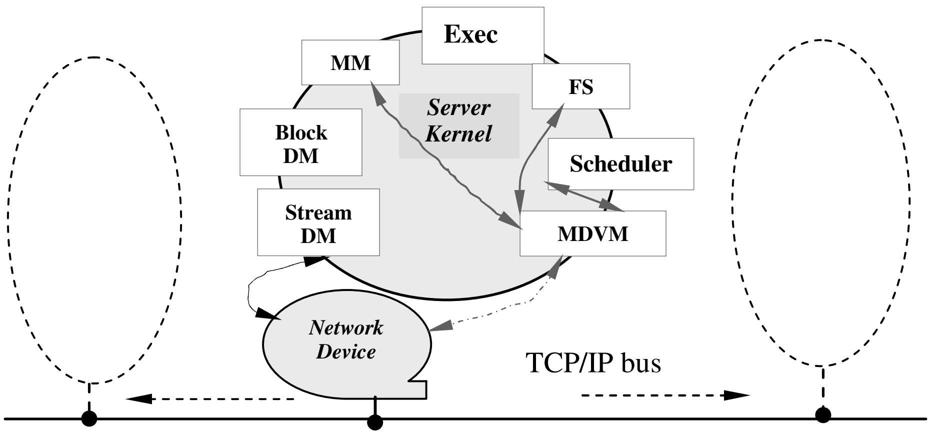


Fig. 5. Interfaces between MDVM Device and Monolithic Kernel Subsystems

set of requested virtual pages for data cache. The architecture of MDVM system is comprised of two components. These are a kernel-stub (D) and a kernel-server (T) as shown in Figure 6. The D and T have interface among each other and can avail services from kernel through the kernel interface. The MDVM system can be modelled as a list of mapping functions given by $\langle \delta_D, \{\delta^P{}_T, \delta^C{}_T\}\rangle$. The δ_D and $\delta^C{}_T$ map the data cache pages for a mobile client and the $\delta^P{}_T$ maps the virtual addresses for process execution on MDVM servers as requested by the client. The definitions of the δ_D and $\delta^P{}_T$ functions are given as, $\delta_D : P^z{}_V \rightarrow P^z{}_M$ and $\delta^P{}_T : P^z{}_V \rightarrow (\alpha^r{}_k \text{ x } P^z{}_V) \cup \{\phi\}$ such that, $\beta_t(\alpha \subseteq \alpha^r{}_k) \in p_z$ and $p_z \in P^z{}_M$. Let $B \in [0, 1]$ represents the state of the set of cache pages of k maintained at z. The values in B indicate whether the data cache pages are swappable or not. The values in B make it possible to either realize a binary logic or a multi-valued logic to recognize the state of the cache pages. Suppose, the function $f^C{}_T$ is performed by T for the cache pages such that $f^C{}_T : v^c{}_k \rightarrow (\{\delta_D(v^c{}_k)\} \cup L) \text{ x } B$, where $L \subset P^z{}_M \cup \{\phi\}$. The symbol L signifies a set of page frames that are mapped by D but not used by T. Hence, L can be considered as a list of mapped free page frames. The list is maintained in order to enhance the memory utilization by keeping track of the previously mapped but unused page frames, if any. Suppose, at time instant t, $L \neq \phi$. This indicates that in the next time space t+, the memory map performed by δ_D and the residue in L cannot be overlapped, i.e. $\{\delta_D(v^c{}_k)\} \cap L = \phi$. Due to the mobility and limitation of battery power of clients or due to the change in the state of the wireless communication, a mobile client may go into doze mode or may get disconnected from the MDVM server z. Let H_z is the set of disk swap spaces at z maintained by T, $H_z \not\subset S_z$. Then, the function $f^S{}_T$ is defined as, $f^S{}_T : P^z{}_M \text{ x } B \rightarrow H_z$. Thus, the cache page map of MDVM system, $\delta^C{}_T$, can be realized as a composition of functions given by $(f^S{}_T \circ f^C{}_T)$. In the time space t, the virtual memory $v^c{}_k$ may reside in page frames or in disk swap space depending on the state of the mobile client.

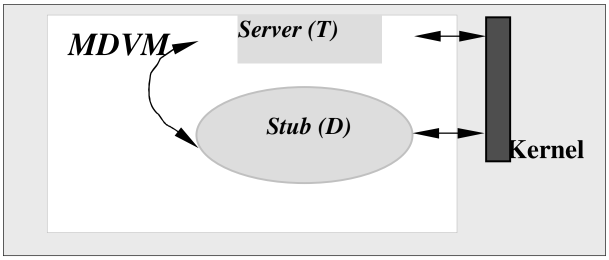


Fig. 6. MDVM Architecture Model

4.3 Memory Utilization Versus Page Fault Frequency

The page fault is a phenomenon that occurs when a process tries to access a page not resident in main memory. The mobile clients in C_z can request a set of pages for the data cache. Let, n is the number of pages requested by a mobile client. If n = 1, then a single page fault will be enough to allocate a page frame to the client. However, for n>1 there may exist two possible outcomes. Suppose, $m = \log_2 n$ for some $m \in Z$, m>0. In this case, mapping of n pages will incur n page faults if the memory manager allocates a single page frame at a time. However, the modern operating systems allow the buddy allocation algorithm where n number of such page frames would be allocated in a single page fault. On the other hand, if $p = \log_2 n$ and $m = \lceil p \rceil$ where $p \in R^+$, then allocating n pages one by one will still incur n page faults. On the contrary, use of buddy algorithm will create the possibility of memory under-utilization as $n<2^m$. The amount of under-utilization is $(2^m - n)$. The list L can hold $(2^m - n)$ pages for future use. The page fault frequency (PFF) is considered as the deterrent to the overall system performance. Hence, to reduce PFF to a lower level, δ_D may map 2^m page frames, and T may maintain list L. The T may try to utilize the page frames from L in subsequent future requests until $L = \phi$. It is interesting to note that if $|L|$ becomes very large, then it may lead to considerable amount of the memory waste although the PFF will be reduced substantially. Hence, the memory management algorithm employed in MDVM system should try to balance the reduction of PFF and enhancement of the memory resource utilization by keeping $|L|$ as much low as possible while reducing PFF.

5 Related Work

Prior works have addressed the issues in operating system related to mobility in the area of file systems [16][17][18], data management [19][11][20][21] and network-layer routing protocols addressing schemes and packet filtering [22][23][24]. Other works include issues related to caching and file system [16][18][25][26] and mobile communication system [27][28][29]. The existing DVM system [30] does not provide an easy way to dynamically use and share DVM resources preserving transparency, adaptability and extensibility [31]. The DVM system becomes non-scalable under the condition of mobility of clients. The DVM system design bases on the external pager-server in the microkernel. The remote paging concept employed in DVM assumes that network bandwidth is high and network topology is static and both the clients/servers are equipped with adequate RAM and disk drive. Such assumptions are not applicable in the MDVM system design. The bandwidth of existing disk drives is lower than that of high-speed network [14][32][33]. The aim of existing DVM system is to investigate the performance gain through remote paging over high-speed network. Hence, the performance of existing DVM systems is controlled by the network performance [32]. The majority of the remote memory paging system ([32][33][34][35]) and the DVM system ([30][31][37][34][36]) target to the stationary client-server architectures on wired LAN. However, the issues related to the location transparency of the remote memory under mobility, virtual memory management under dynamic memory-pressure and virtual memory migration among servers are not investigated. The concept of MDVM extending the server resources to the mobile clients did not get much attention. It is important to make operating system kernel "mobility aware" to tackle challenges of mobile computing. Unlike microkernel, the monolithic kernel offers enhanced overall system performance.

6 Conclusion

The MDVM system is intended to meet the resource constraints of the mobile devices by extending server resources to the mobile clients over wireless communication network. The system architecture, placing the MDVM servers in mobile communication framework, is outlined. The designing of MDVM in monolithic kernel will provide user transparency, performance, greater control on system resources and required system services. The required subsystems of a monolithic kernel in order to design MDVM as a software module are outlined. An abstract model of the MDVM system architecture is constructed. The Linux operating system is chosen for building experimental prototype of MDVM system because it is a monolithic kernel, free source and comparable to other commercial operating systems in terms of efficiency. We are currently implementing various building blocks of MDVM system in Linux kernel 2.4.22.

References

[1] Black A., Inouye J., System Support for Mobility, ACM SIGOPS, Ireland, 1996.
[2] Duchamp D., Issues in Wireless Mobile Computing, 3rd Workshop on Workstation OS, 1992.
[3] Bolosky W. et. al., OS Direction for the Next Millennium, Microsoft Research, Redmond.
[4] Forman G., Zahorjan J., The Challenges of Mobile Computing, UW CSE TR#93-11-03, 1994.
[5] Marsh B. et. al., Systems Issues in Mobile Computing, MITL-TR-50-93, Princeton, 1993.
[6] Nadia M., Kin Y., Designing Wireless Enterprise Applications on Mobile Devices, ICITA 2002.
[7] MOWAHS, IDI, NTNU, 2003, www.mowahs.com.

[8] Shigemori Y. et. al., A proposal of a Memory Management Architecture for Mobile Computing Environment, IEEE DEXA, 2000.

[9] Weiser M., Some Computer Issues in Ubiquitous Computing, ACM Communications, 1993.

[10] Pitoura E. et. al., Dealing with Mobility: Issues and Research Challenges, TR-CSD-93-070, 1993.

[11] Badrinath R. et. al., Impact of Mobility on Distributed Computations, ACM OS Review, 1993.

[12] Bender M. et. al., Unix for Nomads: Making Unix Support Mobile Computing, USENIX, Mobile & Location-Independent Computing Symposium, 1993.

[13] Susmit B., Mads N., On the Concept of Mobile Distributed Virtual Memory System, IEEE DSN, International Conference on Dependable Systems and Networks, Italy, 2004.

[14] Schilit B., Duchamp D., Adaptive Remote Paging for Mobile Computers, TR-CUCS-004-91, Columbia University, February 1991.

[15] Chen B., The Impact of Software Structure and Policy on CPU and Memory System Performance, PhD Thesis, CMU-CS-94-145, 1994.

[16] Tait D. et. al., Detection and Exploitation of File Working Sets, TR-CUCS-050-90, Columbia, 1990.

[17] Kistler J., Satyanarayanan M., Disconnected Operation in the Coda File System, ACM Transactions on Computer Systems, February, 1992.

[18] Tait D., Duchamp D., Service Interface and Replica Management Algorithm for Mobile File System Clients, 1st International Conference on Parallel and Distributed Information Systems, 1991.

[19] Badrinath R., Tomasz I., Replication and Mobility, In Proc. Of 2nd IEEE Workshop on Management of Replicated Data, November 1992, pp. 9-12.

[20] Alonso R., Korth H., Database System Issues in Nomadic Computing, MITL, December 1992.

[21] Tomasz I., Badrinath R., Querying in Highly Mobile Distributed Environments, In 8th International Conference on Very Large Databases, 1992, pp. 41-52.

[22] Ioannidis J., Duchamp D., Maguire G., IP-Based Protocols for Mobile Internetworking, ACM SIGCOMM, September 1991, pp. 235-245.

[23] Wada H. et. al., Mobile Computing Environment Based on Internet Packet Forwarding, In Winter USENIX, January, 1993.

[24] Zenel B., Duchamp D., Intelligent Communication Filtering for Limited Bandwidth Environments, IEEE 5th Workshop on HotOS-V, May 1995.

[25] Mummert L. et. al., Variable Granularity Cache Coherence, Operating Systems Review, 28(1), 1994, pp. 55-60.

[26] Mummert L., Exploiting Weak Connectivity in a Distributed File System, PhD Thesis, CMU, 1996.

[27] Lin C., An Architecture for a Campus-Sized Wireless Mobile Network, PhD Thesis, Purdue, 1996.

[28] Lee J., Routing and Multicasting Strategies in Wireless Mobile Ad Hoc Network, PhD thesis, California, 2000.

[29] Akyol B., An Architecture for a Future Wireless ATM Network, PhD Thesis, Stanford, June 1997.

[30] Khalidi Y. et. al., The Spring Virtual Memory System, Sun Microsystem Lab., TR-SMLI-93-9, February 1993.

[31] Ballesteros F. et. al., Adaptable and Extensible Distributed Virtual Memory in the Off Microkernel, TR-UC3M-CS-1997-02, Madrid, January 1997.

[32] Markatos E. et. al., Implementation of a Reliable Remote Memory Pager, In Proc. Of USENIX, San Diego, January, 1996.

[33] Liviu I. et. al., Memory Servers for Multicomputers, In Proc. Of 38th IEEE COMPCON, Spring 1993.

[34] Feeley M. et. al., Implementing Global Memory Management in a Workstation Cluster, 15th SOSP, 1995, pp. 130-146.

[35] McDonald I., Remote Paging in a Single Address Space Operating System Supporting Quality of Service, Technical Report, Glasgow, October 1999.

[36] McKusick M. et. al., A New Virtual Memory Implementation for Berkeley UNIX, Computer Systems Research Group, Univ. of California, Berkeley, 1986.

[37] Ballesteros J. et. al., An Adaptable and Extensible Framework for Distributed Object Management, In Proc. of ECOOP'96, Workshop on Mobility and Replication, Austria, 1996.

[38] Gary N., Operating Systems, 3rd Edition, Addison Wesley, 2004, pp. 152-154, 773.

[39] Liedtke J., On μ-kernel Construction, 15th ACM Symposium on Operating Systems Principles, SOSP, Colorado, December 1995.

[40] Greg M. et. al., Abstract Models of Memory Management, In the Proc. Of 7th International Conference on Functional Programming Languages and Computer Architecture, La Jolla, 1995.