PATHWORKS for VMS File Server

By Edward W. Bresnahan and Siu Yin Cheng

1 Abstract

The PATHWORKS for VMS file server integrates industry-standard personal computers with VAX VMS systems over a communications network. It implements Microsoft's server message block (SMB) core protocol, which provides resource sharing using a client-server model. The server provides transparent network access to VAX VMS FILES-11 files from a PC's native operating system. The architecture supports multiple transports to ensure interoperability among all PCs connected on an open network. Due to the performance constraints of many PC applications, data caching and a variety of other algorithms and heuristics were employed to decrease request response time. The file server also implements a security model to provide VMS security mechanisms to PC users.

2 Introduction

Coupled with the PATHWORKS for DOS or PATHWORKS for OS/2 product, PATHWORKS for VMS creates a distributed computing environment, based on a client-server model. This environment allows personal computer (PC) users to access VMS system resources transparently. PC clients access the system server from their native operating systems, typically MS-DOS, as if it were local to the PC. The VAX VMS system resources to be shared, i.e., files or printers, are offered as services over the network to PC clients. The computer systems providing the shared resources are referred to as servers; and the PCs requesting the resources as clients. The SMB protocol from the Microsoft Networks/OpenNET (MS-NET) Architecture was chosen to provide file sharing from a VAX VMS system to MS-DOS and OS/2 clients.[1] The SMB protocol is a command/response application-layer protocol designed to provide file sharing in a PC network. Since SMB is an application-layer protocol, it is transport independent and thus can be implemented over heterogeneous networks.

Central to this environment is the file server, the component that processes the SMB requests to provide file and print sharing along with management functions. The file server maps SMB file requests to the appropriate calls for the VAX VMS FILES-11 file system interface and honors applicable security mechanisms. MS-DOS and VAX VMS systems have different file systems and security models. To integrate these different environments, mapping policies, along with an architecture appropriate for the VMS system, had to be developed and implemented.

This paper describes the design and implementation of a nondedicated personal computer file server (PCFS) on a VAX VMS computer system. It details the PATHWORKS for VMS file system and discusses its transport layer interface and performance considerations, including data caching effects and disk space allocation. The paper then explains file sharing among server processes in a cluster environment and concludes with a discussion of the server configuration and management interface.

3 File Server Architecture

The file server is implemented as a single, multithreaded, nonblocking detached process with an associated permanent DECnet object. This user-mode process is privileged and has a high priority. Figure 1 shows the architecture of the server. Only one file server process exists on any one computer to handle all client requests. An alternative choice would be to have multiple processes service the clients. The use of a single process reduces system resource requirements and eliminates the latency that is incurred from context switches among the multiple server processes. Also eliminated is the latency that results from process creation at the time a client connects.

A threads package with multiple independent threads of execution within a single process supports multiple clients and periodic operations within the file server. The file server creates a thread for a client when it requests establishment of a virtual circuit to the file server. The thread is deleted when the client terminates its connections. A client's thread carries out the operation specified in the request SMB without blocking the process. With this scheme, processing SMB requests is synchronous with respect to the client, yet asynchronous with respect to the file server process.

Since a server process may be processing the requests of hundreds of clients simultaneously, the server operates in real-time. The threads package contributes to these goals by providing an environment in which the process never enters a wait state and a client thread is safe from CPU starvation. Preventing the process from blocking is accomplished by performing all file I/O asynchronously and by calling operating system routines asynchronously when possible. Starvation is prevented by scheduling clients using a nonpreemptive first-in, first-out (FIFO) scheduling algorithm. With this policy, a thread executes until it voluntarily yields, usually due to an I/O operation or an operating system call. Using a nonpreemptive scheduling algorithm also eliminates the latency that would result from a thread switch in a preemptive environment.

4 Pathworks File System

A file server needs to provide transparent file access to a VMS file system and ensure file accessibility between DOS and VMS users. Since these operating systems have different file systems, PATHWORKS for VMS must store the files in VAX VMS FILES-11 format and provide a mapping algorithm to bridge the two operating systems. Because the OS/2 and DOS systems use the same file system, the mappings performed to address the difference between the DOS and VMS systems can be applied to support transparent file access from an OS/2 client.

File Name Mapping

DOS and VMS FILES-11 support different naming syntaxes. DOS supports 8.3 naming format; that is, the file name is composed of a maximum of eight characters with a maximum of three characters as the extension. In contrast, the VMS FILES-11 file name supports 39.39 format and includes a third component, the file generation number. In addition, the legal character set for a file name is larger in DOS than it is in the VMS system.

The PATHWORKS file server does not include a mapping algorithm to convert a 39.39 VMS file naming syntax to be accessible to DOS. Any VMS file that DOS system users need to share must be created with a file name that conforms to DOS 8.3 format. Since the 8.3 naming format maps directly to the 39.39 format, no mapping algorithm is required to guarantee a VMS system user access to files named by a DOS system user.

To overcome the difference in character sets, a comprehensive mapping algorithm was written to ensure shareability and transparency. Since neither operating system is case sensitive, the file server changes the file name to uppercase before any operation is performed on the file. The legal character set for VMS FILES-11 file names includes uppercase alphanumerics, dollar sign, hyphen, and underscore. The character set in DOS includes all noncontrol characters with the exception of a few special signs. The PATHWORKS server maps the character sets based on the following rules:

- All alphanumeric characters are changed to uppercase letters; any character that is valid in a VMS file name is passed through unchanged.

- All other characters are changed to two underscores, followed by two hexadecimal digits that represent the ASCII code of the character being mapped.

VMS FILES-11 allows multiple versions of a file to be generated and stored in a directory. These files are identified by the numeric component, which represents the version number, of a file name. There is no equivalent concept in the DOS system. The PATHWORKS server maps the highest version (or most recent generation) to be accessible to DOS. Similarly, the server, when creating a file on behalf of a DOS client, generates the file with a version limit of 1. To preserve and honor the version limit information for the VMS environment, the server preserves the VMS file attributes of previous versions of the file. Consequently, if the file is created by a VMS user, and is later updated by a DOS user, a new version of the file is generated, and the version limit information is preserved.

Directory Mapping

The VMS system requires a directory name to end with "dir" as an extension, but the DOS system does not post any restriction in this area. PATHWORKS maps directory names in DOS by including the ".ext" characters as part of a directory name. Since the period is not a legal character for a DOS directory, it is mapped using the double underscore followed by the hex digit rule. Any directory name in DOS that conforms to the VMS directory naming syntax is passed through untouched.

DOS File Attribute Mapping

Both file systems associate a set of attributes to the files, but the file attributes on a DOS file do not have a one-to-one correspondence with those on a VMS file. A DOS file has four types of file attributes: archive, system, hidden, and read-only. The concepts of archive, system, and hidden are not recognized in the VMS file system. PATHWORKS software stores the DOS file attributes in an application access control entry when creating a file on behalf of a PC workstation. Furthermore, the read-only attribute of a DOS file is mapped to the read-only bit of the record management services (RMS) protection field for system, owner, and group.

File Organization

A DOS file is organized as a byte stream, but a VMS file is organized as collections of records. Although the VMS system supports a form of stream file, most VMS files are stored in record format. Furthermore, a VMS file with a stream record format does not map directly to a DOS stream format. This poses an interesting problem in integrating VMS and DOS file systems.

Since PATHWORKS software provides transparent access to the VMS host system, a DOS client views all files on file services as streams of bytes, just as if these files were stored locally. When the server creates a file on behalf of a PC, it specifies the file organization as sequential with stream record format. Thus, the byte stream characteristic of the DOS system is preserved.

The more complex part of the problem is to resolve the shareability issues between VMS and DOS applications. The PATHWORKS server is implemented to provide the necessary conversion between VMS and DOS file organization on stream files. The file server views a file as stream if it can read and write the file without regard to any record boundaries. This includes any files with file organization as sequential and record format as stream, stream_cr, stream_lf, and undefined, as well as fixed. If a sequential file has fixed record format, it must conform to record size and attributes as follows: even with no record attribute; 512 with no block_span; and power of 2 with no block_span. Thus, an RMS overhead in reading and writing these files is avoided.

Any file that does not meet the criteria of the stream category is said to be nonstream. The PATHWORKS server provides read-only access to any VMS nonstream file. This is achieved by using a VAX C run-time library call that provides stream file semantics and a conversion algorithm to properly map any carriage return and line feed information. The file server cannot support writing to these files because the SMB protocol does not preserve record boundary information. Thus, the protocol makes it impossible for the file server to guarantee data integrity when updating a nonstream file.

Byte Range Locking

The MS-NET architecture allows for concurrent access to server-based files by multiple clients. PC applications acquire this functionality through the MS-DOS byte range locking calls. These calls allow PC applications to lock and unlock ranges of bytes in a file and to detect conflicts. Conflicts occur when part or all of a range specified to be locked has been locked from a previous call. In contrast, the approach taken by RMS provides locking on a record basis. RMS uses the VMS distributed lock manager to implement this functionality. Unfortunately, the lock manager is not well suited to implementing byte range locks because the byte range is represented in a form that allows the lock manager to arbitrate access. Therefore, the file server implements its own lock database and arbitrates access to shared files. Internally, the server process maintains a list of locks for each file the server has open and arbitrates access based on these lock structures. Files opened by the file server cannot be shared with other VMS processes because the file server has an exclusive mode lock on each file it has open through the VMS lock manager. The exclusive mode lock guarantees protection from other VMS processes.

Open Mode Mapping

The DOS file system defines open access modes to allow applications to synchronize shared access to a file. The open modes are deny_none, deny_read, deny_write, deny_read_write, and compatibility. Each provides a different level of file sharing capability. Although these modes do not map directly to the VMS file system, no mapping is needed to handle the differences.

The PATHWORKS server opens a file that is being accessed by a client with exclusive access on the VMS system. It assumes the responsibility to arbitrate shared access among multiple clients. The server supports DOS open access modes by implementing the shared access resolution algorithm described in the SMB protocol specification.

5 Pathworks Transport Layer Interface

The PATHWORKS for VMS product supports multiple transports through a common transport layer interface. These include the local area system transport (LAST), the transmission control protocol/internet protocol (TCP/IP), and the DECnet transport protocol over Ethernet and token ring networks. This well-defined, uniform mechanism dynamically adds support for network transports and protocols. By conforming to this specification, transports can be added to a server platform without upgrading or changing the existing file server.

The performance goals of the file server had an impact on the development of the transport layer interface. The file server utilizes an optimized transport layer interface that reduces buffer copies and eliminates some of the standard VMS I/O paths. This optimized interface is used with the LAST transport and is described in detail in "The Development of an Optimized PATHWORKS Transport Interface" paper in this issue.[2]

6 Performance Considerations

Achieving an acceptable level of performance from a nondedicated file server layered on a general-purpose operating system proved to be a challenging task. One of the performance goals for the file server was that it perform tasks within 10 to 20 percent of the speed of a dedicated PC file server running on a similarly sized CPU performing the same tasks. This goal was achieved by employing a variety of caches, algorithms, and heuristics. Many of these heuristics were based on the analysis of the SMB messages passed between the server and the client for typical PC applications. As discussed in this section, the response time of the server is improved if the memory contains the information necessary to satisfy a request when it arrives.

Data Caching

An obvious approach to implementing the read and write functions in the file server is to issue these operations to the FILES-11 file system, wait for their completion, and then send a response to the client. This method is simple and persistent, but does not perform well due to the bottleneck formed at the FILES-11 interface and disk. The file server implements a software write-behind data cache to reduce this bottleneck and to eliminate waiting for disk writes to complete before returning a response to the client. Caching is a technique used to decrease access time to information by using a faster intermediate medium to store the most commonly accessed pieces of information. The caching algorithm implemented by the server is a logical block cache. The cache is a region of memory that is segmented into fixed-sized buffers. Each file opened by the server has a dynamic set of buffers that increase and decrease based on a least recently used (LRU) algorithm.

Effects on Client Read Requests. Although it is an optimal environment for servicing read requests, reserving data in memory to satisfy all read requests is not practical. A number of mechanisms were implemented to approach the ideal. The data cache retains recently accessed data in memory with the expectation that it will be referenced again soon. This is based on the concept of locality of reference, both spatial and temporal. Once the server receives a read request, it determines if the buffers associated with the read request are in the cache by using a hashing algorithm for the lookup function. If the data to satisfy the read request is in memory, it is immediately returned to the client, and the file system access is eliminated. If some of the data needed to satisfy the request is not in the cache, then reads are started on each of the cache buffers needed to satisfy the request. Once all of the data is read into cache memory, a response is formed and returned to the client.

Effects on Client Write Requests. When a client write request is received by the server, three processes are performed. The cache buffers needed for the specified write range are located, the client data is copied to the cache buffers, and a response is sent to the client. The data copied to the cache is written to the disk at a later time. This write-behind scheme allows write requests to be serviced quickly because the response is returned to the client before the write to disk completes. By not synchronizing on-disk write completions before returning a response, the turnaround time of client write requests is greatly reduced. The cache is also optimized when a client write request is received and a disk read operation is in progress for the range. In this case, the data being written to the cache is copied into an intermediate buffer and merged with the data from disk after the read operation completes. These intermediate buffers are known as ghost buffers, since they are not visible from the buffer hash table.

Writing Data to Disk. Since the file server acknowledges write requests before performing the write operation, a mechanism is needed to write the cache buffers to disk and ensure data integrity. The file server implements a permanent thread, the flush thread, dedicated to this task. The flush thread starts disk write operations on buffers that contain modified data. Flushing data to disk occurs (1) periodically, based on a user-configurable interval; (2) when a file is closed; (3) when the ratio of dirty to free cache buffers reaches a user-configurable threshold; and (4) when cache buffers are not available to support the current request.

On the VMS system, RMS also employs a write-behind algorithm similar to the one used by the file server. RMS is not used by the file server for disk reads and disk writes for performance reasons. The crossing of the VMS architectural boundary that occurs during RMS calls adds an unacceptable amount of processing time to the read and write paths. The file server uses the VMS queued I/O (QIO)/extended QIO processor (XQP) interface, which is below the RMS layer, to read and write data to disk.

Disk Space Allocation

Sufficient disk space must be available for any write operation that is performed as a background operation. To allow sufficient space, any disk allocation must be completed when the write request is received. This restriction slows down write operations which, in turn, results in file expansion. Performance testing in this area shows that such expansion operations can reduce the server's response time in the overall operating environment. To alleviate this problem, the PATHWORKS server preallocates a fixed amount of disk space, often much greater than required, to complete the current write request, in anticipation of further file expansion. This mechanism greatly reduces the system overhead incurred in disk allocation; thus it improves the overall response time to write operations.

Read Ahead

Another mechanism used by the file server to improve the turnaround time of read requests is read ahead. As with data caching, the goal is to increase the probability that data referenced in the near future will be in the cache. Read ahead is the process of prefetching previously unreferenced data from the disk into the cache. Data is prefetched into cache memory under several conditions. When a file is opened, the first two cache buffers of the data are read from the disk into the cache. Data is also prefetched when the server detects that the file is being accessed sequentially. The SMB protocol also supports read ahead. The protocol provides a field in the read request that specifies the amount of data that the client intends to read in the future. This advisory field is used by the server to initiate prefetches.

Directory Search-ahead Cache

A DOS directory operation can translate to multiple exchanges of request and response operations between the server and client. This behavior is inherent to the SMB protocol definition. The file server initiates a search-ahead thread when the first request is received. While the PC is processing the first response, the search-ahead thread accumulates directory information in a circular buffer. Thus, this information is available in memory for subsequent requests.

Open-file Cache

Operations, such as create, open, and close, impact performance in the VMS system. Benchmark tests show that these operations become blocking factors for a fast performance server. This problem is compounded by the inherent behavior of many PC applications because they often use the result of an open operation as a deterministic tool on file accessibility. Frequently, files are opened and closed and reopened in consecutive requests. To minimize the overhead incurred for these operations, the PATHWORKS server implements a cache to store opened file information. This open-file cache maintains the file header information after the file has been closed by the user for a short duration. If a user requests to open a file that is already cached, no request to VMS FILES-11 system is required. This greatly reduces the response time of the server on the second open request.

Furthermore, many DOS database applications use index files to synchronize data access. These files are frequently accessed by many DOS users when working in an networked office environment. Open-file caching is beneficial to this environment because it incurs a minimal amount of open requests to the VMS file system.

Byte Range Locking Back-off Algorithm

The file server implements an algorithm to improve overall performance of the server and network when PC applications are sharing files and using byte range locking to arbitrate access. The analysis of many networked PC database applications revealed that a client typically entered a tight retry loop when it detected a lock conflict. This spinning produces an excessive amount of lock-related network traffic, especially for very fast clients. The server also has to spend a significant amount of time processing these numerous lock requests. The server attempts to regulate this lock traffic and reduce its lock processing time by deferring the return of the response when a lock conflict is detected. If a request to lock a range conflicts with a previous lock, the server makes repeated attempts to access the range using a pseudorandom exponential back-off algorithm to determine the retry interval. If the lock conflict is not resolved after a user-configurable time period, the server returns a response indicating a lock conflict. By deferring this response to the client, the server exercises flow control over clients spinning on locked regions of the file. The implementation of the pseudorandom exponential back-off algorithm prevents the server from using an excessive amount of CPU time to determine if the locked byte range has been unlocked.

7 Security

The VMS operating system offers a well-defined security architecture, but DOS has no comparable security scheme. Since the PATHWORKS file server is implemented as a privileged process, it is necessary to control file access on the VMS host system from a DOS client. There is no one-to-one correspondence between a DOS user and a VMS user. That is, in the PATHWORKS environment, each network client, much like a terminal in this respect, can be multiple VMS users. The problem is to ensure maximum shareability among PC clients and maintain the desired level of VMS security.

The PATHWORKS file server implements two types of securities: share and user. It makes use of the PCFS$SERVICE_DATABASE to control access to a share area; and the VMS user authorization file (UAF) database to control access to directories and files based on a VMS user account. A share, referred to as file service, is a VMS directory that can be accessed by

PATHWORKS clients. PATHWORKS software defines three types of file services: system/application, common, and personal. Access to file services is based on VMS user account information. A privileged system manager must explicitly grant user access to system/application and common services.

The system manager must also specify the types of access: read, write, or create. This information is stored in the PCFS$SERVICE_DATABASE. Access to personal service is implicit with the existence of a user account.

To provide maximum shareability among PC clients, PATHWORKS software includes a default user account. When accessing a file service that has been granted to the default account, each PC assumes the identity of the default account. Thus the access, though it might be issued by different PC users, is viewed as the same user. This mechanism provides a "share level" of security.

A more restrictive environment is achieved by providing access to a share area based on individual user account. When a PC client establishes access to a service, it presents a user account and its corresponding password. This information is authenticated based on information returned by the sys$getuai system service call. The PATHWORKS server then verifies that this user has been granted access to the service.

Access to a file service does not necessarily imply access to any individual files. In order to preserve the desired level of VMS security, PATHWORKS honors access control entries. The server ensures access to a share area as defined in the database by mapping the access types to two identifiers: pcfs$read and pcfs$update. These identifiers are added to the root directory of a share area, and to any files that are created, when appropriate. As the server impersonates the user, the appropriate identifier is associated when access privilege to files and directory is checked. This security implementation is not applicable when servicing a personal area. Access to files stored in a personal area is based on RMS protections mask.

To ease system management tasks, PATHWORKS software implements "group" support. A group is a collection of users. A PATHWORKS group has no dependency on user group identification code. When a share is granted to a group, each member of the group gains access. Note that authentication is still performed based on an individual user account.

Since a DOS client can gain access to the VMS environment, it is imperative that the file server support the VMS system's break-in evasion mechanism. The server honors the login-related system parameters. These parameters are read at the file server start-up, and the values are in effect for the duration of the server process. The server tallies any failed or unsuccessful login attempts. When the file server receives a connection (login) request to service, the file server extracts the related counter information from the UAF and adds it to its internal counter to determine whether evasive action is to take place. When a break-in is detected, the server takes the appropriate evasive action and signals the condition in the server log file.

8 Printing Support

The server process also implements the printing functionality specified in the SMB protocol. The file server implements the print-related commands by using $SNDJBC and $GETQUI system services to communicate with the VMS job controller. Each print service available to clients has a VMS print queue associated with it.

The VMS system has a much richer printing environment than the one provided to the PC clients through the SMB protocol. The PATHWORKS server provides VMS printing features to the clients by extending the SMB protocol to accommodate PATHWORKS needs. These protocol extensions are described in the section Digital Protocol Extensions.

9 File Sharing Among Server Processes

Each node on a VAXcluster system can be a host for the PATHWORKS server process. One of the more challenging problems in supporting VAXcluster systems is the synchronization of file access by multiple server processes. As stated earlier, the PATHWORKS file server requires exclusive access to files that are opened by PCs in order to support byte range locking in DOS. Furthermore, in a cluster, each server process needs the ability to provide identical access to the same resources.

PATHWORKS software implements its own lock management algorithm to resolve file access conflicts in a VAXcluster system. Although multiple server processes are allowed in the environment, only one process can handle the requests to a file that is accessed by PC clients. By using the VMS lock manager, the server process that services the first open request acquires an exclusive mode lock on the file. It thus becomes the master of the file and is responsible for synchronizing access requests to the file. When a server process is requested to service a file that has another PATHWORKS server as its master, it makes a network connection to the master process and forwards the requests. This process serves as the routing agent. It communicates both requests and responses between the master server process and the PC client. The master releases ownership when no outstanding open file handles are on the file. File mastering is established on a per file basis.

The rerouting mechanism uses the DECnet transport because its existence on the remote server host is guaranteed in a cluster environment. To minimize the number of required DECnet sessions, the routing agent funnels all forwarding SMBs through an existing session. The forwarding packets include information that the master process can use to differentiate among the clients' access requests.

10 Pathworks Server Configuration

The multithreaded PATHWORKS file server can be considered a small operating system in which each PC is a process (or a thread). In addition to the basic resource requirement that the server be activated, the server requires a set of process resources to support each client thread. These resources can be mapped to VMS process parameters which, in turn, translate into system parameters.

The amount of VMS system resources which the file server consumes is a function of the number of clients and the workload generated by the individual PC. Mapping the PC resource requirement to the appropriate VMS process and system parameters proves to be a complex problem. Since the PC workload profile is unknown at the time of server initialization, the amount of required system resources for the server process can only be estimated.

PATHWORKS system managers include users with little VMS system management experience. The level of VMS system expertise required to configure (or set up) a PATHWORKS server is minimized by the addition of a "configurator." This part of the management functionality is implemented to generate information on required system and process resources when the desired configuration is supplied. During the server start-up phase, the configurator checks for availability of necessary resources and provides appropriate run-time parameters for the launching of the server process.

11 Management Interface

To provide integration between different file systems, the file server utilizes PATHWORKS specific databases (such as the service database), standard VMS databases (such as the UAF and DECnet databases), and VMS security mechanisms. These entities must work in harmony and be consistent with each other to provide the desired integration. The PCSA_MANAGER utility was designed to manage this environment. It allows users to perform all management tasks related to PATHWORKS software through one utility from a menu-driven user interface or a command line interface. The PCSA_MANAGER utility allows system administrators to manage the following objects: users, services, print queues, logical user groups, the event logger, and the server process. The file server uses interfaces supported by VMS to manipulate VMS specific databases, private interfaces to access PATHWORKS specific databases, and SMB protocol extensions to interact with a server process.

Digital Protocol Extensions

Management of a running server requires a method to send and receive well-defined messages between the server and other processes. The PCSA_

MANAGER utility sends a management request to the server; the server processes it, and sends an appropriate response back to the PCSA_MANAGER. The communication channel used for server management is a DECnet logical link. The PCSA_MANAGER issues a connection request to the DECnet object associated with the file server process. The file server receives this request and creates a virtual circuit with a corresponding thread to process requests for this management session. This is similar to a client session.

Since the SMB protocol does not provide commands sufficient to manage a PATHWORKS server, a Digital proprietary protocol was developed to provide this functionality. This protocol is merely an extension of the SMB core protocol; that is, the messages developed for server management have valid SMB headers with command codes that are meaningful only to a PATHWORKS server. This implementation allows remote management of the file server. To manage a server, a management utility only has to establish a virtual circuit and exchange these extended SMBs. Protocol extensions are also used to integrate the VMS print system with PATHWORKS clients, along with other PATHWORKS specific utilities.

Event Logging

The PATHWORKS server includes an event logging mechanism to provide an error and event reporting facility to assist system management. Events are categorized based on server operations, including errors, protocols, security, management, and file-related functions (open/close, read/write). The server uses an event code to determine whether a given event is to be recorded. A Digital extended SMB command toggles these event codes dynamically. The event messages are logged to the file server log file. The overhead is minimized by caching the event messages in a data buffer, which is periodically written out to the log file. A thread is created at server start-up to handle the log file update function. The scheduling of this thread is based on a time interval, with a default value of 60 seconds.

12 Summary

The PATHWORKS for VMS file server integrates the DOS, OS/2, and VMS operating system environments on a network. The server architecture achieves transparent integration of PCs connected on an open network over multiple transports. Data caching, algorithms, and heuristics were used to increase performance. The PATHWORKS for VMS file server provides PC users with access to the VMS system's resources and security environment.

13 Acknowledgments

We thank the people, past and present, who contributed to the design and development of the PATHWORKS for VMS file server. We specifically acknowledge Robert Praetorius for his contribution in the design and implementation of the cache component, Phil Wells for his design and implementation of the network interface and transport support, and Jon

Campbell for his design and implementation of the network interface. We also acknowledge Frank Caccavale for his work on performance analysis, Alan

Abrahams for his direction as architect, and Mark Olson for his leadership of the PATHWORKS for VMS project.

14 References

1. X/Open Developer's Specification-Protocols for X/Open PC Interworking: SMB (Reading, U.K.: X/Open Company Limited, Document No. XO/DEV/91/010, 1991).

2. P. J. Wells, "The Development of an Optimized PATHWORKS Transport Interface," Digital Technical Journal, vol. 4, no. 1 (Winter 1992, this issue):xx-xx.

15 Author Biographies

Edward W. Bresnahan Senior software engineer Edward Bresnahan has been developing the PATHWORKS for VMS software since joining Digital's PCSG Server Engineering Group in 1988. He is currently responsible for the design and development of a high-performance data cache to be used in future PATHWORKS server products. Prior to this, he was a co-op student at General Electric Company and at Charles Stark Draper Laboratory. Ed holds a B.S.C.S. (1988, honors) from Northeastern University and is pursuing an M.S.C.S. part-time.

Siu Yin Cheng Since joining Digital in 1987, Siu Yin Cheng has worked on server software in the Personal Computing Systems Group. As a senior software engineer, she is responsible for the design and development of the server configuration utility for future PATHWORKS products. Siu Yin designed and developed the server collector process to extract performance data from the file server; she also worked on server development. Prior to this, she led the system testing of PATHWORKS server V2.0-2.2. Siu Yin received a B.S.C.S. (1987, honors) from Brown University.

16 Trademarks