

AppleTalk. for VMS

Architecture and
Implementation

 **APPLE COMPUTER, INC.**

This manual and the software described in it are copyrighted, with all rights reserved. Under the copyright laws, this manual may not be copied, in whole or part, without written consent of Apple Computer. Under the law, copying includes translating into another language or format.

© Apple Computer, Inc., 1987
20525 Mariani Avenue
Cupertino, California 95014
(408) 996-1010

Apple, the Apple logo, LaserWriter, and AppleTalk are registered trademarks of Apple Computer, Inc.

Macintosh is a trademark of Apple Computer, Inc.

Digital, DECnet, VAX, VMS, BLISS, and DIBOL are trademarks of Digital Equipment Corporation.

Ethernet is a trademark of Xerox Corporation.

InterBridge is a trademark of Hayes Microcomputer Products, Inc.



Contents



Figures v

Preface About This Manual vii

Intended audience vii

Structure of this document viii

Associated documents viii

Conventions used in this document ix

Chapter 1 AppleTalk on VAX/VMS 1

AppleTalk overview 2

AppleTalk concepts 3

Nodes and node ID 3

Sockets 3

Internets 3

AppleTalk bridges 3

Local bridge 4

Half bridge 4

Backbone bridge 4

Routing tables 4

Zones 4

AppleTalk protocols 5

AppleTalk Link Access Protocol (ALAP) 5

Datagram Delivery Protocol (DDP) 5

Routing Table Maintenance Protocol (RTMP) 5

Name Binding Protocol (NBP) 5

AppleTalk Transaction Protocol (ATP) 6

Zone Information Protocol (ZIP) 6

Printer Access Protocol (PAP) 6

Echo Protocol (EP) 6

AppleTalk for VMS 6

Chapter 2	The Virtual AppleTalk Driver (VN:) 9
	Virtual Link Access Protocol (VLAP) 10
	Dynamic node establishment 10
	Addressing 11
	Performing VLAP communication 11
	Loading the Driver 12
Chapter 3	The AppleTalk for VMS Protocol Support Library 13
	Protocol Library routines 14
	Organization of the Protocol Library 14
Chapter 4	The AppleTalk for VMS Bridge Process 17
	Conceptual description 18
	Routing tables 18
	Internet routing 19
	Broadcast request handler 19
	Bridge ports 20
	Index 21

Figures

Figure 1-1	Layered protocols of AppleTalk	2
Figure 1-2	AppleTalk bridge configurations	4
Figure 1-3	Implementation of AppleTalk on VAX/VMS	7
Figure 3-1	Organization of Protocol Library	15



Preface

About This Manual

This manual provides a conceptual description of version 1.5 of the AppleTalk® for VMS system. It explains the role of the various components of the system and how each communicates with the others to implement the AppleTalk network architecture on VAX/VMS.

The AppleTalk for VMS services can be used only in programs written in languages that support the VAX/VMS system services. VAX/VMS system services can be used only in programs written in languages that produce *native* code for the VAX hardware. At present these languages include VAX MACRO and the following high-level languages:

- | | |
|---------------------------------------|---------------------------------------|
| <input type="checkbox"/> VAX BASIC | <input type="checkbox"/> VAX BLISS-32 |
| <input type="checkbox"/> VAX C | <input type="checkbox"/> VAX COBOL |
| <input type="checkbox"/> VAX COBOL-74 | <input type="checkbox"/> VAX CORAL |
| <input type="checkbox"/> VAX DIBOL | <input type="checkbox"/> VAX FORTRAN |
| <input type="checkbox"/> VAX PASCAL | <input type="checkbox"/> VAX PL/I |

Language systems not supplied by Digital may be able to use the AppleTalk for VMS services if they adhere to the VAX/VMS system services calling conventions and VAX/VMS Procedure Calling and Condition Handling Standard.

Intended audience

This manual is intended for those who want to learn about the implementation of the AppleTalk network architecture on VAX/VMS. It is primarily intended for system and application programmers who want to use the AppleTalk for VMS services. The guide assumes the reader has fluency in VAX/VMS and data communication principles. For those interested in using the AppleTalk for VMS services, VAX/VMS system or application programming experience is assumed. This is *not* a tutorial.

Structure of this document

This manual is organized into four chapters, as follows:

- Chapter 1 provides background information about AppleTalk and the AppleTalk for VMS system. The first section of Chapter 1 introduces fundamental AppleTalk concepts which the reader will encounter throughout AppleTalk for VMS documentation. These concepts are central to the architectures of both AppleTalk and AppleTalk for VMS.
- Chapter 2 provides a description of the AppleTalk for VMS system component known as the Virtual AppleTalk Driver (VN:).
- Chapter 3 provides a brief overview of the AppleTalk Protocol Support Library, which is treated in detail in a separate document.
- Chapter 4 provides a description of the AppleTalk for VMS system component known as the Bridge Process.

Associated documents

The following manuals from Digital Equipment Corporation document the VAX/VMS system services and I/O architecture, and describe the documentation standards used throughout the VMS manuals.

- *Introduction to VAX/VMS System Routines*
- *VAX/VMS Run-Time Library Routines Reference Manual*
- *Guide to Writing a Device Driver for VAX/VMS*
- *I/O User's Guide*
- *VAX/VMS Utilities Reference Manual*
- *VAX/VMS System Services Reference Manual*

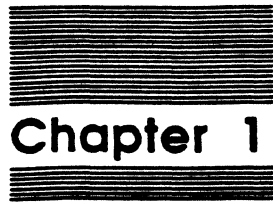
The following manuals from Apple Computer, Inc. document the AppleTalk network architecture, the interface provided to client processes of AppleTalk, the architecture of the AppleTalk for VMS system, and the interface and services it provides to client processes.

- Sidhu, Gursharan S., Richard F. Andrews and Alan B. Oppenheimer, *Inside AppleTalk* (Apple Programmer's and Developer's Association draft)
- *Inside Macintosh*, Volume II, Chapter 10, "The AppleTalk Manager"
- *Inside Macintosh*, Volume IV, Chapter 23, "The AppleTalk Manager"
- *Inside Macintosh*, Volume V, Chapter 28, "The AppleTalk Manager"
- *AppleTalk for VMS Protocol Support Library Reference Manual*
- *AppleTalk for VMS Installation and Operation Guide*
- *AppleTalk for VMS Bridge Control Program Guide*

Conventions used in this document

Convention	Meaning
file-spec,...	Horizontal ellipsis indicates that additional parameters, values, or information can be entered or supplied
[logical-name]	Square brackets indicate that the enclosed item is optional
quotation mark	Used to refer to the "double quote" (")
apostrophe	Used to refer to the "single quote" (')

Other conventions used in this document are described in the Digital VAX/VMS manuals referred to in the previous section, particularly the *VAX/VMS Run-Time Library Routines Reference Manual*.



Chapter 1

AppleTalk on VAX/VMS

AppleTalk overview

The AppleTalk network architecture conforms to the Open Systems Interconnection (OSI) model of the International Standards Organization (ISO) by defining a series of layered protocols that manage network operations. The AppleTalk architecture supports the interconnection of multiple networks via direct internetwork connection or long-distance communication links.

The layered AppleTalk protocols correspond to the ISO/OSI model as illustrated in Figure 1-1. Items shaded in gray represent AppleTalk protocols not currently implemented in AppleTalk for VMS.

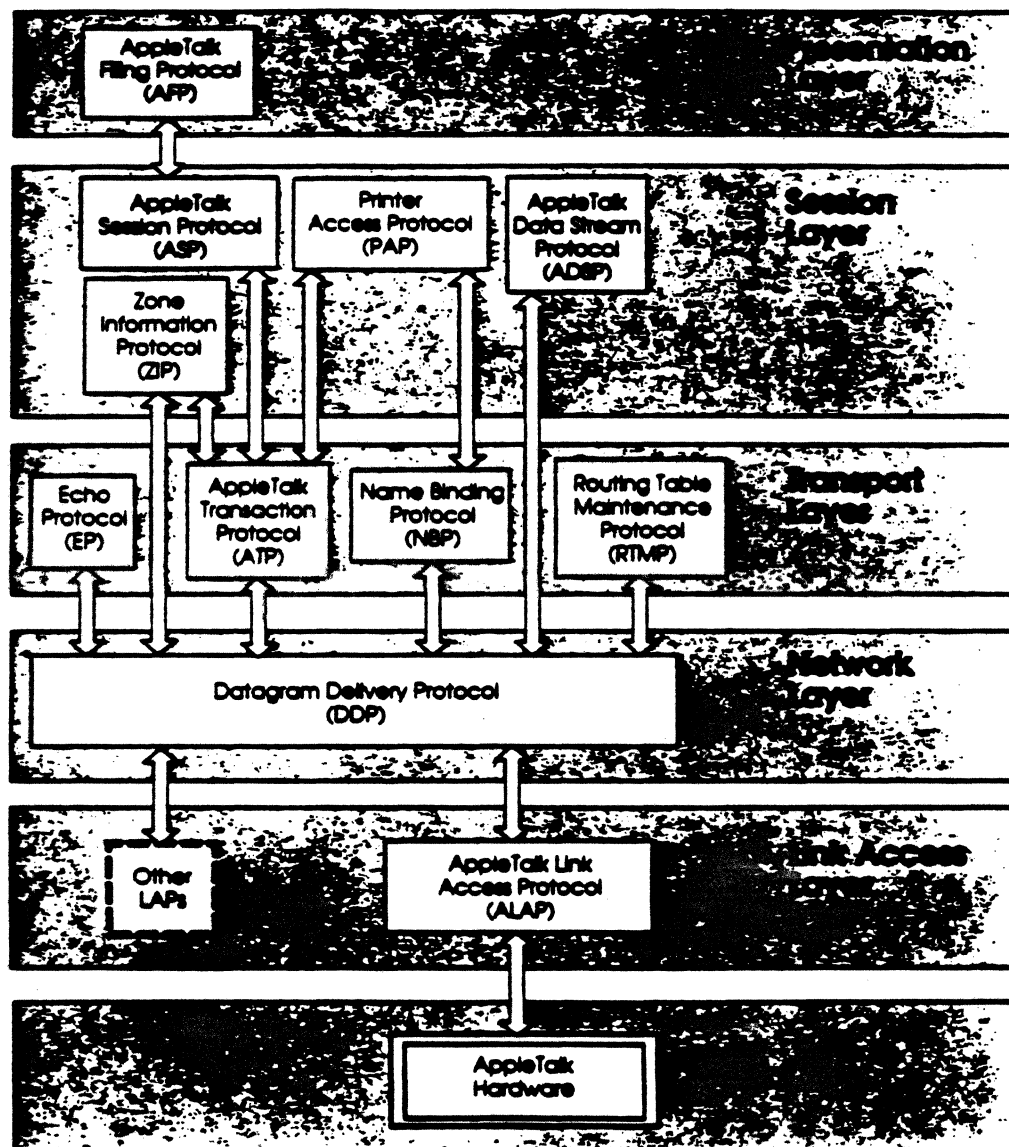


Figure 1-1
Layered protocols of AppleTalk

This chapter introduces principal AppleTalk concepts and briefly defines the function of each AppleTalk protocol. It also provides an overview of AppleTalk network implementation in the VAX/VMS environment.

AppleTalk concepts

An AppleTalk network uses standardized communication protocols which operate on a range of explicitly defined physical and logical entities. These entities are described below, and will be referred to frequently throughout AppleTalk for VMS documentation. AppleTalk protocols are defined in the following section.

Nodes and node ID

A **node** is a single addressable entity connected to an AppleTalk network. A node on a physical AppleTalk network may be a computer, such as a Macintosh™ II, or a peripheral device, such as a LaserWriter®.

Each node is identified by a network device ID number, also called the **node ID**. ID assignment is performed dynamically at the time of connection to the network, and network protocols verify the uniqueness of the assigned address. In this way nodes may be moved between networks without the risk of producing ID conflicts.

Sockets

Sockets are logical entities within nodes. Sockets enable network addressing to be extended beyond node ID to specify individual processes, or **socket clients**. Socket numbers are classified as *statically* or *dynamically* assigned. Static sockets support AppleTalk core protocols and other protocol development. Dynamic socket assignment takes place during network operation, as AppleTalk protocols provide unique identifiers by which to address socket clients.

A socket's node ID together with its socket number constitute the unique AppleTalk address of that socket.

Internets

AppleTalk supports the interconnection of multiple AppleTalk networks into an **internet**. A **network number** is assigned to each connected network for addressing purposes. The complete **internet address** of a socket consists of its node ID, socket number, and the ID number of the network to which the node is connected.

AppleTalk bridges

The linking of an AppleTalk network to one or more additional networks is accomplished with the use of a **bridge**. A bridge device has multiple ports and is capable of executing logical internet routing operations. Bridge-resident routing protocols are subsets of several of the AppleTalk layered protocols, and perform data link functions on bridge ports, message routing management, address maintenance, and other functions detailed in AppleTalk protocol documentation. (See *Inside AppleTalk*.)

The following bridge configurations are supported:

Local bridge: A local bridge is used for the direct interconnection of two or more networks in close proximity.

Half bridge: Where remote networks are interconnected via long-distance links (such as modems), each network is connected to an AppleTalk bridge, which in turn is connected to the datacom link. Thus the bridge at each end is said to serve as a "half bridge."

Backbone bridge: A backbone bridge may have one or more ports connected as a local or half bridge, but it will also be connected to a backbone network of bridges, serving in effect to link multiple internets.

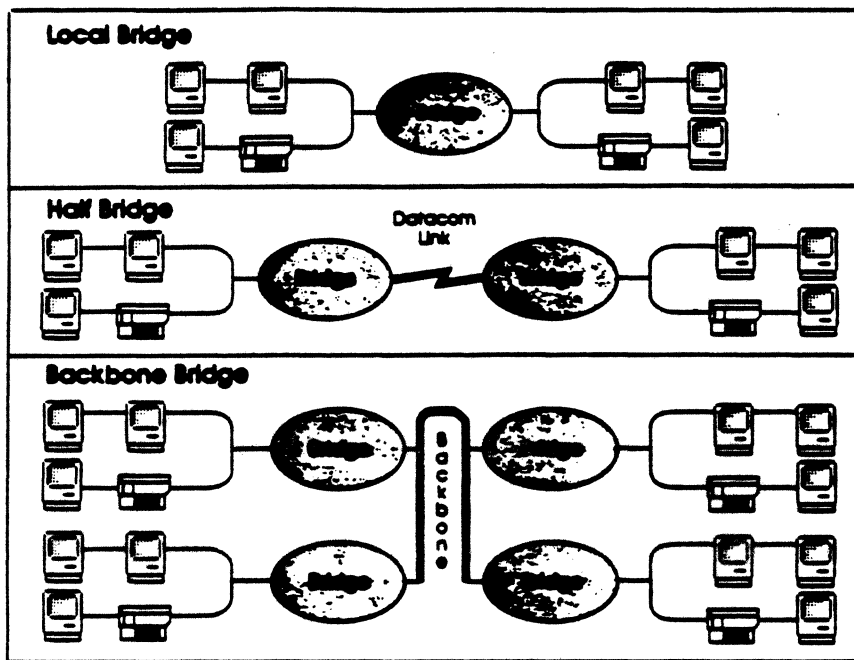


Figure 1-2
AppleTalk bridge configurations

Routing tables

A routing table is resident in each AppleTalk bridge (which is why the bridge is sometimes referred to as an internet router). The routing table serves as a map of the internet, specifying the path and distance between the current bridge and bridges of other networks, as well as the active status of each bridge.

Zones

A zone is an arbitrarily defined grouping of Appletalk networks, which may be used in addressing a subset of the total internet. Zones facilitate the departmental grouping of users.

AppleTalk protocols

As illustrated in Figure 1-1, the AppleTalk protocols implemented in AppleTalk for VMS correspond to the following layers of the ISO/OSI model:

- ☐ data link control
- ☐ network control
- ☐ transport control
- ☐ session control

Higher-level AppleTalk protocols exist for functions and applications beyond these layers, but they are not at this time implemented in AppleTalk for VMS.

The AppleTalk protocols are defined below.

AppleTalk Link Access Protocol (ALAP)

ALAP manages AppleTalk data link control functions for devices using the physical AppleTalk Personal Network cabling scheme. The primary functions of ALAP are to control network bus access, to manage data encapsulation into frames and control frame transmission/reception, and to provide device identification and network addressing mechanisms. Corresponding link access protocols can be used in place of ALAP in other AppleTalk network environments, such as Ethernet. Such link access protocols will provide functions similar to those of ALAP for their respective environments.

Datagram Delivery Protocol (DDP)

DDP defines internet addresses through socket and network ID assignment, and delivers data between individual sockets across an AppleTalk internet. Data packets routed by DDP are called **datagrams**. DDP determines the destination address of datagrams and builds the appropriate address headers onto data packets, to enable routing through internet bridges as required.

Routing Table Maintenance Protocol (RTMP)

RTMP provides the logic required by bridges to route datagrams through bridge ports toward destination networks. The key element used to accomplish this is the routing table resident in each bridge. RTMP dynamically maintains routing tables to reflect changes in internet topography and periodically updates all table entries.

Name Binding Protocol (NBP)

NBP enables AppleTalk protocols to understand user-defined zones and device names, by providing and maintaining translation tables that map these names to corresponding socket addresses.

AppleTalk Transaction Protocol (ATP)

An AppleTalk transaction takes place when a client of one socket transmits to another socket a request for which some response is expected. (Generally a status report or acknowledgment of the requested function is returned.) ATP manages this transaction in a way that binds the request and response together to ensure the reliable exchange of request/response pairs.

Zone Information Protocol (ZIP)

ZIP supports the AppleTalk zone feature through network number-to-zone name mapping. Zone Information Tables are implemented in all internet bridges and maintained by ZIP services.

Printer Access Protocol (PAP)

PAP is a session-level protocol intended for use between workstations and print servers. PAP is a connection-oriented protocol that handles connection setup, maintenance, and teardown in addition to data transfer.

Echo Protocol (EP)

EP is implemented as an echoer process at a particular AppleTalk address. It can be used by a DDP client to determine if the designated address is accessible over the internet. The echoer process is also useful to developers in determining the time required for a packet to reach its destination.

AppleTalk for VMS

The implementation of AppleTalk in the VAX/VMS environment is achieved using the protocols of an AppleTalk internet. Within this internet, the VAX computer represents a virtual AppleTalk network, connected via an internetwork bridge.

The virtual network within the VAX is composed of VMS processes, each of which represents a network node. Three principal architectural components support this design:

- The virtual AppleTalk network resident in VMS is managed by the **Virtual AppleTalk Driver**, to which VMS processes can connect to gain access to network services.
- The **Protocol Support Library** provides VMS processes with an interface to AppleTalk network functions.
- The **AppleTalk for VMS Bridge Process** is the logical equivalent of an AppleTalk internet bridge device. The Bridge Process manages internetwork routing of datagrams.

To any AppleTalk network connected to the VMS host, the VMS Bridge Process appears as an AppleTalk bridge. Each host process connected to the virtual network—including the Bridge Process—is treated as an AppleTalk node (see Figure 1-3).

Nonbridge processes are normally connected only to the virtual network. Bridge processes may be connected to both external physical networks and to the internal virtual network. There can be multiple bridge processes and virtual networks, if desired (possibly useful in a cluster environment).

The key conceptual elements of AppleTalk for VMS are:

- a VMS process is mapped as an AppleTalk node
- the bridge is such a process/node
- AppleTalk network services are available to VMS processes as individually callable Protocol Library routines
- the virtual network is indistinguishable from a real network as far as the attached processes are concerned

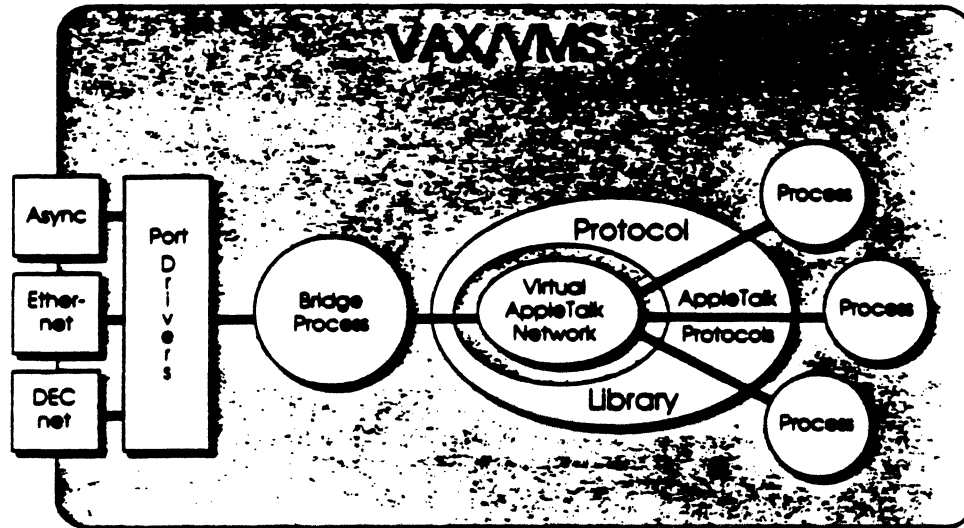


Figure 1-3
Implementation of AppleTalk on VAX/VMS



Chapter 2



The Virtual AppleTalk Driver (VN:)

The Virtual AppleTalk Driver is a software-only driver. It has no associated physical device. Such drivers are known as *pseudodevice* or *virtual device* drivers. A virtual device driver acts as the I/O transfer agent for communication between processes. The Virtual AppleTalk Driver provides a controlled and synchronized method for processes to exchange data.

Virtual Link Access Protocol (VLAP)

In the AppleTalk for VMS system, processes use the Virtual AppleTalk Driver to exchange AppleTalk packets. The Virtual AppleTalk Driver employs a **Virtual Link Access Protocol**. The VLAP essentially provides the same basic service as the AppleTalk Link Access Protocol (ALAP), which is to provide "best effort" delivery of packets between nodes. The ALAP manages the encapsulation and decapsulation of data in an ALAP frame and the transmission and reception of packets on the bus. Likewise, the VLAP manages the encapsulation and decapsulation of data in a VLAP frame and the transmission and reception of packets *between nodes on a virtual bus*.

Like ALAP, VLAP is also responsible for arbitrating access to the virtual bus and provides equal access for all nodes. Whereas the ALAP uses an access method termed Carrier Sense, Multiple Access with Collision Avoidance (CSMA/CA), the VLAP uses the VMS I/O architecture and operating system process state/priority mechanism to perform network access arbitration.

Dynamic node establishment

The VLAP provides the additional service of dynamic node establishment. Through the VLAP, processes may request that they be connected and established as a node on the virtual bus. When a process is connected to the virtual bus, the VLAP creates and establishes a link between the process and the virtual AppleTalk network as well as assigning a node ID. By creating a new link, each process (node) is provided with its own access path to the virtual bus; this link is analogous to the hardware connection in an AppleTalk network running ALAP. As with ALAP, the VLAP uses a dynamic node assignment scheme.

Every VAX/VMS driver defines a unit control block (UCB) for each device unit it supports. Normally, one simply defines and statically creates a separate UCB for each unit. The Virtual AppleTalk Driver maps each virtual link as a separate unit (with its UCB). It is not practical to define the UCBs statically because we are dealing with a dynamically changing number of virtual links. VMS "device unit cloning" makes it possible to create and delete units on demand.

To make use of this feature, the Virtual AppleTalk Driver defines a template UCB. When a client process issues a `SYSS$ASSIGN` system service to a driver with a template UCB, VMS clones the UCB. When a UCB is cloned, it appears as a new device unit to VMS. A device unit corresponds to a node in the AppleTalk for VMS implementation. The Virtual AppleTalk Driver treats each UCB as a node with a node ID equal to the unit number assigned to the UCB by VMS. The UCB or node is removed from the system when all channels have deassigned from it.

Addressing

As mentioned in the previous section, each node on a virtual AppleTalk network has a node address or ID assigned by the VLAP. The node ID is actually the unit number of the UCB that defines the node on the virtual network. It uniquely identifies a node on the local network and is used to determine the source and destination of VLAP packets.

The VLAP delivers packets directly to a node on the local virtual network by matching the destination node address to a unit number. As with ALAP, the address 255 (\$FF) has special significance, and node 0 is not allowed and will generate an error. Packets with a destination address of 255 (\$FF) are delivered to all nodes on the virtual network. This permits broadcasting to all nodes on the virtual AppleTalk network.

Performing VLAP communication

A process prepares for VLAP communication by calling the `ATK$INIT_PORT` and `ATK$OPEN_PORT` routines in the AppleTalk Protocol Library. These routines establish communication with the virtual AppleTalk network by issuing a `$ASSIGN` system call, which requests a system-assigned channel number from VAX/VMS. The routine specifies the pseudodevice of the Virtual AppleTalk Driver (VN:) in the assignment. VMS proceeds to clone a copy of the Virtual AppleTalk Driver's template UCB. The UCB is allocated from the operating system's nonpaged pool. If the assignment is successful, the system returns a channel number to the client process for use in subsequent VLAP I/O requests.

Once the `ATK$OPEN_PORT` routine has obtained a channel number (established itself as a node), it uses the VAX/VMS `$GETDVI` system service to determine its node address. The node address is the unit number of the cloned device unit.

The AppleTalk Protocol Library supports AppleTalk communication by using the VMS `QIO` system service with a function code of either `IO$_READVBLK` or `IO$_WRITEVBLK`. If the I/O request is an `IO$_WRITEVBLK`, the driver completes the write request with success whether or not the destination node actually receives the packet.

Broadcast messages and those sent to a node with no receive pending are buffered in system dynamic memory for up to 2 seconds. If a receive is issued within the 2-second message lifetime, the message is copied to the reader's buffer and the message is deleted from pool.

The buffered system dynamic memory is charged to the writer's buffered I/O byte count quota. Each write uses 603 bytes of system dynamic memory. When the write completes, either when the addressee issues a read or the message lifetime expires, the system dynamic memory is freed and the writer's buffered I/O byte count quota is credited.

A process can remove itself as a node from the virtual network by issuing a `$DASSGN` system service on the corresponding channel assigned to the VN: driver. The deassign completely removes the node associated with the channel from the virtual network. It also causes VMS to delete the cloned UCB.

Loading the Driver

The Virtual AppleTalk Driver (VN:) will normally be loaded during system startup. However, it can be loaded any time after system startup. It cannot be reloaded.

To load the Virtual AppleTalk Driver, run the System Generation Utility (SYSGEN) from the system manager's account or from an account having Change-Mode-to-Kernel privilege using the following command:

```
$ RUN SYS$SYSTEM:SYSGEN
```


SYSGEN responds with a prompt and waits for further input:

```
SYSGEN>
```

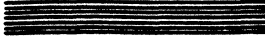
The *VAX/VMS Utilities Reference Manual* describes the full set of SYSGEN commands. To load the VN: driver, use the CONNECT command as follows:

```
SYSGEN> CONNECT VN0/NOADAPTER/DRIVERNAME=VNDRIVER
```

The CONNECT command loads the Virtual AppleTalk Driver, if it is not already loaded, and creates the control blocks in the I/O data base needed to describe the VN: device. The /NOADAPTER qualifier is required for drivers associated with software devices. The /DRIVERNAME qualifier is optional. If this qualifier is omitted, CONNECT concatenates the first two characters of the device code with "DRIVER"; for example, VNDRIVER.



Chapter 3



The AppleTalk for VMS Protocol Support Library

Protocol Library routines

The AppleTalk for VMS Protocol Support Library (or simply the Protocol Library) provides programmers with a coordinated set of procedures for accessing AppleTalk services under VAX/VMS. To the extent practical, the library procedures mirror the functionality of the high-level (Pascal) calls provided by the Macintosh AppleTalk Manager.

The Protocol Library provides a common run-time environment for applications because its procedures follow the VAX Modular Programming Standard. The advantage of a common run-time environment is that any program written in MACRO, BLISS, or another supported high-level language can call any procedure in the Protocol Library.

Protocol Library procedures are written largely in the C language, using the VAX Procedure Calling and Condition Handling Standard as described in the *Introduction to VAX/VMS System Routines*. They may generally be used in conjunction with VAX/VMS system services, which are also callable procedures (see the *VAX/VMS System Services Reference Manual*).

Organization of the Protocol Library

The role of the Protocol Library is to provide the functions needed for a process running under VMS to act as a node on an AppleTalk network, independently of the data link in use. The following data link devices (ports) are supported:

- VLAP: virtual AppleTalk on VMS, DDP-addressable
- ELAP: backbone Ethernet using Digital controllers (DEUNA, DEQNA, DELUA), not DDP-addressable
- AHLAP: asynchronous half bridge (Hayes InterBridge-compatible), using the VMS terminal driver. This protocol is subject to change, and its use in the current implementation does not represent an endorsement by AppleComputer, Inc.
- DLAP: DECnet logical link between bridges

The connection to a particular data link device is handled by a modular object called a **port driver** as defined in the AppleTalk architecture document, *Inside AppleTalk*. The port driver is analogous to a device driver at the data link level. It isolates the layers above it from the specifics of the link device and its addressing. The port drivers for the AppleTalk for VMS Protocol Library are completely isolated in separate shareable images. This makes it possible for developers to write their own port drivers without relinking the main library. It also allows painless adjustments to low-level protocols.

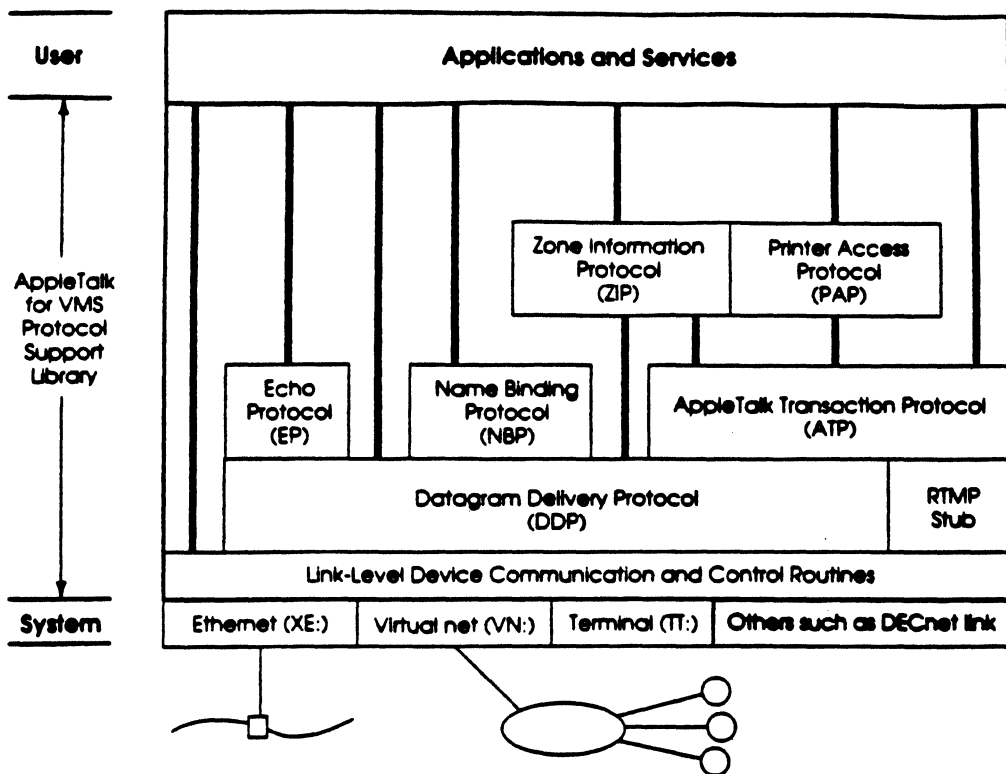


Figure 3-1
Organization of Protocol Library

The AppleTalk for VMS Protocol Support Library is described in greater detail, and each library routine is explicitly defined, in the companion document titled *AppleTalk for VMS Protocol Support Library Reference Manual*.



Chapter 4

The AppleTalk for VMS Bridge Process

This chapter describes the AppleTalk for VMS **Bridge Process**. The Bridge Process provides all the services of an AppleTalk bridge or internetwork router, including an internet routing agent, a routing table maintenance process, a Name Binding Protocol directed broadcast handler, and a zone information manager.

Conceptual description

Conceptually, the bridge is a VMS process that provides the services of a real AppleTalk bridge device. The Bridge Process acts as the datagram routing agent for internet datagrams between networks, including the virtual (VLAP) network. To do this, it has an agent that maintains a routing table according to RTMP. It also handles directed broadcast NBP lookups resulting from specified zone lookup requests. Finally, it participates in the maintenance and use of zone names according to ZIP. In other words, the Bridge Process is a "real" AppleTalk bridge in every sense.

AppleTalk defines three types of bridges that may be used to build an internet (see Chapter 1). They are called local bridges, half bridges, and backbone bridges. The Bridge Process design is general enough to allow support for all types of bridge connections. In fact, a single Bridge Process may have multiple network connections, each using any one of the defined bridge type connections.

This is made possible by the port orientation of AppleTalk Protocol Library I/O routines. Each connection to a network is kept in the context of a port, as defined in the Apple RTMP documentation. Incoming packets are tagged with the ID of the port on which they arrived; no other information is needed. The port drivers are implemented as separate shareable images which are activated by the Bridge Process (via Protocol Library calls) as needed.

The routing decision is based solely on the destination network number in the DDP header, and on the routing table that contains the physical addressing needed to get the packet to the next router or the destination. Physical addressing is kept separate from the program code so that various types of physical layers can be accommodated.

Routing tables

Routing tables are the principal data structures needed for internet routing. They are dynamically changing maps of the internet. In AppleTalk, bridges manage and consult routing tables to allow them to determine how to forward a datagram. Routing tables are managed using the Routing Table Maintenance Protocol (RTMP). In the AppleTalk for VMS system, routing table management is handled by Bridge Processes.

To minimize name lookup broadcast congestion, and to support the logical division of networks into sets, AppleTalk defines the concept of zones. A zone is an arbitrary collection of networks in the internet, which defines the boundaries within which names must be unique. Each network belongs to one and only one zone. Zone-to-network mapping is performed in bridges using a Zone Information Table (ZIT). Bridges manage the ZIT using the Zone Information Protocol (ZIP).

ZIP is intimately dependent upon the routing table information maintained by the RTMP handler. Information about changes in the table must be realized by both the RTMP and ZIP agents. The routing table should also be structured for fast lookup so that the routing of datagrams is not delayed by a time-consuming table lookup. Lastly, the tables must be configured to allow for mapping from network number to zone name and from zone name to the list of networks in the zone.

Internet routing

The primary purpose of an AppleTalk bridge is to serve as a datagram forwarding agent between networks in an internet. Nodes on a network send packets destined for a socket on another network to a bridge on the local network. The Bridge Process examines the destination address of the packet and uses its routing tables to forward the packet through the internet for eventual delivery to its destination node.

In the AppleTalk for VMS system, the internet routing agent is established by calling the `ATKSINIT_ATALK` Protocol Library routine with the address of the internet routing agent process. Specifying the routing agent argument causes the calling process to be initialized as a full internet routing agent. The routing process is called to route packets whose destination address is not a directly connected network. The internet routing agent uses the routing table to find a path to the packet's next immediate destination.

Broadcast request handler

As mentioned in a previous section, AppleTalk defines the concept of zones to minimize name lookup broadcast congestion in the internet. If NBP sent a directed broadcast to every network in the internet, considerable traffic would be generated. On AppleTalk, zones are formed to restrict NBP name lookups to the nodes in a zone. Since bridge processes are responsible for establishing and maintaining the mapping of network numbers to zone names, the bridge processes must participate in the name lookup protocol.

In the AppleTalk for VMS system, the Bridge Process participates in internet NBP name lookups through the use of an NBP broadcast request handler. This handler is established when the Bridge Process initializes the Protocol Library. The Bridge Process passes the address of the broadcast request handler to the `ATKSINIT_ATALK` routine. The Protocol Library calls this routine whenever a broadcast request packet is received.

The NBP broadcast request handler converts the incoming NBP broadcast request packets (`BrRq` requests) into zone-wide broadcasts of lookup requests (`LkUp` requests). It does so by searching the ZIT for a zone name match with the zone name in the `BrRq` request packet. It traverses the routing table entries extending from the ZIT queue entry with the matching zone name to obtain the list of all the networks in the desired zone. It then uses DDP to send a directed broadcast `LkUp` request to each network in the zone. The packet is broadcast on any directly connected networks and sent to the next bridge field in the routing table entry for any network not directly connected to the bridge.

Bridge ports

The Bridge Process uses the bridge port concept defined by AppleTalk to allow for multiple network connections. A bridge port is defined as a hardware port extending from a bridge device, with a unique physical address on the communication line or network to which it is connected. In the case of the AppleTalk for VMS Bridge Process, the bridge port is defined simply as a port ID assigned and managed by the AppleTalk for VMS Protocol Library. For each bridge port, the Bridge Process requests the Protocol Library to allocate and assign a port ID. In the AppleTalk for VMS system, the Bridge Process can have any number of bridge ports and it uses the port ID to identify each.

When the Bridge Process starts up, it scans a port descriptor file and opens the listed bridge ports. Currently, this file is defined as a list of port descriptor lines containing the port name, port driver name, the zone name, and a port information string.



Index

A

address
 AppleTalk 3
 internet 3
 physical 18
addressing, physical 18
AHLAP (Asynchronous Half Bridge Link Access Protocol) 14
ALAP (AppleTalk Link Access Protocol) 2, 5
AppleTalk
 address 3
 concepts 3
 implementation on VAX/VMS 7
 layered protocols 2
 Link Access Protocol (ALAP) 2, 5
 protocols 5, 6
 Transaction Protocol (ATP) 2, 6, 15
\$ASSIGN 11
Asynchronous Half Bridge Link Access Protocol (AHLAP) 14
ATK\$INIT_ATALK 19
ATK\$INIT_PORT 11
ATK\$OPEN_PORT 11
ATP (AppleTalk Transaction Protocol) 2, 6, 15

B

backbone bridge 4, 18
bridge 3, 4, 18
 backbone 4, 18
 half 4, 18
 local 4, 18
bridge port 20
Bridge Process 6, 7, 18, 20
broadcast
 messages 11
 request handler 19
 request packets 19
broadcasting 11
buffered I/O byte count quota 11

C

Carrier Sense, Multiple Access with Collision Avoidance (CSMA/CA) 10

D

\$DASSGN 11
Datagram Delivery Protocol (DDP) 2, 5, 15
datagrams 5, 6
DDP (Datagram Delivery Protocol) 2, 5, 15
DECnet Link Access Protocol (DLAP) 14
DLAP (DECnet Link Access Protocol) 14
driver, port 14
Driver, Virtual AppleTalk 6, 7, 12
dynamic node establishment 10

E, F

Echo Protocol (EP) 2, 6, 15
ELAP (Ethernet Link Access Protocol) 14
EP (Echo Protocol) 2, 6, 15
Ethernet Link Access Protocol (ELAP) 14

G

\$GETDVI 11

H

half bridge 4, 18

I, J, K

International Standards Organization (ISO) 2
internet address 3
internet router 4, 18
internets
 definition 3

routing 19
routing agent 19

L, M

languages, supporting vii
layered protocols 2
local bridge 4, 18
lookup requests 19

N

Name Binding Protocol (NBP) 2, 5, 15
NBP (Name Binding Protocol) 2, 5, 15
node establishment 10
node ID 3
nodes 3, 10-11
network number 3

O

Open Systems Interconnection (OSI) 2
OSI (Open Systems Interconnection) 2

P

PAP (Printer Access Protocol) 2, 6, 15
physical addressing 18
port
 descriptor file 20
 driver 14
 driver name 20
 ID 20
 information string 20
 name 20
ports 14
Printer Access Protocol (PAP) 2, 6, 15
Protocol Library 6, 7, 14, 15
protocols, AppleTalk 5-6

Q
QIO system service 11

R
routing table
 definition 4
 maintenance process 18
 Maintenance Protocol (RTMP) 5
 structure 18, 19
RTMP (Routing Table Maintenance
 Protocol) 5

S
socket clients 3
sockets 3

system calls, VMS
 \$ASSIGN 11
 \$DASSGN 11
 \$GETDVI 11
System Generation Utility
 (SYSGEN) 12

T
transaction, AppleTalk 6

U
unit control block (UCB) 10

V, W, X, Y
VAX/VMS system services vii
Virtual AppleTalk Driver 6, 7, 12

Virtual AppleTalk network 6
Virtual Link Access Protocol
 (VLAP) 10
VLAP communication 11
VLAP (Virtual Link Access
 Protocol) 10
VMS processes 6

Z
ZIP (Zone Information Protocol)
 2, 6, 15
zone 4
Zone Information Protocol (ZIP)
 2, 6, 15
Zone Information Table (ZIT) 18
zone name 20
zones 18

THE APPLE PUBLISHING SYSTEM

This Apple manual was written, edited, and composed on a desktop publishing system using the Apple Macintosh™ Plus and Microsoft® Word. Proof and final pages were created on the Apple LaserWriter® Plus. POSTSCRIPT™, the LaserWriter's page-description language, was developed by Adobe Systems Incorporated.

Text type is ITC Garamond® (a downloadable font distributed by Adobe Systems). Display type is ITC Avant Garde Gothic®. Bullets are ITC Zapf Dingbats®.

