Multimedia Applications and Network Management Support in Video DialTone ATM Network

by

Dang Van Tran

Submitted to the Department of Electrical Engineering and Computer Science in partial fulfillment of the requirements for the degrees of

Master of Science and Bachelor of Science in Computer Science and Engineering

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

May, 1994

Author .....
Department of Electrical Engineering and Computer Science
May 6, 1994

Certified by .....
Alexander Gelman, Member of Technical Staff
Bellcore (Bell Communications Research)
Company Supervisor

Certified by .....
Andrew B. Lippman, Associate Director of Media Laboratory
Department of Electrical Engineering and Computer Science
Thesis Supervisor

Accepted by .....
Frederic R. Morgenthaler
Chairman
Department Committee on Graduate Students

Multimedia Applications and Network Management Support in Video DialTone ATM Network

by

Dang Van Tran

Submitted to the

Department of Electrical Engineering and Computer Science

May 6, 1994

In Partial Fulfillment of the Requirements for the Degrees of Bachelor of Science and

Master of Science in Computer Science and Engineering

Abstract

Reliable transport protocols are necessary in the Internet world due to the underlying physical network. Network packets can be lost, misordered, delayed, or duplicated for a variety of reasons, all of which can corrupt the data that arrives at the destination. A key improvement can be made with some of the Video Dial Tone (VDT) systems is the communication among all the architecture's internal entities. These communication protocols are currently primitive and unreliable. One of the primary reasons for the unreliable and unstructured protocols in the VDT architecture is its embedment in a single monolithic manner with no operating system. Thus, the system lacks modularity, flexibility, and scalability. Enhancement, maintenance, and debugging of the existing protocols and system are very difficult. There is a definite need for a real-time operating system, and the communication protocols require redesign, re-implementation, and enhancement. Also to meet the fast changing standardized technology and specifications of the future, the redesigning of the protocols will focus on interconnection with other existing networks (such as the Internet). This thesis has two objectives. The first objective is describing a collaborated effort in investigating, designing, and re-implementing the problems of network management in Video Dial Tone architecture. The second object is commenting upon the current approaches, recommending any alternative or future efforts. The results of this research will be used to create generic requirements for the Video Dial Tone system.

Thesis Supervisor: Andrew B. Lippman
Title: Associate Director of Media Laboratory

Pages 3 and 4 intentionally omitted

Acknowledgements

This thesis is dedicated to my family. I would not be where I am today without their love and care.

More than ever, I would like to thank Alex Gelman, Dave Braun, Rajesh Khandelwal and Chet Day at Bellcore for all their guidance and help without fail throughout the time I have known them. But most of all, I would like to thank them for their patience, tolerance, and understanding. Thanks for everything guys !!!

I also want to thank Andy Lippman, my thesis advisor at MIT. Without his help in the class, the project, and the thesis, I would not be able to graduate. Thanks Andy for pulling me through.

Lastly, I would like to thank Thuy Ai, Donald Tanguay, and the whole VSA gang for always being there to care, comfort, and support me through their sincerity. Thanks everybody ...

Pages 6, 7, and 8 intentionally omitted

Table of Contents

List of Figures

Chapter 1

Introduction

1.1 Initial Motivation

Today's life-styles and work-styles have been transformed dramatically by the revolution of the information age; this transformation is not yet over. Of the many possibilities, imagine: shopping from stores without leaving home, playing interactive video games with someone from another country, watching a missed TV show later in the week without programming a video cassette recorder (VCR), viewing any of thousands of movies at any time and being able to interactively control televisions and pay-per-view programs (such as "select program", "play", "fast forward", "frame by frame", and "pause") without leaving one's couch, and communicating to another party via video telephony.

These possibilities are the driving forces and behind the research and advances made in network technology. Today's availability of wideband communication technology contributes greatly to these needs. Networks such as Asynchronous Transfer Mode (ATM), Broadband Integrated Services Digital Networks (B-ISDN), and Fiber Distributed Data Interface (FDDI) are becoming increasingly important due to their high bandwidths and low latency; and high bandwidth and low latency are important for applications such as those found in supercomputing, visualization, and multimedia.

In the past ten years, telephone companies have laid tens of thousands of miles of optical fiber, with a bandwidth (information-carrying capacity) greatly exceeding that of the copper wire it replaced. While some 65 percent of U.S. homes have cable TV, more than 90 percent have telephone service[10]. With this installation rate and pool of customers, there will be more than 50 million cable households equipped with digital equipment by the end of the century[11], and roughly 10 million[1] of those households will have access to the Video On Demand application (described later).

Also, as more and more network operators begin to upgrade their switching systems with broadband capabilities and experimenting with Synchronous Optical Network (SONET) and ATM switching, industry will begin to gear rapidly toward interactive video and multimedia applications for personal computers.

---

1. Predicted by John Hendricks, Chairman and Chief Executive Officer of Discovery Communication

Today, the telecommunications marketplace has awakened to a full range of information-handling service opportunities, including conception, transfer, storage, access, processing, manipulation, and presentation. This telecommunication transformation has stimulated a number of telecommunication operators and has resulted in a number of recent takeovers, financial participations in related companies, or plans for huge investments in the network. Also, a growing number of telecommunications, consumer electronics, entertainment, and video-game companies are listening to this market[15].

Recently, the Federal Communications Commission (FCC) proposed a model called the Video Dial Tone (VDT) system prescribing the scenario of telephone companies (TELCO) video and multimedia offerings. Bellcore's Multimedia Communications Systems Research Group[1] is prototyping its own VDT system, an architecture to meet the demands for these developing video and multimedia applications. This architecture will provide a platform for all these applications, and assumes that a Broadband Backbone Network and a Wideband Access Network will soon be common in the Public Switched Telephone Network (PSTN).

A key improvement can be made with some of the VDT systems (including Bellcore's) in the communication among all the architecture's internal entities. Reliable transport protocols are necessary in the Internet world due to the underlying physical network. Network packets can be lost, miss-ordered, delayed, or duplicated for a variety of reasons, all of which can corrupt the data that arrives at the destination. To meet the fast changing standardized technology and specifications of the future, the redesigning of the protocols will focus in interconnection with other existing networks[2].

This thesis has two objectives. The first objective is describing a collaborated efforts in investigating, designing, and re-implementing the network management in the Video Dial Tone architecture. The second objective is commenting on the current approaches, and recommending any alternative or future works.

Listed below are the procedural steps used to accomplish these objectives:

- **Learn Bellcore's existing Video Dial Tone architecture** (both its components and functionality)
- **Examine numerous operating systems** available and aid in determining which will best fit for this architecture
- **Learn that operating system** in detail (functionality, structure, code)
- **Examine numerous protocols** available and aid in determining which will best fit for

1. Group 21462 at Morristown, New Jersey.
2. Such as the Internet and all its interconnecting networks

this architecture

- **Learn those chosen protocols** in detail
- Aid in **porting the operating system** into the whole VDT architecture
- Aid in **porting/implementing the chosen protocol** onto the new OS
- Aid in **examining, modularizing, and filtering the desired existing network management code** onto the new OS
- Aid in **modifying and linking all the protocols** together to support the VDT architecture
- **Document and comment** all issues, problems, results, conclusions, and future recommendations

The results of this research will be used to create generic requirements for Bellcore's (and others') Video Dial Tone system.

1.2 Organization

The format of the rest of this thesis is as follows:

- **Chapter 2** summarizes general background information on the Video Dial Tone system, its supporting applications, and specifically the Video On Demand application.
- **Chapter 3** starts discussing problems with some of the VDT systems.
- **Chapter 4** describes the recommended solutions and their reasons and benefits.
- **Chapter 5** lists general technical background information on the recommendations and their related adjuncts for better understanding of the implantation in later chapters.
- **Chapter 6** briefs the implementation processes.
- **Chapter 7** shows all the major end-to-end components of the VDT system and their modification and effects.
- **Chapter 8** discusses general conclusions and possible future improvements and works.

Chapter 2

Video Dial Tone Background

2.1 Video Dial Tone Architecture Overview

Many of the Video Dial Tone (VDT) systems (including Bellcore's), also called Store and Forward Architectures (SFA), have three main network components[2]:

- **Broadband Backbone Network.** Based on ATM/SONET switches interconnected with high speed fiber links
- **Wideband Access Network.** Either a High-speed Digital Subscriber Loop (HDSL) or an Asymmetric Digital Subscriber Loop (ADSL) utilizing existing copper pairs
- **Public Switched Telephone Network.** Carries traditional telephony service, Plain Old Telephone Services (POTS)

The execution environment for applications consists of three different entities:

- **Consumer Premises Equipment** (CPE)
- **Central Office** (CO)
- **Information Warehouse** (IWH)

The CPE is customer's end equipment such as a set-top box or a personal computer. It is responsible for providing the user control interface and parsing of received data such as JPEG/ MPEG data decompression in case of still/moving images. It is also responsible for taking user control inputs and passing them to the Central Office entity for parsing. The Central Office (CO) takes care of data request generation and data flow management. Its primary purposes are to control, direct, and transport data from one application entity to another. The Information Warehouse (IWH) is basically a file server, responsible for providing all data types ranging from the application's executable code/scripts to archival program data.

Figure 2.1.0.1 illustrates the overall network architecture, linking the execution entities using various network components. Information Warehouses (IWH) and Central Offices (CO) are directly connected to the Broadband Network, taking advantage of its capability of massive data transfer to other entities such as other IWHs or COs. The CPEs are connected to specialized Central

Office Service Circuits (CO-SC) via the Wideband Access Network (i.e., ADSL or HDSL). These connections will give the network the flexibility to control wideband traffic and to monitor all data transfer rates and synchronization among all the entity components.

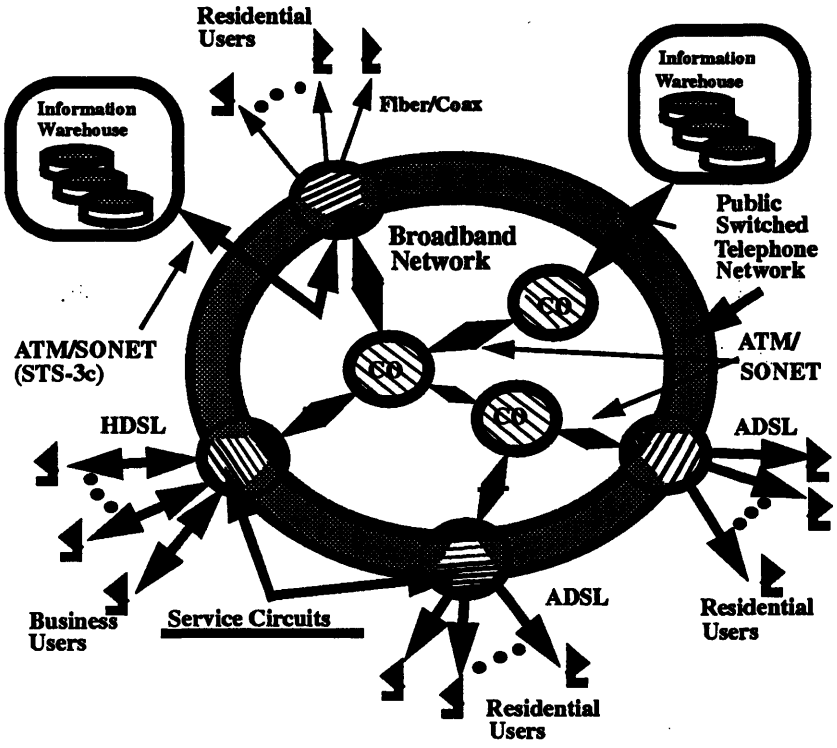

Figure 2.1.0.1: Overall Network Architecture[2]

This architecture is not only able to support numerous interactive applications but also allows the network to effectively utilize various technologies and media (such as Fiber/Coax, ADSL, HDSL, ATM/SONET) without requiring knowledge of the information content or format of delivered data.

2.2 Application Candidates

With the VDT architecture, many video and multimedia applications can be easily proposed and implemented. All these multimedia and interactive video applications are best categorized by the required access capability: Asymmetrical Transport or Bidirectional Transport[2].

Figure 2.2.0.1: Asymmetrical Transport

Asymmetrical Transport capability, depicted in Figure 2.2.0.1, provides a wideband data path toward the end user and a narrow band data path toward the central office. This type of transport can support information services such as:

- **Video on Demand** (VOD). Subscribers can instantaneously access and VCR-like control any selected programs (such as movies, music videos, or any multimedia programs) without leaving home to rent/purchase them.
- **Interactive Games.** Subscribers can interactively play any available networked video games without the trouble of leaving home to purchase an individual game box and rent/purchase video game cartridges/disks.
- **Home Shopping.** Subscribers can view, request information, and purchase products without leaving home.
- **Past Television.** Subscribers can access, customize, and view any broadcasted programs such as news, sports, dramas, etc.
- **Interactive Television.** Subscribers can interactively control numerous available features for a viewing program such as changing viewing perspective (top view, side view, close view, wide view, etc.).

Figure 2.2.0.2: Bidirectional Transport

Bidirectional Transport capability, depicted in Figure 2.2.0.2, provides wideband data both toward the Central Office and toward the end user. This type of transport can support information services such as:

- **Video Telephony/Conference.** Telephone service with live video images.
- **Video Mail.** Mail service using recorded videos or any multimedia clips.
- **Video Surveillance.** Live video images accessible from any remote locations.

With the current Video Dial Tone network architecture, applications using Asymmetrical Transport can readily be implemented. Thus, the first phase of the system experiment will target information access applications which require asymmetric access capability. These applications will be provided by the application vendors, for they will customize and alter the applications to their own needs. According to the Deloitte & Touche study[11], Video On Demand (VOD) related applications are expected to be among the most popular interactive multimedia offerings on the electronic highway[1]. Therefore, for demonstration and testing purposes of the proposed network, a sample Video On Demand (VOD) application will be implemented on the VDT system.

As for Bidirectional Transport supported applications, these applications will be delayed until enhancement of the prototype architecture is completed.

2.3 Video On Demand (VOD) Application

Traditionally, a movie is viewed from beginning to end. The advent of the Video Cassette Recorder (VCR) has added the concept of stopping, rewinding, and fast forwarding to conventional viewing habits. Using a VCR to watch rented movies at home has become a popular consumer activity. The convenience of watching a movie in one's residence, however, is not entirely problem-free. For example, demand for popular titles frequently exceeds supply. Of course, customers may try another day, may rent a less desirable title, or may walk out without making a purchase. But none of these alternatives fully satisfies the consumer's original expectations. An impulse to rent a movie at a time when local stores are closed is another potential source of customer dissatisfaction; the consumer will also need to make a longer trip than necessary - twice in fact, since tapes must be returned to the original place of rental in most cases.

This is where Video On Demand[2] (VOD) steps in to eliminate all of these dissatisfactions. The VOD application may be best described as an interactive "virtual Video Cassette Recorder (VCR)." With this application, one will be able to select any provided program in the comfort of one's own home and manipulate the program with VCR-like control such as: play, fast-forward, fast-reverse, pause, slow-motion, etc.

Figure 2.3.0.1 below illustrates the VOD application. When a subscriber "orders" a program,

---

1. Other offerings expected to be popular include electronic shopping, video games via network, airline passenger entertainment and communications systems and video conferencing.
2. Extension of Pay-Per-View and Near-Video-On-Demand (NVOD) applications

a remotely located file server, called the Information Warehouse, will download the selected program to that subscriber via a high speed network called the ATM network. The subscriber then can use a VCR-like infra-red remote control to manipulate the program.

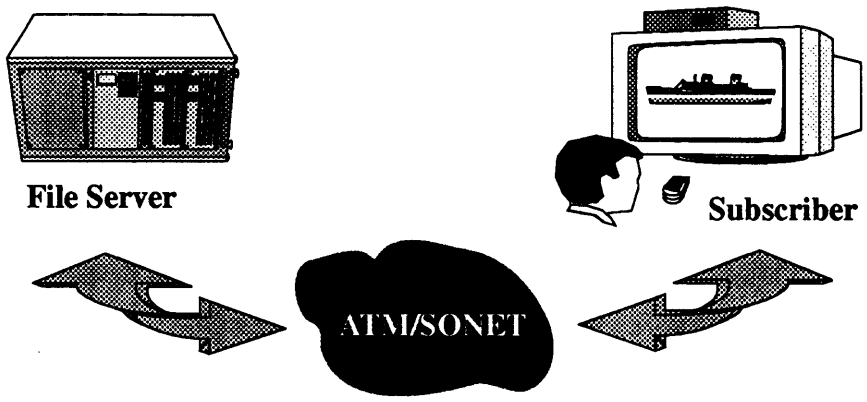

Figure 2.3.0.1: Video on Demand (VOD) Application

Today's technology is capable of offering such VOD service over existing telecommunication networks, on both classical telephone and CATV networks. One of the main advantages of such a network combination is delivery of a large selection of programs directly to any Asynchronous Digital Subscriber Line (ADSL)-equipped telephone residence at any time. Thus, the cost competitiveness against the video rental market has been reached thanks to a number of advances in the telecommunications and electronics industries:

- **Computerized Data Storage System.** Capacity of hard disks has doubled almost every year at a constant cost.
- **Image Compression.** Compression ratio for video has improved. Good quality video can be transferred through existing telecommunication networks using a compression scheme (i.e. able to transfer at a rate of 1.5-8Mbit/s using MPEG compression). Full movies can be stored in a few Gigabytes using compression.
- **Convenient Navigation System.** Just as a telephone caller has access to an unlimited number of telephone lines, ATM switching systems allow any bit rate to single or multiple customers. These switching networks can give the viewer access to a much greater number of channels.
- **High-Bandwidth Transmission.** Digital signal processing techniques allow the transportation of 1.5-6 Mbits/s over existing telephone wire over a distance between 1.5-6 km and 15 to 50 Mbits/s over one analog TV channel. Also, the current expansion of fiber-optic cable layout and connection makes it possible to transport at least 150 Mbits/s for a very long distance.

Chapter 3

Problem Analysis

This chapter presents an overview of the problems associated with the VDT system and describes some recommended solutions.

3.1 Network Stack Model

An ideal networking environment for a VDT system will be envisioned as a well structured protocol layering system. These protocol stacks consist of the following layers, as shown in Figure 3.1.0.1:

- **Application Layer**
- **Application Support Layer**
- **Transport Layer**
- **ATM/T1 Interface Layer**
- **Broadband/Wideband Physical Layer**

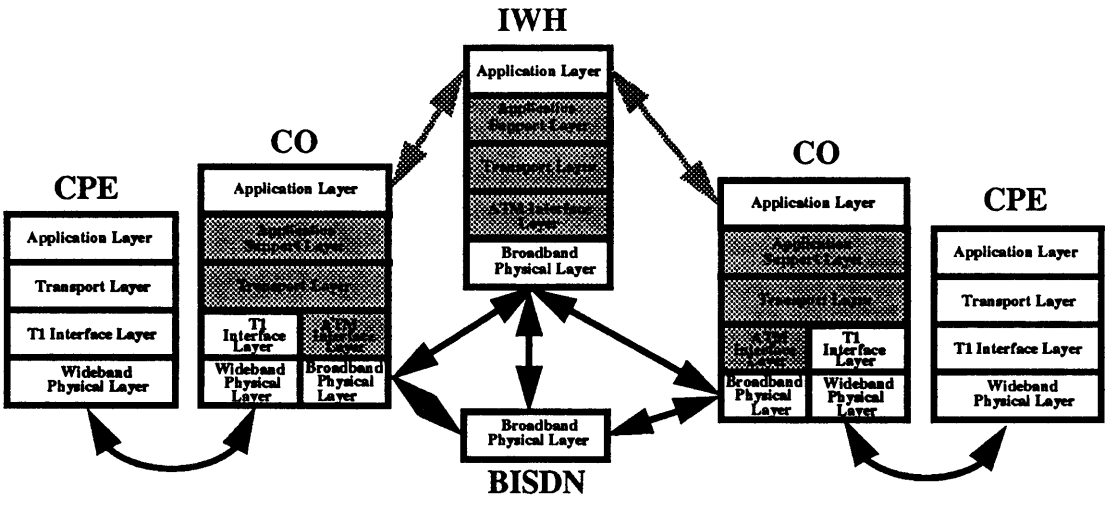

Figure 3.1.0.1: Video Dial Tone Protocol Stack Model

The Application Layer will contain the application programs used to implement the functionality of the applications (i.e. VOD). The Application Support Layer handles connection establishment and peer to peer data transfer among the applications running in the Application Levels. The Transport Layer is responsible for all the network management such as scheduling management, path reservation, and packet prioritizing. All these high level layers are linked to their ATM Physical Layer or T1 Physical Layer using the ATM Interface Layer or the T1 Physical Interface Layer, respectively. The Broadband Physical Layer and the Wideband Physical Layer are the physical hardware in the Video Dial Tone architecture, providing basic end to end packet transport for the whole network.

In the VOD application, all the application scripts will be stored at the Application Layer of the protocol stack, and all the low layers will serve as the interpreters and data transport mechanisms. The following is a simple example of a CPE entity's use of the protocol layering system to send its subscriber's command across the network to the CO side. When a subscriber presses "play" at the CPE end, the CPE application script stored in the Application Layer will direct the CPE entity to pass the "play" command to its connecting CO by using a simple functional call setup in the Application Support Layer. This application script does not have to be aware of the VDT architecture and its supporting protocols at all. All these network complexities are well hidden under the Application Support Layer. The Application Support Layer will then redirect the responsibility of sending the "play" command down to the Transport Layer by using the Transport Layer's available abstracted functions. Since the Transport Layer's responsibility is network management, it will then prioritize all its requests and process them accordingly. Eventually, when the 'send "play" command to the CO request is on top of the processing queue, the Transport Layer will use functional calls provided by the T1 interfaces (since CPE is connected to the whole VDT system by T1 protocol) to send the request across the network via the Wideband Physical Layer. The CO and the IWH entities will also use the same protocol layering principle as the CPE entity to send, receive, and process all the massive information exchanges among them.

3.2 Problems

This Video Dial Tone (VDT) system must provide reliable and standardized protocols for communication among all its various application processes. It is desirable if these protocols enable the system to interconnect with existing, widely available networks. These protocols will determine the performance and applications that will be available to the end-users of the system. Currently, the VDT architecture lacks a formal Application Support Layer and Transport Layer.

The Application Layer programs currently have to directly use low level function calls to perform certain tasks, such as accessing the hardware directly. The desired abstraction of invoking a user-friendly, machine independent communication protocol module is not there. The current non-standard protocols make it very difficult for outside application vendors to implement their own applications for the VDT system without the full-knowledge of its design and functionality. Thus, the absence of the Application Layer also reduces the portability of the Application Layer. This new Application Support Layer will allow application programmers to develop network-based applications without knowledge of the underlying transport protocols or communication facilities.

The network management within the Transport Layer is currently unreliable. This occurs for two reasons:

- **In the past, some organizations (including Bellcore) have elected to write their own real-time, operating system-like kernel for their own VDT systems.** This took a great deal of support and documentation and is vulnerable to the departure of key personnel. Also, without the existence of a formal operating system, maintenance and future enhancement of the system and its network management software will be difficult tasks. This problem will worsen exponentially as the network management becomes much more complex and makes it difficult to keep up with current fast changing networking technology.
- **End-to-end packet delivery is not guaranteed in ATM networks.** This is a design choice that is intended to improve overall efficiency by eliminating redundant error correction and recovery.

Therefore, the need for the creation of a user-friendly, familiar Application Support Layer and re-implementation for a more reliable, standardized, efficient, and robust Transport Layer[1] is required. The ATM Interface Layer will also need minor alteration due to modification of its higher layers. The effected layers are shaded in Figure 3.1.0.1 above.

As for the VOD application, the creation and reimplementation of these layers will enable the application to send reliable control/data messages among all the entities: Central Offices, Information Warehouses, and Customer Premises Equipment. These new layers also will give the VOD application (or any other applications) much greater flexibility of accessing data from other networks (*e.g.* Internet).

1. For real-time interprocess communication protocols within the VDT system

Chapter 4

Recommendation

An initial approach to alleviate these problems is to use a third party operating system. There are advantages to using a purchased, widely used operating system. A known, documented, and fixed environment simplifies large projects, and new generations of applications. Additional functionality to simplify the applications programming is likely to be available, because a generally accepted operating system available on a wide range of hardware promotes the development of third-party software products and documentation.

4.1 OS-9

OS-9[1] is the recommended operating system for Bellcore's VDT prototype system. This decision, reached by the group after the study of available alternatives, was based on numerous technical, financial, and practical factors. This thesis will only focus on the technical perspective of OS-9 and will use OS-9 as a "guideline" or a "template" for any other similar operating systems which might be chosen by different VDT research organizations.

OS-9 is an unusual operating system. It uses advanced techniques to allow it to address a very wide spectrum of applications effectively, and yet remain small. OS-9 also makes no demands regarding the hardware configuration it runs on. It can run programs with nothing more than a processor and a small memory. The following summarizes some of OS-9 special features[35]:

- **OS-9 modules.** It allows the operating system to be dynamically configured, for programs and operating system components to be in ROM, and for programs to be held in memory even when not running. Programs can also use memory modules as common data pools, for inter-process communication.
- **Re-locatable, re-entrant, ROMable operating system and programs.** Program does not use any absolute program addresses. Instead, program accesses such as calls to subroutines are done relative to the current program counter, so the program module can be placed anywhere in memory without modification.
- **Tree-structured I/O system.** The OS-9 I/O system is separated into file managers, device drivers, and device descriptors. This fragmentation of the I/O system into

1. Version 2.4 is available at the time of this investigation.

completely separate modules makes the OS-9 I/O system easy to customize.

- **Dynamically modifiable I/O system.** The OS-9 I/O system is dynamically modifiable while the system is running. New file managers, device drivers, and device descriptors can be loaded at any time. An I/O interface can be used in one way, and then used for a completely different purpose by loading a new file manager and/or device driver. Not only does this add to the flexibility of the system, but it makes debugging new I/O system components very much easier, as the system does not have to be rebooted to test each revision.
- **Customization hooks throughout.** Almost every aspect of OS-9 is customizable (often in more than one way) by providing well defined mechanisms to modify or extend the operating system.
- **User interface is similar to UNIX.** Users with UNIX experience can rapidly feel at home with OS-9 (although there are differences). The programmer's view of OS-9 is also very similar to UNIX, especially at the C programming level. Most UNIX programs can be ported (at the source code level) to OS-9 with little or no modification.
- **Regular utility command line syntax.** The utilities provided by Microware all conform closely to the same command line syntax. This significantly improves the user-friendliness of the operating system.

Regardless of OS-9's special features listed above, any chosen operating system should be truly powerful and versatile, enabling wider design and implementation flexibility without sacrificing system performance. After the new operating system is ported into the architecture, the next logical step is to find the desired protocol to enhance the VDT system with/by:

- **Reliable communication protocol** with efficiency as a focus (since the VDT system is a real-time system).
- **System extensibility** to interconnect with other existing networks.
- **Standardized and well-known protocols** so a large portion of the programming community has a good understanding of them.
- **Protocol is efficiently supportable by the chosen operating system.**
- **Protocol is implantable on most (if not all) VDT system's physical communication media** (i.e. copper twisted pair, coaxial, and fiber optic).

Luckily this protocol listed above already exists (no blank-page design needed), is able to perform all the desired tasks in one complete package (less protocol to learn and support), is strongly supported by the networking community (giving a strong networking foundation for implementation and maintenance). Best of all for Bellcore, Microware Systems Inc.[1] also has this protocol package developed to run under its OS-9 operating system (giving OS-9 another reason to be an

1. OS-9 operating system's vendor

excellent recommendation). Therefore, the protocol of choice for Bellcore's VDT architecture is the Internet Protocol Family (TCP/UDP/IP).

4.2 IP

IP is the dominant networking protocol set in use today. It has existed for numerous years, and is one of the most extensively analyzed and simulated pieces of software in the networking community. Numerous extensions and improvements have been made to it in the past decade to increase its performance and efficiency.

IP has its roots in the Department of Defense (DOD) dating back to the 1970s and was given broad support in 1983 when the Defense Advanced Project Research Agency (DARPA) stated that any computer connected to the Advanced Research Projects Agency Network (ARPANET also know as the Internet) must use IP. This had far reaching implications because of the number of computers involved.

Furthermore, because of its origins IP is considered public domain; hence many vendors have "created" IP protocol stacks to operate on different computers. In its early years IP was found in places such as universities, government agencies, research departments, nonprofit entities, and other predominantly non-corporate business; not to mention a significant presence in the world-wide marketplace.

Below highlights[26] some of the advantages of IP when integrated into the VDT system:

- **IP software is nonproprietary.** Any vendor wanting to develop IP software or a specific IP application can do it.
- **IP is reliable with a reliability VS. efficiency trade-off feature.** A system can switch between TCP (extremely reliable protocol) and UDP (somewhat reliable but efficient) transport protocols.
- **IP has extra applications/features** that enable the VDT system to expand its application possibilities. Among the most common are: TELNET, File Transfer Protocol (FTP), Simple Mail Transport Protocol (SMTP), and X protocol.
- **Its method of communication is based on a client/server architecture,** similar to the VDT architecture.
- **Communication between nodes on IP based architecture is generally considered asynchronous.**
- **IP has been implemented on many different types of media:** coaxial cable, twisted pair cable, and fiber optic cable.

- **IP can be implemented over numerous existing link layer protocols** including: Ethernet, Token Ring (TR), Fiber Distributed Data Interface (FDDI), Asynchronous Transfer Mode (ATM).

For Bellcore, IP support software can be obtained through Microware's OS-9/Internet Support Package (OS-9/ISP). OS-9/ISP is a hardware independent software package that provides communication between OS-9 and other Internet systems with IP. Since both the OS-9 operating system package and the OS-9/ISP protocol package are developed at Microware, porting the OS-9 onto the VDT system and the OS-9/ISP onto the OS-9 would be an ideal solution to optimize system throughput efficiency and minimize any porting incompatibility.

What follows is the introduction of the OS-9/ISP package in greater detail.

OS-9/ISP consists of four major software components: the socket manager (*SOCKMAN*), the protocol handlers (*TCP/UDP/IP*), the *mbuf* facility (*F$MBuf*), and the interface manager (*IFMAN*).

The socket manager provides program-level access to the network systems. The socket abstraction was designed as a part of the Berkeley Standard Distribution of UNIX (BSD4.x). This abstraction was designed to support multiple protocols and address families under one data-driven interface. *SOCKMAN* calls the protocol handlers on behalf of the user programs. *SOCKMAN* automatically binds and calls the protocol modules. *SOCKMAN* provides a calling interface and standard services (such as timers) to allow future protocols to be developed and integrated into the system without affecting existing protocols. Finally, *SOCKMAN* (with the cooperation of *IFMAN*) provides a list of active device drivers which are called by the lowest level of the protocol routines.

The Internet system uses the *mbuf* facility to dynamically allocate the memory it needs. This *mbuf* memory pool is allocated from the kernel when the Internet system is started. The Internet system then allocates memory exclusively from this *mbuf* memory pool. This makes memory requests for the Internet system faster than allocating memory from the kernel.

The interface manager (*IFMAN*) maintains information about configured network link-level interfaces in a hardware-independent manner. *IFMAN* is unusual compared to traditional OS-9 file managers in that it is a "passive" entity. *IFMAN* simply maintains a list of data structures that describe the characteristics and state of the network interfaces. It runs the execution thread which receives data from the device drivers and passes it up to the protocol handlers.

The data structure for the particular device contains all the appropriate information for protocol modules directly calling the driver. This is important because some of Bellcore's network data is not intended for any process on the target machine but rather for the network software itself (for example, routing or broadcast data).

See Figure 4.2.0.1 below.

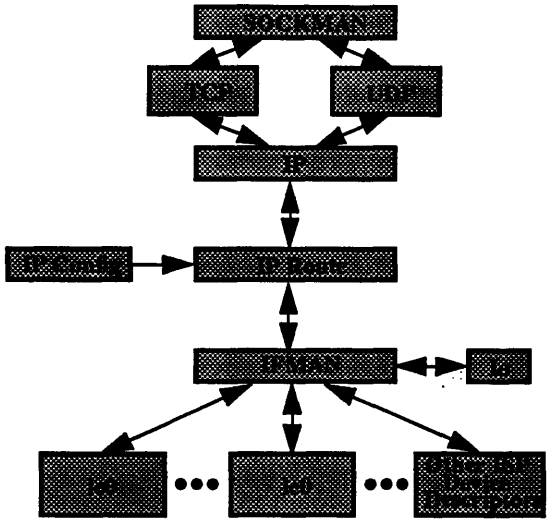

Figure 4.2.0.1: OS-9/ISP Protocol Stack[38]

When the VDT system has its new protocol support package implemented in its new operating system, the next step needed to complete the improvement of the Transport Layer is integration of the existing network management software with this new network development base. This will involve dissecting the existing network management code, picking out the needed code components (such as the various hardware interface routines), linking the appropriate components (specifically the ATM network management) to the installed IP protocol stack, and then compiling and porting the whole thing onto the VDT architecture using the new operating system.

4.3 Integration

Integrating networks means taking dissimilar network protocols and assimilating them in such a way that interoperability is achieved. This requires understanding devices, concepts, and terminology in both environments. This is especially true when attempting to integrate IP into other existing protocols because of its great complexity.

A simple way to incorporate the newly installed protocol IP into the existing ATM network is to layer the ATM network beneath the IP protocol stack at the IFMAN level. In this way, ATM provides the physical transport only (analogous to an ethernet, a token ring, or even a serial line).

Visualize this integration into a layered model (bottom to top order):

- A layered implementation would have the ATM layer move cells through to the **ATM hardware FIFOs.**
- The next layer would perform **Segmentation and Reassemble** (SAR), possibly using linked buffers since the SAR layer does not know the total message length and cannot allocate a continuous buffer of the total size necessary on reception.
- A **Convergence Sublayer** (CS) on top of SAR would insert or remove the CS header and trailer.
- **Internet Protocol** (IP), which provides routing, fragmentation and reassembly, would be layered on top of CS.
- **User Datagram Protocol** (UDP), which provides connectionless service and demultiplexing of packets to user endpoints, and **Transmission Control Protocol** (TCP), which provides connection-oriented, reliable stream communication.

The main advantage for having IP encapsulation inside the ATM network is to utilize IP's reliable transport for the ATM network (sacrificing some efficiency, of course). With this method, an initial packet of data will first be encapsulated by TCP or UDP. Then, IP will encapsulate this new TCP/UDP datagram to form an IP datagram. The IP datagram will be segmented into small size packets (44 bytes each), and each of these packets will be combined with ATM/CS/SAR headers and trailers to form ATM cells. These tasks take place at the CS and the SAR layers. Then these ATM cells will be transported to their destination via fiber optic cables. Figure 4.3.0.1 depicts this method.

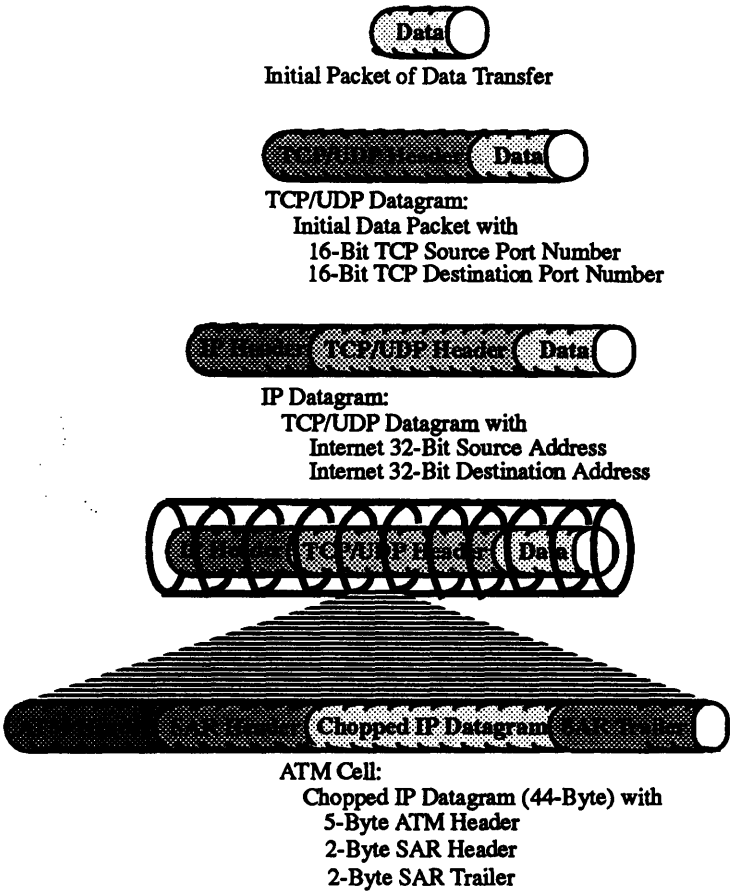

Figure 4.3.0.1: TCP/IP-ATM Protocol Stack

At the receiving end, the ATM interface will extract out the ATM/CS/SAR heading, reassemble all the available packets (some might be lost) into an IP datagram and pass this to the IP layer. The IP layer will deencapsulate its datagram and pass the contents to the TCP layer or the UDP layer depending upon the original message type. The TCP layer will examine its TCP datagram and determine if it is error-free (by recalculating and comparing the checksums). It will acknowledge the receipt of the datagram and pass the data part to the application.

When integrating IP and ATM networks, special connection management and routing mechanisms are required because ATM uses its own signaling protocols to establish and tear down connections. These protocols are different from the ones used by higher-level protocols such as IP.

For the moment, let us assume that switching/routing decisions will be made based on the ATM protocol address rather than the IP address inside the public carrier's network. At the CO, the IP address of the destination is used to determine the appropriate ATM connection to forward the packet segments on. The destination IP address is used to determine the ATM's VPI/VCI address

of the destination CO or IWH entities. A table mapping the IP address to an outgoing VPI/VCI is needed to perform this function. When the CO maps the destination IP address to the appropriate public carrier protocol address (VPI/VCI), the whole IP packet will ride inside of the ATM protocol's cells to the destination CO or IWH. Since, the sending entity's VPI/VCI address appears in the sending packet, the receiving entity can use it as the sending address if data transmission response is needed.

4.4 Application Support Library

Upon completing the development of the Transport Layer, the Application Support Layer can be built using the available network interfaces of the VDT system. A proposed set of application support library functions will provide an interface to support all the interprocess communication over the VDT network. This library will hide the complexity of the network structure and code, enabling the application writers to concentrate only on the application itself.

This proposed set of library functions varies very much in its size and structure for a variety of reasons. A system can just have a basic set of functional application support library if that is all its creator needs or there are simply not enough physical resources (i.e. memory) to support a more complicated one. Nevertheless, the application support library set should have, at a minimum, the following functional characteristics:

- **Setup Status.** This function sets the necessary parameters before any transportation takes place. Examples of these parameters are: reliability vs. efficiency status, retransmission timer, and buffer sizes for both receiving and sending ends.
- **Setup Send.** This function sets the necessary parameters before sending a message. Examples of these parameters are: destination location/type, type of message (i.e. data or control), message importance, clear sending buffer, and resizing the buffer.
- **Setup Receive.** This function sets the necessary parameters before receiving a message is possible. Examples of these parameters are: type of switch (i.e. data only, control only, any type), receiving from certain destinations only, clear receiving buffer, and resizing receiving buffer.
- **Receive Message.** Read a message from the receiving buffer with status specified by the Setup Receive function.
- **Send Message.** Send a message to the sending buffer with status specified by the Setup Send.function.

These basic communication functions can be written incorporating the currently available net-

work interface software in the VDT system.

A Bellcore application support library will merge the network interfaces provided by Microware's OS-9/ISP package (SOCKMAN) and Bellcore's own existing hardware interface routines.

Chapter 5

Technical Specification

This chapter contains the technical background information on the operating system selected by Bellcore. This information will be helpful and/or necessary for the future discussion of the implementation, results, and recommendations in later chapters.

5.1 OS-9

An operating system is defined as a body of software that provides functions to allocate and manage the hardware resources. It divides up the processor's time between multiple programs running concurrently, allocates memory to programs as they need it, and arbitrates between programs trying to use the same input/output (I/O) device.

OS-9 is an operating system by Microware Systems Inc. The following summarizes the main features of OS-9[35]:

- **Multi-tasking.** OS-9 performs automatic time-slicing between any number of programs. OS-9 also has several mechanisms to allow the advanced programmer to control the scheduling of these programs.
- **Real Time.** Real-time means that real world events are being processed as they happen. A real time system is one that must respond to an external event within a specified time.
- **ROMable.** The unique memory module mechanism of OS-9 makes the operating system and application programs inherently ROMmable. Also, there is no need to provide any information as to where in ROM the modules are - at startup the OS-9 kernel scans all the ROM areas to find all modules present in ROM, and builds a module directory indicating where they are.
- **Modular.** The operating system itself is separated into several memory modules. This allows very easy customization. If certain functionality is not required, that module is simply omitted. If additional functionality is required (even dynamically at run-time), additional modules[1] can be loaded, and their functionality is immediately available.
- **Unified Device Independent I/O System.** A device independent I/O system is one in which the same basic functions (i.e. *open*, *read*, and write) are used by a program to

1. Such as Internet Support Package (ISP) package for Internet networking

access all types of I/O devices. This simplifies programming, and permits "redirection".

- **Inter-process Communication Functions.** OS-9 has several different inter-process communication functions available for use by application programs. They are very important to the effective generation of multi-tasking applications.
- **High Performance.** OS-9 was designed with execution speed and code size very much in mind, and so was written in assembly language. This makes it very well suited to even small, cost-sensitive products.
- **Adaptation to New Hardware.** The modular arrangement of the I/O system and other parts of the operating system makes OS-9 very adaptable to new hardware, without the need for any of the source code of hardware-independent parts (i.e. kernel). The user can also customize the operating system, by adding or removing memory modules. This applies to all parts of the operating systems[1].
- **Complete Set of Functions.** OS-9 provides a full set of functions for the management and allocation of all resources. Although OS-9 can be reduced for low-end applications, all of its functions are sophisticated enough to support top-end applications as well.
- **Broad Spectrum of Applications.** This comes from its modular, ROMmable construction, its full set of sophisticated functions (including multi-user support), and its relatively small size. OS-9 can be used in small, diskless, embedded products[2], on large multi-user systems, and on everything in between, such as personal computers and home computers[3].
- **The Future.** The advent of devices like Phillips Compact Disc Interactive[4] (CD-I) player may well make OS-9 the most popular operating system, since the operating system in the CD-I player is OS-9 (under the name CD-RTOS[5]) and all CD-I applications run under OS-9. OS-9 might become the standard operating system in industrial products, home computers, personal computers, and workstation.

5.2 IP

The Internet Protocol (IP) is the Internet datagram delivery protocol. IP is a lower-level protocol located above the network interface drivers and below the higher-level protocols such as the User Datagram Protocol (UDP) and the Transmission Control Protocol (TCP). The IP layer provides for a checksum of the header portion, but not the data portion of the datagram. IP computes the checksum value and sets it when datagrams are sent. The checksum is checked when datagrams are received. The IP layer also supports segmentation and reassembly. If the datagram is larger than

1. Even the kernel can be adapted or extended, using "kernel customization modules".
2. Such as the VDT system's CPE, CO-CP, IWH-CP, and LC.
3. In the future, these PCs can be used as CPE instead of set-top boxes.
4. CD-I is also a candidate for a set-top box.
5. Standard developed by Phillips in conjunction with Sony.

the Maximum Transfer Unix (MTU) of the network interface, datagrams are fragmented on output. Fragments of received datagrams are dropped from the reassembly queues if the complete datagram is not reconstructed within a short time period. If an error is discovered while a datagram is in the network interface driver layer, the error is passed to the user process.

The Transmission Control Protocol (TCP) is layered on top of the IP layer. It is a standard transport level protocol that allows a process on one machine to send a stream of data to a process on another machine. TCP provides reliable, flow controlled, orderly, two-way transmission of data between connected processes. You can also shut down one direction of flow across a TCP connection, leaving a one-way (simplex) connection.

Software implementation of TCP usually resides in the operating system and uses IP to transmit information across the underlying network. TCP assumes that the underlying datagram service is unreliable. Therefore, it performs a checksum of all data to help implement reliability. TCP uses the IP's host level addressing and adds its one per-host collection of port addresses. The endpoints of a TCP connection are identified by the combination of an IP address and a TCP port number. The TCP packets are encapsulated into the IP datagrams.

The User Data Protocol (UDP) is also layered on top of the IP layer. UDP is a simple, unreliable datagram protocol that allows an application on one machine to send a datagram to an application on another machine using IP to deliver datagrams. Conceptually, the important difference between UDP datagrams and IP datagrams is that the UDP includes a protocol port number, allowing the sender to distinguish among multiple application programs on the remote machine.

UDP datagrams are not reliable. They can be lost or discarded in a variety of ways, including a failure of the underlying communication mechanism. UDP implements a checksum over the data portion of the packet. If the checksum of a received packet is incorrect, the packet is dropped without sending an error message to the application. Each UDP socket is provided with a queue for receiving packets. This queue has a limited capacity, and any datagrams that arrive once the capacity of the queue is reached are silently discarded. The UDP packets are also encapsulated into the IP datagrams.

The International Standard Organization (ISO) created a model for evaluating networks (or modeling networks) in the late 1970s; they called it the Open Systems Interconnect (OSI) reference model. Understanding the OSI model network layers and where protocols, functions, and components fit respectively is helpful because it can be used as a baseline to understanding IP. Knowing which lower level protocol is supported by a specific upper level protocol is important. Most upper layer protocols define which lower level protocols they support. IP on the other hand does not specifically define which lower level protocols must be used because it is versatile. The following depicts the seven layers and their names as defined by the ISO organization[26]:

- **Layer one is the physical layer.** From a component perspective, this is typically an interface card, or part, in a host. At this layer data is represented by voltages or light pulses. This layer is responsible for generating the voltages or light pulses and transmitting them onto media for transmission. This is the lowest layer within a node (a node being a general term used to refer to a device on a network or in a stand-alone position). A few examples of interfaces represented at this level include: RS-232, RS-449, V.35

- **Layer two[1] is the data link layer.** It serves two basic functions. First, it serves the function of establishing, maintaining, and gracefully ending a logical connection of two interface cards. The data link layer has two sublayers: the Media Access Control (MAC) and the Logical Link Control (LLC). The MAC sublayer is next to the physical layer. It is responsible for framing data, then passing it to the physical layer for transmission across the media. The LLC sublayer is responsible for the establishment, maintenance, and termination of the logical link between two nodes.

- **Layer three[2] is the network layer.** Here software begins to play a significant role. Routing, flow control, and messages relating to routing are performed here. (For example, in a IP network, the IP addressing scheme is implemented at this layer). This layer is not always implemented in software; sometimes it is implemented in firmware, but this is usually contingent on the purpose of the particular network device, such as terminal servers.

- **Layer four is the transport layer.** Here lies the part of the protocol responsible for getting data from the source host to the target host. End-to-end control between a source and a destination node is done here. Transport layer protocols are generally characterized as being either connection oriented or connectionless oriented. They are also characterized as being reliable, in the sense they perform retransmissions if necessary; or unreliable if they do not perform retransmissions. Examples of transport level protocols are: Transmission Control Protocol (TCP) and User Datagram Protocol (UDP).

- **Layer five is the session layer.** This layer provides network functions necessary for the initiation, activation, termination, and release of sessions (logical connections). Synchronization, transaction program control, and the type of data transfer is specified here.

- **Layer six is the presentation layer.** Here data is formatted, message syntax is determined, and a data stream protocol is selected. Functionally, this layer selects a presentation syntax and encrypts data (if desired).

- **Layer seven**, in the OSI model, consists of three parts: **User Element** - represents the end points of the user information. **Application Services** - examples recognized by OSI include X.400 (message services), X.500 (directory services), File Transfer and Management protocol (FTAM), others exist and can be vendor or protocol suite specific. **Common Application Services** - protocols that can select the type and structure of conversations between users. This also provides control protocols: for

---

1. List of data link level protocols applicable at layer two include: Internet, SDLC, Token Ring, Fiber Distributed Data Interface (FDDI), 802.3, Enterprise System Connection Architecture (ESCON), 802.4 (Token Bus), HDLC, Parallel Channels.
2. The following are some examples: SNA, TCP/IP SPX/IPX, NetBios.

example, recovery, commitment, and concurrence.

Regardless of which protocol is used to send messages across a network, each host is assigned a 32-bit Internet address (or IP address). IP addresses are usually represented visually as four decimal numbers, where each decimal number encodes one byte of the 32-bit IP address. This is referred to as dot notation. Specifying IP addresses in dot notation allows determination of the network class from the three high-order bits:

- **Class A.** Used for networks that have more than 65536 hosts. Seven bits are allocated to the *netid* and 24 bits to the *hostid.*
- **Class B.** Used for intermediate-sized networks with between 256 and 65536 hosts. Fourteen bits are allocated to the *netid* and 16 bits to the *hostid.*
- **Class C.** Used for networks with fewer than 256 hosts. Twenty-one bits are allocated to the *netid* and only 8 bits to the *hostid.*
- **Class D.** Used for multicasting, sending a message to a group of hosts connected to the Internet.
- **Class E.** Virtually unused and considered experimental.

Address classes provide a way for assigning more networks and fewer hosts, the same amount of networks and the same amount of hosts, and fewer networks and more hosts. Also, the IP address has been defined to allow extraction of either the hostid or the netid. Gateways base routing on the netid and depend on such efficient extraction.

5.3 ATM

Asynchronous Transfer Mode (ATM) is discussed here since it impacts the VDT system directly. ATM is the primary transport protocol used in the Video Dial Tone system. The following is an overview of its technical background information.

ATM was the method selected by the Consultative Committee of International Telephony and Telegraphy (CCITT) in 1987 to serve as the technological basis for the future Broadband Integrated Services Digital Network[1] (BISDN) networks, the proposed global fiber-optic network that integrates and replaces the current separate networks for data, voice, and video communication. This is because ATM networks can be designed to satisfy all of the timing and semantic require-

1. Conceived as an all-purpose digital network

ments for BISDN. Transmission rates of 155.520 Mbits/s and 622.080 Mbits/s for the ATM network have been defined and recommended by the CCITT.

ATM is a packet oriented transport mode that is based on small, fixed-length cells. ATM cells have a fixed length of 53 bytes. Each cell is composed of a 5 byte header field and a 48 byte data field. The ATM header, with the virtual path identifier and virtual connection identifier (VPI/VCI), is used to route the cell to the destination. The other 48 bytes of the cell are the payload, which carries user data.

The 5 byte header field of an ATM cell has the following format[9]:

- **Virtual Channel Identifier** (VCI) (12 bits). Allows the network to support virtual connections over the same communications line. VCIs are unique only over a single network node input port and are usually mapped to other values upon leaving each node. The mapping of the input VCI to the output VCI basically defines the routing function of an ATM network node.
- **Virtual Path Identifier** (VPI) (16 bits). Allows the network to support semi-permanent connections between endpoints so a type of virtual network can be supported.
- **Payload Type** (2 bits). Specifies the type of data in the payload. Currently the default bit value of 00 has been defined for user information.
- **Reserve for future use** (RES) (1 bit).
- **Cell Loss Priority** (CLP) (1 bit). This field defines whether a cell has a "high" priority (0) or a "low" priority. Low priority means that the cell can be discarded by the network if necessary without significantly degrading the quality of the associated service.
- **Header Error Control** (HEC) (8 bits). This field is used to detect and correct bit errors in the header field.

The size and functionality of the header field in ATM packets is considerably reduced in comparison to headers used by other packet-switching networks. Its basic function is allowing the internal nodes of the network to correctly route the packets to their destination. This is accomplished through the use of Virtual Circuit Identifier (VCI) and Virtual Path Identifier (VPI) fields contained in the header.

During transmission, messages are broken up into cells according to one of several ATM Adaptation Layer options (AAL). There are 4 generic AALs currently defined: AAL1, AAL2, AAL3/4, and AAL5. Among these adaptation layers, AAL5 standardization is the most advanced. More importantly, AAL5 is also able to support IP with available transfer load capacity.

Every host in an ATM network is connected to one or more switches that route packets between hosts and other switches[1]. A connection is established or torn down using specific signaling

protocols running on top of the Convergence Sublayer (CS) and is identified by the VPI/VCI fields on the ATM header. Cells from a given connection are routed over a path determined at connection establishment time and cannot be miss-ordered or duplicated by the network. But, cells may be lost due to corruption, congestion, and buffer depletion at intermediate switches and endpoint nodes. ATM networks do not support any mechanisms for correcting any type of cell errors. Neither the ATM hardware nor the adaptation layers provide for automatic acknowledgment and retransmission. Damaged cells are simply discarded.

5.4 Bellcore's Data Interface

In order to ease the complication of sending and receiving a message using an ATM network, a network interface will be used. Bellcore will be using a custom ATM Data Interface (ADI). This interface was developed at Bellcore to create a user-friendly interface layer on top of Bellcore's own prototype ATM network. This board provides a hardware interface to the ATM physical network. The following family of functions provides an interface to the ADI board:

- **diNew()** creates an object representing the ADI board and returns a handle to that object. All the other functions take this handle as their first argument.
- **diSet()** initialize the board's type, buffer size, and VCI.
- **diDelete()** deletes the ADI object.
- **diCSinit()** must be executed before interrupts are enabled, so that critical sections are handled properly.
- **diCSbegin()** and **diCSend()** can be used by applications for ADI related code that should not be interrupted.
- **diSet()** and **diGet()** provide a means of writing/reading information to/from the board.
- **diGo()** instructs the board to perform a data transfer.
- **diReset()** resets the board.
- **diRestart()** resets the board and reloads all of the parameters that were previously set.
- **diIntrClr()** clears an interrupt, which includes clearing event bits for which interrupts are not masked.
- **diSend()** and **diRcv()** provide a means of sending and receiving messages over the ADI's ATM link. **diSend()** sends a message to an ATM vci. **diRcv()** receives a message into a buffer.

---

1. In contrast to networks such as Internet, a connection must be established between any pair of hosts through intermediate switches before they can communicate.

- **diSendStart()** and **diRcvStart()** are similar to **diSend()** and **diRcv()** but they only set up the data transfer and do not actually initiate the transfer of data.
- **diMsgsInfo()** places a vector of structures into the receiving queue.
- **diQueueLen()** returns the number of items in the specified queue.
- **diQueueClr()** removes everything from the specified queue and returns the number of items that were removed.
- **diFreeSize()** returns the amount of free memory in the receive queue.
- **diErrCode()** returns the last error code. **diErrCnt()** returns the count of errors and resets the count and last error code to 0.

Chapter 6

Implementation

This chapter will focus on the work needing to be done on Bellcore's VDT system. The steps required are specific to the products chosen for Bellcore's VDT system (OS-9 operating system and OS-9/ISP protocol package) The following discussion might not reflect what other VDT implementors may require. Hopefully, Bellcore's approach will provide an overall idea or guide-line to some of the necessary steps. In general, these tasks are divided into five major steps:

- **Installing the chosen operating system** into the VDT components
- **Porting the chosen protocol's modules** onto the installed operating system
- **Porting all indispensable existing works** onto the new operating system
- **Linking the ported protocol's modules** into the old existing protocols (mainly the ATM)
- **Building an application support user-interface** for future application scripts.

These implementing steps on Bellcore's VDT system are discussed in detail below.

6.1 Installing OS-9 Operating System

Giving the VDT components a formal operating system environment enhances the whole VDT system in that it provides much greater modularity and scalability as well as the flexibility to support current maintenance or future upgrades.

6.1.1 System Requirement

There are two types of OS-9 systems:

- **Host System.** The development system used to edit, compile, and assemble OS-9 source

files.

- **Target Systems.** The systems[1] on which OS-9 will actually run.

On Bellcore's host system, these are the (minimum) hardware requirements for OS-9:

- **A Sun computer** running SunOS 4.x.
- **A hard disk.** This is for storing the OS-9 cross development system; speed is not an issue.
- An **RS-232 serial port** for communication with the target systems for debugging purposes.
- **A EPROM programmer** that can accept data from the host system to create EPROMs.

For Bellcore's target systems, the (minimum) hardware requirements are the following:

- **68000 family CPUs.**
- At least **128K RAM.**
- At least **64K ROM**[2]
- **Two serial I/O**[3] **ports;** one for a terminal and one for communications with the host system.
- **Any other I/O devices**[4] which OS-9 must eventually support. These are not used in the initial installation steps.

6.1.2 Installing the Distribution Package on Host System

The distribution package contains a large number of files that make up the operating system and its utilities. A few files are source code text files. Most others are object code files. The entire distribution package is copied into subdirectories according to major subsystems of the distribution packet. These are the major subsystems loaded into the host system: *I/O*, *CMDS* (*BOOTOBJ* is also ported inside *CMDS*), *ROM*, *DEFS*, *LIB*, *SYS*, *SYSMODS*, *CBOOT*, *MACROS*, and *SCCI*.

1. Components in the VDT system such as CPE's, LC's, IWH-CP's, and CO-CP's microprocessors.
2. Both the RAM and the ROM should be somewhat larger (2 MB and 128K respectively) on some systems to allow for debugging utilities and code.
3. RS-232
4. Such as Random Block File Manager (RBF), Sequential Character File Manager (SCF), Sequential Block File Manager (SBF), T1, VME, Differential Fast Bus (DFB), (additional) RS-232, RS-423, TCP/IP, and ATM

Two debuggers are available in the distribution package for use with boot ROM code: *debug* and *RomBug*. *RomBug* is chosen since it is a full-featured debugger. It supports all of the features of all 68000-family processors and also supports symbolic debugging. But *debug* is also used on systems where the amount of available ROM space is limited.

6.1.3 Booting Process

OS-9 boot code consists of several different files linked together and programmed into ROM. This ROM bootfile is simply a merged OS-9 kernel and program modules, with the kernel being the first module.

These are the three steps performed by the OS-9 system initialization (boot) code:

- **Power up to the ROM debugger prompt.** Once this is done, the processor will execute machine language instructions like it normally does.
- ***RomBug* prompt to kernel entry.** In this step, the *SysBoot* finds the bootfile, finds the kernel, sets up the registers according to the kernel's specifications, and jumps to the execution entry point in the kernel.
- **Kernel entry point to *$* prompt.** This part of the kernel initialization sets up the variables in the system global data table. It also builds the kernel's RAM memory pools, builds the module directory, initializes the system tables, opens the console device, sets the current directory to the boot device, executes any initialization modules from the Init module's *Extens* list, and forks the first process. The boot code then disinherits the first process and exits by calling the kernel's system execution loop.

6.1.4 Porting OS-9 onto Target Systems

Four steps are required to port OS-9 onto the target systems. Of course, for different target systems, these steps vary a little.

- **Porting the boot code.** This involves creating and installing a ROM that contains the system initialization code and a special ROM debugger. All the boot code is compiled, assembled, linked, and download to an EPROM programmer to make a set of EPROMs. At first, the ROM debugger is included in the ROM to provide a boot menu. But, the final "production" ROM version would have no ROM debugging facilities to minimize ROM memory space. This involves modification of these files: *boot.a, ioxxx.a, ioyyy.a, sysinit.a, systype.d, sysboot.a,* and *vectors.a.*

- **Porting the OS-9 kernel and basic I/O system.** This involves more modification to the *systype.d* file. The Init module and high-level serial drivers and descriptors are made for each particular target system's hardware. ROM based OS-9 system exists once this is completed and is working. The following are the steps used to accomplish the task: create the Init module, create a Console I/O driver, prepare the download file, and download and run the system. The following modules are the minimum requirement needed for a basic OS-9 system: *kernel, sysgo, cio, scf, shell, Init, Term,* and *scxxx.* The following utility modules are also installed on most target system since they are quite useful: *csl, mfree, mdir,* and *math.*
- **Creating customized I/O drivers and finishing the boot code.** In this porting procedure, more high-level drivers are developed and debugged for other serial ports, disk drivers and controllers, clocks, and any other available devices. Once the high-level drivers are working, modification of the boot code is made to boot from the various devices which are then available. The following are the steps used to accomplish the task: creating disk drivers, testing the disk drivers, testing the disk boot, making a new boot/debug ROM, creating and debugging additional drivers, adding the additional OS-9 modules and testing them, editing and re-assembling the *init* and *sysgo* modules, and making final versions of the boot ROM.
- **Testing and Validation.** This is the final testing and verification of the complete system. The quickest basic test of a new installation is to start using the system immediately. But a thorough testing and validation include: kernel tests, serial I/O[1] tests, disk I/O[2] tests, clock tests, other supported I/O devices tests, and multi-user interactive operation tests.

The installation of the OS-9 operating system into both the host and the target systems is now complete.

6.2 Installing IP Modules onto OS-9[43]

Installing the Internet management modules into the operating system will create a powerful networking base for Bellcore's VDT system. These modules will establish reliable communication among VDT components and give the flexibility of interacting with the Internet network for greater application possibilities.

The Internet modules for the OS-9 operating system are delivered in Microware's OS-9/ISP package. The modules in this package are divided into two classes: target system independent and target system dependent.

---

1. Sequential Character File Manager (SCF) devices.
2. Random Block File Manager (RBF) devices.

6.2.1 Target System Independent

Target system independent modules are pre-made, ready-to-run modules. The following system independent modules are needed:

- **Program utility modules**: *arpstat, bootptest, ftp/ftp_nocsl, ftpd/ftpd_nocsl, ftpdc/ftpdc_nocsl, hostname, idbdump, ifstat, inetstat, ipstat, ispstart, mbinstall, routed, telnet/telnet_nocsl, telnetd/telnetd_nocsl, telnetdc/telnetdc_nocsl, tftpd,* and *tftpdc.*
- **System modules**: *SysMbuf, af_ether, af_unix, ifloop, ifman, inet, ip, lo0, pk, pkdvr, pkman, pks, sockdvr, sockman, tcp,* and *udp.*

Each of the different hardware entities of the VDT system has a different hardware/protocol specification (different ROM, RAM, communication connection, etc.), requiring only a subset of the target system independent modules. Those not needed are omitted to conserve memory.

The following system modules are omitted on some VDT target components (such as the CPE's microprocessor and IWH Service Processor) since the *telnetd/telnetdc* daemon programs are not used/needed: *pk, pkdvr, pkman,* and *pks*. The following system modules are address family modules and are also omitted on those target components since they are not used[1]: *af_ether, af_unix,* and *inet (af_inet)*. The following protocol modules again are omitted if they are not used on some target components (such as the CPE's microprocessor): *tcp, ip,* and *udp*. But, the remaining modules are vital to the Internet on the OS-9 system and are not omitted: *SysMbuf, ifloop, ifman, lo0, sockdvr,* and *sockman.*

6.2.2 Target System Dependent

In order to use target system dependent modules, they first are customized to each individual target.

For VDT system's components that use Internet access (i.e. Control Processor in Central Office entity), the following information is obtained before porting the target dependent modules onto the VDT system: component's host name, host IP-address, network broadcast, IP-address[2], name and IP-address of hosts and networks with which this component will communicate.

Next, is the installation of the Internet modules onto the OS-9 operating system of the devel-

1. Socket descriptor for these components are changed to not link to the left-out modules
2. Subnetwork mask also if needed

opment components. The following are the steps necessary to produce a working Internet port on OS-9:

- **Step 1: Make the Internet Device Descriptors.** Change the current data directory to the appropriate ISP descriptors subdirectory in the configuring component's port directory. Edit the *if_devices* file in this subdirectory, customizing the host IP-address (*inetaddr*), network broadcast address (*bdaddr*), and subnet mask (*submask*) fields for the appropriate descriptors[1]. Depending on the component's Internet interface, customization of other fields in the *if_devices* file and/or the descriptor file in the appropriate descriptor subdirectory is needed. Run *make* to update all of the base Internet descriptor files and re-make all of the Internet descriptors.
- **Step 2: Make the Socket Device Descriptors.** Change the current data directory to the appropriate ISP socket descriptor subdirectory. Edit the *socket.a* file in this subdirectory, customizing the host name (*net_name*) and Internet device names (*device_names*). Customize other fields in the *socket.a* file as required. Run *make* to remake the socket descriptor.
- **Step 3: Make the IP Configuration Module.** Since dynamic routes are desired on the CO and the IWH entities (one entity can send messages to another remote entity by using the intermediate linked entities in between; these intermediate entities act almost like gateways on the Internet), alteration of the shipped IP configuration modules is not needed.
- **Step 4: Make the Internet Database Module.** Change the current data directory to the ETC directory. Edit the hosts file. Change the host *IP-address/host* and *name/host name aliases* combinations to include only the hosts which the component knows about. Edits the networks file. Change the network *name/network IP-address/network name aliases* combinations to include only those networks which the component knows about. Edit the protocol file. Change the protocol *names/protocol numbers/protocol name aliases* combinations to include only those protocols which the component use. Edit the services file. Change the services *name/port number/protocol name/service aliases* combinations are use. Run *make* to remake the internet database module.
- **Step 5: Transfer Internet Modules to VDT Components.** All the relevant Internet modules are transferred to the target systems. This consists of making a new bootfile, programming new EPROMs, and copying the Internet modules to a floppy/hard disk for loading upon the next boot[2].

---

1. The component may not be assigned an IP-address with a hostid containing all zeros or all ones as these addresses are reserved for broadcast use only
2. An EPROM emulator makes debugging ROM based routines much easier by eliminating the repetitive reprogramming of EPROMs.

6.3 Porting Existing Works onto OS-9

Since Bellcore's network transport layer previously had no operating system, all network management and connection handling code were developed in a monolithic manner. Recreation of new code from the existing code is needed before porting into the new operating system. These are the necessary steps in porting existing programs to the OS-9 environment:

- **Redesign the network coding.** This step involves planning and designing of what needs to be done. This can be accomplished by first examining the available code resources, then determining what is needed to be done to complete the task.
- **Dissect the Existing Code.** Scrutiny of the existing code implemented in the old VDT system is needed in order to determine which section of the code is re-usable or modifiable and what is needed to be implemented or trashed in order to work on the new system.
- **Modularize Existing Code.** After gathering the reusable code, it is necessary to organize the code to a modular form for both ease of implementation and integration with other sections and future modification.
- **Implement Necessary Code.** Due to the transfer of code from one system to another, lots of components' code needs to be implemented in order to perform properly.
- **Relink Existing Code.** This step involves integrating all the code components (both old and new) together into a working unit.
- **Port New Code to VDT Components.** This step involves programming EPROMs and installing them into the appropriate VDT components.

The VDT system is now ready to run in the new environment, under the OS-9 operating system. But this system does not add any new features (such as the application support library function), nor is it more reliable (such as benefiting from the IP protocol) since none of these features are ported into the system yet. The system has the same functionality as before, with, perhaps, a slight degradation in the overall system performance due to additional overhead of the new operating system.

6.4 Linking TCP/IP in OS-9 to ATM network[35]

Linking TCP/IP protocol to an ATM network in Bellcore's VDT system directly means encapsulation of the TCP/IP protocol into the ATM protocol. The TCP/IP datagram will be segmented into ATM cells and travel through the network. This implementation involves the follow-

ing steps:

- **Examine Bellcore's ADI** in detail. The ADI is used as the main interface to the ATM network. All ATM cells pass through this interface.
- **Examine the OS-9/ISP package** in detail. Dissecting the OS-9/ISP components is necessary to determine the appropriate section to modify.
- **Modify the IFMAN section** of the OS-9/ISP package. The Interface Manager (IFMAN) provides the socket protocol modules with a hardware-independent layer for configuration and management of the network communication interfaces. ADI drivers must be developed and added to IFMAN's list of available interfaces.

6.5 Building The Application Support User-Interface

Application support libraries make an application programmer's job easier by providing them with an abstract view of the VDT system. The libraries can contain either a few simple routines or many complex ones, whatever best suits the system implementors' wishes.

Since the application support library to be created will be used for prototype testing and demonstration purposes, only basic functions will be implemented. These are the functions included in the application support library:

- **setstatus()**. This sets the necessary general status of the sending or receiving message.
- **setsend()**. This sets the necessary specific status of the sending message.
- **setreceive()**. This sets the necessary specific status of the receiving message.
- **recvmsg()**. This receives a message.
- **sendmsg()**. This sends a message.

Due to the popularity of the socket abstractions of IP, the library mimics that approach. Programmers familiar with sockets and TCP/UDP/IP networking software will see the similarity. this will save training time.

Similar to the IP interface, the following are the necessary steps for sending and receiving a message in the new application support library:

- **Create** the socket. This creates the socket, specifying the address format, type, and protocol to be used on the socket.
- **Bind a name** to the socket. This assigns a user specified name to an unnamed socket.

- **Connect** to a "listening" socket. This opens a socket in an active state, initiating a connection with a specific passive sockets.
- or **Listen** for and then accept a "connecting" socket. This opens a socket in a passive state, waiting for an active socket to connect to it.

Once a full connection has been made between two sockets, data may be read from or written to either socket using the familiar functions: *read(), write(), recv(), recvfrom(), send(),* and/or *sendto().*

Chapter 7

Experimental Results and Discussion

All major components and associated interconnection networks of the VDT system will be examined with emphasis on the impact of the improvements previously discussed. Additional comments will be introduced where appropriate.

7.1 Customer Premises Equipment

The Customer Premises Equipment (CPE) is the part of the VDT system by which users communicate their selections. It will be available to all VDT's subscribers, but its type and functionality depends upon which application the viewer subscribes to.

Bellcore currently has two working research prototypes, differing in their level of sophistication. These prototypes demonstrate the potential of the openness in the Video Dial Tone architecture. This open architecture allows the CPE to range from a simple set-top box, to a Compact Disc-Interactive (CD-I) player, or even a personal computer.

Bellcore's first CPE prototype is a simple set-top box which extracts the wideband digital signals from the subscriber loop, decodes (using MPEG v2.1) the compressed video and audio, and presents the audio and video signals to the television set in a suitable format (NTSC, PAL, or SECAM). It also routes customer's requests from the selector function in digital format back to the CO. The intelligence providing the interactivity resides either in the customer's LC or in the IWH depending on the service being provided.

The other prototype represents much greater "intelligence" than the simple set-top box prototype. It is a CD-I player with an OS-9 operating system installed into its microprocessor, giving this CPE a locally intelligent interface between the customer and the rest of the system. It will act as a customer's agent in dealing with both the network and the information provider. Its microprocessor technology includes the new operating system to handle a combination of fixed and downloadable programs. This feature offers maximum flexibility to facilitate the VDT system as an open-systems architecture. Refer to Figure 7.1.0.1.

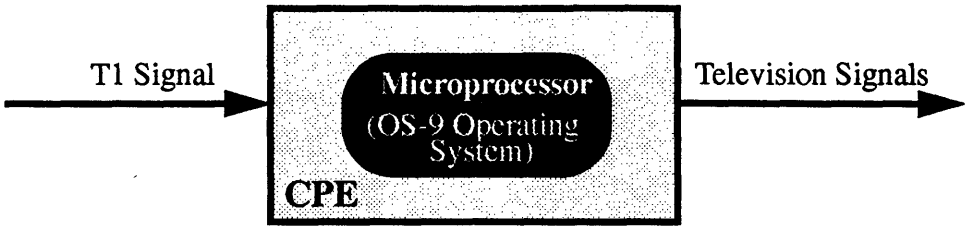

Figure 7.1.0.1: Intelligent Customer Premises Equipment (CPE)

The use of an OS-9 operating system and stored program control, coupled with the ability to modify the program by downloading, is an extremely powerful way to reliably provide video delivery and, at the same time, be able to add features or change system functions now or in the future. For example, if the VOD application is selected by a subscriber, an appropriate IWH will download to that subscriber's CPE all the necessary application scripts such as a menu of next program choices, a VCR-like control interface, and a decompression scheme (i.e. MPEG). But, if the subscriber spontaneously decides to select an interactive-game application instead of VOD application, new interactive-game application scripts will download to the CPE, replacing the VOD application scripts.

Thus, OS-9 would give this CPE greater flexibility to embody a wider range of programmable handlers and features such as:

- **Indicate status** of the CPE, network transmission, and program source
- **Display** the CO and/or IWH-originated **messages and menus**
- **Transmit subscriber choices** to the CO and/or IWH
- **Provide user control functions** such as control of TV, VCR, interactive-shopping, alarm transmission, utility monitoring, and interface to home computer.

Since the set-top box CPE can only perform simple functions (such as receive encoded/compressed data, decode/uncompress data, output data, and receive then pass control inputs), all of its "intelligence" to perform complicated tasks resides in its Line Card (to be discussed in section 7.3.2) in the Central Office.

7.2 Consumer Premises Equipment-Central Office Connection

Bellcore has developed an approach that permits video transmission through the ubiquitous

copper wires that already enter homes. The technology is called Asymmetric Digital Subscriber Line (ADSL), and it allows an ordinary copper telephone line to carry a high-speed digital signal while simultaneously transmitting a regular telephone voice conversation. ADSL provides a 1.5Mbits/s T1 channel toward the customer and a 16Kbit/s bidirectional channel.

Passive ADSL splitters are used at both ends of the copper pair to separate the POTS and ADSL signals. The POTS signal intercepted at the CPE end connects to a device such as a telephone instrument. On the CO end, the POTS signal will be handled by Telco's POTS switch system. As for the ADSL bitstream, both the CPE and the CO use T1 interface devices to control the high-speed digital signal. The T1 interface on the CPE end is most likely to reside in the CPE itself, while the T1 interface on the CO end lies in the Line Card, a component inside the CO, designated for each CPE. See Figure 7.2.0.1 below.

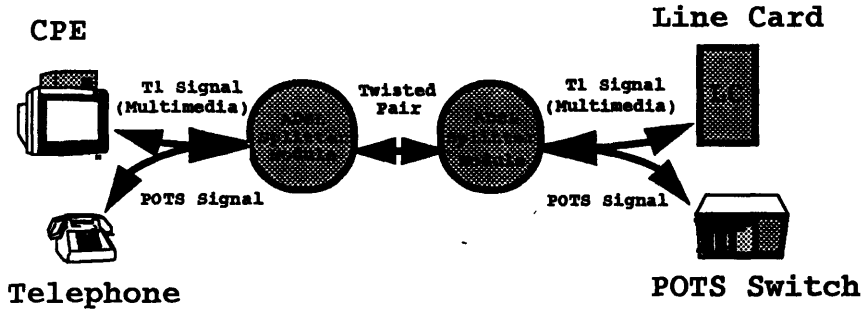

Figure 7.2.0.1: CPE-CO Connections

The VDT system at Bellcore running the VOD application transmits a video program over existing copper twisted pairs using this ADSL technology. The data stream, formatted according to the MPEG standard, delivers full-motion video and stereo audio to the subscriber's television set. Unlike cable TV, there is a separate transmission path to each subscriber, permitting each one to view a different program. In contrast to the cable television system, encryption (scrambling), a procedure designed to limit reception to intended parties only, is no longer necessary since a video is not transmitted to the subscriber until his request is validated, and then it is placed on a customer-specific copper pair.

Tests indicate that ADSL will work on only about 75% of the installed copper wire pair telephone lines[12]. Another deterrent to deployment is that the available bandwidth is only 1.5Mbits/s. But improvement in this area can be made with the installation of newer versions of ADSL for better bandwidth. Also, standard AT&T Paradyne ADSL technology can be used, which would evolve into an adaptive ADSL technology that will permit multiple video sources to be transmitted

to each subscriber. Finally, optical fiber is becoming cost effective as a means to reach subscribers essentially eliminating the bandwidth availability problem.

7.3 Central Office

The Central Office (CO) provides support for all its connected CPE's functions, including supplying raw data, application scripts, accounting, and switch control. It is also responsible for network management of its own entity and network support for other COs and IWHs.

The major components in Bellcore's CO are (refer to Figure 7.3.0.1):

- **Line Card** (LC). There are 64 LCs in a single rack in the CO. Each Line Card is designated for a single subscriber connection and is connected to their CPE.
- **Control Processor** (CP). There is only 1 CP for every 64 LCs.
- **ATM Interface.** There is 1 ATM Interface for each CP. This is used to link the CO to the VDT ATM network.

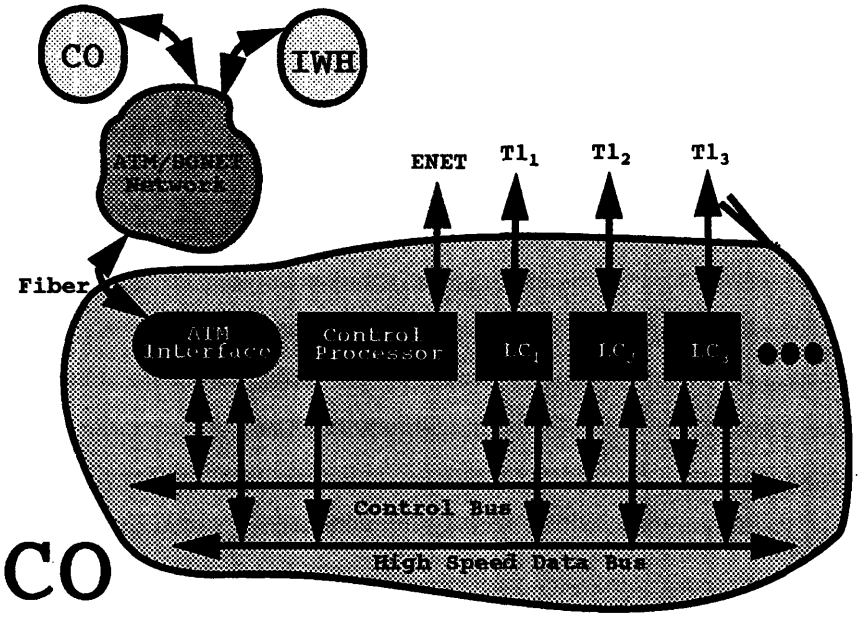

Figure 7.3.0.1: Central Office (CO)

All these major components listed above are linked by the following CO's internal connections (refer to Figure 7.3.0.1):

- **Control Bus (CB).** This is an internal connection between the CP and its LC's. This connection will carry only control data. This is a standard computer bus.
- **High Speed Data Bus (HSDB).** This is an internal connection between the ADI and the LC's. This connection is used to carry the bulk raw data and has a bandwidth of at least 150 Mb/s. The CO will use this connection to transfer data bulk (such as video and multimedia data) from the ATM Interface to the LCs.
- **T1.** This is an external connection. This connection is used to connect each individual LC to its CPE via the ADSL modems for this LC's CPE. The bandwidth for the T1 is 1.544 Mb/s.
- **Fiber optic.** This is an external connection. This is the connection to the ATM network. This connection will enable the CO to communicate with other CO and IWH entities.
- **Ethernet.** This is an external connection. This line is used for numerous TELCO purposes (such as Level 1 gateway[1] and billing).

When a subscriber message from the CPE, traveling through its T1 connection, arrives at its LC, the LC will mark its identification stamp (the LC identifier) and pass the message to the CP's processing queue via the Control Bus. The CP prioritizes its queue and processes the messages accordingly. This algorithm uses information in the messages such as arrive time, due time, priority ranks, and emergency status.

If the CP processing involves connecting and transferring data to the ATM network (such as retrieving data from some specified IWH or passing a message to certain other COs), the CP will pass these messages through the ADI via the CB.

The implementation of this project has tremendous effects on the CO specifically and the VDT system generally. It helps make the CO into a more manageable and modular entity.

The next few subsections will detail the CO's components, their modification, and effects.

7.3.1 Control Processor

The heart and soul of the CO entity reside in this component called Control Processor (CP). The flow of data in and out of all the CO's data paths is controlled by this CP. This CP regulates all the traffic among the CO's entities via it's CB. Besides handling timing and scheduling for the LCs, the CP also takes care of its internal communication among itself, the LCs, and its connecting

1. Used for "white-paging", mapping different types of network addresses (i.e. Telephone number <-> miniWAN.Cluster.LC number <-> IP address <-> ATM address (VPI/VCI))

ATM Interface.

Bellcore currently uses Motorola's VME167 board for its Control Processor.

Figure 7.3.3 below displays the connections available for the Control Processor.

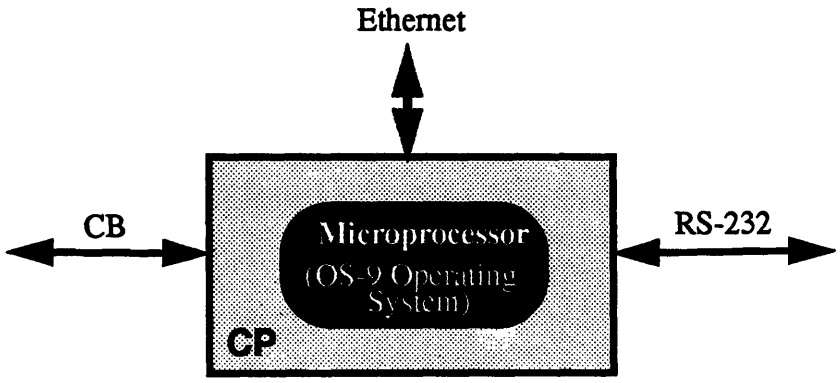

Figure 7.3.1.1: Control Processor (CP)

With the CB connection, the CP is able to retrieve all the required status and interrupt information from its LCs and ATM Interface. The CB is also used for sending out any requests, commands, or interrupts to these local entities. The RS-232 connection is used primarily for debugging purposes (such as printing or observing logging status information).

The Ethernet connection is used to connect to a TELCO operations system. Its applications are undetermined since they will be changed, updated, and added as the VDT system matures. But these are some of its possible applications: white paging, level 1 gateway, billing, and security. Also, the microprocessor inside the CP will run the new operating system, OS-9. This should aid maintenance and future enhancement of the CP a great deal.

7.3.2 Line Card

The Line Cards are the gateways to the end users. Bellcore is using custom-made cards for its LCs. This board has two main components (refer to Figure 7.3.2.1):

- **Buffer Processor (BP).** The Buffer Processor's sole function is to move data between the HSDB and the T1 interface. The BP controls the flow of data from the HSDB and it's buffers, and between it's buffers and the T1 interface. It also handles data coming from its end user via a narrow band RS-232 line and passes it directly to the SP for parsing. During all these data transfers, a limited amount of reformatting of

the data may also be performed in order to be compatible with the different CPE types and data rate requirements[1].

- **Script Processor (SP).** The SP can be considered to be the main "brain" behind the LC. It runs the Application Script (AS) loaded from an external source (such as from the selected IWH) and manages communication between itself, the BP and the CP.

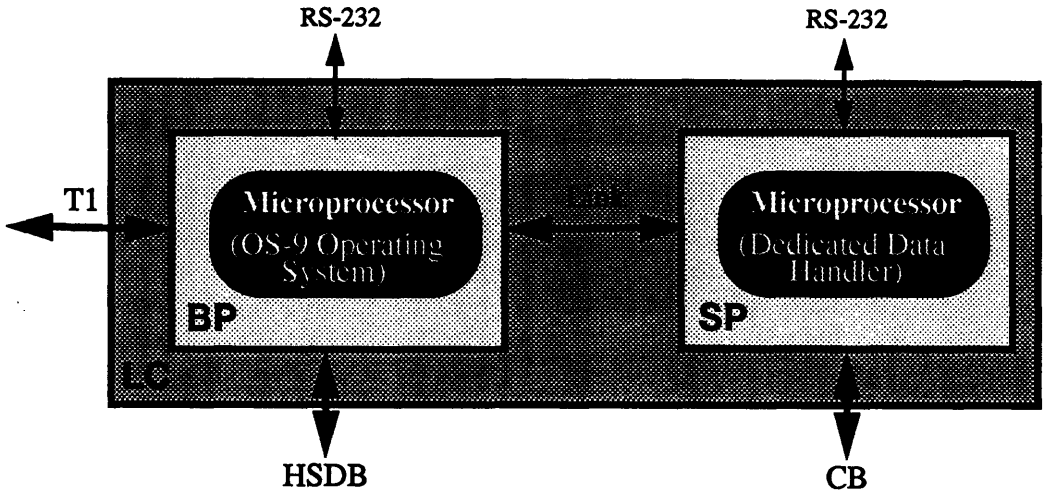

Figure 7.3.2.1: Line Card (LC)

The CB is attached to the Script Processor, while the HSDB and T1 are connected to the Buffer Processor. The two processors are able to communicate with each other via a high-speed serial link. Again, the RS232 connections attached to each processor are used primarily for debugging purposes. In some configurations, one of the RS-232 links may be used for the upstream channel from the CPE and an RS-423 connection might also be used for intercom in the future. Both of these serial connections will be carried on the 16Kbits/s ADSL channel.

In some configurations, the Application Script (AS) is downloaded from a selected IWH. The IWH transports the requested AS through the ATM network, through the ATM Interface, through the HSDP, through the BP, through the high speed serial link that connects the BP and the SP, and finally to the SP. This method might seems cumbersome, but it is very effective in making the VDT more dynamic and robust. If a running LC crashed, installation of a new LC is needed, or upgrade of the LC system is desired, having the ability to download new code is an advantage. In the future, the Application Scripts may be provided by the application vendors[2].

As depicted in Figure 7.3.2.1 above, the OS-9 operating system is ported into SP's microprocessor. The BP will run a fixed routine tailored to efficiently move data between the HSDB and the

1. This serial link is placed on the 16Kbits/s channel provided by ADSL.
2. Such as video rental, video game, cable, retail-store, etc.

T1 interface.

7.3.3 ATM Interface

Bellcore is using its experimental research prototype interface to establish connections, read, and write messages to their ATM prototype network.

When data flows into or out from the ATM/SONET network, it has to pass through the ATM interface. Besides serving all the required tasks that a regular ATM interface does (i.e. Segmentation and Reassembly (SAR), insertion/removal of headers and trailers, etc.), the proposed ATM interface will have an extra component called ATM Header Switch. This component will direct the flow of both raw data information (such as archival programs or application scripts (AS)) and control information (such as user interface controls, segmentation requests, and acknowledgment) between the ATM network and the line card microprocessors. See Figure 7.3.3.1.

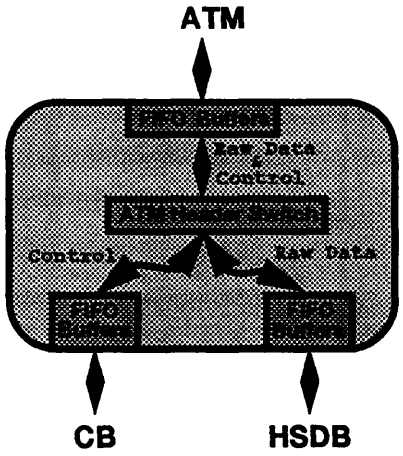

Figure 7.3.3.1: ATM Interface

In the VDT system, the bulk data (i.e. video and multimedia data) flows through the High Speed Data Bus (HSDB) while the control data (i.e. requests, messages, and interrupts) flows through the Control Bus (CB). The ATM Header Switch in the ATM Interface uses information in the ATM header to differentiate and direct incoming cells from the ATM network (*i.e.* cells are marked as either data cells or control cells[1]). These incoming cells will be buffered in the ATM

1. Control and data cells are differentiated using a bit in the address portion of a cell's 5 byte header.This means that there are two VCI/VPI addresses involved in connections between an IWH and a CO for each LC, one for control messages and one for bulk data differing in only one bit.

Incoming FIFO and will be directed to either the CB Outgoing FIFO or the HSDB Outgoing FIFO depending upon the header information. Similarly, the ATM Header Switch will generate cells from data in either the CB Incoming FIFO or the HSDB Incoming FIFO and place the packets in the ATM Outgoing FIFO. If no data is available, empty cells will be generated automatically.

7.4 Central Office-Information Warehouse Connection

The ATM network provides interconnection among the CO and the IWH entities. The network fabric of the VDT testbed is a self-routing implementation, routing cells based on headers generated by the ATM Header Switch. The ATM cell has a fixed-length of 53 bytes, a 5 byte header and a 48 byte payload. ATM is a standard for information transfer developed as part of the overall BISDN protocol stack, which is being standardized by the CCITT and ANSI.

As for the new IP protocol stack's usage effects on the physical network, there are basically none. ATM switches do not recognize the type of data they are carrying. The only effect one might notice is upon performance, since the overhead of the IP protocols would add traffic to the ATM network. There must be a trade-off for network reliability, and network efficiency is the sacrificed parameter.

7.5 Information Warehouse (IWH)

The VDT system requires a storage and retrieval system for its service applications. This facility is referred to as the Information Warehouse (IWH). The IWH are different from other computer servers because of the vast storage required for motion pictures and the high real time throughput requirements for video delivery. The IWH provides random, simultaneous access to frequently used data such as top-hit movies, music, interactive programs, and software (this is represented by On-line Disk in Bellcore's IWH). The IWH also provides for sequential, batched access to on-line archival media, such as digitally stored programs on disks and tapes (this is represented by Archival Tape in Bellcore's IWH). The inventory of information is cost effectively distributed to users, generating revenue streams for service and network suppliers.

Bellcore's IWH prototype currently has storage for 12 Gigabytes (1 Gigabyte contains approximately 100 minutes of MPEG compressed video) of information in its On-line Disk, while its Archival Tape has room for 10 tapes with each tape capable of storing approximately about 12 Gigabyte (10-12 movies) or a total of 100-120 movies.

The IWH's architecture is similar to the CO's. The following are the major components in the IWH (see Figure 7.5.0.1):

- **Control Processor (CP).** This is considered to be the brain of the IWH. It has the same hardware architecture as the CP in the CO. Its basic function is very similar to that of the CP in the CO; but it is designed to serve and regulate the IWH internal networking.
- **On-line Disk (OD).** This is used to store information that is currently in use or is frequently requested. This disk is designed for random, fast access and data transfer.
- **Archival TAPE (AT).** This is used to store information that is not currently in use. This tape is designed for massive space storage, not for fast access.
- **Disk Controller (DC).** This controls the On-line Disk.
- **ATM Interface (ATMI).** This is used to interface the IWH to the ATM network. It has the same hardware architecture and functionality as the ATM Interface in the CO.

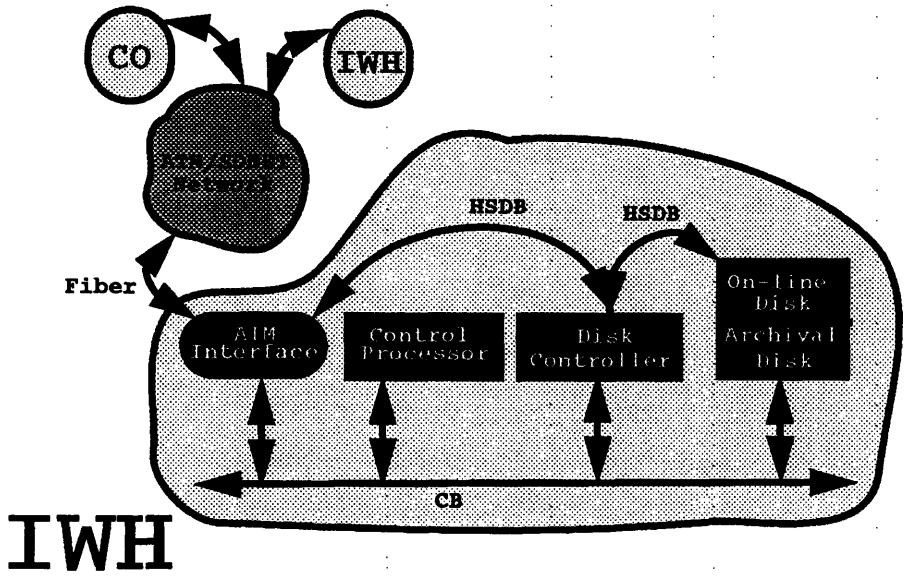

Figure 7.5.0.1: Information Warehouse (IWH)

The IWH's architectural design is similar to that of the CO. But, while the CO's primary objective is to serve the VDT system by providing network management and services, the IWH's primary objective is to serve the VDT system by providing information. Thus, instead of having LCs in the architecture as the CO, the IWH has disk storages (i.e. On-line Disk, and Archival Disk) and disk storage controller (i.e. Disk Controller). In the IWH, OS-9 is ported into the CO. Sharing the processing load among multiple CPs can also provide for the graceful recovery from failure of disk storage equipment or application software.

7.6 Internet

This is an interesting amendment to the VDT system. The VDT system is no longer considered to be an isolated information network thanks to the Internet connection. The VDT becomes able to exchange information with all other networks connected to the Internet. The possible benefits are limited only by the application creators' imagination. Of course, there will be lots of social and political issues to be considered such as who will have access and what information will be available to them. Security will also be a major issue when dealing with integration of public networks. In any case, these issues are definitely beyond the scope of this thesis.

Chapter 8

General Conclusions and Future Works

The overall goal of this thesis is to present recommendations for enhancing the VDT architecture's network management and reliability and for creating a user-friendly support system for future application software development in the Video Dial Tone system. These recommendations include porting of OS-9 operating system and the OS-9/ISP package into the VDT architecture and development machine, modifying the existing network management code, integrating them with the new protocols (TCP/UDP/IP), and compiling all code into the VDT system under the new operating system.

8.1 Evaluation

Unfortunately, by the end of the internship assignment associated with this thesis, benchmarking was only beginning. No data is yet available on the efficiency vs. reliability issue and its improvement. What follows is an extrapolation.

8.1.1 Performance

Protocol and operating system overhead has become one of the limiting factors for communications performance on fast networks using ATM. Protocol redundancy arises when layering higher level protocols on top of lower level ones. The functionality found in higher-level protocols such as TCP/IP is made redundant by services typically found in the lower-level network interface. With multiple protocol layers and repeated passes over the same data, protocol processing quickly becomes the bottleneck in the pipeline between communicating applications running on different hosts. ATM, for example, requires specific support for connection management, flow control, congestion avoidance, and segmentation and reassembly, making similar TCP/IP functionality redundant.

When close investigation is made into the IP protocol family, one finds that the time spent in TCP/IP protocol processing, for instance, has two components: one that depends only on the number of segments, and another that grows with the size of the data being transferred within a given

segment. The latter component is composed of the checksum calculation and a data copy, and the former is composed of protocol control block manipulation, header processing, timer management, and acknowledgment processing. Note that the segment component is roughly constant and is independent of the amount of data being transferred.

It seems that the most widely discussed improvements to TCP/IP's integration into ATM networking are optimizations in the header prediction, checksums, and data copying components. In general, it is also found that for large packet sizes, the TCP segment size, data copies, and checksum calculation significantly affect the overall processing time. For small packet sizes, the scheduling time and the time to do the TCP processing (other than the checksum and data copy on transmit) become significant.

The following will discuss these optimizations in detail.

8.1.1.1 Header Prediction Optimizations.

Header prediction has often been suggested as a performance benefit for TCP/IP. In one optimization case, the header prediction technique involves exploiting traffic locality to predict the next incoming packet and to avoid the protocol control block (PCB) lookup cost. The TCP input processing keeps a single entry cache of the most recently used PCB. If the incoming packet is from the same connection as the previous packet, the call to the PCB lookup routine is avoided. TCP/IP also precomputes what values it expects to find in the next incoming packet header and can then execute a faster processing path if the prediction is correct.

Other header prediction techniques involve prefilling parts of the transport header (known as optimization for lowering latency) or using traffic locality to improve throughput for bulk data transfer protocols. But unfortunately, there is only a very small improvement with header prediction. An experiment demonstrated that header prediction optimization only brings a maximum of a 10% throughput increase.

8.1.1.2 Checksum Optimization.

Checksum optimization is another popular network performance optimization method. It is important to note that the checksum does not scale linearly with the small transfer sizes because the checksum is done over the data and the TCP/IP header (20 bytes for TCP header + 20 bytes for IP overlay + length of TCP options). Therefore, as transfer sizes grow, the checksum calculation begins to dominate most other costs in sending the data, indicating that the checksum is an attractive place for optimization.

One technique combines the checksum calculation with one of the data copies. For example in TCP/IP running under the ULTRIX environment, data was copied at least twice on both send and receive. One copy moved the data between user and kernel space. The other copy moved the data between kernel and the device memory. With this technique, throughput improvement of almost 60% was achieved in an experiment. It suggests that computing the TCP/IP checksum and data copying are the major costs of the overall TCP/IP processing.

Also it has been observed that in local area networks, the TCP/IP checksum does not contribute significantly to the detection of errors. Therefore, it appears that latency can be further reduced by eliminating the checksum calculation altogether for local area traffic. It is already common practice to eliminate the UDP checksum for NFS traffic, a result of previous experiments. An experiment is found to confirm this assumption. An improvement of 56% was achieved through elimination of checksums alone.

8.1.2 Addressing

Another troubling issue in the integration of networks, especially the Internet network, is the rapid exhaustion of free addresses. The Internet is facing serious challenges. Among them, the three primary technical problems which need to be addressed are: routing table expansion beyond router capabilities, premature exhaustion of assigned IP network numbers due to inefficient allocation of the IPv4 address space, and IP's inability to address more than 4 billion hosts connected to a single public internet.

Current proposals[1] attempt to solve all of the above problems with the application and deployment of new, immature and certainly untested technology. Since the problem of routing table size may become serious in the near term (1-2 years), there is a perception that it is necessary to adopt one of the proposed new technologies as soon as possible.

8.2 Future Works

The current Video Dial Tone prototype is mainly targeted toward information access applications which require asymmetric access capability. Other feasible applications should be implemented next to fully test the VDT architecture. Among these applications are: Interactive Television, Interactive Home Shopping, and Interactive Video Game. With these interactive appli-

1. PIP [Francis 93], SIP [Deering 93], TP/IX [Ullman 93], TUBA [Callon 92][Katz 93]

cations, multiple plot ending possibilities can be built into the scripts. The input from viewers can affect what is to happen next in the program. Thus the same movie could now be viewed many times by the same audience but with various outcomes and arrangements. Other related industries, such as advertising and video games, could use similar techniques.

But the next challenge Bellcore faces is to redesign/modify the VDT architecture and network management to handle multi-user and shared applications. The first multi-user application to be implemented should be Shared Video On Demand (SVOD) since it is closely tied to Bellcore's existing VOD application. With a SVOD application, multiple users can simultaneously access, control, and view the same program while communicating with each other over wireless microphones. At the completion of the SVOD application, many other multi-user applications will follow because SVOD will have advanced both architectural and application standards.

Other pending issues also needing special attention are: yellow/white pages, merchant views, content preparation, and security/payment. In any case, the Video Dial Tone system is now ready for the expansion of the creative horizons of the video/multimedia industry.

References

[1] J. C. Brustoloni and B. N. Bershad. Simple Protocol Processing For High-bandwidth Low-latency Networking. Technical report, School of Computer Science at Carnegie Mellon University, March 1992.

[2] A. Gelman and L. S. Smoot. An Architecture For Interactive Applications. Technical report, Multimedia Communications Research Division of Bellcore, June 1993.

[3] A. Gelman and L. S. Smoot, personal communication, August 17,1993.

[4] S. E. Minzer. Broadband ISDN and Asynchronous Transfer Mode (ATM). *IEEE Communications Magazine*, volume 27, no. 9, ppg 17-24, September 1989.

[5] A. Wolman, G. Voelker, and C. A. Thekkath. Latency Analysis of TCP On An ATM Network. Technical report, Department of Computer Science at University of Washington, 1993.

[6] M.-H. Nguyen and M. Schwart. Reducing The Complexities Of Tcp For A High Speed Networking Environment. In *Proceedings of IEEE Globecom '93*, pages 1162–1169. IEEE Computer Society, 1993.

[7] C. Papadopoulos and G. M. Parulkar. Experimental Evaluation Of Sunos Ipc And Tcp/ip Protocol Implementation. In *Proceedings of IEEE Globecom '93*, pages 628–637. IEEE Computer Society, 1993.

[8] D. Sanghi, A. K. Agrawala, O. Gudmundsson, and B. N. Jain. Experimental Assessment Of End-to-end Behavior On Internet. In *Proceedings of IEEE Globecom '93*, pages 867–874. IEEE Computer Society, 1993.

[9] M. DeAddio. TCP/IP Unix Evaluation In A High-bandwidth ATM Network. Master's Thesis, Massachusetts Institute of Technology, May 1992.

[10] H. Brody. The Home Front. *Technology Review*, pages 29–40, August/September 1993.

[11] R. Brown. Pay per view: Driving The Superhighway. *Broadcasting & Cable*, pages 54–59, November 1993.

[12] D. T. Chai, G. G. Hartwick, E. Lubchenko, and J. Nachem. Networked Compact Disc Interactive. *SMPTE Journal*, pages 773–776, September 1993.

[13] S. Coe. Will There Be Demand For Video On Demand. *Broadcasting & Cable*, pages 60–61, November 1993.

[14] W. Hodge, S. Mabon, and J. T. J. Powers. Video On Demand. *SMPTE Journal*, pages 791–803, September 1993.

[15] R. Kaplan. Video On Demand. *American Demographics*, pages 38–43, June 1992.

[16] Latency Analysis Of TCP On An ATM Network. Technical report, Department of Computer Science at University of Washington, 1993.

[17] S. Scully. From Videotape To Video Servers, Technology Drives PPV. *Broadcasting & Cable*, pages 66, 85, November 1993.

[18] R. J. Bates. *Introduction To T1/T3 Networking*. Artech House, Boston, MA and London, England, 1992.

[19] U. Black. *TCP/IP And Related Protocols.* McGraw Hill, Inc., USA, 1992.

[20] D. E. Comer and D. L. Stevens. *Internetworking With TCP/IP*, Volume 1 of *Principles, Protocols, And Architecture.* Prentice Hall, Englewood Cliffs, NJ, 1988.

[21] D. E. Comer and D. L. Stevens. *Internetworking With TCP/IP*, Volume 2 Of *Design, Implementation, And Internals.* Prentice Hall, Englewood Cliffs, NJ, 1991.

[22] D. E. Comer and D. L. Stevens. *Internetworking With TCP/IP*, Volume 3 *Of Client-Server Programming And Applications.* Prentice Hall, Englewood Cliffs, NJ, 1993.

[23] R. K. Heldman. *Future Telecommunications: Information Applications, Services, & Infrastructure.* McGraw-Hill, Inc., USA, 1993.

[24] R. K. Heldman. *Information Telecommunications: Networks, Products, & Services.* McGraw-Hill, Inc., USA, 1994.

[25] P. Miller, Mark A. *Troubleshooting TCP/IP, Analyzing The Protocols Of The Internet.* M&T Books, San Mateo, CA, 1992.

[26] E. Taylor. *Integrating TCP/IP Into SNA.* Worldware Publishing, Inc., Plano, Texas, 1993.

[27] G. White. *Internetworking And Addressing.* McGraw Hill, Inc., USA, 1992.

[28] F. Wilder. *A Guide To The TCP/IP Protocol Suite.* Artech House, Boston, MA, 1993.

[29] Bellcore. *Asynchronous Transfer Mode (ATM) And ATM Adaptation Layer (AAL) Protocol Generic Requirements*, 1992.

[30] Apple Computer, Bellcore, Sun Microsystems, and Xerox. *Network Compatible ATM For Local Network Applications*, 1992.

[31] Microware. *OS-9 Assember/Linker User's Manual*, 1988.

[32] Microware. *OS-9 C Compiler User's Manual*, 1988.

[33] Microware. *OS-9 Debugger User's Manual*, 1988.

[34] Microware. *OS-9/NET Technical Reference Manual*, 1988.

[35] P. S. Dayan. *The OS-9 Guru, 1-The Facts*, 1992.

[36] Microware. *OS-9 OEM Installation Manual*, 1988.

[37] Microware. *OS-9 Technical Manual*, 1988.

[38] Microware. *Using OS-9/Net*, 1988.

[39] Microware. *Using Professional OS-9*, 1988.

[40] Microware. *Using umacs*, 1988.

[41] Motorola Semiconductor Products Inc. *VMEbus Specification Manual*, Revision C3 Edition, 1985.

[42] P. Francis. IP - Next generation. Talk on December 14, 1993 at Bellcore, December 1993.

[43] Microware. *Using OS-9/Internet*, 1993.