Analysis of ATM Call Detail Records and Recommendations for Standards

by

Xianrui Roger Wang

Thesis submitted to the Faculty of the

Virginia Polytechnic Institute and State University

in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

in

Computer Engineering

Scott F. Midkiff, Chairman

Ira Jacobs

Alex Q. Huang

June 1999

Blacksburg, Virginia

Keywords: ATM, Call Detail Records (CDRs), Network Management

Analysis of ATM Call Detail Records and Recommendations for Standards

By

Xianrui Roger Wang

Dr. Scott F. Midkiff, Committee Chair

Department of Electrical and Computer Engineering

(ABSTRACT)

Data network resource management and capacity planning are critical for network design, operation, and management. Equipment vendors often provide good information for traffic management and control and associated tools, but this information and the tools are based on independent, individual switches or routers rather than the whole network. There is a critical need for tools to monitor general resource usage in a network as a whole. In this research, we develop a toolkit to collect ATM Call Detail Records (CDRs) from two types of ATM switches from IBM and FORE Systems. Data records collected by the toolkit can then be used to assess network resource utilization and traffic characteristics with the objective of predicting future needs, making proper network management decisions, and ultimately, assisting in the ability to provide reliable quality of service (QoS) in the network. In addition, we examine current call detail records and requirements for more comprehensive network management and make recommendations for a standardized CDR.

Acknowledgments

This work has been supported by the IBM Corporation through a University Partnership Program grant. The support of Dr. Andy Rindos and Mr. Jonathan Knop of IBM in securing this grant are gratefully acknowledged.

This research was conducted as a collaborative project between the Bradley Department of Electrical and Computer Engineering and Communication Network Services (CNS). Ideas, guidance, and technical advice provided by Mr. Eric Brown and Mr. Clark Gaylord of CNS were essential to the success of this research.

Dr. Scott Midkiff had been giving inspiring ideas from the very beginning of the research through the completion of thesis writing, and preparation of defense slides. His patience and his precision will leave me with a lasting impression. Without his help, I could not have finished my thesis. I would like to give my most sincere thanks to him.

Dr. Ira Jacobs and Dr. Alex Huang took their valuable time to read my thesis and contributed valuable suggestions. I appreciate their help very much.

I would like to give special thanks for John Nichols, who was my supervisor when I worked in CNS in summer 1998. His consistent support and care for my study at Virginia Tech will be unforgettable.

I feel indebted to Phil Benchoff, Carl Harris, John Pollard and the bright and nice people in CNS. I also thank CNS for providing me the office, workstation and many up-to-date documents.

Thanks for Creighton Hager, my research-mate. I really appreciate the way in which we were able to work together. He helped testing some codes in my toolkits and he also gave valuable suggestions to my thesis.

For years, my wife, Yanhua, has been supporting and helping me silently. It is not easy to simply say “thanks” to her. I owe her a lot.

Thanks to my Christian brothers, Chuck Schumann, Johnny Yu and Xinhua Shi, for helping me with my final defense.

Give all the glory to God Almighty.

Table of Contents

List of Figures

List of Tables

Chapter 1. Introduction

1.1. Background

Asynchronous Transfer Mode (ATM) has emerged as a promising technology to support advanced broadband multimedia communication services. Accordingly, its emergence has generated new challenges for data network designers and managers. One of these challenges lies in network resource management to guarantee the reliable transfer of traffic for critical applications such as video conferencing, distance learning, and telemedicine.

NET.WORK.VIRGINIA [1] is a Virginia statewide public broadband network. It is regarded as one model for the Next Generation Internet (NGI). It supports video conferencing and traditional data networking and is based on ATM technology. Data networking is based on best-effort service within the network, while video conferencing relies on the inherent quality of service (QoS) mechanisms in the network. At present, video conferencing is shifting from using permanent virtual circuit (PVC) service to switched virtual circuit (SVC) service. The basic features of PVC and SVC are listed in Table 1-1.

Table 1-1. Features of PVC and SVC

VC Type	**Setup mode**	**Setup time**	**Duration**	**Disconnect mode**	**Speed** (Supported by NET.WORK. VIRGINIA)
PVC	Network management system	Minutes to weeks. Available in advance.	Hours to years	Human intervention or permanent network element failure	Max. 1152 Kbps
SVC	Signaling	Seconds to minutes. Maybe available before a call.	Minutes to hours	Signaling	56-1920 Kbps

Providing a reliable resource management strategy will be a challenge for network managers and developers for NET.WORK.VIRGINIA, or any network, to run SVC and provide QoS successfully.

1.2. Research Goals

There are two key goals for this research. The first is to develop a toolkit to automatically process ATM call detail records (CDRs), extract useful information and provide a uniform format for high-level analysis even though the underlying switches are different. There are existing mechanisms to monitor separate circuits in the network. These mechanisms are used to monitor the general network usage on a link-by-link basis rather than on a cell-by-cell basis. Based on this idea, we created a toolkit to automatically collect data from two different types of ATM switches from IBM and FORE Systems. FORE Systems ATM switches are used both in the Virginia Tech campus backbone and in NET.WORK.VIRGINIA. An IBM 8265 switch was also available for testing. Analysis of the collected data and the monitoring of network flows will allow the development of traffic engineering models and monitoring of the network to ensure appropriate capacity, although this higher level analysis is beyond the scope of this thesis.

The second goal of this research is to provide recommendations for ATM CDR standards, including methods to generate CDRs and standard objects that should be available within CDRs. The recommendations are based on accomplishing the first goal. It is a common goal that vendors strive to make their products have higher performance and a lower price. Almost all ATM switch vendors can provide smart congestion control and traffic management algorithms in their switches. But, at present, few vendors seem to pay attention to the overall network traffic management task. The recommendations for CDR standards are intended to give new ideas to ATM switch vendors by answering the following questions.

- What kind of features should be included in ATM switches to help the network manager optimize a whole network's resource management?
- What kind of MIB data is needed besides the standard MIB data?

- How can a vendor make ATM switches that are “superior” to competitors’ products with respect to network management and monitoring?

1.3. Thesis Organization

This thesis is divided into six chapters. Chapter 2 discusses ATM technology. ATM technology is not easy to explain in a technical paper or even in a single book. Chapter 2 covers just the basic concepts used in this thesis. Topics included in Chapter 2 are justifications for choosing ATM, ATM service classification, quality of service and QoS parameters, ATM traffic control and traffic parameters, ATM adaptation layer, ABR service parameters and resource management cells, MIBs, ATM signaling, message types and information elements. This chapter is intended to help readers understand the succeeding parts of the thesis.

Chapter 3 describes the research approach. It introduces the classification of CDRs, data collection approaches required for different switches and the related toolkit. This chapter discusses the types of CDRs of interest, collection of CDRs from switches from different vendors, and programming languages and techniques.

Chapter 4 discusses the detailed implementation of the toolkit used to collect CDRs from the FORE Systems ASX series ATM switch and the IBM ATM 8265 switch. To help readers understand the toolkit, program flow charts, technical tips and important code segments are given.

Chapter 5 presents the recommended standards for CDRs, including generation methods and objects. The researcher hopes these recommendations can improve the design of next generation ATM switches.

Chapter 6 draws conclusions about the research. The work that needs to be continued in the future is also described.

Appendix A lists most of the acronyms used in the thesis. The toolkit user manual is given in Appendix B. Appendix C includes the classification of the collected CDRs, such as identification parameters, QoS parameters, traffic parameters and statistical parameters. It provides background for Chapter 5 since the recommended standards are partly based on the analysis of the available CDR parameters.

Chapter 2. Background

To help readers' further reading, the related background knowledge is introduced in this Chapter. This Chapter includes ATM technology background such as introduction to ATM, the ATM cell structure, ATM service classifications, ATM traffic control, etc. This chapter also describes the basic knowledge for management information base (MIB) and MIBs supported by the IBM 8265 switch since the CDR collection from the IBM switch needs well understanding of them.

2.1. ATM: Integration of LAN/WAN

ATM is an integration of local area network (LAN) and wide area network (WAN) technology. The major differentiation between ATM-based services and other existing communication services, such as IP, X.25, Frame Relay, leased line, SMDS, and Ethernet, is that ATM is the first technology and protocol structure to effectively integrate voice, data, and video over the same communication channel with guaranteed quality of service (QoS). Compared with LANs, such as FDDI, Ethernet and Token Ring, ATM can achieve more reliable QoS. Compared with WANs, such as Frame Relay, X.25 and Switched Multimegabit Data Service (SMDS), ATM can provide higher access and backbone speeds. ATM hardware and software platforms form a communication architecture based on the switching and relaying of cells. The size of the cell is fixed at 53 bytes, which makes it more convenient for hardware to do most of the switching [2].

There are business drivers, application drivers, technology enablers, information and computer networking trends leading to the full deployment and development of ATM technology [2]. The basic motivating forces for ATM technology are given below.

Business Drivers:

- The move from centralized to distributed operations
- Client-server applications, the need for LAN/MAN/WAN interconneciton
- Bandwidth on demand
- Enterprise networking world flexibility and investment protection

Application Drivers:

- Distance learning, training and video conferencing
- Remote medicine and remote distributed database access
- Multimedia applications
- Various consumer applications

Technology Enablers:

- Development of hardware technology (processors, memory, electronics)
- Availability of digital transmission media with higher performance/price ratio
- Worldwide support of ATM technology

Information and Computer Networking Trends:

- Crisis of routing and need of switching
- Deployment of virtual private networks (VPNs), intranets and extranets
- Security and bandwidth requirement

2.2. ATM Cell Header Structure

ATM cells have a fixed length of 53 bytes including a 5-byte header [3] [4]. The header is different depending on whether the cell is used at the User-Network Interface (UNI) or Network-Network Interface (NNI). The UNI and NNI cell header structures are shown in Figure 2-1. The difference between them is only in byte 1. A UNI cell has a General Flow Control (GFC) field that is in the first byte (bit 8 to bit 5). An NNI cell uses all of byte 1 as the Virtual Path Identifier (VPI).

The fields in the header are defined as follows [4].

General Flow Control (GFC): A 4-bit field that supports simple multiplexing. It is always coded as all zeroes under current ATM specifications.

Virtual Path Identifier/Virtual Channel Identifier (VPI/VCI): The length of the VPI field is 8 bits in UNI cells and 12 bits in NNI cells. The VCI field is always 16 bits long. Some values are reserved for special purposes, such as signaling, and some values are available upon negotiation between the user and the network and are used for routing.

Payload Type (PT): The 3-bit PT field discriminates between a cell carrying management information or a cell that is carrying user information.

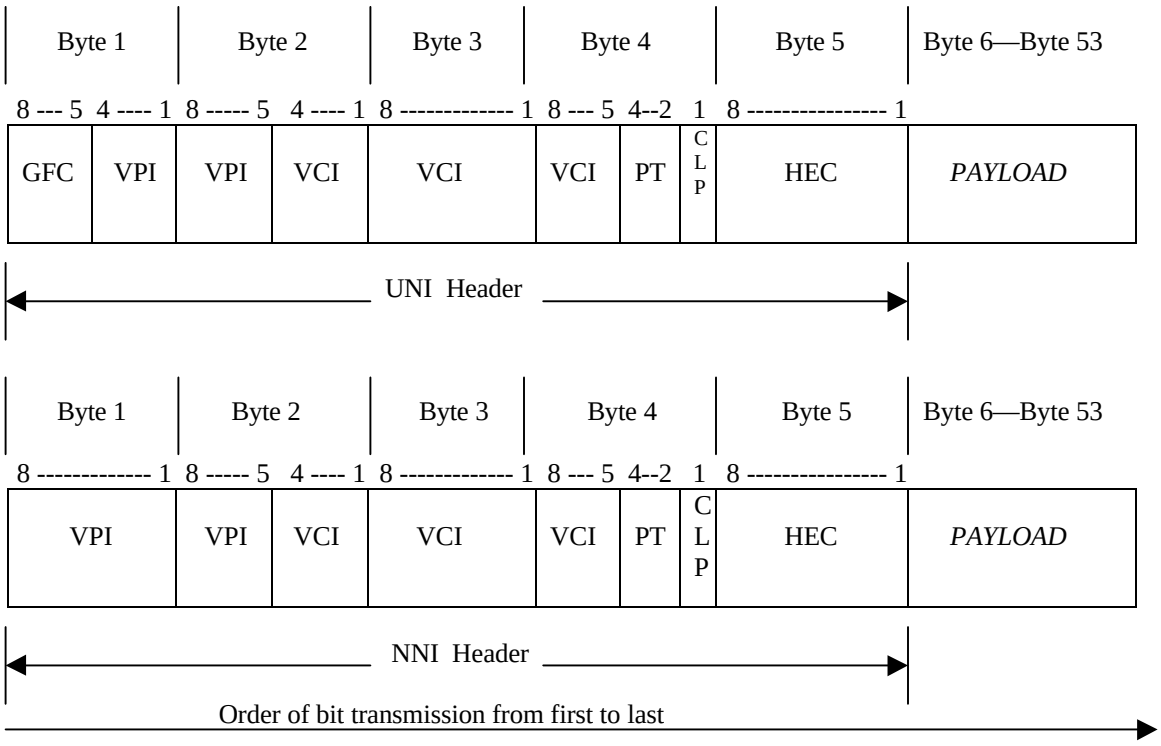

Figure 2-1. UNI and NNI cell header structure.

Cell Loss Priority (CLP): CLP is a single bit. It indicates two levels of priority for ATM cells. Cells with CLP = 0 have higher priority than cells with CLP = 1. Cells with CLP = 1 may be discarded during periods of congestion to preserve the Cell Loss Rate (CLR) of cells with CLP = 0.

Header Error Control (HEC): The one-byte HEC field is calculated using a Cyclic Redundancy Check (CRC) algorithm to correct single bit errors and to detect multiple bit errors. It may be used for cell delineation.

2.3. Classification of ATM Services

The ATM Forum [5] has specified five “service categories” in relation to traffic management in an ATM network. They are listed below.

Constant Bit Rate (CBR): CBR service provides a constant or guaranteed rate to transport services, such as video, voice, or circuit emulation, which require rigorous timing control and performance parameters.

Real-time Variable Bit Rate (rt-VBR): rt-VBR service is like CBR in the sense that it achieves low transit delay, but the traffic can vary in its data rate. The data here might be compressed video, compressed voice with silence suppression, or HDLC link emulation with idle removal.

Non-real-time Variable Bit Rate (nrt-VBR): nrt-VBR is also a guaranteed delivery service where transit delay and jitter are less important than for rt-VBR. An example traffic type is an MPEG-2 encoded video stream. In this case, the information may be being retrieved from a disk for one-way video distribution. A network transit delay of even a few seconds is not a problem here. But low cell loss rate and cell error rate are required.

Unspecified Bit Rate (UBR): The UBR service category is for "best effort" delivery of data. No QoS parameters are specified so the network provides no quality commitment. The only traffic parameter specified is peak cell rate (PCR) and all submitted traffic has the CLP bit in the ATM header set to 1. It is suitable for traditional data applications, such as LAN emulation (LANE) and IP transport.

Available Bit Rate (ABR): The ABR service category guarantees a minimum cell (bit) rate and allows a higher bit rate up to PCR if there is available capacity in the network. ABR provides a flow control mechanism to regulate the source's traffic rate according to the existing network conditions. It also attempts to keep the cell loss rate as low as possible.

2.4. Quality of Service Classes and Related Parameters

2.4.1. QoS Classes Defined by the ATM Forum

Five QoS classes are defined by the ATM Forum [3][4].

Class 0 (No equivalent ITU-T Service Class): Class 0 provides best effort service with no objectives specified for the performance parameters, i.e., it provides UBR service.

Class 1 (ITU-T Service Class A): Class 1 service is CBR service with end-to-end timing.

Class 2 (ITU-T Service Class B): Class 2 service is VBR service with end-to-end timing. It is intended for packetized video and audio in teleconferencing and multimedia applications, i.e., it provides rt-VBR service.

Class 3 (ITU-T Service Class C): Class 3 service is VBR service with no end-to-end timing required. It is intended for interoperation of connection-oriented protocols such as Frame Relay. It is equivalent to nrt-VBR and ABR service.

Class 4 (ITU-T Service Class D): Class 4 service is VBR service with no timing required. It is intended for interoperation of connectionless protocols such as SMDS and IP.

2.4.2. Quality of Service Parameters

ATM is presently the best technology to guarantee QoS and reserve bandwidth. Below are the most commonly used ATM QoS parameters [5].

Cell Transfer Delay (CTD): CTD specifies the maximum time for a cell to be transferred from its source to its destination over a virtual connection (VC). It is the sum of buffering delays, propagation delay (PD), processing and queuing delays [7].

Cell Delay Variation (CDV): CDV specifies the peak-to-peak cell delay through the network, i.e., the difference between the best case and the worst case achievable on the VC. It is an important parameter for time-sensitive traffic such as CBR and rt-VBR since it is used to specify how closely cells are spaced in a virtual channel connection (VCC). Due to multiplexing, buffering, or the insertion of operation and maintenance (OAM) cells, ATM switches are the main cause of introducing cell delay variation [7].

Cell Loss Ratio (CLR): CLR specifies the ratio of the lost cells to the total transmitted cells for a given virtual channel connection (VCC).

Cell Error Ratio (CER): CER specifies the fraction of the transmitted cells that contain bit errors when received by the destination.

Cell Misinsertion Rate (CMR): CMR specifies the ratio of cells received at an endpoint that were not originally transmitted by the source end to the total number of cells properly transmitted.

For all applications, CER and CMR must be on the order of 10^{-8} [2]. The principal QoS parameters are CTD, CDV and CLR. Human sensory perceptions and other factors determine the acceptable values of these major QoS parameters for a given application, while data communication protocol dynamics define the rest [2].

2.5. ATM Traffic Control Approaches and Related Parameters

2.5.1. ATM Traffic Control Approaches

A number of ATM traffic management functions, defined by the ITU-T [21] and the ATM Forum [5], support the standard service categories. These functions are used for monitoring and controlling traffic and congestion occurring at an individual switch or a group of switches. Some of the functions are listed below.

Connection Admission Control (CAC): Before establishing an SVC or a PVC, the ATM network uses the CAC function to determine whether to accept or reject a connection request. The CAC accepts a connection request only if sufficient resources are available at all related network elements [5].

Usage Parameter Control (UPC or Traffic Policing): The UPC function monitors and ensures that a connection complies with the traffic contract and the source traffic parameters so that the QoS provided to other connections is not jeopardized. It is performed at the UNI. To determine compliance with a traffic contract, the UPC function implements the “leaky bucket” algorithm, also called the Generic Cell Rate Algorithm GCRA [5].

Traffic Shaping: Traffic shaping is a method for smoothing bursty traffic that might arrive on a virtual circuit to present a more uniform traffic stream to the network.

Frame Discard: A network element can discard cells on a frame, i.e., AAL protocol data unit. The two frame discard mechanisms are Early Packet Discard (EPD) and Partial Packet Discard (PPD) [2][5].

Explicit Forward Congestion Indication (EFCI): EFCI is a 1-bit field in the PTI field of the ATM header of a data cell, as opposed to an operation and maintenance cell, to identify whether congestion has been experienced by the cell. A network element in a congested state may set the EFCI bit for use by other network nodes or the destination

equipment to avoid further congestion. It is used in resource management (RM) cells in ABR services [5] [21].

Buffer Management: A switch's output port buffer can provide temporary storage for the cells traveling to this port and exceeding the port's transmission capacity. Efficient buffer management can maintain the appropriate QoS for each traffic type through prioritization and intelligent buffer allocation. The buffer architectures include First-In First-Out, Strict Priority Queuing, Fair Queuing, Weighted Round-Robin Queuing, and Weighted Fair Queuing. A detailed description is provided in [8].

Route Management: ATM switches maintain a detailed and frequently updated topology of the network, including information such as cost, bandwidth, congestion, and connection status. Neighboring ATM switches communicate with each other using the PNNI protocol. After receiving a call request, the routing algorithm strives to supply the optimal end-to-end route through the network [2][21].

2.5.2. ATM Traffic Parameters

The ATM traffic contract is an agreement between a user and a network. The network provides a specified QoS, as mentioned in the previous section, if and only if the user's cell flow conforms to a negotiated set of traffic parameters. The ATM Forum [5] defined the following traffic parameters.

Peak Cell Rate (PCR): PCR, expressed in cells per second, characterizes the maximum source transmission rate. Accordingly, 1/PCR is the minimum inter-cell interval time for a given virtual circuit. The UPC function is responsible for policing the PCR of each virtual circuit over the access circuit. Any cell that does not conform to PCR is discarded.

Cell Delay Variation Tolerance (CDVT): CDVT, expressed in seconds, is used for CBR traffic to specify the acceptable tolerance to cell-by-cell variations of the CDV.

Sustainable Cell Rate (SCR): SCR, expressed in cells per second, characterizes the bursty source and specifies the maximum average rate at which cells can be sent over a given virtual circuit. UPC is responsible for policing this parameter for rt-VBR and nrt-VBR virtual circuits.

Maximum Burst Size (MBS): MBS specifies the maximum number of cells that can be transmitted at the PCR, assuming the receiving buffers are empty at the beginning of the burst.

Minimum Cell Rate (MCR): MCR, expressed in cells per second, is an ABR traffic parameter that defines the minimum transmission rate to which the network will constrain a source in the event of network congestion. RM cells contains this field.

2.6. ATM Adaptation Layer (AAL)

The ATM Forum [4] and ITU-T Recommendation [22] define ATM in terms of three layers: the physical layer, the ATM layer and the AAL layer. The AAL layer provides support for higher layer services such as signaling and data, voice, and video applications. It divides messages or bit streams from higher layers into ATM cells at the transmitter and reconstructs them at the receiver. The AAL is sub-divided into the following classes.

AAL 1: AAL 1 is used for CBR and connection-oriented traffic, e.g., 64 Kbps voice and fixed-rate video, or TDM-based circuit traffic. It does not support any multiplexing functions. Timing information is required to be exchanged between the source and the destination. It supports QoS Class 1 (ITU-T Class A).

AAL 2: AAL 2 is used for rt-VBR and connection-oriented traffic to be delivered with fixed delay, e.g., packetized voice or video. It supports the multiplexing of multiple traffic streams into one virtual connection. Timing information is required to be exchanged between the source and the destination. It supports QoS Class 2 (ITU-T Class B).

AAL 3/4: AAL 3/4 is a combination of AAL3 and AAL4. Originally AAL3 and AAL4 supported encapsulation of delay-tolerant, bursty, connectionless and connection-oriented traffic. AAL3 supported QoS Class 3 and AAL4 supported QoS Class 4. AAL 3/4 also performs re-sequencing and cell identification operations in support of multiplexing traffic streams into one virtual connection. Generally, AAL 3/4 supports QoS Classes 3 and 4 (ITU-T Class C, D) and QoS Class 0.

AAL 5, or Simple and Efficient Adaptation Layer (SEAL): AAL 5 is similar to class 3/4, but without multiplexing or error detection. It has been adapted by the computer

networking industry in response to the complexity and implementation difficulties in AAL 3/4. No timing relationship is required between source and destination. AAL 5 supports QoS Class 3 (ITU-T Class C) and QoS Class 0. In the data communication area, it has become the predominant AAL class.

2.7. Service Categories, Traffic Parameters, and QoS Parameters

In the previous sections, we described the ATM service categories, traffic parameters and QoS parameters independently. Some parameters are related to some particular categories. Table 2-1 lists the relations among them [5].

Table 2-1. ATM Service Category Attributes

Attribute	**ATM Layer Service Category**				
	CBR	Rt-VBR	Nrt-VBR	UBR	ABR
Traffic Parameters:					
PCR, CDVT	Specified			Specified	Specified
SCR, MBS, CDVT	N/A	Specified		N/A	
MCR	N/A			N/A	Specified
QoS Parameters:					
Peak-to-peak CDV	Specified		Unspecified		
MaxCTD	Specified		Unspecified		
CLR	Specified			Unspecified	[note]
Other Attributes:					
Feedback	Unspecified				Specified

Notes: CLR is low for sources that adjust cell flow in response to control information. Whether CLR is specified is network specific.

2.8. ABR Service Parameters and Resource Management Cells

ABR service is introduced in ATM Forum UNI specification 4.0 [3]. Compared with other service categories, ABR's flow control scheme and its use of resource management cells are unique. There are some parameters specifically defined for ABR service category. We discuss ABR service in this section.

2.8.1. ABR Service Parameters

The following parameters are used to implement ABR flow control on a per-connection basis. Note that except for the PCR parameter that we introduced in Section 2.5.2, the parameters are specific to ABR service.

Peak Cell Rate (PCR): PCR is the cell rate that the source may never exceed. It is expressed as cells per second.

Minimum Cell Rate (MCR): MCR is the rate at which the source is always allowed to send. It is expressed as cells per second.

Initial Cell Rate (ICR): ICR is the rate at which a source should send initially and after an idle period. It is expressed as cells per second.

Rate Increase Factor (RIF): RIF controls the amount by which the cell transmission rate may increase upon receipt of an RM-cell. RIF is given as an inverse of a power of 2, ranging from 1/32,678 to 1.

Nrm: Nrm is the maximum number of cells a source may send for each forward RM-cell. Nrm is expressed as a power of 2, ranging from 2 to 256.

Mrm: Mrm controls the allocation of bandwidth between forward RM-cells, backward RM cells and data cells. Mrm is a constant and is fixed at 2.

Rate Decrease Factor (RDF): RDF controls the decrease in the cell transmission rate. It is a power of 2, ranging from 1/32,678 to 1.

Allowed Cell Rate (ACR): ACR is the current rate at which a source is allowed to send. It is expressed as cells per second.

Missing RM-cell Count (CRM): CRM defines an upper bound on the number of forward RM-cells to be sent in the absence of received backward RM-cells. The value is an implementation specific integer.

ACR Decrease Time Factor (ADTF): ADTF is the time permitted between sending RM-cells before the rate is decreased to ICR. It is expressed in seconds, ranging from .01 to 10.23 with a granularity of 10 ms.

Trm: Trm is the limit of the maximum time allowed between RM-cells forwarded by a source. It is expressed in milliseconds, with a range from 100×2^{-7} to 100×2^{0}.

Fixed Round-Trip Time (FRTT): FRTT is the sum of the fixed and propagation delays from the source to a destination and back. The value ranges from 0 to 16.7 seconds.

TBE: The Transient Buffer Exposure is the maximum number of cells a source can initially transmit, prior to the first RM-cell returning. It is expressed as cells, ranging from 0 to 16,777,215.

CDF: The Cutoff Decrease Factor controls the decrease in ACR associated with CRM. It has value 0 or is the inverse of a power of 2 from 1/64 to 1.)

TCR: The Tagged Cell Rate limits the rate at which a source can transmit extra forward RM cells. TCR is a constant fixed at 10 cells/second.

2.8.2 RM Cell Structure

Table 2-2 shows the structure of RM-cell, including each field, position and necessary description [3].

2.9. Introduction to Management Information Base (MIB)

The MIB is the database of information maintained by the agent that a manager can query or set. The agent is the software in the network element that runs the management software. The manager here refers to the network management system. Objects in the MIB are defined using a subset of Abstract Syntax Notation One (ASN.1) defined in the Structure of Management Information (SMI), which is a collection of documents [19]. An object identifier is a sequence of integers separated by decimal points specifying an object defined by some organization. These integers traverse a tree structure similar to a Unix file system. The object identifiers start at an unnamed root that is at the top of the tree. IBM, as a network product vendor, starts at the “enterprise” object ID [20]. A simple object identifier tree is shown in Figure 2-2.

Table 2-2. RM Cell Structure

Field		Octet	Bit(s)	Description
Header		1-5	All	ATM Header. PTI = 110. If the RM-cell is in-rate, then the CLP = 0; else CLP = 1.
ID		6	All	Protocol ID. Identifies the service using the RM-cell. ID = 1 for ABR service.
MESSGE TYPE	DIR	7	8	Direction. DIR = 0 for source generated RM-cells. DIR = 1 for switch or destination generated RM-cells.
	BN	7	7	BECN (Backward Explicit Congestion Notification) cell. This bit indicates whether the RM-cell is a BECN cell or not. BN = 0 indicates a source generated RM-cell and vice versa.
	CI	7	6	Congestion Indication. It is set by the destination or intermediate switches to indicate congestion in the network. The source will decrease its ACR (Allowed Cell Rate) when receiving a RM-cell with CI = 1.
	NI	7	5	No Increase. It is set to 1 by the destination or intermediate switches to inform the source not to increase ACR. The default NI = 0 will allow the source to increase its ACR.
	RA	7	4	Request/Acknowledge. Not used yet.
	Reserved	7	3-1	
ER		8-9	All	Explicit (Cell) Rate. It specifies the maximum value of ACR for the source. It is initialized as PCR and can be marked down by the switches depending on the network condition.
CCR		10-11	All	Current Cell Rate. Set by the source to its current ACR. Its value is between MCR and PCR. For BECN cells, CCR = 0.
MCR		12-13	All	Minimum Cell Rate. It defines the minimum transmission rate to which the source is constrained. For BECN cells, MCR = 0.
QL		14-17	All	Queue Length. Not used.
SN		18-21	All	Sequence Number. Not used.
Reserved		22-51	All	
Reserved		52	8-3	
CRC-10		52	2-1	Cyclic Redundancy Check. The CRC-10 generating polynomial is: $1 + x + x(4) + x(5) + x(9) + x(10)$
		53	All	

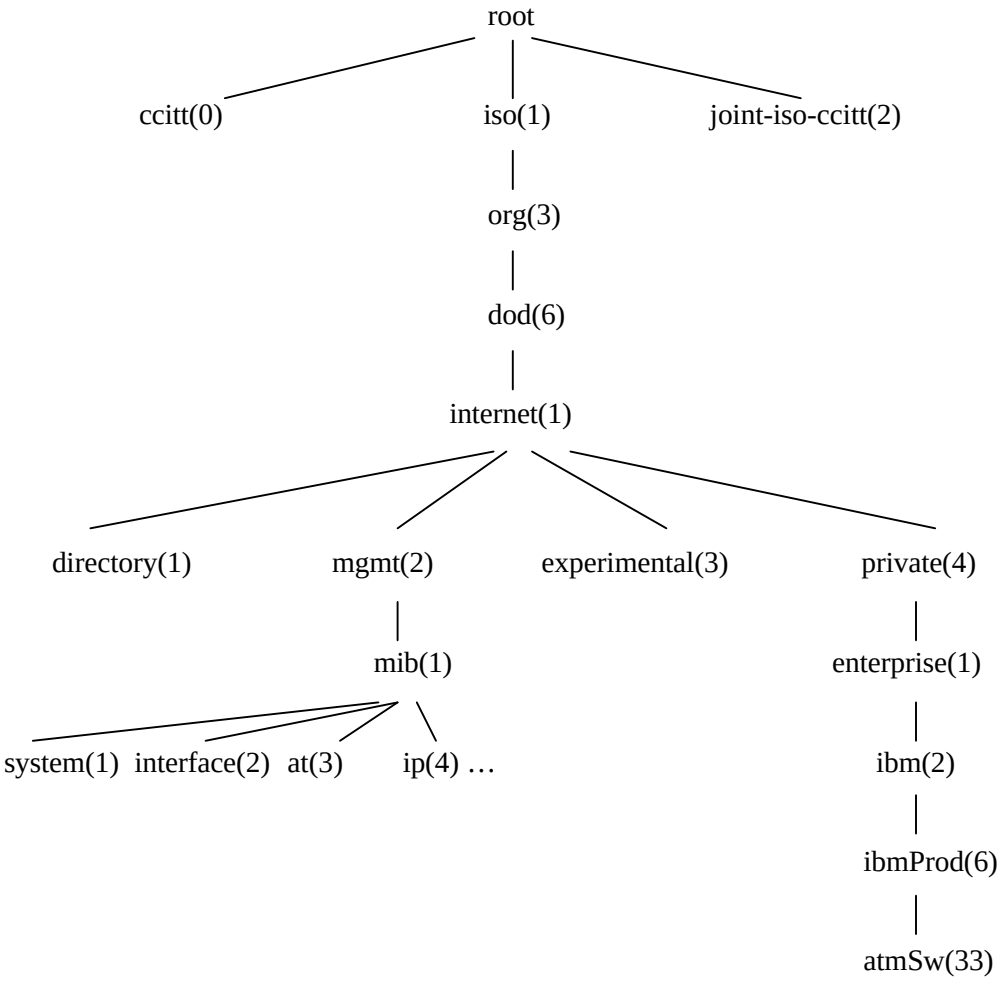

Figure 2-2. Object identifiers in the MIB.

From the simple structure above, we know every object's ID in an IBM 8265 switch's private MIB begins with 1.3.6.1.4.1.2.6.33.

2.9.1. Public MIBs Supported by the IBM 8265 Switch

The IBM 8265 switch supports the following SNMP-based public MIBs [14].

- MIB-II Version 1.1 and 1.2 (RFC 1213) and MIB-II Evolution (RFC 1573). This MIB describes managed objects used for managing network interfaces.
- IETF AToMIB (RFC 1695). This MIB describes objects used for managing ATM-based interfaces, devices, networks and services. It includes the following groups.

(1) The ATM Interface Configuration Group. It contains information on ATM cell layer configuration of local ATM interfaces on an ATM device in addition to the information on such interfaces contained in the interface table (ifTable).

(2) The ATM Interface DS3 PLCP Group. It contains the DS3 physical layer convergence protocol (PLCP) configuration and state parameters of those ATM interfaces that use DS3 PLCP for carrying ATM cells over DS3.

(3) The ATM Interface TC Sublayer Group. It contains transmission convergence (TC) sublayer configuration and state parameters of those ATM interfaces which use the TC sublayer for carrying ATM cells over SONET or DS3.

(4) The ATM Virtual Link and Cross-Connect Groups. They model bi-directional ATM virtual links and ATM cross-connects. They are used to create, delete or modify ATM virtual links in an ATM host, ATM switch and ATM network.

(5) AAL5 Connection Performance Statistics Group. It is an AAL5 connection table used to provide AAL5 performance information for each AAL5 virtual connection that is terminated at the AAL5 entity contained within an ATM switch or host.

- ATM Supplemental MIB (draft-ietf-atommib-atm2). This MIB is a draft from the Internet Engineering Task Force (IETF) that describes objects used for managing ATM-based interfaces, devices, and services in addition to those defined in some other ATM-MIB documents. It expired on September 18, 1998.
- PNNI MIB and PNNI MIB Extension. The PNNI MIB contains necessary PNNI routing information parameters. The PNNI MIB Extension allows the creation, deletion and management of soft-PVCs. (These two MIBs can be downloaded from ftp://ftp.atmforum.com/pub/approved-specs.)
- Interim Local Management Interface (ILMI) MIB. This MIB includes following groups. (It can also be downloaded from ftp://ftp.atmforum.com/pub/approved-specs.)

  (1) The System Group. It gives information on the system where interfaces run ILMI.

  (2) The Physical Port Group. It gives physical layer information on a particular port such as the status, transmission types and cable type.

(3) The ATM Layer Group. It tells the maximum number of supported VPs and VCs on the UNI, the number of VPs and VCs configured on the UNI and the number of active VP and VC on the interface.

(4) The ATM Statistics Group. It indicates the number of cells received, dropped and transmitted on the UNI.

(5) The VP and VC Groups. They provide information on virtual paths and virtual channels, such as status, traffic shape parameters, policing parameters and QoS parameters.

(6) The Network Prefix and Address Groups. They include information about the network prefix in use and the ATM address in use on the user side of the UNI and its validity.

(7) The Service Registry Group. It provides a general service registry for locating ATM network services.

2.9.2. Private MIBs Supported by 8265 Switch

The IBM 8265 switch supports following private MIBs [14].

- IBM Hub-Specific MIB Extensions. This MIB has the following groups.

(1) The Traps Control Group. It allows the user to configure traps to be sent.

(2) The Physical Group. This group can be subdivided into module table, port table, interface table and global throughput statistics.

(2.1) The ATM Modules Table gives details about the maximum number of supported VPs and VCs per module and the number of VPs and VCs in use. It also includes the type of module, the number of ports on a module, cable type, status and supported interfaces, e.g., private UNI/private NNI/public UNI.

(2.2) The ATM Ports Table gives information such as the number of ports on a module, cable type, status and supported interfaces, e.g., private UNI/private NNI/public UNI.

(2.3) The ATM Interfaces Table gives information about MIB-II interface index and the physical slot and port numbers.

(2.4) The Global Throughput Statistics can be turned on to monitor the total number of cells transmitted through the switch. If the monitoring is on, the system performance will be negatively influenced.

(3) The Cross Connect Group. It contains the cross-connections set-up in the switch for all existing virtual channel link (VCL)-based PVCs and SVCs.

(4) The Neighbor Devices Group contains basic characteristics of adjacent ATM devices attached to this switch.

Some other groups are also supported but are not particularly relevant to this research, so they are not described here. Details are in reference [14].

- IBM Signaling Extensions MIB. This MIB extension defines ATM signaling support on the device. The following information can be accessed from the MIB.

  (1) Number of supported signaling channels.

  (2) Range of reserved VPs and VCs.

  (3) VPI/VCI values of the signaling channel on a port.

  (4) The Q.2931 protocol information for each signaling link defined per port.

  (5) Details of failed calls for the ATM interface involved, such as the called party and calling party, call creation time and call log (or collection) time, failed cause, QoS, bandwidth, service category and clear cause location.

  (6) Details of in-progress calls, such as signaling VPI/VCI of the call, the call reference, the called party and calling party, call creation time and VPI/VCI value of the SVC call.

  (7) Details of PVC calls, including both entry and exit point of the PVC call.

  Some other information is also included in this MIB but will not be discussed here as it is not relevant for this work.

2.10. ATM Signaling Procedure, Message Types and Information Elements

2.10.1. ATM Signaling Procedure for Point-to-Point Connection

Our discussions focus on point-to-point SVC connections rather than point-to-multipoint connections. The signaling procedure for setting up a point-to-point SVC

connection is shown in Figure 2-3. The signaling procedure for disconnecting a point-to-point connection is similar.

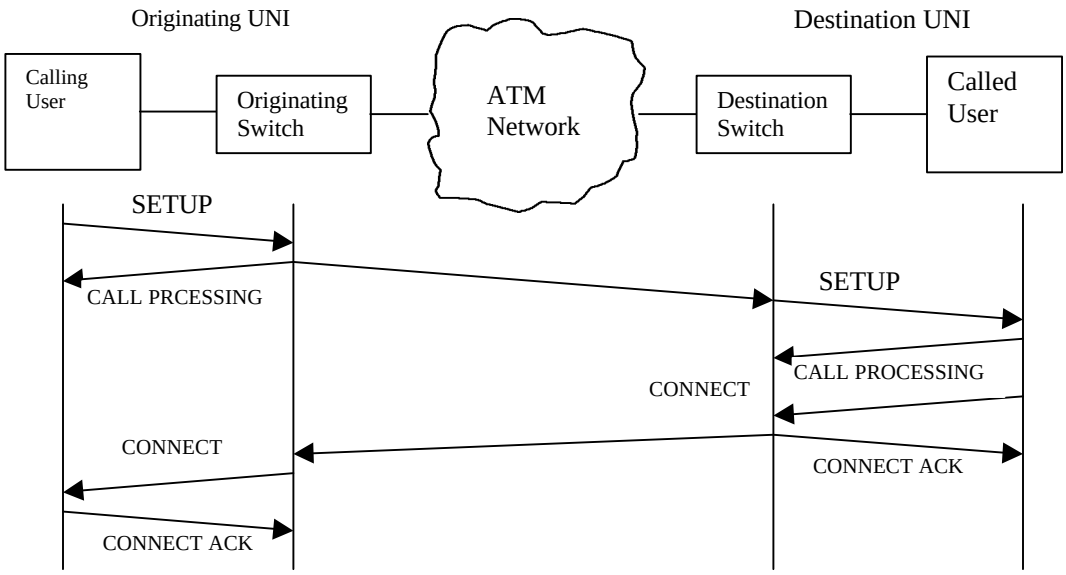

Figure 2-3. Signaling Procedure for a Point-to-Point Connection.

2.10.2. Messages for ATM Call and Connection Control

Table 2-3 lists signaling messages for point-to-point connection control defined by the ATM Forum's UNI 4.0 specification [3].

Table 2-3. UNI 4.0 Signaling Message Types for Point-to-Point Connection

Call Establishment Messages	Call Clearing Messages	Status Messages	Signaling Link Management
ALERTING	RELEASE	STATUS ENQUIRY	RESTART
CALL PROCEEDING	RELEASE COMPLETE	STATUS RESPONSE	RESTART ACKNOWLEDGE
CONNECT		NOTIFY	
CONNECT ACKNOWLEDGE			
SETUP			

2.10.3. Information Elements Defined by ATM Forum

UNI 4.0 defines 39 information elements (IEs) used for various messages. All defined IEs can be subdivided into the following groups: connection identification IEs,

ABR IEs (specified by UNI 4.0), AAL IEs, traffic parameter IEs, and QoS IEs. The most important IEs for our purposes are discussed in this section.

The messages specified by the ATM Forum are based on the ITU Q.2931 specification. Each message has a general format as shown in Figure 2-4.

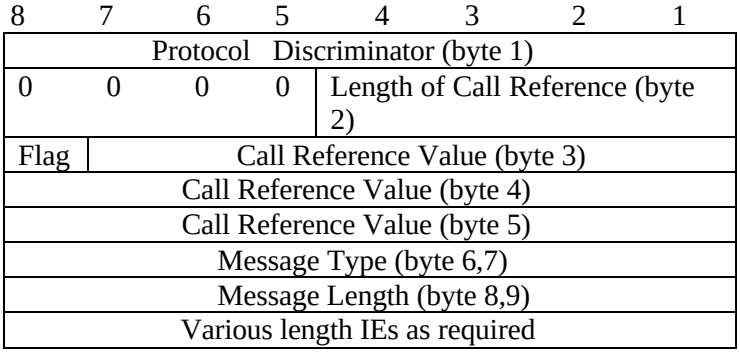

Figure 2-4. General Message Format.

The four IEs, protocol discriminator, call reference, message type, and message length, are mandatory for every message type. The protocol discriminator (1 byte) is the first part of every message. It is used to distinguish Q.2931 messages from other messages. The call reference field (4 bytes) identifies the call to which this message is related because one user may have multiple simultaneous calls. The message type (2 bytes) identifies various message types such as those shown in Table 2-3. The message length (2 bytes) identifies the length of the whole message.

Detailed descriptions of ATM messages and information elements are available in the ATM Forum's UNI 3.1 [4] and UNI 4.0 [3] specifications.

2.11. Summary

This chapter provided basic background knowledge of ATM needed to follow the remainder of this thesis. All ATM technology related discussions are rooted from ATM Forum UNI 4.0 [3] and ATM Forum Traffic Management Specification V4.0 [5]. The discussions on Management Information Bases (MIBs) are based on [14], [23], [24].

For further information on ATM technology, refer to [2], [3], [4] and [5].

Chapter 3. Approach

Through this research, we hope to provide some fresh idea about dynamic network resource management and to optimize large network design. Statistical information about successful connections should be helpful. The interesting information includes:

- The service categories that the network supports
- The bandwidth, QoS and traffic parameters to describe each successful connection
- The duration of each individual successful connection
- The number of cells which have been transferred or received for a successful connection

As a statewide public network, NET.WORK.VIRGINIA provides us with a good research and test bed. We need to create a toolkit to collect call records that can include interesting objects. High level statistical analysis can be exercised on the available data.

The toolkit has components that are specific to the particular ATM switches of interest. We initially collected CDRs from FORE Systems ATM switches in the Virginia Tech campus backbone switches. The architecture of Virginia Tech's campus backbone network is similar to that of NET.WORK.VIRGINIA, which is the ultimate specific target for this work. Most of the backbone switches are FORE Systems ASX series switches. The campus network carries traffic for a variety of applications including local and Internet data, videoconferencing for distance learning, and voice. NET.WORK.VIRGINIA carries similar information, although relative utilization may differ. After exercising the first toolkit developed for FORE Systems ASX switches, we developed a second toolkit to collect CDRs data from IBM 8265 ATM switches which are available in a laboratory environment within CNS. The toolkits are to provide a uniform CDR output to make the switch type as transparent as possible to high level analysis applications. The design of the toolkits for the two different switches and the associated collection of CDR data provided experience that can be, we hope, applied to other network elements, such as routers, switches, and hubs, from other vendors. After understanding ATM CDRs from different vendors, specifically IBM and FORE Systems, we can provide recommendations for standardizing CDRs for future ATM switch products.

In this chapter, our classification of CDRs is introduced. Then, the implementation of the toolkits is discussed. Since the toolkit supporting the IBM 8265 switch is created based on Management Information Base (MIB) files, a basic introduction to MIB and MIBs supported by the IBM 8265 switch was discussed in Section 2.9.

3.1. Classification of Call Detail Records

A CDR is created when a virtual circuit is established. We have interest in successful calls since only successful calls will occupy significant network resources. At present, we have interest primarily in successful SVC calls. We classified potentially successful calls (which might fail later) as type A and type B. We classified known successful calls as type C and type D. These call types are illustrated in Figure 3-1.

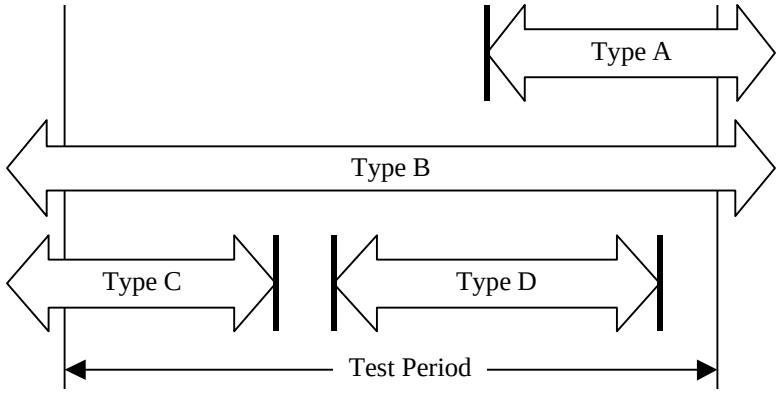

Figure 3-1. Classification of CDRs.

We record calls of type C and D since they are successful, but type A and B calls add some complexity. We ignore type A calls in our analysis. In equilibrium, type C calls balance type A calls, i.e. the number of type A calls created within the test period that terminate outside of the period should be statistically equivalent to the number of type C calls created outside the period but that terminate within the period. Type B calls are less easily dealt with. It is not feasible to collect data for type B PVC calls since the duration of such calls may be infinite. If type B calls are SVC calls, we can ignore them as long as our test period is long enough. Few SVC calls observed to date from the Virginia Tech campus backbone lasted more than 20 hours. However, we can reexamine or reprogram

the duration of test period in the future as we broaden the scope of our data collection. The test period in Figure 3-1 is explained further in Section 3.3.

3.2. Creation of Toolkit

Source files in the first toolkit that supports FORE Systems ASX switches are written in Perl [10], Unix Shell [11], and C languages. Source files for the second toolkit that supports the IBM 8265 switch are written in Tool Command Language (TCL) as part of the Scotty system [15], Perl, and Unix Shell script languages.

The following sections explain why multiple toolkits are required and why multiple programming languages were used.

3.3. Data Collection from FORE ASX Switches

The FORE ASX switch generates the raw call record data file in a 5-minute interval. (Note that five minutes is the default interval value and this value can be adjusted [13].) The raw CDRs from FORE switches are temporarily stored in the switch's random access memory (RAM) and automatically transferred to an external Unix file server in real-time using the File Transfer Protocol (FTP). In the first toolkit, we chose the test, or data collection, period shown in Figure 3-1 to be 24 hours. There are several reasons for this. The first reason is that since a large amount of data is stored in the data server and the size of the data server's hard disk is limited, data more than 54 hours old is automatically purged by the data server. The second reason is that we consider that some SVC calls may be long, but the duration is usually less than 20 hours. So few type B calls (see Figure 3-1) will be seen. The third reason is that if the test period is too long, then the size of final CDR output file would be large and difficult to process and, if the test period is too short, more final CDR files would be generated. Assuming the collection period is set as 12 hours, for example, two final CDR files will be generated per day. Either case will create complexities for the high-level analysis application. With one day (24 hours) as a test period and based on a five-minute collection interval, there are 288 (12×24) raw data files that need to be processed and only one final CDR file will be generated per day.

The first toolkit includes following steps.

Step 1: Collect raw CDRs and generate one successful CDR file.

As part of its private network management system, FORE Systems provides a CDR file post-processing utility, *fvcrb2a* [12], which can convert a file of CDRs from machine recognizable binary format to an ASCII format which is recognizable by human beings. The toolkit calls this utility to generate the raw ASCII CDR files. Then it extracts successful CDRs from the available raw ASCII data files by using the Unix *grep* command. After this, by calling the Unix *cat* command, the toolkit merges all the new generated data files into one integrated file for further processing. The executable file *CollCate* written in Unix C-shell script implements these functions. The detailed function of *CollCate* is introduced in Section 4.1.1.

Step 2: Re-format the CDR file generated in step 1.

The newly generated data file includes records with fields separated by commas. For high level processing, it is more convenient to separate each item with a TAB character. The Perl script utility *SubsComma* furnishes this function. To run *SubsComma*, the user needs to specify the following information.

- The input file name, which was generated from step 1.
- The string needing to be changed, which, for this specific application, is the character ",".
- The name of the output file.

Note that the format of the output file name is strictly defined. If the user types the improper output file name, the program can run but the user must re-name this file with the application specified name. The user needs to read the prompts carefully when he or she is not familiar with the file naming method in the toolkit.

Step 3: Calculate the call duration and generate the final CDR file

The executable Unix C-shell script *FinalGenerate* finishes the final step, as its name implies. This script first calls an executable file *TimeCalculate* to calculate the call

duration of every call. Then it adds the "Call Duration" field to the CDR file. Finally, it adds a standard header line to the file and removes the interim files to save disk space.

The calculation of call duration is important for high level analysis. The original CDRs only provide the call start time and call collect (terminate) time using the ISO 8601 time format [16]. *TimeCalculate*, written in C, is used to calculate the call duration of every call. The C language was chosen because a C library provides a header file, *time.h*, which includes a useful data structure "*tm*" to record various time formats and the C library includes a special function *mktime* that can convert the local time to a calendar value by using "*tm*" as its input variable. By calling these library functions and using the data structure, the code *TimeCalculate* can easily modify values in the ISO 8601 time format to use the Unix epoch format. After format conversion, the call duration (time difference) is easy to calculate. Other script languages, like Perl, could also have been used.

The name of the final file has the format *$SubDirectory$final$YEAR$DATE*. The *SubDirectory* and files naming convention is discussed in Chapter 4. The flowchart of the first toolkit is illustrated in Figure 3-2.

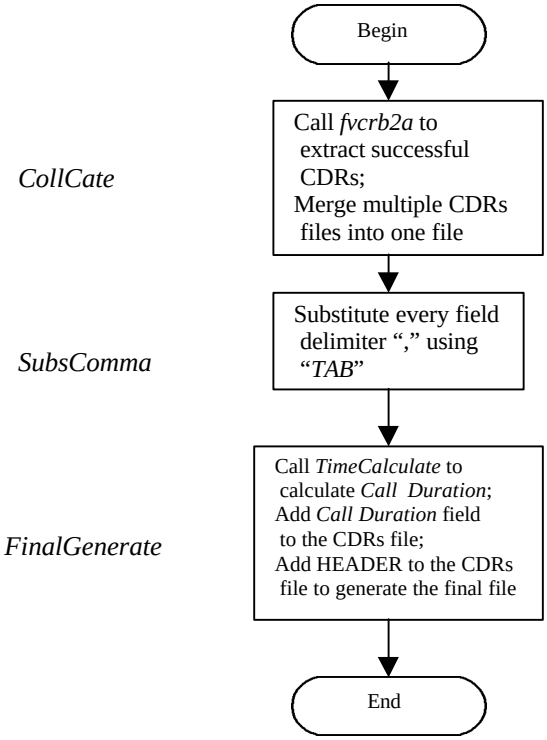

Figure 3-2. Flowchart of Toolkit 1.

3.4. Data Collection from IBM 8265 Switches

Data collection from IBM 8265 switches is done quite differently. FORE ASX series switches transfer their CDRs to the network management station (NMS) or data server in real time. The IBM 8265 switch holds a history log that contains up to 100 CDRs entries, but it does not provide schemes to automatically transfer CDRs file to some external file server. The history log can be sent to the NMS when a predefined threshold is reached or when it is requested by the NMS [14]. We have to use a special language to talk with the IBM 8265 switch directly based on the analysis of supported MIBs, including both public and private MIBs.

We first need to understand the MIB file, including syntax and the semantics of various objects, which have been discussed in Chapter 2. Then, we need to select a proper language to "talk" with the MIBs supported by the switch. We found that TCL/Scotty is a particularly powerful script language. Scotty is a software package that allows implementing site specific network management software using high-level and string-based application programming interfaces (APIs) [15]. It is based on TCL, but adds functions for the Simple Network Management Protocol (SNMP) that are critical for this research project. It also has a special *mib load* command that can load any MIB data file. TCL/Scotty makes it much easier to collect CDRs from any network element from any vendor as long as the corresponding MIB data file for the network element is available. The core files of the second toolkit are written using the Tcl/Scotty package. Some necessary Unix Shell and Perl script files are included to facilitate the operation.

Based on the analysis of IBM-supported MIBs, we know that only objects to describe the in-progress SVC calls and failed SVC calls are provided in the MIBs. We need to "infer" the call duration of successful SVC calls from the in-progress SVC calls. Assume that an SVC call is active at some fixed time since the call record of this particular call was first available in the in-progress SVC call list, and, after some time, for example, 6 minutes, that particular call is not shown in the in-progress SVC call list. Then, we consider this call as a successful call in the toolkit although this call may, in fact, have terminated abnormally. If this call is successful, it would be terminated at some time within the 6-minute duration though we cannot tell the exact time. Based on this, we can calculate the approximate value of the duration of this call.

The second toolkit includes multiple packages. As part of the second toolkit, several codes have been created to finish the approximate calculation of “call duration” and they are subdivided into the following steps.

Step1: Collect in-progress SVC calls automatically at some fixed time interval.

In the first toolkit, the SVC calls are output automatically in some fixed interval that is configurable by the FORE switch’s configuration command. However, we did not find such a function supported by the IBM 8265 switch. The choice of the CDR output interval is fully controlled by the toolkit. If the interval is too long, some calls may be lost permanently since the history log can contain at most 100 entries. If the interval is too short, many in-progress calls will be redundant. Assume an in-progress call appeared in the CDRs at 10:20 and it still lasts one more hour, then any CDRs collected before 11:20 will include this particular call. In the code, the connection interval is set as 5 minutes which is same as the interval for the first toolkit.

Step 2: Add the *Collection Time* object to every CDRs file.

The *Collection Time* object is not included in the group describing in-progress calls. For the first toolkit, “collection time” means the time when the call is terminated normally. However, here this object means a specific SVC call is not terminated at this time. It will be used for the calculation of call duration. In the second toolkit, the collect interval is set to 5 minutes so for any given day, the *Collection Time* object can only be 00:00, 00:05, …, 23:55.

Step 3: Merge all files collected during collection period.

The collection period has been temporarily set as 24 hours in the code. This value should be adjusted in a real production switch. This function was discussed during for the first toolkit.

Step 4: Make all records collected within the collection period unique.

Based on the approach used, many CDRs will be redundant although the “collection time” value is different for the redundant records. The toolkit needs to extract only the calls with latest “collection time”.

Step5: Calculate the call duration and standardize the output format.

In the latest file generated in step 4, assume a call's "call start time" is 10:30 and the "call collection time" is 13:55. Then, the call duration is 3 hours and 25 minutes, plus 2.5 minutes. Adding 2.5 minutes accounts for the fact that the call was terminated between 13:55 and 14:00. We cannot consider calls with "collection time" equal to 23:55 as successful call yet since these calls may last to the next day. Then we have to compare files collected in successive days. This adds complexity to the toolkit.

The flow chart of the calculation of call duration is shown in Figure 3-3.

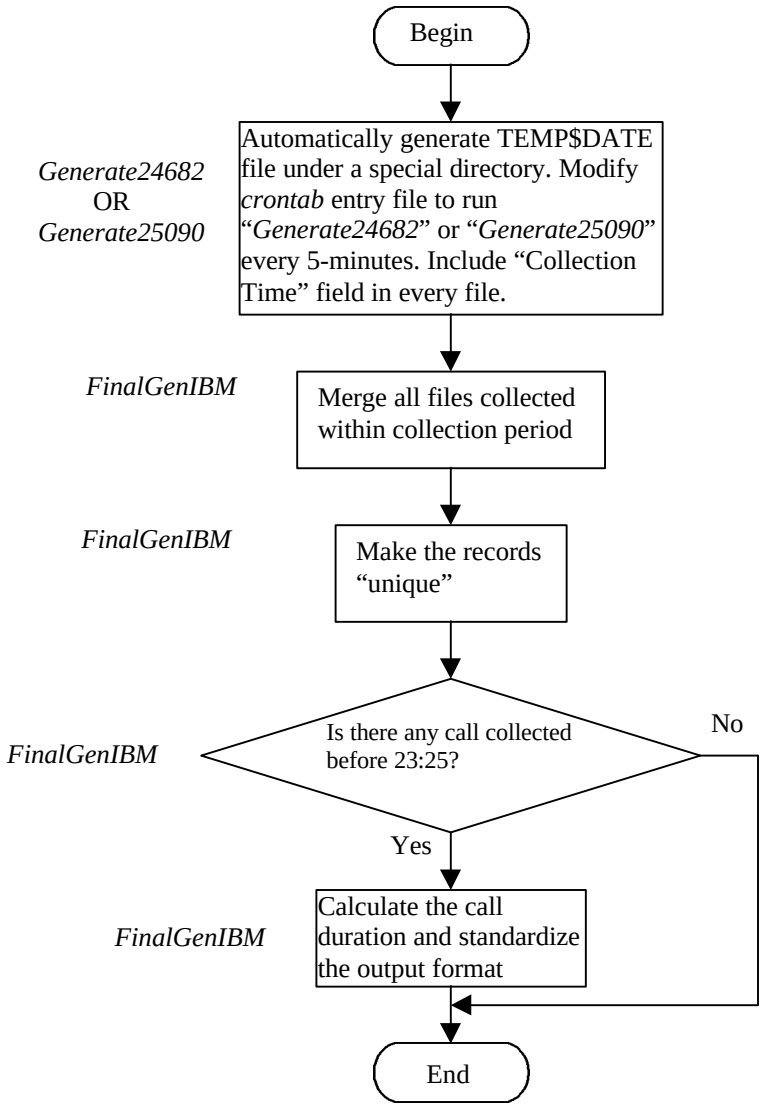

Figure 3-3. Flow chart of the calculation of approximate call duration.

The above discussion about the approach to calculate the call duration applies to IBM 8265 switches which do not provide objects describing successful SVC calls. For any vendor, including a group to describe successful SVC calls in the products' private MIB should not be a problem. Once the successful SVC calls are described, the inference of call duration will not be necessary. A successful call's call duration should be calculated based on the call start time and call termination time of this call. However, if such a group is not included in the MIBs supported by the product, the calculation of call duration will be a hurdle for the creation of a toolkit. We recommend that the switch vendor include necessary objects to describe successful SVC calls.

We exercised the second toolkit in a lab environment where there are fewer calls compared with production switches. It would be hard to select the collection interval and collection period without a comprehensive test.

As mentioned before, the second toolkit includes multiple packages. All functions included in the second toolkit are discussed further in Chapter 4.

3.5.Summary

This chapter introduced approaches to collect CDR from ATM switches. To collect CDRs from the FORE ASX switches, as used in NET.WORK.VIRGINIA and the Virginia Tech campus backbone, we need to configure the switches properly so that the CDRs are transferred to the file server automatically. To collect CDRs from IBM 8265 switches, we must first analyze MIB data files supported by IBM and then write code using a script language to collect the interesting CDRs. The call duration of successful 8265 SVC calls is estimated, so only approximate values are calculated. Chapter 4 provides detailed descriptions of the toolkits.

Chapter 4. Description of the Toolkits

Chapter 3 discussed the general approaches for data collection from different ATM switches. One toolkit is used to collect CDRs from FORE switches while the second toolkit is used to collect CDRs from IBM 8265 switches. This chapter discusses the detailed implementation of the toolkits and includes flow charts and discusses problems encountered during programming and testing, related algorithms, and technical tips. The thesis attempts to provide necessary descriptions to help users exercise the toolkits effectively. However, it is the user's duty to read prompt messages carefully while running the program since the prompt messages provide important information.

4.1. CDR Collection from FORE Systems ASX Switches

The first toolkit is used to collect CDRs from FORE Systems ASX switches, such as those used in NET.WORK.VIRGINIA and the Virginia Tech campus backbone. The toolkit includes three programs: *CollCate* (the B-shell version is *CollCateBshell)*, *SubsComma*, and *FinalGenerate*. *CollCate* collects successful CDRs and generates one integrated CDR file for further processing. *SubsComma* replaces comma delimiters with the TAB character. *FinalGenerate* calls the executable *TimeCalculate* function and generates the final CDR file for processing by high-level analysis applications.

4.1.1. Generation of successful CDR files and concatenation: *CollCate*

CollCate performs the functions of data collection and file concatenation. We introduce data collection code and file concatenation code separately.

4.1.1.1. Generation of the Interim Successful CDRs: *collect*

The data collection function is written in Unix C-shell script. It can collect successful CDRs from any FORE Systems ASX switch assuming the program can access a file that is written by the switch and has code required name format. Now this program is

configured to operate with switches in the Virginia Tech campus backbone in the following locations.

- WHI (*Whittemore Hall)* - a FORE ASX-200BX switch
- BUR (*Burrus Hall)* - a FORE ASX-1000 switch
- CAS (*Cassell Colliseum)* - a FORE ASX-1000 switch
- ISB (*Information System Building)* - a FORE ASX-1000 switch and an ASX-200BX switch
- OWE (*Owens Hall),* a FORE ASX-200BX switch
- SHA (*Shanks Hall),* a FORE ASX-200BX switch

Most of above switches are campus network backbone switches.

This code plays an important role in the whole toolkit. The flow chart is shown in Figure 4-1.

As illustrated in Figure 4-1, the user has to input the starting date and time when the CDRs were generated. The program has a special format requirement for the date (YEAR and DATE) and time (HOUR and MINUTE) input. The format for YEAR is YYYY (e.g., 1999 or 2000). The format for DATE is MMDD (e.g,, 0403 or 1126). The format for HOUR is HH (e.g., 00, 01, 02, … , 23). The format for MINUTE is MM (e.g., 00, 05, 10, … , 55). Note the MINUTE has to be a multiple of 5 assuming the default collection interval is set to be 5 minutes, as explained in Section 3.2.1. This code should typically be run once a day to collect the previous day's data. As an example, assume a user is running the code on April 14, 1999. On this day, the user needs to collect CDR data from 00:00 April 13, 1999, to 23:55 April 14, 1999, since the test period we suggest is 24 hours, as explained in Section 3.2.1. The following values should be input by the user: YEAR = 1999, DATE = 0413 (yesterday), HOUR (the start hour) = 00, MINUTE (the start minute) = 0.

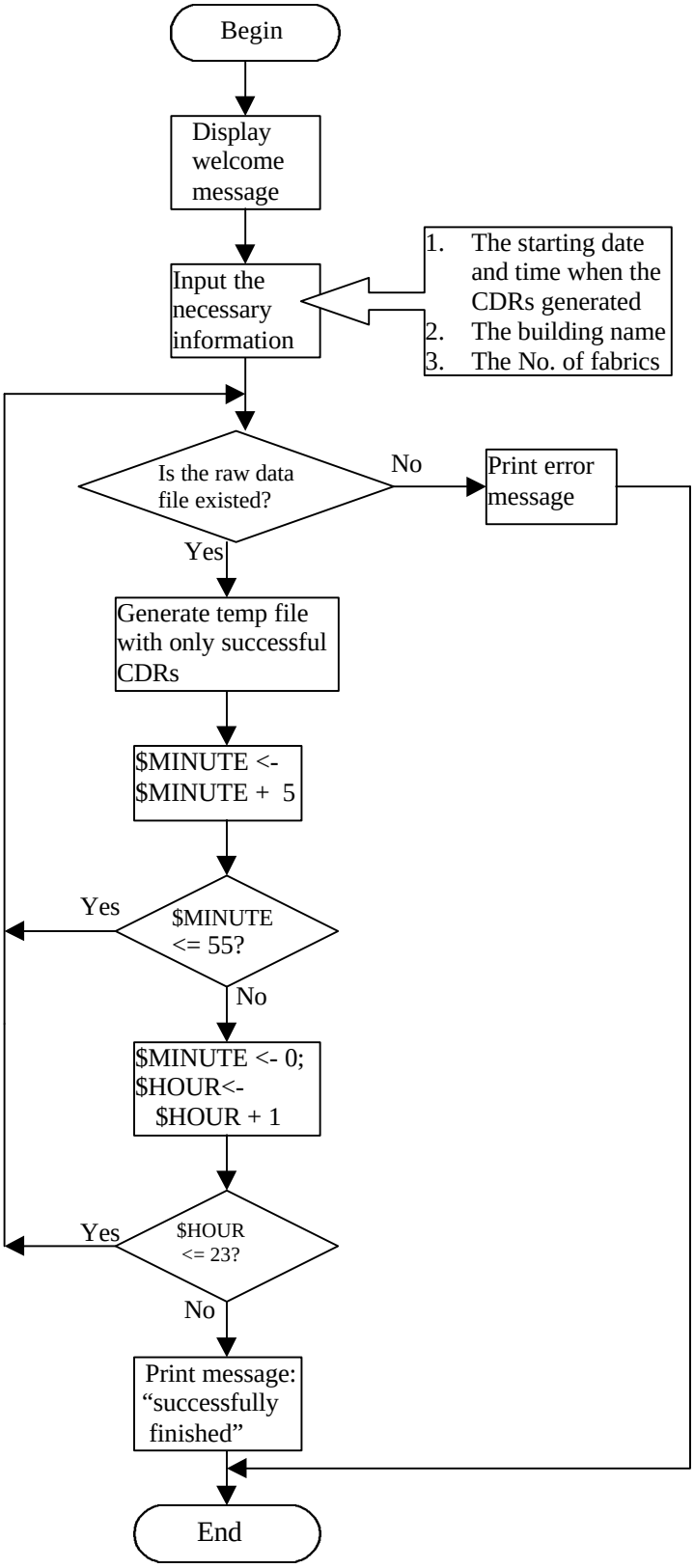

Figure 4-1. Flow chart of collect program.

After inputting time information, the program asks the user to input the location of the switch. Presently, the user can choose from *BUR, CAS, ISB, OWE, SHA* or *WHI*, as prompted. After this, the data collection begins or the program asks the user to input either the fabric number or room number. This depends on which building the user chooses. For example, if the user chooses building *ISB*, then the following prompt will appear.

```
Input the fabric # (1 / 2 / 3) or room #(118 / 208):
```

If the user chooses building *SHA*, then the data collection will begin immediately without further questions since there is only one switch with just one fabric in *Shanks Hall*. Before running this toolkit, the user should know exactly from which switch he or she is going to collect data.

After the user provides time and location information, the CDR collection starts without further intervention. The key function in the code is to extract successful CDRs. This is implemented by following pipe and filter command.

```
/usr/fore/foreview/bin/fvcrb2a
/kibitz/cdr/VT/$SubDirectory/$YEAR$DATE$PADHOUR$PADMINUTE$INTERVAL.cc \
| grep _TERM | grep -v _SETUP_ | grep normal | “grep 470000” > \
/test/TEMP$SubDirectory$DATE$PADHOUR$PADMINUTE
```

The long string after the *fvcrb2a* utility is invoked is the raw binary CDR file with its path. The CDR parameter “*call status*” is used as the primary filter to extract successful CDRs. There are five possible “values” of “*call status*” classified by FORE Systems [12]. For successful CDRs, the “*call status*” can only be CRM_CALL_INPROG_TERM, which represents call type C in Figure 3-1, or CRM_CALL_NEW_N_TERM, which is type D in Figure 3-1. So “*grep _TERM*” is used in the command line. However, the “*call status*” value of an abnormal case can be CRM_CALL_SETUP_TERM that also includes string “_TERM”. So we need to pick out CDRs which include character string “_SETUP_”. That is done by “*grep –v _SETUP_*”. The parameter “*call termination cause*” is used as the secondary filter to extract successful calls. During the test process, we found that some CDRs might meet the “*call status*” requirement but the “*call*

termination cause" can be "*41: temporary failure*" [4] or "*35: requested VPI/VCI not available*" [4]. We still want to exclude these calls to include only cause number 16 (*normal call clearing*) [4] and cause number 31 (*normal, unspecified*) [4]. So "*grep normal*" was added to the command line to finish this function.

In some cases, some raw data files do not exist, for example because of a malfunction somewhere in the switch, the data server, or the network. In this case, the following error message appears.

```
**********************
DATE = 0413
HOUR = 6
MINUTE=00
INPUT_FILE= 199904130600_05.cc

COULDN'T FIND ABOVE DATA FILE
You Need to Double Check The Directory /kibitz/cdr/VT/BUR-
ASX1K-2 to Make Sure the Input File Exists There.
**********************
```

In the example above, the raw data file *199904130600_05.cc* under */kibitz/cdr/VT/BUR-ASX1K-1* is not available. The user should check if the next raw data file *199904130605_05.cc* is available under the directory */kibitz/cdr/VT*. If it is available (which means that, perhaps, only one specific data file was lost), then the user has to re-start the data collection process. Special attention has to be paid to specifying the time input. In the example, after starting the collection the second time, the user needs to tell the program to start at *HOUR = 06*, rather than the default value of *00*, and to start at *MINUTE = 05*, rather than the default value of *00*. In this example, after restarting the data collection process and if the CDRs files are collected successfully the second time, then the user should use the following command.

touch /test/TEMPBUR-ASX1K-104130600

This will manually create the file *TEMPBUR-ASX1K-104130600* under */test*, where "*TEMP*" is the required string, "*TEMPBUR-ASX1K-1*" is the switch's identification,

"*0413*" is the date string, and "*0600*" is the time string. If one file for a specific time is missing, then the *concatenation* function, which is introduced in next section, will not finish successfully. The "*touch*" command makes up for the machine's "malfunction".

In more than six months of use, this abnormal case has occurred three times.

After running data collection successfully, the successful interim CDRs files are stored under directory */test/* named as:

TEMP$SubDirectory$DATE$HOUR$MINUTE.

The variable *$SubDirectory* is both the directory name under */kibitz/cdr/VT* and the switch's identification name stored in the raw CDRs files. Just as the file name implies, files under */test/* are temporary files to be accessed by "*concatenate*", described in next section. They are eventually deleted.

4.1.1.2. Concatenation of CDR Files: *concatenate*

The "*concatenate*" function is also written as a Unix C-shell script. This code merges all files with a name of the form *TEMP$SubDirectory$DATE$HOUR$MINUTE* in directory */test/* into one file. The key command lines in the script are as follows.

```
cp /test/TEMP$SubDirectory$DATE$PADHOUR$PADMINUTE
   /test/OLDfinal1$DATE;

cat /test/OLDfinal1$DATE
   /test/TEMP$SubDirectory$DATE$PADHOUR$PADMINUTE >
   /test/final1$DATE;
```

The basic function is to first create an initial file *OLDfinal1$DATE* when *HOUR = MINUTE = 00*, then concatenate the succeeding files one by one. There are two while loop conditions not listed above. The inter loop is to check if *$MINUTE <= 55*, the outer loop is to check if *$HOUR<=23*. The file *TEMP$SubDirectory$DATE$PADHOUR$PADMINUTE* is accessed sequentially according to the generated time. After finishing concatenation, a new file named *$SubDirectory$final1$DATE* is generated under directory */finaldata* for the next stage.

The strings "*PADHOUR*" and "*PADMINUTE*" appear in some toolkit codes. The reason is explained in the following example.

Assume *HOUR = 00* at the beginning of data collection or file concatenation. If the value of *HOUR* appears in a file name as *TEMP$HOUR*, then the correct file name is *TEMP00*. Later we will increase the value of *HOUR* using the C-shell command *set HOUR = `expr 1 + $HOUR`*. After evaluating this command, *HOUR* will become "*1*" rather than "*01*". So *TEMP$HOUR* will be *TEMP1* and *TEMP$PADHOUR* will be *TEMP01*, assuming *$PADHOUR = $Zero$HOUR*. They are two different files. So we have to add the "*0*" padding in front of *HOUR*. For the same reason, we use *$PADMINUTE* rather than *$MINUTE*.

4.1.1.3. B-Shell Version of CollCate: *CollCateBshell*

Before concluding Section 4.1.1, we briefly introduce *CollCateBshell* which is another version of *CollCate* written as a B-shell script. We created a B-shell script *CollCateBshell* and a *cron* entry file *wroger.crontab* to automatically collect CDRs data at a fixed time every day. A B-shell script is required since the B-shell is the default shell for *cron*. The use of *cron* eliminates the need to manually log in every day to run *CollCate*.

A *crontab* file consists of lines with a fixed format as follows. Each line consists of six fields separated by spaces or TABs. The first five fields are integer patterns that specify the minute (0-59), hour (0-23), day of the month (1-31), month of the year (1-12), and day of the week (0-6 with 0=Sunday). All five fields can be set to "*" meaning all legal values. The sixth field is a command that is executed by the shell at the specified times.

The following two lines in file *wroger.crontab* are related to the automatic execution of *CollCateBshell*.

```
00 12 *  *  *  /batch/CollCateBWHI;
10 12 *  *  * /batch/CollCateBISB;
```

As explained, the B-shell script file *CollCateBWHI* is evaluated at 12:00 and *CollCateBISB* is evaluated at 12:10 every day. The first command is to collect CDRs from a switch in *Whittemore Hall* and the second command is to collect CDRs from a switch in the Interop Lab in the *ISB*. The user can modify the code to collect data from any switch automatically. The initial successful CDRs file *$SubDirectory$final1$DATE* is automatically created under */finaldata*. The status is sent to the user automatically as an electronic mail message. The user only needs to run *SubsComma* and *FinalGenerate* to post-process the collected CDR file. These two functions are introduced below. If the file *wroger.crontab* is modified, the user must run the command "*crontab wroger.crontab*" to reactivate the entry.

4.1.2. Commas Substitution: *SubsComma*

As the name implies, the function of *SubsComma* is to substitute a TAB character for every comma in the file generated by "*CollCate*". As mentioned in Section 3.2.1, it is more convenient for high-level analysis programs to have each field separated by *TAB*. *SubsComma* is a simple Perl script. The flow chart is shown in Figure 4-2.

During execution, the user is asked to input the file name, with form *$SubDirectory$final1$DATE*, which was generated by *CollCate*. The output file name must have the form *$SubDirectory$final2$DATE* since this file will be processed later. The user should pay attention to the prompt messages when running the program.

4.1.3. Generation of the Final CDR File

The Unix C-shell script *FinalGenerate* finishes the final generation of the CDR file. *FinalGenerate* is a combination of multiple functions. We introduce the individual functions below.

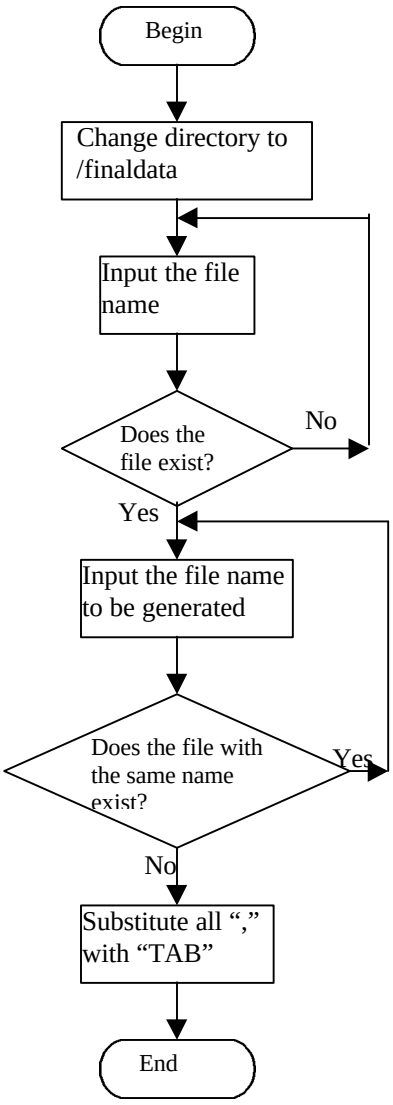

Figure 4-2. Flow chart of "SubsComma".

4.1.3.1. Generation of Interim File before Call Duration Calculation

The data file *$SubDirectory$final2$DATE*, described in Section 4.1.2, includes 43 CDR parameters (columns or fields). The file includes *call start time* at column 4 and *call collect time* at column 5. It does not include *call duration*, an important parameter. Before calculating the call duration, an interim file is generated with name of the form *$SubDirectory$final3$DATE*. It includes only two columns: *Call Start Time* and *Call Collect Time*.

The following Unix command is used to create this file.

```
cut -f4,5 $ROUTE/$SubDirectory$final2$DATE >
        $ROUTE/$SubDirectory$final3$DATE
```

It is much easier to calculate the call duration with only the two necessary columns available.

4.1.3.2. Generation of Call Duration

The *TimeCalculate* function is written in C. The input time format (ISO 8601 [16]) is of the form *YYYYMMDDTHHMMSS.S*, where character *T* separates the values of *Date* and *Time*. It is easier to calculate call duration after changing the time format above to some standard epoch-based time format. We chose the Unix epoch time format as our object time format. The Unix time epoch is 00:00 January 1, 1970 [17].

The standard C library provides a function *mktime(struct tm *timeptr)* that converts various time formats into time since the epoch, where structure *tm* is defined in the standard header file *Time.h* as shown below.

```
#ifndef _TM_DEFINED
struct tm {
      int tm_sec;     /* seconds after the minute - [0,59] */
      int tm_min;     /* minutes after the hour - [0,59] */
      int tm_hour;    /* hours since midnight - [0,23] */
      int tm_mday;    /* day of the month - [1,31] */
      int tm_mon;     /* months since January - [0,11] */
      int tm_year;    /* years since 1900 */
      int tm_wday;    /* days since Sunday - [0,6] */
      int tm_yday;    /* days since January 1 - [0,365] */
      int tm_isdst;   /* daylight savings time flag */
      };
#define _TM_DEFINED
#endif
```

The key function in *TimeCalculate.c* is "*fill*". It fills the fields *tm_year*, *tm_mon*, *tm_mday*, *tm_hour*, *tm_min* and *tm_sec* in the "*tm*" object. To do that, two standard C library functions, *memcpy* (copying characters between buffers) and *atoi* (converting a

string to an integer) are used. Note that *tm_year* is the number of years after 1900, so 1900 should be subtracted after changing *YYYY* to integer. Otherwise, this code will have trouble for calls started in 1999 and terminated in 2000. Also, the value of *tm_mon* is 0 to 11, where 0 represents January and 11 represents December accordingly, so 1 should be subtracted after changing *MM* to integer. (This problem existed in the source code until testing found some negative values for call duration in some final CDR files.)

TimeCalculate.c has been compiled successfully on both Windows 95 with the Microsoft VC 6.0 compiler and SunOS with the *gcc* compiler [18]. The executable code *TimeCalculate* was generated for the SunOS platform. To run *TimeCalculate*, the user needs to input file *$SubDirectory$final3$DATE* after *TimeCalculate* at the command line. The result is piped to file */finaldata/$SubDirectory$final4$DATE* for further processing. The file *$SubDirectory$final4$DATE* includes three columns, *Call Start Time*, *Call Collect Time* and *Call Duration*, where the first two parameters have been changed to the Unix epoch time format.

4.1.3.3. Generation of the CDR File with Call Duration

Given the call duration, this value must be inserted into the original CDR file. In Unix, *cut* and *paste* commands are useful in performing such functions. The *cut* command can select a list of columns or fields from one or more files. The default field delimiter is *TAB*. The *paste* command can merge corresponding lines of one or more files into vertical columns, separated by a *TAB*.

Columns 4 and 5 of file *$ROUTE/$SubDirectory$final2$DATE* are cut to calculate call duration and to generate the file *$ROUTE/$SubDirectory$final4$DATE*. We use the *cut* command to cut the first three columns of file *$ROUTE/$SubDirectory$final2$DATE* to a file *temp1* and the columns 6 through 43, inclusive, to another file *temp2*. Then we use the *paste* command to merge three files *temp1*, *$ROUTE/$SubDirectory$final4$DATE* and *temp2* to generate the file *$ROUTE/$SubDirectory$final5$DATE*.

4.1.3.4. Final Generation of CDR File with Header

The file *$ROUTE/$SubDirectory$final5$DATE* includes only a list of CDRs without object names. A header is created manually and concatenated to the beginning of file *$ROUTE/$SubDirectory$final5$DATE*. Until now, the final CDR file has been generated with the name *$SubDirectory$final$YEAR$DATE* under directory */finaldata*. All the interim files should be purged. The final file is ready for high-level processing.

4.1.3.5. Standardization of Final File

Section 5.2.1 in Chapter 5 discusses the recommendation for CDRs objects. Under directory */batch*, we created a "*Standard_Header*" file that includes every recommended CDR object. At the end of *FinalGenerate* evaluation, following message will display:

```
Do you want to standardize the final file? (y/n)
```

If the answer is "*y*", then a file with name *$SubDirectory$Standard$YEAR$DATE* is generated under */finaldata*. This file includes some empty fields for objects whose values are not available.

4.2. CDR Collection from IBM 8265 Switches

The second toolkit is used to collect CDRs from IBM 8265 switches. It has been mentioned that collection of CDRs from IBM 8265 switches is different from collection of CDRs from FORE switches. A good understanding of MIBs supported by the IBM 8265 is critical. After loading and checking every public MIB supported by the IBM 8265 switch, we did not find any relevant CDR data. Some objects identifiers included in public MIBs are empty. The IBM 8265 private MIB data file V4.12 (the latest version at the time of this writing) did not contain information about successful SVC calls. However, the private MIB does contain some CDR parameters for in-progress SVC calls, failed (abnormal) SVC calls, counters associated with the virtual connections, and information about the switch's internal cross connections. Therefore, the second toolkit generates CDR files including this information. The second toolkit also includes two B-

shell scripts, *Generate24682* and *Generate25090*, which are used as entries in the *crontab* file so the in-progress CDRs can be collected automatically from two 8265 switches. The code *FinalGenIBM* is used to post process these automatically collected CDR files, estimate the call duration, and standardize the final file.

Although the generated CDRs will not be as useful for high-level analysis as those collected by the first toolkit, we are able to accumulate valuable experience from the process of CDR collection and programming that will be helpful for collection of CDRs from other network elements from other vendors. Also, the CDRs generated from the IBM 8265 switch together with CDRs generated from FORE Systems ASX switches provide a basis for recommended standards for CDR objects as discussed in Chapter 5.

The second toolkit is in directory */tcl*. There are four independent sub-directories, *8265SvcEntry*, *8265SvcLogEntry*, *8265CxVclEntry*, and *8265vcXConnectEntry*. The following sections discuss the available CDR parameters in the four different directories. Details of the toolkit are also provided.

4.2.1. Introduction to CDR Parameters

The scripts under *8265SvcEntry* are used to collect CDRs for in-progress SVC calls. The final generated CDR file includes the following parameters.

- *atmSvcInterfaceIndex*: The index value of the ATM interface used by this SVC.
- *atmSvcSiVpi*: The VPI value of the signaling channel for this entry.
- *atmSvcSiVci*: The VCI value of the signaling channel for this entry (usually there is one signaling channel per interface defined by VPI/VCI = 0/5).
- *atmSvcCallReference*: The Q.2931 call reference value for this SVC.
- *atmSvcEndPointReference*: One of the Q.2931 end-point reference values used by this SVC.
- *atmSvcCallingNumber*: The ATM address of the calling party.
- *atmSvcCalledNumber*: The ATM address of the called party.
- *atmSvcClear*: This variable allows a network manager to clear this SVC.
- *atmSvcCreationTime*: The time when the call was placed.
- *atmSvcVpi*: The VPI value used by this SVC.

- *atmSvcVci*: The VCI value used by this SVC.

The scripts under *8265SvcLogEntry* collect CDR information about the latest unsuccessful SVC completed in this node. The following CDR parameters are collected for SVC calls finished abnormally.

- *atmSvcLogIndex*: An ID value for this entry assigned by the local SNMP agent. It is allocated in decreasing order so that a "get-next" request on the table will retrieve the latest calls first.
- *atmSvcLogInterfaceIndex*: The ifIndex value of the ATM interface used by this SVC.
- *atmSvcLogCallingNumber*: The ATM address of the calling party.
- *atmSvcLogCalledNumber*: The ATM address of the called party.
- *atmSvcLogCreationTime*: The time when the call was placed.
- *atmSvcLogTime*: The time when the call was cleared.
- *atmSvcLogClearCause*: The cause of the clearing of the SVC.
- *atmSvcLogForwardQoS*: QoS with 0 for unspecified, 1 for class A, 2 for class B, 3 for class C, and 4 for class D.
- *atmSvcLogBackwardQoS*: QoS with 0 for unspecified, 1 for class A, 2 for class B, 3 for class C, and 4 for class D.
- *atmSvcLogForwardBW*: The forward bandwidth requested for this call.
- *atmSvcLogBackwardBW*: The backward bandwidth requested for this call.
- *atmSvcLogServiceCategory*: The service category, with 1 for other (none of the following), 2 for CBR, 3 for rt-VBR, 4 for nrt-VBR, 5 for ABR, and 6 for UBR.
- *atmSvcLogClearLocation*: The originator that caused this call to be cleared.

The scripts under *8265CxVclEntry* collect counters for each VC connection. The following CDRs parameters can be collected.

- *cxVclIndex*: The interface index (ifIndex) value of MIB-II.
- *cxVclVpi*: The VPI value for this connection.
- *cxVclVci*: The VCI value for this connection.

- *cxVclInCells*: The number of valid cells received on this connection since monitoring started.
- *cxVclInDiscards*: The number of cells discarded on this connection since monitoring started.
- *cxVcOutCells*: The number of cells transmitted on this connection since monitoring started.
- *cxVclOutDiscards*: The number of cells to be transmitted and discarded on this connection since monitoring started.
- *cxVclRowStatus*: An entry in this table starts the corresponding monitoring counters for this connection.

The scripts under *8265vcXConnectEntry* collect information about the cross-connections established within the switch for all existing Virtual Channel Link (VCL)-based PVCs and SVCs. The following information can be collected.

- *vcXIndex*: The connection entrance interface number for this ATM port.
- *vcXInVpi*: The VPI value for this connection in the entrance.
- *vcXInVci*: The VCI value for this connection in the entrance.
- *vcXOutIndex*: The connection exit interface number for this ATM port.
- *vcXOutVpi*: The VPI value for this connection in the exit.
- *vcXOutVci*: The VCI value for this connection in the exit.
- *vcXType*: Indicates if the call is unicast or multicast.
- *vcXDirection*: Indicates the direction of the connection, upstream versus downstream, as seen from the root.

4.2.2. The Code Introduction

The file structure under each of the four directories is the same except that two B-Shell version scripts and *FinalGenIBM* are included in directory *8265SvcEntry*, which includes code to collect CDRs of in-progress calls. As introduced in Chapter 3, the reason is that we need to infer the call duration of successful calls from the in-progress calls. In every directory, there is a *readme* file, an interim file generation program written as a

Unix shell script, a *SubsBrace* script written in Perl, and multiple pairs of *coll_parameter* and *gen_parameter* programs written using Tcl/Scotty. Compared with *8265SvcEntry*, *8265SvcLogEntry* category contains more interesting objects such as QoS value, bandwidth, call start/call collect time, and service category. In the following discussions, we consider scripts in the *8265SvcLogEntry* directory as an example.

There are thirteen related objects (parameters) within this category. Each collection of objects is associated with two files, *coll_parameter* and *gen_parameter*, where *parameter* is just the abbreviation of the full object that uniquely identifies the object. So there are 13 pairs of *coll-parameter and gen_parameter* files. For example, the first parameter is *atmSvcLogIndex*. The pair of files *coll_SLI1* and *gen_SLI1* is used to collect this object. Both files are executable. The file *gen_parameter* calls *coll_parameter*, but both of them are transparent to the final user. A user only needs to run *8265SvcLogGenerate*, the interim file generation script, written as a Unix C-Shell script, and *SubsBrace*.

The *8265SvcLogGenerate* script calls various *gen_parameter* functions to generate an interim file with proper format. By running *SubsBrace*, all brace characters ("}") in the interim file are deleted to generate the final CDR file. The following sections introduce programs *coll_SLI1*, *gen_SLI1*, *8265SvcLogGenerate*, and *SubsBrace*. All programs are under the directory */tcl/8265SvcLogEntry*.

4.2.2.1. Object Collection: *coll_SLI1*

As the file's name implies, file *coll_SLI1* collects object *atmSvcLogIndex* and outputs the result to the standard output. The code is simple, but important, so we introduce most key statements in the code.

```
mib load /ibm/65MIBV4.MIB
```

;Specific MIB data file must be loaded first to collect the object data. Here the IBM private MIB file is loaded.

```
set ipAddress [lindex $argv 0]
```

; ipAddress will be the first parameter of the command line.

```
set s [snmp session –address $ipAddress –community
```

```
private]
```

;Set “*s*” as an SNMP session. The *-address* option defines the network address of the SNMP peer. The value of address may be an IP address in dotted notation (e.g. 198.169.34.1) or a hostname that can be resolved to an IP address. The *-community* option is specific for SNMPv1 and SNMPv2C sessions. It defines the community string that is used to identify the sender of SNMP messages. The default community string is "*public*". The user needs to check the specific 8265 switch if a “*private*” community has been defined.

```
puts stdout SvcLogIndex
```

;This command simply adds a title for the output.

```
$s walk SvcLogIndex {atmSvcLogIndex} {puts $SvcLogIndex}
```

;The format of this command is: *snmp# walk varName varbindlist body*. The walk session traverses, or walks a whole MIB subtree. The command repeats sending *getbulk (getnext)* requests until the returned *varbind list* (a list of MIB instances) is outside of the subtree given by *varbind list*. If *varbind list* is retrieved from the agent, the Tcl script body is evaluated. In this example, the *varbindlist* is just one simple instance *atmSvcLogIndex*. The *varName SvcLogIndex* will contain the value of *atmSvcLogIndex*. After evaluating the *body* (*puts $SvcLogIndex*), the value of *atmSvcLogIndex* is printed on the screen.

4.2.2.2. Parameter Generation: *gen_SLI1*

The file *gen_SLI1*, as its name implies, generates a parameter file, *SLI1*, that includes the results of *atmSvcLogIndex* collected by *coll_SLI1*. The flow chart of *gen_SLI1* is shown in Figure 4-3.

The following code determines if the IP address is active.

```
   set active [ catch { exec ping $ipAddress 2} result]
   if {$active != 0 } {
   puts stderr "$ipAddress is not active!"
   exit
}
```

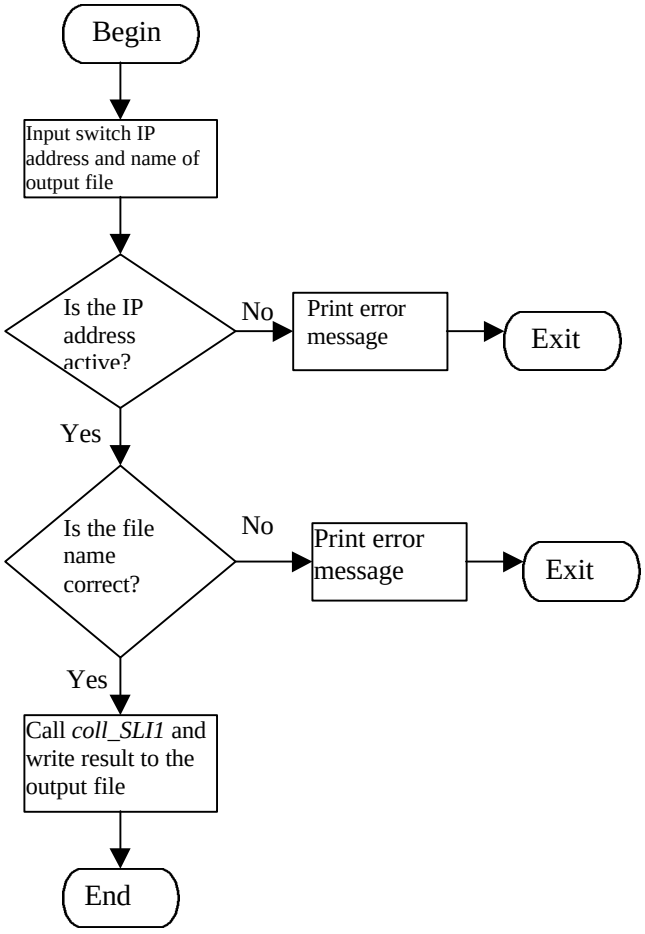

Figure 4-3. Flow chart of *gen_SLI1* code.

In the above, *set*, *catch* and *exec* are TCL commands and *ping* is the standard TCP/IP command embedded in most Unix systems. The *catch* command in TCL is used to evaluate and trap errors on a specified script. It "catches" any errors that may occur in the script. If a script executes without error, the result is "0". Otherwise the result is "1". The character "2" in the code is the timeout value. If there is no reply from the IP address within two seconds, this address is considered inactive.

The code to write the final result in the output file is shown below.

```
set fileId [open $outputFile {RDWR CREAT}]
set result [exec coll_SLI1 $ipAddress ]
puts $fileId $result
close $fileId
```

Here, the name *outputFile* must be *SLI1* for this example. More generally, *gen_parameter* calls *coll_parameter* to generate a new temporary file. The name of this file must be *parameter*. It is transparent to the final user since both *gen_parameter* and *coll_parameter* scripts are transparent to the user.

4.2.2.3. Interim File Generation: *8265SvcLogGenerate*

As indicated above, there are 13 objects (parameters) related to the category *SvcLogEntry*, so there are 13 pairs of *coll-parameter* and *gen_parameter* files. After running each pair of collect and generate functions, 13 files will be generated. Each file has a header, containing the parameter names, and multiple records in rows. These files are meaningful, but the information included in each file is not "user friendly." The Unix Shell script *8265SvcLogGenerate* is used to extract one specific column in every file and merge all the extracted columns into one interim file. This script file *8265SvcLogGenerate* shields all the *gen_parameter* and *coll_parameter* functions from the user. The user only runs *8265SvcLogGenerate* which generates an interim CDR file. Some sample commands in the *8265SvcLogGenerate* script are listed below.

- `gen_SLI1 $ipAddress SLI1 ;` $ipAddress is the IP address of 8265 switch

The file *SLI1* will be generated with the following format.

```
SvcLogIndex
{1.3.6.1.4.1.2.6.33.1.9.1.6.1.1.2147483453 INTEGER 2147483453}
{1.3.6.1.4.1.2.6.33.1.9.1.6.1.1.2147483454 INTEGER 2147483454}
{1.3.6.1.4.1.2.6.33.1.9.1.6.1.1.2147483455 INTEGER 2147483455}
. . .
```

Only the last column is the value of *atmSvcLogIndex.*

- `cut -d" " -f3 SLI1 > SLI1_temp`

This command is used to cut the third field separated by "space" to generate a temporary file *SLI1_temp* with format as follows.

```
SvcLogIndex
2147483453}
```

```
2147483454}
2147483455}
. . .
```

Note the brace character (“}”) is not useful and it is later deleted.

Similarly, the format of other temporary files, such as *SLII2_temp* (*atmSvcLogInterfaceIndex*), is shown below.

```
SvcLogInterfaceIndex
601}
601}
501}
. . .
```

After generating all thirteen temporary files, the following command merges them to form an interim file.

```
paste SLI1_temp SLII2_temp SLCN3_temp SLCN4_temp
   SLCT5_temp SLT6_temp SLCC7_temp SLFQ8_temp SLBQ9_temp
   SLFB10_temp SLBB11_temp SLSC12_temp SLCCL13_temp   >
   8265SvcLogInterim
```

The latest generated file, *8265SvcLogInterim*, contains vertical columns separated by a TAB character. The format of file *8265SvcLogInterim* is as follows.

```
SvcLogIndex  SvcLogInterfaceIndex   SvcLogCallingNumber ...
2147483453}  601}
 47:00:00:15:40:34:20:00:10:20:00:00:07:00:A0:B1:07:0A:C6:00}...
2147483454}  601}
 47:00:00:15:40:34:20:00:10:20:00:00:07:40:00:82:10:00:01:03}...
2147483455}  601}
 47:00:00:15:40:34:20:00:10:20:00:00:07:00:A0:B1:07:0A:C6:00}...
. . .
```

Every item in this file is followed by a useless brace (“}”) which is deleted as described in the next section.

4.2.2.4. Elimination of Braces: *SubsBrace*

The *SubsBrace* script is used to delete every brace character ("}") in the interim file. It is written as a Perl script and is a modified version of the file *SubsComma* introduced in Section 4.1.2.

After running *SubsBrace*, the final CDRs file for this category is generated. Note that there are four categories so there are four CDR files generated for each IBM 8265 switch.

4.2.2.5. B-Shell Version of Interim File Generation: *GenerateIpAddress*

The two B-shell versions of *8265SvcEntryGenerate* in the *8265SvcEntry* directory are *Generate24682* and *Generate25090*. In the lab, the last two bytes of the two IBM 8265 switches' IP addresses are setup as 246.82 and 250.90. The user can easily recognize the code with the corresponding switch if the user knows the switch's IP address. After properly configured in *wroger.crontab* file, which was introduced in Section 4.1.1.3, these two files can collect in-progress calls in 5-minute intervals. The collected files are stored in directory */tcl/8265SvcEntry/TempDirectory* with name format as:

TEMP246SvcEntry$DATE$TIME or *TEMP246SvcEntry$DATE$TIME*.

There are 11 objects in the *SvcEntry* group. Call collection time is not given and it cannot be given for an in-progress call. These temporary files include *collection time* as the 12th object. Compared with the C-shell version of this file, the inclusion of collection time field is new.

The following command is used to first create a temporary file which includes one line.

```
echo $YEAR$DATE$T$HOUR$MINUTE > CollectionTimeTemp;
```

Then the "*cat*" command is used to merge *CollectionTimeTemp* multiple times (the number of in-progress calls). A temporary file named *CollectionTimeTemp2* is generated which includes exactly one column. The number of rows in this file is equal to the number of in-progress calls.

4.2.2.6. Final Generation of CDRs File with Call Duration and Standard Header: *FinalGenIBM*

As shown in Figure 3-3, *FinalGenIBM* code contains multiple functions. Most of them, such as file concatenation and standardization of file header, have been introduced during the discussion of the first toolkit. Three new commands are introduced below.

- ```
  awk '{print $12, $1, $2, $3, $4, $5, $6, $7, $8, $9,
  $10, $11 }' \
     $Route/finaldata/$Switch$final1$DATE >
     $Route/finaldata/temp1;
  ```

Awk is a utility to search and process a pattern in a file in Unix [11]. The above command re-formats the output file to make the last field become the first field.

- ```
  sort -n -r +4 +5 $Route/finaldata/temp2 >
  $Route/finaldata/temp3;
  ```

Sort is a Unix command to sort and/or merge files [11]. The option –n makes the file sorted by numeric values of the specific fields. The option –r keeps the sorted list in reverse order (for example, *z* precedes *a* and *2* precedes *1*). The *+field* arguments +4 and +5 make the file sorted according to the values in the fifth column and sixth column.

- ```
  uniq –1 $Route/finaldata/temp3|sort -n >
  $Route/finaldata/$Switch$final2$DATE;
  ```

Uniq is a Unix command to display lines of a file that are unique [11]. The *–field* argument (–1) skips the first field in each line. The input file *temp*3's first field is "collection time". However, *temp3* contains the same record multiple times with different values of "collection time". For example, a call from 12:00 to 13:01 will appear at least twelve times with different "collection time" given polling interval as 5 minutes. We are only interested in the call with collection time equal to 13:00. The *uniq* command only keeps the latest record by deleting the redundant records without comparing the first field.

For example, assumes the *temp3* file contains following lines.

```
collection_time    interface_index    VPI    VCI
19990608T0925      601                0      33
19990608T0925      601                0      33
```
```

```
19990608T0920     601               0     33
19990608T0920     601               0     33
19990608T0915     601               0     33
19990608T0915     601               0     33
19990608T0925     601               0     36
```

After running the "*uniq temp3*" command, the following result will be displayed.

```
collection_time   interface_index   VPI   VCI
19990608T0925     601               0     33
19990608T0925     601               0     36
```

To run this code, the user needs to input the year and date when the data was collected. The IP address of the switch is also input by the user. The file will first check whether the required files in */tcl/8265SvcEntry/TempDirectory* are readable or not. In the lab environment, the two 8265 switches are sometimes turned off due to various reasons. This added some difficulty during testing. The final generated files with name formatted as *$Switch$final$DATE* are saved in the */tcl/8265SvcEntry/finaldata* directory, where *$Switch* can be the string *246* or *250* depending on from which switch the data was collected.

4.3 Summary

This chapter introduced scripts and programs included in the two toolkits. The first toolkit, for the FORE Systems ASX switches, is stored under directory */batch*. For every switch, exactly one final CDR file will be generated. To date, we have been automatically collecting CDRs from both *Whittemore Hall* and the *Interop* Lab in the *ISB.* Every day two CDRs files are automatically generated under */finaldata*.

The second toolkit, for IBM 8265 switches, is stored under */tcl*. For each switch, there are four categories of parameters, so four different CDR files are generated. These CDR files provide parameters for in-progress SVC calls, failed SVC calls, counters for each VC connection, and information about the cross-connection setup in the switch for all existing VCL-based PVCs and SVCs. The values of call duration of possibly

successful SVC calls are estimated. These CDR files have not, to date, been used for high-level analysis. However, the method used to create toolkits should be helpful for further research and data collection from IBM and other switches.

User manuals for both toolkits are included in Appendix B. The classification of available CDRs is shown in Appendix C.

Chapter 5. Recommendations for Standards for CDR Generation and Objects

We have created toolkits to collect CDRs from two different types of switches. Based on the different methods used to collect CDRs, the various objects present in the CDRs, and the requirements for analysis and network management, we have developed a set of recommendations for both the generation and conveyance of CDRs and objects within CDRs.

5.1. Recommendation for CDR Generation and Conveyance

The approaches used in the collection of CDRs from two different ATM switches are different since the CDR generation methods are different. For FORE ASX switches, the CDRs are transferred to an external server in real time so the first toolkit needs to post-process the available raw CDR file. For IBM 8265 switches, the CDRs are stored in the switches' RAM and will not be transferred to an external server automatically. The second toolkit works as an SNMP manager to talk with the switches first. The second toolkit also includes similar functions finished in the first toolkit. The approach used in the second toolkit is more complex than that used in the first toolkit.

We recommend that the CDRs should be generated periodically and automatically by the switch. The generated CDRs should also be updated frequently to provide usage details, on an end-to-end basis, regardless how long the connections may last. A recommended functional model for generation and collection of CDRs is shown in Figure 5-1. The components to match functions shown in Figure 5-1 are shown in Figure 5-2.

The following commands related to CDR generation and conveyance should be supported by the ATM switch configuration commands.

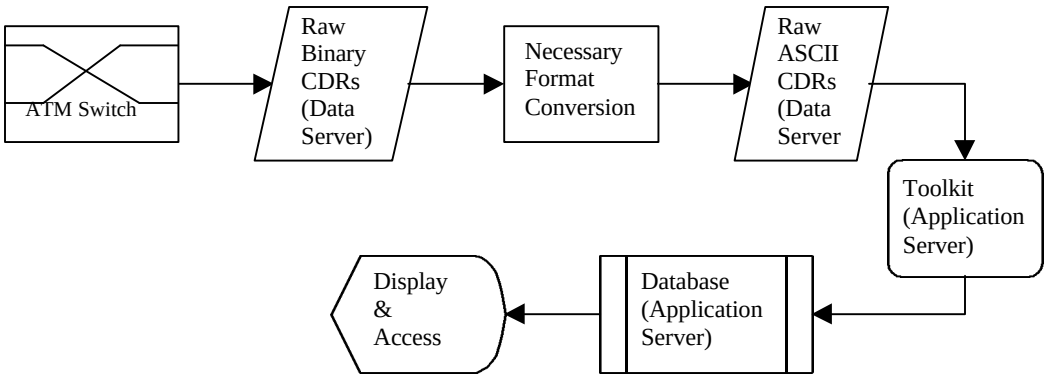

Figure 5-1. A functional model for CDRs generation and collection.

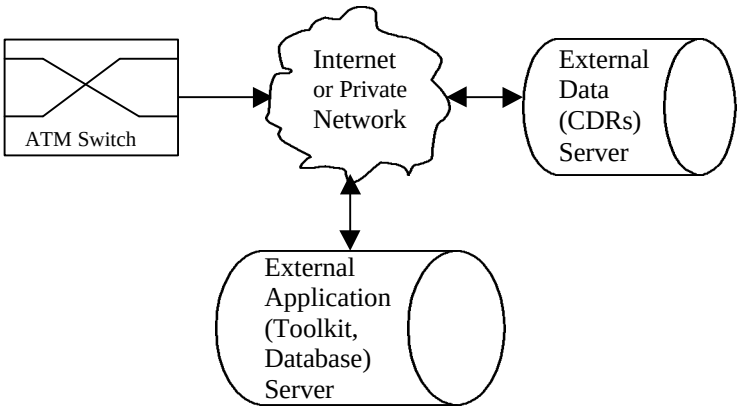

Figure 5-2. A component model for CDRs generation and collection.

5.1.1. CDR Output On/Off

This command should turn the switch's output of CDRs either on or off. It may be desirable to disable generation of CDRs from some special switches for security or other reasons. The default for CDR output should be "off" since some customers may not be interested in the analysis of CDRs.

5.1.2. CDR Filter

This command should allow users to record or ignore certain types of calls, such as PVCs, SVCs, Smart or Soft Permanent Virtual Path Connections (SPVPs), Smart or Soft Permanent Virtual Circuit Connections (SPVCs), or failed calls. In our own project, we

are presently only interested in successful SVC calls. If the filter command was available, we could use it to select the interesting calls, thus simplifying programming and reducing the amount of data that must be processed.

5.1.3. CDR Output Destination Configuration

This command should allow the user to choose the destination data server IP address, port ID, output file name and destination path in the server. This command should also allow the user to configure the owner of the files, where the owner should be an existing account name at the server.

5.1.4. CDR Generation Interval Setup

This command should allow the user to setup the interval for output of CDRs. For backbone switches, the CDRs can result in a huge amount of data, so the output interval could be set to be shorter than the output interval of non-backbone switches, thus reducing the storage required at the switch.

5.1.5. CDR Configuration Display

The configuration display command should allow the user to view all current configurations. This command is useful if the user wants to check whether the current configuration meets the requirements.

5.1.6. Additional Commands

The above commands are needed to configure the generation and conveyance of CDRs. Vendors may also choose to provide more commands. For example, a vendor might allow specification of a second output server that could be used in the event of a failure at the first server. A vendor could also provide commands to turn CDR output on

or off at some specific logical ports or on the basis of a VPI/VCI rather than the whole switch. These schemes are left for the vendor to select and are not considered to be mandatory.

5.2. Recommendations for CDR Object and Output Format Standard

CDR objects can be subdivided into two groups. The first group includes objects such as traffic parameters, calling and called party addresses, etc. These objects can be extracted from ATM information elements (IEs) and can be stored in a management information base. The second group includes objects such as call start time and cells transmitted, etc. These objects are not included in any IEs. They must be observed and measured by the switch at the point where the user accesses the service and, presumably, recorded in the management information base. Every CDR object can be extracted either from the switch's private MIBs or from public MIBs supported by the switch.

This section provides recommendation for standard CDR objects. The related background knowledge about ATM messages, information elements and signaling procedure has been described in Chapter 2. The reader should understand how objects in the first group are extracted upon understanding the background knowledge. The extraction of objects in the second group is based on embedded software in the switch.

5.2.1. Recommendations for Standard CDR Objects (Parameters)

One goal of our research is to provide recommendations for standard CDR objects. There is presently no such standard because of several reasons. Compared with other services mentioned in Chapter 2, ATM technology is newer. ATM backbone networks are not widely deployed yet. There exist some backbone switches but most network managers still have not attempted to optimize network management from a higher level, such as on a per-connection basis. Accordingly, most vendors have not paid much attention to the inclusion of CDR objects in their ATM switch products. We hope that the recommended CDR standard will be helpful for both the public network managers and product vendors.

The standard CDR objects are recommended based on the collection of CDRs and ATM Forum documents [3,4,5], ITU-T recommendations [6,21,22] and Internet Engineering Task Force (IETF) RFCs (http://www.faqs.org/rfcs/rfc-titles.html). The ATM product manuals and technical documents from IBM [7,14,20] and FORE [9,12,13] are also helpful for developing the standard recommendation.

The CDR objects are subdivided into eight groups. They are discussed in Sections 5.2.2.1 to 5.2.2.8.

5.2.1.1. Call Identification Objects

Call identification objects are used to specify the call type, address, reference number, port ID, and VPI/VCI values. There are ten objects in this group and descriptions are given below.

- Call_Source_Reference and Call_Term_Reference: These are integer values used to uniquely identify a call. The Call Reference IE is included in every message. The Call Reference field identifies the call at the local and remote user-network interface to which the particular message applies. The call source reference value is assigned by the originating side of the interface for a call and the call term reference value is assigned by the destination site of the interface for this call. They remain fixed for the lifetime of this call.
- Call_Method: This field is used to identify a call as a point-to-point or point-to-multipoint call. This field can be extracted from the Broadband Bearer Capability (BBC) IE. This IE must be and can only be included in the SETUP message.
- Call_Type: The possible value can be PVC, SVC, PVP, SVP, etc. This field is not included in any IEs. It is the vendor's responsibility to provide the proper values.
- Called_Party_Address: This is the destination's ATM address. This field is specified in the Called Party Number IE. This IE must appear in SETUP, LEAF SETUP REQUEST and ADD PARTY messages. Its appearance in LEAF SETUP FAILURE is optional.

- Called_Party_Subaddress: The format of this field is similar to Called Party Address. This field is specified in the Called Party Subaddress IE. The subaddress of the called party is used only to convey an ATM address in the ATM Endsystem Address format across a public network that supports the E.164 format. The ATM Endsystem Address is based on the ISO NSAP format but is not an NSAP. This IE can be included in any message that includes the Called Party Number IE.
- Calling_Party_Address: It is the source's ATM end-system address. This field is specified in the Calling Party Number IE. This IE can be included in the SETUP message.
- Calling_Party_Subaddress: The format of this field is similar to Calling Party Address. This field is specified in the Calling Party Subaddress IE. This IE can be included in SETUP, LEAF SETUP REQUEST and ADD PARTY messages. Its appearance is optional.
- Call_In_Port/VPI/VCI (the source port ID and VPI/VCI value of source port) and Call_Out_Port/VPI/VCI (the destination port ID and VPI/VCI value of destination port): The port ID can be obtained through public MIB II (IETF RFC 1213). The VPI/VCI values are specified in the Connection Identifier IE. This IE can appear in SETUP, CALL PROCEEDING and CONNECT messages. If this IE appears in SETUP message, then it must be included in the other two messages. Some switches may not support the full range of VPI values (0-255) since some VPI bits may be not active. VCI values 0-15 are reserved by ITU-T. VCI values 16-31 are reserved by ATM Forum. Only values 32-65,535 are available for users. VPI/VCI values used by a PVC cannot be used by an SVC.

5.2.1.2. ATM Traffic Related Objects

There are fourteen objects to describe ATM traffic. They are shown below.

- F_PCR_CLP_0 (Forward PCR with CLP = 0)
- F_PCR_CLP_01 (Forward PCR with CLP = 0/1)
- B_PCR_CLP_0 (Backward PCR with CLP = 0)

- B_PCR_CLP_01 (Backward PCR with CLP = 0/1)
- F_SCR_CLP_0 (Forward SCR with CLP = 0)
- F_SCR_CLP_01 (Forward SCR with CLP = 0/1)
- B_SCR_CLP_0 (Backward SCR with CLP = 0)
- B_SCR_CLP_01 (Backward SCR with CLP = 0/1)
- F_MBS_CLP_0 (Forward MBS with CLP = 0)
- F_MBS_CLP_01 (Forward MBS with CLP = 0/1)
- B_MBS_CLP_0 (Backward MBS with CLP = 0)
- B_MBS_CLP_01(Backward MBS with CLP = 0/1)
- Tagging_Forward (0: tagging not requested; 1: tagging requested)
- Tagging _Backward (0: tagging not requested; 1: tagging requested)

All of the above parameters are specified in the ATM Traffic Descriptor (ATD) IE. This IE also includes a one-byte Best Effort Indicator (BEI) that is an ATM service category parameter. It will be further discussed with the service category parameters. The ATM Traffic Descriptor IE must be included in a SETUP message. If an ABR connection is being requested, this IE must be included in a CONNECT message to specify MCR value.

These parameters can also be specified in the Alternative ATD IE and Minimum Acceptable ATD IE. The purpose of the Alternative ATD IE is to specify an alternative traffic descriptor for the negotiation of traffic parameters during call/connection setup. The purpose of the Minimum ATD IE is to specify the minimum acceptable ATM traffic parameter in the negotiation of traffic parameters during call/connection setup. These two IEs may be included in SETUP messages.

Not all combinations of these parameters are valid. Table 5-1 shows the valid combination of traffic parameters for best-effort connections. Table 5-2 lists the valid combination of traffic parameters for any service category.

Table 5-1. Valid Combinations of Traffic Parameters for Best Effort [4]

Combination of Traffic Parameters for Best Effort Connections
Forward PCR (CLP = 0 + 1) Backward PCR (CLP = 0 + 1) Best Effort Indication Available

Table 5-2. Valid Combinations of Traffic Parameters in a Given Direction [4]

Allowable Combinations of Traffic Parameters for a Given Direction
PCR (CLP = 0) / PCR (CLP = 0 + 1)
PCR (CLP = 0) / PCR (CLP = 0 + 1) / Tagging = 1 (requested)
PCR (CLP = 0 + 1) / SCR (CLP = 0) / MBS (CLP = 0)
PCR (CLP = 0 + 1) / SCR (CLP = 0) / MBS (CLP = 0) / Tagging = 1(requested)
PCR (CLP = 0 + 1)
PCR (CLP = 0 + 1) / SCR (CLP = 0 + 1) / MBS (CLP = 0 + 1)

5.2.1.3. QoS Related Objects

There are ten objects related to quality of service that are used to describe QoS classes and QoS parameters. They are shown below.

- F_QoS (Forward QoS Class) and B_QoS (Backward QoS Class): These two objects are specified in the QoS Parameter IE. This IE may appear in a SETUP message. The possible values can be QoS Class 0, 1, 2, 3 or 4. The meaning of these values has been discussed in Chapter 2.
- C_CTD (Cumulative End-to-End CTD) and M_CTD (Maximum End-to-End CTD): These two objects are specified in the End-to-End Transit Delay IE. This IE can be included in SETUP, CONNECT, ADD PARTY and ADD PARTY ACK messages.
- AF_CDV (Acceptable Forward Peak-to-Peak CDV)
- AB_CDV (Acceptable Backward Peak-to-Peak CDV)
- CF_CDV (Cumulative Forward Peak-to-Peak CDV)
- CB_CDV (Cumulative Backward Peak-to-Peak CDV)
- AF_CLR (Acceptable Forward CLR)
- AB_CLR (Acceptable Backward CLR)

The last six objects are specified in the Extended QoS Parameter IE. This IE can be included in SETUP and CONNECT messages. The Extended QoS Parameter IE is used to indicate the individual QoS parameter values acceptable on a per call basis and to indicate the cumulative QoS parameter values.

5.2.1.4. ATM Adaptation Layer (AAL) Object

There is only one object to identify the AAL category. It is described below.

- AAL_Type: This field is specified in the AAL Parameter IE. The AAL IE can be included in messages SETUP, CONNECT, ADD PARTY and ADD PARTY ACK. The inclusion of this IE is not mandatory. The possible AAL values and associated meanings are: 0, AAL Type 1; 3, AAL Type 3 / 4; 5: AAL Type 5; 16: undefined AAL.

5.2.1.5. ATM Service Category Objects

There are three objects to describe the ATM service category. They are described below.

- Bearer_Class: The possible values are BCOB-A (Broadband Connection Oriented Bearer, Class A), BCOB-C, BCOB-X or Transparent VP Service. BCOB-A is used for connection-oriented CBR service. BCOB-C is used for connection-oriented VBR service. A network supporting BCOB-A or BCOB-C may perform inter-networking based on the AAL information element. BCOB-X is used for ATM transport service where AAL, traffic type and timing requirements are transparent to the network. If Transparent VP Service is specified, the user is requesting an ATM-only service from the network. This service differs from BCOB-X in that with the Transparent VP Service, both the VCI field and PT (Payload Type) field will be transported transparently by the network. It is included in the BBC (Broadband Bearer Capability) IE. This IE must be and can only be included in the SETUP message.
- Service_Type or ATC (ATM Transfer Capability): The possible values for this field are CBR, rt-VBR, nrt-VBR and ABR. It is included in the BBC IE, too.
- BEI (Best Effort Indicator): This field was discussed in Section 6.2.1.2 since it is included in the ATM Traffic Descriptor IE. It is used to indicate if the service is a UBR (best effort) service or a non-UBR (non-best effort) service. “1” means

the service is a “best effort” service and “0” implies it is “non-best effort” service.

Not all combinations of the above service category parameters are valid. The valid combinations of Bearer Class, ATC and BEI are clearly defined in Table A9-1 of the UNI 4.0 specification [3].

5.2.1.6. ABR Related Objects

The objects to describe ABR service are listed as a separate category because ATM Forum UNI 4.0 [3] added ABR capability. The nine objects are listed below.

- F_ICR (Forward ABR Initial Cell Rate)
- B_ICR (Backward ABR ICR)
- F_TBE (Forward ABR Transient Buffer Exposure Identifier)
- B_TBE (Backward ABR TBE)
- FRTT (Cumulative RM Fixed Round Trip Time)
- F_RIF (Forward Rate Increase Factor)
- B_RIF (Backward RIF)

The above seven objects are specified in the ABR Setup Parameters IE that contains mandatory parameters. UNI 4.0 also specifies an ABR Additional Parameters IE that contains optional parameters during the call/connection establishment. Both of these two IEs may be included in SETUP and CONNECT messages. The ABR Setup Parameters IE is required if the service category is ABR.

- F_MCR_CLP_01 (Forward ABR Minimum Cell Rate with CLP = 0 + 1) and B_MCR_CLP_01 (Backward ABR MCR with CLP = 0 + 1): These two parameters are traffic parameters specific to ABR traffic. They are included in the ATM Traffic Descriptor (ATD) IE.

Objects related to the Call ID, ATM Traffic, QoS, AAL and ATM Service category were discussed previously. Except for the connection ID objects, the objects described in the other four groups cannot be mixed randomly. The fields specified in the Broadband Bearer Capability (BBC) IE, the ATM Traffic Descriptor (ATC) IE, the Extended QoS

Parameters IE, the End-to-End Transmit Delay IE, and the QoS Parameter IE should be consistent. All of these IEs can appear in the SETUP message. If a SETUP message is received containing an invalid combination of BBC, ATC and BEI, the call is cleared with "*Cause #65, Bearer Capability Not Implemented*" [3]. If the combination of traffic parameters, QoS parameters, and QoS class in a SETUP message is not a valid combination for the particular ATM service category, the call is cleared with "*Cause #73, unsupported combination of traffic parameters*" [3]. The valid combination of different status fields is listed in Table A9-2 of the UNI 4.0 specification [3].

5.2.1.7. Statistical Parameters

There are three statistical parameters. None of them can be obtained from an IE. It is the vendor's responsibility to provide these statistical parameters. The descriptions are listed below.

- Cell_Received: The number of received cells since the connection is up.
- Cell_Transmitted: The number of transmitted cells since the connection is up.
- Cell_Rejected: The number of rejected cells since the connection is up.

5.2.1.8. Miscellaneous Parameters

Objects not included in other groups are classified as miscellaneous parameters. The six parameters are described below.

- Call_Status: The vendor should provide this object to describe the status of the call. The possible values can be: Call Setup, Call In-progress and Call Terminated.
- Cause: The integer value of Call Termination Cause. This field can be found in the Cause IE. The Cause IE is mandatory in STATUS, DROP PARTY, LEAF SETUP FAILURE and ADD PARTY REJECT messages. Its appearance in RELEASE and RELEASE COMPLETE messages is optional. More than 40 possible values are defined in ATM Forum UNI 3.1 [4] and 4.0 [3] specifications.

- Call_Termination_Location: This field is used to tell where the disconnection occurred. It is included in Cause IE, too. There are seven possible values defined in UNI 3.1, such as User, Private Network Serving the Local User, Public Network Serving the Local User, Transit Network, etc.
- Call_Start_Time and Call_Collect_Time: These two parameters cannot be obtained from any IEs. It is the vendor's responsibility to provide these two fields. The format of these fields can be YYYYMMDDThhmmss, where T is a field deliminator to separate date and time.
- Call_Duration: The vendor should provide this field. The unit of this field should be seconds.

5.2.2. CDR Output Format

Each CDRs output file should have a standard header with the TAB character as the field delimiter. All objects (parameters) discussed in Section 5.2.1 should be included in the header. The standard header file that includes 56 fields is shown below.

```
Call_Source_Reference  Call_Term_Reference Call_Method  Call_Type
   Call_Start_Time Call_Collect_Time Call_Duration Call_In_Port/VPI/VCI
   Call_Status Calling_Party_Address Calling_Party_Subaddress
Called_Party_Address Called_Party_Subaddress  Cells_Received
   Cells_Rejected Cells_Transmitted  F_PCR_CLP_0  F_PCR_CLP_01
   B_PCR_CLP_0 B_PCR_CLP_01  F_SCR_CLP_0  F_SCR_CLP_01 B_SCR_CLP_0
B_SCR_CLP_01  F_MBS_CLP_0  F_MBS_CLP_01 B_MBS_CLP_0 B_MBS_CLP_01
   F_QoS  B_QoS Cause  Call_Out_Port/VPI/VCI  BEI Tagging_Forward
   Tagging_Backward C_CTD  M_CTD  AF_CDV AB_CDV CF_CDV  CB_CDV AF_CLR
   AB_CLR AAL_Type  Bearer_Class Service_Type F_ICR  B_ICR  F_TBE
   B_TBE  FRTT  F_RIF  B_RIF  F_MCR_CLP_01  B_MCR_CLP_01
Call_Termination_Location
```

This standard header file was discussed in Section 4.1.3.5 and Section 4.2.2.6. For any given call, some fields must be empty. For example, for a UBR call, the objects related to ABR parameters will be empty.

5.3. Summary

This chapter recommended standards for both the generation and conveyance of call detail records and objects that should be contained by call detail records. An ideal model for CDR generation and processing is given and some commands related to CDRs are suggested. The standard CDR should include 56 objects that are distributed in eight categories. Some objects can be extracted from ATM information elements and some objects can be obtained from embedded software within the switch.

Chapter 6. Conclusions and Future Research

6.1. Summary

As part of a research project investigating network management and traffic characterization for networks with dynamically allocated resources, such as ATM, two toolkits for collecting call detail records were developed. In addition, based on requirements and the experience of understanding and collecting the CDRs, recommendations for a standard method to generate and convey CDRs and for standard CDR objects have been developed.

The two toolkits developed work with FORE Systems ASX ATM switches and IBM 8265 ATM switches. The toolkit for the FORE Systems ASX switches has been exercised using switches in the Virginia Tech campus backbone and should also work with FORE Systems switches in NET.WORK.VIRGINIA. The toolkit for the IBM 8265 ATM switch has been exercised using two switches in the CNS Interop Lab.

Since the mechanisms to generate CDRs are different for the two types of switches, the toolkit designs vary accordingly. FORE Systems switches transfer their CDRs to an external data server in real time. The toolkit for the FORE Systems switch needs to extract the successful (or interesting) CDRs from the raw data files, merge all CDR files generated from the previous stage, and generate the call duration of each successful call. The languages used to develop the first toolkit are Unix Shell, C, and Perl. The IBM 8265 switch does not automatically move CDRs to an external server, so this toolkit needs to "talk" with the switch directly as an SNMP manager and collect CDRs from the switch at some fixed interval. The main language and system used in the second toolkit is the Tcl/Scotty script language.

Based on the approaches we used to extract CDRs from IBM and FORE Systems switches, we should be able to develop other toolkits to collect CDRs data from other network elements from other vendors. Experience from developing CDR collection procedures from various switches also allowed us to develop a recommended standard for CDR generation and conveyance. Study of the CDR objects that were available allowed us to provide recommendations for standard CDR fields. We hope that this

recommendation will help vendors to consider resource management from both the switch level and the whole network level. Based on the recommended standards and the quantitative analysis of the usage distribution for each class (which will be provided in a later thesis), the network manager should be able to get better ideas about network resource utilization and requirements.

6.2. Future Work and Research

6.2.1. Data Collection from NET.WORK.VIRGINIA Backbone Switches

We have been exercising the toolkit for FORE Systems ASX ATM switches using backbone switches in the Virginia Tech campus network. We have not yet tested the toolkit on backbone switches in NET.WORK.VIRGINIA. The CDRs from these backbone switches should have the same output format, but will be produced at a much higher rate. It may be more convenient to modify the collection period in the toolkit from 24 hours to 12 hours or less. It is not difficult to re-program the toolkit codes to do this, but testing is needed. Both the size of the final data file and the collection time need to be considered. More system resources such as RAM and hard disk space, are needed to open a data file with bigger size. Another consideration for collecting CDRs from NET.WORK.VIRGINIA backbone switches is that a specific data server needs to be assigned for data collection.

6.2.2. Calculation of Standard Deviation of Received Cells

Another pending job is the calculation of the standard deviation of the quantity of received cells per minute in the toolkits. It would be helpful to add two columns in the final generated data file. The first column would be MEAN (mean value of the quantity of received cells per minute). The second column would be STD (the standard deviation of quantity of received cells per minute). The reason why the mean and standard deviation are not currently calculated is briefly discussed below.

The calculation of mean and standard deviation of the quantity of received cells per second for a specific successful call depends on the previous calculation of mean and standard deviation for this call in the previous file.

Given the five-minute collection interval, if a call is started and terminated within the collection interval, then the MEAN is the received cells divided by call duration and the STD value of this call is zero. However, most calls will last more than 5 minutes. To make the discussion less complex, we assume that a successful SVC call lasts 14 minutes and that it spans three CDR files. Note that this call spans at least three CDR files since a temporary CDR file is generated in the toolkits every five minutes.

To calculate the final mean and standard deviation of the quantity of received cells per minute for this call, we have to calculate mean and standard deviation for the second interim CDR file of this call. Similarly, we have to calculate the mean and standard deviation of the first interim file for this call. To accomplish this, the existing toolkit needs to be modified to collect CDRs of both in-progress and successful SVC calls. For a campus backbone switch in Burrus hall, with collection every five minutes, the size of a CDR file with both in-progress and successful calls included is more than 30 kilobytes. To merge large files makes the toolkit run extremely slowly. Each call is uniquely identified by the Call Connection ID (CCI). The CCI for each call is 32 bytes long. It is difficult for the toolkit described in this thesis to extract calls with same CCI in the final files and to keep track of them. Because of the above reasons, the mean and standard deviation for this call cannot be calculated by the existing toolkit. However, it is possible to calculate them with the help of advanced database system, such as Oracle, which has been shown in the ideal functional model of a CDR collection system in Figure 5-1.

To further explain the calculation of the mean and standard deviation, the related formulas to calculate the MEAN and STD values of a successful call are provided below.

$\overline{x_i}$: Mean value of the quality of received cells per unit time in the i*th* interim file;

x_i : The quantity of received cells of the i*th* interim file;

S_i^2 : Standard deviation of the quantity of received cells per unit time in the i*th* interim file;

C_i : The quantity of total received cells of until the i*th* interim file;

T_{1i} : CDR generation time of the (i -1)*th* interim file;

T_{2i} : CDR generation time of the i*th* interim file.

The initial values :

$x_1 = C_{1;}\ \overline{x_1} = C_1/(T_{21} - T_{11});\quad S_1^2 = 0.$

$$\overline{x_{k+1}} = \frac{1}{k+1}\sum_{i=1}^{k+1} x_i = \frac{1}{k+1}(k\overline{x_k} + x_{k+1}) \qquad (1)$$

where $x_{k+1} = C_{k+1} - C_k$

$$(k-1)S_k^2 = \sum_{i=1}^{k}(x_i - \overline{x_k})^2 = \sum_{i=1}^{k} x_i^2 - 2\overline{x_k}\sum_{i=1}^{k} x_i + k\overline{x_k}^2 = \sum_{i=1}^{k} x_i^2 - k\overline{x_k}^2$$

$$\Rightarrow \sum_{i=1}^{k} x_i^2 = (k-1)S_k^2 + k\overline{x_k}^2 \qquad (2)$$

$$kS_{k+1}^2 = \sum_{i=1}^{k+1} x_i^2 - (k+1)\overline{x_{k+1}}^2 = x_{k+1}^2 + \sum_{i=1}^{k} x_i^2 - (k+1)\left[\frac{1}{k+1}(k\overline{x_k} + x_{k+1})\right]^2$$

$$= x_{k+1}^2 + (k-1)S_k^2 + k\overline{x_k}^2 - \frac{1}{k+1}(k^2\overline{x_k}^2 + 2k\overline{x_k}x_{k+1} + x_{k+1}^2)$$

$$= (k-1)S_k^2 + \frac{k}{k+1}(x_{k+1} - \overline{x_k})^2 \qquad (3)$$

References

[1] J. Crowder, *et al.*, *NET.WORK.VIRGINIA*, http://www.networkvirginia.net.

[2] D. McDysan and D. Spohn, *ATM Theory and Applications*, McGraw-Hill Companies, Inc., Hightstown, NJ, 1998.

[3] ATM Forum, *UNI Signaling 4.0*, October 1995. ftp://ftp.atmforum.com/pub/.

[4] ATM Forum, *UNI 3.1 Signaling 3.1*, September 1994. ftp://ftp.atmforum.com/pub/.

[5] ATM Forum, *Traffic Management Specification, Version 4.0*, April 1996.

[6] ITU-T, *B-ISDN Application Protocols for Access Signaling (Q.2931)*, February 1995.

[7] IBM International Technical Support Organization, Raleigh Center, *Asynchronous Transfer Mode (ATM) Technical Overview*, October 1995.

[8] Christos Kolias, *The ATM Glossary*, Imark Communications, http://www.imark-com.com, March 1998.

[9] Geoff Bennett, *ATM Quality of Service, Part 2, QoS in Network Components*, FORE Systems Inc., http://academy.fore.com, 1997.

[10] Randal L. Schwards, *Learning Perl*, O'Reilly & Associates, Inc., 1993.

[11] Daniel Gilly, *Unix in a Nutshell*, O'Reilly & Associates, Inc., 1998.

[12] Fore Systems, Inc., *ForeView Call and Performance Records*, http://www.fore.com/products/4vw/crdinfo.htm.

[13] Fore Systems, Inc. *AMI Configuration Commands Reference Manual*, March 1999.

[14] Haissam Alaiwan, Rene Castel, *8265 BoilerPlate release 4.0*, IBM France, December 1998.

[15] Jürgen Schönwälder, TU Braunschweig, *Scotty - Tcl Extensions for Network Management Applications*, http://wwwhome.cs.utwente.nl/~schoenw/scotty/, June 1998.

[16] ISO8601, *Data elements and interchange formats -- Information interchange – Representation of dates and times*, 1988.

[17] The Open Group, *The Single UNIX ® Specification, Version 2*, http://www.opengroup.org/onlinepubs/7908799/xsh/mktime.html, 1997.

[18] Free Software Foundation (FSF), *The GCC Compiler*, http://www.fsf.org/software/gcc/gcc.html, 1998.

[19] W. Richard Stevens, *TCP/IP Illustrated Volume 1*, Addison-Wesley Longman, Inc., 1997.

[20] IBM, *8265 MIB Data File Version 4.12*, http://www.networking.ibm.com, 1999.

[21] ITU-T, *B-ISDN ATM Layer Cell Transfer Performance (I.356)*, October 1996.

[22] ITU-T, *B-ISDN ATM AAL Functional Description (I.362)*, March, 1993.

[23] Sidnie M.Feit, *SNMP: A Guide to Network Management*, McGraw-Hill, Inc., 1995.

[24] D. Perkins and E. McGinnis, *Understanding SNMP MIBs*, Prentice Hall PTR, 1997.

[25] D. Zeltserman and G. Puoplo, *Building Network Management Tools with TCL/TK*, Prentice Hall PTR, 1998.

Appendix A: Acronyms

AAL: ATM Adaptation Layer

ABR: Available Bit Rate

ATC: ATM Transfer Capability

ATD: ATM Traffic Descriptor

ATM: Asynchronous Transfer Mode

BBC: Broadband Bearer Capability

BCOB: Broadband Connection Oriented Bearer

BEI: Best Effort Indicator

B-ISDN: Broadband Integrated Service Data Network

CAC: Connection admission Control

CBR: Constant Bit Rate

CCI: Call Connection ID

CDR: Call Detail Record

CDVT: Cell Delay Variation Tolerance

CLP: Cell Loss Priority

CLR: Cell Loss Rate

CMR: Cell Misinsertion Rate

CRC: Cyclic Redundancy Check

CTD: Cell Transfer Delay

EFCI: Explicit Forward Congestion Indication

EPD: Early Packet Discard

FDDI: Fiber Distributed Data Interface

FRTT: Fixed Round-Trip Time

FT1: Fractional T1

GCRA: Generic Cell Rate Algorithm

GFC: Generic Flow Control

HDLC: High level Data Link Control

HEC: Header Error Check

ICR: Initial Cell Rate

IE: Information Element

ILMI: Interim Local Management Interface

IP: Internet Protocol

ISO: International Standards Organization

ITU: International Telecommunications Union

LAN: Local Area Network

LANE: LAN Emulation

MBS: Maximum Burst Size

MCR: Minimum Cell Rate

MIB: Management Information Base

NNI: Network-Network Interface

nrt-VBR: Non-real-time Variable Bit Rate

OAM: Operations, Administration and Maintenance

OC: Optical Carrier

PCR: Peak Cell Rate

PLCP: Physical Layer Convergence Protocol

PNNI: Private Network-Network Interface

PPD: Partial Packet Discard

PT: Payload Type

PVC: Permanent Virtual Circuit

QoS: Quality of Service

RIF: Rate Increase Factor

RM: Resource Management

rt-VBR: Real-time Variable Bit Rate

SCR: Sustainable Cell Rate

SEAL: Simple and Efficient Adaptation Layer

SMDS: Switched Multi-Megabit Data Services

SNMP: Simple Network Management Protocol

SONET: Synchronous Optical Network

SPVC: Smart Permanent Virtual Circuit

SPVP: Smart Permanent Virtual Path

STD: Standard Deviation

SVC: Switched Virtual Circuit

TBE: Transient Buffer Exposure

TC: Transmission Convergence

TDM: Time Division Multiplex

UBR: Unspecified Bit Rate

UNI: User-Network Interface

UPC: Usage Parameter Control

VBR: Variable Bit Rate

VCI: Virtual Channel Identifier

VPI: Virtual Path Identifier

VPN: Virtual Private Network

WAN: Wide Area Network

Appendix B: User's Guide for the Toolkits

I. Fore Systems ASX Toolkit User's Guide

STEP 1. Login at server as *wroger*.

```
UNIX(r) System V Release 4.0 (server)

login: wroger
Password:
Last login: Tue May 25 23:16:23 from as5300-5.sl016.c
Sun Microsystems Inc.   SunOS 5.5   Generic November 1995

This is server.  Unauthorized access or use is prohibited.

You have new mail.
```

STEP 2. Change directory to */batch*.

```
server >cd batch
server/batch >
```

STEP 3. Run CollCate.

```
server/batch >CollCate

WELCOME TO USE The DATA Collection Utility
Today Is:
Wed May 26 10:55:48 EDT 1999

Please enter the YEAR(XXXX) when you collect the data

1999                              ; The year

YEAR =  1999
Enter the DATE when the data was collected
The format of the input should be 4-digit Numbers: ####
The initial two #s are from 01-12(month); The last two #s are
from 01-31(day)

0525                              ; The date

DATE = 0525
Enter the HOUR(00-23) when you collect the data
Usually  the beginning value should be set as 00
If you don't begin with 00, just type the double-digit number
like 04, 11,etc
```

00 ***; The starting hour***

```
HOUR = 00
Enter the MINUTE (00/05/10,,,/55) when you collect the data
Usually the beginning value should be set as 00
If you don't start at 00, type 05, 10, 15, ...,50, 55
```

00 ***; The starting minute***

```
MINUTE = 00

You Are Going To Collect Data From One of The
Following Six Buildings:

BUR-Burrus Hall
CAS-Cassell Colliseum
ISB -Information System Building
OWE -Owens Hall
SHA -Shanks Hall
WHI -Whittemore Hall

Please Input The Building Name Where Your Switch Is:
############################################################
Make Sure The Building Name You Input is CAPITAL LETTERS
############################################################
```

WHI ***; The building name***

```
BUILDING = WHI
You can collect data from following switches:
############

WHI-287C-BX-1
WHI-287C-BX-2

############
Please input the fabric number ( 1 / 2 ):
```

1 ***; The fabric number***

```
You are going to collect data from WHI-287C-BX-1.

HOUR = 00
MINUTE = 00
input file = /kibitz/cdr/VT/WHI-287C-BX-1/199905250000_05.cc
MINUTE = 5
input file = /kibitz/cdr/VT/WHI-287C-BX-1/199905250005_05.cc
MINUTE = 10
input file = /kibitz/cdr/VT/WHI-287C-BX-1/199905250010_05.cc
MINUTE = 15
```

```
input file = /kibitz/cdr/VT/WHI-287C-BX-1/199905250015_05.cc

. . .      . . .

HOUR = 23
. . .      . . .
MINUTE = 50
input file = /kibitz/cdr/VT/WHI-287C-BX-1/199905252350_05.cc
MINUTE = 55
input file = /kibitz/cdr/VT/WHI-287C-BX-1/199905252355_05.cc
HOUR =  24
The Data Collection is Done

Starting File Catenation...

The HOUR is set as 00 at the beginning
The MINUTE is set as 00 at the beginning
MINUTE = 00
INPUT_FILE =  /test/TEMPWHI-287C-BX-105250000
MINUTE = 5
INPUT_FILE =  /test/TEMPWHI-287C-BX-105250005
MINUTE = 10
INPUT_FILE =  /test/TEMPWHI-287C-BX-105250010
MINUTE = 15
INPUT_FILE =  /test/TEMPWHI-287C-BX-105250015

. . .      . . .

HOUR = 23
. . .      . . .
MINUTE = 40
INPUT_FILE =  /test/TEMPWHI-287C-BX-105252340
MINUTE = 45
INPUT_FILE =  /test/TEMPWHI-287C-BX-105252345
MINUTE = 50
INPUT_FILE =  /test/TEMPWHI-287C-BX-105252350
MINUTE = 55
INPUT_FILE =  /test/TEMPWHI-287C-BX-105252355
HOUR =  24
The File Catenation is Done
Waiting Please ...
You need to further process the generated file:
 /finaldata/WHI-287C-BX-1final10525
```

STEP 4. Run SubsComma.

```
 server /batch >SubsComma

WELCOME!!
```

```
 You are going to process a special file.
To substitute a string in this file with TAB

The current directory has been changed to:
    /finaldata

Hint: The possible input file name could be
   SubDirectoryfinal1DATE under current directory, such as:
   WHI-287C-BX-1final10507 or any other FILE existed.

SubDirectory is any directory name under /kibitz/cdr/VT.

Make sure you input the correct file name

input file name: WHI-287C-BX-1final10525 ; The file name generated in step 3

If you are using SubDirectoryfinal1DATE as input,
the output file should have name: SubDirectoryfinal2DATE.
        Such as: WHI-287C-BX-1final20507

Output File Name: WHI-287C-BX-1final20525
                                          ; The output file name as requested


The output file is: WHI-287C-BX-1final20525.
  Correct(Y/N)? y                 ;Yes

HINT: For CDR data collection, you need to substitute
chracter ',' by the TAB

search String: ,            ; Character "," is the only choice

You are going to substitute , in WHI-287C-BX-1final10525 with TAB

The string substitution is done

The final output file is WHI-287C-BX-1final20525
```

STEP 5. Run FinalGenerate

```
 server /batch >FinalGenerate
Please enter the YEAR(XXXX) when you collected the data
1999
YEAR = 1999                 ;The year
Enter the DATE when you collected the data
The format of the input should be 4-digit Numbers: ####
The initial two #s are from 01-12(month); The last two #s are
from 01-31(day)
```

0525 ***; The date***

```
DATE= 0525
You Are Going To Collect Data From One of The Following Six
Buildings:

BUR-Burrus Hall
CAS-Cassell Colliseum
ISB -Information System Building
OWE -Owens Hall
SHA -Shanks Hall
WHI -Whittemore Hall

Please Input The Building Name Where Your Switch Is:
############################################################
Make Sure The Building Name You Input is CAPITAL LETTERS
############################################################
```

WHI ***; The building name***

```
BUILDING = WHI
You can collect data from following switches:
############

WHI-287C-BX-1
WHI-287C-BX-2

############
Please input the fabric number ( 1 / 2 ):
```

1 ***; The fabric number***

```
You are going to collect data from WHI-287C-BX-1.

INPUTFILE =  /finaldata/WHI-287C-BX-1final20525
please wait...

File Columns cut is done. The new generated file is
 /finaldata/WHI-287C-BX-1final30525

Starting generating call_duration...

Waiting please ...
Time Duration calculation is finished
The new generated file is  /finaldata/WHI-287C-BX-1final40525

Starting pasting...

Waiting please ...
File paste is finished
The new generated file is  /finaldata/WHI-287C-BX-1final50525.
```

```
Final date file generating...

Waiting please ...
HEADER CATENATION is finished

The final generated file is:
  /finaldata/WHI-287C-BX-1final19990525

This file is ready for SAS Analysis.


Do you want to standardize the final file? (y/n)
y

The file with standard header is:
          /finaldata/WHI-287C-BX-1Standard19990525

 server /batch >
```

To collect CDRs from *Whittemore* or *ISB inter-op* lab, STEP 3 is not necessary since *CollCate* for these switches will be automatically run every day.

II. IBM 8265 Toolkit User's Guide

STEP 1. Login at *server* as *wroger*. (same as step 1 for Toolkit 1)

STEP 2. Change directory to *home/wroger/tcl*.

```
 server  >cd tcl

 server /tcl >ls
8265CxVclEntry       8265vcXConnectEntry  snmp
8265SvcEntry         SubsBrace
8265SvcLogEntry      exercise
```

STEP 3. Change directory to the "Entry" sub-directory where you want to collect CDRs.

```
 server /tcl >cd 8265SvcEntry
```

; We use 8265SvcEntry in the example

STEP 4. Run interim file generation code.

```
 server /tcl/8265SvcEntry >8265SvcEntryGenerate

Enter the ipAddress of IBM 8265 Switch:
198.82.250.90
ipAddress =  198.82.250.90
```

```
Waiting Please...

The data file SII1 is generated. You may further process it
Waiting Please...

The data file SSV2 is generated. You may further process it
Waiting Please...

The data file SSV3 is generated. You may further process it
Waiting Please...

The data file SCR4 is generated. You may further process it
Waiting Please...

The data file SEPR5 is generated. You may further process it
Waiting Please...

The data file SCN6 is generated. You may further process it
Waiting Please...

The data file SCN7 is generated. You may further process it
Waiting Please...

The data file SC8 is generated. You may further process it
Waiting Please...

The data file SCT9 is generated. You may further process it
Waiting Please...

The data file SV10 is generated. You may further process it
Waiting Please...

The data file SV11 is generated. You may further process it


File 8265SvcEntryInterim is generated

Please run SubsBrace to generate the final CDR file for this
switch
```

STEP 5. Run SubsBrace.

```
 server /tcl/8265SvcEntry >SubsBrace

        WELCOME!!

You are going to process a special file. To substitute a string
in this file with space.
input file name: 8265SvcEntryInterim
```
;Interim file generated in step 4

```
If you are using 8265CxVclEntry or 8265SvcEntry or 8265SvcLog or
8265vcXConnectEntry +Interim as input file,

It is strongly recommended that the output file should have
following name format:


8265CxVclEntry or 8265SvcEntry or 8265SvcLog or
8265vcXConnectEntry + Today's date

Output File Name: 8265SvcEntry20526
```

;The final file name. There are two 8265 switches. This one is regarded as the second one by the researcher so "2" is added before the date "0526"

```
The output file is: 8265SvcEntry20526. Correct(Y/N)?y  ;Yes

HINT: For IBM CDR data collection, you need to substitute } by
the white space.

search String: }                          ; "}" is the only choice

You are going to substitute } in 8265SvcEntryInterim with SPACE
The string substitution is done

The final output file is 8265SvcEntry20526
```

Note:

If the file *wroger.crontab* under */batch* includes following lines:

```
00 * * * *  /tcl/8265SvcEntry/Generate24682
05 * * * *  /tcl/8265SvcEntry/Generate24682
10 * * * *  /tcl/8265SvcEntry/Generate24682
15 * * * *  /tcl/8265SvcEntry/Generate24682
20 * * * *  /tcl/8265SvcEntry/Generate24682
25 * * * *  /tcl/8265SvcEntry/Generate24682
30 * * * *  /tcl/8265SvcEntry/Generate24682
35 * * * *  /tcl/8265SvcEntry/Generate24682
40 * * * *  /tcl/8265SvcEntry/Generate24682
45 * * * *  /tcl/8265SvcEntry/Generate24682
50 * * * *  /tcl/8265SvcEntry/Generate24682
55 * * * *  /tcl/8265SvcEntry/Generate24682
```

Then step 4 and step 5 will not be necessary if the user is going to collect data from switch 246. However, the user still needs to check */tcl/8265SvcEntry/TempDirectory* to make sure the required files are not empty. If these files exist, then the user can run FinalGenIBM under /tcl/8265SvcEntry. The result is shown below.

```
 server /tcl/8265SvcEntry >FinalGenIBM
WELCOME TO USE The DATA Collection Utility
Today Is:
Mon Jun 14 02:38:49 EDT 1999


Please enter the YEAR(XXXX) when you collect the data:
1999                          ;  The year inputted

YEAR =  1999

Enter the DATE when the data was collected
The format of the input should be 4-digit Numbers: ####
The initial two #s are from 01-12(month); The last two #s are
from 01-31(day).
0608                          ; the date inputted
DATE= 0608

You Are Going To Collect Data From One of The following IBM
switches:
Following IP Addresses are available:

**.**.246.82
**.**.250.90    ; Here the ** is used instead of the real value for security

Please Input 246 for the first switch OR 250 for the second
Switch :
246                         ;The value inputted
Starting File Catenation...

The HOUR is set as 00 at the beginning
The MINUTE is set as 00 at the beginning
MINUTE = 00
INPUT_FILE =
/tcl/8265SvcEntry/TempDirectory/TEMP246SvcEntry06080000
MINUTE = 5
INPUT_FILE =  /tcl/8265SvcEntry/TempDirectory/TEMP246SvcEntry06080005
MINUTE = 10
INPUT_FILE =  /tcl/8265SvcEntry/TempDirectory/TEMP246SvcEntry06080010
MINUTE = 15
INPUT_FILE =  /tcl/8265SvcEntry/TempDirectory/TEMP246SvcEntry0608001

....
....


MINUTE = 45
```

```
INPUT_FILE =  /tcl/8265SvcEntry/TempDirectory/TEMP246SvcEntry06082345
MINUTE = 50
INPUT_FILE =  /tcl/8265SvcEntry/TempDirectory/TEMP246SvcEntry06082350
MINUTE = 55
INPUT_FILE =  /tcl/8265SvcEntry/TempDirectory/TEMP246SvcEntry06082355
HOUR =  24
The File Catenation is Done
Waiting Please ...
further processing the generated file:
 /tcl/8265SvcEntry/finaldata/246final10608
Waiting Please ...

 /tcl/8265SvcEntry/finaldata/246final20608 is generated

You are going to change TempData5 to TempData6

The current directory is:  /tcl/8265SvcEntry/finaldata.
The file exists. Waiting Please...
The string substitution is done

The final output file is TempData6 under
/tcl/8265SvcEntry/finaldata

TempData7 has been generated with CallDuration included


The string substitution is done

The files temp1out and temp2out are generated
The final file is:  /tcl/8265SvcEntry/finaldata/246final0608

 server /tcl/8265SvcEntry >
```

Appendix C: Classification of Call Detail Records

The following tables list the categories of parameters that are collected by the toolkits introduced in Chapter 3 and Chapter 4. They are available for use by high-level analysis packages.

There are five categories of parameters.

- Call identification parameters, listed in Table C-1, that identify the connection type, the switch's logical port number, the virtual path identifier (VPI), and virtual channel identifier (VCI).
- Traffic parameters, listed in Table C-2, that are used for statistical analysis.
- Class of service parameters, listed in Table C-3, that distinguish various classes of service.
- Statistical parameters, listed in Table C-4, which may be useful for statistical analysis.
- Miscellaneous parameters, listed in Table C-5, including time parameters and the "call status" parameter.

The toolkit modifies the time format of time parameters from ISO 8601 format to the Unix epoch time format, as discussed in Chapter 4, to facilitate the calculation of call duration. The call status is used as a filter as mentioned in Chapter 4.

Table C-1. Call Identification Parameters

Parameters	Explanation
Call Reference	The Q.2931 call reference value for this SVC
Calling and Called Number	NSAP (Network Service Access Point) address of calling and called party
Call Origination Method	Point-to-point, point-to-multipoint, multipoint-to-point or multipoint-to-multipoint
Call Type	PVC, PVP, Q.2931 SVC or PNNI SPVC
Call Source / Destination Port ID, VPI and VCI	The switch's logical port where the call enters or leaves and the value of VPI/VCI

Table C-2. Traffic Parameters

Parameters	Explanation
PCR	Peak cell rate
SCR	Sustained cell rate
MBS	Maximum burst size

Table C-3. Class of Service Parameters

Parameters	Explanation
Call Best Effort Indication	A binary bit. "1" means the call is a "best effort" call and "0" means the call is not a "best-effort" call.
Call Forward/Backward QoS	0: UBR; 1: CBR; 2: rt-VBR; 3: nrt-VBR or ABR; 4: nrt-VBR

Table C-4. Statistical Parameters

Parameters	Explanation
Cell Received	Cumulative count of the received cells
Cell Rejected	Cumulative count of the rejected cells
Cell Transmitted	Cumulative count of the transmitted cells

Table C-5. Miscellaneous Parameters

Parameters	Explanation
Call Start Time	Time when the call is initiated
Call Collect Time	Time when the call is cleared
Call Duration	Duration of the call
Call Status	Current status of the call

Vita

Xianrui Roger Wang graduated from Department of Computer Science and Engineering, Huazhong University of Science and Technology, Wuhan, China in July 1988. The degree he received there was Bachelor of Engineering. Before studying in Virginia Tech in 1997, he had more than 8 years' working experience in three different companies. His last job title in China was network consultant in Timeplex Group Beijing Office. He has interest in working as a network and system engineer. He will be employed by Comrise Technology Inc. working as a system engineer for AT&T, New Jersey.