UNIVERSITY OF ALBERTA

Visualization of ATM Network Traffic

BY

Feng Luo ©

A thesis submitted to the Faculty of Graduate Studies and Research in partial fulfillment of the requirements for the degree of Masters of Science.

DEPARTMENT OF COMPUTING SCIENCE

Edmonton, Alberta
Spring 1995

ISBN 0-612-01627-7

UNIVERSITY OF ALBERTA

RELEASE FORM

NAME OF AUTHOR: Feng Luo

TITLE OF THESIS: Visualization of ATM Network Traffic

DEGREE: Masters of Science

YEAR THIS DEGREE GRANTED: 1995

(Signed) [illegible] . . . .

Feng Luo
Suite 203, 10035 - 83 Avenue
Edmonton, Alberta Canada, T6E 2C3

Date: April 18th

"The purpose of computing is insight, not numbers"
by Richard Hamming

UNIVERSITY OF ALBERTA

FACULTY OF GRADUATE STUDIES AND RESEARCH

The undersigned certify that they have read, and recommend to the Faculty of Graduate Studies and Research for acceptance, a thesis entitled **Visualization of ATM Network Traffic** submitted by Feng Luo in partial fulfillment of the requirements for the degree of Masters of Science.

[illegible]
Prof. Mark Green (Supervisor)

[illegible]
Prof. Peter Bartl (External)

J. Harms
Prof. Janelle Harms (Examiner)

Prof. William Armstrong (Chair)

Date: April 12, 1995

To my husband, Guanghong Zhang
and my parents

Abstract

Visualization is a key technology for understanding complex, data-rich systems. Effective visualizations make important features of the data immediately recognizable. In this thesis, we investigated visualization techniques for large collections of ATM cells or PDUs (Protocol Data Unit) in high-speed ATM networks.

The speed of an ATM network can be as high as giga bits per second. In such a high-speed network, tens or even hundreds of thousand of communication events occur every second. Traditional text displays create up to a million lines of information for seconds of collected high speed network traffic data. It becomes critical to provide users with high level visualization tools that enable them to explore, manipulate, and relate large information spaces to their interests in an interactive way.

Acknowledgements

I would like to thank, first of all, my supervisor, Professor Mark Green, for his guidance and support throughout this project. I would also like to thank my examining committee members, Professor Peter Bartl and Dr. Janelle Harms, for their careful review of the thesis.

Thanks goes to HP/IDACOM for giving me this opportunity to work in this fantastic area. Extended thanks goes to Bill Mortimer, his management team and engineers for giving us a number of very timely progress reviews and technical guidance on issues related to ATM networks. My deep thanks to Jane Tilley who worked with me as a graphic designer in this project. Without her creative ideas and cooperation, this thesis would not have been possible. Jane, you are a wonderful person to work with. Thanks also go to Lloyd White and Robert Lake for creating a nice work environment in Graphics lab and maintaining good running facilities. I would like to acknowledge Chris Shaw for his enthusiasm for my work and useful feedback.

My last, but not least, thanks goes to my husband, Guanghong Zhang, whom this thesis is dedicated to. I would not be able to concentrate on my work without his love and support. Thanks for spenting numerous hours reading my thesis and correcting my writing and coming up with constructive suggestions.

Contents

List of Figures

List of Symbols

AAL	ATM Adaption Layer
ATM	Asynchronous Transfer Mode
BISDN	Broadband Integrated Services Digital Network
BSTS	Broadband Series Testing System
CBR	Constant Bit Rate
CCITT	Consultative Committee on International Telephone and Telegraph
CDV	Cell Delay Variation
CLNAP	ConnectionLess Network Access Protocol
CLP	Cell Loss Priority
CLR	Cell Loss Ratio
CMY	Cyan, Magenta, and Yellow
CRC	Cyclick Redundancy Check
CPCS	Common Part Convergence Sublayer
CS	Convergence Sublayer
CS_PDU	Convergence Sublayer Protocol Data Uit
GFC	General Flow Control
HEC	Header Error Correction
HSV	Hue, Saturation, and Value
IE	Information Elements
ISDN	Integrated Services Digital Network
LAN	Local Area Network

NNI	Netowrk-Network Interface
UNI	User-Network Interface
OAM	Operations, Administration, and Management
OSF	Open System Fundations
PDU	Protocol Data Unit
PT	Payload Type
PVC	Permanent Virtual Connection
QoS	Quality of Services
RGB	Red, Green, and Blue
SAR	Segmentation And Re-assembly
SAR_PDU	Segmentation And Re-assembly Protocol Data Unit
SDU	Service Data Unit
SMDS	Switched Multimegabit Digital Service
SN	Sequence Number
SONET	Synchronous Optical NETwork
SUT	System Under Test
SVC	Switched Virtual Connection
VBR	Variable Bit Rate
VC	Virtual Channel
VCC	Virtual Channel Connection
VCI	Virtual Channel Identifier
VP	Virtual Path
VPC	Virtual Path Connection
VPI	Virtual Path Identifier

Chapter 1

Introduction

1.1 Introduction

Visualization has been the cornerstone of scientific progress throughout history. Virtually all comprehension in science, technology, and even art calls on our ability to visualize. In fact, the ability to visualize is almost synonymous with understanding. We have all used the expression "I see" to mean "I understand".

The objective of scientific visualization is to explore data and information in such a way that promotes new dimensions of insight. The power of visualization is due largely to the strength of human perception [23]. Our visual system is the highest capacity data channel to our brain if we think of our five senses as data input channels. Empirical studies show that the bandwidth of the human visual channel is approximately 42 million bits per second [1]. However, most humans can read at between 600 and 1200 words per minute, that is 400 to 800 bits per second. This means that the spoken or written word is not the most efficient way to make use of the tremendous data capacity of the human visual system.

Graphical representation of information is common in daily life. It can be readily used to represent anything in a real world with apparent geometric relations. However, when it comes to the scientific world, dealing with highly abstract statistical data and information flow, very limited visualization techniques have been developed. Most of these techniques in scientific visualization deal with objects that possess spatial geometric relations giving researchers a head-start in developing associated graphical representations. The visualization of non-spatial data, often called (scientific) information visualization, has received relatively less attention, since there is no obvious way to convert the data into a graphical representation. Information visualization is therefore more difficult. Visualization of telecommunication data transmitted over ATM networks certainly falls into this category.

With the explosive growth of computer networks and the rapid evolution of telecommunication technology, network capacity and data transmission speed increase dramatically. This results in large volumes of network data/events taking place every second. Analyzing such data is difficult because of its large volume. Little attention has been paid so far to visualization techniques for such data other than simply displaying data and events in a linear, line by line text format. In light of the growing importance of network management and analysis, new visualization tools for network data are a must.

1.2 Problem Statement

Text based information visualization has been used by all commercial protocol analyzers in narrowband applications. In such applications, for example, WAN protocol X.25 and narrowband ISDN basic rate (2B+D), normally there is on the order of a hundred of events happening per second. Due to the low speed of the physical port in such networks, the application processor can be engineered to display both live data and data collected in a capture buffer. Interesting spots can be located via screen scrolling, either line by line, or page by page. The problem of using text based visualization to analyze network data is not so obvious. However, when it comes to broadband application, there are typically hundreds of thousand of events occurring every second over the communication line. Using text based visualization technique, it is very difficult, if not impossible, to visualize millions of lines of data.

This thesis is based on a visualization project for the HP 75000 Broadband Series Test System (BSTS). This research project was motivated by the problem of visualizing ATM network traffic data collected by HP BSTS. This protocol testing system is used to test and measure the performance of devices connected to an ATM network. In the testing process, the system can collect up to 8 Mbytes of data, representing 131,271 ATM cells. The technique currently used in HP BSTS to represent communication data flow and incoming/outgoing events is still text based. Each cell/event is decoded and displayed linearly by several lines of text. The interface to the user is a 9 inch scrolled text window. Snapshots of live data are collected in capture RAM, having a 8 Mbytes capacity, for playback and later analysis. The task of users to analyze playback data is difficult because of the volume of data. Even though the HP BSTS provides the user with a number of filter functions to pre-filter captured data and reduce the amount of data to scroll through in the text window, there can still be thousands of lines left for detailed analysis. Users need to scroll the screen line-by-line or page-by-page (cell-by-cell) to search for interesting spots.

There is no doubt that text based visualization is ill-suited to high speed networks. First of all, with the large volume of data, it is difficult for users to gain insight from line oriented statistics. It is hard for a user to go through hundred of thousands lines of text to find an interesting spot. Secondly, listings of text lines do not provide the user with a perspective view of overall information. Most importantly, text based information fails to provide the user with the underlying structure of data, preventing the user from detecting interesting patterns. The consequence of the text based visualization technique is that the collected data is underutilized, since it is hard to analyze.

1.3 Thesis Objective

The aim of this thesis is to develop graphical visualization techniques that are used to visualize large volumes of non-spatial data and to assist the user in detecting errors and performance problems in the data collected by HP BSTS.

Our goal is to investigate techniques that illustrate the information, provide an overall view of the data collected, and provide more structured information per screen. Our techniques should not only provide the user with ways of navigating through the large data collections, searching for the data/events that are interesting to them, but also reveal the underlying data patterns and help the user to detect problems and communication anomalies developed from the collected network traffic data.

We also need to develop mappings from the data values to graphical properties, such as position, size, and color, that can be displayed graphically. Finally an insightful visualization tool will be developed as a prototype.

1.4 Outline of Thesis

The thesis begins with an introduction to the visualization problem for the HP BSTS and goals to be achieved by this thesis. An overview of previous work in the area of information visualization, and techniques that are used for graphically representing large volumes of statistical data in scientific visualization are described. Then, a brief background on the ATM networks and ATM testing issues are given. The role and functionalities of HP 75000 BSTS in B-ISDN/ATM networks are also briefly mentioned. We then focus on describing techniqes we propose to visualize communication data/events collected by HP BSTS from ATM networks. Possible extensions of our techniques to applications in other visualization areas are briefly mentioned. Implementation details of our visualization techniques are presented in the following chapter. Finally, we conclude with evaluation of our techniques, and discussion of ideas for future work in next phase of this project.

Chapter 2

Background

Most scientific visualization techniques deal with continuous data with multivariate fields that vary over space and time. Such visualization problems are relatively simple, since the data has an explicit spatial structure to be displayed. In this literature review, we aim to study the techniques used to visualize abstract information and non-spatial statistical data, that is, data with no explicit spatial properties.

2.1 Introduction

Information visualization has many applications. Large layouts, data listings, diagrams and images occur in many areas of computer science. Users have to view all of these huge collections of data through windows, since many applications demand its users to retrieve, manipulate, and understand large amounts of information.

The large volume nature of the collected data is impossible to display in detail on a typical computer screen. The most common solution to this problem is to provide the user with a scrollable viewport. As a result, the user often feels lost and deprived of context, since it shows only the region currently visible through the viewport. People naturally perceive the world using both local detail and global context. Biologically, our eyes see detail in only a small focused region. Yet we remain constantly aware of the global context through our peripheral vision and by glancing around. Our eyes rely heavily on global context to orient ourselves and to understand local detail. Therefore, the availability of adequate context is crucial, and it may improve the usability of the data significantly.

One early graphical technique, called pan and zoom [5], was design to allow the users to see both detail and context of the information display. The users can pan or scroll a window across a virtual canvas, and adjust the scale of their view through zooming. The problem with this technique is that when users zoom in for details, the context is lost. When they zoom out for orientation, there is not enough detail to do any real analysis.

Another common strategy for providing both detail and context is to show detail and layouts using two separate windows. One window contains an overview of the entire structure, while the other shows a detailed view of a selected region. The overview typically contains an indication of the detailed view's location by a small box or a point. The technique is simple and effective in many situations. However, it requires extra space for the overview window by providing two separate windows, and forces the user to mentally integrate detail and context.

An alternative to the above approaches is to distort the layout so that both detail and context can be displayed within a single window. Recent advances in information visualization have acknowledged the importance of integrating local detail with global context into a single view. A number of techniques relying on this strategy have been proposed in the last few years [10, 19, 17, 13, 16, 20]. These techniques distort the layout so that both detail and context can be displayed within a single window. They allow users to choose a subset of information items and show them in detail, and at the same time provide context by displaying the remaining items in successively less detail. Most of the visualization techniques that we review in this survey belong to this category.

2.2 Related Work

2.2.1 2D

- Fisheye Views

  Fisheye views, proposed by Furnas in [10] and extended later by Sarkar and Brown in [19], is a technique used to visualize large 2D data structure. The technique is based on an analogy of a wide angle lens, or "fisheye" lens. Fisheye lens, used in photography, can show places nearby in great detail while showing surrounding regions in successively less detail. The motivation of this technique is to provide a balance of local detail and global context.

  The technique uses a function to assign each point in the structure a number, telling how far that point is from the current focus. For any given focus, the function assigns this Dot of Interest (DOI) number to each point on the display. The display of a point depend on the number it is assigned, and only those points having a value greater than a certain threshold will be displayed. The size of a point in the display is determined by the current focus and its distance to it. A display of any size, for example N, can then be made by simply showing the N most "interesting points", as indicated by the Dot Of Interest function.

  Fisheye view can be used to browse a large layout by clicking and dragging. It can enlarge the regions near the focus in real time, and animate the display smoothly as the user changes focus. One problem with this technique is that its focus is always a single information item or point, and the system can not treat an arbitrary region or an arbitrary set of information items as the focus. Moreover, since a "Fisheye view" display is generated using functions that are thresholded to determine the the contents of the display, each point on the layout is either displayed or omitted. This causes the visualization to have gaps. One of the gaps in the current display may happen to be a desired destination. When the user moves from one focus to another, some items from the display may suddenly disappear into gaps. This may cause confusion.

- Rubber Sheet

  Sarkar et al [20] proposed a rubber sheet metaphor to visualize arbitrary 2D layouts or images. With the Rubber Sheet technique, users can select and enlarge different areas of the rubber sheet by holding and stretching it. As the user stretches an area, a greater level of detail in the selected area can be seen.

  Rubber Sheet is another technique for visualizing large graphical layouts and integrating both details and context into a single window. It differs from the Fisheye view as follows: it allows the user to specify the size of the exact focus, and support for multiple focus visualization. Uniform scaling at the focus is used to preserve spatial properties of the layout, making this technique attractive, especially for viewing spatially sensitive layouts.

- Value Bar

  Value bar [7] is an information visualization and navigation tool for linearly oriented multi-attribute data. The motivation is to solve the problem of visualizing large information listings where each item has more than one attribute to compare. This technique is useful to analyze multi-attribute listings, where a particular sorting order is maintained between items and analysis of the percentage of items within each attribute is important.

  Value bar is a thin vertical strip attached to a text window placed next to the scrollbar. There may be several value bars added to a visualizing window, each of them maps a specific quantifiable attribute shared by all items. Each item's weight is converted to a height in the value bar proportional to its part of the total weight of all items. Items represented in the value bar provide a global view of the distribution of an attribute's value in the listings.

  One important advantage that the value bar has is its ability to provide an overview of the distribution of attribute values in a single view. In this sense, this technique is a variant of the fisheye view. Users are able to compare an item's attribute value to other items in the same value bar without the need to scroll or rearrange the sorting order many times.

- Variable zoom

  Variable Zoom is an extension of Furnas's fisheye view lens. It deals with hierarchically clustered graphs and supports multiple areas of detail or focus [21].

  Variable Zoom works on 2D connected graphs, and assumes that a hierarchical clustering of nodes has been superimposed on the graph. The nodes in the connected graph are clustered into several hierarchical levels. The root cluster of the hierarchy contains several smaller clusters and each of them may contain other clusters. At each level above the node level, the clusters are represented as icons that may be opened to show the next level down. The hierarchy is used by a visualization algorithm to allow clusters in a network to be viewed at different levels of hierarchical detail.

- TreeViz

  TreeViz [12] is treemap visualization technique used to visualize hierarchical information structures. It maps information hierarchies onto a rectangular region in a space-filling manner. Treemap partitions the display space into a collection of rectangular bounding boxes representing the tree structure. The size of each node is drawn based on the magnitude of its weight relative to the total weight of the tree. Because of the efficient use of space, it can display large hierarchies per screen, and is suitable for visualizing large data structures.

- Multiple Window Visualization

  Painting Multiple Views [14] and Focusing and Link [6] are two similar visualization techniques that directly paints and manipulates large collections of data in separate windows. Both techniques present large complex information in several separate and easy-to-comprehend pictures, with each focus clearly on certain aspects of the data. The purpose is to solve the problem of visual overload, which often occur in data-rich visualizations.

Both techniques link multiple views to help the user to integrate scattered information and gain understanding over the large and complicated data sets. In Painting Multiple Views, linking is done by painting corresponding parts of the data in different views with the same color. In Focusing and Linking, straight lines are used to link multiple views together.

- Code Visualization

A difficult problem in software engineering is understanding statistics collected about source code. Software statistics includes information such as who wrote each line, when it was last modified, whether it fixes a bug or adds new functionality, how often it is executed, and so on.

In [9, 8], Eick et al present a remarkable visualization technique, called SeeSoft, to analyze software source files. Their approach is to think of source code files and lines as entities in an ordered database. In order to view a large volume of data on a single screen, they use a compact representation of source code lines. Files are displayed as columns and lines of code are displayed as thin rows. The height of each column tells the user how large each file is. The row representation shows the indentation and length of each line of the actual code. The rows are just large enough so the user can identify block comments, functions, and control structures such as if and case. The visual impression is that of a miniature picture of all of the source code with the indentation showing the usual C control structure. The color of rows is used to code the software statistics, such as the age of the code, the time it was last modified, etc..

Interaction graphics and direct manipulation are used in their user interface. This updates the screen in real-time in response to mouse actions. As the user moves the mouse around the screen, the entity under the mouse is automatically activated and the corresponding color and lines turned on. When the mouse is moved away from an entity it is automatically deactivated. This makes it easy for the user to probe and find interesting patterns. To allow the user to read the code, a magnifying box and a code reading window are provided to magnify and display the actual code. This technique works well because it allows the user to have both an overview of the statistic and also read the interesting parts of the code.

2.2.2 3D

- Document Lens

  Document lens [17] describes a general technique for understanding paper documents with pages of document laid out in a rectangular array. It is a 3D technology, the user uses a rectangular magnifying lens and pulls it around to focus on the desired area at the desired magnification. The document outside the lens is stretched to provide a continuous display of the global context. The Document Lens is similar to a normal magnifying lens, except that its sides are elastic and can pull the surrounding parts of the region toward the lens providing a truncated pyramid. The sides of the pyramid contain all the document that are not directly visible in the lens, stretched properly. This technique allows the user to quickly focus on a part of the document while continuously remaining in context.

- The Perspective Wall

  The Perspective Wall [13] is used to visualize linearly structured information with wide aspect ratios. Typical examples of such information are project records ordered by chronology and directory entries ordered alphabetically.

  The Perspective Wall uses hardware support for 3D interactive animation to imitate the architecture of the human vision system. It displays the entire layout in a single view, and folds a 2D layout into a 3D wall, having a center panel for detail and two perspective panels for context. By folding a 2D layout into a 3D wall, the perspective wall technique shows detail and preserves relative distances at the focus, and integrates the detail smoothly with the rest of the layout.

This technique is meant to solve the problem of inefficient utilization of screen space, since the length of the information space to be visualized is much larger than its width.

- Cone Tree

Cone tree [16] allows its users to make sense out of large data collections by using animation to stimulate the human perceptual system to recognize patterns in the data. It is a 3D technique used to visualize information with hierarchical structure.

Cone trees are hierarchies laid out uniformly in three dimensions. The top of the hierarchy is the apex of a cone with its children placed evenly spaced along its base. The next layer of nodes is drawn below the first, with their children in cones. Cones in each layer have the same height, and the body of each cone is shaded transparently so that the cone is easily perceived yet does not block the view of cones behind it. When the user selects a node with the mouse, the cone tree rotates and brings to the front the selected node, and nodes in the path from the selected node up to the top of the cone tree, with all of them highlighted. The whole tree can also be rotated continuously to allow the user to see how the substructures are related to each other.

The Cone Tree is similar to the "Perspective Wall" technique in the sense that they both provide a fisheye effect, without distorting the view, by using the natural fisheye effects of 3D perspective. The difference is that Cone Tree handles 2D layouts that are large in both dimensions.

- InfoCrystal

InfoCrystal [22] is a high level information visualization and retrieval tool. It enables the users to explore, manipulate, and understand large information spaces in an interactive way.

The infoCrystal uses proximity, rank, shape, color, size coding to enable the user to see, in a single view, how a large information space relates to several their interests. The technique uses shape can color coding to indicate the number of criteria an icon satisfies. A circle shape shows that this icon satisfies one criteria, rectangle represents two, triangle three, and square four. The position of an interior icon is used to represent how close the content of this icon is related to the criteria icon, and icons' orientation reflect the criteria they satisfy. Size of an icon is used to code quantitative information, that is the number of elements represented by the icon.

2.3 Visual Representation Techniques

2.3.1 Color Coding

Representing data using color is attractive because the human visual system is capable of differentiating easily among hundreds of colors [15]. As an important graphical property, color has been used in many visualization applications. For example, in Painting Multiple View [14] discussed earlier, color is used to paint the corresponding parts of the data displayed and link separate views together. In Seesoft [9], another visualization techniques studied earlier, color is used to represent statistics associated with each line of code. The statistics can be age, programmer, the time it was last modified, how it is reached, and so on. For example, when color is used to represent the age of code lines, the color of each line is determined by its age. Red is used to represent the most recently added code, blue for the oldest, and a rainbow color spectrum between.

However, the choice of color is important, especially when it is used to represent many types of statistical variables. As it is pointed out in Eick's paper [9], choosing color naively is ineffective and will result in nearly all data items represented by the same color. The choice of a color scheme is crucial in visualization applications. Fienberg [18] has highlighted the unsatisfactory state of developments in this area, indicating the need to examine the effectiveness of certain color schemes. In the following section, we describe some of the existing color systems.

Color Coding Systems

There are three major color coding systems, namely RGB, CMY, and HSV. In all these color systems, color is determined by three independent components. In RGB, these three components are Red (R), Green (G), and Blue (B), and they correspond with the red, green, and blue cathode ray tubes. In CMY color system, color is determined by Cyan, Magenta, and Yellow. The CMY system is related to RGB as follows: $C = 1 - R$, $M = 1 - B$, $Y = 1 - G$. Clearly, they are complementary to each other. The three components in HSV system are Hue (H), Saturation (S), and value (V). As with the other two systems, they all have values between 0 and 1. The Saturation determines the saturation of the color. $S = 1$ corresponds to full saturation and $S = 0$ corresponds to no color (white). The Value determines the intensity of the color. $V = 1$ corresponds to maximum intensity, and $V = 0$ is no intensity (black).

The color coding used most frequently in visual environment is the HSV system. First of all, the effects of manipulating the components are much more predictable than in both RGB and CMY system. Secondly, HSV corresponds to a more natural experience of colors than the other systems [1], since saturation and intensity are natural variables in the HSV system.

[1]The relation between the RGB, CMY and HSV system is very complicated. Interested readers are referred to reference [18]

2.3.2 Shape Coding

Shape is another graphical variable that is used often in visualization applications to encode data values. It has significant implications for both scientific visualization and information visualization. Much more complex data sets can be graphically represented and visualized using shape coding technique. It serves two main purposes [4]: first, it simplifies the visual field and provides a more compact illustration of the data visualized. Secondly, it serves as a quantitative strategy and identifies distribution of extreme values within the data set.

2.3.3 Sound Coding

Sound is an independent channel to the user for conveying information. Sound is used in the human-machine interface in two ways. One way is to provide the user with the occurrence of an important events, and the other is to use sound as a medium for representing data [23]. Values of multivariate data can be represented by various sound parameters such as pitch, loudness, duration and so on.

When sound is used to provide cues, it is important that the user be able to identify the source of the sound rapidly and effortlessly. When sound is used for data representation, it is important that the user be able to hear the data-bearing relationships within and between sounds. The focus is on the sound itself, as when we listen to music.

2.3.4 Dynamic Graphical Method

A dynamic graphical method is one in which a data analyst interacts in real time with a data display by using a mouse on a computer screen [2, 3, 11]. "We see more when we interact with the picture, especially if it reacts instantaneously, than when we merely watch." described by Huber [2].

In dynamic graphics, the mouse is used to adjust, among other things, various sliders that control the parameters such as: overall line thickness, overall symbol size, line length, and so on. The net effect is the ability to quickly and naturally filter through many static pictures that may be uninteresting, and to focus on the ones that are meaningful. The authors in [11] believe that the combination of the dynamic modification of the parameters and visual feedback allows the user to adjust the parameters efficiently to produce informative displays.

Dynamic manipulation can be used to solve the following problem encountered in both scientific and information visualizations. One approach to visualization is to draw symbols such as lines or circles with the symbol scaled to represent the value of the statistic. In the case of a line, the line's corresponding statistics, such as line length and sickness, falls above or below some threshold. The difficulty with this approach is that it is very hard to come up with a good heuristic for setting these thresholds, line lengths or overall line thickness before making the display.

2.4 Human Perception Factors

When scientific data is presented in a graphical display, the viewers uses their visual perception to make judgments about the display, to compare relative sizes, orientations, colors, densities, textures, etc, of the elements displayed. Understanding how humans perceive visual information could help improve the quality and effectiveness of visualization. Unfortunately human visual perception is a very complicated and subjective process and hinges crucially on a wide range of subtle factors. For example, different people may have different perceptions of the same picture due to color blindness and other visual impairments (physiology). Perceptual psychologists and graphic designers have not developed a complete inventory of all the factors and interactions that affect visual perception.

Fortunately, certain broad principles and common sense can be used to guide our design. In [23], Tukey describes some of them as follows:

- Choice and expression of scales
- Objects are perceived in relation to their surroundings

- Straight lines are easier to perceive than curves. Horizontal lines are easier to perceive than oblique ones.
- Irrelevant material can seriously interfere with a plot
- Things that are closer together are easier to compare than things far apart
- Things of equal importance should have roughly equal visual impact
- Motion is more effective for conveying 3D depth than perspective

Tukey believes that the issues of good graphics design and human visual perception must be taken into consideration in order to make graphical display in visualization applications convey the insights they are meant to. As data visualization becomes more and more important in computer science, it becomes more and more urgent for psychophysics and graphic designers to study issues of visual perception.

Chapter 3

Overview of BISDN Networks

3.1 Introduction

Broadband Integrated Services and Digital Networks (B-ISDN) represents a revolutionary communications technology capable of delivering a variety of services over a common switching and transmission framework. Such networks offer unprecedented flexibility by allowing services with different requirements to map onto the same transmission systems. The B-ISDN environment is defined in terms of a layered protocol architecture consisting of service and signaling descriptions, adaption formats, a common information transfer standard - Asynchronous Transfer Mode (ATM), and a number of physical layers.

A major application of B-ISDN networks will be high-speed data communication. Computer networking will be enhanced with high-speed host-to-host bulk data transfer, enabling fast interconnectivity for distributed applications. Another major application area of B-ISDN networks is video. Television and video services ranging from 2 Mbps to 600 Mbps can be transported by the same network. Examples are video telephony, video conferencing and TV distribution. Yet another major application area of B-ISDN networks is imaging. Services based on high resolution still images will be available. Examples include high resolution color fax machines, telemedicine, and telepublishing. The B-ISDN network will integrate the four major broadband service types, namely, voice, data, image and video, on one network, resulting in truly multimedia communication networks.

3.2 ATM

ATM stands for Asynchronous Transfer Mode. It is the technology selected by the standard bodies of the CCITT to realize a B-ISDN network. ATM is a fast packet-switched technology based on a fixed length packet. All services, both broadband and narrowband, are divided into a series of fixed size ATM cells and routed across the B-ISDN network.

ATM takes its name from the method of transferring cells across an ATM switch. Incoming cells contend for slots in outgoing cell streams. If a slot is not available, the cell is queued within the ATM switch. This introduces a variable cell delay across the ATM switch, which is dependent upon traffic density. Owing to this variable cell delay, the ATM cells are said to be transferred asynchronously across the ATM switch.

3.2.1 ATM Cell Structure

The fundamental unit of ATM is the ATM cell. An ATM cell is a 53 bytes fixed-sized data unit. It consists of a 5 byte header field and a 48 byte information field. The header field contains information about the cell, which describing the type of cell and its routing information. The information field carries the B-ISDN service, that is the user's data to be transfer over the network. The ATM cell is the basic unit of information transfer within the B-ISDN/ATM network.

The ATM cell header is a very important portion of the cell. It contains information such as routing and congestion control. Without the cell header, ATM data transmission is impossible. The header contains GFC (in the case of User to Network Interface only), VPI, VCI, PT, CLP, HEC fields as shown in Fig.1.

The Generic Flow Control (GFC) field is to be used for end-to-end flow control,

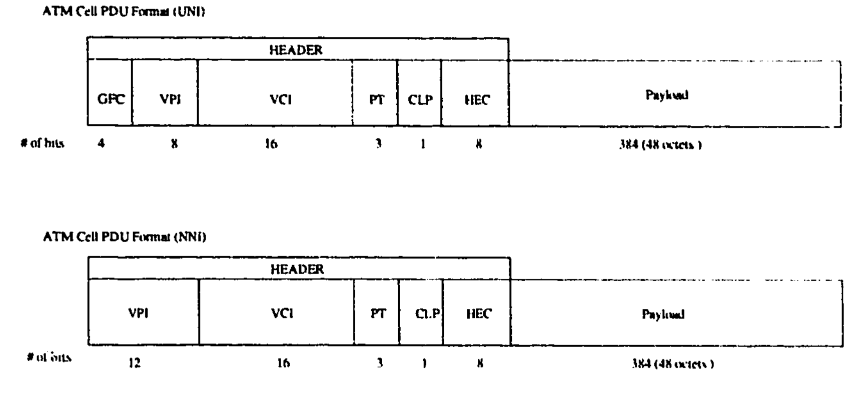

Figure 1: ATM Cell Format.

since multiple terminals may share a single access link to the network. The details of its application are yet to be defined by the standards body. The field could be used to assist the customer in controlling the flow of traffic for different qualities of service. One candidate for the use of this field is a multiple-priority level indicator to control the flow of information in a service dependent manner.

The Virtual Path Identifier (VPI) and Virtual Channel Identifier (VCI) fields constitute routing fields for the network. The VPI indicates a user-to-user or user-to-network Virtual Path Connection (VPC). The VCI indicates a user-to-user or user-to-network Virtual Channel Connection (VCC). A VPC, identified by unique VPI value, may contain a number of VCCs identified by different VCI values. VPIs and VCIs are network resources inside an ATM switch to be managed by means of use-on-demand (Switched Virtual Connection) or on permanent basis (Permanent Virtual Connection). There are commercially two types of ATM switches available, VPI based and VCI-based. A VPI based switch selects a new VPI for its output port based on its built-in input/output VPI mapping table. All VCIs inside a VPI based switch remain unaffected. On the other hand, a VCI based switch uses both VCI and VPI to select its output port based on its built-in VCI/VPI mapping table for different ports. In both cases, a pre-occupied VCI/VPI has to be "torn down" before making it ready for reuse. Therefore, all VCIs and VPIs have local significance (as with X.25 and frame relay) and constantly change as the ATM cells traverse the network.

The Payload Type (PT) field indicates the type of information contained in the information field. It distinguishes between user data and network information. In general, user data and associated adaption information traverse an ATM network and the user access lines, whereas network information is for operation and management purposes within the network. The first bit of the Payload Type is set to 0 for user data and 1 for network information. The second bit is set to 1 as an indication to the destination, that user cells have passed through congested switches, and 0 otherwise. The third bit is for user-to-user signaling and can be used in conjunction with AAL-5, which will be introduced and defined later, to indicate which ATM cell is last in the segmentation of convergence sublayer data units.

The Cell Loss Priority (CLP) field is used to provide guidance to the network in the event of congestion. A value of 1 indicates a cell with lower priority. Such cells are discarded by a congested ATM switch before discarding regular cells. The CLP field is used only when some cells within a single VC have higher priority than others. For example, in a channel carrying video, the synchronization information is more important than information about individual pixel values. The loss of synchronization information will have more effect on the entire viewing quality than loss of a group of pixels. The percentage of cells that are high priority is agreed upon between user and network at setup time. The network will set all the data cells that are in violation of a traffic agreement. It is important to note that Cell Loss Priority is not the only mechanism for controlling the quality of services.

The Header Error Control (HEC) provides error control on information contained in the cell header. The HEC field is an 8-bit error code that can be used to correct single-bit errors and detect double-bit errors. The transmit side calculates a Cyclic Redundancy Check (CRC) error-code value based on the contents of the cell header and inserts the resulting code into the HEC field. The receiver side, using the same CRC generation algorithm, calculates an error code value based on the contents of the received header. Then, the receiver side compares the value that it has calculated with the contents of the error-code field that is received as part of the transmission. If the code matches, it is assumed that no error has occurred. If there is no match, then an error has been detected. HEC field only provides protection of the ATM cell header against possible corruption during the cell's journey within the ATM network. The cell structure doesn't provide a. v means to protect its payload information from errors. It is the responsibility of upper layer protocols, for example ATM Adaption Layer protocol (AAL), to do error control for the payload.

3.2.2 ATM Cell Size

A key feature of the ATM protocol is the use of short and fixed-length cells. All user and network information, whether it be voice, data, fax, image, or video, is transported using the small cell format.

There are several advantages to the use of fixed-size small cells. First of all, the use of small cells may reduce queuing delay for a high priority cell, since it waits less if it arrives sightly behind a lower priority cell that has gained access to a resource (e.g. transmitter). Secondly, flexible bandwidth allocation is possible using ATM. A service can dynamically use as many or as few of the ATM cells as it requires. This provides greater flexibility and allows different services to be transported in a uniform manner. Thirdly, it appears that fixed-size cells can be switched more efficiently. Because of its fixed cell structure, ATM cell switching can be done by hardware based on the cell header. This allow networks to operate in the gigabit range, which is important for the very high data rates of ATM. Consequently, ATM provides a uniform, flexible, fast, and cheap delivery mechanism for the variety of broadband services.

As to the question of why the ATM cell size is exactly 53 octets, interesting enough, it is actually a compromise between the European communication authority and its counterpart in North America. Everybody agrees on the 5 octet length of the header but, Europeans prefers to use a shorter (32 octets) payload, while Americans prefer to use a longer payload (64 octets). Therefore, the standard body decided to adopt the structure of 48 ((64+32)/2) octets of payload plus 5 octets of header, producing a cell size of 53 octets.

3.3 ATM Networks

3.3.1 ATM Protocol Stack

B-ISDN/ATM is a layered approach and uses a layered protocol stack model. Starting from bottom of the ATM protocol stack, they are Physical Layer ATM layer, ATM Adaption Layer, and Service Layer.

Physical Layer

The job of the physical layer is to transport the bits that are passed from the higher layer. The associated function is encoding/decoding the bits into/from electrical/optical systems. The convergence layer maps cells passed from the ATM layer into a flow of data units over a physical medium.

ATM Layer

ATM provides the transport of fixed-length cells from an interface with one user to an interface with another. An essential function of ATM is addressing using VCI/VPI fields of the cell header, so that a cell can be routed correctly to reach its destination. The other functions of ATM include traffic management (CLP field) and multiplexing (PT field) of the traffic. As we described in the previous section, all ATM functions are done exclusively using the 5 bytes ATM cell header.

ATM Adaption Layer

The ATM layer is highly simplified to allow for hardware implementation. The purpose of AAL is to adapt the pure transport function of ATM to the requirements of different user services. The functions of AAL are divided into two sublayers: Convergence Sublayer (CS) and Segmentation and Reassembly Sublayer (SAR). The functions performed by the AAL include error control on user data and the packaging of user data into fixed length ATM cells. There are five different adaption protocols specified to handle different upper layer services, namely AAL-0, AAL-1, AAL-2, AAL-3/4 and AAL-5. We mainly concentrate on the AAL-5 protocol in this thesis.

AAL-5 is a simple and efficient Adaption Layer Protocol. It is designed for connection-oriented variable bit rate services, such as X.25 and Frame Relay. AAL-5 is a streamlined type of adaption for services requiring no end-to-end timing relation. The main functions of AAL-5 are performed at the CS sublayer, and consist of control of bit errors and maintaining a record of the length of the AAL service data unit.

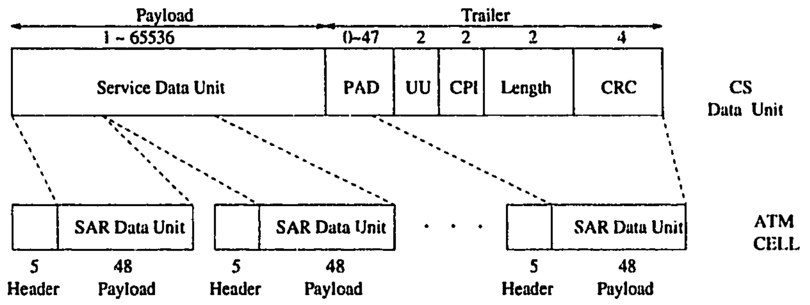

Figure 2: AAL-5 protocol data unit.

Fig.2 indicates the format of the CS data unit, which can carry a service data unit of up to 65,536 octets. The trailer attached to the service data unit contains 4 octets for CRC error control, 2 octets specifying the length of the payload, a control field, and padding, to make the total length up to a multiple of 48 octets. Since the CS data unit is a multiple of 48-octets long, it can be segmented directly into 48-octet payloads for ATM cells without segment type and length indicators. This means that there is no SAR sublayer in AAL-5.

One characteristics of AAL-5 is that it requires separate ATM connections for each service being carried. That is, the ATM cell components of those AAL-5 PDUs that belong to different service data have different VCI/VPI field values. This can be used to reassemble AAL-5 PDUs from ATM cells at the receiver side of the ATM network. It is this characteristics that makes it suited to connection-oriented traffic.

Services Layer

Services are the user data that is transferred over a B-ISDN/ATM network. The following are some of the services to be offered under the umbrella of B-ISDN. Switched Multimegabit Digital Service (SMDS) is a wide area data service being offered by regional telephone companies in North America and in Europe. Frame Relay is another example which evolved from older technology, the X.25 protocol. X.25 is a common protocol used in our public data network. Credit card data transfer, for example, is done using X.25. Video transmission for video conferencing and TV transmission can easily be deployed over ATM networks. Signaling is an embedded service that the network provides to the user, enabling users to set up and disconnect their connection dynamically.

3.3.2 Data Transfer Over ATM

The ATM network consists of a set of ATM links connected by ATM switches. Each ATM link comprises a constant stream of ATM cell slots into which services are placed.

There are a few stages involved in transmitting a B-ISDN service through an ATM network. The first stage is know as cell segmentation, and it occurs at the network termination point. This is the process of mapping the broadband service into an ATM cell stream, service data from a particular user is assembled and placed sequentially in the information field of the ATM cells. The header field is then added, containing the cell's virtual channel identifier (VCI), its virtual path identifier (VPI), and its cell loss priority (CLP). Cell loss priority indicates whether a particular cell can be lost in transmission without seriously impacting the quality of the received service. The cell is then multiplexed into vacant slots in the ATM cells stream. Multiplexing allows more than one service to share a particular ATM link at any given time.

The next step in transporting the broadband service is routing the cells through the ATM network. This function is handled by the ATM switch. Each cell within the incoming cell stream is switched to an outgoing cell stream on the basis of its VCI or VPI, which are contained within the cell header.

Finally, at the receiver side, the ATM cell stream needs to be reassembled into the original broadband service. Using the same type of AAL protocol as at the transfer side, the information field of ATM cells belonging to a particular virtual channel are sequentially extracted and reassembled into AAL PDUs, and finally the original broadband service data.

3.3.3 Traffic Control and Congestion Control

Three levels of traffic control are used in ATM networks. At the call level, a call is rejected if the requested bandwidth is not available at the time of the call setup request. At the connection level a call is rejected if there is no available path to its destination. At the cell level, the network provides functions for sharing buffer storage as well as switching and multiplexing. Admission control is used to guarantee a certain Quality of Service for the already established connections. A new connection is accepted only if the aggregated traffic stream results in an ATM cell loss probability smaller than a certain maximum predefined tolerable loss probability. Otherwise, the connection is rejected. The acceptance of a call is only possible after the successful application of the resource allocation mechanism at all three levels.

When a virtual channel consistently exceeds its bandwidth limits, or when the output bandwidth of an ATM switch is insufficient for the traffic conditions, ATM policing will occur. In either case, some cells must be removed from the network. The cell lose priority (CLP) defined in the cell header field is used to determine which cells to discard. Cell loss will continuously occur until the network congestion clears.

3.4 Performance of ATM

There are a number of parameters that characterize the performance of ATM networks. Parameters concerning network performance, both traditional and unique to

ATM, have arisen. Errors in communication data is certainly not new. Bit errors still occur in ATM networks, even though the employment of modern transmission techniques and media (optical fiber) has significantly reduced error rates. Cell loss and cell delay are performance problems that are unique to ATM. Cell Loss Ratio and Cell Delay Variation are important parameters to measure in a given ATM network.

3.4.1 Errors

Errors can be classified into header error, and higher level PDU decode error, and sequence number error. They could be caused by cell corruption, cell loss and cell misinsertion.

Cell Header Error

Header errors include correctable, detectable and non-detectable. As we described earlier, the CRC check sum embedded in ATM cell header is capable of correcting single bit error, and detecting double bits error. If more than two bits in the header are corrupted, the error will become non-detectable using this technique. Non-detectable errors might be detected at higher protocol layers.

Higher Level PDU Decode Error

At the receiver side of a transmission, the payload of ATM cells will be extracted and reassembled into higher level PDUs, and eventually they are reassembled into user service data units. The process of mapping the payload portion of ATM cells into AAL PDUs is called PDU decoding. Similar to an ATM cell, a higher layer PDU has a header or tailer or both, plus a payload portion used by the next higher protocol layer as its PDU content. The header or the trailer of a PDU contains important information about the payload, and is checked or verified upon decoding. If the value of one or more fields in either the header or trailer does not comply with the corresponding protocol, errors occur. Such errors are called high level PDU decode errors.

Sequence Number Error

A sequence number included in the cell payload can be used to determine if cells are misordered as they are switched through the network. Sequence errors occur if sequence numbers embedded in the received PDUs are out of order.

3.4.2 Cell Loss

Cell Loss happens in two cases. One possible cause is due to ATM cell header errors. In order to ensure user security and prevent mis-routing of errored cells, any cell found at a network node with an error in the routing fields of the cell header is automatically discarded. Cell loss can also occur when the network becomes congested and the output buffers in an ATM switch overflow. When congestion is detected, cells within the queue are selectively discarded according to the CLP field value in the cell header.

3.4.3 Cell Delay and Delay Variation

In addition to the normal delay through network elements and lines, extra delay is added to an ATM network at the ATM switch. The causes are the asynchronous statistical multiplexing and asynchronous path of each ATM cell. The delay of a cell depends on the amount of traffic within a switch. One or more cells could be held in a buffer waiting for the next available slot to continue transmission.

The asynchronous nature of ATM means that transmission delay is not only magnified, but also inconsistent throughout an ATM network. The variations in cell delay is called Cell Delay Variation (CDV). CDV is caused by the multiplexing of ATM cells, and the extent of variation depends on buffer size within ATM switches.

3.5 Network Testing

Quality of Services (QoS) offered by an ATM network is defined by a set of parameters. QoS parameters depend on the type of user services. Constant Bit Rate (CBR) services, including voice, require uniform repetitive information transfer. Typically they are connection-oriented real-time applications (for example, telephone). The real-time nature of these applications makes them extremely sensitive to cell delay and cell delay variation. An important feature about voice data is that if it arrives late, it becomes useless. Moderate cell loss, however, is acceptable for CBR services. Variable Bit Rate (VBR) traffic describes random and bursty data. These services can be connection-oriented and connectionless. VBR applications include multimedia, LAN traffic, and file transfer. These service are not real-time and therefore tolerate a degree of cell delay. However, VBR may rely heavily on data integrity, and in some cases a single lost cell can mean complete data retransmission. Therefore, cell losses must be minimized in VBR traffic.

The success or failure of B-ISDN, and ultimately ATM, relies on the ability of network providers and manufacturers to provide high quality services and products. To ensure high quality products, QoS parameters need to be tested or monitored and measured under different traffic load conditions. Specific test tools are required for testing. The HP 75000 Broadband Series Test System is one of the earliest tools to appear on the market to test products in ATM networks.

Network Testing is a very complicated issue, and involves many aspects of performance parameters. In this thesis, we mainly focus on some of the tests for ATM and AAL protocol layers, and performance tests such as cell delay.

3.5.1 Testing At The ATM Protocol Layer

ATM layer testing is concerned with ensuring the robust transfer of ATM cells across a B-ISDN network. Testing would confirm the cell routing, cell integrity and congestion capabilities of a B-ISDN element.

3.5.2 Testing At The AAL Protocol Layer

AAL testing is concerned with two aspects: Segmentation and reassembly testing verifies that services transported by the B-ISDN network are mapped correctly into/out of an ATM cell stream. Performance testing of the AAL layer confirms the convergence process and integrity of the SAR_PDU. Both cell loss and misinsertion belong to this category.

Testing Cell Loss

As we described earlier, cell loss in an ATM network is primarily caused by congestion within switches. By overloading the switches' input ports using ATM load generator and examining cells from the switches' output, it is possible to test cell tagging and discard algorithm and the policing function within the switches. Cell loss is calculated by monitoring of the sequence number in the AAL layer PDU header of an ATM connection. The cell loss ratio is calculated as the number of cells lost divided by total number of cells sent.

Testing Cell Misinsertion

Cell misinsertion happens when the protocol analyzer receives more ATM cells than it transmitted. Cells can be counted over periodic time intervals. If a greater number of cells are received than were transmitted, cell misinsertions occur. It means that the network inserted those cells on the wrong output port.

Similarly to cell loss, cell misinsertion can also be calculated by monitoring the sequence number in the header of AAL PDUs. However, if both cell loss and misinsertion occur on a switch, it may be difficult to determine how much of each is happening, since the effects "cancel" each other.

3.5.3 Testing Cell Delay

QoS testing can be performed for one or a number of ATM network components, which are called the System Under Test (SUT). Measuring the absolute cell delay through a single SUT is relatively simple. By sending a test cell through the SUT and recording the exact time of generation and detection, the transmission time can be calculated.

Cell Delay Variation statistics describe the characteristics of a switch or network. It shows the maximum and minimum cell delay through a network and the distribution of delays in-between. CDV can be measured without the use of timestamps. By generating Constant Bit Rate traffic of a known frequency and sending it through the SUT, the interarrival time of the received cells can be used to determine the cell delay variation. The major advantage of this method is that synchronization of the test equipment is not necessary.

There are two ways of looking at both cell delay and CDV. The real-time view using fractions of seconds, and the cell length view using cell slot times.

Chapter 4

Visualization Approaches

Our approach to visualization can be divided into two parts, time lines and cell visualization. The time lines provide the structure for the visualization. They represent a summary of all, or a significant fraction, of the data to be visualized. The cell visualization provides information on individual data units displayed on the time lines. It is a graphical representation of the important properties of either a individual PDU or a group of PDUs along time lines.

This division of the visualization into two parts highlights two important aspects of the data. The time line highlights the temporal nature of the data and provides a summary of the data set. Since the cells are not evenly distributed over time, time line visualization allows the user to quickly see how events arrive over time, and how this distribution relates to the different types of cells and events (for example, errors) in the data. The cell visualization focuses on the individual graphical representation of data. It provides a much more compact representation of the data than the current textual representation. Using graphical properties to encode important fields of ATM cells and high level PDUs, cell visualization provides the user a very intuitive way to spot interesting data packets.

In the remainder of this chapter, we describe in detail the visualization techniques that we have explored. Examples of how these visualization can be used are presented. These examples are drawn from prototypes that use data collected from the protocol tester, paper design sketches, and discussions of design alternatives.

4.1 Time Line Visualization

The purpose of the time line visualization is to present the temporal distribution of the data and provide a structure for the visualization. A time line visualization presents a linear sequence of images, where each image represents either a single data packet, or a collection of them. The information in the time lines is presented in sequential order according to the arrival time stamps on the individual data packets. Since most of the data collected by the protocol analyzer has a time stamp associated with it, time lines are a natural way of structuring the data.

4.1.1 Hierarchy of time lines

Our time lines must use a compact representation of the data packets, so the user can have an overall view of all the data collected, not just one or two packets. At the same time, the time lines should also provide the user with a detailed display, so the user can examine individual data packets in detail. In order to achieve this, a hierarchical time line visualization is proposed. The idea of hierarchical time lines is that at the top layer of the hierarchy the time line represents all the data packets in the entire capture buffer, and at the bottom of the hierarchy the time line holds the individual data representations.

Hierarchical time lines allow the user to view the information at several levels of detail. At the highest level of the hierarchy, the time line uses a reduced representation of the data collected, so the user can scroll through the complete contents of the capture buffer on the screen. At this level, there is very little detail on the individual data packets, with each point on the time line representing hundreds or thousands of data packets. The purpose of this time line is to provide the user with a summary view of all the data in the capture buffer. At the lowest level of the hierarchy, the time line represents a stream of individual data packets. This allows the user to see the details of each individual data packet.

Hierarchical representation also allows the user to view the data at several different time scales. At the highest level of the hierarchy, the time line displays the high level structure of the traffic flow. With reduced representation, the user can see how the data space distributes over time along this single time line. This allows the user to see patterns that are developing over time, and correlate events that occur at widely separate points in time. The user can zoom into the parts of the time line that are most interesting. Each level of zoom exposes more detail on the individual data packets. At the lowest level of detail the user can examine the content of each individual data packets in detail.

4.1.2 Scrolling Mechanism

Even with the most compact representation, all the data in the capture buffer can not be displayed at the same time with limited display space. Scrolling can be used to solve this problem. Using scrolling, a time line can be many times larger than the window that is used to display it.

A scrolling mechanism can easily be combined with our hierarchical time line representation. Scrolling operates at one level of detail, it does not provide a zoom-in or zoom-out facility. Only the top level of the hierarchy can represent the entire capture buffer, lower levels of representation only show a part of the buffer. At the highest level, by scrolling, the user can visualize the complete capture buffer. At the lower levels, the scrolling mechanism can be used to move back and forth and show each individual data packets through the capture buffer. By making all time lines scrollable, the user can move back and forth through time. This provides the user with a way of navigating through a time line, so the user can see the part that is of current interest.

Using a hierarchical organization and scrolling technique, our visualization lets users see individual cells in detail, and also allows them to examine the values of individual fields of these cells when necessary. All time lines in the visualization hierarchy should also be synchronized, so when the user scrolls one of them, all the lower level time lines will update themselves to show the new part of the time line.

4.2 Cell Visualization

We use the term "cell visualization" to represent the techniques used to visualize a single data unit on a time line. It could be a single data packet or group of packets displayed along a time line. In the case of a single data packet visualization, which happens in the bottom level of our time line hierarchy, a data packet could be either an ATM cell or a higher protocol layer PDU.

The cell visualizations are used to show the important properties of a data unit using a graphical representation. These visualizations have two main purposes. The first purpose is to provide a compact representation of the data displayed along a time line. We want to get as much data onto the screen as possible, and at the same time provide as much detail as possible on each data unit. The second purpose is to draw attention to patterns, unusual data packets, or anomalies from the data collected. Since the data tends to be quite repetitive, it can be very difficult to spot rare events and patterns unless they are highlighted. The graphical representation facilitates the detection of patterns in the data, and highlights unusual events.

4.2.1 Upper Level Time Lines

Cell visualization for upper level time lines, that is all time lines that are above the bottom level, graphically encode a group of data packets into a graphical image. The individual images provide information on collections of data packets along the time line. The shape of such an image could be a circle or a rectangle. The length of an image along the time line axis (called the x axis) encodes a certain period of time, or time span, in the time line. The image shape, precisely its length along x axis, codes or determines the group of packets it represents. The graphical variables of such images are used to represent the following two things: one is the number of data packets that arrived during the time span of this image; the other is the type of data packets it represents. We call these the data density visualization and alarm condition visualization respectively.

Alarm Condition Visualization

Alarm Condition is a term used by engineers who work on the HP BSTS at HP/Idacom. It is used to describe a certain type of data packet that the user wants to be notified of upon its arrival. The first type of alarm condition would be cells that have errors or do not obey certain performance parameters. Typical errors are ATM cell header error or HEC error and higher level PDU decode error. Cell delay visualization, discussed later in this chapter, is one example of cells that do not satisfy a certain performance requirement. Another type of alarm condition is a set of data packets that satisfies certain conditions. In this case, the data specified is normal in the sense that there is no errors occurring within its data unit. They could be some data packets that either have a specific field value, or contain a certain data patterns in its payload. For example, a user might want to look at those ATM cells that are routed through a specific virtual channel, that is their VCI field equals to a certain value.

Technically alarm conditions can be coded either using color or using a certain graphical shape. In all our designs, we use the color red to code the alarm conditions, so that they will be easily recognized.

Data Density Visualization

As we described earlier, each visualization unit in the upper level of the time lines represents a group of data packets. Since the arrival time of individual data packets is not distributed evenly over time, one thing the visualizations show is the density of cells. That is, the number of cells that have arrived within this particular time interval.

The number of data packets represented by higher level time lines can be graphically coded in two ways as shown in Fig.3. One is to use color encoding. That is, we

Figure 3: Cell Density Visualization.

use colors that have an uniform hue distribution to code cell density. The units with sparse cell density are drawn with lighter colors, and the ones that represent dense cells are drawn with darker colors. The other technique is to use the length of each unit along the y axis to represent the number of cells that have arrived within its time interval. The more cells it represents, the taller or wider the image is, depending on how they are laid out.

4.2.2 The Bottom Time Line

Cell visualization at the bottom line focuses on individual data packets at both the ATM layer and AAL layer. Each protocol data unit is graphically represented as a single image. The purpose is to provide the user with a compact representation of protocol data units, with color and shape to highlight their important properties.

Protocol data units at different protocol layers are coded in different colors to show their difference. As shown in Fig.4, ATM cells are drawn in lighter colors, and AAL PDUs are drawn in darker colors. Besides çolor, shape can also be used to differentiate ATM cells from higher layer PDUs. For example, ATM cells are drawn as squares, while AAL PDUs are drawn as octagons and hexagons. Using two sets of coding, namely color and shape, at the same time emphasizes the differences and makes them to be easily identified.

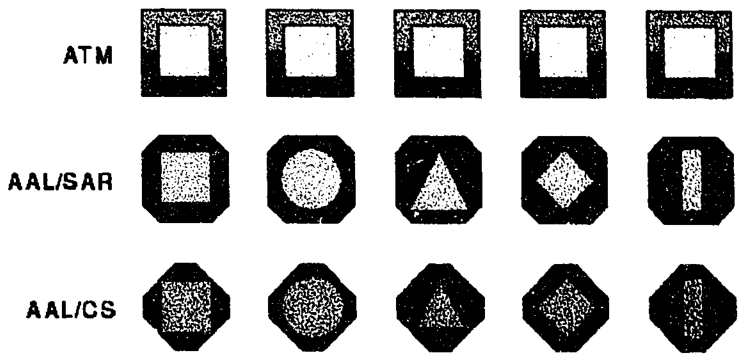

Figure 4: Cell Visualization at the bottom time line.

In Fig.4, we use the inside shape of a data unit to code the payload part of both the ATM cells and AAL PDUs. When ATM cells are decoded into upper level protocol layer PDUs, the payload of a PDU starts to show the type of service it carries. The type of services an AAL PDU carries can be coded using the inside shape of an AAL PDU. For example, an AAL PDU with a triangle inside can represent voice data, diamond shape represents video data, circle represents images, and a double rectangle shaped AAL PDU shows it carries computer data. Since type of data embedded in an ATM cell is not visible, we use a rectangle to represent the payload portion of an ATM cell regardless the type of service it carries.

Protocol Stack Visualization

As we described in the previous chapter, B-ISDN/ATM network users do not communicate at ATM cell level, even though all service data are transferred through the network in the format of ATM cells. One of the AAL protocols must be used as the interface between user data and ATM cells before and after the transmission.

Protocol stack visualization correlates a higher protocol layer PDU with its ATM cell components using cell visualization techniques. It is important to provide the mapping between AAL PDUs and their ATM cell components to the user. It allows the user to trace down through the protocol stack when an error occurs in a higher level PDU. The purpose of this visualization is to solve the problem of determining the origin of erroneous data when testing higher level PDUs. As we stated before, in this thesis we mainly focus on AAL-5 protocol, a simple yet effective protocol from the AAL protocol family.

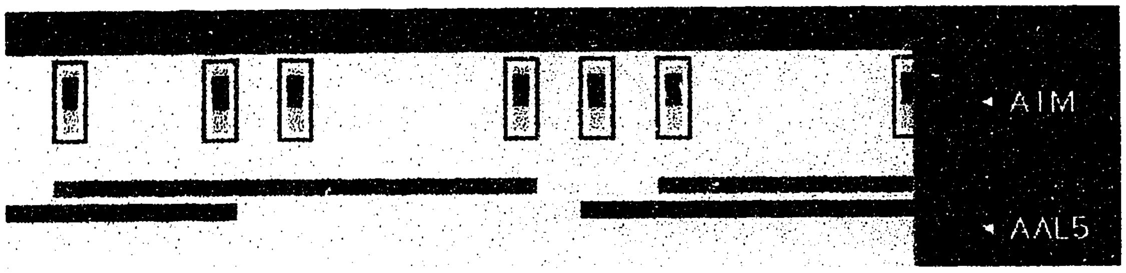

Figure 5: the protocol stack of ATM to AAL-5.

Fig.5 is a snapshot taken from the bottom time line of one of our prototypes. It shows the technique we used to visualize protocol stacks. As we can see from Fig.5, an AAL-5 PDU is drawn as a rectangle with its color corresponding to the ATM cells it was reassembled from. More precisely, an AAL-5 PDU is painted using the same color as the VCI field of its ATM cell components. The length of an AAL-5 PDU corresponds to the time span between the arrival of the first and the last ATM cells that it is reassembled from. As we can see, both VCI and VPI fields of an ATM cell header are highlighted using color, because they are the important information in the protocol stack of ATM. As we recall from the previous chapter, AAL-5 PDUs belonging to different types of service data are carried over the network through different virtual channels. In other words, ATM cells that map to different types of AAL-5 PDUs have different VPI/VCIs fields in the cell header. Moreover, there is no cell multiplexing at AAL-5 PDU level. That is, all ATM cells belonging to a single AAL-5 CS_PDU are sent sequentially. These two properties of AAL-5 protocol makes it straight forward to reassemble and visualize them.

One more thing needs to be noticed in Fig.5 is that AAL-5 PDUs, mapped from ATM cells that are transmitted from different virtual channels, are drawn using different y values. This is to solve the display problem of multiple AAL-5 PDUs that are transmitted from different virtual channels overlapping in time. As we described in last chapter, ATM cells that are transmitted from different virtual channels are multiplexed together. That is, their transmission is based on first-come-first-served bases. As a result, the time span occupied by AAL-5 PDUs that are transmitted from different virtual channels overlap with each other along the time line. In order to allow the user to trace down to the ATM cells origins from an AAL PDUs, our protocol stack visualization should show the mapping between an AAL PDU and its ATM cells components. Our solution is to shift the drawing of the overlapped AAL PDUs vertically along the time line.

As one can see from Fig.5, the combination of color, width and position of a certain AAL-5 PDU readily reveals to the user the underlying mapping between an AAL-5 PDU and its corresponding ATM cells. Using our protocol stack visualization techniques to draw ATM cells and AAL-5 PDUs, the correspondence between an AAL-5 PDU and its ATM cell components can be easily recognized.

4.2.3 Text Display

The main focus in our visualization discussed so far is graphically representing data. There should be a technique to allow users to read the contents of the cells and PDUs, since they may need to access the real value of a certain data unit. This has been done in two ways: One is to use a scrolling text window, in the same way as the current BSTS does. This has the advantage of being a technique that the users are already familiar with. The other approach is to pop up a text window whenever the user needs to look at the contents of some interesting cell headers and possibly the payload. This can be done by clicking the text icon in the main window. This approach saves window space, since the size of the HP BSTS display window is very limited.

Our text display window is synchronized with our graphical display window in either approaches. The contents of the individual ATM cells or AAL PDUs displayed at the bottom of our time line hierarchy are always at the nearest scrolling pages as our text window. When the user clicks on an ATM cell or AAL PDU, the text representation of that data unit is brought to the front page of the text window. This has the advantage of tying the text display to the graphical representation, and providing a quick way of accessing the complete contents of the cell.

4.3 Alternative Approaches to Time Line

There are alternatives to time lines for organizing the protocol data. One of the disadvantages of time lines is the need to scroll them in order to view the complete capture buffer. This can be avoided by arranging the data in a rectangle instead of a line as shown in Fig.6. The data is still divided into a number of units, with each summarizing the contents of a large number of ATM cells or PDUs. Instead of placing data units on a line, they are arranged in a rectangular format, with the top line of the rectangle representing the start of the capture buffer, and the last line representing the end of the capture buffer. In this way all the data in the capture buffer can be displayed at the same time.

The rectangular arrangement can also be used hierarchically. The high level rectangle shown in the upper left corner of Fig.6 serves as an overview of all the data in the capture buffer. It is a compact representation of all the data captured. Each of the images in the overview window represents a group of data units that can be expanded to a rectangle of cells at the next level of detail. When the user clicks on a compact image in the overview window, the group of data represented by that image is expanded and displayed in the detail window, as shown in the lower right rectangle in Fig.6. The size of the two rectangles don't need to be the same, since they serve different purposes in this visualization. For example, the top level rectangle could have 100 cells, while the lower level rectangles could have only 25 cells. Since the lower level rectangle will only have 1% of the data, it doesn't need as many cells, and can use cell visualization that show more detail.

The data unit in the low level rectangle can be zoomed in further for greater level

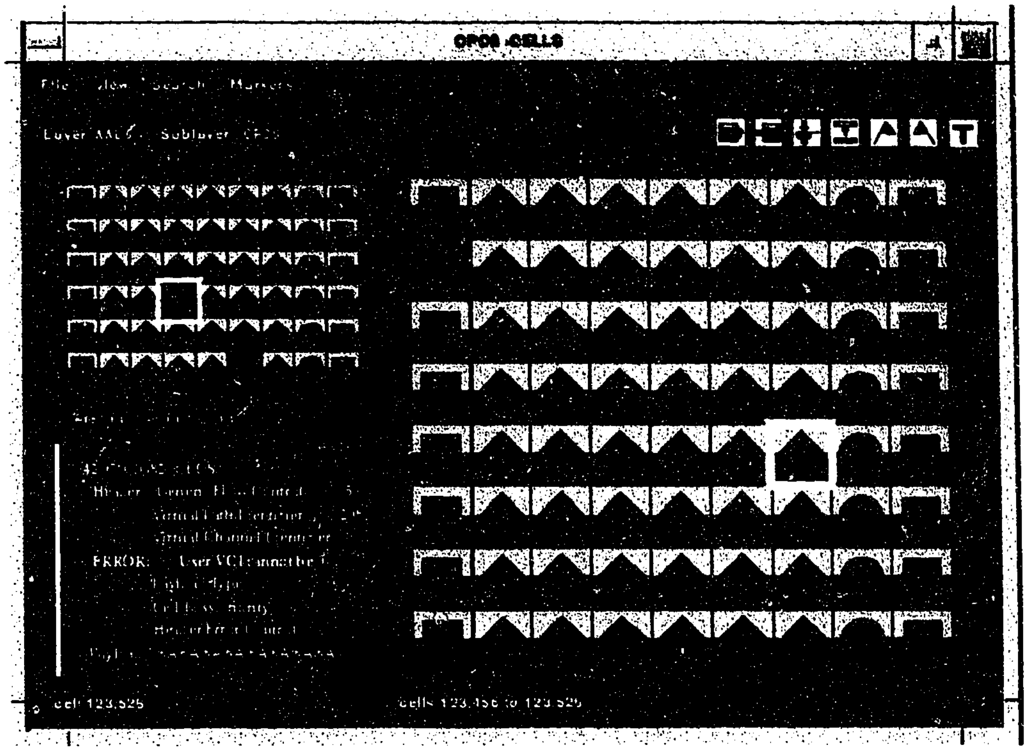

Figure 6: Alternative Approach to Time Line.

of detail. If each representation in this window does not correspond to a single data unit in the capture buffer, then, the graphical image in the detail window can be further expanded to show details of individual ATM cells or AAL PDUs. In that case, the overview window will be updated to show the level of details displayed in low level rectangle.

The main disadvantage of this technique is that it takes up more screen space than a time line. More importantly, this technique doesn't reveal the temporal and linear nature of the traffic data collected from the ATM network.

4.4 Cell Delay Visualization

As we described in the third chapter, cell delay is the amount of time taken for an ATM cell to traverse through a network. Cell delay testing is used to ensure the performance of ATM networks. Different types of services carried by a network have different tolerance to cell delay. The tolerance of cell delay by a certain service type is expressed by a number in seconds or nanoseconds, called the threshold. Cells with delay less than their threshold are normal cells. However, when a cell delay is greater than its threshold, this cell becomes intolerable. A large number of cells being highlighted demonstrates that there must be a problem in the network(s).

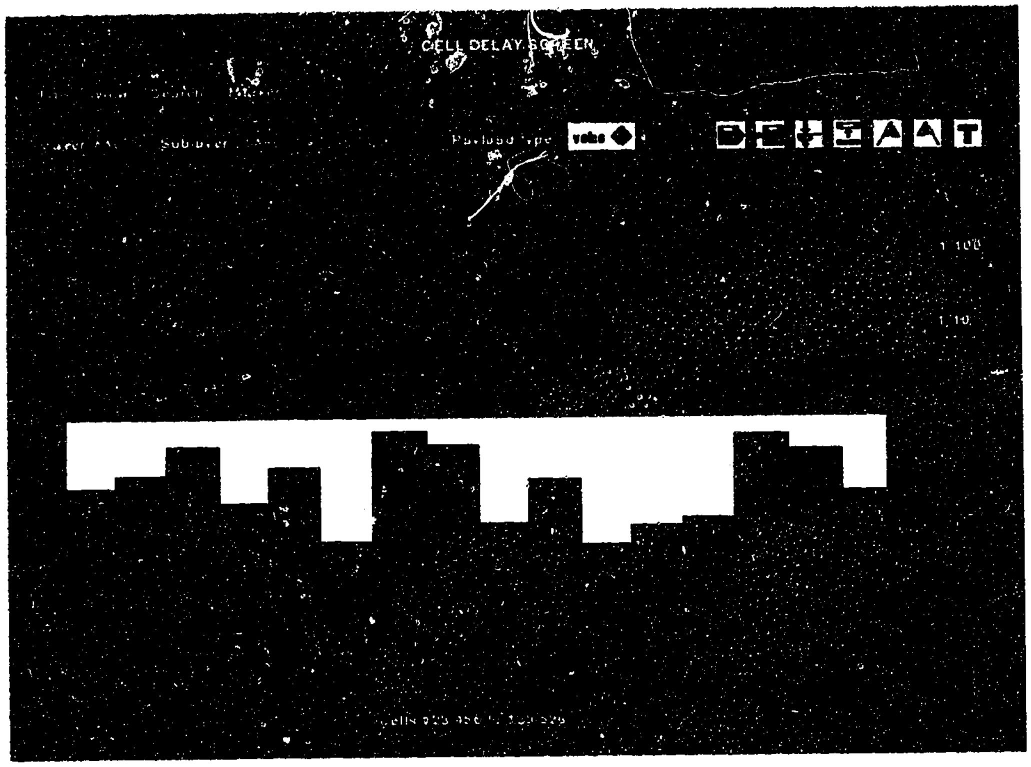

Figure 7: Cell Delay Visualization.

Cell delay visualization is one type of "Alarm Condition" visualization as we described earlier. As shown in Fig.7, hierarchical time lines are used to visualize cell delays, and all cells that have excessive amount of delay are highlighted in red. As described in the time line visualization section, the top time line gives the user an overview of the delay the cells in the capture buffer have gone through. This allows the user to see quickly the worst delay spot. In the bottom time line of our cell delay visualization, shape is used to code the amount of delay for individual cells. As one can see from Fig.7, a rectangle is used to code an individual ATM cell. The height of a rectangle represents the amount of time or delay taken to traverse through the network(s). If a cell delay is beyond its threshold, the rectangle representing it will be highlighted in red.

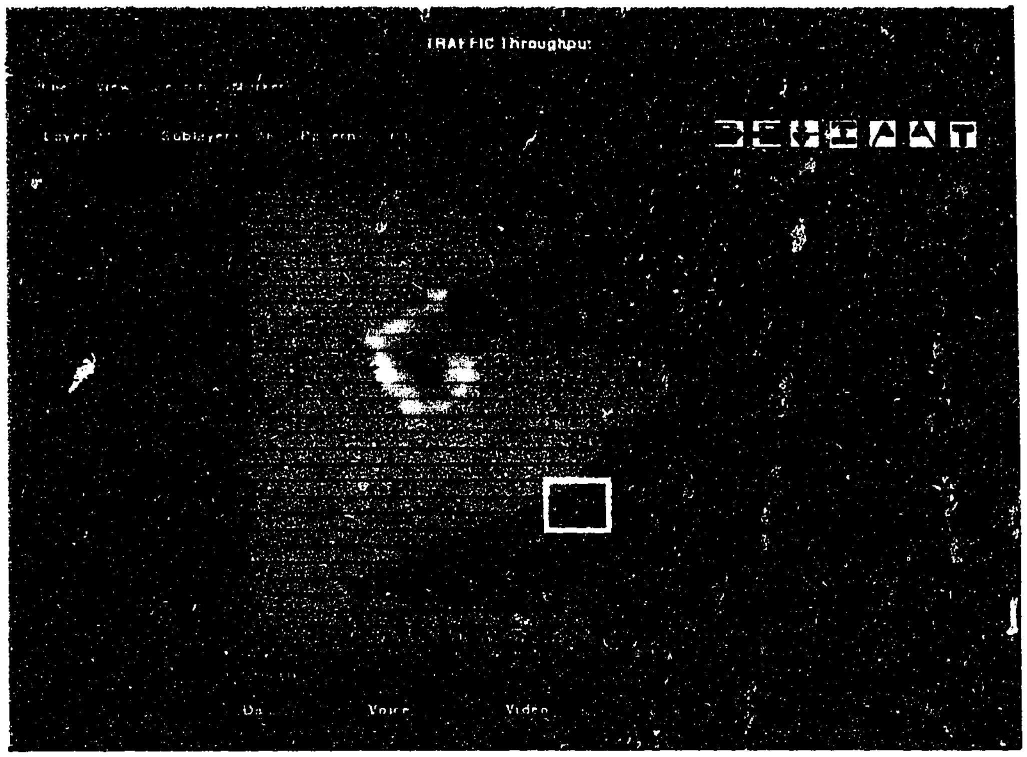

Figure 8: Network Traffic Visualization

4.5 Network Bandwidth Visualization

Network bandwidth allocation is an important data communication issue. The term bandwidth is used to describe the amount of transmission resources allocated to a service data. Our network traffic visualization techniques intends to show the bandwidth distribution of different types of network service data carried by a network at a given time. Fig.8 shows our traffic visualization technique from one of our design sketches. The vertical axis represents the time, and horizontal axis is the amount of traffic or bandwidth. Both shape and color are used to represent different types of data traffic. We use different color to represent different type of services, for example, voice, video, and computer data. The shape of the waves represents the bandwidth change along the time axis.

The waves with different color can overlap as shown in Fig.8. By clicking on a data icon along the bottom of the window, the bandwidth wave for that service data is brought up to the front. The proportion of the bandwidth distribution, at a certain time, among all data types is accessible to users. In our design, a probing box is used, shown in Fig.8, allowing the user to choose a specific area. The real value of the bandwidth distribution at the chosen spot is displayed at the right of the displaying window.

4.6 Generalizing our techniques

While the motivation for this research is quite specific, its results have a much wider significance. The protocol analyzer produces a time sequence of data values, and these data values don't have a spatial component. This type of data is frequently encountered in information visualization. In contrast to scientific visualization, very few techniques have been developed for their graphical display. Visualizing non-spatial data is a major open research problem, since there is no obvious mapping from the data to an image. Examples of this include the structure of software systems, and medical data (such as CAT scans). Users of these visualizations attempt to find patterns in the data, or rare events in large data sets. Good visualizations can significantly reduce the time required to perform these tasks and reduce the probability of error.

Some of the techniques developed in this project are general enough to be applied to data-rich visualization applications, especially any time sequence of non-spatial data. Our hierarchical time line technique can be used readily to organize large volumes of data, with the top time line representing an overview of the data space and the rest of time lines showing increased levels of detail. Our Cell Visualization technique may need adjustment to suit different applications. However, the idea of compact or reduced representation as well as shape and color encoding techniques can be used to represent almost any statistical data.

Chapter 5

Implementation

This chapter describes the implementations of visualization techniques proposed and described in the last chapter. Screen dumps of our prototypes are included to demonstrate our work. All the implementation designs use Object-Oriented methodologies, and coding is done using X11/OSF Motif in C++. The design of the main classes is included in Appendix A. All the data used in our prototype implementations was collected from the HP BSTS. Not all the proposed designs have been implemented, due to the time required to construct the prototypes and the similarities between the various techniques.

5.1 Time Line Implementation

All our time lines are implemented using OSF Motif's ScrolledWindow and DrawingArea widgets. The ScrolledWindow widget provides a viewport into a large viewing area, the DrawingArea widget. All the images on the time line are drawn on the DrawingArea widget, however, only the part that is inside the ScrolledWindow's viewport is visible. The viewport can be adjusted and scrolled back and forth by users to view all the images drawn on the DrawingArea widget, through the use of Scrollbars attached to the ScrolledWindow.

5.1.1 Time Represented By a Time Line

The first task in our time line implementation is to calculate the time span represented by each time line in the hierarchy. It determines the length of the capture buffer segment represented by each time line. We use T to represent the time between the arrival of the first and the last event in capture buffer, and Δt_i the time span for the $i'th$ ($i = 0, 1, 2, \ldots$) time line in the time line hierarchy. Since our top time line represents the content of the whole captured buffer, the time span of the top time line Δt_0, is equals to T. For the rest of the time lines in the hierarchy, Δt_i ($i \neq 0$) is a portion of its parent's time line [1]. It is the part of its parent time line covered by the scrollbar attached to the parent timeline's. This is how our time lines provide the zooming function, that is, a child time line zooms into its parent time line for the display of greater level of details.

Since the length of both the viewports of ScrolledWindows and the windows of DrawingAreas used in our time lines are known [2], the formula for the length of the $i'th$ time line's time span can be deduced. Let L and l be the length, in number of pixels, of the window of the DrawingArea and the viewport of the ScrolledWindow widget respectively. As illustrated in Fig.9, we have the following:

Level 0 (the topmost level):

$$\Delta t_0 = T$$

Level 1:

$$\frac{l}{\Delta t_1} = \frac{L}{\Delta t_0} = \frac{L}{T}$$

$$\Delta t_1 = \frac{T}{L} * l = T * \frac{l}{L}$$

Level 2:

$$\frac{l}{\Delta t_2} = \frac{L}{\Delta t_1}$$

$$\frac{l}{\Delta t_2} = \frac{L^2}{T * l}$$

[1] By parent time line, we mean the time line that is directly above the current time line in the hierarchy.

[2] We assume, in our calculation, that the length of the time line axis for the DrawingArea windows is constant for all time lines, as is the length of the ScrolledWidnows' viewports.

$$\Delta t_2 = T * (\tfrac{l}{L})^2$$

$$\vdots$$

Level n:

$$\Delta t_n = T * (\tfrac{l}{L})^n, \qquad (n = 0,1,2 \ldots)$$

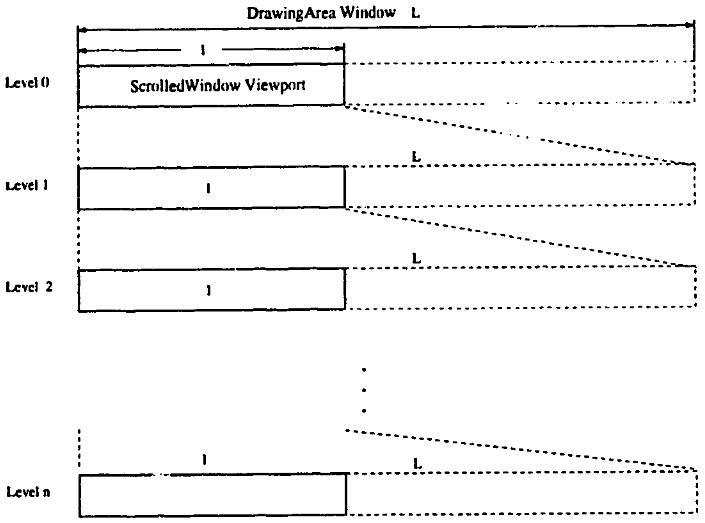

Figure 9: Hierarchical Time Lines

$\Delta t_i = T*(\frac{l}{L})^i$ is the formula to calculate the amount of time, in number of seconds, represented by a time line other than the one on the top of the hierarchy.

The portion of the capture buffer that a time line (other than the top time line) represents changes dynamically, as the user scrolls any of the time lines that are above it in the hierarchy. Consequently, the time represented by the *i'th* time line ($i \neq 0$) is a function of the scrollbar positions of all time lines above it and i. Since X/Motif keeps track of the current scrollbar position and stores it in the resource database, our program can get the current position of a scrollbar by accessing its resources every time it is scrolled, and then updates all the time lines below it in the hierarchy. The time represented by a time line determines the portion of the cells in the capture buffer to be displayed.

5.1.2 Time Represented by a Cell Image

In all of our implementations, rectangles are used to represent cell images along our time line in both compact and individual cell visualizations. For upper time lines, a rectangle image represents a group of cells. In the case of the bottom time line, one rectangle represents a single ATM cell or a higher level PDU.

Upper Time Lines

The time represented by a rectangle in upper time lines can be calculated easily. Since equal-sized rectangles are used to represent a group of cells, the length of time coded by a rectangle equals to Δt_i, the time span of the time line this rectangle is on, divided by the number of rectangles along the time line. The group of cells represented by a rectangle are simply those that arrived within the time coded by it. In the case of the top time line, a rectangle's position determines the group of cells it represents. For the middle time lines, however, the group of cells represented by a rectangle changes as the scrollbar of the top time line is scrolled. They are a function of both the position of the top time line scrollbar and the position of this rectangle on the middle time line.

The Bottom Time Line

ATM cells or higher layer PDUs are represented individually as rectangles on the bottom time line. The time stamp of an ATM cell determines the position, along the bottom time line, of the rectangle. The width of the rectangle approximates the duration of the cell, and its left edge corresponding to the cell's time stamp.

5.1.3 Data Density Scale

The number of cells represented by rectangles on a time line will not be the same, unless the ATM cells are evenly distributed in time, which is not the case in real life data communication. Both color and shape encoding for visualizing data density are implemented. Here, we discuss the color and shape scale used to implement these techniques. In order to represent numbers by color and shape, we need to compute a color scale and shape scale. To do so, we need to know the average number of cells arrived within the length (in time) of a rectangle, which is computed by assuming all cells are distributed evenly over time. We then use the scales to decide the thresholds, in the amount of hue value in the case of color coding and the number of pixels in shape coding, to use to code the number of cells arrived.

5.2 Prototypes

Two sets of prototypes have been produced to show how variations in our visualization techniques work. Time lines are oriented horizontally in one prototype, and vertically in the other. Horizontal time line seems to make more sense, since people generally view time as flowing from left to right, and not from up to down(or down to up?). However, since the text window, which we included in most of our designs, is scrolled vertically up and down, horizontal time line design have the problem of forcing users to work mentally in two directions. It may cause confusion and possibly fatigue, as users map the horizontal graphical scrolling to the vertical text display, especially if they work long period of time with the visualization. Fortunately, this problem does not exist in our vertical time line design, where both the time lines and the text window scroll vertically. Besides the difference in time line orientations, the cell density encoding techniques used on the two sets of time lines are different, with color coding on the horizontal time lines and shape coding on the vertical ones. The intention is to show how different encoding techniques work. In all of our prototypes, a three level time line hierarchy is used to visualize the contents of the HP BSTS capture buffer.

5.2.1 Horizontal Time Lines

Fig.10 is a screen dump of the horizontal time line prototype. The top time line provides an overview of the complete capture buffer, using a compact data representation. Each filled rectangles on top time line represents approximately 500 cells. By scrolling through it, users can see how 8Mbytes of data is distributed over time. Error cells are represented as follows: if there are relatively few error cells (we use 10 as threshold in our implementations), a red line is drawn for each cell in error. If there are large number of error cells, the border of the rectangles is painted red, since drawing a line for every error cell would cover the entire rectangle and hide the cell density information. As shown in Fig.10, a rectangle with red border around it means it contains quite few error cells.

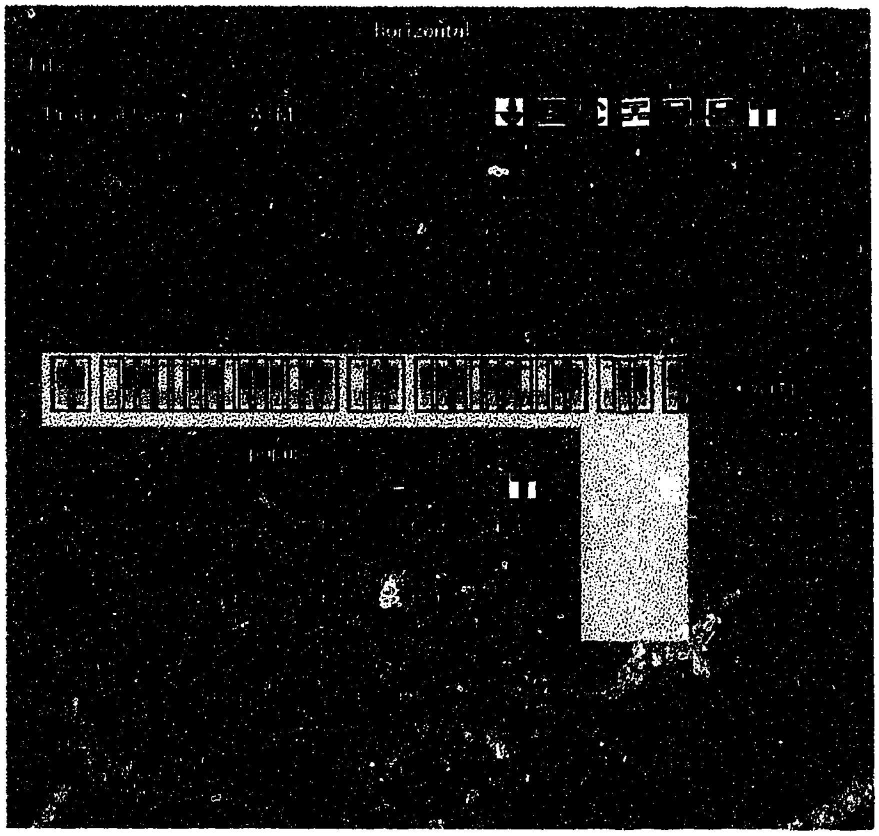

Figure 10: Horizontal Visualization with Popup Text Display

The middle time line represents the portion of the capture buffer covered by the top time line's scrollbar. Since the data are not as dense, each of the error cell is drawn as a red line. The position of such a line is computed from the cell's time stamp. This gives an impression of how the cells are distributed over the rectangle. The bottom time line paints each ATM cell as an individual rectangle image. The user can see the time relation of each cell, since the position of a cell corresponds to its time stamp. Error cells are painted using a red border around them. The height of the bottom time line in Fig.10 is larger than the other ones. The rest of the space below ATM cells is for a protocol stack visualization discussed later.

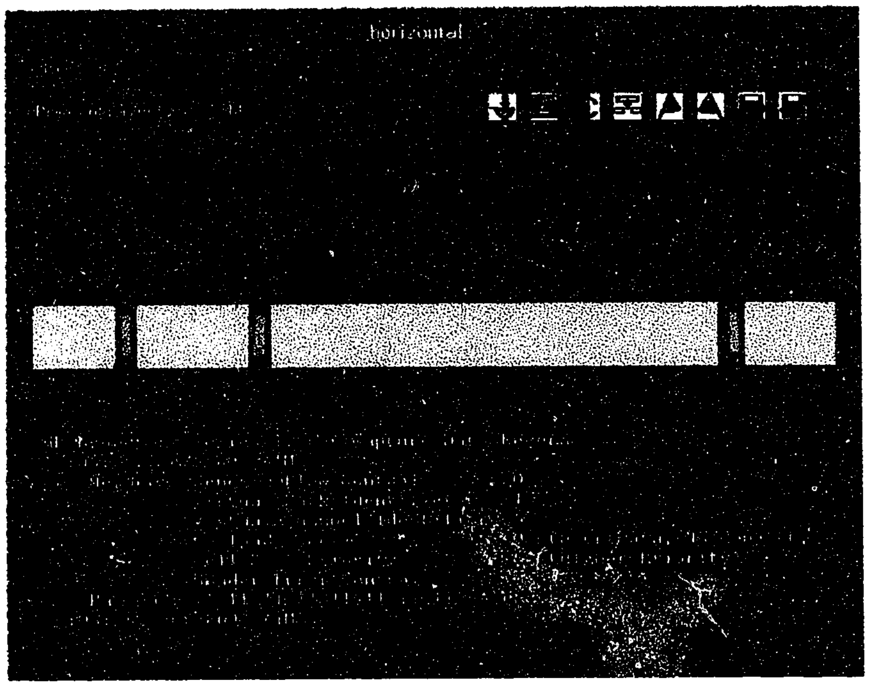

Figure 11: Horizontal Time Line Prototype (2)

Fig.11 is a screen dump of another prototype using horizontal time lines. The data used in this prototype is much sparser than that used in Fig.10. This is the reason why error cells on both the top and middle time lines are drawn as lines. The background color shows how cells are distributed along the time line. In the bottom time line, cells are painted in a different way. The inside color of an cell image represent the VCI field of its header, and outside color codes its VPI field. For an error cell, the outside is painted red.

Both Fig.10 and Fig.11 provide users with a text window, so a textual representation of the data can be accessed on demand. The difference is that a popup window is used to display the text in Fig.10. The user can pop it up by using clicking on the "T" icon in right top corner of the main screen. The same icon on the text display is used to pop it down, when it is not needed. The text window is made static in Fig.11, and the textual representation of the data is always displayed on the screen. Static text window takes up more display space than the popup one. It depends on the users to evaluate how they prefer the text to be displayed.

5.2.2 Vertical Time Lines

Fig.12 and Fig.13 are two screen dumps of our vertical time line implementation. As can be seen from these figures, all the time lines in this design are laid out vertically from left to right. The left most time line corresponds to the top or first time line in our time line hierarchy. It is called the first time line in the rest of this section to avoid possible confusions, since it is not located above the other time lines. The next time line to the right is the second or middle time line. The bottom time line appears at the bottom right of the screen, with adjustable time line length (height). Since both the bottom time line and the text window scroll vertically, they can be paned together using OSF Motif's panedWindow widget. By doing this, these two windows can share the space allocated to them. Users can enlarge one window and shrink the other, by dragging the sash up and down. For example, when users focus on the individual cell display, they can enlarge the bottom time line window by dragging the sash up. Similarly, if users want to read the actual text of ATM cells, they can drag the sash down and shrink the bottom time line window. This technique has the advantage of allowing two windows to share the same space, which is important because of the very limited display space on the current HP BSTS monitor. As shown in Fig.12, the size of the bottom window is larger than the text window, and Fig.13 shows the opposite.

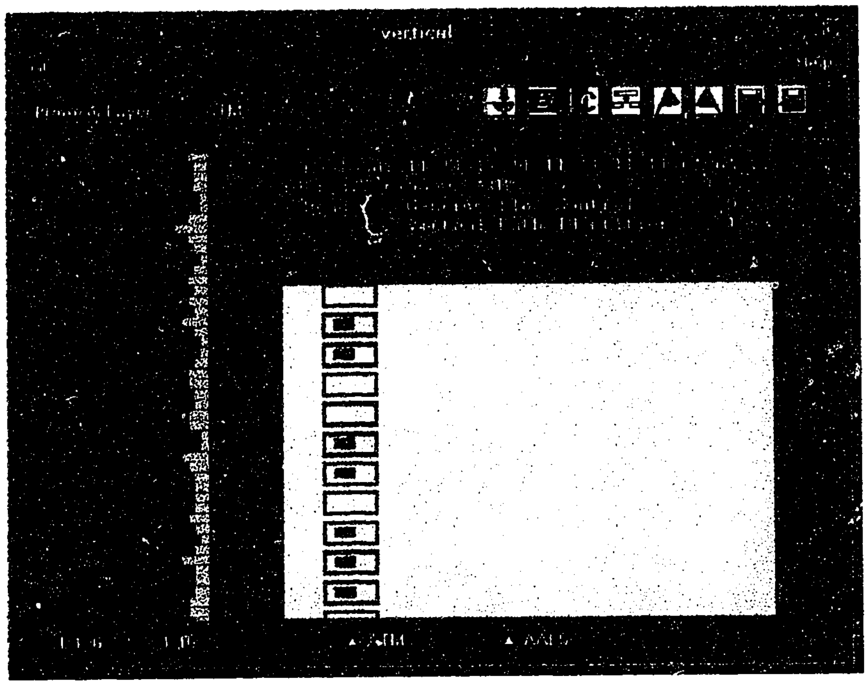

Figure 12: Vertical Time Line Prototype

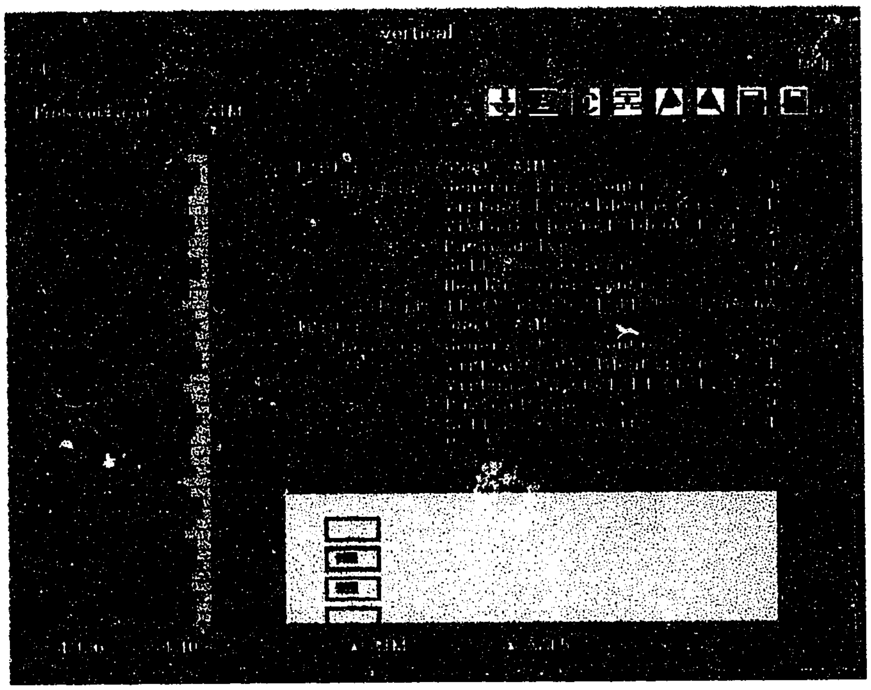

Figure 13: Vertical Time Line Prototype

shape encodes cell density

As one can see from both Figures, the shape or length of the rectangles is used to represent cell distributions on the first two time lines. For each rectangle in those two time lines, its width is drawn according to the number of cells that arrived within its length (height) in time. A blue color is used to represent normal cells, and red is used to paint those cells that are in error. Since the top time line represents all the data in the capture buffer, and the second time line represents only a part of the top time line, the data density on the first time line is greater than that of the second. This is the reason why darker blue is used to paint images on the first time line and lighter blue for the second time line. In addition to color, the approximate number of data packets represented by a single rectangle is displayed. In Fig.12, the number below the first time line, "1:126" means that the full width of a rectangle in that time line represents approximately 126 cells. The label "1:10" shows that the full width of a rectangle in the second time line represents 10 cells. It serves as a second cue besides color to show that the first time line is much denser than the second. The purpose is to avoid possibly misleading and confusing users, especially for those who are color-blind.

5.2.3 Protocol Stack

The first step in protocol stack visualization is to decide which part of the protocol stack to visualize. Users choose a protocol (layer), by using a X/Motif's Option Menu as shown at the top of the screen in Fig.14. All the PDUs at the chosen protocol layer down to the ATM layer, will be displayed. The purpose is show not only the PDUs of the chosen protocol, but also how these PDUs are constructed from lower layer PDUs and ATM cells.

Fig.14 is a screen dump of our protocol stack visualization, with AAL-5 as the chosen protocol. Protocol stack visualization happens in the bottom of the time line. Because there is not enough detail to be shown in higher time lines in the hierarchy, and each data packet is not differentiated as either an ATM cell or a higher level PDU.

The reassembly of AAL-5 PDUs is based on the virtual channel. AAL-5 PDUs transmitted from different VCs belongs to different services, and will be reassembled separately. A bit in the PT field in ATM cell headers is used to delineate AAL-5 PDUs within the same VC. It indicates that the current ATM cell is the last cell for the current AAL-5 PDU. Since there is no multiplexing within a virtual channel, one AAL-5 PDU can be distinguished from subsequent ones using the PT field in the ATM cell header.

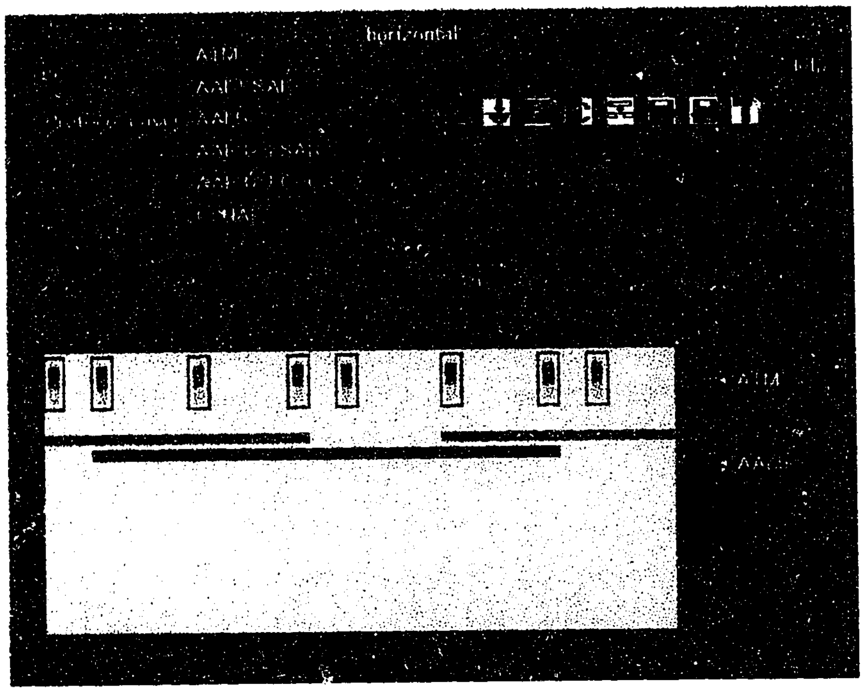

Figure 14: Protocol Stack Visualization

As shown in Fig.14, ATM cells are drawn at the top of the bottom time line window, and AAL-5 PDUs are drawn below the ATM cells. As we mentioned earlier, the left edge of an ATM image on the bottom time line corresponds to its arrival time stamp, and its length along the time line is approximately its duration. This is also true for AAL-5 PDUs. The arrival time of an AAL-5 PDU is the arrival time of the first ATM cell containing its data, and its duration starts with the arrival of the first

ATM cell and continues until that of the last ATM cell it is reassembled. That is, the left edge of an AAL-5 PDU image is painted with the same x value as its first ATM component, and its right edge lines up with the right edge of the last ATM cell it is reassembled from.

Color in addition to position is used to show the mapping from several ATM cells to an AAL PDU. AAL PDUs are painted using the same color as that of the VCI field of the ATM cells they are reassembled from. Thus, AAL-5 PDUs with different VPI are painted using different y value. The combination of shape and color shows how an AAL-5 PDU is mapped from ATM cells.

Protocol stacks will grow, if there are more protocol layers between the chosen protocol layer and the ATM layer. In the case of the AAL-5 protocol, it does not have a SAR sublayer, Therefore, AAL-5 CS_PDU and ATM are the only two layers to visualize. However, the service layer that is above AAL-5 could be chosen for visualization. In that case, the protocol stack visualization will include SDUs at the service layer, AAL-5 PDUs and ATM cells, with SDUs drawn below AAL-5 PDUs. As described earlier, all our time lines are implemented using the OSF Motif DrawingArea widget enclosed by a ScrolledWindow widget. When there are more protocol layers to visualize, the height of the bottom time line window could grow and exceed that of the ScrolledWindow viewport. A vertical scrollbar in that window will appear to allow the user scrolling up and down to view the whole DrawingArea window. With this design, our bottom time line can hold as many PDUs and ATM cells as a user can possibly choose to visualize.

5.3 Cell Visualization Implementation

In all our prototypes, both shape and color are used to code important information in an ATM cell or higher layer PDU. At the ATM protocol level, a rectangle is used to represent the header part of an ATM cell, since only this part of the cell is decoded at that layer. The ATM payload is relatively not as important and is dropped from our cell visualization. This is why the border of an cell with HEC error is painted red.

Among all fields in the ATM cell header, VCI and VPI are the most important with respect to the function of ATM protocol layer. This is the reason why they are always highlighted in color. In the case of protocol stack visualization, the VCI/VPI serves as the information used to map ATM cells to AAL-5 PDUs. For AAL protocols other than AAL-5, other fields in the ATM cell header may have to be highlighted to show the mapping between ATM cells and their PDUs. For instance, the sequence number field embedded in most of the AAL PDUs is one important field to be highlighted. It is used to maintain cell ordering. If sequence error occurs in such a protocol PDU, the sequence number field should be painted red to indicate the error.

Chapter 6

Evaluation

In this chapter, we evaluate the visualization techniques proposed and implemented in the thesis. It outlines the main purpose of the visualization techniques, their strengths and weaknesses, and the range of the design space. Most of the evaluation is based on our interactions with HP/Idacom engineers and feedbacks from HP BSTS customers.

6.1 Achievements

One of the most important achievements of our visualization techniques is that they provide to the user with both an overview and a detailed display of the capture buffer at the same time, using very limited screen space.

At the top of the hierarchy, our time line visualization technique provides the user with a summary of all the data in the capture buffer. This gives the user context information and helps them to orient themselves while navigating through the large data space. As pointed out at the beginning of this thesis, context information is important in large information visualizations. Its importance has been acknowledged in recent visualization research. In the case of ATM traffic data/events, showing all the data on a single time line also correlates events occurring over a wide time space.

Compact representation is one of the key ideas in our visualization. Using a single graphical image to represent a group of cells, it is possible to display the whole capture buffer on a single time line.

Graphical encoding, with shape representation and color highlighting is important. First of all, it is more illustrative of information. Secondly, it draws attention to patterns and unusual events. The ATM traffic data tends to be repetitive. For example, each data packet (or event) has a header and a payload, plus a time stamp. Without graphical coding and highlighting it is hard to spot patterns, especially among large data collections.

At the bottom time line, the user can examine individual events in detail. Our design of the bottom time line can show not only ATM cells, but also higher level PDUs at the same time. This shows the user the mapping from ATM cells to higher level PDUs, or from one level of PDUs to another. Our protocol stack visualization allows the user to trace the origin of errors occurring at higher protocol layer.

All our time lines show how data is distributed over time. This reveals the underlying network traffic load, which is a crucial factor in telecommunication. Many network device performance problems are related to it. In fact, many network tests are carried out under different traffic load to ensur high quality products. The purpose is to see how they perform under different traffic conditions. For example, an ATM switch may start to function abnormally at a certain traffic load. By showing data on time lines, our visualization techniques help the user to detect errors and unusual patterns among the data.

6.2 Weakness

One limitation of our time line is the need to scroll them in order to view the complete capture buffer. This is due to the large volume of the data to be visualized.

6.3 Time Lines

- Orientation of time lines

  As we pointed out, horizontal time lines are natural and comply to human habit, since people generally view time as flowing from left to right. The problem with this design is the different scrolling directions of the text display window and the time lines. This problem may be improved by using asynchronous scrolling between these two displays. For example, when the user scrolls through one of the time lines horizontally, the text does not scroll. The text window will update to the current position of the time line after the user releases the mouse button and ceases scrolling.

  One of the advantages of the vertical time lines is that they naturally avoid the above problem. Because the scrolling directions agree for both the time lines and the text window, they can be paned together and share display space with each other. This design allows us to include protocol stack visualization and use a static text window, using the same amount of screen space as the horizontal time line prototype. This is an attractive feature because the current HP BSTS has very limited screen display space.

- Level of Hierarchies

  We use three levels of time line hierarchy to visualize the whole capture buffer. More layers will provide more levels of detail, but it may have the problem of too many layers to scroll through. The user has to move several intermediate time lines before reaching the bottom time line to see the individual cells.

6.4 Cell Visualization

- Data Density Coding

  Comparing the two graphical encoding techniques for data density, shape works better than color. For representing quantitative information, the length of a graphical image has better visual impact than the color or texture. This has been confirmed by HP/Idacom engineers and the HP BSTS customers.

In both of our prototypes, heuristics are used to estimate the number of events arriving during a time interval. This technique works well for most cases. However when data are close to evenly distributed, the technique may not work well. If data are evenly distributed, there is little color variation between the rectangles. The user may not be able to see the difference. The solution to this problem is to use dynamic graphics. That is, allow users to dynamically choose the color range for different types of data.

- Individual ATM/PDU visualization

  There are many ways that the individual cells can be visualized. The main data that we want to display in these visualizations is the information in the headers. The payload portion doesn't appear to be as important. In our design, not all the fields in the header are highlighted, the fields to be highlighted depend upon the current visualization interest. In this thesis, most of the visualizations address the VCI/VPI fields, since our main interested is in ATM cells and AAL-5 PDUs. The next field in the cell header to show is the Payload Type field. It will introduce the issue of congestion control and cell loss.

  One of the main graphical variable we have to deal with is shape. Most of the shapes that we have investigated are symmetric about the y axis (perpendicular to the time line. We call the time axis the x axis.), so there is no bias along the time axis. The other major graphical variable we work with is color. Each of the cell displays can be divided into separate regions that represent different header fields. Each of these regions can then be color coded to represent the header value. Our current prototypes use perceptually based color spaces, so equal differences in color values represent equal differences in color perception. This provides a more faithful encoding of the field values, and also highlights major differences. There are two reasons for this. First, the legend would take up screen space that is better used for displaying cells. Second, the color scheme is mainly used to highlight the differences in field values.

6.4.1 Text Display

In our prototypes, both static text display and pop up text window are implemented. According to the feedbacks from some of the HP BSTS customers, a static text window is desirable. Because from time to time, the engineers want to refer to the real value of either an ATM cell or a higher layer PDU. It is something that the users want to be there of all times, so they can access it effortlessly. HP/Idacom engineers also pointed out that another reason they prefer static text display is because there are already too many pop up windows in the current HP BSTS screen design.

Chapter 7

Future Work

7.1 Conclusion

Time line visualizations and cell visualizations are two of our main approaches to visualization, with each highlighting different aspects of the data. The combination of both techniques makes it possible to overview the whole capture buffer and at the same time be able to examine individual data packets in detail. Most importantly, they reveal the underlying traffic patterns to users, and allow them to spot interesting events and network communication anomalies. Cell delay visualization and network bandwidth visualization focus on specific issues in ATM network testing. Cell delay visualization shows the amount of delay taken by ATM cells passing through one or several networks. Network bandwidth visualization allows the user to visualize, at a given time, the network traffic throughput and bandwidth distribution among different types of services. Both cell delay visualization and network bandwidth visualization are meant to show the performance and quality of services offered by an ATM network.

Two prototypes were implemented based on our time line visualization and cell visualization techniques. Different graphical representation techniques are used to demonstrate and compare how these techniques work. We concludes with our contributions to the area of information visualization: that is the techniques developed, especially hierarchical time line visualization, graphical representation, and highlighting can be applied to any time sequence of non-spatial statistical data.

7.2 Future Work

As we mentioned in the beginning, this thesis is part of a visualization project for the HP BSTS. The work accomplished in this thesis belongs to the first phase of this ongoing project. There are several areas where more work needs to be done. We will discuss these areas in the rest of this section.

7.2.1 Sequence Number and Related Errors

As we described before, errors can be classified into cell header errors, cell loss, cell misinsertion and sequence errors. We have described HEC errors as well as how they are visualized. Cell loss, cell misinsertion, and sequence error are all related to sequence number, which is embedded in AAL SAR_PDUs. Sequence number fields in AAL SAR_PDUs are used to detect cell loss and cell misinsertion. At the sender side of a transmission a sequence order is maintained and inserted into the sequence number field of AAL SAR_PDUs. At the receiver side of a transmission, ATM cells are reassembled into AAL SAR_PDUs and the sequence number in each SAR_PDUs is checked. Missing a certain sequence number means a cell is lost. An out of order sequence number is interpreted as the result of a cell misinsertion from other virtual channels. Due to the importance of the sequence number, a sequence number protection field in AAL-2 SAR_PDU, and CRC error checking in the other AAL SAR_PDUs are used to protect sequence numbers from being damaged during transmission. Since either sequence number protection or CRC check sums can only correct few error bits (usually two), it is possible for the sequence number field to be corrupted, but unable to be corrected. This is how a sequence number error occurs.

Our cell visualization techniques should be extended to represent cell loss, cell misinsertion and sequence number errors at AAL PDU level. Either color or shape or both can be used to highlight these sequence number related errors and allow the user to spot them easily.

7.2.2 Protocol Stack Visualization

Future work on protocol stack visualization should focus on the reassembly of ATM cells to AAL-3/4 PDUs, which is the most complicated AAL protocol. AAL-0, AAL-1 and AAL-2 are simple protocols to visualize. The visualization technique developed for AAL-3/4 and AAL-5 can be used to easily visualize AAL-0, AAL-1 and AAL-2.

The functions performed by both AAL-3/4 and AAL-5 concern the transport of data services, which is regarded as the main initial application of ATM. The difference between the AAL-3/4 protocol and AAL-5 is that different types of service data are transmitted using the same virtual channel in AAL-3/4. Therefore, in AAL-3/4 SAR_PDU frame design, the header contains a Message ID (MID) field. It is used to identify which AAL-3/4 SAR_PDU belongs to an AAL-3/4 CS_PDU. Another field in the SAR_PDU header contains one of Begin Of Message (BOM), Continue Of Message (COM) and End Of Message (EOM). This is used to delineate an AAL-3/4 CS_PDU.

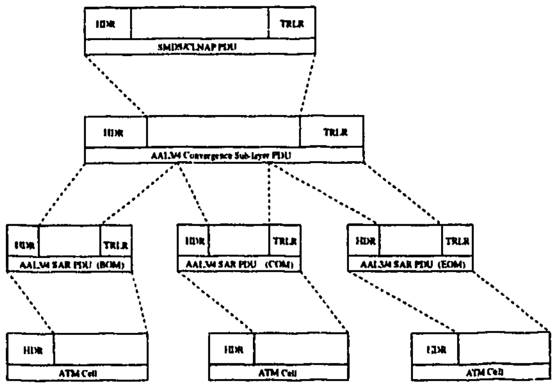

Figure 15: AAL3/4 protocol data unit.

As shown in Fig.15, AAL-3/4 protocol PDUs include both SAR_PDUs and CS_PDUs. Protocol stack visualization from ATM to AAL-3/4 should include not only the mapping from ATM cells to AAL-3/4 SAR_PDUs, but also the mapping from AAL-

3/4 SAR_PDUs to AAL-3/4 CS_PDUs. The mapping from ATM cells to AAL-3/4 SAR_PDUs is an one ATM cell to one AAL-3/4 SAR_PDU mapping. The mapping from AAL-3/4 SAR_PDUs to AAL-3/4 CS_PDU is many to one. Several SAR_PDUs map to a single CS_PDU based on the MID and BOM, COM and EOM fields embedded in SAR_PDUs. The mapping from SAR_PDUs to CS_PDUs is similar to that of ATM cells to AAL-5 PDUs. However, since there is an extra mapping from ATM cells to AAL-3/4 SAR_PDUs to visualize, the technique developed for AAL-5 must be extended to show the mappings all the way from ATM cells to AAL-3/4 CS_PDUs, and how a ATM cells correlated to an AAL-3/4 CS_PDUs.

7.2.3 Stream Comparison

A number of tests involve taking a stream of cells, feeding them through a device, and collecting the device's output as a second stream of cells. The user then wants to determine if the device has properly processed the input stream. At the present time, there is no easy way of making this comparison, even though the input and output streams can both be captured. We need to develop visualization techniques that allow the user to compare two streams of cells to determine if the device is functioning [illegible]y. These visualization techniques should highlight the timing relationships between the two streams, dropped cells, inserted cells and other important properties.

7.2.4 Signaling

Signaling is used to do network control and to establish and release connections ahead of service data transmission. Signaling protocols also let the user negotiate dynamically for parameters such as QoS, VP/VC number allocation, etc, over a network.

A signaling protocol works by exchanging variable length messages. Messages can be grouped into several basic categories, including "call establish", "call clearing" and "supplementary services". Examples of call establish message are "CALL PROCEEDING", "CONNECT" and "CONNECT ACKNOWLEDGE". Examples of call clearing include "DISCONNECT", "RELEASE", and "RELEASE COMPLETE". Message for supplementary services can be "NOTIFY", "RESUME" and "SUSPEND".

Each signaling message has a number of Information Elements (IEs) fields and a message type associated with it. Each IE field contains parameter values for the circuit attributes being negotiated. Some IEs are mandatory in a particular kind of message while others are optional. Signaling visualization techniques should consider the representation of the different types of signaling messages. Each of the IE fields, especially the mandatory IEs, should be highlighted and allow the user to differentiate message parameter values.

7.2.5 Real Time Visualization

Our current visualization techniques only address captured data visualization or off-line static visualization. Real time on-line visualization should be considered eventually. As the B-ISDN/ATM standards are continually evolving with additional physical media and upper layer services, hard-wired ATM networks will soon become reality. With hundreds events per second, real time visualization will be necessary to help the user do network analysis and testing.

Appendix A

C++ Objects

A.1 Main Objects and their associated Interfaces

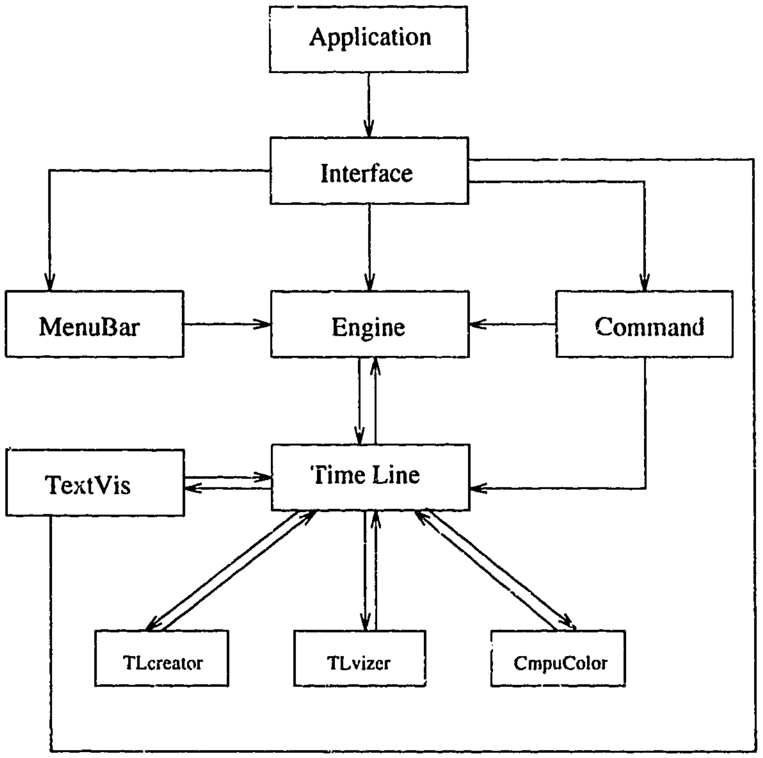

Figure 16: Object-Oriented Design

A.2 The definition of Main Classes

- class::Application

  Application object is responsible for initializing the Xt toolkit, connecting to the X server and the usual X bookkeeping.

- class::ATMVis

  The ATMVis object acts like a canvas to hold all the time lines buttons, icons, and display windows, which are created as its children.

- class::Engine

  The Engine object serves as the engine of the whole visualization. It does not have a user interface. Its role is to read in the data file and start time line display.

- class::TimeLine

  The Time line object, as its name implies, serves as the time line in our visualization. A time line is created using the X11/Motif scrolledWindow widget and the DrawingArea widget. Its function is to draw individual cells or cell images, and update the time line as users scroll through a time line.

- class::TextDisplay

  This object is responsible for displaying a textual representation of the data collected. Two versions of the TextDisplay object exit. In our horizontal time line prototype, TextDisplay is simply a scrolledWindow widget with a Text widget as the child. In the horizontal time line prototype, a Motif applicationShell widget is used as the parent of the above version. The text file is not displayed until the user clicks on the text icon.

A.3 Engine Object

```
#ifndef ENGINE_H
#define ENGINE_H

class HorizontalVis;
class Engine {
      protected:
                    Line _atmLine;
                    Line _aalLine;
                    HorizontalVis *_vis;
      public:
                    Engine( HorizontalVis * );
                    const char *const className() { return("Engine"); }
                    void startVisualization();
                    void abort();
                    char *get_cellfile();
                    void read_atm_cells( char * );
                    void read_aal5_cells( char * );
                    double time_diff(int, int, int, int);
}
#endif
```

A.4 TimeLine Object

```
#ifndef TIMELINEOBJ_H
#define TIMELINEOBJ_H


class ATMVis;
class ColourObj;
class TimeLineObj : public UIComponent {
    friend class Engine;
    private:
            int _id; // scrolling bar id (0,1,2,3)
            int _drawAreaLength; //length of the time line
            int _width; // width of the scrolling window
            int _height; // height of the scrolling window
            Window _window; // drawable = XtWindow;
            Display *_dpy; // = XtDisplay;
    protected:
            ColourObj *_color;
            ATMVis *_vis; // a copy of hvis
            Widget _draw; // draw area widget
            Line _line; // atm cell timeline
            Line _aalLine;
            line create_time_line(Tline, Line, Line):
            void draw_lineCallback(Widget, XtPointer, XtPointer);
            void draw_cellCallback( Widget, XtPointer, XtPointer);
            void scrollCallback( Widget, XtPointer, XtPointer );
            void draw_cells();
            void draw_line();
            void scrollCB();
    public:
            TimeLineObj(Widget, char *, HorizontalVis *, int);
             TimeLineObj();
            const char *const className() { return("TimeLineObj"); }
};
#endif
```

Bibliography

[1] Hary Baeverstad. Engineering and scientific visualization using high-performance graphics workstations. *IEEE Proceedingss on*, pages 328 - 333, 1989.

[2] Richard A. Becker, William S. Cleveland, and Allan R. Wilks. Dynamic graphics for data analysis. *Statistical Science*, 2(4):355 - 395, November 1987.

[3] Richard A. Becker, Stephen G. Eick, Eileen O. Miller, and Allan R. Wilks. Dynamic graphics for network visualization. *IEEE Conference on Visualization'90*, pages 93 - 96, 1990.

[4] Jeff Beddow. Shape coding of multidimensional data on microcomputer display. In Arie Kaufman, editor, *Proceedings of the first IEEE Conference on Visualization*, pages 238 - 246, 10663 Los Vaqueros Circle, Los Alamitos, CA 90720-1264, October 1990. IEEE Computer Society, IEEE Computer Society Press.

[5] Andreas Buja, Catherine Hurley, and John Alan McDonald. Elements of a viewing pipeline for data analysis. In W.S. Cleveland and M.E. McGill, editors, *Dynamic Graphics for Statistics*, Belmont, California, 1988.

[6] Andreas Buja, John Alan McDonald, John Michalak, and Werner Stuetzle. Interactive data visualization using focusing and linking. *IEEE Conference on Visualization'91*, pages 156 - 163, 1991.

[7] Richard Chimera. Value bars: An information visualization and navigation tool for multi-attribute listings. In *IEEE CHI'92*, pages 293 - 294, 1992.

[8] Stephen G. Eick. Dynamic graphics for data analysis. *Graphically Display Text*, 3(2):127 - 142, 1994.

[9] Stephen G. Eick, Joseph L. Stephen, and Eric E. Summer. Seesoft a tool for visualizing line oriented software statistics. *IEEE Transactions on Software Engineering*, 18(11):957 - 968, November 1992.

[10] George W. Furnas. Generalized fisheye views. *CHI'86 Proceedings*, pages 16 - 23, April 1986.

[11] Murray HIll. Dynamic graphics for network visualization. In Arie Kaufman, editor, *Proceedings of the first IEEE Conference on Visualization*, pages 93 - 95, 10662 Los Vaqueros, Los Alamitos, CA 90720-1264, October 1990. IEEE Computer Society, IEEE Computer Society Press.

[12] Brian Johnson. Treeviz: Treemap visualization of hierarchically structured information. In *IEEE CHI'92*, pages 369 - 370, 1992.

[13] Jock D. Mackinlay, George G. Robertson, and Stuart K. Card. The perspective wall: Detail and context smoothly integrated. *CHI'91 Proceedings*, pages 173 - 194, 1991.

[14] John Alan McDonald and Werner Stuetzle. Painting multiple views of complex objects. *SIGPLAN Notices*, 25(10):245 - 257, October 1990.

[15] Penny Rheingans. Color, change, and control for quantitative data display. *IEEE Conference on Visualization'92*, pages 252 - 259, 1992.

[16] Geoge G. Robertson, Jack D. Mackinlay, and Stuart K. Card. Cone trees: Animatied 3d visualizations of hierarchical information. *CHI'91 Proceedings*, pages 189 - 194, 1991.

[17] George G. Robertson and Jock D. Mackinlay. The document lens. *Proceedings of the ACM Symposium on User Interface Software & Technology*, pages 101 - 108, November 1993.

[18] Philip K. Robertson and John F. O'Callaghan. The generation of color sequences for univariate and bivariate mapping. *IEEE Computer Graphics and Applications*, pages 24 - 32, February 1986.

[19] Manojit Sarkar and Marc H. Brown. Graphical fisheye views of graphs. *Proceedings of CHI'92*, pages 83 - 91, 1992.

[20] Manojit Sarkar, Scott S. Snibbe, Oren J. Tversky, and Steven P. Reiss. Stretching the rubber sheet: A metaphor for viewing large layouts on small screens. *Proceedings of the ACM Symposium on User Interface Software & Technology*, pages 81 - 91, November 1993.

[21] Doug Schaffer, Zhengping Zuo, Lyn Bartram, John Dill, Shelli Dubs, Saul Greenberg, and Mark Roseman. Comparing fisheye and full-zoom techiques for navigation of hierarchically clustered networks. *Graphics Interface'93*, pages 87 - 96, 1993.

[22] Anselm Spoerri. Infocrystal. *IEEE Proceedings of Visualization'93*, pages 150 - 157, 1993.

[23] Bellcore Paul Tukey, Laurence T. Maloney, John R. Pani, and Stuart Smith. Human perception and visualization. In Arie Kaufman, editor, *Proceedings of the first IEEE Conference on Visualization*, pages 401 - 406, 10662 Los Vaqueros Circle, P.O.Box 3014, Los Alamitos, CA 90720-1264, October 1990. IEEE Computer Society, IEEE Computer Society Press.