ABSTRACT

Title of Thesis: TEMPORAL DATABASES IN NETWORK MANAGEMENT

Degree Candidate: Ajay Gupta

Degree and year: Master of Science, 1998

Thesis directed by: Professor John S. Baras
Department of Electrical Engineering

Computer networks are becoming a crucial part of a business' lifeline, therefore, managing these networks, and ensuring they remain operational, is an increasingly important task. This thesis discusses issues involved with performing network management, specifically with means of reducing and storing the large quantity of data that networks management tools and systems generate.

The value of the network management data collected diminishes as the data ages. The value that remains is in the trend that the data outlines of the macroscopic behavior of the network. This thesis proposes an algorithm which highlight these trends and, in the process, causes a significant data reduction. The proposed algorithm's reduction methodology is compared with standard data reduction techniques. It is further shown that the processed data is suited for storage in a temporal database. The benefits of storing time series data, such as the network

management data discussed (i.e., Link Utilization, CPU Load, and Device Reachability), in a temporal databases is demonstrated by storing the data in both a relational Oracle 7 and a temporal TIGER database, and examining results of queries against both databases.

TEMPORAL DATABASES
IN
NETWORK MANAGEMENT

by

Ajay Gupta

Thesis submitted to the Faculty of the Graduate School of the
University of Maryland, College Park in partial fulfillment
of the requirements for the degree of
Master of Science
1998

Advisory committee

Professor John S. Baras, Chairman/Advisor
Professor Wesley Lawson
Dr. Mathew Scott Corson
Dr. George Mykoniatis

ACKNOWLEDGMENTS

I wish to express my sincere gratitude to several individuals who made this thesis possible. Firstly, Dr. John Baras for giving me the opportunity and providing the facilities to work on this thesis throughout my graduate studies. I'd like to thank Professor Wesley Lawson, Dr. Mathew Scott Corson, and Dr. George Mykoniatis for joining my defense committee. Farooq Anjum's assistance in the simulations proved invaluable and irreplaceable. Vineet Brimani helped greatly in the study of temporal databases. The technical assistance given by Professor Michael Boehlen and the TIGER team at Aalborg University with the use and operation of TIGER is greatly appreciated and saved considerable development time. I also appreciate Professor Boehlen's inclusion of the Relational, Temporal database script in TIGER's web site.

I'd like to thank all my friends in the SEIL lab group and the Network Management research group, especially Dr. George Mykoniatis, who provided help, assistance and much needed comic relief throughout the course of this work.

This work was supported by the Center for Satellite & Hybrid Communication Networks, a NASA Commercial Space Center (CSC), under NASA Cooperative Agreement NCC3-528.

Prepared through collaborative participation in the Advanced Telecommunications/Information Distribution Research Program (ATRIP) Consortium

sponsored by the U.S. Army Research Laboratory under the Federated Laboratory Program, Cooperative Agreement DAAL01-96-2-0002.

The views and conclusions contained in this document are those of the author and should not be interpreted as representing the official policies, either expressed or implied, of the Army Research Laboratory or the U.S. Government.

This work was supported by Lockheed Martin Telecommunications, Sunnyvale, CA.

Finally, this work is more a testament to my parent's will than my own efforts. Without their constant support and challenge it would not exist today.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1

Introduction

In recent years, the computer network has transformed from a tool used to speed business processes to the very lifeline of the enterprise. An increasing number of businesses, from the fast-paced such as financial services, to the less real-time, such as gasoline service stations, are becoming reliant on computer networks for their day-to-day operation.

It is ever becoming a higher priority that these networks remain in operation 100% of the time [13]. This has spurred growth in the field of network management. A host of new companies have been born, producing an array of software and hardware intended to assist in the management of a network in one fashion or another. For example, there are software packages available commercially which simply create trouble tickets based on alarms from network management suites.

Along with the increased need for network management in today's workplace, the difficulties in performing network management have also grown. This can be attributed in part to the heterogeneity prevalent in today's computer networks, in areas of physical transmission facilities, network devices, multi-media information transmission, network protocols and objectives [4], and partly because industry emphasis has been placed on development as opposed to research.

The development of a network management system typically involves addressing

the following four issues. 1) Developing an understanding of the network is necessary in order to determine which set of statistics will prove most valuable for the purposes of network management. 2) A method to collect these statistics must be identified. 3) A scheme for the storage of the collected data must exist. This scheme is subject to the finite memory space available, and involves defining a data query method or language. 4) How to process the data, in other words, what can be learned from the data. This is related to step 1. Keeping in mind what information is desired from the network management system can guide the selection process as to which statistics to collect.

There is a lack of clear understanding of which network statistics are "most relevant" to proper management. The various standards bodies, such as the Internet Engineering Task Force and ATM Forum, have specified Management Information Bases (MIBs) which identify all possible network information which may be of value to network management in one manner or another [1,24]. The size of these MIBs can grow quite large especially as modern day networks span the globe. The heterogeneous nature of today's networks is certainly a contributing factor to the size of databases, as different networks have differing management requirements which must all be covered in a "standard" database. For example, in ATM networks, where there are quality of service (QoS) guarantees, monitoring the number of incorrectly delivered packets may be worthwhile, whereas in TCP/IP networks without such QoS parameters, such a statistic may prove less useful. Often, different requirements come from differing points of view. Within an Internet service provider (ISP), the accounting staff may desire the capability to monitor the network for billing purposes while network operations staff

may be more interested in the ability to detect and prevent security leaks.

Once statistics have been selected, there are a variety of means of performing data collection. Network data can be gathered by any combination of traps, polls, SNMP gets or other methods. The search for optimum means of performing data collection on computer networks is an on-going area of research. This work, however, does not deal with this issue and assumes the data has been collected by any means.

Similarly, this work does not discuss the nature of the network management system acting upon the data. It deals with the development of means for storing the data in a manner in which the information content of the data is readily accessible to the network management system.

It is well documented in the literature that network management data can grow quite large over a period of time. It has the potential to grow so large that the sheer volume of data available to network managers can be overwhelming, severely compromising their ability to administer the network [49,50]. For example, if a single network statistic, such as link utilization, is collected by sampling every 30 seconds, then 86,400 samples will be collected in one month (30 days x 24 hours x 60 minutes x 2 samples per minute) and 1,051,200 samples will be collected in one year (365 days x 24 hours x 60 minutes x 2 samples per minute). As the volume of statistics grows, the burden on the system's memory increases as well as the delays to network managers caused by the query process. In order to avoid these scenarios, it is necessary to perform data reduction before storing the data. A process by which the information content of the data can be retained without storing the actual data is proposed. This

information content will occupy significantly less memory space allowing faster query response time (due to smaller database size), and the database to contain more historical information (due to the nature of the storage scheme).

The storage scheme is designed for link utilization, and similar statistics which vary by time and in which the trend is more important than the actual numerical value. CPU load is another such statistic. A variation of the proposed scheme can also be employed for Device Reachability. This is a statistic in which the value is boolean. Either the device is reachable (1) or unreachable (0). The storing of data for long term use raises issues in databases, often the storage vehicles of choice for storing data of most any kind.

1.1 Data Compression vs. Reduction

The proposed scheme reduces the quantity of the data to be stored; it is not, however, a 'data compression' scheme. Various schemes for data compression have been presented in the literature and are in use today, a common compression scheme is the UNIX compress utility. The goal of such compression schemes is to allow a certain amount of data to be stored in a smaller amount of disk space (typically measured in bytes).

These schemes accomplish various levels of data compression and are essentially two-step processes. Data is first compressed from its original form through a certain algorithm. When the data is to be retrieved, it must be un-compressed. Compression

schemes are based on the principle that this un-compressed version is in some sense functionally equivalent to the original data. The definition of functional equivalence is determined by the requirements of the application and based on these requirements, different schemes have been presented. Such schemes do not intend for the data to be used while in the compressed form, indeed, compressed data is generally not human or application readable.

This is a distinction between these schemes and the proposed algorithm. In the proposed algorithm, classic "data compression" does not take place, in that there is no attempt made to extract the original data after the algorithm has been applied. Instead, the algorithm processes the data, causing a reduction and this reduced form is used in further analysis. A second distinction is that compression schemes do not attempt to preserve trends in the data, but merely store the data in a form requiring less disk space. Finally, the proposed scheme is a one-step process as aggregation needs to take place only once, as opposed to the compression scheme's requirement to compress and decompress the data.

1.1.1 Data Reduction

Data reduction techniques are utilized in applications where the quantity of data generated can overwhelm efforts to analyze the data. The general approach to data reduction has been to compare the data with standard, or previously identified, distributions or curves, such as a cubic spline or linear regressions in order to highlight

subsets of data containing similarities to the previously known information [7,27,28,42]. These subsets are most critical to the experiment and can be stored and used for analysis in place of the raw data.

For example, commercial data reduction packages exist to reduce data by computing a non-linear curve of known concentrations (of various chemical properties present in a particular solution). This curve is back fit onto the raw data (taken from assays where the concentrations are not known) to determine the concentrations. This is done by a logarithmic, non-linear curve fitting. Then the generated curve can be stored, which can be significantly smaller than the raw data [7].

This a priori knowledge is based on the characteristics of the application from which the data originates. Therefore, data reduction is highly application dependant. In certain cases, data reduction techniques are used to remove potentially flawed data, data that does not offer information about the underlying application, or can be attributed to errors in the experiment, imperfections in the test subject, etc. One such instance of this approach out of the Defense Research Establishment in Victoria, BC, a data reduction tool is used to display time-series data from satellite images and replace data points with the average of the power spectral density estimate over time intervals. This process removes erroneous values from the raw data while reducing the size of the data to be stored.

While there are distinctions between the proposed algorithm and standard data reduction algorithms, this scheme falls in the category of a data reduction approach. The differences will be further discussed in Section 4.2.1.

1.1.2 Dependant Variable Indexing

In the process of performing the data reduction, the proposed algorithm can serve as a precursor to a dependent-variable indexing scheme [25,26]. Such a scheme would, in this case, index the data on the link utilization or CPU load (dependent variables) values as opposed to the time values (independent variable). With the current indexing approach, the time sequence is scanned to process a queries. For queries, such as computing the time when the value reaches a certain threshold, the entire time sequence must be scanned in order to respond. In the literature [25,26], a method has been proposed to perform inverse queries - queries based on indexes of the dependant variable. These schemes can reduce the processing time of queries such as the one discussed above [25,26].

The proposed algorithm, as will be seen in further chapters, naturally subdivides the data into windows based on ranges along the dependant variable. These ranges have an associated window along the time access - with a start time corresponding to the initial point and an end time corresponding to the last data point in the range. These ranges can be used either as an indexing scheme itself, or as the basis for an indexing scheme. The proposed algorithm has a multiresolution approach, therefore, the indexing scheme can also have multi-layered granularity of its index.

1.2 Databases in Network Management

Databases play a crucial role in network management. Databases in the management arena are called Management Information Bases (MIBs). MIBs store all data collected for network management. They also define the data storage model.

Traditional databases are designed to store the views of the data which comprise the current snapshots of the system. In other words, databases store data for the network at a present point in time. While this view of the network is certainly important, past history is increasingly seen as crucial [1]. Past history can play a role in determining trends in usage and patterns in network behavior, such patterns utilization rates.

In order for databases to allow their users to access such information, all the data values recorded through history of the network must be recorded. Clearly this presents a problem as this quantity of data will certainly outgrow both any practical storage space and the capability of competent network managers to sift through, as discussed above.

Therefore, there is a need to develop a storage scheme which can store the network management data in a manner that allows the data's information, i.e., the value that can be drawn from that data, while not being required to store the full amount of the data. This is the question of how to discount data over time.

In order to accomplish this, we need to look carefully at the data itself to understand its defining characteristics. Data collected from computer networks for the purpose of network management are valid at the time instant they were collected. (This

may include a small time interval around the measurement time as precise timing is generally not possible and some flexibility in the 'stated' time of measurement is accepted).

Given that network management data can define the state of the network (for at least the network statistics collected) at the time of collection; network management data can define the state of the network over a time period if all data measurements over that time period are known.

Therefore, it is clear that relational databases which could store the complete collection of network management data over the time period of interest - the entire operation time of the network - could function as the repository, the MIB, for the purposes of network management.

However, as stated above, storing this quantity of data is simply not practical. The relational database model must be modified in some way so as to allow the information of the data to be available while the data itself is no longer available. A possible modification is evident upon closer inspection of the data.

Inherently, network data is valid over a period of time, say between two sample times. We understand the data to be an accurate representation of the network statistic's value at the time instant it was collected and the time interval associated with that time instant. We accept two values taken in two consecutive samples to define the range in which that particular network statistic takes its value over that time period defined by those two consecutive samples.

It is this temporal characteristic of the network that we intend to exploit. Given that the value recorded in one sampling, conducted at one time instant, is the value accepted over the range surrounding the time of measurement and that the values recorded over consecutive measurements define the value range over that time interval, then if those values are close enough to be considered equivalent (i.e., that the difference between them is not significant to the management of the network such as the case of a link having 42% utilization or 46% utilization, either rate being considered acceptable) then the data points which are stored defining those measurements can be replaced with one data point defining the value and the range over which that value holds.

For example, suppose at sample time t_1 link utilization is 42% and at time t_m the value is 46% and at time t_n it is 44%. In place of recording all three readings in the database, we can simply record one entry as the average of the above three and designate it time t_1. It will have to be made clear that the value of t_1 holds for three samples.

This can be done in a straightforward manner in temporal databases. Temporal databases, sometimes called historical databases, are essentially relational databases in which the time interval, over which the stored data is valid, is specified.

In many applications, previous states are important, as are the changes in state over time, the trend of the data. For example, in order to upgrade a network, it may very well be helpful to understand how usage of the network changed over time. To know which links are no longer used and which are getting more extensive use; to know which services and applications are gaining popularity among network users and

which are losing favor.

The remainder of this thesis is organized as follows. Chapter Two introduces the area of temporal databases. Chapter Three presents the storage scheme upon which the work is based and Chapter Four presents and analyzes the results of the simulation and comparison between a standard relational database, Oracle 7, and a temporal databases, TIGER. The TIGER temporal database is a product of the computer science department of Aalborg University in Denmark[1]. Chapter Five discusses few envisioned applications of the proposed scheme and Chapter Six is the conclusion, which includes a section on possible future direction for this research. Appendix A and B contain the source code for the simulations used.

[1] The TIGER temporal database is a product of the computer science department of Aalborg University in Denmark. It is accessible from the web at http://www.cs.auc.dk/~tigeradm.

Chapter 2

Comparison of Relational and Temporal Databases

The need to view the change in data over time led to the concept of temporal databases. Temporal, or historical, databases are essentially databases with an additional dimension, the time dimension. There has been a great deal of research into temporal databases reported in the literature for applications in which the systems concerned have data which varies over time, and therefore fits this time-dependant model. For example, scientific and statistical data such as seismological data, CAD/CAM applications, and also financial applications such as monitoring the value of a stock [8,10].

A number of temporal data models have been developed [9,17,40,41,51-54] which support time varying information. The research community has taken the approach of interpreting time as being discrete, mainly for the reason of simplicity and relative ease of implementation. Time is represented as: T = {0, 1, 2, ..., now}, with *0* = start time and *now* = a constant to represent current time. In some literature, the term *fe* is used to represent the largest possible time. The time units are application specific. Any subset of T is considered the Temporal Set or Temporal Element [51-54].

2.1 Discussion of Temporal Data Models

A set of general requirements have been established for the data models. While

not all proposed models fulfill these requirements, omissions being made for the sake of ease of implementation, reviewing them here is worthwhile.

2.1.1 Definition of Terms

A few terms should be identified before a full discussion of a temporal data model can take place. A temporal atom is an ordered pair <t,v> where t is the temporal set or an interval [t , T) and v is an atomic value from a specified domain. There can be four types of attributes in temporal relations:

1. Simple valued (Atomic)	e.g.,	12	Tom
2. Set valued	e.g.,	{12,16}	{Tom, Mary, John}
3. Triplet valued	e.g.,	<{12,16}, 47>	
4. Set Triplet valued	e.g.,	{<{12,16}, 47>, <{18,20},55>}	

Atomic attributes contain just one data point. Relational databases deal exclusively with atomic attributes. Set valued attributes can have multiple data points. Triplet valued attributes are formed through a concatenation of set valued or atomic attributes. Likewise, set triplet valued attributes are multiple triplet valued attributes.

2.1.2 Temporal Data Model

Temporal data models should provide the ability to query the database for any point in time (within the lifetime of the stored data). Also, the model should allow

querying the data for the current state of the database as if it were a traditional relational database. This requirement essentially states that the addition of time into the database should be transparent to the user. Certainly this is a desired characteristic, however, it has proven difficult to achieve in practice [6,51-54].

The data model should be capable of modeling and querying the database at two different time points. This is straightforward in relational databases if time is recorded as a single valued attribute, i.e., restricted to time points. In temporal databases, the goal is to have the database be able to distinguish between states for different, yet overlapping intervals in time. For example, we want the database to be able to protect its consistency under the condition in Table 2.1.

Link	Start Time	End Time	Link Utilization
1	T_1	T_4	72%
1	T_2	T_6	75%

Table 2.1: An example of possible inconsistency in temporal databases.

The above tuples illustrate a danger with temporal databases. As the value for link utilization is an average, both tuples may be accurate, i.e., that the average link utilization from T_1 to T_4 may well be 72%, while the average from T_2 to T_6 be 75%. However, this clearly offers conflicting values for T_2 and T_3. In this case, the second of the tuples must be rejected and the operator flagged as to the inconsistency.

The data model should allow for different periods of validity in attributes within a tuple, i.e., nonhomogeneous (heterogeneous) tuples. This condition is not always

agreed upon in the literature, and therefore not always proposed. Several proposals restrict the temporal database to homogeneous tuples. Such a restriction does retard performance, however it simultaneously increases ease of implementation [8,51-54]. Both heterogeneous and homogeneous tuples are illustrated in Table 2.2.

Link	Utilization (%)
1	$<[T_1, T_2, T_5, T_6, T_9, T_{10}), 57>$
1	$<[T_3, T_4, T_7, T_8), 63>$

(A)

Link	Utilization (%)
1	$<[T_1, T_2), 57>$
1	$<[T_5, T_6), 57>$
1	$<[T_9, T_{10}), 57>$
1	$<[T_3, T_4), 63>$
1	$<[T_7, T_8), 63>$

(B)

Table 2.2: Illustration of heterogeneous and homogeneous attributes. A) heterogeneous attributes B) homogeneous attributes.

The data model should allow multiple valued attributes at any point of time. This again is a point of disagreement in the temporal database community. Allowing multiple valued attributes increases the effectiveness of the database, however, increases the difficulty involved with implementation as well [51-54]. Several approaches have been made without allowing multiple valued tuples. Not only does this requirement complicate implementation, but it also complicates the formation of an effective query language, as will be shown later in this chapter.

2.2 Characteristics of Temporal Databases

2.2.1 Time Stamping

Time relations have been referred to as three dimensional cubes, with the third dimension being time [39]. These three dimensional cubes are transformed into two dimensional constructs by time stamping [39]. This time attribute being the major distinction between temporal and relational databases.

Three methods for time stamping are illustrated in Table 2.3:

	Time-Stamps:	
CPU Load	CPU Load	CPU Load
$<T_1, 45\%>$ $<T_2, 48\%>$ $<T_3, 43\%>$	$<[T_1, T_4), 45\%>$ $<[T_4, T_8), 45\%>$ $<[T_8, now), 48\%>$	$<[T_1\text{-}T_7, T_{12}, T_{14}), 45\%>$
Time Points	Intervals	Temporal Set
A)	B)	C)

Table 2.3: Illustration of time stamps in temporal database. A) is a time point, B) is a time interval and C) is a temporal set.

Time points are closest to the static data currently maintained in relational databases. They can be recorded simply as another attribute well within the relational model.

Intervals and Temporal Set offer greater database size reduction, as multiple

tuples with the same data value and differing only in the time element can be reduced to a single tuple as will be shown later. The difference between Interval time stamping and Temporal Set time stamping is merely that a temporal set may contain discontinuous time values. Essentially, a temporal set can be comprised of a concatenation of time intervals.

The location of the time stamp is also an issue. That is a time stamp can either be applied to the entire tuple or the individual attributes. The former is called tuple time stamping and the later is called attribute time stamping.

In tuple time stamping, a relation is augmented with two time values, a start time and an end time delineating the time interval. Such databases maintain First Normal Form [9,41,51-54]. Therefore, such a scheme can benefit from traditional database technology.

Snodgrass [41] introduced a data model employing tuple time stamping. This model maintains static attributes, along with or including the primary key, as a separate relation and time stamps each time-varying attribute individually in its own relation, as shown in Table 2.4 [39,51-54].

E#	ENAME
121	TOM
133	ANN
147	JOHN

E#	SALARY	VALID	
		FROM	TO
121	20K	10	15
121	20K	15	17
121	30K	17	18
133	35K	25	30
147	42K	18	now

E#	DEPARTMENT	VALID	
		From	To
121	Sales	10	12
121	Mktg.	14	18
133	Sales	25	30
147	Toys	18	now

Table 2.4: Snodgrass' temporal database model.

While maintaining First Normal Form is certainly advantageous, there are disadvantages to the above approach. The data is broken into several tuples and creates smaller fragmented relations causing data redundancy resulting from duplication of the static attributes and primary key. This scheme also complicates the management of database consistency. If a static value is to be updated, it must be updated not only in the static relation, but also in each relation in which it appears. For example, in order for the Ann's employee number to be changed, the change must be made simultaneously in all three relations, in place of one. Maintaining database consistency becomes significantly harder [51-54].

The database is in first normal form, however, SQL will require modification before it can be directly applied to the database, as it must be clear that a query requesting Salary for Employee Tom at time 17 should return 30K in place of 25K.

The alternative to tuple time stamping is attribute time stamping. In attribute

time stamping, the attributes themselves are given a time stamp. This method of time stamping introduces a new domain of attribute values.

Tansel introduced a data model utilizing attribute time stamping [53]. In this model, time stamped attributes consists of two components, the timestamp and the data values, for example <(10, 12), Sales> would be one attribute as opposed to three. [See Table 2.5 below]

E#	ENAME	DEPARTMENT	SALARY
121	Tom	<{[10, 12]}, SALES>	<{[10, 15]}, 20K>
		<{[14, 18]}, MKTG>	<{[15, 17]}, 25K>
			<{[17, 20]}, 30K>
133	Ann	<{[25, 30]}, SALES>	<{[25, 30]}, 35K>
147	John	<{[18, now]}, SALES>	<{[18, now]}, 42K>

Table 2.5: Attribute Time Stamping fromTansel's Data Model.

This scheme does not maintain first normal form, causing a break with traditional database technology. Non-first normal form relations are more difficult to implement. The case of homogeneous of tuples has been mentioned. A homogeneous tuple is one in which every attribute of a tuple is time stamped with the same time interval or temporal set. Gadia's [18] approach includes homogeneous tuples, see Table 2.6.

E#	ENAME	DEPARTMENT	SALARY
[10, 12) 121	[10, 12) Tom	[10, 12) Sales	[10, 12) 20K
[14, 18)	[14, 18)	[14, 18) Mktg.	[14, 15) [15, 17) 25K [17,18) 30K
[25,30) 133	[25,30) Ann	[25,30) Sales	[25,30) 35K
[18, now) 147	[18, now) John	[18, now)Toys	[18,now)42K

Table 2.6: Homogeneous tuples in temporal databases.

2.2.2 Requirements for the Query Language

The languages proposed in the literature fall into two categories, they are either extensions of SQL, the defacto standard language among relational databases, or new temporal query languages, such as Snodgrass', TQUEL[40] and Tansel's Time-by-Example[50,51]. Many of these new languages are also derived from, or inspired by, SQL.

An ideal temporal query language is one which can address all of the above listed data model requirements, can handle either method of time stamping, with any kind of stamp. Certainly, choices must be made as to the nature of the language to employ. A list of requirements for query languages has also been proposed in the literature.

The belief that the query language should have the capability to return the same type of object that it operates on is fairly universally accepted [8,51-54]. This implies that if the attribute(s) requested is (are) time stamped, then that complete time stamp should be returned. However, if the value of an attribute(s) time stamped with either an Interval or Temporal Set is requested at a particular instant in time, then that value should be returned with just that time point and not the complete time stamp.

The language should be able to perform the same comparison tests on time stamps (either of the three kinds of time stamps) that SQL and other relational database languages can perform on standard single-valued attributes.

Similarly, the language should have the ability to regroup data according to different criteria. Essentially, temporal query languages should have the ability to perform the same actions on temporal databases as existing query languages can perform on relational databases.

Temporal databases draw a distinction from relational databases in their query languages as well. A relational query language need not treat the time element any differently from other attributes stored in the database. Queries made against a relational databases return values that are explicitly entered into the database. If time is recorded as an attribute, and a query is made against a particular time or a set of time values, the database will be capable of returning the values for time points that are specified in the database, however, if the time points are not specified, it is as if the data does not exist in the database and no values are returned. Temporal query languages can read time as an interval therefore allowing queries to be made for time points that are not explicitly specified in the database. This functionality allows the stored data to represent movement *through* time, as opposed to merely a snapshot of data *in* time. The benefits of this will be seen in the results described in Chapter 4.

Chapter 3

Data Storage Schemes

The scheme presented here are for storing link utilization, CPU load and device reachability statistics. Link utilization is defined as the utilized bandwidth of individual links throughout the network and is stored in terms of percentages. For ATM networks, the utilization of individual Virtual Paths (VPs) can also be maintained. In this case, the percentage of utilized bandwidth over guaranteed bandwidth in percentage is recorded. CPU load is the amount of processor power currently engaged, also recorded as a percentage. Device reachability is the ability to reach of network devices, such as printers, servers, etc., regardless of the path (over logical or physical links) chosen. Device reachability is a boolean value as the device is either reachable (value = 1) or unreachable (value = 0). The data storage schemes are introduced and discussed below. The data can be gathered by use of traps, probes or SNMP commands, however, discussion of data collection techniques is beyond the scope of this work. The schemes for link utilization and CPU load are identical and that for device reachability has only a slight modification.

3.1 Selection of Network Statistics, Development of Storage Scheme

Before the discussion of the scheme itself, it is prudent to mention the rationale that went into its development. The statistics for which this scheme is developed, Link Utilization and CPU Load, are both time varying statistics, can be measured in percentages and their overall trend is more important that their individual values in a historical sense. As mentioned above, the individual numerical values of these statistics take on lesser importance over time than the pattern in movement of the statistics. (This point will be discussed in greater detail in section 3.3) Therefore, a storage scheme that places greater emphasis on the trend of the data than the data itself is a prime candidate for use in this application.

With this point in mind, the scheme was designed by noticing a parallel between the network management statistics under examination and the quote price of publicly traded stocks. For up to the minute, or real-time stock quotes, as reported on the Internet[2], the values are given in the same granularity with which they are measured but are shown for the present day only. For historical values, say over the past week, stock quotes are presented on ½ hour granularity. For longer term, say the past 3 months or year, only the closing values are shown. Essentially, the granularity with which stock quotes are reported decreases as the view window becomes larger. This multiple resolution system accomplishes two things, first, the highest level of granularity does not need to be recorded and retained in the database permanently and second, investors are

[2] The value of publicly traded stocks are reported on several World Wide Web sites, www.foxmarketwire.com andwww.quote.com are two examples.

able to see the movement of the stock price and obtain the information they seek.

This same multi-resolution approach can be employed in the storage of network management statistics.

3.2 Description of the Storage Schemes

3.2.1 Storage Scheme for Link Utilization and CPU Load

The proposed scheme stores data in three resolutions. The first resolution stores recorded data for the most recent period of time. This time period is sufficient only to provide the current state of the network. The first resolution is illustrated in Table 3.1:

First Resolution Table

Link	Utilization
01	$<t_1, 25>$
01	$<t_2, 26>$
01	$<t_3, 27>$
01	$<t_4, 31>$
01	$<t_5, 29>$

Table 3.1: First resolution table proposed for link utilization.

In this case, the past five (5) data points are stored in the first resolution. Data is gathered every twenty (20) seconds, therefore the first resolution comprises one (1) minute and forty (40) seconds. There is one such relation for each link in the network. In an ATM network, utilization data may be collected for the physical link as well as the logical VP links.

As this resolution stores only current data, old data is pushed out as new data is collected. (Older, historical data is stored in the second and third resolutions.) For example, after two further data collections, the relation in Table 3.1 may look like Table 3.2:

First Resolution Table

Link	Utilization
01	$<t_3, 27>$
01	$<t_4, 31>$
01	$<t_5, 29>$
01	$<t_6, 35>$
01	$<t_7, 38>$

Table 3.2: First resolution table for link utilization.

Essentially, the first resolution holds the recent data and is updated every time a new measurement is taken. In the format described, time stamps are used to attach a time point to the data values, however, time can be considered another attribute and the data can possibly be stored in a standard relational database. This applies only to the first resolution data.

The second resolution stores aggregates of the data - as opposed to the actual data - in tuples in the temporal database. The data values of the first resolution are aggregated into tuples according to the following rules.

Rules for the creation of second stage resolution tuple:

The first data value creates a new tuple. After the first data value a new tuple is created if

I. $ABS(v_o - v_n) > \delta$
II. If $v_n >$ *High threshold (85%)*
III. If $v_n <$ *Low threshold (15%)*

where v_o is the value of the first data collection point of the current tuple, v_n is the value in the current data collection point and $\delta = 5$. A value for δ has been selected by considering that the exact value of the link utilization is less important than the trend of the link utilization. Experimental results with values of 5, 10 and 20 show that significant reduction in database size can be achieved while maintaining the trend of the data with a $\delta = 5$. This value can be set more or less tightly depending on the application. An appliction in which more precise records are necessary can reduce the value of δ, applications where less precision is necessary, or if greater data reduction is necessary and increase δ. The amount of data reduction achieved by this aggregation is dependant upon the value of δ and the fluctuation of the data, as will be discussed in the following chapter.

It is worthwhile to point out that the comparison above is performed between the first point of the tuple (v_o) and the current point (v_n) and does not involve the value currently stored in the data. This is to ensure that both upward and downward runs of the system will be evident from the second resolution. Using the first point as the reference point for a tuple ensures that any significant run in either direction will move more than d from v_o and create a new tuple allowing the run to be stored.

The second and third rules of the aggregation scheme are designed to retain all information regarding the system operating in either a high or low utilization range. While the purpose behind the scheme is to identify trends in the data, storing all data in these periods is of value. With high utilization, it is possible that further spikes in usage

overload the system, bringing it down and defeating the purpose of network management. In low utilization, resources are not being used; when resources are not being used they cannot generate revenue, defeating the purpose of the network. This lower threshold should be set lower than the typical off-hours usage of the network, in the case of a corporate network, to avoid a situation where the lower threshold is crossed each night. The values in the first resolution above will aggregate according to the proposed scheme and fill the second resolution as in Table 3.3:

Second Resolution Table

Link	Utilization
01	$<[t_1, t_4), 26>$
01	$<[t_4, t_7), 32>$
01	$<[t_7, t_8), 38>$

Table 3.3: Second resolution table for link utilization.

Again, there would be one such relation for each measured link. The time interval above employs a closed left side and an open right side, indicating that the left time point is included and the right time point is not. In this resolution, a time interval is employed and offers a more stark contrast with the traditional relational model than the first resolution. In the next chapter, the benefit of allowing a time interval to be associated with the values of the database is discussed in detail.

The third resolution offers further aggregation, the rules for the creation of tuples in the third resolution are similar to those in the second, the difference being a larger value (ε) used in place of δ.

Rules for the creation of third stage resolution tuple:

The first data value creates a new tuple. After the first data value a new tuple is created if

I. $Abs(v_o - v_n) > \varepsilon$
II. *If* $v_n >$ *High threshold (85%)*
III. *If* $v_n <$ *low threshold (15%)*

where v_o is the first value in the current time interval, v_n is the value in the current time interval and $\varepsilon = 20$. The determination of $\varepsilon = 20$ follows the same argument for the determination of δ given above. In this resolution, the condition for the high and low thresholds are maintained, so that there is a permanent record of all periods of over or under utilization.

The values above will aggregate according to the purposed scheme as illustrated in Table 3.4:

Third Resolution Table

Link	Utilization
01	$<[t_1, t_8), 32>$

Table 3.4: Third resolution table for link utilization.

There would be one such relation for each link.

3.2.2 Storage Scheme for Device Reachability

For the case of Link/Device Reachability, the value is boolean, either a device is reachable (device is up, defined as value = 1) or the device is unreachable (device is

down, defined as value = 0). Due to the Boolean nature of the statistic, the scheme can be modified to require only two resolutions. Data can be gathered, as before, by either the use of traps, polling or SNMP commands. The data in the first resolution stores only current values and purges old values as new value enters, once the capacity is filled. A sample relation is shown below (see Table 3.5):

First Resolution Table

Link/Device	Reachability
01	$<t_1, 1>$
01	$<t_2, 0>$
01	$<t_3, 1>$
01	$<t_4, 1>$
01	$<t_5, 1>$

Table 3.5: First resolution table for link/device reachability.

Again, one such relation is required for each link and device in the network.

As the value for Reachability is Boolean, there is no concept of value crossing a high or low thresholds (it will be either zero or one). Therefore, the rule for the creation of a tuple in the second resolution need only have one condition.

The second resolution for Device Reachability has slightly different tuple creation rules. There is only one tuple for each device. That tuple contains the temporal set in which the device was unreachable (value = 0). At other times, the device can be assumed, with probability one (1), to be reachable. We choose to store the times when the device was unreachable as we believe this will be a significantly lesser time than when it is reachable, and therefore require much less storage space than the times when the link or device is reachable.

Rules for the creation of second stage resolution tuple:

Store in single tuples the time intervals when device was unreachable.

The relation may look like the following (see Table 3.6):

Second Resolution Table

Device	Time
01	$<[t_2, t_3)>$

Table 3.6: Second resolution table for link/device reachability.

There would be one such relation (tuple) for each device. No further resolution is required. Nor is it necessary to utilize a single byte of memory to indicate that the link was unreachable. As it has been determined that data will only be stored for the downtime, it becomes redundant to store the Boolean attribute to that effect.

3.3 Cost of Data Storage Scheme

The proposed scheme will allow for data reduction, therefore some (possibly a significant) quantity of data will be lost and replaced with an aggregate of this data. In order to analyze the cost associated with this scheme, it is necessary to re-examine the aggregation process.

Each tuple is established by the aggregation of consecutive data readings which lie within the specified δ of the first reading assigned to the tuple. The actual data and the aggregated value, then, lies in a window of length $2\ \delta$. For applications requiring precise data values, the value of δ can be made very low so that the aggregation is

performed over a smaller range and a higher granularity is achieved. For applications where such precision is not necessary, δ can be set relatively high, offering less granularity but significantly more data reduction.

The value of the lost data is dependant upon the application to which the data belongs. In certain applications, such as seismology, the particular values of measurements, as well as the trends, are of high importance as they can be used to estimate location and severity of tremors and earthquakes. In financial applications, such as the reporting of stock prices on the Internet, they again are of great importance as large sums of money are traded on the actual values.

In network management, less of an emphasis is placed on actual values [36] and therefore the cost associated with lost data is considerable less than in the above two applications.

Recall, in this scheme, that any time the data rises above or drops below the high or low thresholds, a new tuple is created. This is to ensure that the alarms due to both high and low utilization are safeguarded. There is a concern that certain alarms may be set within these two thresholds and further that they could be lost in the aggregation process. The value of these alarms are implementation specific and must be evaluated on a case by case basis. However, if they are deemed worth recording, then one alternative is to add a flag attribute to the temporal database to record these alarms. This attribute can store a number consisting of as many Boolean digits as possible alarms (or at least prominent alarms). Each time an alarm is triggered within an interval, the corresponding alarm bit in the flag attribute is set. This approach will reduce the cost of

the lost data while retaining information about the occurrence of alarms.

Chapter 4

Results

In order to illustrate the data reduction and the trend identification achieved by the storage scheme discussed in the previous chapter, the following experimentation and data analysis procedure has been identified. Data is gathered through two simulations discussed in Section 4.1. This data is then processed in accordance with the proposed scheme in Matlab. The output from Matlab is the second and third resolution of the data. These resolutions, along with the original data appear in graphical form in Section 4.2 where the graphs are discussed in detail. The data is then loaded into an Oracle 7 and a TIGER temporal database. The functionality of these databases is compared and contrasted by performing queries on both databases. The database loading and query processes are both discussed in Section 4.3 where the added functionality afforded by TIGER's handling of the time element is illustrated.

4.1 Network Simulations

In order to generate data to test the proposed storage scheme, two network simulations were designed and run. The simulations were performed with the NS simulation package. This package is supported by the Van Jacobson group. The NS simulation files are written in the TCL language. All simulations were conducted on the Sun Ultra Sparc 2 machines in the Systems Engineering and Integration Laboratory of

the Institute for Systems Research. The architecture for both simulation appears in Figure 4.1. The first simulation is of a varying number of TCP and ATM (CBR and UBR traffic classes) sources generating packets of length between 100 bytes and 1000 bytes are simulated. Certain streams output packets continuously at a moderate rate while others are active only for certain periods of time but output traffic at a higher rate. The number of total sources for a particular run of a simulation is specified, however, the number of active sources at a given time instance is determined by a uniform random distribution between one and the maximum specified number of sources for that run of the simulation. The length of time a TCP source is active is determined by a Pareto random distribution and are drop-tail sources. The length of time an ATM source is active is determined by an exponential random distribution.

The network configuration for the simulation is as follows. All sources are connected to a router on one end of a link. The connection between the sources and this router are error free, and are assumed to have unlimited buffer capacity. On the other end of this link is the destination node. The buffer at the destination is also assumed to be infinite.

The second simulation is programmed to demonstrate the aggregation possible for ATM traffic by generating only CBR sources. This simulation has a similar network configuration and similar parameters, with a key exception that only ATM CBR sources are simulated.

In each of the above simulations, the link utilization of the link between the router and destination node is measured and recorded along with the sample time.

Samples are taken every twenty seconds over the course of the simulation. Link utilization is calculated by multiplying the number of packets generated per traffic stream for a specified duration of time, by the number of bytes per packet and dividing by the bandwidth of the link between the router and destination nodes.

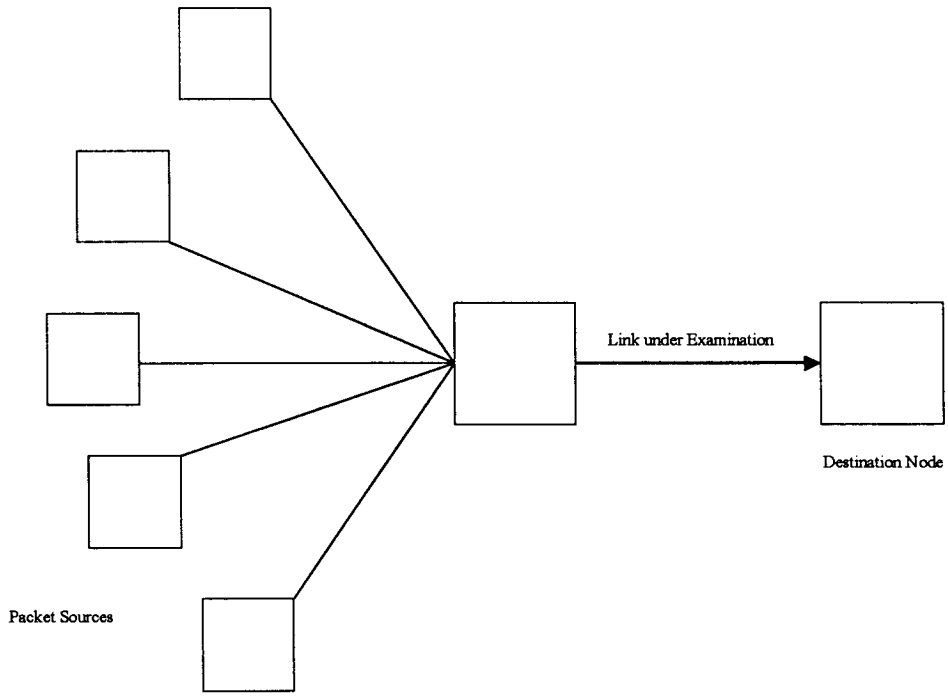

Figure 4.1: The architecture utilized for both simulations.

4.2 Data Processing

The data generated from the simulations is processed in Matlab. The result of this processing is the second and third resolution of the data storage scheme. As the first resolution is merely the real time data for the current period of time, the data need

not be processed to generate this resolution.

The second and third resolutions are plotted against the original data in order to illustrate both the data reduction and trend identification achieved by the storage scheme. Again, the first resolution is not plotted as it is simply a storage of the most recent data.

Figure 4.2 is a graphical illustration of the data generated by the TCP and ATM simulation, along with the second and third resolutions of this data. In this simulation, 36 hours, seven (7) minutes and twenty (20) seconds of network time are simulated. Data is gathered every twenty (20) seconds (of simulated time) for a total of 6502 data points. The simulated data appears as the top plot. The middle plot represents the second resolution of the data and the lower plot represents the third resolution. From the graph, it can be seen that the contour of the processed data, at both resolutions, closely resembles that of the simulated data while there is a significant data reduction. The trends in the underlying data are preserved in both resolutions. Between the first and second resolution the data reduction is approximately a factor of two (2), and between the first and third, there is a data reduction of approximately a factor of six (6). This graph shows the second resolution of the data taken for the entire generated data. However, the proposed scheme recommends storing data in the second resolution only for a limited time. Therefore, figure 4.3 illustrated the second resolution (middle plot) for the last eight (8) hours of the simulation, or 1440 data measurements. The top and lower plot are again the generated data and third resolution respectively. From this graph the overall data reduction is more clear. The actual data represents 6502 data

points collected in over 36 hours. The second resolution (705 data points) and third resolution (2173) combined amount to 2878 data points. This is a reduction of approximately 55% in data volume. (As the first resolution stores data for only a brief period of time, and serves essentially like a cache, its contribution to the data store can be ignored as this amount will have negligible effect when considered over the long term.) This reduction applied to the link utilization statistic mentioned in the introduction can allow one year of history to be recorded in 462,528 data points, substantially smaller than the generated number of data points, 1,051,200. (It is clear that in this regard, adding the first resolution data in this case will have a negligible overall effect on the magnitude of data reduction.)

Figures 4.4 shows the second resolutions (lower plot) against the actual data for the time period from which it is derived. The fact that the trend in the data is preserved in the aggregated form of the data can be seen by counting the peaks in the data. A similar case for the third resolution data appears in figure 4.5.

The data reduction case for ATM CBR traffic is significantly greater, as was anticipated. Due to the fact that the utilization rates are relatively stable and the aggregation is far more significant. In this case, five (5) hours and seven (7) minutes of network time are simulated. Again, data is gathered in twenty (20) second intervals for a total of 921 data points. Figure 4.6 shows the generated data (top plot) along with the second and third resolutions (middle and lower plots, respectively) taken from the entire data stream. From 921 data points in the first resolution, there are only forty-six (46) data points in the second resolution while the overall trend remains. The third resolution

has 26 data points. The aggregation is even more significant when the second resolution is taken over only the last half of the data, or two (2) hours and thirty (30) minutes. This is illustrated in figure 4.7, where the top plot again represents the collected data and the lower plots represent the second and third resolution values. This period of time constitutes 450 data points and is represented in twenty-two (22) data point. Therefore, the overall reduction for the ATM CBR traffic case is from 921 data points to merely forty-eight (48) data points. The level of reduction in this case, while expected to be high due to the nature of the underlying data, exceeds expectations.

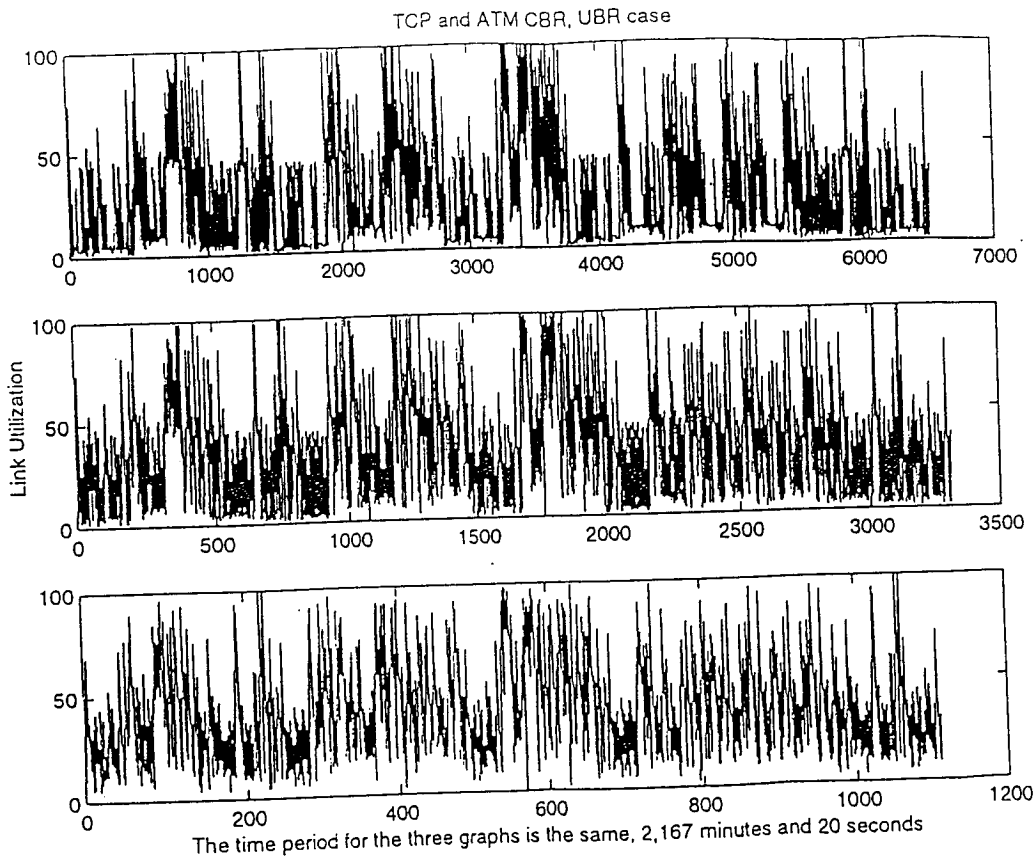

Figure 4.2: TCP and ATM simulation, data processed to second and third resolutions.

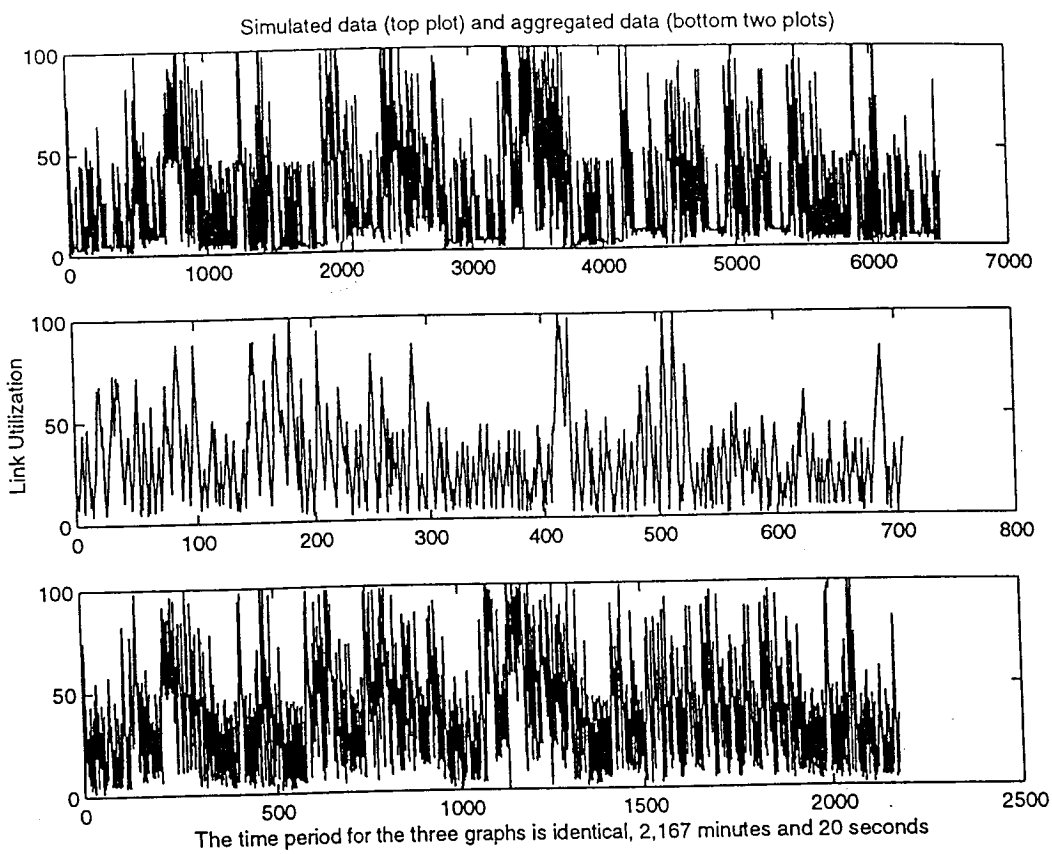

Figure 4.3: TCP and ATM simulation, latter half of data processed into second resolution.

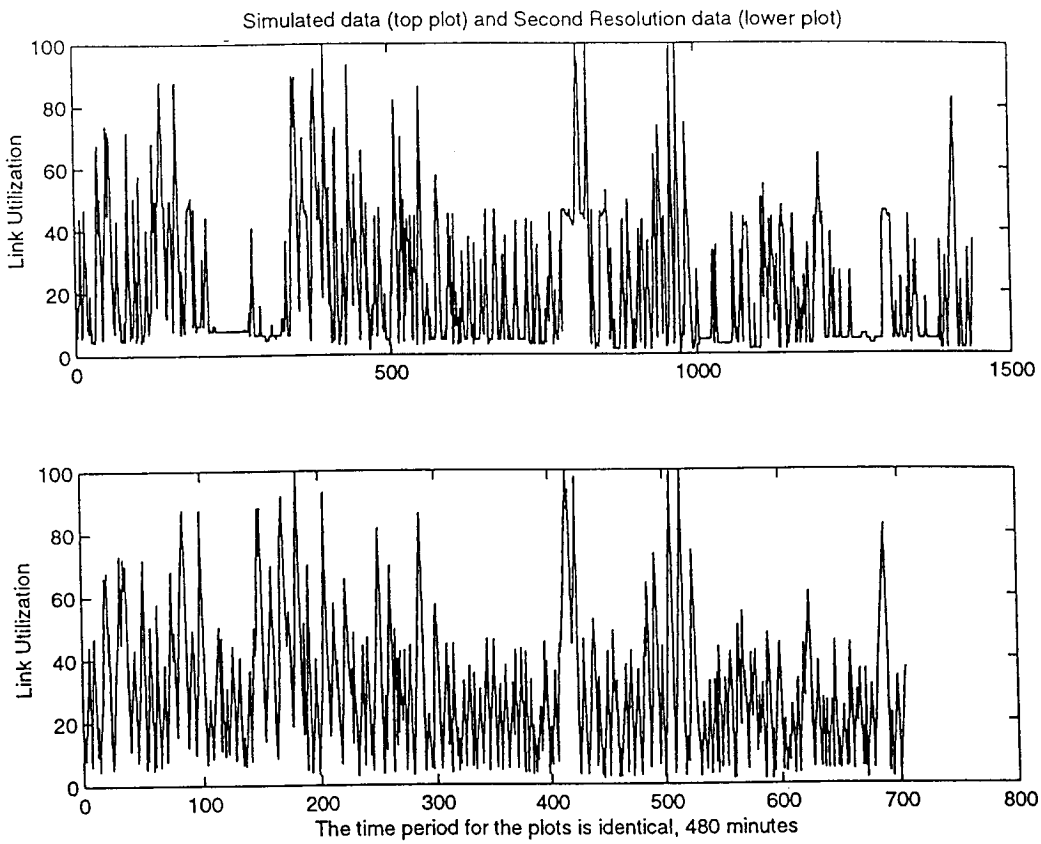

Figure 4.4: Second resolution and original data, TCP and ATM simulation.

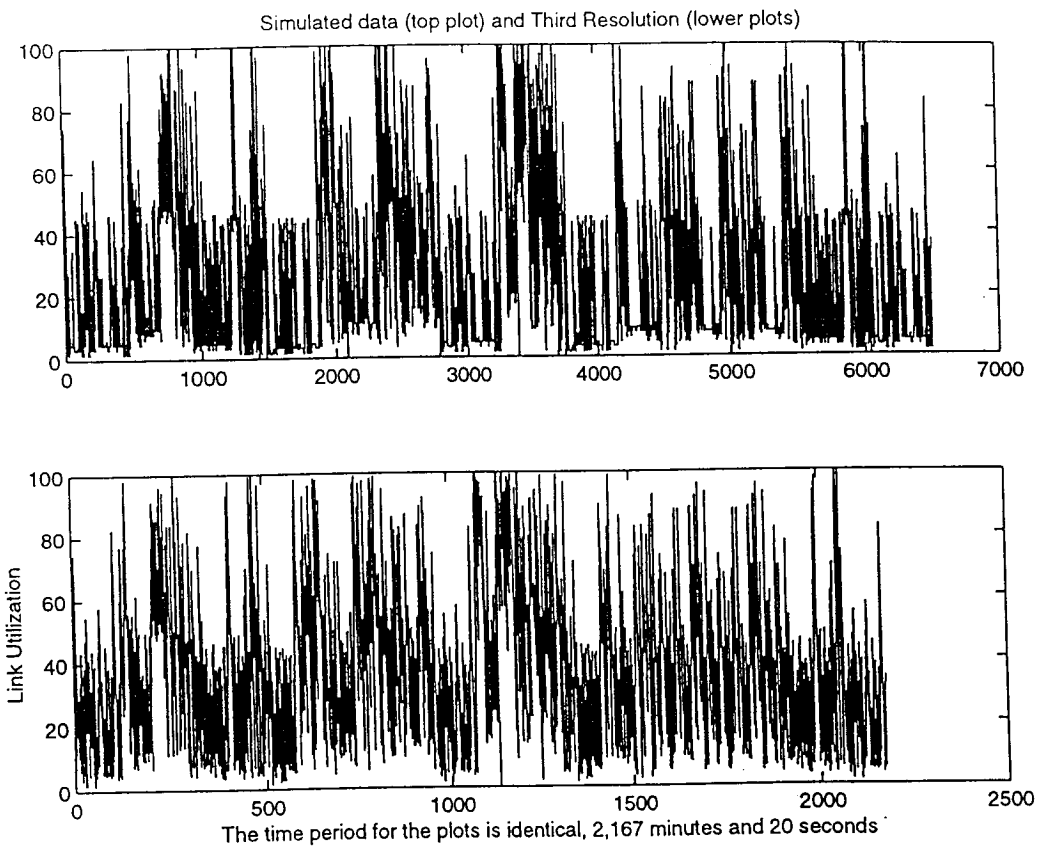

Figure 4.5: Third resolution and original data, TCP and ATM simulation.

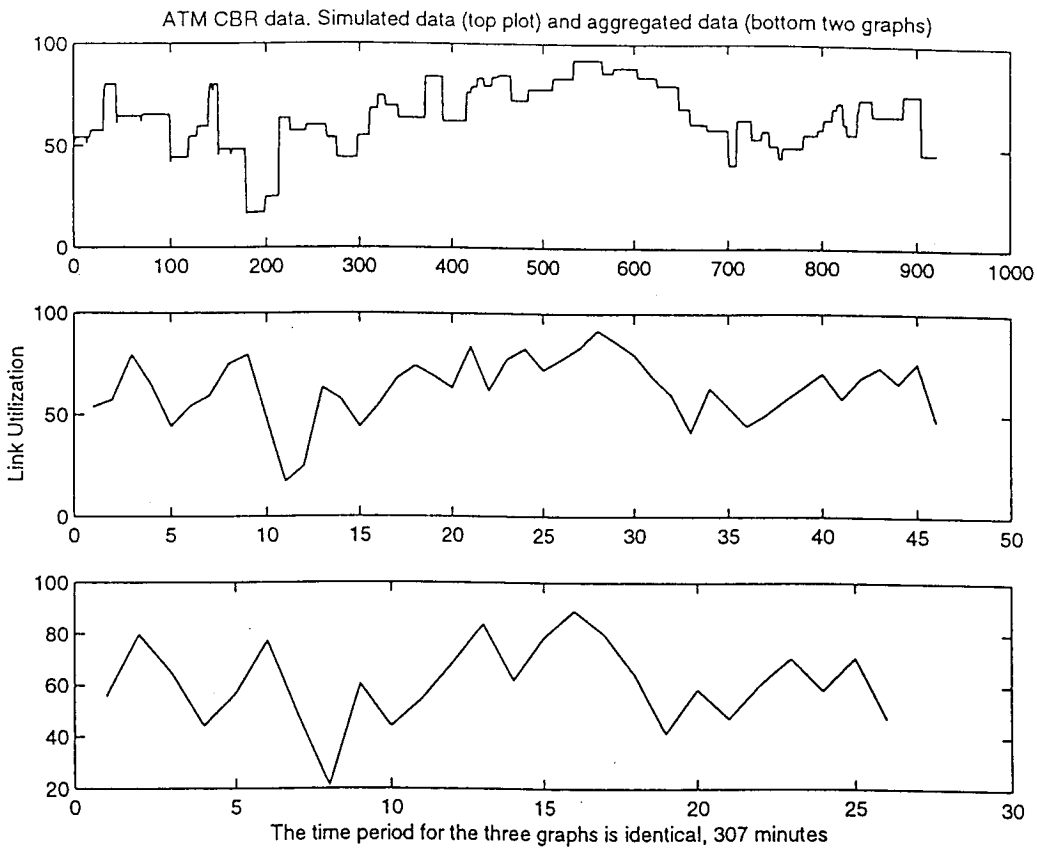

Figure 4.6: ATM CBR simulation, with data processed to the second and third resolution.

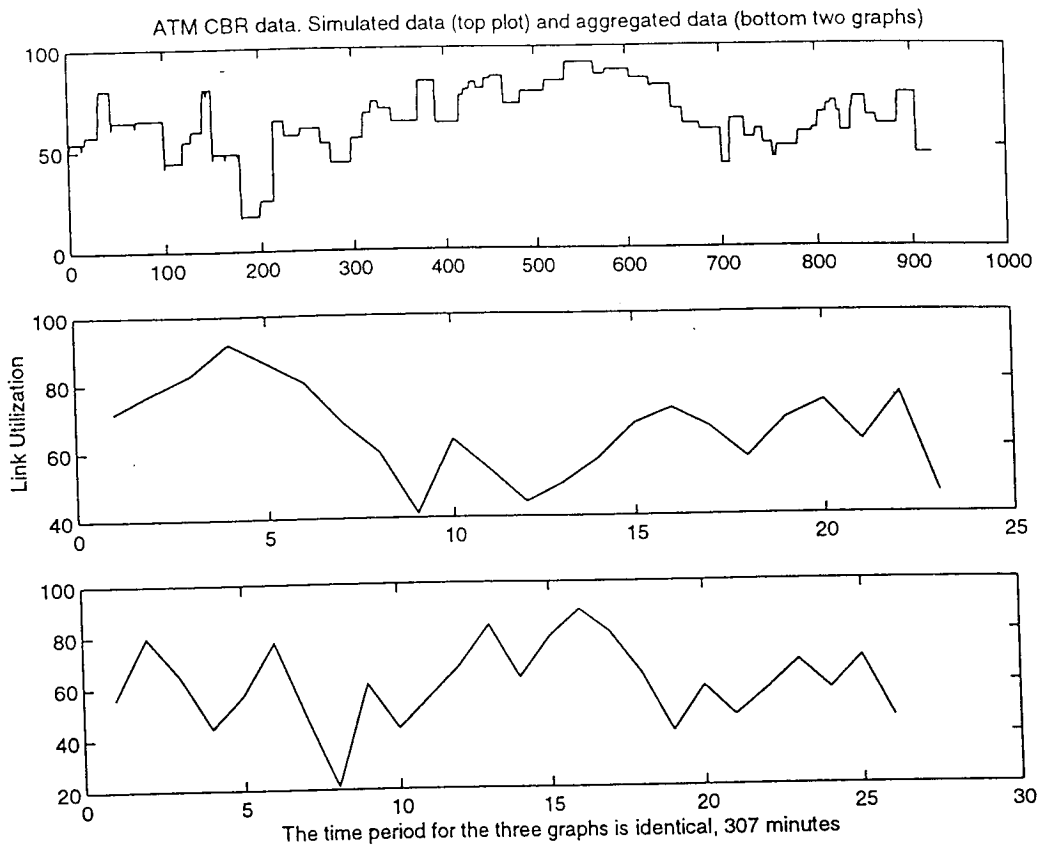

Figure 4.7: ATM CBR simulation, with later half of data processed to second resolution.

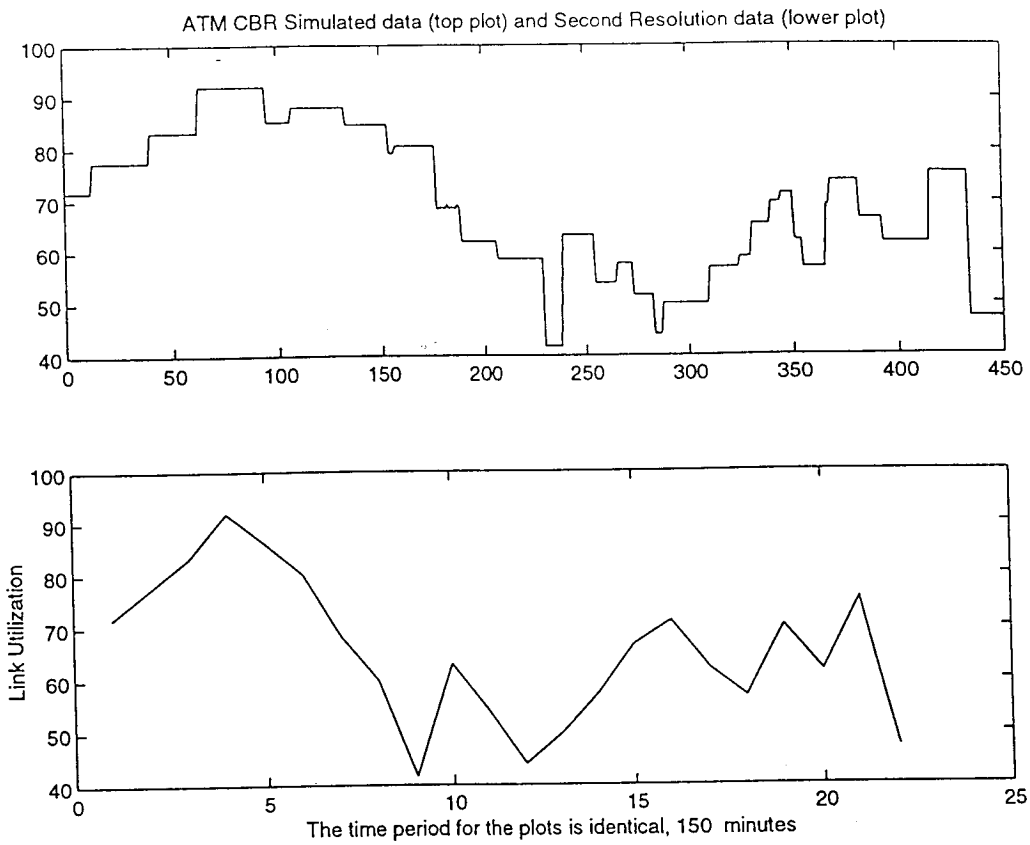

Figure 4.8: Second resolution and original data, ATM CBR simulation.

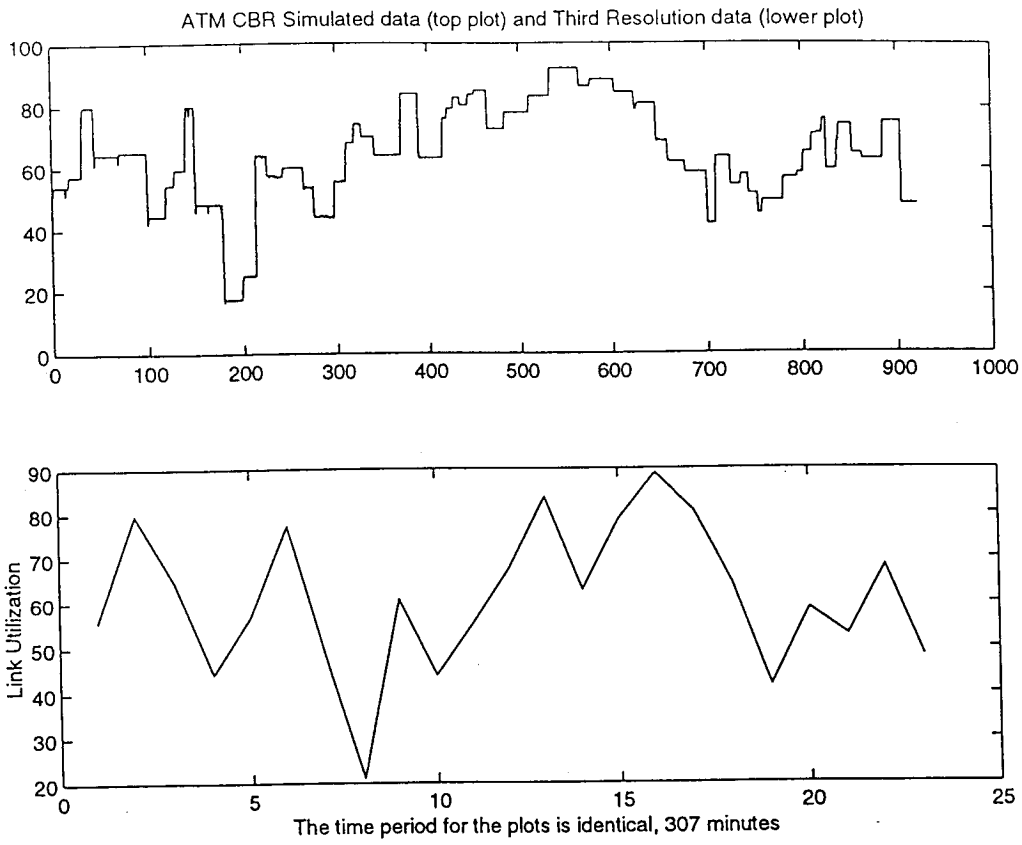

Figure 4.9: Third resolution and original data, ATM CBR simulation.

4.2.1 Data Reduction

As seen above, the proposed algorithm affects a 55% reduction in the raw data collected. The means by which the algorithm achieves this reduction draws some contrast with the methods used in commercially available data reduction packages. In such data reduction packages, there is a reliance on a priori information based on an understanding of the application space as well as the characteristics of the entity being monitored or examined. This requirement for a priori information can be quite costly, especially in newly developing fields where the necessary research is yet to be conducted and the requisite base of knowledge has not been developed [7,27,28,42]. As an example of data reduction scenarios where information of the application space is necessary, consider the StatLIA product introduced in Chapter 1. For the product to effectively reduce data, the application must have curves of known concentrations of various chemicals in the solution in order to perform the back fit. Without such curves, the data reduction package cannot be effective [7].

The proposed algorithm does not share this dependency as it does not attempt to back fit known statistical distributions on the collected data, but observes the movement of the data itself to determine the time intervals over which to perform the aggregation. Removing the need for a priori information greatly simplifies both the algorithm itself as well as its usage. While knowledge of the network and the particular component from which the network statistic is generated can help in establishing the values of δ and ε, it

is not necessary to have any historical values when processing the data. This fact can allow the algorithm to be modified so as to be applicable to networks and statistics different from those discussed in this paper.

Commercially available schemes are also highly complex as compared to the proposed algorithm. The back fitting of known data curves onto raw data is typically a complex mathematical process. As mentioned above, the proposed algorithm does not require any such curve fitting. At its core, this algorithm repeatedly performs comparisons of two values and aggregation of two values. The complexity of such operations is significantly less than a back fit onto a data set of a cubic spline or a logarithmic, non-linear curve as in the examples in the Introduction [7].

The proposed algorithm has these two benefits as compared with commercially available data reduction schemes. These benefits, along with the percentage of data reduction achieved make this an attractive and simple way of reducing the size of the network management data to be stored.

4.3 Database Storage

The data reduction caused by the scheme is further apparent upon examination of data when stored in the databases. One hundred (100) consecutive data points generated from a running of the TCP and ATM simulation is stored in an Oracle 7 database in original form and in its second resolution form in the TIGER database. This data represents thirty-three (33) minutes of simulation time. Both sets of data are

queried to ensure the accessibility of the data as well as the benefits of temporal databases. Queries of the Oracle database are made in SQL while queries of the TIGER database are made in ATSQL, a query language derived from SQL with temporal capabilities. The databases are utilized through a web-based GUI which serves as a front-end. The database script used to enter data, along with the results appears in Section 4.4.

The particular applications and scenarios that are investigated here are instances where network managers are interested in the values of link utilization over variable length periods of time. For example, managers may wish to examine the link utilization of links in question for a certain time interval before a server failure, or before a suspected denial or service attack occurred to determine attack signatures.

The following categories of queries can be executed to retrieve this information:

1) Show the entire table for link utilization

2) What is the value of link utilization at t = <specified time point>

3) What is the value of link utilization over time interval t = <specified time interval>

Figure 4.11 illustrates how data is entered into TIGER, and shows the GUI interface for the Oracle and TIGER databases. The left hand column shows the SQL or ATSQL commands used to enter data or make queries and the right column shows their results.

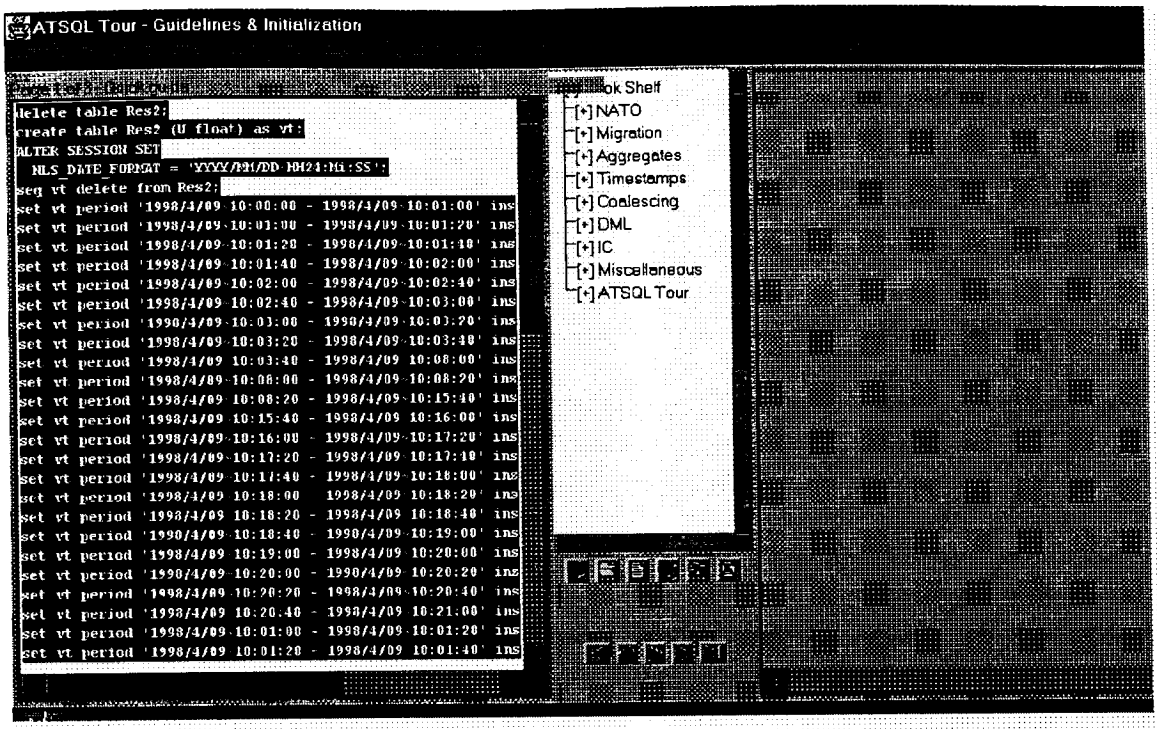

Figure 4.11. Loading data into TIGER.

The first class of queries, shown below in table 4.1 as well as in next section, returns the entire table.

Table 4.1 Class 1 Queries on the Databases

select * from orig;	seq vt select * from Res2;
(A)	(B)

Table 4.1: Class 1 Queries on the Databases. (A) illustrates the query on Oracle 7 and (B) illustrates the query on TIGER.

The results are shown in the following section.. They show the size of the database, i.e., the number of tuples required to illustrate the thirty-three (33) minutes of simulated

network time. For relational databases, 100 tuples will be required to store data for the 100 individual data points generated over this period. However, under the proposed scheme, the information of that data can be compressed into forty-six (46) data points and the temporal database allows the time interval over which they are valid to also be identified. These figures illustrate that both databases can accurately respond to these queries; they also show that the relational database is significantly larger than the temporal database covering the same time period.

A distinction between temporal and relational databases becomes apparent in the second class of queries - querying for the value at a particular time instant. As long as the time instant desired is explicitly specified in the relational database, the correct value can be returned. Such a query appears in table 4.2. In this case, the relational table 'orig' is queries for the value of link utilization at time point '10:32:20'. Data was taken at this time point, therefore it exists in the database and the requested value can be returned.

Table 4.2 Class 2 Query on a Relational Database

```
select U from orig where t1 = '10:32:20';

Response:
        .*** Executing an UC statement ***.
        U
        ----------------------------------------
        23.3766
```

Table 4.2: Class 2 Query on a Oracle 7, desired time point is in database.

However, if the time point of interest is not specified in the relational database, the desired response is not retrieved, as shown in table 4.3. In this case, the relational

database table, orig, is queried against the time point '10:19:10'. However, this point does not exist within the database as data was taken every twenty (20) seconds [data was taken at time points '10:19:00' and '10:19:20'].

Table 4.3 Class 2 Query on a Relational Database.

```
select U from orig where t1 = '10:19:10';

Response:
        .*** Executing an UC statement ***.
        U
        ----------------------------------------
```

Table 4.3: Class 2 Query on a Oracle 7, desired time point is not in database.

A temporal databases, however, can answer a query for time points explicitly specified as well as time points which are not explicitly specified. Temporal databases allow queries to be made against time points that lie within defined time intervals, therefore, time points not explicitly stated in the database but contained within any defined interval can be successfully queried against. For example, in table 4.4(a), below, the table in the temporal database is queried for utilization at time '10:32:20', a point that is in the database. The correct value of 23.3766 is returned. In table 4.4(b), a query is made for utilization at time '10:19:10', a point not explicitly in the database. Again, the correct value of 4.8485 is returned. In this case, the database took the value from the tuple in which the time interval that contained the point 10:19:10 existed.

Table 4.4 Class 2 Queries on the TIGER Database

seq vt period '1998/4/09~10:32:20' select U from Res2;
Response:

```
VT  U
---------------------------------------
1998/04/09~10:15:20-1998/04/09~10:15:20 23.3766
```

(A)

seq vt period '1998/4/09~10:19:10' select U from Res2;

Response:

```
VT  U
---------------------------------------
1998/04/09~10:19:10-1998/04/09~10:19:10  4.8485
```

(B)

Table 4.4: Class 2 Queries on the TIGER Database. (A) the queried time point is in the database and (B) the queried time point is not in the database.

In the third class of queries mentioned above, TIGER, and temporal databases, again distinguish themselves from relational databases. As relational databases store snapshots of data, querying for data over several time instants implies multiple queries. For example, table 4.5 illustrates the queries necessary to retrieve link utilization from time 10:15:00 to 10:20:00.

Table 4.5 Class 3 Queries on an Oracle 7 Relational Database

```
select U from orig where t1 = '10:15:00';
select U from orig where t1 = '10:15:20';
select U from orig where t1 = '10:15:40';
select U from orig where t1 = '10:16:00';
select U from orig where t1 = '10:16:20';
select U from orig where t1 = '10:16:40';
select U from orig where t1 = '10:17:00';
select U from orig where t1 = '10:17:20';
select U from orig where t1 = '10:17:40';
select U from orig where t1 = '10:18:00';
select U from orig where t1 = '10:18:20';
select U from orig where t1 = '10:18:40';
select U from orig where t1 = '10:19:00';
select U from orig where t1 = '10:19:20';
select U from orig where t1 = '10:19:40';
select U from orig where t1 = '10:20:00';
```

Table 4.5: Class 3 Queries on an Oracle 7. 'U' is the name of the Utilization attribute, 'orig' is the name of the table, and 't1' is the name of the time attribute.

The database must be separately queried for each time point specified within the interval of interest. In temporal databases, this is not the case. As time is stored as intervals, time can be queried as intervals. TIGER requires only one query to retrieve the link utilization data over a specified time interval, as in Table 4.6.

Table 4.6 Time Interval Queries on the TIGER Database.

```
seq vt period '1998/4/09~10:15:00 - 1998/4/09~10:20:00' select * from Res2;
```

Table 4.6: Class 3 Query on Tiger. Res2 is the name of the table, and * specifies to retrieve the entire database between the specified time points.

The responses to the queries in the above two tables are presented in an entire script appearing in the next section of this chapter. In the Oracle case, the data is presented one query at a time. Therefore, the format is not as convenient for network

managers as the format for the response to a single query made against the temporal database where all the data can be presented in a single view.

There are many benefits to this single time interval query as opposed to multiple time point queries against relational databases. With multiple queries, there are multiple hits on the database server, multiple searches on the database, and multiple responses returned to the user. Each of these delays information being presented to the user. This is even more pronounced in distributed databases where queries may be generated at remote sites from the central server. The capability of employing single, time interval queries allows this delay to be minimized.

Also, single queries are more user-friendly as network managers can specify the time period of interest in one query as opposed to generating several queries requesting data over the desired period.

4.4 Script for Storing and Retrieving Data from the Databases

The script for storing and retrieving data into both the Oracle 7 relational and TIGER temporal databases is below.

```
drop table orig;
create table orig (U float, t1 char(14));
insert into orig values (3.1169, '10:00:00');
insert into orig values (3.1169, '10:00:20');
insert into orig values (7.1948, '10:00:40');
insert into orig values (14.6234, '10:01:00');
insert into orig values (20.7792, '10:01:20');
insert into orig values (29.6104, '10:01:40');
insert into orig values (18.1818, '10:02:00');
insert into orig values (21.2987, '10:02:20');
```

```
insert into orig values (27.0130, '10:02:40');
insert into orig values (36.3636, '10:03:00');
insert into orig values (20.2592, '10:03:20');
insert into orig values (5.3766, '10:03:40');
insert into orig values (7.7922, '10:04:00');
insert into orig values (4.6753, '10:04:20');
insert into orig values (4.6753, '10:04:40');
insert into orig values (4.6753, '10:05:00');
insert into orig values (4.6753, '10:05:20');
insert into orig values (4.6753, '10:05:40');
insert into orig values (4.6753, '10:06:00');
insert into orig values (4.6753, '10:06:20');
insert into orig values (4.6753, '10:06:40');
insert into orig values (4.6753, '10:07:00');
insert into orig values (6.2338, '10:07:20');
insert into orig values (9.3506, '10:07:40');
insert into orig values (17.6623, '10:08:00');
insert into orig values (6.2338, '10:08:20');
insert into orig values (4.7635, '10:08:40');
insert into orig values (5.1311, '10:09:00');
insert into orig values (4.5378, '10:09:20');
insert into orig values (4.6753, '10:09:40');
insert into orig values (4.6753, '10:10:00');
insert into orig values (4.6753, '10:10:20');
insert into orig values (4.6631, '10:10:40');
insert into orig values (4.6744, '10:11:00');
insert into orig values (4.8454, '10:11:20');
insert into orig values (4.6465, '10:11:40');
insert into orig values (4.6752, '10:12:00');
insert into orig values (4.7586, '10:12:20');
insert into orig values (4.9553, '10:12:40');
insert into orig values (4.9463, '10:13:00');
insert into orig values (4.6752, '10:13:20');
insert into orig values (4.2666, '10:13:40');
insert into orig values (4.6985, '10:14:00');
insert into orig values (6.2338, '10:14:20');
insert into orig values (10.9091, '10:14:40');
insert into orig values (3.6364, '10:15:00');
insert into orig values (6.2338, '10:15:20');
insert into orig values (36.3636, '10:15:40');
insert into orig values (1.5584, '10:16:00');
insert into orig values (1.5584, '10:16:20');
insert into orig values (1.5584, '10:16:40');
insert into orig values (2.5974, '10:17:00');
insert into orig values (9.3506, '10:17:20');
insert into orig values (16.1039, '10:17:40');
insert into orig values (23.8961, '10:18:00');
```

```
insert into orig values (31.1688, '10:18:20');
insert into orig values (21.2987, '10:18:40');
insert into orig values (5.7143, '10:19:00');
insert into orig values (1.5584, '10:19:20');
insert into orig values (7.2727, '10:19:40');
insert into orig values (14.5455, '10:20:00');
insert into orig values (21.8182, '10:20:20');
insert into orig values (31.0649, '10:20:40');
insert into orig values (40.0000, '10:21:00');
insert into orig values (43.1169, '10:21:20');
insert into orig values (46.7532, '10:21:40');
insert into orig values (54.0260, '10:22:00');
insert into orig values (60.7792, '10:22:20');
insert into orig values (69.0909, '10:22:40');
insert into orig values (76.8831, '10:23:00');
insert into orig values (82.0779, '10:23:20');
insert into orig values (82.5974, '10:23:40');
insert into orig values (60.2597, '10:24:00');
insert into orig values (43.1169, '10:24:20');
insert into orig values (41558, '10:24:40');
insert into orig values (1.5584, '10:25:00');
insert into orig values (2.0779, '10:25:20');
insert into orig values (7.727, '10:25:40');
insert into orig values (14.5455, '10:26:00');
insert into orig values (21.2987, '10:26:20');
insert into orig values (23.3766, '10:26:40');
insert into orig values (1.5584, '10:27:00');
insert into orig values (1.5584, '10:27:20');
insert into orig values (1.5584, '10:27:40');
insert into orig values (1.7297, '10:28:00');
insert into orig values (2.3634, '10:28:20');
insert into orig values (4.6753, '10:28:40');
insert into orig values (10.3896, '10:29:00');
insert into orig values (17.6623, '10:29:20');
insert into orig values (25.3766, '10:29:40');
insert into orig values (33.8442, '10:30:00');
insert into orig values (24.4156, '10:30:20');
insert into orig values (1.5584, '10:30:40');
insert into orig values (1.5584, '10:31:00');
insert into orig values (3.1169, '10:31:20');
insert into orig values (8.7532, '10:31:40');
insert into orig values (16.1818, '10:32:00');
insert into orig values (23.3766, '10:32:20');
insert into orig values (31.6883, '10:32:40');
insert into orig values (36.8831, '10:33:00');

/* Create Temporal Table
```

```
*/

delete table Res2;
create table Res2 (U float) as vt;
ALTER SESSION SET
 NLS_DATE_FORMAT = 'YYYY/MM/DD~HH24:Mi:SS';
seq vt delete from Res2;
set vt period '1998/4/09~10:00:00 - 1998/4/09~10:01:00' insert into Res2 values (4.4762);
set vt period '1998/4/09~10:01:00 - 1998/4/09~10:01:20' insert into Res2 values (14.6234);
set vt period '1998/4/09~10:01:20 - 1998/4/09~10:01:40' insert into Res2 values (20.7792);
set vt period '1998/4/09~10:01:40 - 1998/4/09~10:02:00' insert into Res2 values (29.6104);
set vt period '1998/4/09~10:02:00 - 1998/4/09~10:02:40' insert into Res2 values (19.7402);
set vt period '1998/4/09~10:02:40 - 1998/4/09~10:03:00' insert into Res2 values (27.0130);
set vt period '1998/4/09~10:03:00 - 1998/4/09~10:03:20' insert into Res2 values (36.3636);
set vt period '1998/4/09~10:03:20 - 1998/4/09~10:03:40' insert into Res2 values (20.2592);
set vt period '1998/4/09~10:03:40 - 1998/4/09~10:08:00' insert into Res2 values (5.5130);
set vt period '1998/4/09~10:08:00 - 1998/4/09~10:08:20' insert into Res2 values (17.6623);
set vt period '1998/4/09~10:08:20 - 1998/4/09~10:15:40' insert into Res2 values (5.1240);
set vt period '1998/4/09~10:15:40 - 1998/4/09~10:16:00' insert into Res2 values (36.3636);
set vt period '1998/4/09~10:16:00 - 1998/4/09~10:17:20' insert into Res2 values (1.8182);
set vt period '1998/4/09~10:17:20 - 1998/4/09~10:17:40' insert into Res2 values (9.3506);
set vt period '1998/4/09~10:17:40 - 1998/4/09~10:18:00' insert into Res2 values (16.1039);
set vt period '1998/4/09~10:18:00 - 1998/4/09~10:18:20' insert into Res2 values (23.8961);
set vt period '1998/4/09~10:18:20 - 1998/4/09~10:18:40' insert into Res2 values (31.1688);
set vt period '1998/4/09~10:18:40 - 1998/4/09~10:19:00' insert into Res2 values (21.2987);
set vt period '1998/4/09~10:19:00 - 1998/4/09~10:20:00' insert into Res2 values (4.8485);
set vt period '1998/4/09~10:20:00 - 1998/4/09~10:20:20' insert into Res2 values (14.5455);
set vt period '1998/4/09~10:20:20 - 1998/4/09~10:20:40' insert into Res2 values (21.8182);
set vt period '1998/4/09~10:20:40 - 1998/4/09~10:21:00' insert into Res2 values (31.0649);
set vt period '1998/4/09~10:01:00 - 1998/4/09~10:01:20' insert into Res2 values (14.6234);
set vt period '1998/4/09~10:01:20 - 1998/4/09~10:01:40' insert into Res2 values (20.7792);
set vt period '1998/4/09~10:01:40 - 1998/4/09~10:02:00' insert into Res2 values (29.6104);
set vt period '1998/4/09~10:02:00 - 1998/4/09~10:02:40' insert into Res2 values (19.7402);
set vt period '1998/4/09~10:02:40 - 1998/4/09~10:03:00' insert into Res2 values (27.0130);
set vt period '1998/4/09~10:03:00 - 1998/4/09~10:03:20' insert into Res2 values (36.3636);
set vt period '1998/4/09~10:03:20 - 1998/4/09~10:03:40' insert into Res2 values (20.2592);
set vt period '1998/4/09~10:03:40 - 1998/4/09~10:08:00' insert into Res2 values (5.5130);
set vt period '1998/4/09~10:08:00 - 1998/4/09~10:08:20' insert into Res2 values (17.6623);
set vt period '1998/4/09~10:08:20 - 1998/4/09~10:15:40' insert into Res2 values (5.1240);
set vt period '1998/4/09~10:15:40 - 1998/4/09~10:16:00' insert into Res2 values (36.3636);
set vt period '1998/4/09~10:16:00 - 1998/4/09~10:17:20' insert into Res2 values (1.8182);
set vt period '1998/4/09~10:17:20 - 1998/4/09~10:17:40' insert into Res2 values (9.3506);
set vt period '1998/4/09~10:17:40 - 1998/4/09~10:18:00' insert into Res2 values (16.1039);
set vt period '1998/4/09~10:18:00 - 1998/4/09~10:18:20' insert into Res2 values (23.8961);
set vt period '1998/4/09~10:18:20 - 1998/4/09~10:18:40' insert into Res2 values (31.1688);
set vt period '1998/4/09~10:18:40 - 1998/4/09~10:19:00' insert into Res2 values (21.2987);
set vt period '1998/4/09~10:19:00 - 1998/4/09~10:20:00' insert into Res2 values (4.8485);
```

```
set vt period '1998/4/09~10:20:00 - 1998/4/09~10:20:20' insert into Res2 values (14.5455);
set vt period '1998/4/09~10:20:20 - 1998/4/09~10:20:40' insert into Res2 values (21.8182);
set vt period '1998/4/09~10:20:40 - 1998/4/09~10:21:00' insert into Res2 values(31.0649);
set vt period '1998/4/09~10:21:00 - 1998/4/09~10:21:40' insert into Res2 values (41.5585);
set vt period '1998/4/09~10:21:40 - 1998/4/09~10:22:00' insert into Res2 values (46.7532);
set vt period '1998/4/09~10:22:00 - 1998/4/09~10:22:20' insert into Res2 values (54.0206);
set vt period '1998/4/09~10:22:20 - 1998/4/09~10:22:40' insert into Res2 values (60.7792);
set vt period '1998/4/09~10:22:40 - 1998/4/09~10:22:40' insert into Res2 values (69.0909);
set vt period '1998/4/09~10:23:00 - 1998/4/09~10:23:20' insert into Res2 values (76.8831);
set vt period '1998/4/09~10:23:20 - 1998/4/09~10:24:00' insert into Res2 values (82.3376);
set vt period '1998/4/09~10:24:00 - 1998/4/09~10:24:20' insert into Res2 values (60.2597);
set vt period '1998/4/09~10:24:20 - 1998/4/09~10:25:00' insert into Res2 values (43.1169);
set vt period '1998/4/09~10:25:00 - 1998/4/09~10:25:40' insert into Res2 values (3.7662);
set vt period '1998/4/09~10:25:40 - 1998/4/09~10:26:20' insert into Res2 values (14.5455);
set vt period '1998/4/09~10:26:20 - 1998/4/09~10:27:00' insert into Res2 values (22.3377);
set vt period '1998/4/09~10:27:00 - 1998/4/09~10:29:00' insert into Res2 values (2.0037);
set vt period '1998/4/09~10:29:00 - 1998/4/09~10:29:20' insert into Res2 values (10.3896);
set vt period '1998/4/09~10:29:20 - 1998/4/09~10:29:40' insert into Res2 values (17.6623);
set vt period '1998/4/09~10:29:40 - 1998/4/09~10:30:00' insert into Res2 values (25.3766);
set vt period '1998/4/09~10:30:00 - 1998/4/09~10:30:20' insert into Res2 values (33.8442);
set vt period '1998/4/09~10:30:20 - 1998/4/09~10:30:40' insert into Res2 values (24.4156);
set vt period '1998/4/09~10:30:40 - 1998/4/09~10:31:40' insert into Res2 values (2.0779);
set vt period '1998/4/09~10:31:40 - 1998/4/09~10:32:00' insert into Res2 values (8.7532);
set vt period '1998/4/09~10:32:00 - 1998/4/09~10:32:20' insert into Res2 values (16.1818);
set vt period '1998/4/09~10:32:20 - 1998/4/09~10:32:40' insert into Res2 values (23.3766);
set vt period '1998/4/09~10:32:40 - 1998/4/09~10:33:00' insert into Res2 values (31.6883);
set vt period '1998/4/09~10:33:00 - 1998/4/09~10:33:20' insert into Res2 values (36.8831);


/* Query Class 1) Retrieve the entire table.

Relational Database
*/

select * from orig

/* Temporal Database
*/

seq vt select * from Res2;

/* Response of the aboe two queries, respectively.
*/

.*** Executing an UC statement ***.

U T1
```

3.1169 10:00:00
3.1169 10:00:20
7.1948 10:00:40
14.6234 10:01:00
20.7792 10:01:20
29.6104 10:01:40
18.1818 10:02:00
21.2987 10:02:20
27.013 10:02:40
36.3636 10:03:00
20.2592 10:03:20
5.3766 10:03:40
7.7922 10:04:00
4.6753 10:04:20
4.6753 10:04:40
4.6753 10:05:00
4.6753 10:05:20
4.6753 10:05:40
4.6753 10:06:00
4.6753 10:06:20
4.6753 10:06:40
4.6753 10:07:00
6.2338 10:07:20
9.3506 10:07:40
17.6623 10:08:00
6.2338 10:08:20
4.7635 10:08:40
5.1311 10:09:00
4.5378 10:09:20
4.6753 10:09:40
4.6753 10:10:00
4.6753 10:10:20
4.6631 10:10:40
4.6744 10:11:00
4.8454 10:11:20
4.6465 10:11:40
4.6752 10:12:00
4.7586 10:12:20
4.9553 10:12:40
4.9463 10:13:00
4.6752 10:13:20
4.2666 10:13:40
4.6985 10:14:00
6.2338 10:14:20
10.9091 10:14:40
3.6364 10:15:00

6.2338 10:15:20
36.3636 10:15:40
1.5584 10:16:00
1.5584 10:16:20
1.5584 10:16:40
2.5974 10:17:00
9.3506 10:17:20
16.1039 10:17:40
23.8961 10:18:00
31.1688 10:18:20
21.2987 10:18:40
5.7143 10:19:00
1.5584 10:19:20
7.2727 10:19:40
14.5455 10:20:00
21.8182 10:20:20
31.0649 10:20:40
40 10:21:00
43.1169 10:21:20
46.7532 10:21:40
54.026 10:22:00
60.7792 10:22:20
69.0909 10:22:40
76.8831 10:23:00
82.0779 10:23:20
82.5974 10:23:40
60.2597 10:24:00
43.1169 10:24:20
41558 10:24:40
1.5584 10:25:00
2.0779 10:25:20
7.727 10:25:40
14.5455 10:26:00
21.2987 10:26:20
23.3766 10:26:40
1.5584 10:27:00
1.5584 10:27:20
1.5584 10:27:40
1.7297 10:28:00
2.3634 10:28:20
4.6753 10:28:40
10.3896 10:29:00
17.6623 10:29:20
25.3766 10:29:40
33.8442 10:30:00
24.4156 10:30:20
1.5584 10:30:40

1.5584 10:31:00
3.1169 10:31:20
8.7532 10:31:40
16.1818 10:32:00
23.3766 10:32:20
31.6883 10:32:40
36.8831 10:33:00

/* Temporal Database
*/

VT U

1998/04/09~10-1998/04/09~10:01:00 4.4762
1998/04/09~10:01-1998/04/09~10:01:20 14.6234
1998/04/09~10:01:20-1998/04/09~10:01:40 20.7792
1998/04/09~10:01:40-1998/04/09~10:02:00 29.6104
1998/04/09~10:02-1998/04/09~10:02:40 19.7402
1998/04/09~10:02:40-1998/04/09~10:03:00 27.013
1998/04/09~10:03-1998/04/09~10:03:20 36.3636
1998/04/09~10:03:20-1998/04/09~10:03:40 20.2592
1998/04/09~10:03:40-1998/04/09~10:08:00 5.513
1998/04/09~10:08-1998/04/09~10:08:20 17.6623
1998/04/09~10:08:20-1998/04/09~10:15:40 5.124
1998/04/09~10:15:40-1998/04/09~10:16:00 36.3636
1998/04/09~10:16-1998/04/09~10:17:20 1.8182
1998/04/09~10:17:20-1998/04/09~10:17:40 9.3506
1998/04/09~10:17:40-1998/04/09~10:18:00 16.1039
1998/04/09~10:18-1998/04/09~10:18:20 23.8961
1998/04/09~10:18:20-1998/04/09~10:18:40 31.1688
1998/04/09~10:18:40-1998/04/09~10:19:00 21.2987
1998/04/09~10:19-1998/04/09~10:20:00 4.8485
1998/04/09~10:20-1998/04/09~10:20:20 14.5455
1998/04/09~10:20:20-1998/04/09~10:20:40 21.8182
1998/04/09~10:20:40-1998/04/09~10:21:00 31.0649
1998/04/09~10:01-1998/04/09~10:01:20 14.6234
1998/04/09~10:01:20-1998/04/09~10:01:40 20.7792
1998/04/09~10:01:40-1998/04/09~10:02:00 29.6104
1998/04/09~10:02-1998/04/09~10:02:40 19.7402
1998/04/09~10:02:40-1998/04/09~10:03:00 27.013
1998/04/09~10:03-1998/04/09~10:03:20 36.3636
1998/04/09~10:03:20-1998/04/09~10:03:40 20.2592
1998/04/09~10:03:40-1998/04/09~10:08:00 5.513
1998/04/09~10:08-1998/04/09~10:08:20 17.6623
1998/04/09~10:08:20-1998/04/09~10:15:40 5.124
1998/04/09~10:15:40-1998/04/09~10:16:00 36.3636

1998/04/09~10:16-1998/04/09~10:17:20 1.8182
1998/04/09~10:17:20-1998/04/09~10:17:40 9.3506
1998/04/09~10:17:40-1998/04/09~10:18:00 16.1039
1998/04/09~10:18-1998/04/09~10:18:20 23.8961
1998/04/09~10:18:20-1998/04/09~10:18:40 31.1688
1998/04/09~10:18:40-1998/04/09~10:19:00 21.2987
1998/04/09~10:19-1998/04/09~10:20:00 4.8485
1998/04/09~10:20-1998/04/09~10:20:20 14.5455
1998/04/09~10:20:20-1998/04/09~10:20:40 21.8182
1998/04/09~10:20:40-1998/04/09~10:21:00 31.0649
1998/04/09~10:21-1998/04/09~10:21:40 41.5585
1998/04/09~10:21:40-1998/04/09~10:22:00 46.7532
1998/04/09~10:22-1998/04/09~10:22:20 54.0206
1998/04/09~10:22:20-1998/04/09~10:22:40 60.7792
1998/04/09~10:22:40-1998/04/09~10:22:40 69.0909
1998/04/09~10:23-1998/04/09~10:23:20 76.8831
1998/04/09~10:23:20-1998/04/09~10:24:00 82.3376
1998/04/09~10:24-1998/04/09~10:24:20 60.2597
1998/04/09~10:24:20-1998/04/09~10:25:00 43.1169
1998/04/09~10:25-1998/04/09~10:25:40 3.7662
1998/04/09~10:25:40-1998/04/09~10:26:20 14.5455
1998/04/09~10:26:20-1998/04/09~10:27:00 22.3377
1998/04/09~10:27-1998/04/09~10:29:00 2.0037
1998/04/09~10:29-1998/04/09~10:29:20 10.3896
1998/04/09~10:29:20-1998/04/09~10:29:40 17.6623
1998/04/09~10:29:40-1998/04/09~10:30:00 25.3766
1998/04/09~10:30-1998/04/09~10:30:20 33.8442
1998/04/09~10:30:20-1998/04/09~10:30:40 24.4156
1998/04/09~10:30:40-1998/04/09~10:31:40 2.0779
1998/04/09~10:31:40-1998/04/09~10:32:00 8.7532
1998/04/09~10:32-1998/04/09~10:32:20 16.1818
1998/04/09~10:32:20-1998/04/09~10:32:40 23.3766
1998/04/09~10:32:40-1998/04/09~10:33:00 31.6883
1998/04/09~10:33-1998/04/09~10:33:20 36.8831

/* Query Class 2 Query against a particular time point.

The queries from the Relational and Tiger Proof.
*/

select U from orig where t1 = '10:32:20';

.*** Executing an UC statement ***.

U

```
23.3766

select U from orig where t1 = '10:19:10';

.*** Executing an UC statement ***.

U

----------------------------------------


/* Temporal
*/

seq vt period '1998/4/09~10:32:20' select U from Res2;


VT  U
----------------------------------------
1998/04/09~10:15:20-1998/04/09~10:15:20 23.3766

seq vt period '1998/4/09~10:19:10' select U from Res2;


VT  U
----------------------------------------
1998/04/09~10:19:10-1998/04/09~10:19:10  4.8485


/* Query Class 3. Request the values over a range of times.
*/

select U from orig where t1 = '10:15:00';
select U from orig where t1 = '10:15:20';
select U from orig where t1 = '10:15:40';
select U from orig where t1 = '10:16:00';
select U from orig where t1 = '10:16:20';
select U from orig where t1 = '10:16:40';
select U from orig where t1 = '10:17:00';
select U from orig where t1 = '10:17:20';
select U from orig where t1 = '10:17:40';
select U from orig where t1 = '10:18:00';
select U from orig where t1 = '10:18:20';
select U from orig where t1 = '10:18:40';
select U from orig where t1 = '10:19:00';
select U from orig where t1 = '10:19:20';
```

```
select U from orig where t1 = '10:19:40';
select U from orig where t1 = '10:20:00';


.*** Executing an UC statement ***.

U
----------------------------------------
3.6364

.*** Executing an UC statement ***.

U
----------------------------------------
6.2338

.*** Executing an UC statement ***.

U
----------------------------------------
36.3636

.*** Executing an UC statement ***.

U
----------------------------------------
1.5584

.*** Executing an UC statement ***.

U
----------------------------------------
1.5584

.*** Executing an UC statement ***.

U
----------------------------------------
1.5584

.*** Executing an UC statement ***.

U
----------------------------------------
2.5974

.*** Executing an UC statement ***.
```

```
U
----------
9.3506

.*** Executing an UC statement ***.

U
----------
16.1039

.*** Executing an UC statement ***.

U
----------
23.8961

.*** Executing an UC statement ***.

U
----------
31.1688

.*** Executing an UC statement ***.

U
----------
21.2987

.*** Executing an UC statement ***.

U
----------
5.7143

.*** Executing an UC statement ***.

U
----------
1.5584

.*** Executing an UC statement ***.

U
----------
7.2727
```

.*** Executing an UC statement ***.

U

14.5455

seq vt period '1998/4/09~10:15:00 - 1998/4/09~10:20:00' select * from Res2;

1998/04/09~10:15-1998/04/09~10:15:40 5.124
1998/04/09~10:15:40-1998/04/09~10:16:00 36.3636
1998/04/09~10:16-1998/04/09~10:17:20 1.8182
1998/04/09~10:17:20-1998/04/09~10:17:40 9.3506
1998/04/09~10:17:40-1998/04/09~10:18:00 16.1039
1998/04/09~10:18-1998/04/09~10:18:20 23.8961
1998/04/09~10:18:20-1998/04/09~10:18:40 31.1688
1998/04/09~10:18:40-1998/04/09~10:19:00 21.2987
1998/04/09~10:19-1998/04/09~10:20:00 4.8485

Chapter 5

Applications of the Data Storage Scheme

The scheme proposed here for data storage in temporal databases has two advantages. These advantages are the scheme's ability to affect a reduction in the quantity of data stored in memory as well as the ability to highlight the movement of the data over time.

While past or historical network management data is valuable in the management of a network, as the data ages, it does lose its individual value in the sense that knowledge of the precise utilization rate of a particular link at a particular time (7:00am on January 19th, 1998, for example) in the past is in itself an unimportant piece of information. What is important, what is of value to the process of network management, is knowing how the utilization rate at that time point fits into the link utilization pattern taken around that time point. In other words, knowing that the data point in question is part of a sudden rise in link utilization can be useful.

This implies that storing the trends minimizes the need for storing the voluminous data generated in network management. Applications of the scheme's data reduction and trend identification are discussed in the following sections.

5.1 Data Reduction

It is clear from inspection of a MIB from any of the various standards bodies (e.g., IETF, ATM Forum) that networks have the capacity to generate enough data over time to exceed the storage and retrieval capabilities of most databases. Therefore, any data storage scheme needs to preserve the information contained in the overall data without having to store all collected data points. The proposed scheme has this characteristic. It reduces the amount of historical link utilization data by partitioning the data into groups (size determined by δ and ε) and taking the aggregate of the data that falls within each group. This aggregation has its most pronounced effect when network links are stable and experience constant utilization.

These conditions arise in the case of ATM constant bit rate (CBR) traffic. An example of this is demonstrated by simulations in the previous chapter. Also, traffic shaping can force links to exhibit constant and stable utilization through manipulation of the shaping parameters. These parameters can be selected such that the traffic will be stable and relatively constant, essentially, the parameters can be selected so as to optimize data aggregation.

Other scenarios in which this scheme can effectively reduce historical data is in the link utilization for a corporate Intranet environment when the network typically experiences low off-hours (e.g., evenings, weekends, and holidays) usage.

Also, in large scale networks, the large number of users can balance out the overall link utilization so that it appear stable and constant, even if the individual users streams are not constant but varying. This is a design goal of ATM traffic and a

property of statistical multiplexing.

5.2 Trend Identification

Another area in which the proposed scheme can be applicable is that of developing baseline models of the network. Baseline models are used often in proactive network management [15,20,24,30,31,33,34]. The capability of this scheme to highlight trends in the data is directly applicable to this endeavor.

5.2.1 Proactive Approach to Network Management

For companies that demand their computer networks remain in operation 100% of the time, where even one second of downtime can be disastrous, there is a growing consensus within the network management industry that proactive management must be employed to meet this requirement [15,20,24,30,31,33,34]. Proactive network management is, in its most simple definition, taking action to avert a critical situation before a situation becomes critical, similar in concept to preventive medicine.

In order to effectively perform proactive network management, some sort of knowledge of the behavior of network users, as well as the usage trends and patterns experienced by the network must be available to the network manager [15,20,24,30,31,33,34]. Typically, this data is stored in views in a database.

The first step in proactive management is to perform a long term study of the network and gather relevant network statistics; analyze them off line to determine performance and quality of service (QoS) thresholds for the network.

These baselines are used to develop models of network behavior for various operations. A particular model may, for instance, detail a user log in process. These models, or profiles are used, along with the baseline performance levels developed, for simulating the network under particular conditions. A common use of such a simulation is to determine the effect of adding new applications to the existing network infrastructure [16,22,30].

The task of baselining the network as described can be time-intensive, involving long man-hours of the network management staff as it often requires a long-term study of the network, involving data collection over weeks, possibly months [16,22,30,36]. The scheme discussed here can be used as a means of determining this baseline value on-line possibly saving a tremendous amount of time and manpower.

Several examples of situations in which such an analysis of network trends can help prepare managers to respond quickly and effectively to adverse situations in the network exist in the literature. For example, when there is prior knowledge that an adverse situation is going to arise, such as the common pattern of increased electronic mail activity towards the end of the business day or the spike in hits to the server at the start of business when employees are logging onto their accounts [16,22]. These events can be recorded through the procedure of baselining and characteristics of their occurrence along with strategies for dealing with them can be integrated to the network

management approach.

As another example, if a network link or set of links in close proximity are being overloaded during the same hours each day leading to congestion, then the network manager may decide to hire a high speed, high bandwidth satellite link for that duration of time to eliminate the congestion and increase productivity. A common theme amongst such cases is that having the trend of the data present and available is an important first step in this sort of management scheme.

A typical approach to implementing a proactive management scheme is to perform the baseline and apply various proactive management algorithms on the identified trends. The various algorithms proposed in the literature allow such functionality as described above. In [46,47] Bakshi and Stephanopoulos describe procedures to apply induction decision trees to trends identified from process data in order to gain insight into the chemical process itself and assist with the supervision of the process. Similar scenarios can exist in the network management arena.

Chapter 6

Conclusions

The proposed scheme retains the information contained within the data while being able to discard the actual, measured data values. This is accomplished by taking aggregates of the data over time. Therefore, the overall trend in the data will be retained while reducing the quantity of information to be stored. This alleviates the problem of storing network management data, which can easily grow quite large over time, overwhelming network managers.

This thesis has demonstrated that temporal databases can serve the needs of network management and that the data storage scheme proposed allows a reduction in the necessary amount of data to be stored, as well as it facilitates the baselining of the network for the purposes of proactive management.

A framework for the use of temporal databases along with a temporal query language in network management has also been identified. With this temporal query language, ATSQL, it is possible to load data into and query the TIGER temporal database in an effective manner. The rich functionality of temporal databases has also been demonstrated and compared to the functionality of the standard (Oracle 7) relational database.[3] Temporal databases can employ time ranges to retrieve multiple values in a single query, whereas relational databases need to formulate multiple queries

[3] The script used to store and query the data into and from both databases has been added to the TIGER web site maintained by the Computer Science Department of the University of Aalborg.

for the same purpose. This single query can be beneficial to overall network management as it places a lesser burden on network bandwidth, which often is of consideration in a distributed environment. Single queries are also easier for the network managers to use as in one (1) query, network managers can get the information they are after, as opposed to having to generate several individual queries.

It is worthwhile to mention the place of query processors in connection with databases. Often, these processors can be configured to extrapolate from the data present in the database. For instance, it may be possible for a distributed Oracle 7 database to operate through query processors and return values for time points that are not explicitly mentioned in the database. In some cases, these processors have the ability to take single queries and construct the multiple queries that are required to respond to the query. However, these processors add another layer of processing and can add a significant source of delay to network managers [8,31]. Temporal databases do not require any query processors to offer the functionality of responding to queries of time points that are not explicitly stated, or to allow queries against time intervals. This saves organizations the cost of query processors in terms of development time and monetary expense.

The following section details possible research that can build upon the issues presented in this thesis.

6.1 Future Work

Future work towards the goals of this thesis can be to develop storage schemes for different network statistics. As has been mentioned, link utilization, CPU load and device reachabiliy were selected based upon certain characteristics they possess. Different network statistics can be expected to have different characteristics which can be exploited to accomplish data reduction and on-line trend identification. For example, for CPU memory usage, which may vary significantly from one measurement to the next as it depends on the requirements of the application currently being run, it may be sufficient to record high and low threshold crossings.

Further, the correlation between these statistics may be exploited to some degree. For example, when the number of packets transmitted on a particular link rises dramatically, one would expect the utilization across that link to also rise. Such a relationship may allow the creation of symbolic variables which can be stored in place of separately storing both the utilization rate and the number of packets of various types of traffic.

Examination of the data storage issue may also be worthwhile. As the amount of data to be stored has been reduced, often significantly, it can be expected that the overall storage space requirement also be reduced in a temporal database following the proposed scheme as compared to a traditional relational database. The techniques for data storage in relational databases may be applicable to temporal databases allowing even further storage space reduction.

An area of growing interest, both in academia and industry, is in designing strategy for the use of the trends, i.e., past history [16,22,30-32,35,36,44-47]. Mentioned above were a few basic application of trend analysis. Determining more sophisticated trends of the network, and how to implement an enterprise-wide system employing these trends in its maintenance is being actively pursued and holds promise [16,22,30-32,35,36,44-47].

The use of expert systems in network management is also drawing greater attention from the research and industrial communities in recent years. An expert system or a belief network, can be used to set, and modify as needed, the parameters of the aggregation bounds in this scheme (the δ and ε). This can allow the aggregation stages to better represent the characteristics of the particular network.

The application of trend analysis in network resource planning is also only recently been examined. Clearly there is tremendous benefit envisioned in this area. The need to upgrade networks periodically will likely be high for the next several years, (with, in some cases, the time between upgrades shrinking). It is clear that resources can be more effectively managed, upgrades can more appropriately be performed with knowledge of the network's usage [16,22,30-32,35,36]. Managers who are aware that any portion of their network routinely becomes taxed at certain times, may decide to arrange for a high bandwidth, possibly satellite link for that portion. ISP's and other network providers for whom maintaining profitability, hence the network, is a high priority, can follow the network to ensure that packet streams which are flooding the network in one area can be transfered, or re-routed to flow over another area which is

being underutilized. This and other such actions can be taken to maximize the profitablity of the network.

The scheme presented here can be optimized to serve a particular network. ATM networks have been mentioned above. In ATM networks, it may indeed be of more relevance to network management to collect VP and VC utilization in place of physical link utilization. While in corporate networks or virtual LANs, logical link utilization may be of more importance.

Appendix A

TCP and ATM (CBR and UBR) Simulation Source Code.

```
#for the purpose of random number generation
  global jran
  set jran [ns random 0]

proc create_testnet2 num {
   #building up the network with $num nodes and a single backbone link
   global r1 r2 s akn
   set r1 [ns node]
   set r2 [ns node]
   # change numbers after expr to change bandwidth of backbone link
   set L [ns_duplex $r1 $r2  1.54Mb 20ms drop-tail]
   #queue-limit for the queues
   [lindex $L 0] set queue-limit 25
   [lindex $L 1] set queue-limit 250

   for {set i 1} {$i <= $num} {incr i} {
        set ii [expr $num+$i]
        set s($i) [ns node]
        set s($ii) [ns node]
        #parameters for the outer links
        set L1 [ns_duplex $s($i) $r1 10Mb 100ms drop-tail]
        set L2 [ns_duplex $s($ii) $r2 10Mb 4ms drop-tail]
        set akn($i) 0
   }
}


proc finish1 file {

        #
        # split queue/drop events into two separate files.
        # we don't bother checking for the link we're interested in
        # since we know only such events are in our trace file
        #
        set awkCode {
                {
                        if (($1 == "+"  $1 == "-" ) && \
```

```
                    ($5 == "tcp"  $5 == "ack"  $5 == "cbr"))
                        print $2, $8 + ($11 % 90) * 0.01 >> "temp.p";
                    else if ($1 == "d")
                        print $2, $8 + ($11 % 90) * 0.01 >> "temp.d";
                }
        }

        set f [open temp.rands w]
        #puts $f "TitleText: $file"
        #puts $f "Device: Postscript"
        exit 0
}


proc openTrace { stopTime testName } {
   global r1 k1
   set traceFile [open out.tr w]
   ns at $stopTime \
           "close $traceFile ; finish1 $testName"
   set T [ns trace]
   $T attach $traceFile
   return $T
}

#returns uniformly dist.  numbers between 0.0 and 1.0
proc urand {} {
   global jran
   set jran [expr $jran*4096+150889]
   set jran [expr $jran%714025]
   set ran [expr $jran/714025.0]
   return $ran
}

 #returns numbers with a Pareto dis.
 proc prand beta {
    set ran [urand]
    set pranda [expr 1-$ran]
    set beta [expr 1/$beta]
    set pranda [expr pow($pranda,$beta)]
    #assuming minimum file size is 1024 bytes
    set pranda [expr 1024/$pranda]
        return $pranda
 }
```

```
 # returns exponentially dist.  numbers
 proc erand mean {
    set ran [urand]
    set eranda [expr -log($ran)]
    set eranda [expr $eranda*$mean]
    return $eranda
 }
# calculating utilization
proc checkingnow {num time prin} {
   global  tcp cbr cbrmon akn ftp band
   set time1 [expr $time*1000000]
    foreach num_ftp [array names ftp] {
   set akn($num_ftp) [expr [$tcp($num_ftp) get ack]- $akn($num_ftp)]
   set band($num_ftp) [expr $akn($num_ftp)*8000.0/$time1]
         #puts "$band($num_ftp) is tcp"
    }
    foreach num_cbr [array names cbr] {
   set akn($num_cbr) [expr [$cbrmon($num_cbr) set bytes_] - $akn($num_cbr)]
   set band($num_cbr) [expr $akn($num_cbr)*8.0/$time1]
         #puts "$band($num_cbr) is cbr"
    }
    set sum 0
   for {set i 1} {$i <= $num} {incr i} {
        set sum [expr $sum+$band($i)]
   }
   #puts $sum
   set sum [expr $sum*100/1.54]
   set f [open main.file a]
   if {$sum > 100} {
        set sum 100.00
   }
   if { $prin <=1 && $sum > 0} {
   puts $f "$sum"
   }
   close $f
    foreach num_ftp [array names ftp] {
   set akn($num_ftp) [$tcp($num_ftp) get ack]
    }
    foreach num_cbr [array names cbr] {
   set akn($num_cbr) [$cbrmon($num_cbr) set bytes_]
    }
```

```
}

#to check continually on the ftp connection to see if packets exhausted
#i denotes the source and r_fils denotes if addtl files remain to be sent
proc check {i r_fils} {
   global tcp ftp
   #next lines return present ack and the pkts to be sent in this connection
   #puts "tcp ack is [$tcp($i) get ack]"
   #puts "Maxpackets for source $i is [$ftp($i) set maxpkts_]"
   if {[$tcp($i) get ack] >= [expr [$ftp($i) set maxpkts_] - 2] && $r_fils} {
        set stopftp [ns now]
        #if here means that the session hasn't ended yet, so schedule new time
        set start [expr $stopftp + [erand 5]]
   } else {
        #return negative start time if connection not yet finished
        set start -1000
   }
   return $start
}


#checks every connection to see if finished and then schedules new start time
proc checking {} {
   global  r1 r2 s tcp ftp http ftp_files htp_page htpar number start
   foreach num_ftp [array names ftp] {
        #check only if session has started and has not yet ended
        if {[$tcp($num_ftp) get ack] >= 0 && $ftp_files($num_ftp)} {
          #puts "At [ns now] **********FTP**********"
          #puts "$num_ftp th user's  ftp connection"
          set start_new [check  $num_ftp $ftp_files($num_ftp)]
          if {$start_new > 0 && $ftp_files($num_ftp)} {
               #enter here only if connection ended but session not finished
               #connection ended indicated by start_new >0 and
               #session ended indicated by ftp_files(present user) >0
               incr ftp_files($num_ftp) -1
               #to decide the new connection start time
               set add_new [erand 200]
               #to decide the new connection file size
               $ftp($num_ftp) set maxpkts_ [expr [$ftp($num_ftp) set maxpkts_]+
$add_new]
               if {$ftp_files($num_ftp)} {
                  ns at $start_new "$ftp($num_ftp) start"
                  set $start($num_ftp) $start_new
```

```
                #puts "At time $start_new start source $num_ftp"
            } else {
                #to decide the new session starting time
                #deciding new session time from start_new
                #can also decide new session time from time now
                #in that case use ns now instead of start_new
                set new_session [expr $start_new + [erand 5]]
                ns at $start_new "initialize $num_ftp $new_session"
                ns at [expr $start_new -0.01] "unset ftp($num_ftp)"
            }
        }
    }
  }

}

#to initialize a connection by selectingeither tcp or cbr and then the rest of the
parameters
 proc initialize {i startn} {
    global  r1 r2 s tcp ftp http ftp_files htp_page htpar number cbr cbrmon
    set ii [expr $number +$i]
    set decide [urand]
    #change the decide cutoff values to select different applications
    puts
"**********************$i***********************************????????"
    puts "source $i starts at $startn"
    if {$decide <0.5} {
        # setting up the FTP connection
        set tcp($i) [ns_create_connection tcp-reno $s($i) tcp-sink $s($ii) $i]
        set ftp($i) [$tcp($i) source ftp]
        #next set the packet sizes for tcp
        $tcp($i) set packetSize_ 1000
        # The maximum number of packets in the first pass of FTP
        $ftp($i) set maxpkts_ [expr [prand 0.5]/1000]
        # number of passes for ftp being between 1 and 8
        set ftp_files($i) [expr int([expr [urand]*8 + 1])]
        #puts "no of ftp passes is $ftp_files($i)"
        ns at $startn "$ftp($i) start"
    } else {
        #this is to model traffic like Cu-See-Me
        puts cbr
        #set cbr($i) [ns_create_cbr $s($i) $s($ii) 500 0.01 $i]
        #remove the comments if you need to monitor cbr pkt losses etc
```

```
        set cbr($i) [ns create-agent $s($i) cbr $i]
        set cbrmon($i) [ns create-agent $s($ii) loss-monitor $i]
        ns connect $cbr($i) $cbrmon($i)
        $cbr($i) set interval_ 0.05
        $cbr($i) set packetSize_ 150
        set stop [erand 100]
        set stop [expr $startn + $stop]
        #puts $stop
        ns at $startn "$cbr($i) start"
        ns at $stop "$cbr($i) stop"
        #to determine time of new session for this source
        set new_session [expr $stop + [erand 25]]
        ns at $stop "initialize $i $new_session"
        ns at [expr $stop -0.01] "unset cbr($i)"
    }
}


#the main program
# We can set the time duration of the simulation here (stoptime)
proc test_num num {
    global  r1 r2 s tcp ftp http ftp_files htp_page htpar number start dns cbr
    set number $num
    create_testnet2 $num
    set rng [new RNG]
    $rng seed 0
    #set stoptime [$rng uniform 250 400]
    set stoptime 500
    #set stoptim [expr 4*$num]
    set testname test_num
    set start(0) 0.0
    #to initialize the different sources
    for {set i 1} {$i <= $num} {incr i} {
        set eran [erand 0.3]
        #deciding the start times for the source
        set start($i) [expr $start([expr $i-1]) + $eran]
        initialize  $i $start($i)
    }



    #next starts the continual checking part to see if connection has finished
    #granularity right now is 0.1s can be made further fine
```

```
  #for this change the increments on chec below
  if {[array size ftp]} {
        set nftp 1
        for {set chec $start($nftp)} {$chec <= $stoptime} \
                {set chec [expr $chec+0.1] } {
           ns at $chec "checking"
        }
  }

  set ainterval 1
  set perturb $ainterval
  set sam_time $perturb
  set inter [expr $stoptime/$perturb]
  puts $inter
  for {set i 1} {$i <= $inter} {incr i} {
  ns at $perturb "checkingnow $num $sam_time 1"
  set perturb [expr $perturb+$ainterval]
  }
  set sam_time  [expr $stoptime - $inter*$perturb]
  #ns at [expr $stoptime-0.001] "checkingnow $num $sam_time 1"
  #ns at [expr $stoptime-0.001] "checkingnow $num $sam_time 1"
  # trace only the bottleneck link
  ns at $stoptime "[ns link $r1 $r2] trace [openTrace $stoptime $testname ]"

  puts seed=[ns random 0]
  ns run
}

if { $argc != 2 } {
   puts stderr {usage: ns $argv [ two ]}
   exit 1
}
set j 0
foreach i $argv {
   set arg($j) $i
   incr j
}


if { "[info procs test_$arg(0)]" != "test_$arg(0)" } {
   puts stderr "$argv: no such test: $argv"
}
test_$arg(0) $arg(1)
```

Appendix B

ATM CBR Simulation Source Code.

```
 set randseed 0
proc create_testnet2 thresh {
   #building up the network with $num nodes and a single backbone link
   global r1 s1 s2 s3 s4 s5 s6 akn
   set r1 [ns node]
   set s1 [ns node]
   set s2 [ns node]
   set s3 [ns node]
   set s4 [ns node]
   set s5 [ns node]
   set s6 [ns node]
   #set capa [expr 0.4*$thresh]
   #THE BACKBONE LINK PARAMETERS TO BE ADJUSTED HERE
   set L [ns_duplex $r1 $s6 1.54Mb 999ms drop-tail]
   #queue-limit for the queues
   [lindex $L 0] set queue-limit $thresh
   [lindex $L 1] set queue-limit $thresh
   # PARAMS FOR RED Q MGT
   [lindex $L 0] set thresh_ [expr $thresh/6.0]
   [lindex $L 0] set maxthresh_ [expr $thresh/2.]
   set L1(1) [ns_duplex $s1 $r1 5Mb 30ms sfq]
   set L1(2) [ns_duplex $s2 $r1 5Mb 10ms sfq]
   set L1(3) [ns_duplex $s3 $r1 5Mb 1ms sfq]
   set L1(4) [ns_duplex $s4 $r1 5Mb 1ms sfq]
   set L1(5) [ns_duplex $s5 $r1 5Mb 1ms sfq]
   for {set i 1} {$i <= 5} {incr i} {
        set akn($i) 0
   }

}

proc finish1 file {
        #
        # split queue/drop events into two separate files.
        # we don't bother checking for the link we're interested in
        # since we know only such events are in our trace file
        #
        set awkCode {
                {
```

```
                        if (($1 == "+" ) && \
                          ($5 == "tcp"  $5 == "ack"  $5 =="cbr"))
                                print $2, $8 + ($11 % 90) * 0.01 >> "temp.p";
                        else if ($1 == "d")
                                print $2, $8 + ($11 % 90) * 0.01 >> "temp.d";
                }
        }

        #set f [open temp.rands w]
        #set f1 [open temp.drop w]
        #puts $f "TitleText: $file"
        #puts $f "Device: Postscript"

        exec rm -f temp.p temp.d
        #exec touch temp.d temp.p
        exit 0
        exec awk $awkCode out.tr
}


proc openTrace { stopTime testName } {
   global r1 k1
   set traceFile [open out.tr w]
   ns at $stopTime \
           "close $traceFile;finish1 $testName"
   set T [ns trace]
   $T attach $traceFile
   return $T
}

proc checking { thresh time prin} {
   global tcp3 tcp2 tcp1 cbrmon4 cbrmon5 akn
   exec rm -f out.tr
   set akn(1) [expr [$tcp1 get ack]- $akn(1)]
   set akn(2) [expr [$tcp2 get ack] - $akn(2)]
   set akn(3) [expr [$tcp3 get ack] - $akn(3)]
   set akn4 [expr [$cbrmon4 set bytes_] - $akn(4)]
   set akn5 [expr [$cbrmon5 set bytes_] - $akn(5)]
   set time [expr $time*1000000]
   set band1 [expr $akn(1)*800.0/$time]
   set band2 [expr $akn(2)*800.0/$time]
   set band3 [expr $akn(3)*800.0/$time]
   set band4 [expr $akn4*8.0/$time]
```

```
    set band5 [expr $akn5*8.0/$time]
    set sum [expr $band1+ $band2+ $band3+ $band4+ $band5]
    set f [open main.file a]
    if { $prin <=1} {
    puts  "$time $band1 $band2 $band3 $band4 $band5  $sum"
    puts $f "[expr $sum*100/1.54]"
    }
    close $f
    #set akn(1) [$tcp1 get ack]
    #set akn(2) [$tcp2 get ack]
    #set akn(3) [$tcp3 get ack]
    set akn(4) [$cbrmon4 set bytes]
    set akn(5) [$cbrmon5 set bytes]
}

#the main program
proc test_num thresh {
    global  r1  s1 s2 s3 s4 s5 s6 tcp3 tcp1 tcp2 cbrmon4 cbrmon5 randseed
    set thresh [expr $thresh *1]
    create_testnet2 $thresh
    set rng [new RNG]
    $rng seed 0
    #set stoptime [$rng uniform 250 550]
    set stoptime 500
    #puts $stoptime
    set testname test_num

    # setting up the FTP connection
    set tcp1 [ns_create_connection tcp $s1 tcp-sink $s6 1]
    set ftp1 [$tcp1 source ftp]
    $tcp1 set window_ 500
    $tcp1 set packetSize_ 100

    set tcp2 [ns_create_connection tcp $s2 tcp-sink $s6 2]
    set ftp2 [$tcp2 source ftp]
    $tcp2 set window_ 500
    $tcp2 set packetSize_ 100

    set tcp3 [ns_create_connection tcp $s3 tcp-sink $s6 3]
    set ftp3 [$tcp3 source ftp]
    $tcp3 set window_ 500
    $tcp3 set packetSize_ 100
    #was 100
```

```
    set cbr4 [ns create-agent $s4 cbr 4]
    set cbrmon4 [ns create-agent $s6 loss-monitor 4]
    ns connect $cbr4 $cbrmon4
    $cbr4 set interval_ 0.015
    #pktsize is in bytes here and above
    $cbr4 set packetSize_ 275


    set cbr5 [ns create-agent $s5 cbr 5]
    set cbrmon5 [ns create-agent $s6 loss-monitor 5]
    ns connect $cbr5 $cbrmon5
    $cbr5 set interval_ 0.15
    $cbr5 set packetSize_ 250


    ns at 0 "$ftp1 start"
    ns at 0 "$ftp2 start"
    ns at 0 "$ftp3 start"
    ns at 0 "$cbr5 start"
    ns at 0 "$cbr4 start"

    set perturb 20
    set sam_time $perturb
    set inter [expr $stoptime/$perturb]
    puts $inter
    for {set i 1} {$i <= $inter} {incr i} {
    ns at $perturb "checking $perturb $sam_time 1"
    set perturb [expr $perturb+20]
    }
    set sam_time  [expr $stoptime - $inter*$perturb]
    #ns at [expr $stoptime-0.001] "checking $thresh $sam_time 1"

    # trace only the bottleneck link
    ns at $stoptime "[ns link $r1 $s6] trace [openTrace $stoptime $testname ]"
    ns run
    #ns at $stoptime "finish1 $testName"
}


if { $argc != 2 } {
    puts stderr {usage: ns $argv [ two ]}
    exit 1
```

```
}
set j 0
foreach i $argv {
   set arg($j) $i
   incr j
}


if { "[info procs test_$arg(0)]" != "test_$arg(0)" } {
   puts stderr "$argv: no such test: $argv"
}
test_$arg(0) $arg(1)
```

BIBLIOGRAPHY

[1] ATM Forum Specification, M4 Network View CMIP MIB Spec v1.0 af-nm-0073.000, Jan. 1997

[2] J. Baras, M. Ball, N. Roussopoulos, J. Haritsa, A. Datta. Design of the MANDATE MIB, Integrated Network Management III, Elsevier Science Publishers, North-Holland, 1993.

[3] J. Baras, M. Ball, R. Karne, S. Kelley, K. Jang, C. Plaisant, N. Roussopoulos, K. Stathatos, A. Vakhutinsky, J. Valluri, D. Whitefield. Integrated Network Management of Hybrid Networks, 1996.

[4] J. Baras, M. Ball, R. Karne, K. Jang, S. Kelley, C. Plaisant, N. Roussopoulos, K. Stathatos, A. Vakhutinsky, J. Valluri, D. Whitefield. Hybrid Network Management. 1996.

[5] J. Baras, G. Atallah, M. Ball, S. Goli, R. Karne, S. Kelley, H. Kumar, C. Plaisant, N. Roussopoulos, B. Schneiderman, M. Srinivasarao, K. Stathatos, M. Teittinen, D. Whitefield. Next Generation Network Management Technology, American Institute of Physics. 1995.

[6] J. Baras, T. Charuhas. Hybrid (Satellite and Terrestrial) Communication Networks: Object Oriented Generic Tools for Simulation and Management, *IEEE Military Communications Conference*, 1992.

[7] Brendan Scientific Information Center, http://www.brendan.com.

[8] James Clifford, Alexander Tuzhilin. *Recent Advances in Temporal Databases.* Springer, 1995.

[9] E. Codd, A Relational Model of Data for Large Shared Databanks, *Communications of the ACM* . 13(6), June 1970.

[10] S. Conrad, M. Hoding, G. Saake, I. Schmitt, C. Turker. Schema Integration with Integrity Constraints; *15th British National Conference on Databases.* July 1997.

[11] X. Delannoy. Understanding the Tension Between Transition Rules and Confidentiality; *14th British National Conference on Databases.* July 1996.

[12] Werner Dreyer, Angelika Cittrich, Duri Schmidt. An Object-Oriented Data Model for a Time Series Management System, *IEEE Transactions on*

Knowledge and Data Engineering, 9(5), September/October 1997.

[13] B. Eaglestone. Keeping Time in Musical Database, *6th British National Conference on Databases*. July 1996.

[14] L. Ellison. *Introduction to SQL*, Oracle Corporation, 1989.

[15] G. Flach H. Meyer, Integration of Load Measurement Parameters into the Cost Evaluation of Database Queries, *14th British National Conference on Databases*. July 1996.

[16] Jim Foxworthy. Capacity Planning, Summit OnLine. wysiwyg://44/http://www.summitonline.com/netmanage/papers/landmark1.html.

[17] A. Freitas, S. Lavington. Speeding up Knowledge Discovery in Large Relational Databases by Means of a New Discretization Algorithm, *14th British National Congerence on Databases*. July 1996.

[18] S. Gadia, J. Vaishnav. A Query Language for a Homogeneous Temporal Database, *Proc. ACM SIGACT SIGMID Symposium. Principles of Database Systems*. 1985.

[19] A. Griffiths, B. Theodoulidis. SQL+: Adding Temporal Indeterminacy to the Database Language SQL, *14th British National Conference on Databases.* July 1996.

[20] S. Gupta, J. Baras, S. Kelley, N. Roussopoulos. Managing File Subsystem Data Streams for Databases on Networked Systems. 1996.

[21] J. Haritsa, M. Ball, N. Roussopoulos, A. Datta, J. Baras. MANDATE: Managing Networks Using Database Technology. *IEEE Journal on Selected Areas in Communications*. 11(9), December 1993.

[22] Steve Hoyt. Network Resource Planning for the Enterprise, Summit OnLine. wysiwyg://44/http://www.summitonline.com/netmanage/papers/make1.html.

[23] Dimitry Haskin, Steve Onishi. Management Information Base for IP Version 6, ICMPv6 Group of the Internet Engineering Task Force. http://search.ietf.org/internet-drafts/draft-ietf-ipngwg-ipv6-icmp-mib-02.txt, Jan. 1998.

[24] Innes Jelly, Jon Kerridge, Chris Bates. Benchmarking Parallel SQL Database Machines, Lecture Notes in Computer Science, *12th British National*

Conference on Databases, July 1994.

[25] Ling Lin, Tore Risch. Using a Sequential Index in Terrain-Aided Navigation. Sixth International Conference on Information and Knowledge Management. November 1997.

[26] Ling Lin, Tore Risch. Querying Continuous Time Sequences. Proceedings of the 24th VLDB Conference. 1998.

[27] Glenn E. Miller. The Data Reduction Expert Assistant, Space Telescope Science Institute, http://www.stsci.edu/~miller/draco/draco-aldb.html.

[28] Glenn E. Miller, Felix Yen. DRACO: An Expert Assistant for Data Reduction and Analysis, Space Telescope Science Institute, asp Conference Series, Vol. 61, 1994.

[29] Nikos Lorentzos, Yannis Mitsopoulos. SQL Extension for Interval Data, *IEEE Transactions on Knowledge and Data Engineering*, 9(3), May/June 1997.

[30] Network General Corporation. Proactive Solutions to the Five Most Critical Networking Problems. SummitOnLine. wysiwyg://44/http://www.summitonline.com/netmanage/papers/netgen2.html.

[31] Network General Corporation. How to Optimize Network Performance While Avoiding Unnecessary Investments, Summit OnLine. wysiwyg://44/http://www.summitonline.com/netmanage/papers/netgen1.html.

[32] Network General Corporation, Oracle Corporation. How Oracle7 Databases Impact Your Network's Performance, Summit OnLine. wysiwyg://44/http://www.summitonline.com/netmanage/papers/netgen4.html.

[33] W. Ng, C. Ravishankar. Block-Oriented Compression Techniques for Large Statistical Databases, *IEEE Transactions on Knowledge and Data Engineering*, 9(2):314-328, March-April, 1997.

[34] P. Nobecourt, C. Rolland, J. Lingat. Temporal Management in an Extended Relational DBMS, *6th British National Conference on Databases.* July 1996.

[35] Oracle Technical White Paper, http://www.oracle.com/... , June 1997.

[36] Richard Ptak, Jasmine Noel, D. Brown/ Managing Complexity Trends and Issues in Distributed Management, Summit OnLine. wysiwyg://44/http://www.summitonline.com/netmanage/papers/brown2.html.

[37] B. Rosenbach, J. Soref. RMON The Enterprise Management Standard, *Data Communications*, pp 67-72, March 21, 1996.

[38] N. Roussopoulos, Incremental Computation Models for AI-Database Systems October 1986.

[39] R. Sadeghi, W. Samson, S. Deen. HQL - Historical Query Language; *6th British National Conference on Databases*. July 1996.

[40] S. Schwiderski and G. Saake. Expressing temporal behaviour with extended ECA rules; *12th British National Conference on Databases*, July 1994.

[41] R. Snodgrass. The Temporal Query Language TQuel, *ACM Transactions on Database Systems*. 12(2), 1987.

[42] Softbase Systems, Inc. http://www.softbase.com.

[43] K. Stathatos. Physical Organization and Indexing of Time-Series Collections, October, 1995.

[44] G. Stephanopoulos, J. Cheung. Representation of Process Trends - Part I A Formal Representation Framework, *Computers Chemical Engineering*, Pergamon Press, 14(4/5): 495-510, 1990.

[45] G. Stephanopoulos, J. Cheung. Representation of Process Trends - Part II The Problem of Scale and Qualitative Scaling, *Computers Chemical Engineering*, Pergamon Press, 14(4/5):511-539, 1990.

[46] G. Stephanopoulos, B. Bakshi. Representation of Process Trends - Part IV Induction of Real-Time Patterns from Operating Data for Diagnosis and Supervisory Control, *Comupters Chemical Engineering*. 18(4):303-332, 1994.

[47] G. Stephanopoulos, B. Bakshi. Representation of Process Trends - Part III Multiscale Extraction of Trends From Process. *Comupters Chemical Engineering*. 18(4):267-302, 1994.

[48] S. Soukeras, P. King. Temporal databases: an event-oriented approach; *12th British National Conference on Databases*. July 1994.

[49] System Management Arts. InCharge Systems Product Information Sheet. Accessed from corporate web site, http://www.smarts.com.

[50] System Management Arts. Root Cause Analysis and its Role in Event Management. Accessed from corporate web site, http://www.smarts.com.

[51] Abdullah Uz Tansel, James Clifford, Shashi Gadia, Sushil Jajodia, Arie Segev, Richard Snodgrass. *Temporal Databases, Theory, Design, and Implementation*. The Benjamin/Cummings Publishing Company, Inc. 1993.

[52] Abdullah Uz Tansel, Erkan Tin. The Expressive Power of Temporal Relational Query Languages, *IEEE Transactions on Knowledge and Data Engineering*, 9(1), January/February 1997.

[53] Abdullah Uz Tansel, M. Erol Arkun, Gultekin Ozsoyoglu. Time-by-Example Query Language for Historical Databases, *IEEE Transactions on Software Engineering*, 15(4), April 1989.

[54] Abdullah Uz Tansel. Temporal Relational Data Model, *IEEE Transactions on Knowledge and Data Engineering*, 9(3), May/June 1997.

[55] Paolo Terenziani. Integrating Calendar Dates and Qualitative Temporal Constraints in the Treatment of Periodic Events, *IEEE Transactions on Knowledge and Data Engineering*. 9(5), September/October 1997.

[56] J. Valluri. Database Models and Architectures for Hybrid Network Management. MS Thesis 96-2. Institute for Systems Research. University of Maryland. 1996.

[57] V. Venkatasubramanian, R. Vaidyanathan, Y. Yamamoto. Process Fault Detection and Diagnosis Using Neural Networks -- I. Steady-State Processes. *Computers and Chemical Engineering*, 14(7):699-712, 1990.

[58] P. Viswanathan. Automated Network Fault Management. MS Thesis 96-14. Institute for Systems Research. University of Maryland. 1996.