DYNAMIC ROUTING OF TELEPHONE TRAFFIC USING NETWORK MANAGEMENT TOOLS

Thesis by
Barry E. Ambrose

In Partial Fulfillment of the Requirements
for the Degree of
Doctor of Philosophy

California Institute of Technology
Pasadena, California

1996

(Defended May 31, 1995)

Publication History:

- Thesis approved by defense committee (defense May 31, 1995):

  Dr. Robert McEliece, EE
  Dr. Michelle Effros, EE
  Dr. Andrea Goldsmith, EE
  Dr. Rodney Goodman, EE (Chair)
  Dr. Padhraic Smyth, JPL

- Final thesis delivered to Graduate Office October 31, 1995

- First printing, October '95 - 15 copies

I would never have written this thesis without the support and encouragement of my loving wife, Mary.

Acknowledgments

First, I would like to thank my advisor, Dr. Goodman, for giving excellent research direction and setting rigorous standards for these studies throughout the last 5 years. All of the members of the Caltech MicroSystems research group have provided valuable feedback for the Time Series Prediction study in this thesis. Dr. K. R. Krishnan has been helpful in sending me details and answering my questions about the DR5 Dynamic Routing algorithm. Finally Pacific Bell has provided all of the traffic data in this thesis for the Time Series Prediction study, and I am very appreciative.

Preface

We live in an age in which computers and computer software are becoming more and more an integral part of the society that we live in. Artificial Intelligence (AI) has made dramatic progress over the past thirty years, and there is plenty of scope for further research. Examples of progress are the development of theories of inference for expert systems and a better understanding of the capabilities and limitations of neural networks. In the years to come, I believe we will find closer and closer ties between AI fields and the fields of statistical modeling and inference. The benefits of robust automated learning techniques that may result from this synergy are enormous.

Abstract

In this thesis, the consequences of automating network management of telephone networks are examined. The role of network managers is to monitor the network for exceptional conditions and place controls into the network if necessary to deal with these network exceptions. One potential consequence of automating network management is a network which is capable of adjusting itself quickly to changing traffic conditions, also known as a network with dynamic routing. Simulations are used to show that there are benefits to be gained from implementing dynamic routing by automating the actions of the network managers.

In this thesis, the application of learning techniques such as neural networks and linear predictors to the tasks of network traffic management is also examined. Three network management tasks considered are: (i) recognition of traffic patterns in the network (ii) learning suitable thresholds for network congestion control and (iii) time series prediction of trunk group occupancy. It is found that non-linear learning techniques such as neural networks can give small gains over the more standard technique of linear predictors.

Table of Contents

List of Figures

List of Tables

Chapter 1
Introduction

1.1 Purpose

For some years, Caltech and Pacific Bell have cooperated to develop an expert system to aid network managers in their job of continually monitoring and controlling the telephone network. Some interesting opportunities for research have resulted from this work. The results are described in this thesis.

The purpose in writing this thesis was twofold:

- To quantify the benefits of using tools that were intended for network management to improve routing of calls from origin to destination in typical Regional Bell telephone networks. An analysis was also carried out of whether such an approach is always stable.

- To demonstrate the applicability of learning systems known as neural networks to problems that arise in day to day management of telephone networks.

Both goals have been successfully achieved.

1.2 Roadmap

This thesis is composed of a number of separate traffic studies grouped into two parts. These parts are:

- Benefits of dynamic routing

- Application of learning techniques to network management

Dynamic routing is a means of routing telephone traffic from its origin to its destination using automatic routing adjustments to handle changing network conditions.

Network management is presently very much a manual activity consisting of monitoring the telephone network to diagnose the cause of any network failures or exceptions and placing controls in the network to reduce the impact of these failures or exceptions.

Learning Techniques refers to the use of neural networks, linear predictors, or other techniques for automatically detecting patterns in data and extrapolating based on those patterns.

The impetus for *all* of these studies was a project called NOAA, Network Operations Analyzer and Assistant, which was a joint project between Pacific Bell and Caltech to develop a system to automate network management. NOAA is described in Appendix A.

1.3 Traffic Studies

To best understand the concepts and terminology used in this thesis, a review of traffic theory may be in order. [Sch87] or [Coo81] give more details for the interested reader.

1.3.1 Trunks and Trunk Groups

A telephone network can be modeled as a set of nodes and links. See Figure 1.1. Telephone traffic is carried on trunk groups. A trunk group is a set of trunks, each of which can support a conversation. Trunk groups are used to link the offices. If all the trunks are busy on a trunk group, then no new calls can be accepted on that trunk group. Typically a switch makes a number of attempts to connect the call, but if all routes available to it are busy, then the call is blocked.

The nodes are the telephone offices which act as switches for the telephone traffic. These offices route the traffic onto the appropriate trunk groups depending on the number dialed by the customer.

1.3.2 Random Arrivals

For network planning and modeling purposes, the usual assumption is that of random arrivals[Sch87]. In other words, the probability of a call arriving in the next small time interval is independent of call arrivals in the previous time intervals. This is also known as memoryless arrivals.

This is an approximation because, if a call is blocked, the customer will usually try again immediately. Thus, if a trunk group is fully occupied, the arrival rate will increase. This phenomenon is usually not included in the model.

Let λ denote the average arrival rate of new calls with units of calls per second. One exception to the assumption of random arrivals is the special case of overflow traffic which

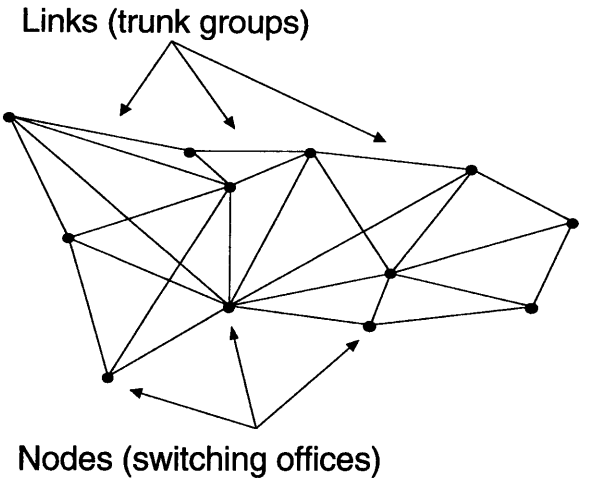

Figure 1.1: Telephone Network

is considered in Section 1.3.9.

1.3.3 Memoryless Holding Times

An assumption that is observed in practice is the assumption of memoryless holding times[Coo81, Sch87]. In other words, the probability of a call completing in the next small time interval is independent of how long the call has been in progress.

This leads to an exponential distribution of holding times. If T is a random variable denoting the observed holding time of a call with units of seconds, then:

$$p(T < T_1) = 1/T_h \int_{u=0}^{T_1} \exp(-u/T_h) du \tag{1.1}$$

From this the average holding time is obtained as T_h seconds and the average rate at which a call is serviced is defined as

$$\mu = 1/T_h \tag{1.2}$$

with units of calls per second. For telephone traffic, T_h is usually on the order of 180 seconds [DS84].

1.3.4 Traffic

The traffic is defined to be:

$$a = \frac{\lambda}{\mu} \tag{1.3}$$

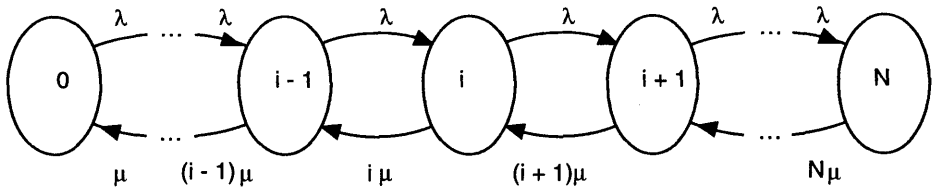

Figure 1.2: State Diagram

and the unit is taken to be Erlangs. The traffic is a measure of the relative rate at which calls are arriving in comparison to the rate at which they are being serviced. In general the number of trunks will need to be larger than the offered traffic if the blocking probability, that is, the probability that a new call arriving to the network is blocked, is to be kept low.

1.3.5 State Diagrams

Given N trunks, an arrival rate of λ calls per second, and a departure rate of μ calls per second, the state of the trunk group is defined as the number of trunks that are occupied on that trunk group. A state diagram which shows all the possible states of the trunk group and the rate at which those states are being entered and exited can be drawn. Here the circle with i in it, indicates the state of i trunks being occupied. This is illustrated in Figure 1.2.

1.3.6 State Probabilities

From the state diagram and its implied differential equations, the state probabilities in steady state can be found. Define $p_i(t)$ as the probability of i trunks occupied at time t. Consider arrivals and departures during a small time interval dt. This gives a set of $N+1$ equations of the form:

$$\frac{dp_i(t)}{dt} = -\lambda p_i(t) + \lambda p_{i-1}(t) - i\mu p_i(t) + (i+1)\mu p_{i+1}(t) \qquad 1 \leq i < N \tag{1.4}$$

$$\frac{dp_i(t)}{dt} = \lambda p_{i-1}(t) - i\mu p_i(t) \qquad i = N \tag{1.5}$$

$$\frac{dp_i(t)}{dt} = -\lambda p_i(t) + (i+1)\mu p_{i+1}(t) \qquad i = 0 \tag{1.6}$$

The total probability must always sum to 1:

$$\sum_{i=0}^{N} p_i(t) = 1 \tag{1.7}$$

Thus from any initial state, the final state can be deduced by setting:

$$\frac{dp_i(t)}{dt} = 0 \qquad \forall i \tag{1.8}$$

which gives:

$$p_i = \frac{a^i/i!}{\sum_{j=0}^{N} a^j/j!} \tag{1.9}$$

1.3.7 Erlang Traffic Tables

From the above the blocking probability, $B(a, N)$, which is the probability that all trunks are occupied when a new call arrives, can be seen as:

$$B(a, N) = \frac{a^N/N!}{\sum_{j=0}^{N} a^j/j!} \tag{1.10}$$

Equation 1.10 is known as the Erlang-B blocking equation [Coo81]. From this equation, the number of trunks needed to keep the blocking probability low can be calculated. An example of such a calculation assuming a 1% blocking probability is given in Table 1.1.

1.3.8 Poisson Distribution

If the number of trunks is large ($N \to \infty$), then Equation 1.9 becomes:

$$p_i = e^{-a}\frac{a^i}{i!} \tag{1.11}$$

which is a Poisson distribution. Recall that a Poisson distribution with mean a has variance equal to a.

1.3.9 First Routed and Overflow Traffic

The above has assumed random arrivals, also known as first routed traffic. When a call arrives to a telephone network, it is first attempted on a direct path to its destination if one exists. If this attempt fails it is then tried on an alternate path.

In general, overflow traffic (traffic that has overflowed from a direct path), will not arrive at random instants of time. In other words, a previous arrival on a trunk group carrying overflow traffic increases the likelihood of a future arrival.

For overflow traffic, the following equations [Coo81] apply. These equations are derived using a more complicated state diagram which models the number of trunks occupied

Number of Trunks	Maximum Traffic (Erlangs)
0	0.00
1	0.01
2	0.15
3	0.45
4	0.86
5	1.36
6	1.90
7	2.50
8	3.12
9	3.78
10	4.46
11	5.15
12	5.87
13	6.60
14	7.35
15	8.10
16	8.87
17	9.65
18	10.43
25	16.12
50	37.90
100	84.06

Table 1.1: Maximum Traffic Assuming 1% Blocking

on the direct route and the number of trunks occupied on the overflow route in a two dimensional state diagram. The state probabilities can then be solved for.

Define α as the mean level of the overflow traffic that results from random traffic of a Erlangs being offered to a trunk group of size N. Then it can be shown:

$$\alpha = aB(a, N) \tag{1.12}$$

Note the use of the Blocking formula, Equation 1.10. Define v as the variance of the overflow traffic. Then it can be shown:

$$v = \alpha\left(1 - \alpha + \frac{a}{N + 1 + \alpha - a}\right) \tag{1.13}$$

The derivation of the formula for the variance is not trivial. See [Coo81] for details. Define the peakedness z as the ratio of the overflow variance v to the mean overflow traffic level α:

$$z = v/\alpha \tag{1.14}$$

For random arrivals, the value of z would be one. For overflow traffic, z is greater than one. This is bad from the point of view of blocking because it means that more trunks must be provided to cope with a given level of traffic, because of its greater variability. The worst case peakedness would occur when a small amount of traffic overflows from a single large trunk group. In such a case, call arrivals would be highly clustered.

1.3.10 Numerical Example

For example, take the case of 210 Erlangs of offered traffic on 200 trunks:

$$N = 200, a = 210.0. \tag{1.15}$$

The calculations give 18.007 Erlangs of overflow traffic:

$$\alpha = 18.007, v = 113.85, z = 6.307. \tag{1.16}$$

Using a table of 1% blocking tells us that 231 trunks carry 209.8 $\approx$ 210 Erlangs. Thus 31 trunks can carry the 18.007 Erlangs of overflow traffic with 1% blocking.

On the other hand, 27.2 trunks would be necessary to carry the 18.007 Erlangs of traffic if the 18.007 Erlangs represented "ordinary" traffic. This demonstrates that overflow traffic requires slightly more trunks than ordinary traffic, because it has higher variability.

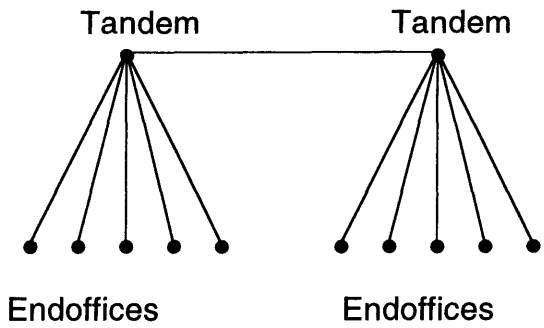

Figure 1.3: Hierarchical Telephone Network Routing

1.4 Routing Arrangements in Telephone Networks

Routing arrangements in telephone networks usually follow either a hierarchical or non-hierarchical arrangement. For example, presently Pacific Bell uses hierarchical routing and AT&T uses non-hierarchical routing.

1.4.1 Hierarchical Routing

In a hierarchical routing network, customers are connected to endoffices which in turn are connected to tandems. See Figure 1.3.

Endoffices are the exchanges that serve customers and *tandems* are the exchanges used for traffic between endoffices that are not directly connected.

In a hierarchical network, *high usage* trunk groups link offices that are close together geographically. They are designed to overflow to *final* trunk groups, which are the backbone of the network. This arrangement minimizes the total network cost.

1.4.2 Non-Hierarchical Routing

In a non-hierarchical network, each endoffice can also act as a tandem. See Figure 1.4.

In a non-hierarchical network, a distinction is made between the *direct* path between two endoffices and an *alternate* path. A direct path is a direct route between two endoffices. An alternate path connects two endoffices via an intermediate endoffice.

1.5 Historical Perspective on Teletraffic Engineering

In the past 100 years, teletraffic engineering has become a recognized branch of engineering. It was in 1917 that the Danish mathematician A. K. Erlang first published the blocking

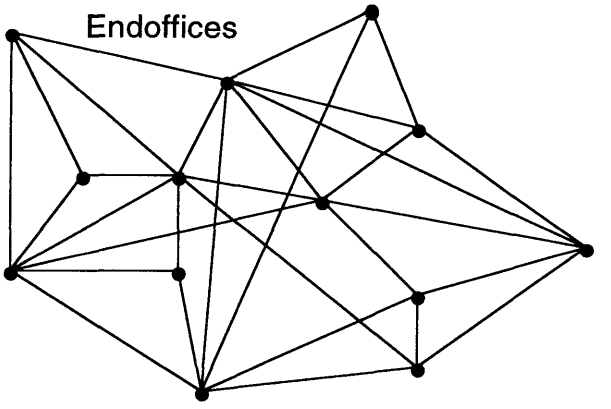

Figure 1.4: Non-Hierarchical Telephone Network Routing

formula, Equation 1.10, that bears his name [Coo81]. This blocking formula enables calculating the correct size for any system with a fixed number of servers, random arrivals, and memoryless holding times.

Developments since then have been made in network designs that use alternate routing. An understanding of the nature of the overflow traffic distribution was obtained in the 1930's. The seminal paper on telephone network design is a 92 page paper by Wilkinson in the 1956 Bell System Technical Journal [Wil56]. This paper needs surprisingly few changes today, when applied to the planning of regional Bell telephone networks.

The most recent changes in telecommunications networks have been the addition of a separate signaling network to handle message packets for call setup and routing, and a move from analog voice transmission to digital with the consequent improvement in signal quality over long distances. These changes have taken place gradually due to the necessity to avoid costly write-offs of the installed base of network equipment.

Over the coming years, Asynchronous Transfer Mode (ATM) technology can be expected to move from the laboratory to the telephone network. ATM is designed to carry video as well as voice traffic. This will result in big changes to the way networks are designed and built. Exciting opportunities exist for the telephone companies, especially those that succeed in finding the winning applications for this new technology.

Part I

Benefits of Dynamic Routing

Chapter 2
Introduction to Dynamic Routing

2.1 Introduction

In telephone networks, not enough equipment is provided to enable every customer to establish a simultaneous connection to every other network customer. Instead a statistical approach is taken in the choice of equipment capacities in telephone networks. Traffic measurements are taken and enough equipment is provided in the trunk network to satisfy the expected number of conversations in progress at the busiest hour of the day. Since the network planners are conservative in the amount of equipment installed, it is the task of routing algorithms to ensure that the networks are used efficiently.

Static routing algorithms are used to calculate routing tables for the network switches. These routing tables are not adjusted to account for traffic conditions in the network. Dynamic routing algorithms make adjustments to the call routing based on the state of the network or based on the time of day.

2.2 Dynamic Routing

A simple example of a small network is given in Figure 2.1. This shows a three node network, with nodes labeled A, B, and C. Each pair of nodes has a set of bi-directional links between them, capable of 10 simultaneous conversations. If the traffic requirements are such that each node has five calls destined toward each other node, there are at least two possible stable states for this network depending on the routing algorithms used.

One stable state would be the *direct routing* approach. In this case a call between A and B would only be allowed to use the direct link from A to B. If this were the case, all the offered calls could be accepted and the network would be capable of supporting the 30 simultaneous conversations.

Another alternative would be the *two hop routing* approach. In this case a call between A and B would be allowed to use the direct link from A to B or alternatively to use a two-hop route from A to C and then C to B. If this were the case, a stable state could arise with all calls in the network taking two hops and only half the traffic, i.e. 15 simultaneous conversations, being accepted.

From this example, it would seem that direct routing is always preferable. This is true when the traffic offered is symmetrical in nature. On the other hand, if a lot of traffic is offered between A and B with no traffic originating from B or C, it can be seen that using direct routing increases the probability of calls from A to B being blocked, since two-hop call completion possibilities are eliminated.

From this simple example it can be seen that a static or dynamic routing algorithm must strike a balance between increasing the global network efficiency by carrying out direct routing whenever possible, and increasing the network resilience to failure or unexpected traffic patterns by offering as many routing possibilities as possible.

The first major implementation of dynamic routing in the North American telephone network came with AT&T's implementation of Dynamic Non-hierarchical Routing (DNHR) which was introduced into the toll network in 1984, with projected network savings of 15% [HSS87]. DNHR is a routing scheme that updates the routing tables depending on time of day. In this way the difference in the busy hours in the different time zones across the USA can be taken advantage of. DNHR was subsequently replaced by RTNR which is described in this chapter.

The remainder of this chapter includes descriptions of:

- RTNR or Real-Time Network Rerouting from AT&T
- DR5 or Dynamic Rerouting Based on 5-Minute Data from Bellcore
- DCR or Dynamically Controlled Rerouting from Prism Systems
- NOAA or Network Operations Analyzer and Assistant developed at Caltech

2.3 Description of RTNR

AT&T has implemented a Real-Time Network Rerouting System (RTNR) in its network. The introduction of RTNR into the AT&T switched network was completed in July 1991 [ACF92], replacing the older DNHR system. The routing tables in the switches are updated in real-time using the signaling network. See Figure 2.2. Thus, at call setup time, the switch can send the call on the route that offers the best probability of completion.

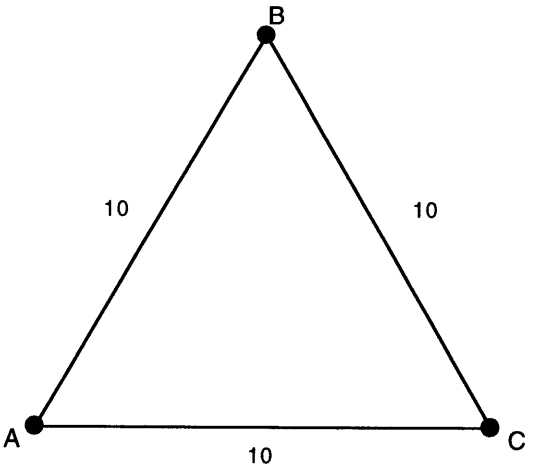

Figure 2.1: Simple Three Node Network

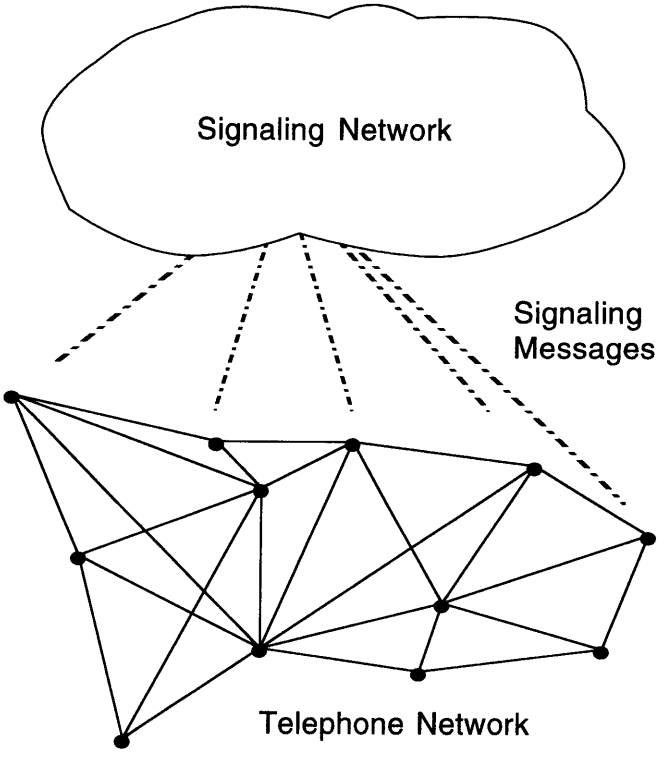

Figure 2.2: Signaling Network

2.3.1 AT&T Network

The AT&T network consists of approximately 100 switches and associated transmission equipment for the switching and transmission of calls. The network switches are distributed across the United States, which means that there should be potential savings by taking advantage of the different time zones in the different states. Another goal of a dynamic routing system is to provide a self-healing network that adjusts to take account of any equipment that may have come into or gone out of service.

2.3.2 Signaling System No. 7

The signaling system is vital to the implementation of RTNR. Signaling System No. 7 is an international standard and allows the transmission of call setup information between switches as the customer dials the number. Rather than transmit the signaling information along the same voice channels that the calls will use, a separate signaling network is used. See Figure 2.2. The messages sent in this network are the subject of the international standard. AT&T has extended the signaling message set to include information about routing possibilities that can be used at call setup.

The routing scheme is a two-hop scheme, which means that a maximum of two hops will be used to set up a call. The originating switch first tries a direct route to the terminating switch. If the direct route is not available, the originating switch tries to find a two-link path by first querying the terminating switch through the signaling network for the busy-idle status of all trunks connected to the terminating switch. It then compares its own trunk group busy-idle status information and finds a list of two-hop possibilities that are lightly loaded on both legs. One of these is chosen at random.

Two points should be noted. The first is that a 16 byte message is sufficient to communicate the busy-idle status of all the trunks in the terminating exchange, assuming the network has at most 128 nodes. If there are at most 128 nodes, then each node will have routes to at most 127 other nodes. A single bit is used for each route busy-idle status. Busy status means that over 90% of the trunks on the route are occupied. The shortness of the message is a key requirement of this dynamic routing scheme, since millions of calls are set up over the course of a day, and the signaling network should not be overloaded.

The second point to note is that a cache scheme is used to speed up the call setup. When a request for trunk status is sent, instead of waiting for a response, a call setup

is attempted using the most recently received status information. If this fails, the switch will make another attempt when the new status information arrives. This speeds the call setup.

2.3.3 Benefits of RTNR

As can be seen from this brief description, RTNR provides a routing scheme that dynamically adjusts itself to network conditions. In other words, if a free route exists for a call, RTNR is very likely to find it. On the other hand, if certain routes are congested or unavailable, RTNR is likely to avoid them. In addition it simplifies the task of network planners, since optimum routing tables for the switches in the network no longer need to be calculated prior to adding new equipment, since the network will adjust itself accordingly.

AT&T network planners have used simulations to show that an RTNR network can carry the same traffic as a DNHR network, with 2-3% fewer facilities [AC93a]. Thus, RTNR results in lower equipment costs. In addition, network planning and administration have been simplified.

Measurable improvements in service quality have also been found since the introduction of RTNR. For example, the busiest day of the year is the Monday after Thanksgiving. In 1990, with DNHR 17,086 of 136 million calls were blocked on AT&T facilities. In 1991, with RTNR 228 of 157 million calls were blocked on AT&T facilities. This represents a substantial improvement. Some of this improvement could be attributed to changes in the number of trunks in the network but AT&T in [ACF92] imply that the implementation of RTNR was the main cause of the improvement. Other cases involving barely degraded quality of service despite transmission facility failures are also cited.

2.4 Description of DR5

DR5 is a system proposed by Bellcore to implement a limited form of dynamic routing in the local telephone network [CKP91, OK85, KO89, Kri89].

2.4.1 Regional Bell Telephone Networks

The main constraint in the local telephone network compared to the long distance network is that switches are supplied by more than one vendor in a given network. Thus, implementation of dynamic routing is more difficult, since cooperation between different

vendors to update routing tables in the switches is required. This is still an unsolved problem.

Instead DR5 takes the approach of using the existing infrastructure for network management and adapting it to provide a limited form of dynamic routing. Existing network management capability in the local telephone network allows network managers to obtain data about every trunk group in the network at 5 minute intervals. If network managers find a problem in the network, they can effect expansive or restrictive controls to reroute traffic or cut down on traffic entering the network. DR5 uses the same interface to network management systems as the network operators, computes reroutes based on trunk data, and implements these reroutes using network management controls. Using simulations, Bellcore has estimated this approach can provide 50% of the potential benefits of full dynamic routing [CKP91].

2.4.2 Limitations of DR5

DR5 has limitations because the network management system was not originally designed for supporting DR5. First, some routing issues need careful handling when implementing controls. Number translation is the step during call setup of translating the telephone number that is dialed by the customer to find a trunk route on which the call can be sent. Number translation issues may mean that two controls must be put in to effect a single reroute. The first control is a precautionary control that prevents routing loops. Occasionally reroutes may not be allowed due to reroutes already in place that could result in routing loops.

Second, the update time of 5 minutes means that not all of the gains of dynamic routing are realized compared to the DCR and RTNR systems.

2.5 Description of DCR

DCR or Dynamically Controlled Routing is a dynamic routing system that works well with Northern Telecom DMS switches. It is currently installed in the Canadian network. Like RTNR, it updates the routing tables in the switches at short time intervals to take account of the current state of the network. Unlike RTNR, there is a central network management location that carries out all of the computations with respect to the new routing tables for the switches. More recently DCR has been enhanced to provide the capability to prevent

traffic associated with a high volume call-in from impacting the network.

2.5.1 Expansive Controls

The heart of the DCR network is a central network processor. Each switch reports to the network processor every 10 seconds with respect to its trunking status, overflow traffic and CPU occupancy [Car89]. Using this information, the network processor updates the routing tables of the switches within 5 seconds. This cycle repeats every 15 seconds.

A call in the DCR network uses at most two hops. Some facility failures in the Canadian network were handled well by DCR. For example when a transmission link was lost on June 16, 1989 between 18:00 and 22:00, DCR enabled all calls to be completed without overflow. Other outage examples were also given.

2.5.2 Restrictive Controls

More recently DCR has been extended to handle restrictive controls in the event of high volume call-in conditions, where call-in traffic has a low probability of completing and should be cut off at source to prevent interference with regular traffic [LR91].

The DCR approach, which is also used by network managers, is one of call gapping. A call gap specifies that only a certain number of calls are to be let through to a given telephone number or range of numbers in a control interval. In the case of DCR, the control interval is of the order of 15 seconds. The number of calls to be let through is a function of the arrival rate, the holding time, and the number of lines serving the call destination. In addition, the switch processor occupancy of all the switches in the network provides feedback as to when further throttling is required. Removal of the implemented controls is only made when certain low thresholds are passed to provide hysteresis in the control process.

Simulations have shown the system to work well in a large network even in the event of update data not always being available, or holding time estimates being inaccurate. In each case, the system is able to update the call gaps to keep switch processors below 85% occupancy, as opposed to 100% occupancy when the algorithm is not implemented. The customer still receives all the calls that they can handle. On the other hand, alternative call gap algorithms have been shown to oscillate when update information is not available [LR91].

2.6 Description of NOAA Dynamic Routing Possibilities

NOAA is described in detail in Appendix A.

Presently NOAA monitors the network for calls that are overflowing from trunk groups. To move more towards dynamic routing, NOAA could monitor trunk groups for occupancy above 90% and put in expansive controls before the trunk groups overflow. The main reason this is not presently done is that there is a bottleneck in obtaining information from the Pacific Bell network management system, which means that NOAA is limited in the number of controls that it can implement and monitor in a 5 minute period.

In Chapter 3 the equipment savings that can be expected from implementing dynamic routing using a system such as NOAA are examined.

2.7 Comparison

In this section some comparisons are drawn between the various dynamic routing systems.

2.7.1 Local Versus Global Knowledge

RTNR is the main dynamic routing system which avoids the use of centralized knowledge to update the routing tables in the switches. This has the advantage of providing a quick response to any exception situations. Also the cost of providing a data path from all the switches to a central location for the transmission of occupancy and routing information is avoided. On the other hand it becomes more difficult to use a system such as this for network management purposes, especially for restrictive controls, since a global view is lacking.

NOAA, DCR and DR5 all carry out processing of controls or routing table updates at a central network management location.

2.7.2 Update Times

DR5 and NOAA use a 5 minute update cycle for controls and network data. The RTNR and DCR schemes have update cycles of seconds. In general, the faster the response, the more traffic can be saved in the event of exceptional situations arising in the network. However, the faster the response, the more data processing capability needs to be present in the network management center.

2.8 Analysis of Dynamic Routing

Most dynamic routing schemes are analyzed using simulation techniques to find the equipment savings associated with implementing the scheme.

Ash et al. in [ACF92] have shown using simulations that RTNR provides capital equipment savings over DNHR. Ash et al. in [AC93a, AC93b] have showed that network design and network capacity management in RTNR networks amounts to a linear programming problem for which known methods with certain heuristic adjustments perform well.

Mitra et al. in [MGH93] carry out an analysis of a generalized RTNR scheme which sends more than one bit to indicate whether a route is busy or not. A simplified model of a symmetric network in which an assumption was made that each link was independent of other links was used. This model is called a Fixed Point Model. Fixed Point Models are discussed in detail in Chapter 4. Trunk reservation in which a small number of trunks was reserved for direct routed traffic was part of the model. Trunk reservation is discussed in detail in Chapter 8. Simulations were carried out to verify the independence assumption for the Fixed Point Model. The results indicate that using two bits (four states) to signify the trunk state is close to optimal.

Chaudhary et al. in [CKP91] used simulations to show that DR5 shows improvements over static rerouting schemes. Koussoulas in [Kou93] presents a model which is the core of the analysis routine used in DR5. It assigns a cost to each routing decision based on the possible loss of traffic due to the inefficiency of two hops over a direct route. Simulations were used to verify an analytical model of a symmetric network. Again independence of link traffic was assumed. The results indicate that the cost scheme proposed produced close to minimum network blocking.

Langlois et al. in [LR91] have carried out simulations to show that the proposed restrictive controls in DCR work well.

Chapter 3
Simulation of Dynamic Routing

3.1 Introduction

In Chapter 2 the various options available for dynamic routing are described. In this chapter, simulations are carried out to evaluate the benefits of dynamic routing based on 5 minute network management data.

3.2 Simulation Setup

The simulations yield estimates of the equipment savings that may be obtained by providing extra routing choices to switches in the network using a system such as NOAA. The simulations are of symmetric networks.

Two routing schemes are considered:

- With *hierarchical* routing, each call is first tried on the direct route and then on a fixed alternate route.

- With *NOAA type* routing, each call is first tried on the direct route, then on a fixed alternate route, and then on a NOAA suggested route. The list of NOAA suggested routes is revised once every five minutes. The list of NOAA suggested routes is obtained by attempting to route traffic from the most heavily loaded direct routes to the most lightly loaded alternate routes. The algorithm is given in more detail in Section 3.4.2. The update time of five minutes is used because the present NOAA application receives new data concerning trunk group occupancy every five minutes. The analysis generalizes to arbitrary update times.

3.3 Calculation of Equipment Savings

For each data point on the following plots, three simulations are performed. The first simulation is of hierarchical routing. The next simulation is of hierarchical routing with one more trunk on each trunk group. This gives a figure for the increase in traffic carrying capacity caused by an increase in equipment. The final simulation is of NOAA type

rerouting. The increase in traffic carrying capacity when NOAA rerouting is used is then converted to an equivalent change in trunk equipment requirements in the network.

In symbols, let λ_1 be the maximum traffic that a hierarchical routing network can accept given C trunks on each trunk group, and 1% blocking, λ_2 the maximum traffic given $C+1$ trunks on each trunk group and 1% blocking, and λ_3 the maximum traffic given C trunks on each trunk group with NOAA rerouting and 1% blocking, then the equipment savings S that would result from implementing NOAA rerouting are given by:

$$S = \frac{1}{C}\left(\frac{\lambda_3 - \lambda_1}{\lambda_2 - \lambda_1}\right) \tag{3.1}$$

3.4 Routing Arrangements

In more detail, the routing arrangements are as follows. The network is a symmetric network (see Figure 8.1 for an example of a 5 node symmetric network) with N nodes and C uni-directional trunks between each pair of nodes. Both N and C are varied during the simulations.

3.4.1 Hierarchical Routing

For the hierarchical routing arrangement, a call from a node i to a node j is first attempted on the direct route from i to j. See Figure 3.1. If the call is blocked on the direct route, because all C trunks are occupied, it is then attempted on a fixed alternate route. The fixed alternate is chosen to be via an intermediate node, k, and requires a free trunk to be found between i and k and also k and j.

The value of k is chosen to equal $(i+1)|N$ where $|$ denotes the modulus operator, unless $(i+1)|N$ happens to equal the destination node j, in which case the intermediate node is set to $(i+2)|N$.

Random call arrivals at an average rate of λ calls per second are considered, where λ is varied during the simulation. Memoryless holding times (See Section 1.3.3) are considered, with an average holding time of 180 seconds.

3.4.2 NOAA Rerouting

For the NOAA rerouting, the call is first tried on the direct route and the fixed alternate route, and finally on a NOAA suggested alternate route. At five minutes intervals the

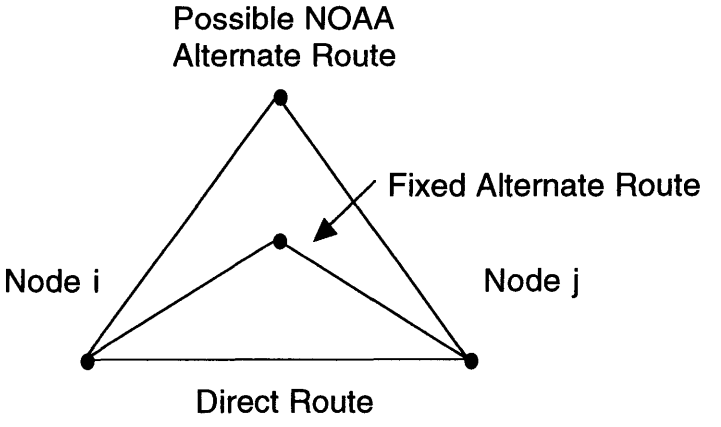

Figure 3.1: Direct and Alternate Routes in Symmetric Network Model

NOAA suggested alternate routes are revised. For each direct route, a search is made through the $(N-3)$ possible alternate routes, not including the fixed alternate route. The capacity of each of these alternates is noted. The alternate route with the most capacity is chosen as a NOAA reroute, provided it has more spare capacity than the fixed alternate. In the event of a tie between a number of equally good alternates, the tie is broken at random.

If there are i trunks occupied on leg number 1 of the alternate route and j trunks occupied on leg number 2 of the alternate route, then the capacity of that alternate is taken to be C - Max(i, j), corresponding to the minimum capacity on each of the two legs.

3.5 Results

The simulation results are shown in Figures 3.2 to 3.4. For comparison, the NOAA algorithm is rerun using 15 second updates of the routing table instead of 5 minute updates. This is done to find the limits as call by call update of the routing tables is approached. The results indicate that carrying out 15 second updates gives approximately double the equipment savings in all scenarios compared to carrying out 5 minute updates.

It could be asked whether 15 second updates is close to call by call update of the routing tables. Assuming 10 trunks between nodes, this would allow about 5 Erlangs to be carried with a small amount of blocking (see Table 1.1). This gives an arrival rate of $\lambda = \mu a = 5.0/180$ calls per second (see Equation 1.3). This is 36.0 seconds between new call arrivals on average. Thus the 15 second update of routing tables should be pretty

close to call by call update of the routing tables.

Figure 3.2 shows that for a 10 node fully interconnected network, the equipment savings for NOAA rerouting with 5 minute updates vary from about 5.8% to 2.3% as the number of trunks between nodes increases. In qualitative terms, this can be understood by noting that large trunk groups are more efficient, in terms of the amount of traffic that can be carried, than smaller trunk groups for a given level of blocking. This can also be seen by inspection of Table 1.1. Thus there is less spare capacity in the network that the NOAA rerouting algorithm can make use of.

Figure 3.3 shows that for a symmetric fully interconnected network with 10 trunks between each node, the equipment savings for NOAA rerouting with 5 minutes updates varies from about 5% to 6.8% as the number of nodes increases. In qualitative terms, this can be understood by noting that the more nodes in the network, the better the choice that the NOAA rerouting algorithm has in seeking a lightly loaded trunk group.

Although the first point in each of Figures 3.3 and 3.2 should agree, they differ due to the size of the interpolation step used to find the traffic for a given level of blocking. Figure 3.2 uses a smaller interpolation step and should be more accurate.

Figure 3.4 shows that for a symmetric network with 30 nodes and 10 trunks on each trunk group, the equipment savings for NOAA rerouting with 5 minute updates varies from about 2% to 6.8% as the connectivity increases from 23% to 100%. Connectivity measures the number of connections between network nodes compared to the total number of possible connections. In a 30 node network, the total number of possible connections would be $30 * 29 = 870$.

This indicates that in a real network, the equipment savings would be less than the results suggested by simulations of fully interconnected symmetric networks. For example, the connectivity in a real network may be close to 20%. The connectivity of metropolitan networks would be higher, perhaps as large as 50%.

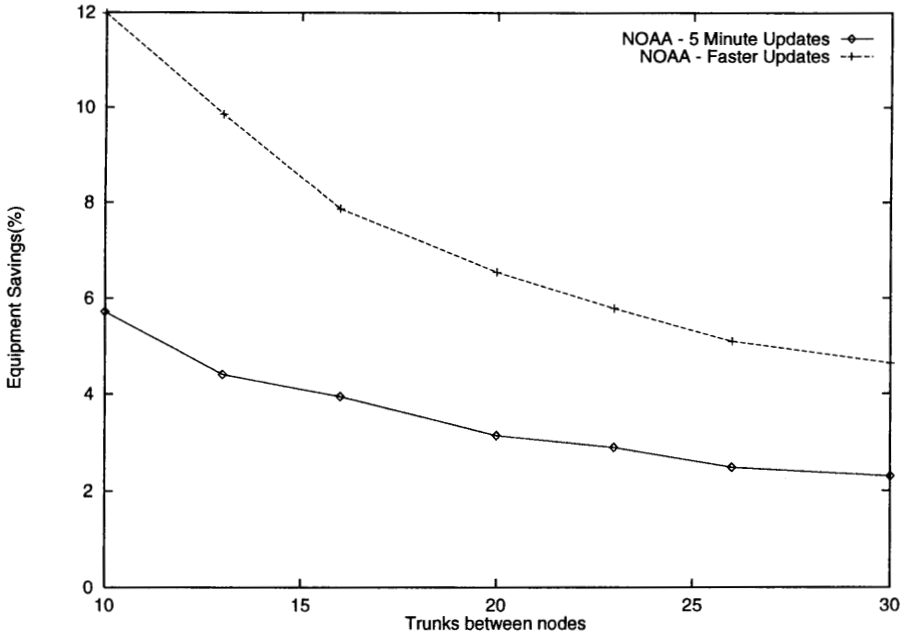

Figure 3.2: Savings as Number of Trunks Varies in a 10 Node Network

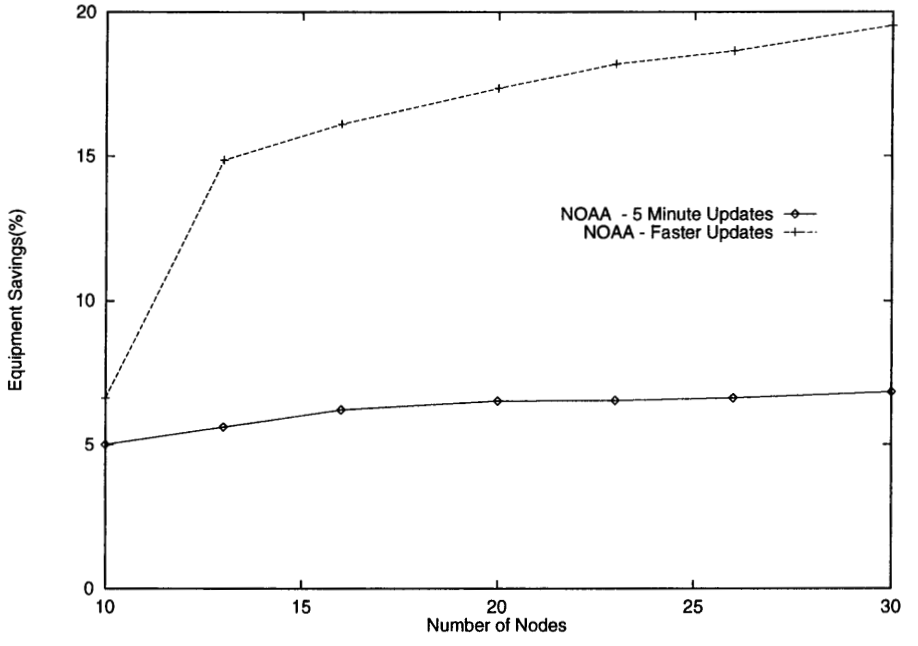

Figure 3.3: Savings as Number of Nodes Varies

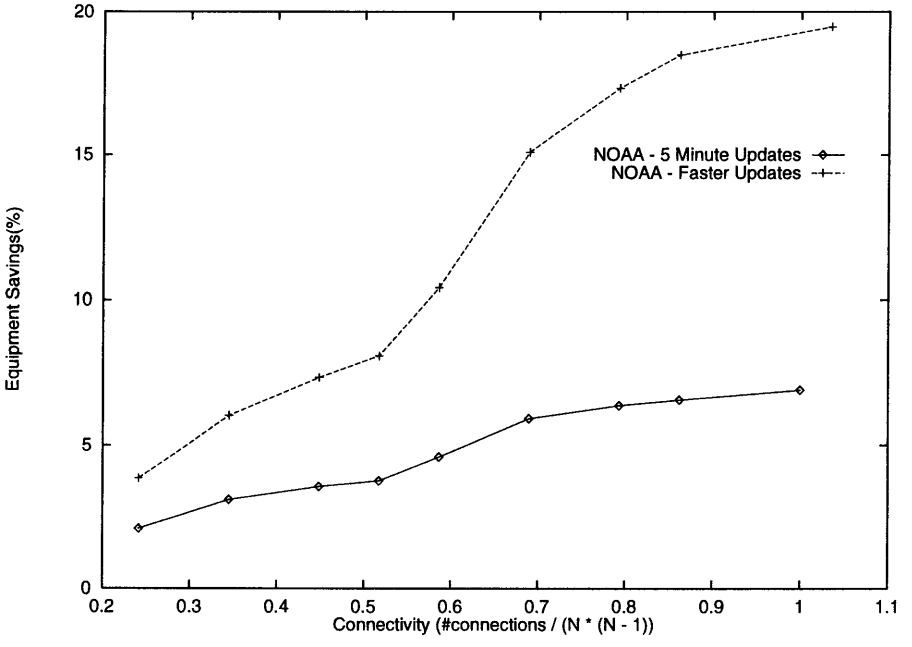

Figure 3.4: Savings as Connectivity Varies in a 30 Node Network

Chapter 4
Fixed Point Models for Dynamic Routing

4.1 Introduction

A Fixed Point Model (FPM) uses a knowledge of arrival rates and departure rates and routing rules to calculate the state probabilities corresponding to a given routing model. See Section 1.3.6 in Chapter 1 for what is meant by a set of state probabilities. An initial set of state probabilities is assumed. Iteration using the arrival rates and departure rates is carried out until there is little change in the state probabilities.

4.2 Simple Example

FPMs are usually used to model networks, but here a single trunk group is modeled for illustrative purposes.

Assume a trunk group with one trunk, whose state transition diagram is illustrated in Figure 4.1. This will have two states, busy or empty. Assuming the arrival rate is $\lambda = 2.0$ to both states and the departure rate is $\mu = 1.0$ from state 1. Calls arriving when the system is in state 1 (i.e. one trunk occupied) are blocked.

Define $p_i(n)$ as the probability of i trunks being occupied at time $n\delta t$. Assume the initial probabilities for the FPM are $p_0(0) = 0.5$ and $p_1(0) = 0.5$. These initial probabilities can be arbitrarily chosen as long as they sum to 1. Take $\delta t = 0.01$ time units. Then iterate for each time n using:

$$p_0(n) = p_0(n-1) - \lambda p_0(n-1)\delta t + \mu p_1(n-1)\delta t \tag{4.1}$$

$$p_1(n) = p_1(n-1) + \lambda p_0(n-1)\delta t - \mu p_1(n-1)\delta t \tag{4.2}$$

Within 400 iterations, this converges to

$$p_0 = 0.333334 \tag{4.3}$$

$$p_1 = 0.666666 \tag{4.4}$$

which compares well with the true answer for the equilibrium probability distribution of 1/3 and 2/3 respectively, obtained from equation 1.9. The blocking probability is equal to the probability that the trunk group is busy when a new call arrives, namely p_1.

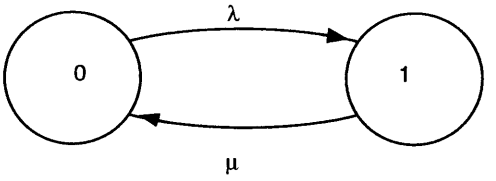

Figure 4.1: Simple State Transition Diagram

When an FPM is used to model a symmetric network, usually a single trunk group is modeled and it is assumed that this trunk group can represent any trunk group in the network [MGH93].

4.3 Benefits and Costs of the FPM Approach

For symmetric networks, it was observed that the FPM is faster than simulation. The FPM does involve some iteration but the run times were a lot less than the run times of simulations to get the same estimates.

Despite the fact that an FPM models a stochastic system, an FPM does not make use of a random number generator. Each run of an FPM model will give the same result, assuming sufficient iterations. Given sufficient iterations, the choice of the initial state probabilities have an exponentially small effect on the final outcome. Since an FPM model may make some approximations, it is difficult to know how much confidence to associate with the FPM result. The best way to characterize the accuracy of an FPM is to compare with simulation. The simulation can be designed to avoid the assumptions implicit in an FPM.

The main assumption for FPMs is independence of trunk groups. In other words the state of each trunk group is assumed to be independent of the state of the other trunk groups. This assumption implies that a call that uses two trunk groups in order to connect to its destination is modeled as two calls, each using one of the trunk groups. Clearly this is inaccurate.

The network blocking is a function of the blocking on the direct route and the blocking on the alternate route(s). The independence assumption implies that the blocking on the direct route is independent of the blocking on the alternate route. The results of this chapter will show that this condition is not met at low traffic levels.

4.4 State of the Art

Fixed Point Models have been used in recent years to analyze different routing algorithms in symmetric networks. In [MS91], D. Mitra et al. use fixed point models to evaluate two routing methods for telephone networks called Dynamic Alternate Routing (DAR) and Fixed Alternate Routing (FAR). The Fixed Alternate Routing used a fixed pre-computed routing table whereas the DAR algorithm varied its routing based on whether blocking was encountered or not. The study concluded that DAR was a feasible method of routing that did better than FAR following network failures.

In [MGH93], D. Mitra et al. use fixed point models to evaluate variants of AT&T's RTNR routing algorithm. The study concludes that it is possible to improve on RTNR by transmitting slightly more state information between nodes. This paper gives references to 8 other papers on the subject of Fixed Point Models for telephone network routing evaluation over the past twenty years.

In [ACL94], G. R. Ash et al. describe how fixed point models are use in the dimensioning of the AT&T network, to calculate the trunk group sizes given a set of traffic demands, desired blocking probabilities and the RTNR routing rules. The planning algorithm is called Fully Shared Network (FSN) design.

4.5 Starting Point

The starting point for the analysis of FPMs in this chapter is a paper by Akinpelu [Aki84] on overload performance of non-hierarchical routing networks. The purpose of the Akinpelu paper is to show that networks which use DNHR can have two stable states under overload conditions. In an appendix to the paper, an FPM is given for networks with a fixed but arbitrary number of choices for alternate routes.

4.5.1 FPM of Akinpelu

It is assumed that:

1. The mixture of traffic offered to a trunk group has random arrivals.

2. Trunk group blocking probabilities are independent.

The term *path* is defined to be a set of distinct trunk groups that forms a connection between two nodes. The term *route* is defined to be an ordered collection of paths connecting the same point-to-point pair.

Let L^j be the expected offered load for point-to-point pair j. Let $L = \sum_j L^j$ be the total offered load. Let p_i, n_i, and a_i denote the blocking probability, trunk group size, and offered load for trunk group i. Let $q_i = 1 - p_i$. A path is denoted by r, a route by R, the route for point-to-point pair j by R^j and the route formed by the first k paths of R^j by

$$R_k^j = (r_1^j, \ldots, r_k^j) \tag{4.5}$$

Finally $D(R)$ is the probability that route R is blocked.

Define c_k^j as the expected carried load for path k for point-to-point pair j. Then

$$c_k^j = L^j[D(R_{k-1}^j) - D(R_k^j)] \tag{4.6}$$

Define K_i as the expected total carried load for trunk group i. Then

$$K_i = \sum_{j,k,i \in r_k^j} c_k^j \tag{4.7}$$

Also

$$K_i = a_i q_i \tag{4.8}$$

from which a_i can be found. To find p_i use:

$$p_i = B(n_i, a_i) \tag{4.9}$$

where $B(n, a)$ is the Erlang B blocking formula which gives the probability of blocking assuming n trunks available, traffic of a, and memoryless arrivals. Define z as the expected network blocking. Then

$$z = \frac{\sum_j L^j D(R^j)}{L} \tag{4.10}$$

Define C as the expected network carried load. Then

$$C = L(1 - z) \tag{4.11}$$

For iteration purposes, the blocking needs to be recalculated. Using the independence assumptions gives:

$$D(R) = \prod_{t=1}^{k} \left(1 - \prod_{i \in r_t} q_i \right) \tag{4.12}$$

assuming the paths of R are disjoint.

4.6 Simple FPM for Hierarchical Routing

The first FPM derived in this chapter is a simple FPM for Hierarchical Routing. The aim is to compare the blocking for an FPM that models NOAA rerouting with the blocking for an FPM that models hierarchical rerouting. The FPM for NOAA rerouting will be given later in Section 4.8.

The routing scheme is meant to model the hierarchical routing typically present in a regional Bell telephone network. The network is symmetric. A call is offered first to a direct route. If this routing attempt fails the call is offered to a fixed alternate route. The alternate route is a two hop route.

4.6.1 FPM

For the fixed point model, define N as the number of nodes, C as the number of trunks on each route, δt as the timestep of the iteration, $p(i)$ as the probability of i trunks being occupied on a route, p_b as the blocking probability for a call offered to a route, λ as the arrival rate of calls, and μ as the service rate of calls. Without loss of generality, μ is taken to be 1. In other words, the time unit is taken to be the average holding time of a call. Initially the occupancy probabilities $p(i)$ can be arbitrary provided they sum to 1.

The probability of a call being blocked on a route is simply the probability of C trunks being occupied on a route.

$$p_b = p(i = C) \tag{4.13}$$

Define λ_T as the total arrival rate of calls on a route. Then

$$\lambda_T = \lambda + 2\lambda p_b(1 - p_b) \tag{4.14}$$

since the route carries direct routed traffic and overflow traffic. The direct routed traffic has an arrival rate λ. The overflow traffic has an arrival rate λp_b provided the other leg of the alternate route is not blocked. Each direct route carries the overflow traffic from two alternate routes.

The state probabilities can be updated as follows:

$$\delta p(i) = (\lambda_T p(i-1) - \lambda_T p(i) - ip(i) + (i+1)p(i+1))\delta t \qquad 0 < i < C \tag{4.15}$$

$$\delta p(i) = (-\lambda_T p(i) + (i+1)p(i+1))\delta t \qquad i = 0 \tag{4.16}$$

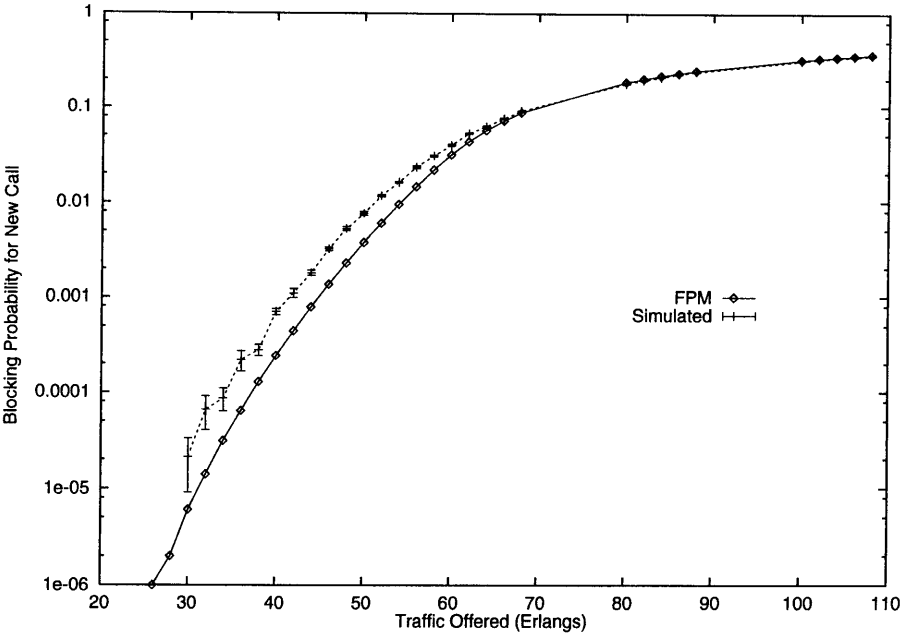

Figure 4.2: Blocking for FPM and Simulation

$$\delta p(i) = (\lambda_T p(i-1) - ip(i))\delta t \qquad i = C \tag{4.17}$$

The network blocking z can be calculated as

$$z = p_b(1 - (1 - p_b)^2) \tag{4.18}$$

corresponding to a call being blocked on both the direct route and on either leg of the alternate route.

A comparison of the FPM results and simulation results for the simple FPM for hierarchical routing is given in Figure 4.2. The simulation is a simulation of 5 hours of telephone traffic on a network with 10 nodes and 10 circuits between each node. Each call arrival and departure was simulated. 10 runs of the simulation were carried out to derive error bars for the estimates.

The lack of agreement for low traffic levels could be due to the assumption that blocking on the alternate route is independent of blocking on the direct route. This clearly is not true for low traffic levels. The more complex FPM in the next section will model this dependency.

4.6.2 Simulation using Latin Squares

When the simulation work for comparison with an FPM started, if a call was to be routed between nodes i and j, then node $(i+1)|N$ was chosen as the intermediate node for the alternate route, unless $(i+1)|N$ was equal to j, in which case the intermediate node was taken to be $(i+2)|N$. It was thought that since no node was treated specially, the agreement with the FPM model, which assumes symmetry, should be good.

However, as the modeling progressed, it became clear that routes of the form $(i,(i+1)|N)$ were more heavily loaded than the other routes as a consequence of this routing scheme. What was needed to restore symmetry was a *Latin Square* format for the routing table in the switches. The Latin Square is used to choose the via node for alternate routing from source to destination. A Latin Square is a table of integers in which the same integer appears only once in each row and column. Since each number appears only once, it can be used as a choice for the via node with the row representing the source and the column representing the destination. This arrangement will spread the traffic evenly over the network. See Figure 4.3.

The special property of this Latin Square is that the elements along the diagonal are equal to the row number. This constraint ensures that the chosen via node will never be equal to the source or the destination node. To construct the Latin Square, a perturbation method was used. The initial square was set to be

$$a_{ij} = (i-j)|C \qquad 0 \le i,j \le C-1, \quad i-j \quad \text{even} \tag{4.19}$$

$$a_{ij} = (i+j)|C \qquad 0 \le i,j \le C-1, \quad i-j \quad \text{odd} \tag{4.20}$$

which is a Latin Square without the special property that is required for the routing table in the switch, namely that the elements on the diagonal be equal to the row number. Then 3 types of random perturbation were carried out:

- row swaps
- column swaps
- element swaps

The element swap started by swapping two elements in some row. This violated the column constraint for the second element, requiring another element swap. This again

DESTINATION NODE \ SOURCE NODE	0	1	2	3	4	5	6	7	8	9
0	0	5	8	2	6	9	1	4	3	7
1	3	1	7	0	8	6	2	5	9	4
2	1	3	2	8	0	4	7	9	5	6
3	9	8	4	3	7	2	0	1	6	5
4	6	7	5	9	4	0	3	8	2	1
5	8	9	0	7	3	5	4	6	1	2
6	4	2	9	5	1	8	6	0	7	3
7	2	4	1	6	5	3	9	7	0	8
8	7	6	3	4	9	1	5	2	8	0
9	5	0	6	1	2	7	8	3	4	9

Figure 4.3: 10x10 Latin Square. Used to choose the Via Node for Alternate Routing from Source to Destination. Each entry appears once in each row and column assuring symmetry.

violated constraints requiring another swap. This set of swaps terminates when the column constraints for the first element in the original row was fixed. In switching theory, it is similar to the algorithm used to set up the crossbar elements in a rearrangeable switch. Without this element swap perturbation, no solution was found.

The stopping criterion was when all the diagonal elements were different from each other. Once such a square was found, a simple re-labeling could generate the type of square in Figure 4.3. The search time was not very long, under 30 seconds on a Sparc 10. It should be noted that the number of Latin Squares in the search space was over $(n!)^{2n}/n^{n^2} \approx 10^{30}$ for $n = 10$ [vLW92], so this type of search would have failed if there was a unique solution.

4.7 Improved FPM for Hierarchical Routing

It was noted in Section 4.6 that a reason for poor agreement between the FPM of a 10 node network and the simulation could be the violation of the independence assumption that blocking on the direct route is independent of blocking on the alternates. In this section, an improved FPM is described that models three routes, a typical direct route and the two legs of its alternate route.

For the fixed point model, define N as the number of nodes, C as the number of trunks on each route, δt as the timestep of the iteration, $p(i,j,k)$ as the probability of i trunks being occupied on a direct route, j being occupied on leg 1 of its alternate route, and k being occupied on leg 2 of its alternate route, p_b as the blocking probability for a call offered to a route, λ as the arrival rate of calls and μ as the service rate of calls. Without loss of generality μ is taken to be 1. In other words, the time unit is taken to be the average holding time of a call. Initially the occupancy probabilities $p(i,j,k)$ can be arbitrary provided they sum to 1.

The state probabilities can be updated as follows:

$$\delta p(i,j,k) = (c_1 + c_2 + c_3 + c_4 + c_5)\delta t \tag{4.21}$$

where $c_1, \ldots, c_5$ are contributory terms due to call arrival and departure events.

Considering departures gives

$$c_1 = (i{+}1)p(i{+}1,j,k){+}(j{+}1)p(i,j{+}1,k){+}(k{+}1)p(i,j,k{+}1){-}ip(i,j,k){-}jp(i,j,k){-}kp(i,j,k) \tag{4.22}$$

where $p(i,j,k) = 0$ for i, j or $k < 0$ and $p(i,j,k) = 0$ for i, j or $k > C$.

Considering external arrivals gives:

$$c_2 = -\lambda p(i,j,k){-}\lambda p(i,j,k){-}\lambda p(i,j,k){+}\lambda p(i{-}1,j,k){+}\lambda p(i,j{-}1,k){+}\lambda p(i,j,k{-}1) \tag{4.23}$$

Considering internal arrivals causing a transition *from* state (i,j,k) gives:

$$c_3 = -c_{3.1} - c_{3.2} - c_{3.3} - c_{3.4} \tag{4.24}$$

where

$$c_{3.1} = 2\lambda_i(i)p(i,j,k) \qquad i < C \tag{4.25}$$

$$c_{3.2} = \lambda_i(j)p(i,j,k) \qquad j < C \tag{4.26}$$

$$c_{3.3} = \lambda_i(k)p(i,j,k) \qquad k < C \tag{4.27}$$

$$c_{3.4} = \lambda p(i,j,k) \qquad i = C, j < C, k < C \tag{4.28}$$

where $\lambda_i(j)$, the internal overflow traffic to j, is the rate of arrivals to leg 1 of the alternate route from the direct route assuming j trunks presently occupied on leg 1 of the alternate route. This is calculated as:

$$\lambda_i(j) = \lambda \frac{\sum_{k=0}^{C-1} p(C,j,k)}{\sum_{i=0}^{C}\sum_{k=0}^{C} p(i,j,k)} \tag{4.29}$$

The numerator represents the probability of being in a state amenable to a call arrival. The denominator represents the probability of j trunks being occupied on leg 1 of the alternate route.

Considering internal arrivals causing a transition *into* state (i,j,k) gives:

$$c_4 = c_{4.1} + c_{4.2} + c_{4.3} + c_{4.4} \tag{4.30}$$

where

$$c_{4.1} = 2\lambda_i(i-1)p(i-1,j,k) \qquad i > 0 \tag{4.31}$$

$$c_{4.2} = \lambda_i(j-1)p(i,j-1,k) \qquad j > 0 \tag{4.32}$$

$$c_{4.3} = \lambda_i(k-1)p(i,j,k-1) \qquad k > 0 \tag{4.33}$$

$$c_{4.4} = 2\lambda_i(i)p(i,j-1,k-1) \qquad i = C, j > 0, k > 0 \tag{4.34}$$

Note that in this model, a simultaneous transition to state (i,j,k) is seen from state $(i,j-1,k-1)$ once a call is re-attempted on an alternate route, following its failure on the direct route.

Finally some departures leave j and k simultaneously:

$$c_5 = c_{5.1} - c_{5.2} \tag{4.35}$$

$$c_{5.1} = 0.5(j+1+k+1)p(i,j+1,k+1)f \qquad j < C, k < C \tag{4.36}$$

$$c_{5.2} = -0.5(j+k)p(i,j,k)f \tag{4.37}$$

where f is an estimate of the fraction of calls that were rerouted from direct route i instead of other direct routes. An approximate value for f based on the offered traffic is computed to be:

$$f = \frac{\overline{\lambda_i}}{2\overline{\lambda_i} + \lambda} \tag{4.38}$$

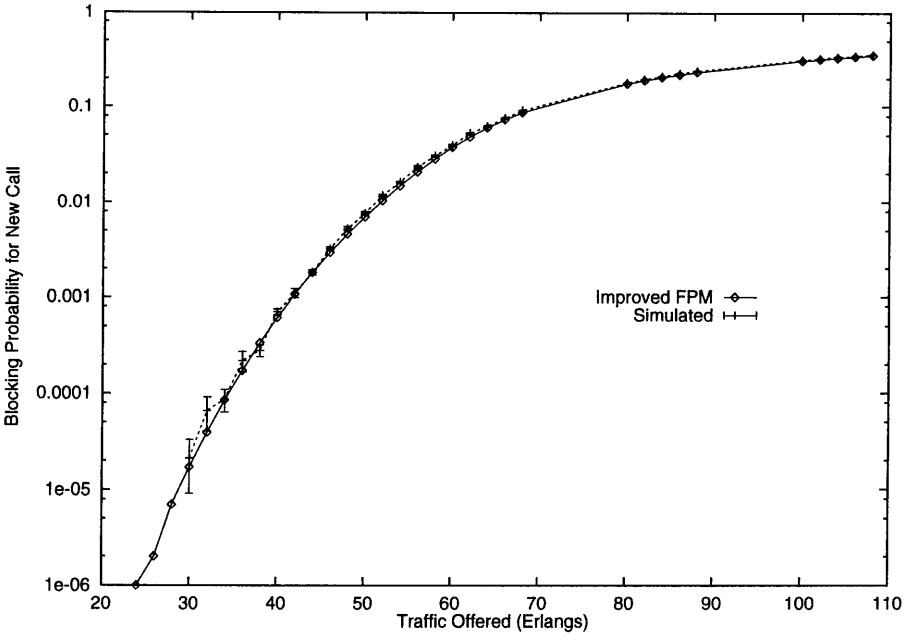

Figure 4.4: Blocking for Improved FPM and Simulation

Here

$$\overline{\lambda_i} = \lambda p_b(1.0 - p_b) \tag{4.39}$$

$$p_b = \sum_{j=0}^{C} \sum_{k=0}^{C} p(C, j, k) \tag{4.40}$$

which is the same set of formulas as used in the simple FPM model of hierarchical routing to yield the offered traffic from the direct route.

The network blocking z can be calculated as

$$z = \sum_{j=0}^{C} p(C, j, C) + \sum_{k=0}^{C} p(C, C, k) - p(C, C, C) \tag{4.41}$$

corresponding to a call being blocked on both the direct route and on one leg of the alternate route.

A comparison of the FPM results and simulation results for the simple FPM for hierarchical routing is given in Figure 4.4. The simulation is a simulation of 5 hours of telephone traffic on a network with 10 nodes and 10 circuits between each node. Each call arrival and departure was simulated. 10 runs of the simulation were carried out to derive error bars for the estimates. The agreement with the simulation is much improved at low traffic levels.

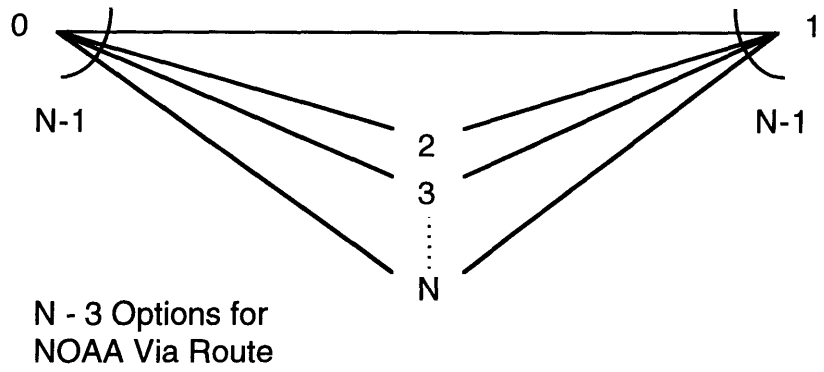

Figure 4.5: NOAA Via Route Options

4.8 FPM for NOAA

With NOAA rerouting, a call will first try the direct route, then a fixed alternate route and finally a NOAA-suggested alternate route. The table of NOAA suggested alternate routes is revised every 5 minutes. The NOAA suggested alternate route is the alternate route with the most available capacity. If there is a tie for the route with most available capacity, then one of the routes is chosen at random. If none of the potential alternates gives more available capacity than the fixed alternate route, no NOAA reroute is chosen. Let $p(i)$ be the probability of i trunks being occupied in a typical route in a symmetric network. A matrix $A = a_{ij}$ is defined, where a_{ij} is the probability of a route appearing i times as a NOAA alternate choice given j trunks occupied on the route at the start of the five minute period. A matrix $B = b_{jk}$ is defined, where b_{jk} is the probability of j trunks being occupied at the start of the 5 minute period given k trunks occupied now.

The initial values for $p(i)$ can be chosen arbitrarily provided the values sum to one. The B matrix is initially set to be the identity matrix.

4.8.1 NOAA Via Routes

Based solely on the $p(i)$ values, the A matrix can be calculated as follows. Recall that the A matrix gives the probability of a route being part of i NOAA alternates, given that j trunks are occupied at the start of the 5 minute period.

An intermediate value p_c is defined to be the probability of a route being chosen once as a NOAA alternate, given i_0 trunks occupied.

The $2(N-3)$ possible cases for which the route could be chosen as an alternate are illustrated in Figure 4.5. This shows $N-1$ routes from any given node. When the choice of a NOAA alternate is made, two of these routes are ruled out because they represent the direct route and the first alternate. The remaining routes may be chosen as NOAA alternates and carry extra traffic as a result.

4.8.2 Binomial Distribution

The $2(N-3)$ possible cases for which the route could be chosen as part of a NOAA alternate are taken to be independent Bernoulli trials. Then

$$a_{ij} = \binom{2(N-3)}{i} p_c^i (1-p_c)^{2(N-3)-i} \tag{4.42}$$

It remains to calculate p_c, the probability of a route being chosen as one NOAA alternate, given i_0 trunks occupied at the start of the 5 minute period.

4.8.3 Multinomial Distribution

Consider the situation where there are i_0 trunks occupied on leg 1 of a potential NOAA alternate and l trunks occupied on leg 2 of a potential NOAA alternate. The overall capacity on that route is denoted as:

$$M(i_0, l) = C - \text{Max}(i_0, l) \tag{4.43}$$

Consider the further condition that there are j and k trunks occupied on the two legs of the fixed alternate route, and that $M(j,k) > M(i_0,l)$.

Looking at the other $N-4$ possible alternates, the probability that a given alternative has more capacity than $M(i_0,l)$ is denoted as p_M, equal capacity as p_E, and less capacity as p_L. In this situation, assuming route independence, there is a multinomial distribution:

$$(p_M + p_E + p_L)^{N-4} \tag{4.44}$$

These probabilities are given by:

$$p_M = \sum_{l'} \sum_{m'} p(l')p(m') \qquad M(l',m') > M(i_0,l) \tag{4.45}$$

$$p_E = \sum_{l'} \sum_{m'} p(l')p(m') \qquad M(l',m') = M(i_0,l) \tag{4.46}$$

$$p_L = \sum_{l'} \sum_{m'} p(l')p(m') \qquad M(l', m') < M(i_0, l) \tag{4.47}$$

The probabilities of cases where there are a paths with equal capacity, $N - 4 - a$ paths with less capacity, and 0 paths with more capacity can be summed. This gives the following formula:

$$p'(i_0, j, k, l, a) = \begin{pmatrix} N-4 \\ a \end{pmatrix} p_E^a p_L^{N-4-a} \tag{4.48}$$

using the formula for the multinomial distribution [Fel50].

A route is chosen as a NOAA alternate if it is a clear winner in the set of choices for a NOAA alternate, or with a probability $1/a$ if there are a equally capable winners. This gives:

$$p_c = \sum_{j=0}^{C} \sum_{k=0}^{C} \sum_{l=0}^{C} \sum_{a=0}^{N-4} p(j)p(k)p(l)\frac{1}{a+1}p'(i_0, j, k, l, a) \qquad M(i_0, l) > M(j, k) \tag{4.49}$$

4.8.4 Update Equations

The change in state probabilities is calculated as:

$$\frac{\delta p(i)}{\delta t} = \lambda'(i-1)p(i-1) - \lambda'(i)p(i) - i + (i+1)p(i+1) \tag{4.50}$$

where $\lambda'(i)$ is the change due to call arrivals when i trunks are occupied on the route, and i is the change due to calls departing.

Considering call arrivals gives

$$\lambda'(i) = \lambda + \lambda 2p_b(1 - p_b) + \lambda_n(i) \tag{4.51}$$

where p_b is the probability of a call experiencing blocking on the direct route and $\lambda_n(i)$ is the rate of arrival of traffic due to NOAA reroutes.

The NOAA traffic depends on i, the number of trunks occupied on the route now, and j, the number of trunks that may have been occupied at the start of the 5 minute period. Use is made of the A and B matrices. The A matrix contains elements a_{ij}, the probability of a route appearing i times as a NOAA alternate choice given j trunks occupied on the route at the start of the five minute period. The B matrix contains elements b_{jk}, the probability of j trunks being occupied at the start of the 5 minute period given k trunks occupied now. The NOAA traffic is given by:

$$\lambda_n(i) = \sum_{j=0}^{C} \sum_{a=0}^{2(N-3)} \lambda p_b p_a (1-c)(1-p_b) a a_{aj} b_{ji} \tag{4.52}$$

where c is the probability of no NOAA reroute found, and p_a, the probability of being blocked on the alternate route, is given by

$$p_a = 1 - (1 - p_b)^2 \tag{4.53}$$

This iteration of the state probability update equations updates the p_i probability vector and the loop can begin again. Iteration terminates when the change in probability is sufficiently small.

4.8.5 Calculating the Probability of No NOAA Reroute Found

A NOAA reroute is only chosen if it does better than the existing fixed alternate route. Let c be the probability that no NOAA reroute is found. In other words, the route chosen by NOAA is the same as the fixed alternate route. Suppose the fixed alternate route has j trunks occupied on leg 1 and k trunks occupied on leg 2. Then the probability that no NOAA reroute is found is:

$$c = \sum_{j=0}^{C} \sum_{k=0}^{C} p(j)p(k) \left((1 - \sum_{l=0}^{M-1} \sum_{m=0}^{M-1} p_l p_m)^{N-3} \right) \tag{4.54}$$

where $M = \max(j, k)$. This accumulates the probability of finding a NOAA reroute and subtracts it from 1 to find the probability of no NOAA reroute.

4.8.6 Calculating the Network Blocking

The network blocking is defined to be:

$$z = p_b p_a (c + (1 - c)p_a) \tag{4.55}$$

where $p_b = p(C)$, the probability of C trunks being occupied on a route, p_a is the probability of the alternate route being blocked (see Equation 4.53), and the final term is the probability of the NOAA reroute not existing or else existing and being blocked.

4.8.7 Updating the B Matrix

The B matrix needs to be updated at each timestep. A backwards Markov approach using the state transition probabilities is needed to calculate the change in probabilities. In matrix form, this is:

$$B(n + 1) = B(n)S \tag{4.56}$$

$$B(0) = I \tag{4.57}$$

where S is the Markov matrix for transition between states if time were reversed. Recall that the B matrix contains elements b_{jk}, the probability of j trunks being occupied at the start of the 5 minute period given k trunks occupied now. The S matrix contains elements s_{kl}, the probability of k trunks being occupied at a previous timestep given l trunks occupied now.

The following equation is given in [Fel50] for a reversed Markov chain:

$$q_{ij} = \frac{u_j p_{ji}}{u_i} \tag{4.58}$$

for the probability that the system was in state j at the previous time step, given that the system is in state i now. Here u_i and u_j are probabilities from the invariant probability distribution (equilibrium distribution) and p_{ji} is the probability of a transition to state i from state j. Using this formula, and a small timestep, it is found that the Markov matrix for a transition between states if time were reversed is the same as the Markov matrix for a transition between states with time going forward. This makes calculation of the S matrix above straightforward.

Using a state transition diagram gives:

$$s_{ij} = \begin{cases} 1 - \lambda'\delta t - i\delta t & i = j \\ \lambda'\delta t & i = j + 1 \\ (i+1)\delta t & i = j - 1 \\ 0 & \text{otherwise} \end{cases} \tag{4.59}$$

Here λ' denotes the total rate of call arrivals, which was used to update the $p(i)$ vector of probabilities.

4.8.8 Results

Figure 4.6 shows a comparison of the FPM model with simulation. The route independence assumption seems to result in an underestimate of the blocking for low traffic levels but otherwise good agreement is seen for high traffic levels. The same phenomenon was seen in the simple model of hierarchical routing in Section 4.6.

4.9 Conclusions

It seems from our observations that an FPM can give a faster answer about network blocking for symmetric networks than simulations, although that answer may not be as

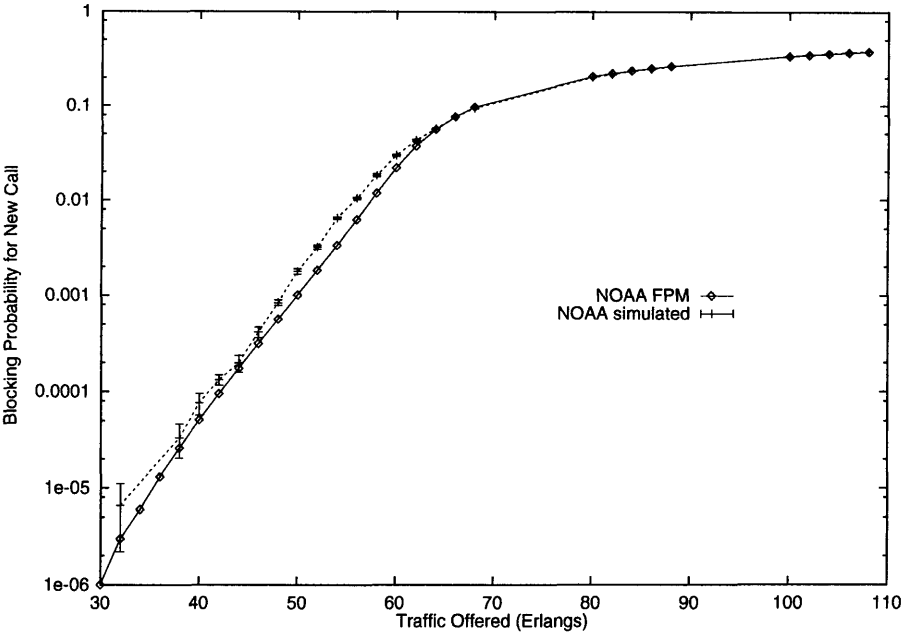

Figure 4.6: Blocking for NOAA FPM and Simulation

accurate as a simulation.

It has been shown that FPMs for symmetric networks are least accurate when traffic levels are low and the network is on the threshold of blocking. The cause of the loss of accuracy seems to be the assumption that blocking on the different paths of a route is independent. For the networks examined in this chapter, it seems to be the case that blocking on the direct route biases the occupancy on the alternate routes to be higher, which means that direct multiplication of the blocking probabilities results in an underestimate of blocking on that route.

For networks with a larger number of alternates used for rerouting, such as AT&T's RTNR rerouting, the increased traffic mix that results from a greater choice of alternates could restore the independence assumption, increasing the accuracy of the FPM. Such networks were not considered in this chapter.

An original FPM for NOAA-style rerouting was derived in this chapter. This was done mainly to verify the accuracy of the results obtained by simulation, since a simulator for NOAA-style rerouting had already been developed. See Chapter 3.

Chapter 5
Stability of Traffic Patterns in Networks with Dynamic Routing

5.1 Introduction

In this chapter, standard control theory is used to model the routing of telephone traffic. A model is developed to investigate the maximum delay in the transmission of routing information that will still allow stability. Simulations are used to validate the model, and to investigate the multi-service case. Finally, conclusions are drawn about applicability of the results.

5.2 Present State of the Art

Much work has already been done in the application of dynamic flow models to telecommunications networks. In a dynamic flow model, stochastic variations of traffic levels are ignored and an approximation is found for the change in the expected levels of traffic using differential equations that relate the time varying call arrival rate, blocking rate, and occupancy levels [KON91, Oht91, FCC89, FC87, LR91].

In [KON91], Kaniyil et al. examine structural instabilities in symmetric telecommunications networks with non-hierarchical routing using potential functions. In this work, time dependent average quantities are used to characterize the state of the system. The existence of two stable states at high traffic levels is shown.

In [Oht91], Ohta uses a dynamic model of a symmetric network to predict the onset of congestion. The intention is to implement controls prior to congestion to keep the network operating at full efficiency. A dynamic flow model will show that there is a delay between the sudden increase in call arrivals and the onset of congestion. Ohta demonstrates the feasibility of an advanced network management system which makes use of congestion prediction.

In [FCC89, FC87], Filipiak et al. present a framework for estimating future occupancy statistics in a communications network based on present measurements of occupancy, arrival rate, and holding time. Good results are obtained by comparing the model predictions

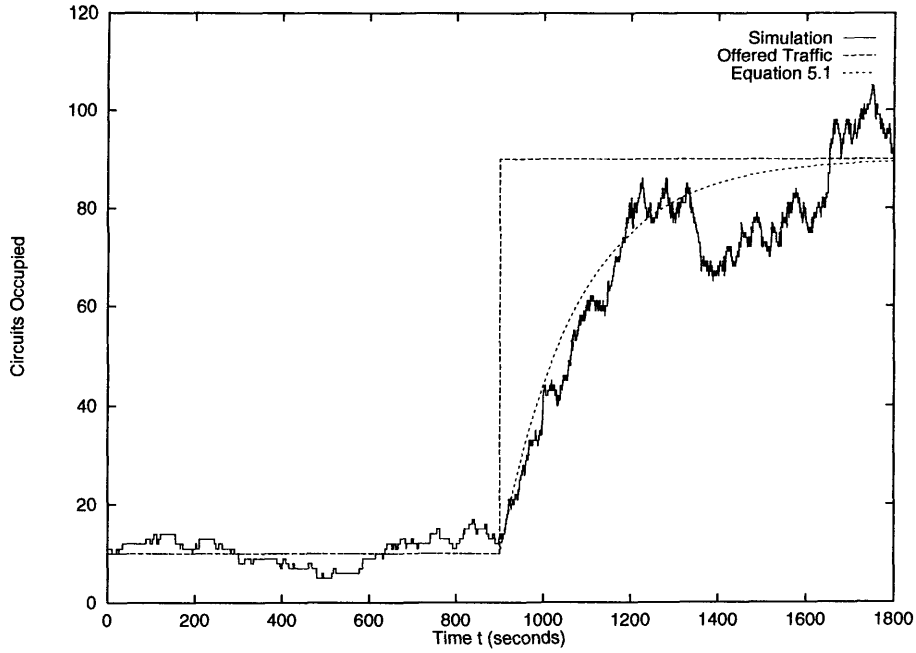

Figure 5.1: Step Increase in Traffic at t = 900

with measured values taken from the French telephone network.

In [LR91], Filipiak et al. apply the same theory to the dynamic rerouting that was part of the Toronto trial of Dynamically Controlled Rerouting (DCR) in the Canadian Network. Their simulations show that the results obtained are more accurate in the case of high load than the estimation and prediction methods used in the trial.

5.3 Model of a Single Route

An important component of a network model is the model of a single route. Figure 5.1 shows the rise in traffic level following a step rise in offered traffic. Using generating functions, Cooper[Coo81] shows that the expected number of trunks occupied following a step increase in traffic from N_0 to N_1 is given by

$$N(t) = N_1 - (N_1 - N_0) * \exp(-t/T_h) \tag{5.1}$$

where T_h is the average holding time of telephone calls, N_0 is the initial traffic level, and N_1 is the new traffic level.

This system can be modeled as a linear time invariant system with a transfer function

of

$$F(s) = \frac{1}{(1 + sT_h)} \tag{5.2}$$

In control theory terms, this is a system with a time lag T_h. The stochastic variation about the expected level is regarded as noise and not explicitly modeled. This is a standard assumption in control theory.

5.4 Stability of Reroute Controls

These studies originated from the development of a telephone network management tool, called NOAA (Network Operations Analyzer and Assistant). NOAA is described in Appendix A.

As reroute controls become more automated concerns arise about the stability of the automated systems. In particular, it is desirable to find out how big of a margin exists between stable and unstable behavior, in terms of the parameters that specify the system.

For this purpose, IRR (Immediate Reroute) and ORR (Overflow Reroute) controls are distinguished. An IRR control reroutes traffic before it attempts the problem route. Traffic is diverted elsewhere in the network where spare capacity exists. An ORR control reroutes traffic after it attempts the problem route and finds no capacity available. The ORR control offers that particular call an extra chance of completion.

From a stability viewpoint, one would expect to see fewer problems with ORR controls as each call tries the standard routing first and then tries the added routing options specified by the control. For each call that tries the problem route, its chances of completion are increased by the implementation of an ORR control. In the case of small overloads, network throughput can only be expected to increase, as individual calls have more possibilities of completion.

For an IRR control, if too much traffic is diverted and the delays in obtaining the feedback about traffic information are too great, there is a possibility of instability. Traffic from route A could be diverted to Route B which may then experience a problem and find spare capacity on Route A. The overload situation could oscillate between route A and route B, causing an overall decrease in throughput.

The model in Figure 5.2 was used to analyze a system with two possible routes between some source destination pair. Observations about the traffic level on either route are used

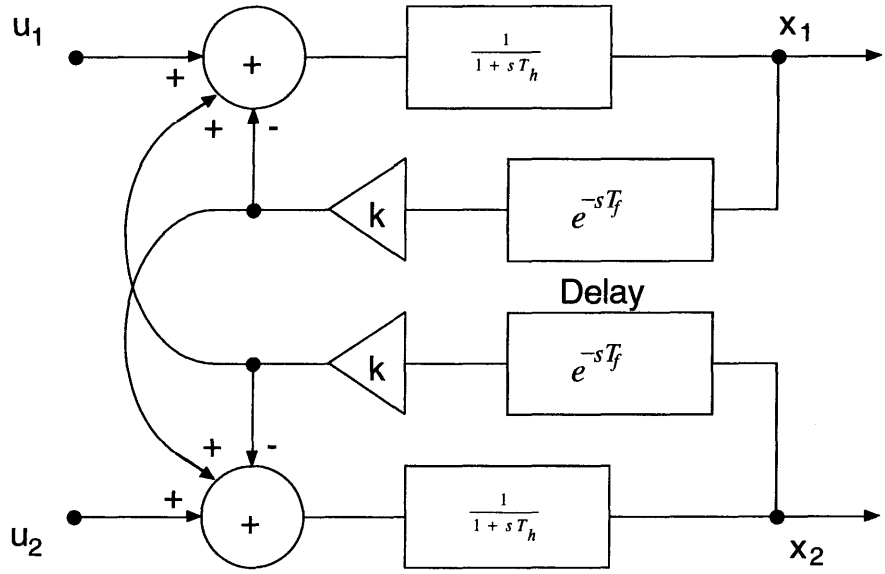

Figure 5.2: Rerouting Model

to decide on what percentage of new call arrivals to divert to the other route. It is assumed that IRR controls may be applied to either route. This model is simplified in two respects:

- Network Management uses sampled information. Traffic information from the switches is typically available every 5 minutes or 30 seconds and not continuously. This is not modeled here.
- Network Management takes no action until a route overflows. This could be modeled by an element in the feedback path such as a relay with dead-space. However that has not been done here.

The purpose of the analysis is to examine the order of magnitude of the time constants that could result in instability.

5.5 Model of Rerouting Algorithm

The model of the rerouting algorithm is shown in Figure 5.2. The figure shows two routes with an independent stream of traffic offered to both, and some feedback controls based on the observed traffic. Observations about the traffic level on either route are used to

decide on what percentage of new call arrivals to divert to the other route. Here u_i is the offered traffic on route i and x_i is the observed traffic on route i.

The model includes a delay term of T_f seconds in the feedback of control information. The model also includes a gain factor k in the feedback path. The gain factor k is a measure of how much traffic is diverted by the network management control from a full route to an empty one in steady state. If α is the fraction of traffic diverted from a full route to an empty one in steady state then

$$k = \frac{\alpha}{1 - 2\alpha} \tag{5.3}$$

This can be derived from the model by removing the lag factor and delay once steady state and constant inputs are assumed. It is assumed that $\alpha \leq 0.5$. In other words, the most aggressive traffic balancing would split the traffic equally between trunk groups.

In the NOAA network management application, $k = 0.187$ but typically k can vary from 0 to ∞ depending on the aggressiveness of the network management.

In the NOAA system, typical values for the holding time T_h and the feedback delay T_f are 3 minutes and 5 minutes respectively.

5.6 Analysis

The system shown in Figure 5.2 is a standard multi-input multi-output system from the control theory point of view. However the presence of the delay terms makes the analysis a little more difficult. Looking at the output of the adders:

$$x_1(1 + sT_h) = u_1 - ke^{-sT_f}x_1 + ke^{-sT_f}x_2 \tag{5.4}$$

$$x_2(1 + sT_h) = u_2 - ke^{-sT_f}x_2 + ke^{-sT_f}x_1 \tag{5.5}$$

Grouping terms gives:

$$x_1(1 + sT_h + ke^{-sT_f}) = u_1 + ke^{-sT_f}x_2 \tag{5.6}$$

$$x_2(1 + sT_h + ke^{-sT_f}) = u_2 + ke^{-sT_f}x_1 \tag{5.7}$$

The definition of stability is from [DSW73]:

Definition 1 *A system is stable if its impulse response approaches zero as time approaches infinity.*

To check for stability, set $u_1 = 0$ and multiply the first equation by $(1 + sT_h + ke^{-sT_f})$ to eliminate x_2. This gives:

$$x_1(1 + sT_h + ke^{-sT_f})^2 = ke^{-sT_f}(u_2 + ke^{sT_f}x_1) \tag{5.8}$$

and thus

$$\frac{x_1}{u_2} = \frac{ke^{-sT_f}}{(1 + T_h s + ke^{-sT_f})^2 - (ke^{-sT_f})^2} \tag{5.9}$$

Note that the transfer function for x_1/u_1 also has the same denominator. A sufficient condition for instability is for the denominator to be 0 for some value of s on the $s = j\omega$ axis. This critical threshold of stability is of interest to us. Looking for a 0 denominator gives:

$$(1 + sT_h + ke^{-sT_f}) = -(ke^{-sT_f}) \tag{5.10}$$

or

$$1 + sT_h = -2ke^{-sT_f} \tag{5.11}$$

Setting $s = j\omega$ and equating real and imaginary parts gives:

$$1 = -2k\cos(\omega T_f) \tag{5.12}$$

$$\omega T_h = 2k\sin(\omega T_f) \tag{5.13}$$

These equations give some interesting results

- For $k < 0.5$, this mode of instability does not arise. This is a consequence of equation 5.12.
- The critical value of the feedback time can be solved for given some value of $k > 0.5$.

For example, if $k = 3$ and $T_h = 3$ minutes, then it is possible to solve for $\omega = 3.416$ which leads to $T_{osc} = 1.840$ minutes for the period of oscillation. Also T_f, the critical value of the feedback delay, is found to be 0.509 minutes. This suggests that if $T_f > 0.509$ one should see some signs of instability.

5.7 Simulation Study #1

To verify the effects found in the analysis, a simulation is carried out. In the simulation, 80 Erlangs of traffic are offered to each of two routes which are assumed to have 100 trunks.

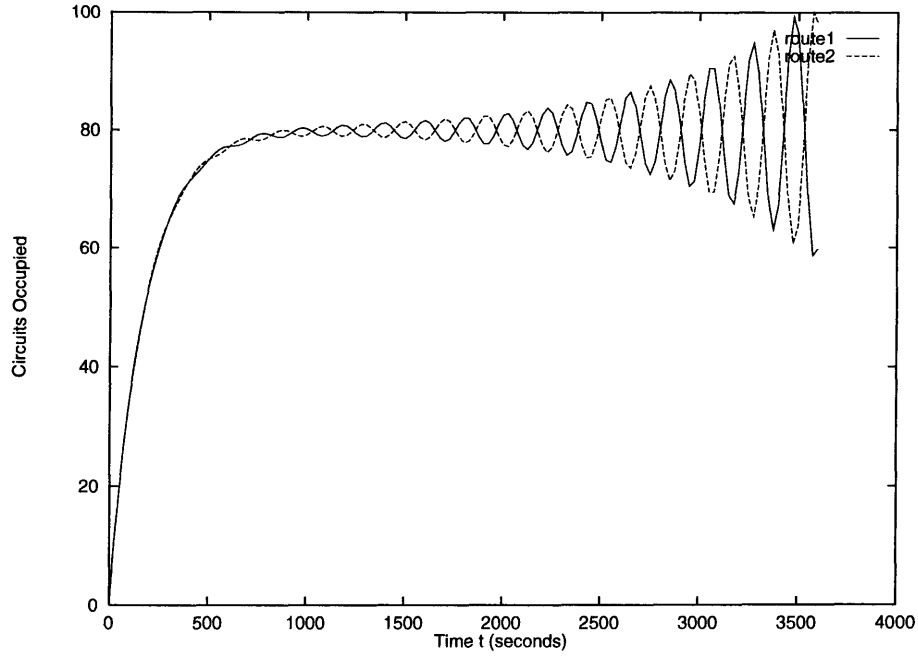

Figure 5.3: Route Occupancy Assuming Automated Rerouting ($T_f = 1$ minute)

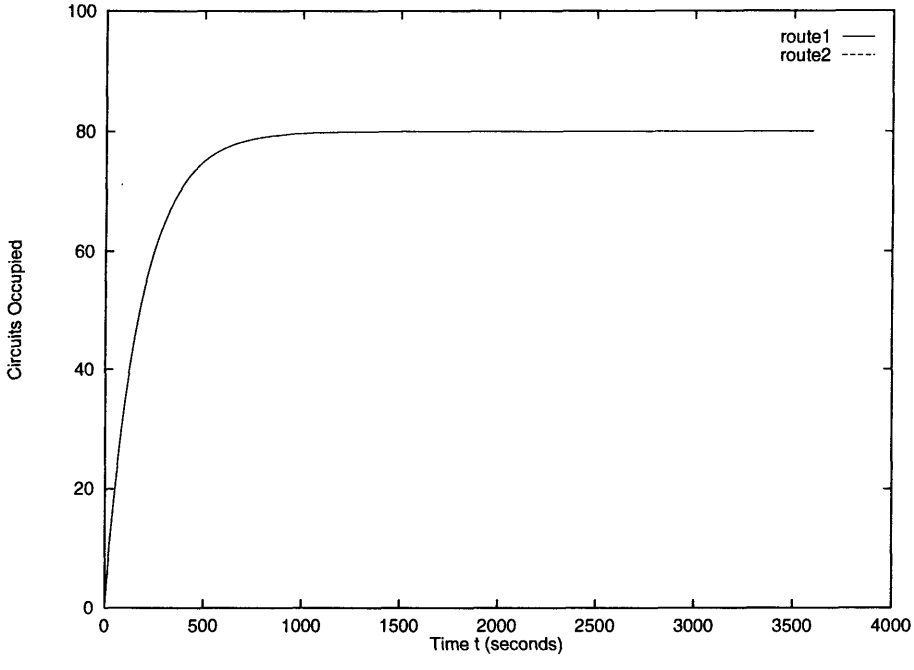

Figure 5.4: Route Occupancy Assuming Automated Rerouting ($T_f = 0.1$ minute)

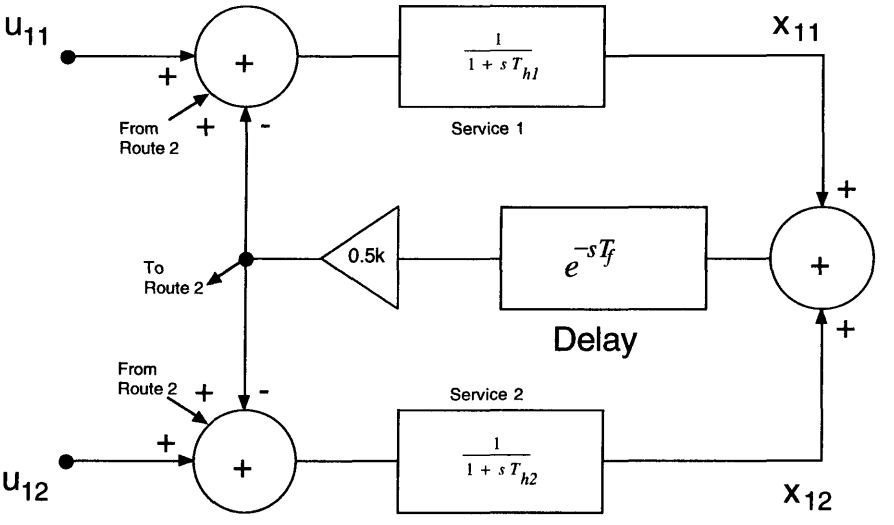

Figure 5.5: Rerouting Model

A holding time of 3 minutes, a variable feedback delay and a gain of $k = 3$ are assumed. The results are shown in Figures 5.3 and 5.4. Instability can clearly be seen for the larger value of T_f.

5.8 Simulation Study #2

To see how this situation is affected by the addition of an additional service with a longer holding time, a second service is added. The top half of the model in Figure 5.2 is replaced by the two service counterpart shown in Figure 5.5. The results indicate that the service with the shorter holding time dominates.

This would seem to indicate that a network that carried speech and short data "conversations" would need feedback controls that had time constants with very small delay to avoid instabilities, assuming a control algorithm similar to those used today.

5.9 Conclusions

Standard control theory has been applied to a network management problem with the aim of seeing whether the automation of network management controls could result in instabilities. The conclusion is that this can happen. Careful control of the delays in the

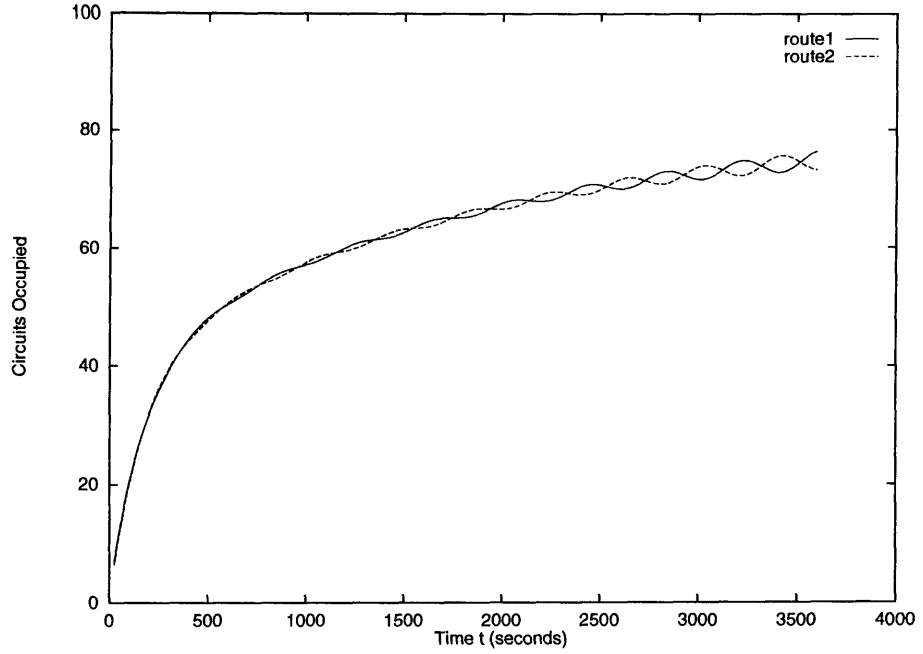

Figure 5.6: Route Occupancy Assuming Automated Rerouting and Two Services ($T_f = 2.0$ minutes)

feedback of information used to calculate routing strategy is necessary in order to prevent this.

This analysis should be applicable to control schemes such as RTNR (Real Time Network Rerouting) described in [ACF92] and presently implemented in the AT&T long distance network. The RTNR routing scheme is a dynamic routing scheme that makes use of cached information to decide the route to be taken for a call. In this chapter, it is suggested that if the cache becomes too old (perhaps due to delays in the signaling network), instabilities can arise.

Part II

Application of Learning Techniques to Network Management

Chapter 6
Application of Learning Techniques to Network Management

6.1 Learning Techniques

The learning techniques used in the following chapters include neural networks and linear predictors. A brief description of both techniques is given here.

6.1.1 Linear Predictors

A linear predictor is really a simple type of neural network whose output is a linear combination of its inputs. See Figure 6.1.

Suppose there are N inputs x_i to the linear predictor, which has weights w_i and an output y. Then the output y for an input vector X is given by:

$$y(X) = \sum_{i=1}^{N} w_i x_i \tag{6.1}$$

Suppose now that there are P input vectors X_p, each of which is of the form $(x_1^p, \ldots, x_N^p)$, and a desired output $\hat{y}_p$ for each of these input vectors. Then a sum of squared error (SSE) function E can be defined as follows:

$$E = \sum_{p=1}^{P} (\hat{y}_p - y(X_p))^2 \tag{6.2}$$

A possible learning rule would be to carry out an adjustment of each weight to minimize the error E over all the training patterns:

$$w_i(\text{new}) = w_i - \eta \frac{\delta E}{\delta w_i} \tag{6.3}$$

where η is a small constant, termed the learning rate. If the learning rate is too small, convergence is slow. If the learning rate is too big, there may be no convergence. Trial and error is used to find the appropriate learning rate.

Straightforward differentiation gives:

$$\frac{\delta E}{\delta w_i} = -2 \sum_{p=1}^{P} x_i^p (\hat{y}_p - y(X_p)) \tag{6.4}$$

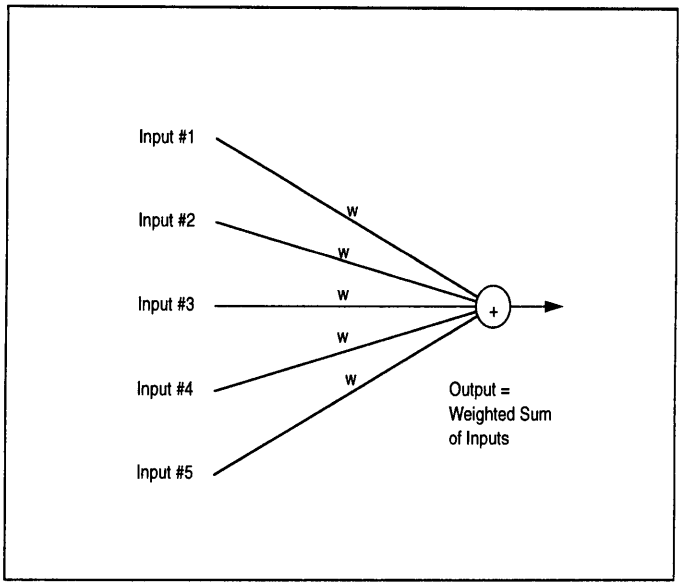

Figure 6.1: Linear Predictor

From Equations 6.3 and 6.4, provided η is sufficiently small, the optimum weights to minimize the error, E, with respect to the weights can be found. As a practical matter, an extra input is always added to the linear predictor which is set to a constant, either 1 or -1. The constant input gets multiplied by a weight to give a constant term in Equation 6.1. This allows the linear predictor to estimate a wider range of functions. Equation 6.1 now becomes:

$$y(X) = \sum_{i=1}^{N} w_i x_i + C \tag{6.5}$$

6.1.2 Neural Network Architectures

Many neural network architectures can be found in the neural network literature, for example, feed-forward neural networks, recurrent neural networks, and Hopfield networks. Feedforward neural network will be discussed shortly. Recurrent neural networks are similar to feed-forward networks, except that they contain feedback connections. Hopfield networks are usually used for optimization problems. In this thesis, feed-forward neural networks are mainly considered. A good reference for all these architectures and associated training techniques is [HKP91].

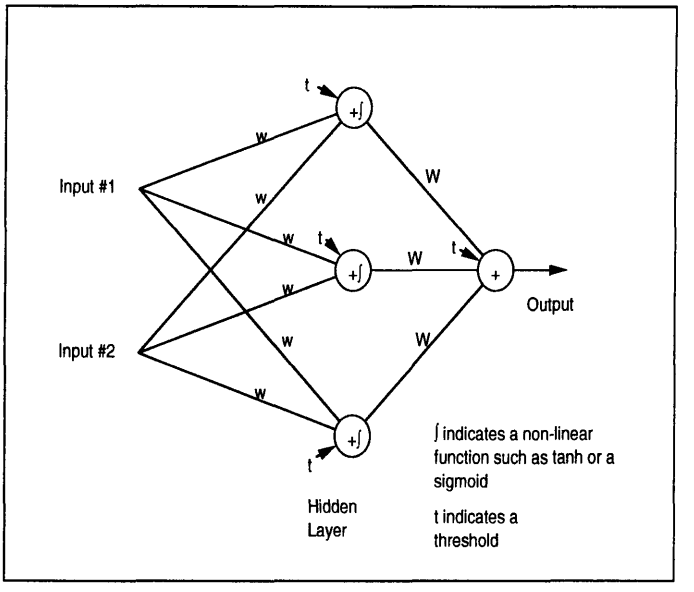

Figure 6.2: Feed-forward Neural Network

6.1.3 Feed-Forward Neural Networks

The feed-forward neural network is one of the most widely used neural networks. An example of a simple feed-forward neural network is shown in Figure 6.2. In this case, the output y as a function of the inputs x_1^p and x_2^p for input pattern p is given by:

$$y = \sum_{i=1}^{3} W_i \tanh(\sum_{j=1}^{2} w_{ij} x_j^p - \theta_i) - \Theta \tag{6.6}$$

Here θ_i, the threshold term, can be treated as a weight applied to an extra input set to a constant -1. In this case, there are 3 so-called hidden units, each with output:

$$h_i = \tanh(\sum_{j=1}^{2} w_{ij} x_j^p - \theta_i) \tag{6.7}$$

If it is assumed that there are P input vectors X_p, each with a desired output $\hat{y}_p$, an SSE term E can be defined:

$$E = \sum_{p=1}^{P} (\hat{y}_p - y(X_p))^2 \tag{6.8}$$

The first layer weights are updated according to:

$$w_{ij}(\text{new}) = w_{ij} - \eta \frac{\delta E}{\delta w_{ij}} \tag{6.9}$$

Similarly, second layer weights are updated according to:

$$W_i(\text{new}) = W_i - \eta \frac{\delta E}{\delta W_i} \tag{6.10}$$

using the Back-Propagation algorithm [HKP91].

6.1.4 Back-Propagation

The Back-Propagation algorithm can be derived by differentiating Equation 6.8 with respect to the appropriate weight. It derives its name from that fact that errors are propagated from the network outputs towards the network inputs.

Define e_p as the error when input pattern p is presented. Then

$$e_p = \hat{y}_p - y(X_p) \tag{6.11}$$

and the total error E is given by:

$$E = \sum_{p=1}^{P} e_p^2 \tag{6.12}$$

The desired derivative for any weight w is given by:

$$\frac{\delta E}{\delta w} = \sum_{p=1}^{P} \frac{\delta e_p}{\delta w} \tag{6.13}$$

The derivative of e_p with respect to W_i is given by:

$$\frac{\delta e_p}{\delta W_i} = -2e_p h_i \tag{6.14}$$

where h_i is the output of the hidden unit, given by Equation 6.7. The derivative of e_p with respect to w_{ij} is given by:

$$\frac{\delta e_p}{\delta w_{ij}} = -2e_p(1 - h_i^2)x_j^p \tag{6.15}$$

using the chain rule for differentiation and the fact that

$$\frac{\delta \tanh(x)}{\delta x} = (1 - \tanh(x)^2) \tag{6.16}$$

By substituting Equations 6.14 and 6.15 in Equation 6.13 and then using Equations 6.9 and 6.10, the desired weight update can be calculated.

6.2 Motivation for the use of Neural Networks

Neural networks have traditionally been used for pattern recognition and pattern classification and are well suited to this task. The weights in such networks can be learned from

sample data and training can be done using the well known backpropagation algorithm [HKP91].

The neural network approach is valuable for a number of reasons. Firstly the neural network has the ability to learn. Many software systems in the telephone network require the intervention of humans if the implemented function is to be modified. Neural networks on the other hand contain their "program" in their weight settings and can even continuously update their weights as they are running. This automated learning capability is a key benefit of neural networks.

In addition to the ability to learn, the neural network has the ability to generalize on data that was not present in the training set. Neural networks have been characterized as universal function approximators [Bar93, Cyb89, HSW89]. In other words, given sufficient number of hidden units, neural networks can approximate any given well behaved function. Conversely, given fewer hidden units, a smoother version of the target function can be learned. This explains the ability of neural networks to generalize well, even on input data that contains a lot of noise and artifacts. Neural network are compared to other non-linear function approximation techniques in chapter 9.

6.3 Network Management

The task of network management is to monitor the network, identify any anomalies in the network, and, if necessary, place controls in the network to allow revenue generating traffic to complete and block any traffic that has a low probability of completion.

6.4 Applications of Learning Techniques to Network Management

In the following chapters, a number of applications of learning techniques to network management are considered:

Recognition of Traffic Patterns Clearly this is an integral part of the job of network traffic managers.

Aiding Congestion Control If the network becomes congested, then various controls are available to the traffic manager to relieve this congestion. The neural network can learn the thresholds of when to apply such controls.

Time Series Prediction of Trunk Group Occupancy As network management becomes more automated, it is possible to predict trunk group occupancy and place reroute controls that require fewer adjustments to changing traffic conditions in the network.

Conclusions on the applicability of the learning techniques are given in each chapter.

Chapter 7
Classifying Telephone Traffic Patterns

7.1 Introduction

Network management of communication networks is an activity that is becoming increasingly automated. The impetus for this automation is firstly the increased computing power available to help provide a rapid response to changing network conditions and secondly the adoption of common standards for the exchange of information between the systems being managed and the management systems.

Operators may use *expansive* or *restrictive* controls to respond to network exceptions. Expansive controls are used if the network contains capacity that can be used to carry some of the extra traffic. Expansive controls reroute calls in order to give them an extra chance of completion. Restrictive controls are used if the extra traffic has a low probability of completion and is interfering with normal network operations. Restrictive controls relieve congestion by cutting down on traffic at the point it enters the network.

Network exceptions that the operators can monitor are trunk group overflows, trunk group high occupancy, and switch alarms. Each trunk group can handle a limited number of conversations, and if too many call attempts are offered, the trunk group occupancy will rise and eventually call attempts will be rejected. This is referred to as trunk group overflow. The network management center receives data for every trunk group in the network every five minutes.

Switches can also indicate the onset of congestion or the existence of certain conditions by means of switch alarms. The network management center receives a list of switch alarms in the network every thirty seconds.

7.2 Pattern Matching of Network Events

The following is a partial list of network events of interest to network management:

- single random trunk group overflow
- loss of a transmission link

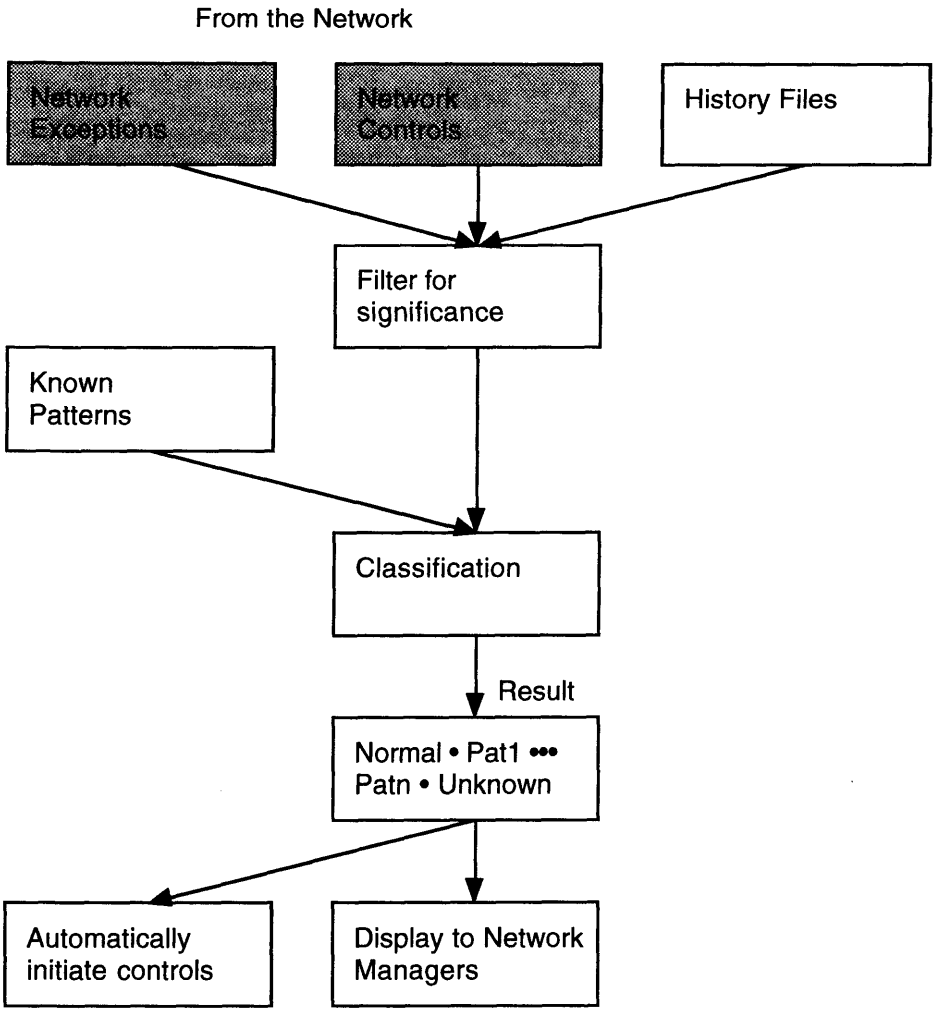

Figure 7.1: Traffic Pattern Recognition System

- loss of a switch
- call-in situation, e.g. concert tickets on sale.
- unusual traffic patterns, e.g. Mothers' Day.
- weather day, e.g. heavy day of snow.
- natural disaster, e.g. earthquake.

Ideally, the operator would wish to know the event type, the event location, and the event severity. If it is assumed that occasionally new event types occur, then the operator would also wish to know to that this event type is not one previously observed.

Figure 7.1 shows the information flow for a system that could provide this information. The current list of exceptions and controls is compared to historical data to check the significance of the departure from normal state. The filtered traffic pattern is then compared to a table of known traffic patterns and classification is performed. If the traffic pattern is sufficiently different from known traffic patterns, then the classifier should indicate this result.

Some similarities exist between this and character recognition. The differences are that the traffic pattern learning system should be capable of unsupervised learning and detection of new classes.

The benefits of such a system as an extra resource for the decision making process of the network management operators or of the NOAA expert system are obvious.

7.3 Data Set

The data set for this study consisted of 125 days of traffic data from June 1994 to Jan 1995 with some gaps. The logged data for each day consisted of a list of network overflow exceptions and network controls that were present in the network every 5 minutes.

This logged data was too much data and too little data at the same time. Too much because of the work involved in extracting historical records from 600 Mbytes of data, and too little because the number of abnormal events was so few. During the period of interest, observed one weather day, and a small number of call-ins were observed. No other significant deviation from normal operations was observed.

7.4 Filtering of Significant Events

The approach taken to filter out significant events from the large data set was to remove routes that regularly or even occasionally overflow. This was done by plotting the data on a grid and then estimating the probability that a given pixel would indicate activity at that particular day at that particular hour. This is equivalent to looking at prediction residuals when inferring the state of the network and is similar to the Hidden Markov Model techniques of Section 9.13 of Chapter 9.

Figure 7.2 shows an example of a 64x64 pixel map. Each pixel is turned on if at that time there is a network exception or control in that geographic area.

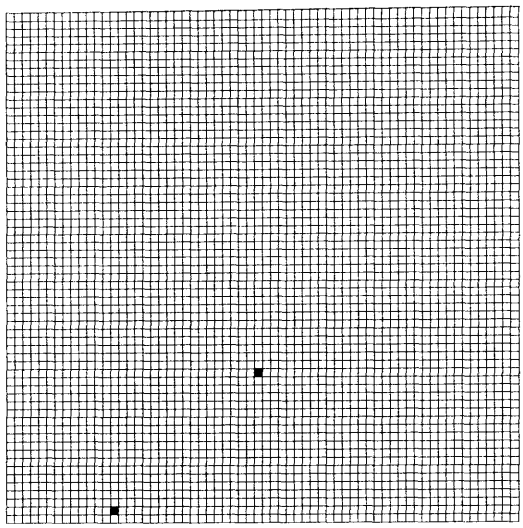

Figure 7.2: Example of a Pixel Map Indicating Network Activity

The real data was examined on a grid of size 50 miles by 50 miles centered on Los Angeles broken up into 64x64 pixels. Each pixel was a square with a width and height of 50/64 miles. The size of grid chosen was a compromise. Too small a grid and one does not get a global view of the network. Too large a grid and the chances of two simultaneous events taking place increases and classification is more difficult. The 50 mile by 50 mile grid seemed to work well.

As mentioned above, an estimate $p(i,j,t)$, the probability of a pixel at position (i,j) being turned on at time t, was sought. The time t was specified to accuracy level of a given hour on a given day. For example, a unique record would be associated with Thursdays 10am to 11am.

The data for estimating this probability was a history of some number N of previous records for this day and time, on which the pixel was turned on on x occasions.

Two choices arose for the estimation of probabilities. The Maximum Likelihood (ML) estimator is:

$$\hat{p}_M(i,j,t) = x/N$$

and a Bayes estimator making certain simplifying assumptions discussed below is:

$$\hat{p}_B(i,j,t) = \frac{x+1}{N+2}$$

The ML estimator maximizes $f(x|\theta)$ with respect to θ where f is the probability density function, x is the observation, and θ is the parameter being estimated.

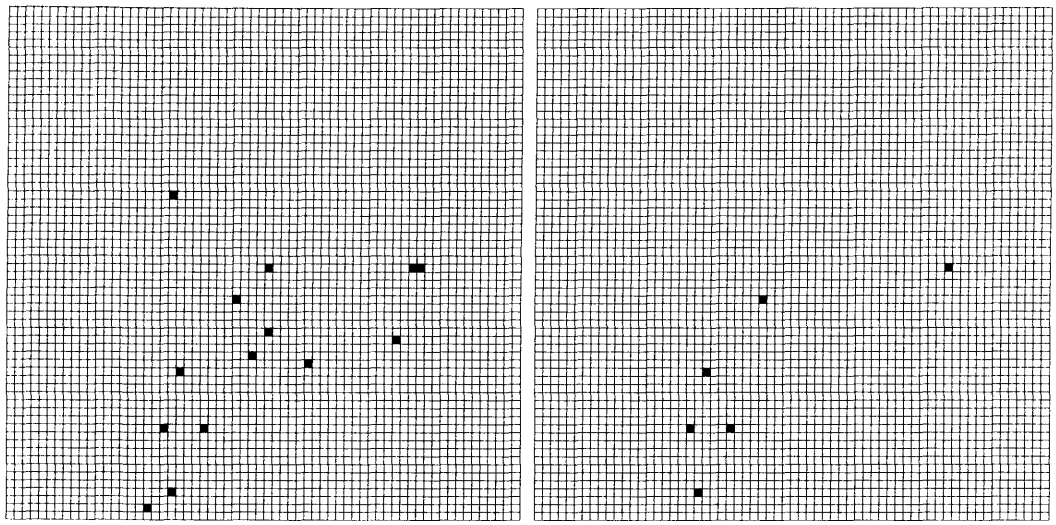

Figure 7.3: A Rainy Tuesday in Los Angeles Before and After Filtering. Filtering Removes Routes that Regularly or Even Occasionally Overflow on that Day at that Time.

The Bayes estimator is E(θ|x), the expected value of the unknown parameter θ given some observation x. The Bayes estimator requires the specification of a prior distribution for the parameter θ. The prior distribution contains the best estimate of θ prior to taking any measurements. Here the prior distribution was taken to be a uniform distribution between 0 and 1. Refer to [Ros87] for details of how both ML and Bayes estimators are derived for a Bernoulli trial as is the case here.

For probabilities that are small and limited data, the Bayes estimator does a better job of estimating the probability, especially with regard to testing the significance of future events. Consider the case where there are seven records, none of which show the pixel on.

The ML estimator gives $\hat{p}_M = 0.000$. The Bayes estimate gives $\hat{p}_B = 0.125$. If in the next record, it is found that the pixel turned on, this has a probability of 0.000 using the ML estimator and 0.125 using the Bayes estimator. The Bayes estimator, because it avoids zero probabilities, is more useful.

Each pixel of the pixel map can be assigned a probability of being on at any given time. A filtering step can then be carried out to show only pixels that are significant at the 97% level. An example of such a filtered image is given in Section 7.5.

7.5 Results

Figure 7.3 shows the results of the filtering step on a Tuesday afternoon with floods and really heavy rain in Los Angeles at 3pm. The left part of the diagram shows the pixel map before filtering. The right part shows the pixel map after filtering. Filtering is at a significance level of 97%. Pixels that are turned on represent network exceptions or controls.

Of all the pixels in the map, only 57 pixels have non-zero probabilities because they denote the location of telephone switches. The hypothesis that this is a normal Tuesday afternoon can be rejected at a 99.4% confidence level, assuming 6 successes from 57 Bernoulli trails with $p \leq 0.03$. The significance figure is information that the event classifier can use to recognize events.

It could be argued that the map pixels are not independent. Taking account of the fact that trunk groups are used to connect switching offices, one can imagine a more strict test for significance that would divide the number of pixels turned on by 2 to account for the fact that a problem on a single trunk group could result in two pixels being turned on, one at either end. The probability of a success would similarly be divided by 2, giving $p \leq 0.015$. Now with 3 successes from 57 trials, the calculated significance figure is still above 96%.

7.6 Conclusions and Future Work

This has been a first attempt at learning traffic patterns that are present in telephone networks. Some success has been obtained in detecting traffic patterns that are different from normal. An efficient and easy to implement pattern filtering stage has been proposed.

As more data is obtained, the emphasis will be on the unsupervised learning of traffic patterns. Options for this unsupervised learning are:

- Generative Models[SM92, Smy94]. Storing a template and using that as the pattern to be matched should be a feasible classification method. If the distance between the new pattern and the stored patterns is too great then a new class is declared.

- ART[HKP91, SB94] is a classification technique that has an adjustable vigilance parameter that controls when new classes are formed. It should be straightforward

to use the pixels as binary inputs for ART.

- clustering e.g. k-means[HKP91]. This is a standard method of unsupervised learning.

The use of a HMM (Hidden Markov Model) suggests itself as a means of increased confidence in the assessment of the hidden state of the system, given a set of observables [Smy93, GA93, Smy94]. HMMs are discussed in Section 9.13 of Chapter 9. However, the "attack phase" of network events is the primary interest, in order to provide the correct response, and the HMM is less useful during this phase.

In summary, there is hope for the development of an automated network traffic pattern recognition system, and some of the necessary steps in filtering network traffic pattern data have been successfully carried out and reported upon in this chapter.

Chapter 8
Learning Telephone Network Trunk Reservation Congestion Control using Neural Networks

8.1 Introduction

Congestion control is an important topic in the design of telephone networks and will be even more so in the design of ATM networks. In this chapter, neural networks are used to decide the level of trunk reservation to apply in congestion situations. The communication networks investigated are symmetric fully connected telephone networks.

8.2 Trunk Reservation

8.2.1 Definition

A simple example of a small network is given in Figure 8.1. This shows a simple five node network, with nodes labeled A to E. Each pair of nodes has a set of links between them. This small network can be used to demonstrate the concepts underlying trunk reservation.

Calls are usually set up using either one or two hops. For example, a call from A to E could be setup using either the trunks from A to E, or else A to B and then B to E. In lightly loaded traffic conditions, using two hops allows more opportunities to complete telephone calls. However in heavily loaded traffic conditions, the two trunk groups used by a two hop call could instead be used to set up two one hop calls and reduce the network blocking.

One hop traffic is called *direct routed* traffic. Two hop traffic is called *alternate routed* traffic.

Definition 2 *Trunk reservation is a policy whereby in each trunk group, alternate routed traffic is blocked if there are fewer than R trunks free on the trunk group. R is known as the trunk reservation parameter.*

8.2.2 Simulation

Trunk reservation favors the direct routed traffic at the expense of alternate routed traffic, increasing the network efficiency in times of high load. The parameter R can be very small and still have a substantial effect, as is illustrated in Figure 8.2.

In this figure, the probability of a new call arrival being blocked in a symmetric network is shown. The network has 10 nodes and 100 trunks between each pair of nodes and 111 Erlangs offered to each trunk group. Trunk reservation is only enabled 2.5 minutes into the simulation. A trunk reservation parameter of 1 is used. The blocking shows an almost immediate decrease. The decrease is not immediate because trunk reservation is a blocking policy which blocks alternate traffic, so that shortly after trunk reservation is enabled the probability of being blocked on a direct route is unchanged but the probability of being blocked on an alternate route has increased. Soon a new equilibrium is reached in which the overall blocking is decreased since less calls are blocked on the direct route.

Figure 8.3 shows an alternate view of the effect of implementing trunk reservation. Here the percentage of direct routed calls to total calls is plotted during the course of the simulation. Once again, enabling trunk reservation on all trunk groups at $t = 2.5$ minutes into the simulation has a dramatic effect on increasing the proportion of direct routed traffic in the network. The peak at $t = 1$ minute is due to an initial transient as calls arrive to an empty network.

Figure 8.4 demonstrates that an overload situation is indeed being simulated. It plots the state probabilities, that is, the probability of n trunks occupied out of a 100 on a typical trunk group, at $t = 2.49$ minutes. This is just prior to trunk reservation being enabled. Clearly there is a significant probability that all 100 trunks are occupied on the trunk group, indicating that the network is in an overloaded state.

8.2.3 Known Results Concerning Trunk Reservation

Akinpelu in [Aki84] shows that trunk reservation prevents networks having two stable states at high loads. Kelly in [Kel94] references a result by Hunt and Laws that a policy that chooses the least busy alternative for routing and implements trunk reservation is an asymptotically optimal policy in minimizing blocked traffic, as the number of network nodes increases.

Closed form solutions for the correct value of the trunk reservation parameter, R, to

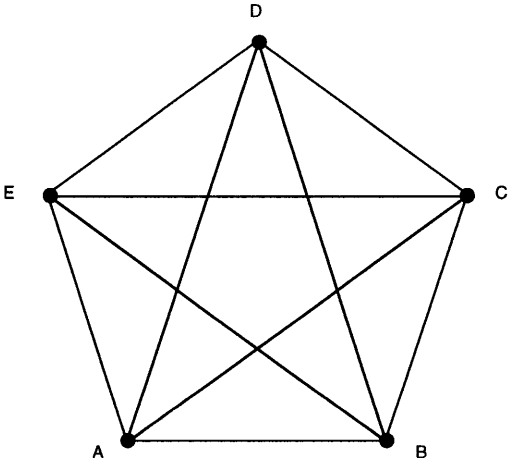

Figure 8.1: Example of a Symmetric Five Node Network

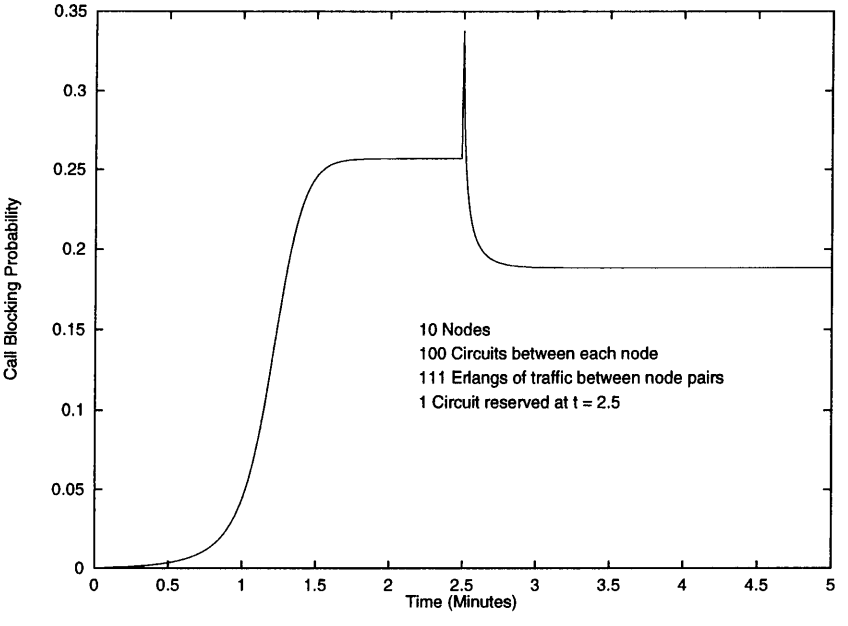

Figure 8.2: Effect of Turning on Trunk Reservation at $t = 2.5$

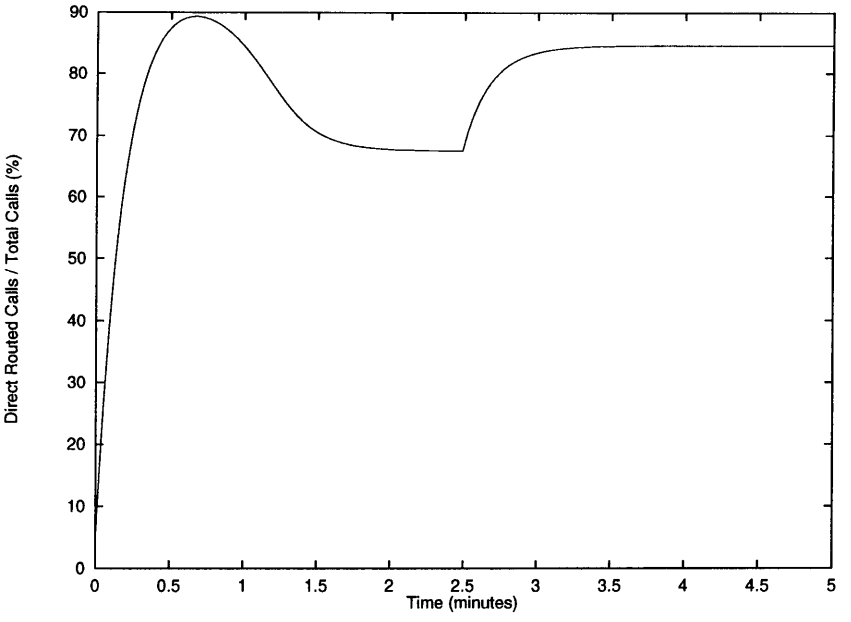

Figure 8.3: Effect of Turning on Trunk Reservation at $t = 2.5$

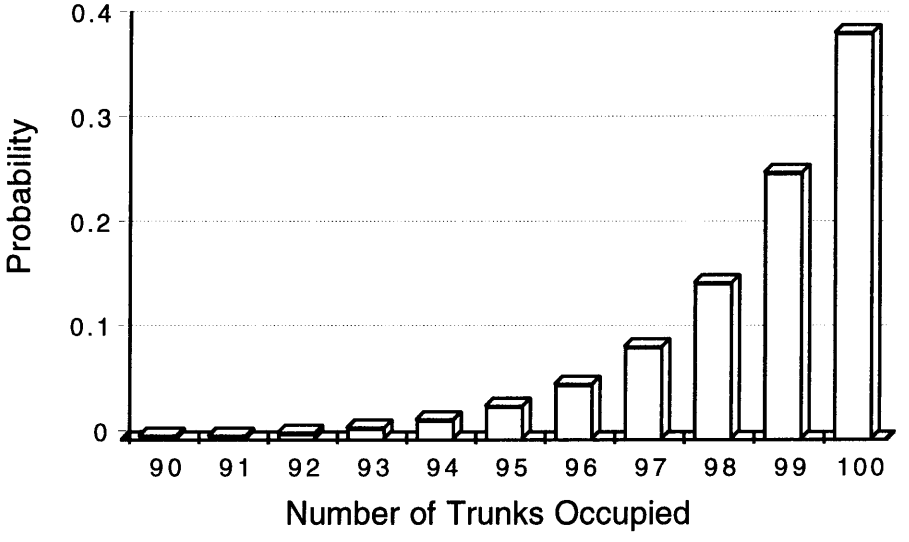

Figure 8.4: Probability of n Trunks Occupied out of 100 on a Typical Trunk Group

use as a function of network load, traffic mix on the trunk group, and number of trunks on the trunk group are not known, except in the asymptotic case for symmetric networks with many nodes and the same traffic offered to each node [MS91, MG92]. In this chapter, a neural network is used to choose the value of the trunk reservation parameter as a function of input variables that will be described later.

In the remainder of the chapter, the constraints for this problem, training the neural network, the architecture of the neural network, the traffic mix, results, and conclusions are discussed.

8.3 Constraints

In a large network, it is unrealistic to expect each node to have a global view of the network. This is especially true when the network contains switches from different vendors. In such a case, there would have to be agreement between the vendors on the format of the messages communicating the switch state between all the switches. Such agreement could take years to accomplish. Instead it was decided to require the inputs to the neural network to be local information, in other words, statistics about the route or statistics about the switch to which the route is attached.

The system proposed would have R_N separate neural networks, where R_N is the number of routes in the network. Since each neural network is being trained to optimize trunk reservation on a single trunk group, questions of global network stability would have to be addressed before implementing such a system. These questions are not addressed in this study.

The main benefit of using a neural network on this problem is the ability of the network to adjust its weights to reflect changes in the nature of the traffic in the communications network. An indirect learning approach is taken. In this case, the neural network is not provided with training examples taken from analytical studies of trunk reservation. Instead, simulation is used to provide the training set. The neural network is required to adjust its weights to minimize the overall network blocking in the simulated network.

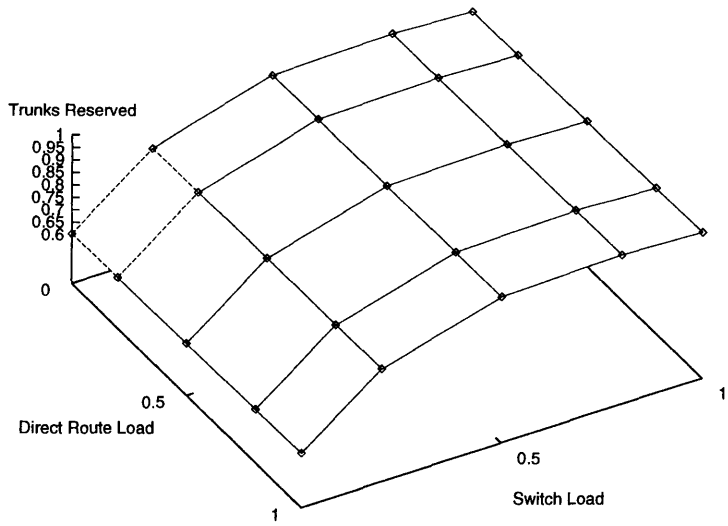

Figure 8.5: Neural Network Output

8.4 Neural Network

The neural network has two inputs and two hidden units. A linear output unit is used to aid in function fitting. Since good results were obtained with two hidden units, the number of hidden units was not varied. Equation 8.1 defines the neural networks output y in terms of the neural network inputs x_i and neural network weights, w_{ij}, W_i, θ_i and Θ:

$$y = \sum_{i=1}^{2} W_i \tanh(\sum_{j=1}^{2} w_{ij} x_j - \theta_i) - \Theta \tag{8.1}$$

The inputs are the switch loading and the traffic mix. The *switch loading* is defined to be the number of occupied trunks attached to the switch divided by the total trunks attached to that switch. The *traffic mix* is defined to be the number of direct routed calls on the trunk group divided by the total number of calls on the trunk group. This second parameter was used experimentally. The results indicate it does not have a strong influence on the value of the trunk reservation parameter chosen and could be left out. See Figure 8.5.

The neural network will output fractional values for trunk reservation. These are interpreted as follows. If the trunk reservation parameter from the neural network is 0.3, then 0.3 of the time a trunk reservation parameter of 1 is used, and the remainder of the

time a trunk reservation parameter of 0 is used.

Each simulation cycle represents a 3000 minute or 50 hour traffic simulation. This generates about 18000 test cases and 18000 training cases. Each training case or test case fell into one of the following classes:

- A direct routed call is blocked because the trunk reservation parameter is not high enough. The training data contains a new higher value of trunk reservation.

- An alternate routed call is blocked, indicating that the trunk reservation parameter is too high. The training data contains a new lower value of trunk reservation.

- To keep the neural network output steady in regions of the input space that rarely experience blocking, 1% of the simulated call arrivals are used as training data, keeping the value of the trunk reservation parameter unaltered.

Quickprop [Fah88] is then run for 200 iterations to tune the neural network weights. About 10 such simulation cycles are necessary to get good results. See Figure 8.6.

8.5 Traffic Mix

For the traffic simulation, a mixture of light (4.4 Erlangs), medium (6.7 Erlangs) and heavy (8.9 Erlangs) traffic is used. The simulated network has 10 nodes and 10 trunks between each node. Because light traffic occurs far more often in real life than medium or heavy, a weight of 0.8 is given to the light, 0.15 to the medium and 0.05 to the heavy traffic for the purpose of computing blocking.

The light, medium or heavy traffic level is indirectly input to the neural network through the switch load input variable. As the traffic level increases, the expected switch load will increase also.

It is important to note that online training in a "live" network could be substituted for the traffic simulations. Once enough new training cases were available, the neural network weights could be tuned. In this way the neural network could continually adjust itself to the traffic mixes found in the real network.

	Blocking	Standard Deviation
Neural Network (train)	0.0129164	0.0000348
Fixed Res of 0 (train)	0.0148592	0.0000568
Fixed Res of 3 (train)	0.0158059	0.0000341
Neural Network (test)	0.0129246	0.0000368
Fixed Res of 0 (test)	0.0148622	0.0000477
Fixed Res of 3 (test)	0.0157703	0.0000417

Table 8.1: Blocking for Neural Network and Fixed Reservation Parameters

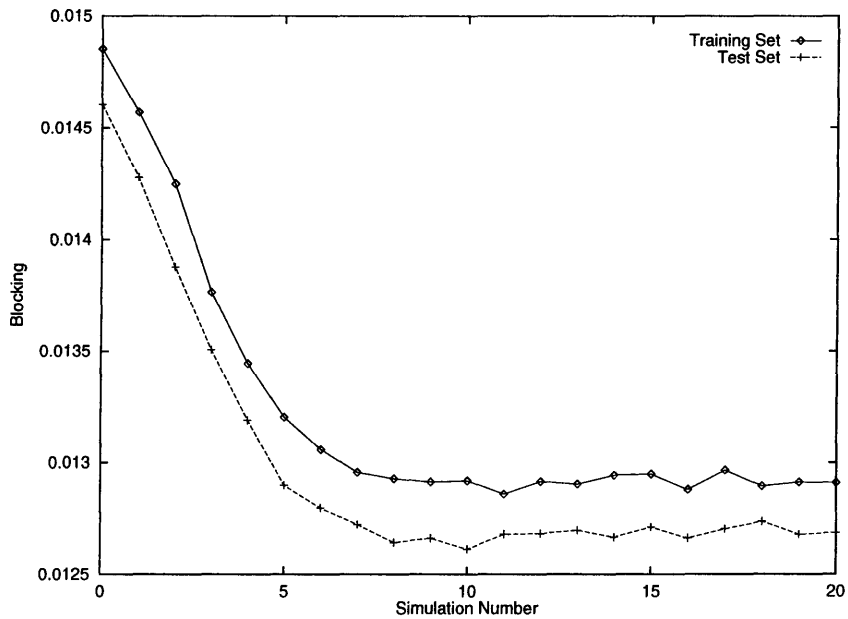

Figure 8.6: Learning the Trunk Reservation Problem

8.6 Results

Figure 8.6 shows how the training and test error decrease with the number of simulation iterations. The results show the averages over 10 runs with different initial random seeds for the traffic simulator. The random nature of the telephone traffic being simulated is the main reason for the difference between the test and training curves. With longer simulation times, the performance on the training set should surpass the performance on the test set, as one would intuitively expect.

The results are given in tabular form in Table 8.1. The results indicate a 13% decrease in blocking compared to not using any trunk reservation. An even greater decrease is found compared to using a value of 3 for the trunk reservation parameter.

8.7 Drawbacks of Algorithm

The algorithm described in Section 8.4 does not necessarily minimize blocking. Indeed a simple counting argument can be used to show that the algorithm performs badly at high traffic levels.

To see this, consider the case where the traffic load is extremely high. For example, the offered traffic may be 10 times the traffic that the network is able to carry. In this case, the majority of calls will be blocked. Each blocked call results in one training case for increased trunk reservation and four training cases for reduced trunk reservation using the algorithm described in Section 8.4. The four cases for reduced trunk reservation are due to call attempts on the two legs of the fixed alternate and the NOAA suggested alternate. The traffic simulator assumes NOAA type routing as in Chapter 3. Since there are more training cases for reduction of trunk reservation than increase of trunk reservation, the trunk reservation will be reduced to zero. Clearly this is not desirable in a high traffic situation.

The following sections explore why the basic algorithm works at all for medium traffic levels, indicate the difficulty of finding the optimum trunk reservation policy and describe an improved training algorithm which exhibits good behavior at all traffic levels.

8.8 Principle of Basic Algorithm

The basic learning algorithm seeks to strike a balance between blocked direct routed calls and blocked alternate routed calls, as shown in Figure 8.7. However the optimum trunk reservation algorithm would allow more blocked alternate calls at high traffic levels, as shown in the figure.

The question arises: can the optimum trunk reservation algorithm that minimizes overall blocking be learned by measuring network blocking? In practical terms, the answer is no. The overall blocking B can be written as a function of the network topology $\mathcal{N}$, the network traffic $\mathcal{T}$, and the trunk reservation policy as expressed by a set of neural network weights $\mathcal{W}$. The network topology $\mathcal{N}$ is a list of nodes and links along with a specification of link sizes. The network traffic $\mathcal{T}$ is a list of offered traffic values between node pairs.

$$B = f(\mathcal{N}, \mathcal{T}, \mathcal{W}) \tag{8.2}$$

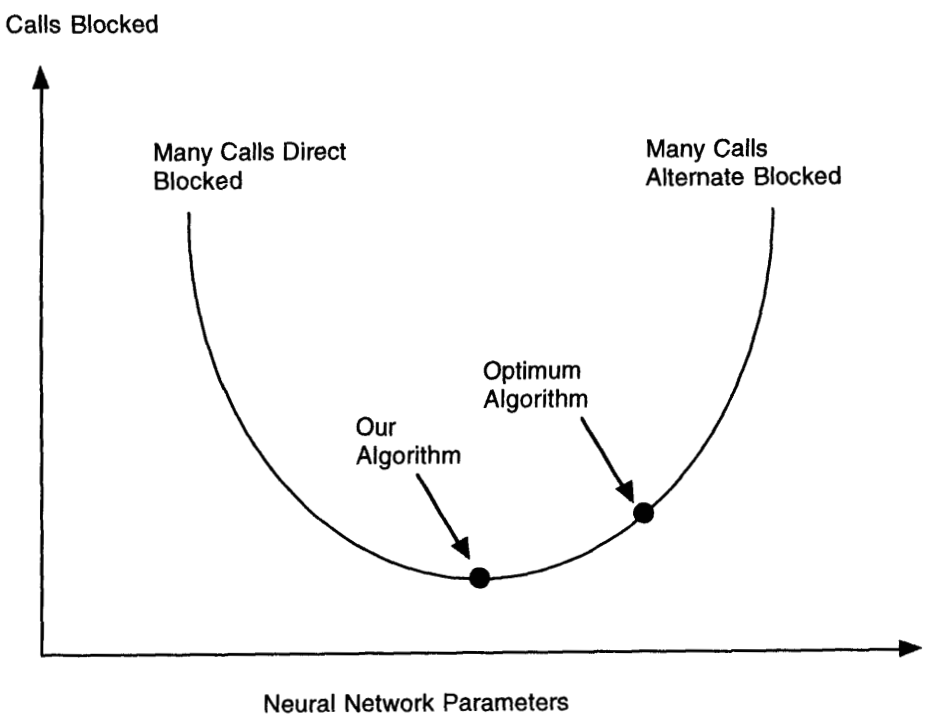

Figure 8.7: Principle of Learning Algorithm

In real life, the network topology changes from day to day and the network traffic from hour to hour. The traffic never exactly repeats itself. Thus it is difficult to gauge the change in blocking produced by a change in weights $\mathcal{W}$ when it is so strongly affected by the changing network topology and the changing traffic. This makes learning the optimum trunk reservation policy by noting changes in blocking and correlating this with changes in the weights extremely difficult.

However the basic learning algorithm can be improved upon by studying more closely the process of trunk reservation.

8.9 Trunk Reservation Revisited

Consider a trunk group of size N with a trunk reservation parameter R of 1. When there are less than $N-1$ trunks occupied or exactly N trunks occupied, the trunk reservation parameter does not influence network behavior.

When there are exactly $N-1$ trunks occupied and an alternate routed call arrives, trunk reservation causes this alternate routed call to be blocked. The expectation is that a direct routed call will arrive shortly that can make better use of the free trunk.

From this viewpoint, trunk reservation can be looked upon as a bet. When an alternate routed call is rejected, the network loses revenue, but the bet is that a direct routed call will arrive within a short length of time that will produce more revenue, because it makes more efficient use of network resources. The longer it takes for the direct routed call to arrive, the bigger the revenue loss from rejecting the alternate routed call. This is illustrated in Figure 8.8.

If it is assumed that the alternate routed call has a holding time of T_h, and the direct routed call has twice the revenue generating potential of an alternate routed call, then the point when the bet has been lost is about $2/3T_h$ after the alternate routed call has been rejected.

8.10 Improved Algorithm

The improved algorithm generates test cases and training cases as follows. For each alternate routed call that encounters trunk reservation, a test is carried out. The neural network inputs are noted at the time the call arrives. Then a count is carried out of the

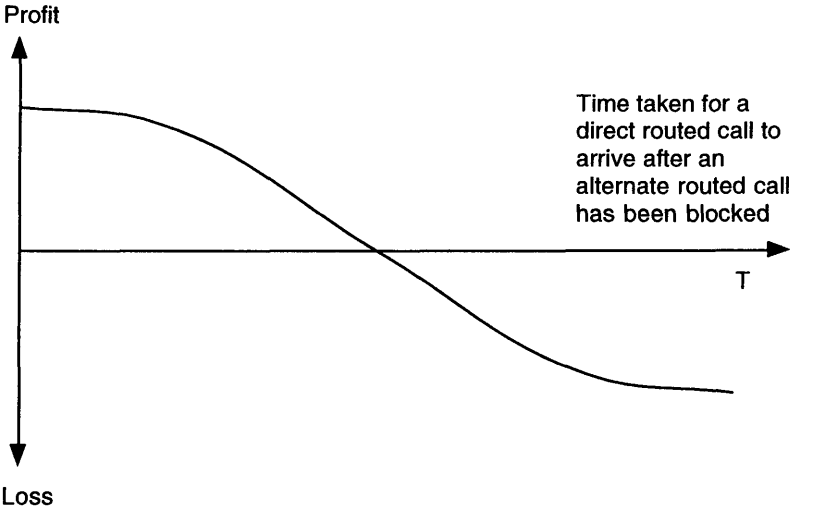

Figure 8.8: Profit Following Trunk Reservation Decision

number of direct routed calls that arrive subsequent to the alternate routed call and get blocked. The counting period is two thirds of a holding time. The number of blocked direct routed calls is written to a training file as the desired value of trunk reservation on the route.

Since the *traffic mix* input variable has little effect on the trunk reservation, this input variable for the neural network is removed and a variable called the *alternate success rate* is substituted. The alternate success rate is an exponential moving average of the rate at which calls are successfully carried after failing to get a trunk on the direct route. The rationale is that if there is spare capacity on the alternate routes, then there is less of a need to provide trunk reservation. The results show it impacts the level of suggested trunk reservation in the expected way.

The interpretation of fractional values of the neural network output is also changed. If the output of the network is 0.1, this indicates that the "bet" would be lost 9 times out of 10. Given this interpretation, a rounding approach is taken, i.e. 0.1 is rounded to a trunk reservation of 0. Similarly a neural network output of 0.9 is rounded to a trunk reservation value of 1. In other words rounding to the nearest integral value is used or,

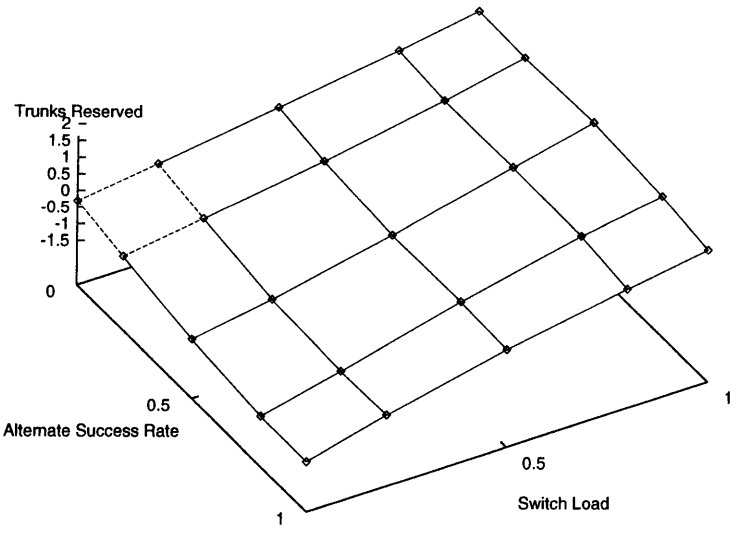

Figure 8.9: Neural Network Output

the bet is taken if the chance of winning is over 50%.

The results for the revised algorithm are given in Table 8.2. The function learned by the neural network is given in Figure 8.9. A new entry has been added to Table 8.2 giving the results for a trunk reservation parameter of 1.

While the old algorithm is beaten by a fixed trunk reservation of 1 trunk, the new algorithm beats any fixed level of trunk reservation.

For comparison, the table also shows the performance of the neural network compared to a linear predictor with the same inputs and outputs. The neural network performs slightly better. Over 10 runs with different random seeds for the traffic simulator, the difference between the two methods is found to be non-zero at the significance level of 98%. The differences in blocking performance are given in Table 8.3. It can be concluded that the neural network would be slightly better than the linear predictor in learning this problem.

8.11 Conclusions

In this chapter, neural networks have been applied to telephone network congestion relief and done better than the conventional technique of a fixed reservation scheme. In addition, it has been shown that:

	Blocking Probability	Standard Deviation
Neural Network (train)	0.0129164	0.0000348
Linear Predictor (train)	0.0127636	0.0001319
Improved Neural Network (train)	0.0127596	0.0000317
Fixed Res of 0 (train)	0.0148592	0.0000568
Fixed Res of 1 (train)	0.0129114	0.0000369
Fixed Res of 3 (train)	0.0158059	0.0000341
Neural Network (test)	0.0129246	0.0000368
Linear Predictor (test)	0.0127756	0.0001106
Improved Neural Network (test)	0.0127517	0.0000374
Fixed Res of 0 (test)	0.0148622	0.0000477
Fixed Res of 1 (test)	0.0129276	0.0000336
Fixed Res of 3 (test)	0.0157703	0.0000417

Table 8.2: Blocking Probabilities for Improved Neural Network and Fixed Reservation Parameters

Blocking Difference
-0.0000195
-0.0000397
-0.0000544
-0.0000159
0.0000205
-0.0000563
-0.0000214
-0.0000293
-0.0000306
0.0000077

Table 8.3: Blocking Difference for Neural Network and Linear Predictor as the Initial Random Seed for Traffic Simulation is Varied. Negative Numbers Indicate the Neural Network is Better.

- The neural network can be applied to a difficult indirect learning problem.
- In the test network, a large decrease in average blocking is seen.
- By substituting live traffic for the simulations in this chapter, the neural network can learn in real-time.

It should be possible to extend the results in this chapter to networks containing trunk groups of different sizes. This would require an extra input to the neural network containing this information, and longer simulation runs to make sure all parts of the input space are well represented.

Chapter 9
Time Series Prediction of Telephone Traffic Occupancy

9.1 Introduction

In this chapter, a number of methods for time series prediction of the occupancy statistic are considered. The advantages and disadvantages of each are discussed. The results are given in Table 9.4 on page 103 and Table 9.5 on page 103.

Although the problems described in this chapter are concerned with monitoring telephony traffic, the techniques should be applicable, with little modification, to the monitoring of any large network. For example, rather than monitor trunk group occupancy, the network manager may be monitoring link throughput, but the same analysis techniques should apply.

It would be useful to be able to predict telephone traffic occupancy on trunk groups prior to putting reroute controls in the network for two reasons. First, it may be possible to monitor occupancy on all the trunk groups in the network and implement a reroute control when the occupancy trend shows that a route is in danger of overflowing. Second, there is a delay between when capacity information is obtained for a trunk group and when a control is put in the network. This data gathering delay is of the order of two or three minutes. If NOAA's recommendations are made on the basis of predicted occupancy rather than occupancy, it may result in a need for fewer adjustments to the control.

For this study, data was gathered while NOAA was running, logged to a file and examined in Caltech with a view to carrying out time series prediction.

One of the main motivations in beginning the study is to see whether the use of neural networks would give a significant advantage in carrying out time series prediction because of their ability to realize non-linear functions of their inputs.

9.2 Data Set

Definition 3 *Occupancy is a moving average of twenty samples of the number of trunks occupied on a route. The samples are taken every 30 seconds, and the result is scaled to*

be between 0 and 100%.

- Occupancy is a good indicator of the spare capacity of a trunk group.
- Occupancy is reported every 5 minutes.

Two data sets were gathered:

- A short data set with about 1500 observations of occupancy on a single trunk group with 144 trunks between Los Angeles and Pasadena over the course of a week. The data set went from Friday to Friday with some gaps. This data set was used to compare various methods for time series prediction. In all cases, the aim was to estimate y_7 given $y_1, y_2, \ldots, y_6$
- A long data set consisting of 18000 observations of occupancy on a trunk group with 360 trunks between Los Angeles and Gardena. This provides about 10 weeks worth of data. This data set was used to help model the daily profile of occupancy on a trunk group and the spike component of the occupancy statistic.

In all cases, an estimate y_7 given $y_1, y_2, \ldots, y_6$ is desired. The reason this many coefficients are chosen is that the autocorrelation function has died down to close to zero by the sixth coefficient. This is illustrated in Figure 9.2. The autocorrelation of the first order difference of occupancy was plotted, since the occupancy process does not have a zero mean. In contrast the first order difference of occupancy is very close to be a zero mean process, making the autocorrelation coefficients easier to interpret.

The occupancy for every trunk group in the network is reported every five minutes. The first 300 points of the short data set are shown in Figure 9.1. Some key features to note are:

- The traffic level varies according to time of day. The plot shows the traffic level dropping on Friday evening, rising again on Saturday morning, and remaining level during Saturday afternoon.
- Spikes may be present in the time series, e.g., close to example 60.
- The variance of the occupancy varies with the traffic level. The plot becomes more jagged as the trunk group occupancy increases.

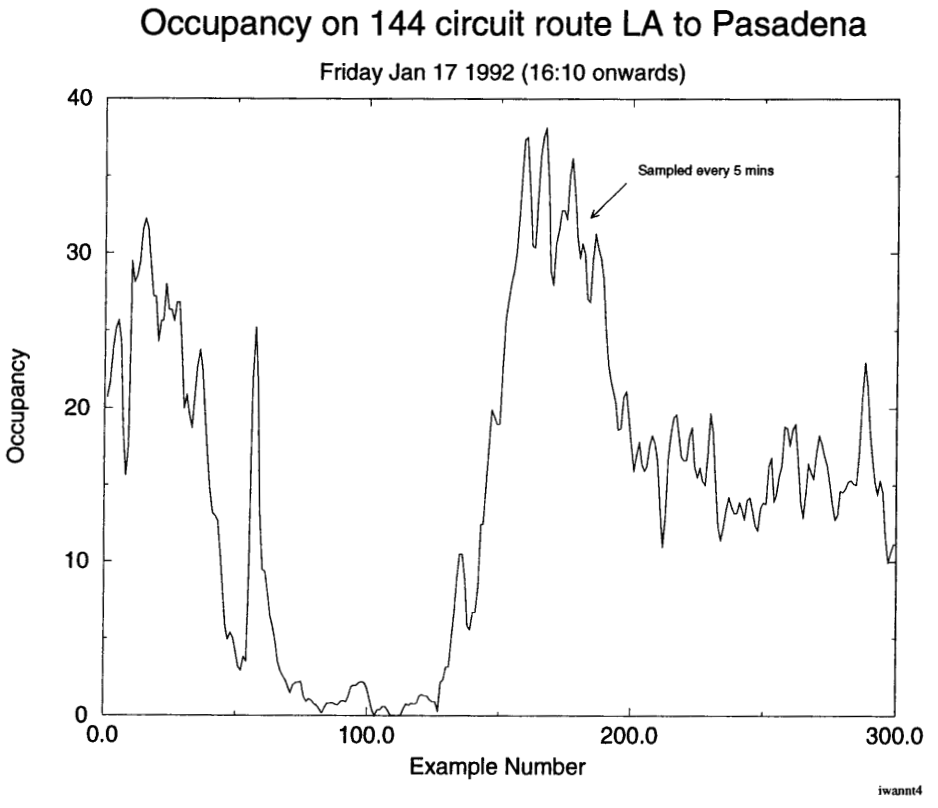

Figure 9.1: Occupancy Dataset

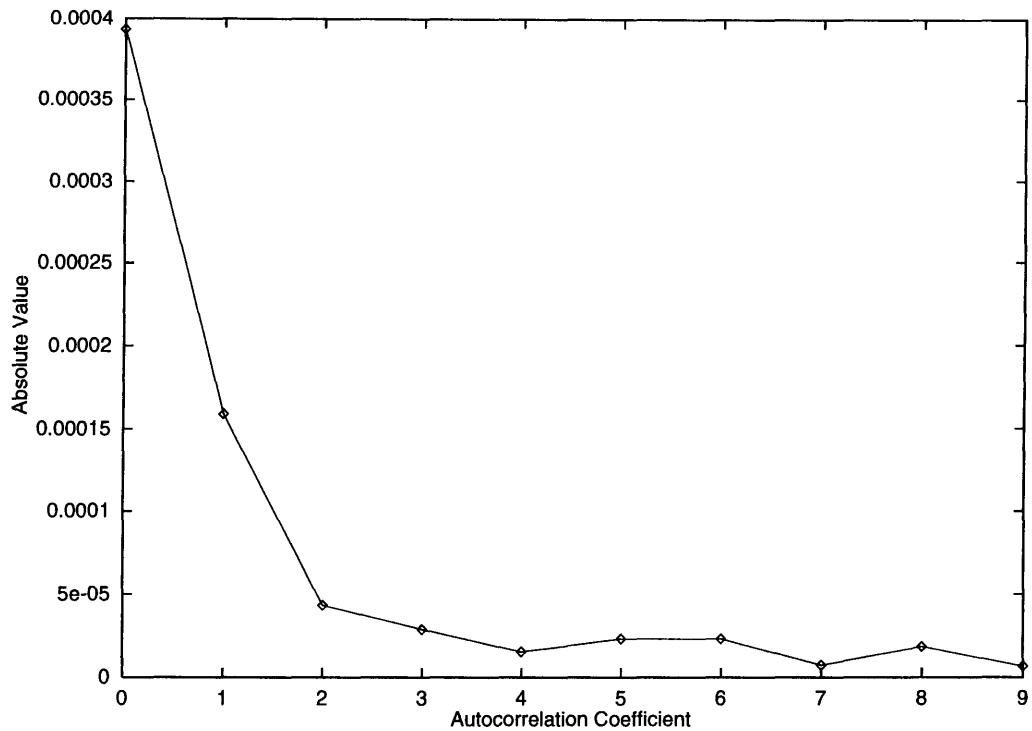

Figure 9.2: Autocorrelation of (First Order Difference of) Occupancy

9.3 Simulation

For comparison, a simulation program was written. For the simulation, memoryless arrivals and memoryless holding times are assumed. A constant traffic level is assumed, although this could have been easily changed to test more advanced models of the daily variation of occupancy or the occurrence of spikes.

A plot of the first 300 points of a simulation of 36 Erlangs of traffic is given in Figure 9.3.

The method of simulation employed is to make a list of instants, each instant corresponding to one of the following.

- A random call arrival (in fact the inter-arrival times which have a negative exponential distribution are simulated).
- A random call departure (using the fact that the holding time has a negative exponential distribution).
- A deterministic sample time, where a record is made of the number of trunks occupied.

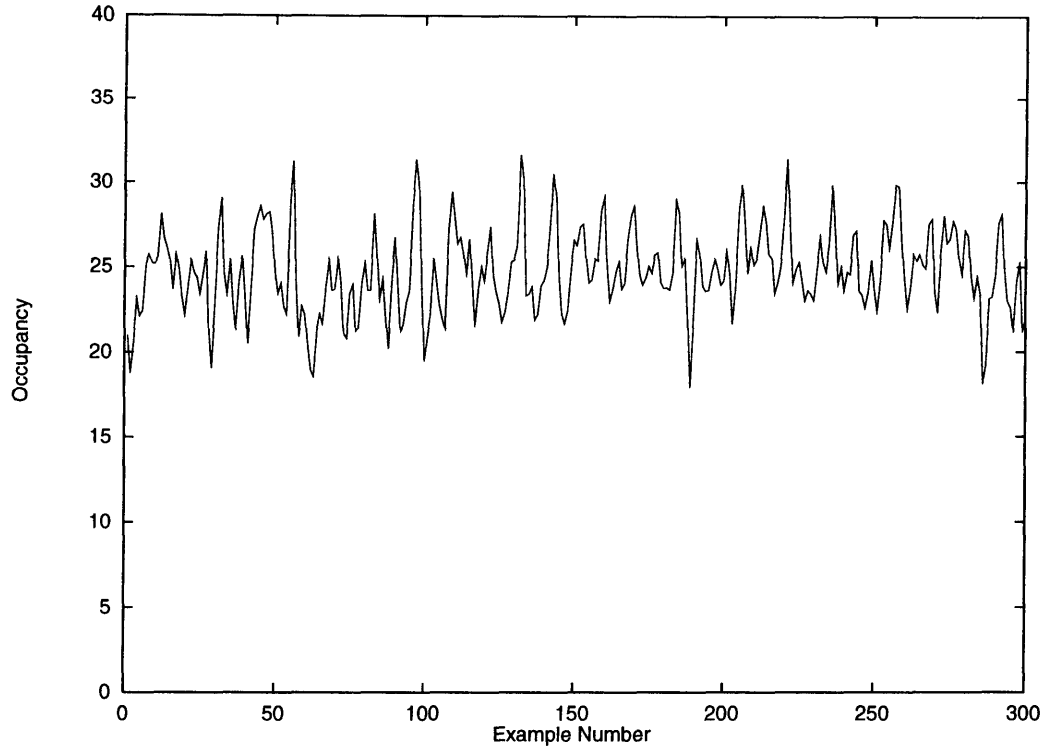

Figure 9.3: Simulation of 36 Erlangs of Traffic

The list is then sorted and processed sequentially to yield a simulation of the occupancy statistics for the trunk route.

The main difference between the simulation and the real data is that the simulation shows much less variability due to time of day or traffic spikes. However the simulation does show some variability due to the random nature of call arrivals.

9.4 Cross-validation

In testing the relative merits of prediction techniques, a distinction must be made between learning ability and generalization ability. A good prediction method will generalize well on examples that have not been seen before, by learning the underlying function without learning the associated noise.

The first step in testing a method is the division of the supplied data set into a test set and a training set. The training set is used to derive the coefficients of the model and the model is then tested on the test set. In testing two or more models, the one that does best on the test set is the better one.

There are two reasons a model may not do well on a test set.

1. The model may not be able to provide a function that closely fits the function that is to be estimated. For example, a linear model will not fit a non-linear function well across all of its input space.

2. The model may provide too rich a space of functions and return a function that implements the noise in the data set as well as the underlying function. Such a model will not generalize well.

This can be tested using cross-validation. With v-fold cross-validation and a data set of size N, v tests are carried out. Each test employs $N - N/v$ samples as the training set and the remaining N/v samples as the test set. This makes maximum use of the data set and allows us to check the significance of the results.

Strict cross-validation involves using a test set of one example and training on the remaining examples. This is done repeatedly leaving out one example at a time. In this way, maximum use is made of the data set in determining the generalization performance of the models. Strict leave-one-out cross-validation would have been better but results in a lot more computation time. A single run for the neural network took about a week on a Sun Sparc 10, and strict leave-one-out cross-validation using the same software would have taken almost a year. The back-propagation algorithm [HKP91] is the culprit for the long run times, in addition to the need to duplicate the runs to assess the affect of the random weights used for initialization.

An advantage of using cross-validation is that it provides some measure of the significance of the result. When comparing two models, one model may do better than another because (i) it more closely represents the underlying function or (ii) by chance it more closely represents the test set noise terms. By repeating the experiment using a second test set, an indication of the significance of the second factor is obtained. The more repetitions of the generalization test are carried out, the more significant the results for generalization capability.

For this study, the examples are first shuffled to randomize the order and then 50-fold cross-validation is carried out. It is found to be important to keep the ordering of examples the same among the different methods.

9.5 Notation

The time series prediction problem can be generalized to one of function approximation. Given a number of examples of a function plus noise, a function is desired that closely matches the training data and generalizes well on any new data that is supplied. It is easily shown that the function that minimizes the mean square error in the long term is the expected value function.

Consider a typical time series $x_1, x_2, \ldots, x_n$. An estimate $\hat{x}_{n+1} = f(x_1, x_2, \ldots, x_n)$ of the value of the time series at the next time step is desired by choosing f to minimize a loss function $\mathcal{L} = \mathrm{E}(x_{n+1} - f)^2$. Differentiating with respect to f and setting the result to zero gives

$$f = \mathrm{E}(x_{n+1}|x_1, x_2, \ldots, x_n) \tag{9.1}$$

Differentiating again with respect to f shows that this is indeed a minimum.

From this result, it is seen that time series prediction is a special case of the more general function approximation problem where the function that must be approximated is the expected value function. This observation allows the use of many results that have been derived over recent years about function approximation [Bar93, Cyb89, HSW89, UM91]. This is of particular importance for neural networks.

9.6 Motivation

An example is used to motivate the use of non-linear predictors. Consider a stationary discrete time random process $x_1, x_2, \ldots, x_n$ with the following properties $\forall n$:

1. x_n is a first-order Markov process, i.e., the value of x_{n+1} depends on the value of x_n only and no extra information is provided by observations $x_1, x_2, \ldots, x_{n-1}$.
2. $p(x_n) = \frac{T}{A}\frac{e^{-\frac{x_n}{T}}}{(1+\epsilon x_n)}$ with a normalizing constant $A = \frac{T}{\epsilon}e^{\frac{1}{\epsilon T}}\mathrm{Ei}(\frac{1}{\epsilon T})$
3. $p(x_n, x_{n-1}) = \frac{1}{A}e^{-\frac{(x_n + x_{n-1} + \epsilon x_n x_{n-1})}{T}}$

The optimum predictor is

$$E(x_{n+1}|x_n) = \frac{T}{(1+\epsilon x_n)} \tag{9.2}$$

which is non-linear in x_n.

9.7 Linear Predictor

9.7.1 Introduction

In the past, the linear predictor has been the mainstay of statistical prediction techniques and regression analysis [Ros87, Chr91, WG94, BJR94, DH87, Pan91]. A regression model attempts to predict the value of a dependent variable from one or more independent variables. If the dependent variable is a continuous function of the independent variables and if the input space for the independent variables is small enough, then a linear regression model can give good results.

9.7.2 State of the Art

The field of regression analysis is well developed [Ros87]. Least squares estimates of the regression parameters for a regression model can be obtained. These estimates minimize the sum of the squares of the residuals with respect to the regression parameters. An assumption that is usually made is that the regression errors are independent, identically distributed normal random variables with mean 0 and some variance σ^2. Given this assumption, the distribution of the regression parameters can be found to be a normal distribution with known mean and variance. Confidence intervals can be derived for the regression parameters. Similarly confidence intervals for the predicted values can also be obtained. An index of fit which is a measure between 0 and 1 is defined which measures how good the regression fit is.

In the seventies, Box and Jenkins wrote a definitive text book on the subject of time series prediction [BJR94]. A procedure for developing models of time series is proposed which put an emphasis on parsimony of model parameters. The analysis of correlation coefficients of the data is shown to be an important step in the selection of an appropriate model. Moving average and autoregressive stochastic models are described. The use of differencing techniques to deal with non-stationarities is described.

In the signal analysis arena, the subject of modeling of time series is equally important and there has been much work on the modeling of associated stochastic processes [Hay91, PP80, Lat89, Sch87]. For stationary processes the linear predictor coefficients are related to the autocorrelation coefficients by the Wiener-Hopf equations [Hay91]. The Wiener-Hopf equations are not used here, but it is interesting to note that they are very similar to

the equations used in statistical regression analysis. In this section, a statistical regression analysis approach is taken.

It is known that for a process that has a multivariate Gaussian distribution the Linear Predictor (LP) is the best predictor[Chr91]. Since the Poisson or telephone traffic distribution looks like a Gaussian for large traffic levels, and the collected traffic data shows no traffic being blocked which would cause the Gaussian property to be lost, and the moving average operation used to calculate occupancy from sampled telephone traffic would not cause the Gaussian property to be lost, it is not surprising that the linear predictor did well. The other nonlinear methods can only give small percentage improvements.

9.7.3 Training a Linear Predictor

The procedure for training a linear predictor to minimize error on the training set in a least squared sense is well known and involves the calculation of the pseudo-inverse matrix. To find the LP weights, a matrix equation can be written for the residual errors r_i of an approximate solution of the prediction problem in terms of the occupancy observations y_i as follows:

$$\begin{bmatrix} r_1 \\ r_2 \\ r_3 \\ r_4 \\ \vdots \\ r_{1500} \end{bmatrix} = \begin{bmatrix} y_1 & y_2 & \cdots & y_6 \\ y_2 & y_3 & \cdots & y_7 \\ y_3 & y_4 & \cdots & y_8 \\ y_4 & y_5 & \cdots & y_9 \\ \vdots & \vdots & \ddots & \vdots \\ y_{1500} & y_{1501} & \cdots & y_{1506} \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_6 \end{bmatrix} - \begin{bmatrix} y_7 \\ y_8 \\ y_9 \\ y_{10} \\ \vdots \\ y_{1507} \end{bmatrix} \tag{9.3}$$

or in short,

$$r = Aw - b \tag{9.4}$$

Then $r^T r$ (the total square residual error) is

$$r^T r = (Aw - b)^T (Aw - b) \tag{9.5}$$

Now, differentiating $r^T r$ with respect to w and setting the result to zero to minimize the error with respect to the weight vector w gives

$$A^T A w = A^T b \tag{9.6}$$

and hence

$$w = (A^T A)^{-1} (A^T b) \tag{9.7}$$

Thus the LP coefficients can be obtained by a simple 6x6 matrix inversion.

9.8 Non-Linear Predictor

Non-linear prediction allows the use of $y_i y_j$ cross-product terms in the A matrix of Section 9.7. For example:

$$\hat{y}_7 = \sum_{i=1}^{6} w_i y_i + \sum_{i=1}^{6} \sum_{j=i}^{6} w_{ij} y_i y_j \tag{9.8}$$

Taking a $y_6 y_6$ cross-product as an example, residual errors r_i now become:

$$\begin{bmatrix} r_1 \\ r_2 \\ r_3 \\ r_4 \\ \vdots \\ r_{1500} \end{bmatrix} = \begin{bmatrix} y_1 & y_2 & \cdots & y_6 & y_6 y_6 \\ y_2 & y_3 & \cdots & y_7 & y_7 y_7 \\ y_3 & y_4 & \cdots & y_8 & y_8 y_8 \\ y_4 & y_5 & \cdots & y_9 & y_9 y_9 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ y_{1500} & y_{1501} & \cdots & y_{1506} & y_{1506} y_{1506} \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_7 \end{bmatrix} - \begin{bmatrix} y_7 \\ y_8 \\ y_9 \\ y_{10} \\ \vdots \\ y_{1507} \end{bmatrix} \tag{9.9}$$

or

$$r = Aw - b \tag{9.10}$$

As in Section 9.7, matrix inversion is used to give the predictor coefficients:

$$w = (A^T A)^{-1} (A^T b) \tag{9.11}$$

Thus the LP coefficients can be obtained by a simple 7x7 matrix inversion.

This method is also known as polynomial regression [Ros87]. It is one of many non-linear techniques studied in this chapter. It shares with the other techniques a non-parsimony of parameters. In other words, for N inputs, there are $O(N^2)$ possible second order crossproducts and $O(N^3)$ possible third order cross products. The more parameters in the model, the more data that is necessary to correctly estimate them. If not enough data is available the generalization results will not be good.

A slight improvement in prediction performance is obtained. Trial and error (see Table 9.1) shows that the best results are obtained with pure squares of the input terms. Overall results are given in Section 9.16 on page 104.

9.9 Neural Network

Comparisons of neural networks and linear predictors show that neural network sometimes can give better results [TdAF91, SP90]. However the data sets used are not particularly long, so the statistical significance of these comparisons may be questionable.

Error	Crossproduct Terms
0.016283	y_4y_4 y_5y_5 y_6y_6 y_4y_5 y_5y_6
0.016256	y_5y_5 y_6y_6 y_5y_6
0.016215	Standard Linear Predictor
0.016190	y_4y_5 y_5y_6
0.016188	y_5y_6 y_6y_6
0.016187	y_1y_1 y_2y_2 y_3y_3 y_4y_4 y_5y_5 y_6y_6
0.016182	y_5y_5 y_6y_6
0.016172	y_6y_5
0.016169	y_4y_4 y_5y_5 y_6y_6
0.016167	y_6y_6

Table 9.1: Scaled RMS Prediction Error for Various Cross-product Terms

Number of Hidden Units	Error
12	0.0191531
10	0.0191281
6	0.0191008
4	0.0189924
3	0.0194110

Table 9.2: Neural Network Test Set Prediction Error for Varying Numbers of Hidden Units

Moody in [UM91] gives learning limits for neural networks. See Section 9.11 for some details. A key point in his paper is that too many hidden units combined with a low value for weight regularization will produce an increase in generalization error. Weight regularization is the step of adding an error term to the function to be learned. This error term penalizes neural networks that have large weight values, or non-smooth outputs. The aim is to avoid learning that noise in the training data set. From Moody's work it can be deduced that there is an optimum number of hidden units for a given learning problem.

In these studies, a feed-forward neural network with a single hidden layer is used. Quickprop[Fah88] is used for training as it has faster convergence than standard back-prop and is freely available on the Internet. For the neural network, trial and error shows that four hidden units and a linear output unit gives best results (See Table 9.2). The linear output unit is used to aid function fitting. This architecture is fixed prior to training. See Section 9.16 on page 104 for results.

A plot of hidden unit activations gives valuable insight into the features of the data set. There are four hidden units. One of the hidden units reacts strongly to the overall traffic level. One of the units reacts strongly to rate of change of traffic level while the

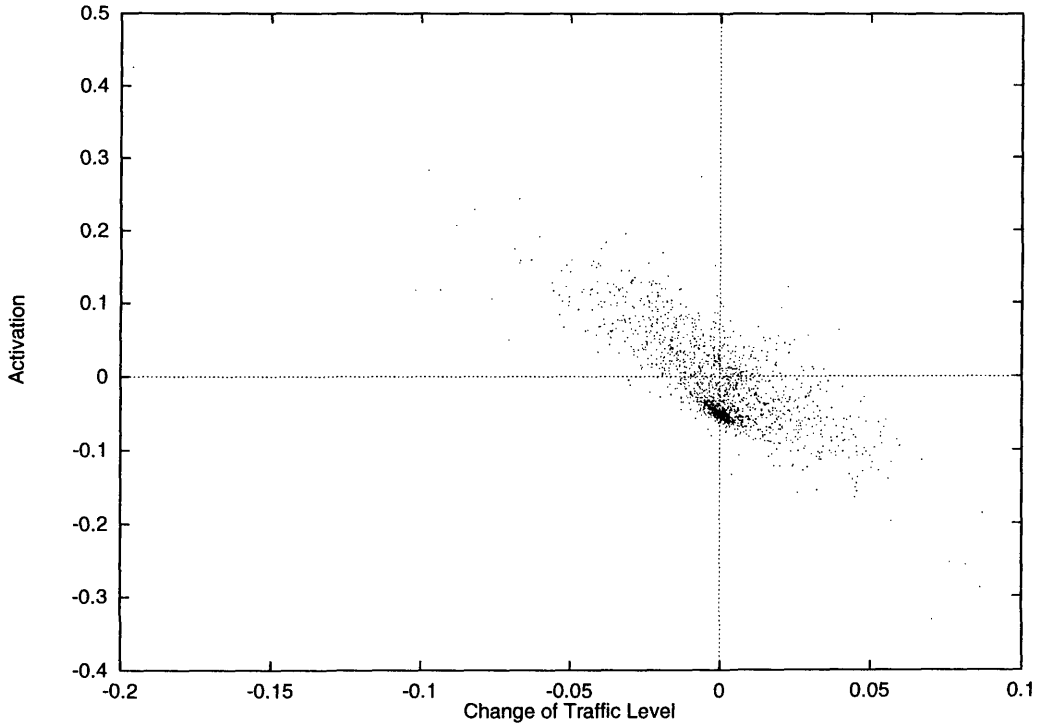

Figure 9.4: Activation of Hidden Unit #3

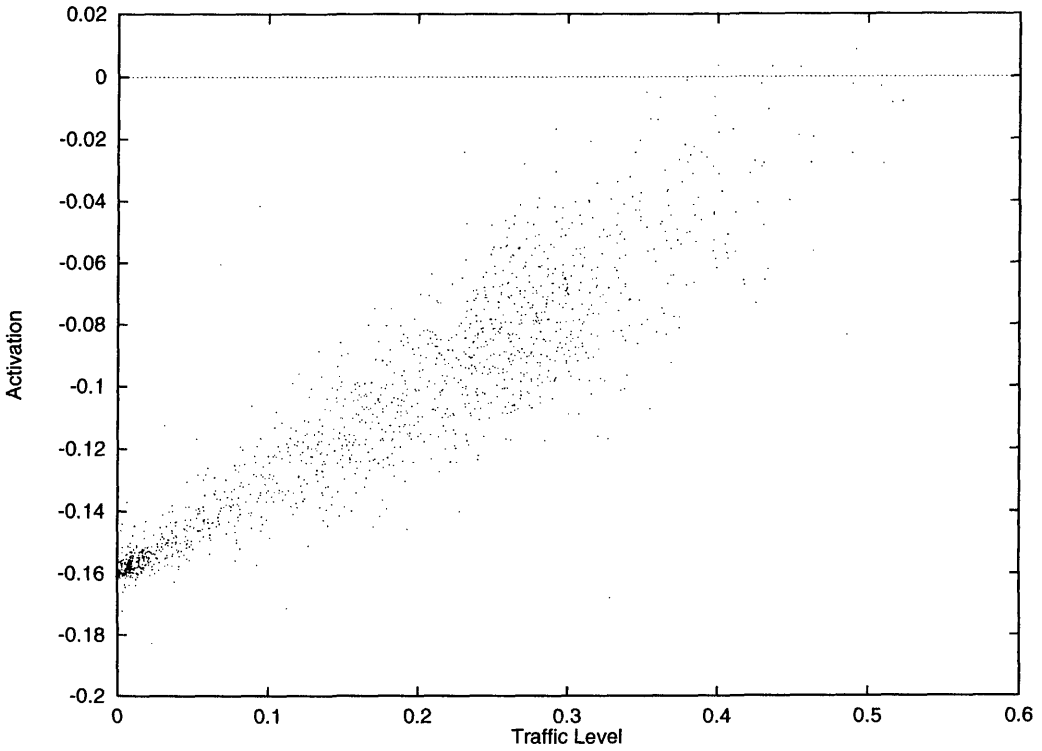

Figure 9.5: Activation of Hidden Unit #4

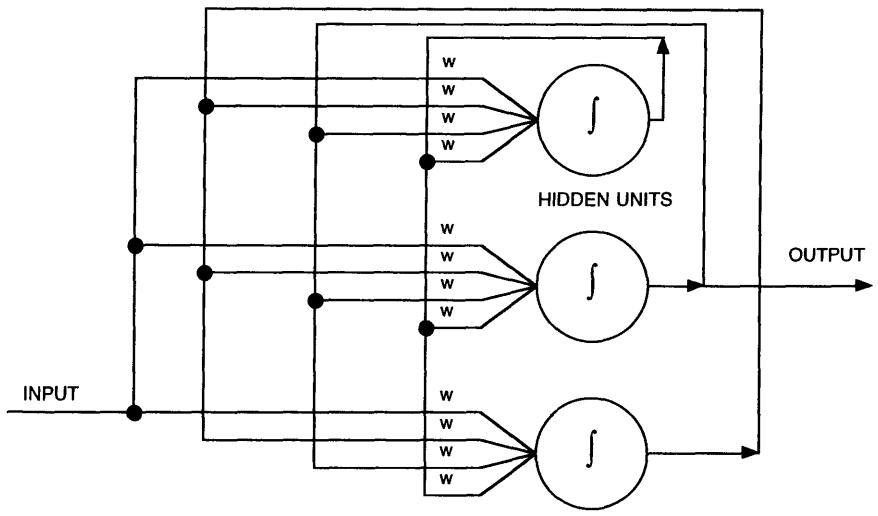

Figure 9.6: Recurrent Neural Network (Cell Threshold Weights not Shown)

other two reacts strongly to the rate of rate of change of traffic level. Examples of two of the plots are given in Figures 9.4 and 9.5. It is possible that the neural network is constructing a Taylor Series of the function of be approximated.

9.10 Recurrent Neural Networks

Recurrent neural networks allow the use of feedback in addition to feedforward connections [WZ89, QSM92, Pin89, Pea89]. In this way, the network retains some state information between the presentation of examples. Recurrent neural networks are more difficult to train, but some successes have been reported for the time series prediction problem [CMA94].

The architecture of the recurrent neural network is somewhat different to the architecture of the other models in this chapter. Instead of 6 inputs, there is only one (See Figure 9.6). Recurrent neural networks are assumed to use their internal hidden units to hold state information, similar to the way finite state automata work. So there should be less need to present the previous inputs at the same time as the present input. Instead the recurrent neural network should learn the input features that are significant, and store some representation of these states in the hidden units to aid in prediction.

Since it is assumed that the previous six examples contain all the necessary information for making a prediction, this state information may not buy us any additional prediction power. See Section 9.15 for details on how time of day information can be more easily incorporated in the prediction process.

Using a recurrent neural network on the long data set, with one input, one output, and a variable number of hidden units, and using the training techniques described in [WZ89], a scaled prediction error of 13952 is the best that could be achieved, which is worse than the performance obtained by the linear predictor. These results suggest that the function space of this recurrent neural network is not rich enough to model the expected value function that needs to be approximated by the time series predictor.

9.11 Learning Limits for Neural Networks

J. E. Moody carries out an analysis of generalization and regularization in non-linear learning systems[UM91].

Assume a set of $2n$ real valued input/output data pairs are given and a function to fit the data is to be estimated. The data is split equally into a test set and a training set. The noise is i.i.d. with mean zero and finite variance σ^2. The noise is not necessarily Gaussian.

For a linear predictor, references are given to the following result for the MSE e_{test}:

$$\mathrm{E}(e_{test}) \approx \mathrm{E}(e_{train}) + 2\sigma^2 \frac{p}{n} \tag{9.12}$$

where p is the number of parameters (weights) being estimated.

For a neural network, a new result correct to second order is given:

$$\mathrm{E}(e_{test}) \approx \mathrm{E}(e_{train}) + 2\sigma^2 \frac{p_{eff}}{n} \tag{9.13}$$

where p_{eff} is a complicated function of various Jacobians. However for a locally linear model, p_{eff} is a decreasing function of λ, the weight decay parameter for the neural network with $p_{eff}(\lambda = 0) = p$. The form of Equation 9.13 suggests that (i) there may be a trade off between having too many hidden units which will cause the second term in equation 9.13 to rise or too few hidden units which will cause the first term in equation 9.13 to rise and (ii) given enough data, there will be an arbitrarily small difference between test set performance and training set performance.

9.12 Local Approximation

Another name for the Local Approximation technique might be "History will Repeat." The idea is straightforward. A training set and a test set are given. To carry out a

prediction for any particular test set example, find examples in the training set which most closely resemble it. Then train a linear predictor on that subset of the training set and use it for prediction. This procedure is repeated for each example in the test set.

The algorithm is more precisely specified as follows:

Given:

S_0 , a training set consisting of n_0 examples,

S_1 , a test set consisting of n_1 examples,

each training set example is of the form $\{y'_1, y'_2, y'_3, y'_4, y'_5, y'_6, y'_7\}$. Each test set example is of the form $\{y_1, y_2, y_3, y_4, y_5, y_6, y_7\}$ where y_7 is to be predicted from $y_1 \ldots y_6$.

A forecast is carried out for each of the n_1 examples based on a model derived from the training data S_0. Instead of using a single model for all the test examples, as is usually the case, a different model is derived for each test example. The model is a linear predictor which is derived as follows. Choose a neighborhood $N_a < n_0$. Pick from the training set the N_a examples which minimize the Euclidean distance d, where

$$d = \sqrt{\sum_{i=1}^{6} (y_i - y'_i)^2} \tag{9.14}$$

Here y' indicates a number from the training set, and y indicates the current example from the test set.

Finding these examples can be done in one pass through the training set and a linear predictor trained using only those examples, as detailed in Section 9.7.

Figure 9.7 shows how the generalization error varies as the neighborhood, N_a, for the local approximation technique varies.

[FS87] reports excellent results for this method on chaotic series using a small neighborhood and noise free measurements to a high precision.

The disadvantage of this method is that the most accurate predictions require keeping on-line a large number of training examples. In this case, if traffic spikes occur infrequently, a large volume of data is needed for each trunk route to be sure that the spike is captured. In contrast, the neural network model is more attractive, requiring a much smaller amount of information to be stored (i.e. the weights) to characterize the function to be estimated.

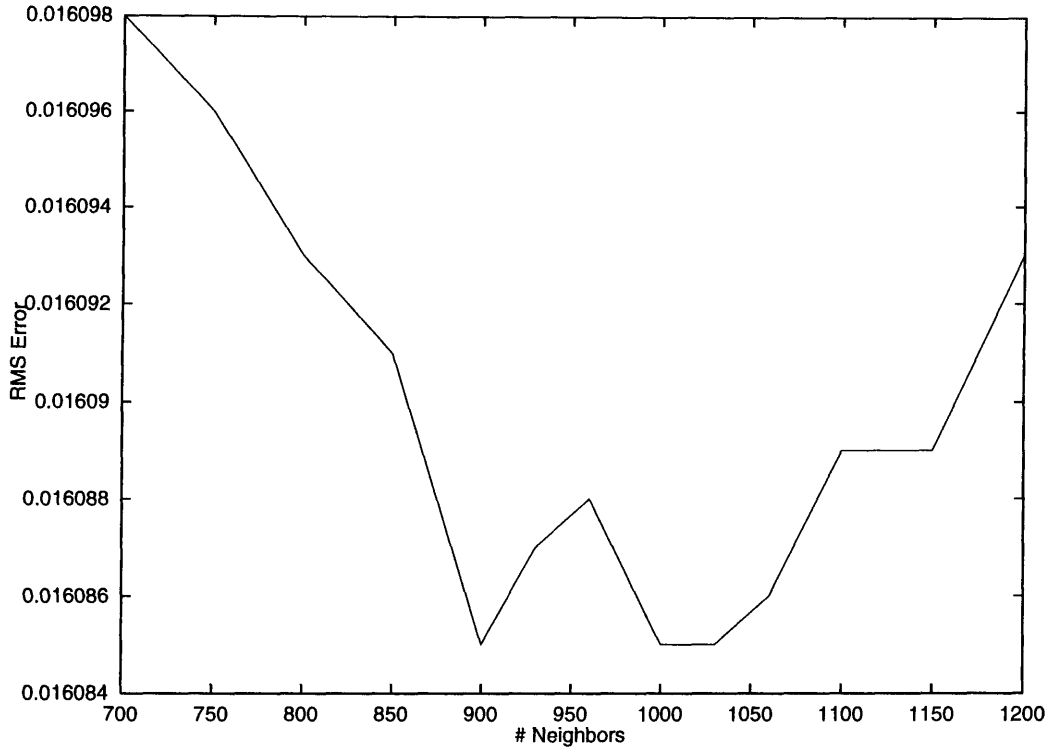

Figure 9.7: Minimum Error for Local Approximation

9.13 Hidden Markov Model

Hidden Markov models (HMMs) are popular for word classification in automatic speech recognition[LRS83, RLS83].

The HMM is defined by a set of states, S, and two matrices A_{ij} and B_{oi}. A_{ij} is the probability of a transition to state i from state j, and B_{oi} is the probability of observing o given that you are in state i. This is illustrated in Fig. 9.9. The A_{ij} matrix gives rise to the *Markov* in the name, since the next state depends on the present state as in a regular first-order Markov model. The B_{oi} matrix gives rise to the *Hidden* in the name, since the states are not directly observable.

The Baum-Welsh algorithm can be used to learn the A and B matrices[LRS83]. A different algorithm, the Baum Backward-Forward algorithm can be used to derive the probabilities of the system states given the observations [LRS83]. In this way, the HMM can be used as a classifier.

In this case, a HMM is used to classify the trunk group as being in one of two modes/states: (i) traffic varying a lot as during a traffic spike or (ii) traffic behaving

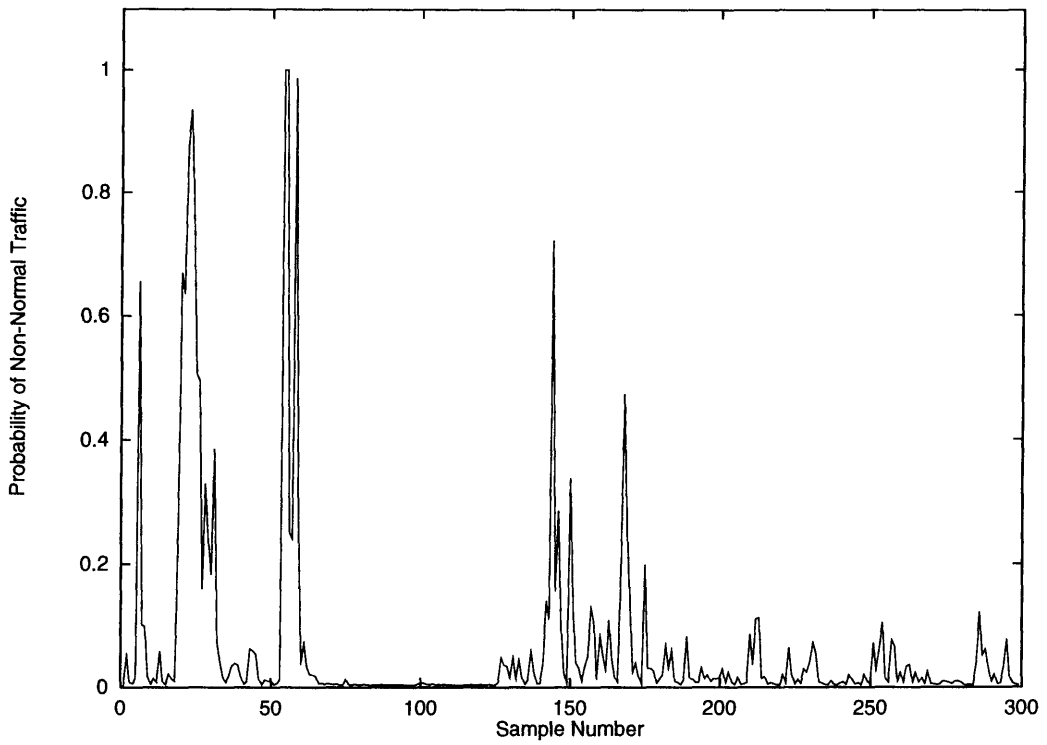

Figure 9.8: Hidden Markov Model State

normally. This classification is used to choose between two linear predictors for training purposes as shown in Fig. 9.10. For testing purposes, as shown in Fig. 9.11, the HMM output probability for state 1 (p_1) is used to combine the estimates from the two linear predictors (LP_i) to yield the occupancy estimate $\hat{y}$ as follows:

$$\hat{y} = p_1 * \mathrm{LP}_1 + (1.0 - p_1) * \mathrm{LP}_2 \tag{9.15}$$

The A-matrix is learned from the data using the Baum-Welsh algorithm. The observation, $o(n)$, used as input to the HMM model is the prediction error from a 5-input linear predictor in predicting the most recent occupancy reading.

A plot of the probability of traffic not being "normal" traffic is given in Figure 9.8. Compare this to the original time series shown in Figure 9.1. The traffic spike close to example 60 can be clearly seen. Also the decline in traffic close to example 25 can be clearly seen.

For the state transition probability matrix, A:

$$\begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \tag{9.16}$$

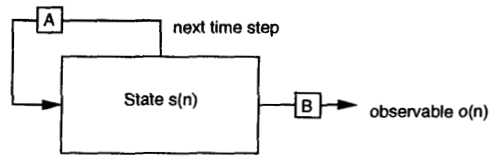

Figure 9.9: Hidden Markov Model

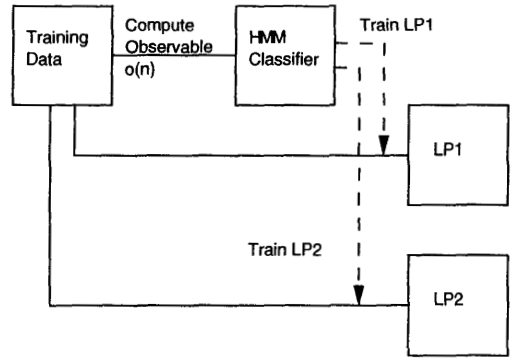

Figure 9.10: Training a Dual Linear Predictor

Then for time-step n:

$$p_1^n = p(1|o_n) \times (a_{11}p_1^{n-1} + a_{12}p_2^{n-1}) \tag{9.17}$$

$$p_2^n = p(2|o_n) \times (a_{21}p_1^{n-1} + a_{22}p_2^{n-1}) \tag{9.18}$$

where p_i^n is the estimate of the relative probability of being in state i at timestep n.

An assumption of a Gaussian distribution of prediction error is used to generate the output matrix, B. For the output matrix:

$$o_n = y_n - \hat{y}_n \tag{9.19}$$

$$\hat{y}_n = w_5 * y_{n-1} + w_4 * y_{n-2} + \ldots + w_1 * y_{n-5} \tag{9.20}$$

$$p(1|o_n) = 0.9 \exp\left(-\frac{o_n^2}{\sigma^2}\right) \tag{9.21}$$

$$p(2|o_n) = 1.0 - p(1|o_n) \tag{9.22}$$

The results (see Table 9.4 on page 103 and Table 9.5 on page 103) indicate that the Hidden Markov Model is able to improve the robustness of the linear predictor by detecting traffic spikes in the input data and training a second linear predictor on the time series that follow traffic spikes. This gives an improvement over the standard linear predictor but is not the best overall method.

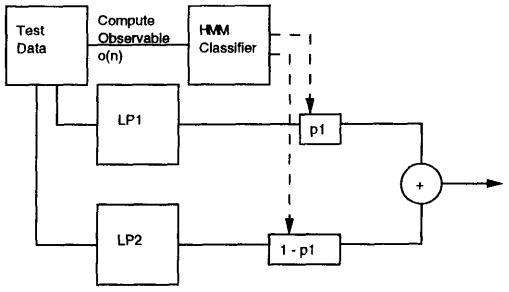

Figure 9.11: Testing a Dual Linear Predictor

9.14 Log Transformation

A log transformation of the data is carried out, prior to training a linear predictor. This is the best prediction method for the short data set. It is believed that the reason that it is so good is that it tends to give the same output as a linear predictor when the traffic levels are about constant. This is because the log predictor has approximately the same weights as the linear predictor, as shown in Figure 9.12 and averaging in "log space" is the same as averaging in linear space if the points being averaged are close together. Figure 9.13 highlights the fact that there is little difference between the log predictor and the linear predictor for most of the predictions.

On the other hand, in the event of spikes (as occurs in a small part of the data set) the log predictor gives much better prediction results. (See Figure 9.14). This may reflect the geometric averaging process as being better than the arithmetic averaging process following a spike. Figure 9.14 highlights the fact that there is a large difference between the log predictor and the linear predictor following traffic spikes.

It might be argued that heteroskedasticity is the cause of the log predictors success. Heteroskedasticity is the change of variance of the occupancy statistic as the traffic level rises. This may interfere with the calculation of the coefficients of the LP. A possible solution suggested in [Ros87] is to take a log transformation of the data to flatten the variance as is done here. However heteroskedasticity can be ruled out as a cause of the log predictors success because in this event, a weighted least squares predictor should obtain the same improvements as the log predictor, and this is found not to be the case.

Weighted least squares is a means to avoid the bias introduced by heteroskedasticity. The weighted least squares experiment trains a linear predictor but weighs each input

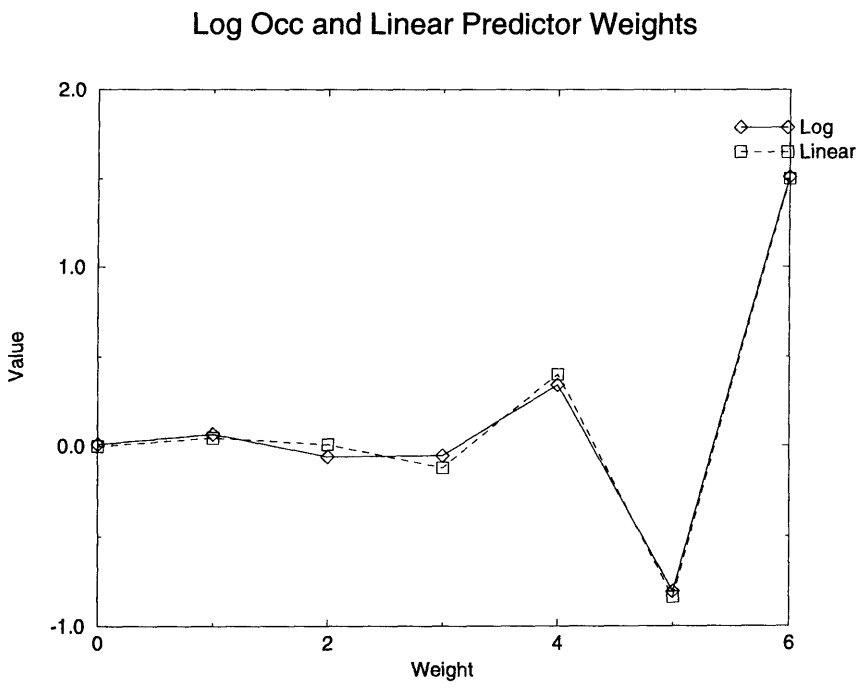

Figure 9.12: Comparison of Log and Linear Coefficients

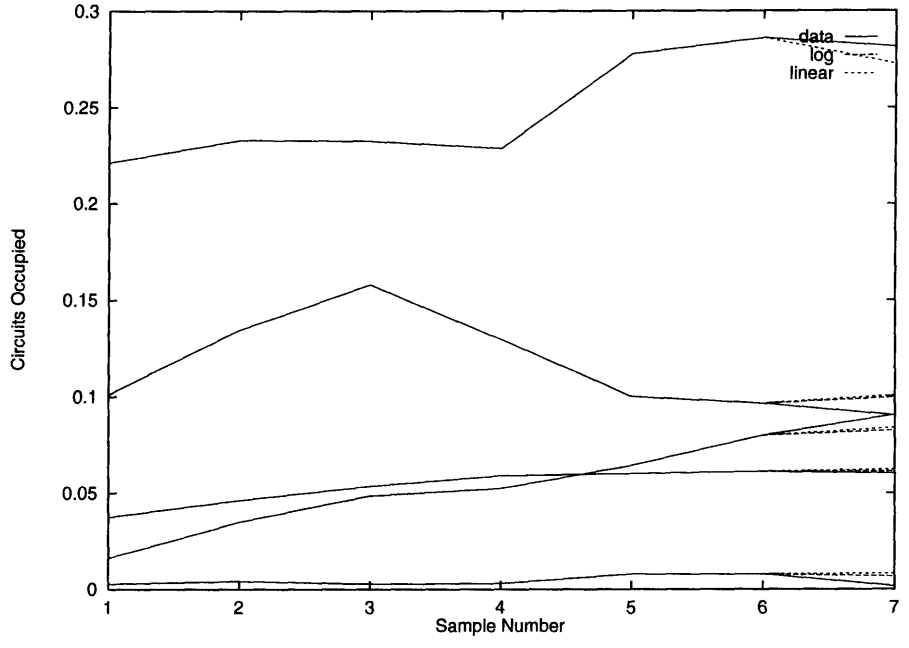

Figure 9.13: Typical Log Predictions and Linear Predictions. Note that Log and Linear Predictions are Close

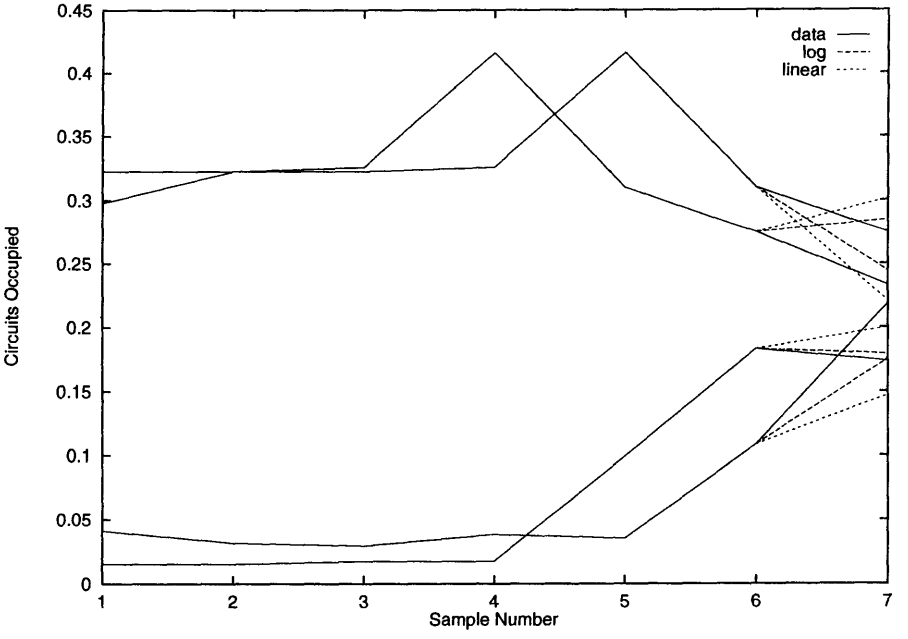

Figure 9.14: Log and Linear Predictions Following Spikes. Note that Log Predictor is Better than Linear

pattern vector according to the average level of traffic present. The details of the algorithm used and the best weights to use are given in [Ros87]. In this case, the best results are obtained with a weighting function that linearly decreases from 2.0 to 1.0 as the traffic level rose from 0 to 100% occupancy.

9.15 Including a daily profile

Using the longer data set, it is possible to generate a daily profile for the trunk group. The traditional approach by utility companies is to classify the day into one of:

- Saturday
- Sunday or holiday
- Day after Sunday or holiday
- Weekday

and then use historical records to derive a profile of each day. This profile is fed into the predictor to aid in prediction. The key point is that part of the daily variation that the occupancy statistic follows every day is deterministic.

Method	Error
Standard LP	13112
Indexed LP	12427

Table 9.3: Scaled RMS Prediction Error using Daily Profile

Method	Error
Using last sample	18308
Using linear predictor	16215
Using non-linear prediction	16167
Using neural network	16119
Using local approximation	16085
Using Hidden Markov Model	16081
Using log transformation	16033

Table 9.4: Scaled RMS Prediction Error for Short Data Set

Best results are obtained by dividing the data into each of the four day types, carrying out a 5 point moving average operation to smooth the data, calculating the profile for each day type by averaging the available data for each data period belonging to that day type and finally using the first order differences of the relevant profile as an extra input to the linear predictor. Carrying out all of these operations resulted in a 5% improvement in the prediction error, as shown in Table 9.3. Because of the 5 point moving average operation, the data set in this section is slightly different from the dataset in earlier sections.

In this implementation, no account is taken of holidays. Instead holidays are treated as normal work-days. It is likely that treating holidays the same as Sundays as suggested by [HY91] would slightly improve the results.

Method	Error
Using last sample	13487
Using linear predictor	13130
Using log transformation	13116
Using non-linear predictor	13061
Using Hidden Markov Model	13061
Using neural network	13054
Using local approximation	13011

Table 9.5: Scaled RMS Prediction Error for Long Data Set

9.16 Results

The results for the short data set are shown in Table 9.4. The results for the long data set are shown in Table 9.5. Neural networks and local approximation techniques do well on the long data set.

All methods predict the next observation based on the previous 6 observations. The linear predictor can be taken as the baseline performance to beat. Error is RMS prediction error multiplied by 10,000, with a difference of 100 for the short data set and 10 for the long data set being significant. In each case, 50-fold cross-validation is used.

9.17 Conclusions

The presence of occasional spikes in telephone traffic occupancy means that it is possible to do better than use a linear predictor for this data set.

The general nonlinear methods of neural networks and local approximation did well and can be expected to be near optimal as the data set size increases, as discussed in Section 9.11. Some progress in understanding the role of each hidden unit in the neural network predictor is obtained, as discussed in Section 9.9.

The HMM has a side effect of giving a classification of the state of the trunk group. This could be useful in a network management context.

Inclusion of a daily profile as one of the inputs, as described in Section 9.15, results in a 5% improvement in prediction capability. It is expected that there exists a fundamental limit to prediction implied by the random nature of call arrivals. This is explored in Chapter 10.

Chapter 10
Model of the Occupancy Process

10.1 Introduction

Figure 9.1 in the previous chapter indicates that occupancy depends on time of day. In this chapter the dependency of occupancy on the day of the week is examined. A model of weekday occupancy will be developed that provides a good visual fit with observed values.

Clearly, occupancy has some randomness due to the random nature of call arrivals. This random nature must be included in the model. The strategy pursued in this chapter is to find a daily profile, add a loading factor to account for the observed variability, simulate the daily profile with the added loading factor and compare it visually with the observed occupancy values. This is similar to the time series models developed by utility companies [HY91, PHK92].

A model of occupancy would have some benefits:

- It would indicate the limits of predictability. In Chapter 9 various methods used to predict trunk group occupancy are compared. With a model of the occupancy process, it is now possible to investigate how close each method comes to the fundamental bound imposed by the random nature of call arrival. Section 10.7 gives details.

- It would indicate the presence of spikes. With a model of occupancy, it is much easier to detect a departure from normal operations. It also becomes possible to attach a statistical significance to such departure. If properly presented, this information could be very valuable to the network traffic manager.

10.2 Day of Week Dependence

Looking at the long data set from Chapter 9, it is possible to check whether occupancy varies from day to day. This is done in Table 10.1. Clearly there is less traffic on Saturdays and Sundays than on weekdays. The most traffic is to be seen on a Monday, when business people return to business after a weekend. The traffic profile on Tuesday, Wednesday, and

Day	08:00	10:00	12:00	14:00	20:00
Mon	12.43 ± 3.32	37.87 ± 7.84	20.95 ± 4.68	25.00 ± 3.82	13.45 ± 3.13
Tue	11.08 ± 2.71	29.79 ± 4.44	18.41 ± 2.53	24.61 ± 2.79	12.15 ± 2.55
Wed	11.22 ± 2.63	26.56 ± 3.31	17.71 ± 2.90	22.44 ± 2.78	10.84 ± 2.70
Thu	11.23 ± 3.11	26.85 ± 8.69	17.62 ± 2.72	23.00 ± 4.40	13.53 ± 6.08
Fri	10.29 ± 3.02	26.98 ± 13.10	17.69 ± 4.31	21.66 ± 6.91	10.41 ± 2.77
Sat	3.89 ± 1.30	7.32 ± 1.35	7.55 ± 1.91	7.01 ± 2.12	6.59 ± 1.99
Sun	2.31 ± 0.88	4.61 ± 0.83	5.53 ± 1.26	5.04 ± 0.99	6.34 ± 1.52

Table 10.1: Occupancy Mean and Standard Deviation for Different Days and Times

Thursday seems to be similar. Based on this, the remainder of the chapter contains a description of the development of a model for occupancy on these days.

The dataset showed some anomalies on Thursday November 25 and Friday November 26, the Thanksgiving holidays. There is also less traffic than usual during the two weeks when many people take their Christmas vacations. To account for these holiday periods, the period from Thursday November 25th to Tuesday November 30th and the period from Saturday December 25th to Thursday January 6th are excluded from the data set. A complete model would include special cases for Thanksgiving and Christmas vacations. In the past, the Monday after Thanksgiving has been found to be the busiest day of the year for telephone traffic.

10.3 Tuesday to Thursday Scatter Plot

Figures 10.1, 10.2, 10.3 and 10.4 show scatter plots of a seven point moving average of occupancy on Tuesdays, Wednesdays, Thursdays and all three days respectively. The seven point moving average is chosen to aid the comparison of the daily profiles, by smoothing the random fluctuations in the traffic. One obvious traffic spike can be seen in Figure 10.3 close to data-point number 250. The similarity of the traffic profiles is remarkable.

10.4 Traffic Profile

Figure 10.5 shows the daily traffic profile. It is derived by simply averaging all the available readings for each data period.

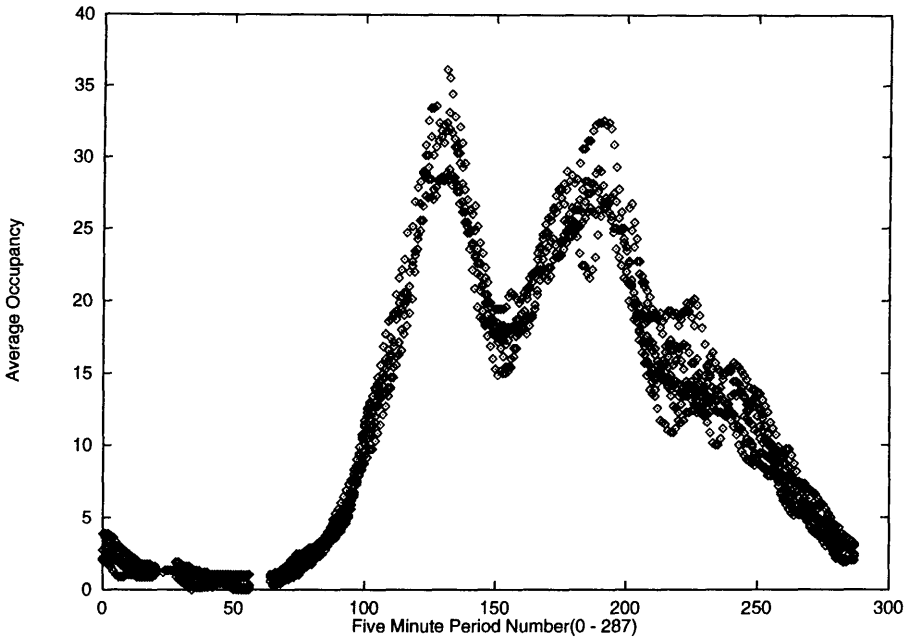

Figure 10.1: Tuesday Occupancy Moving Average

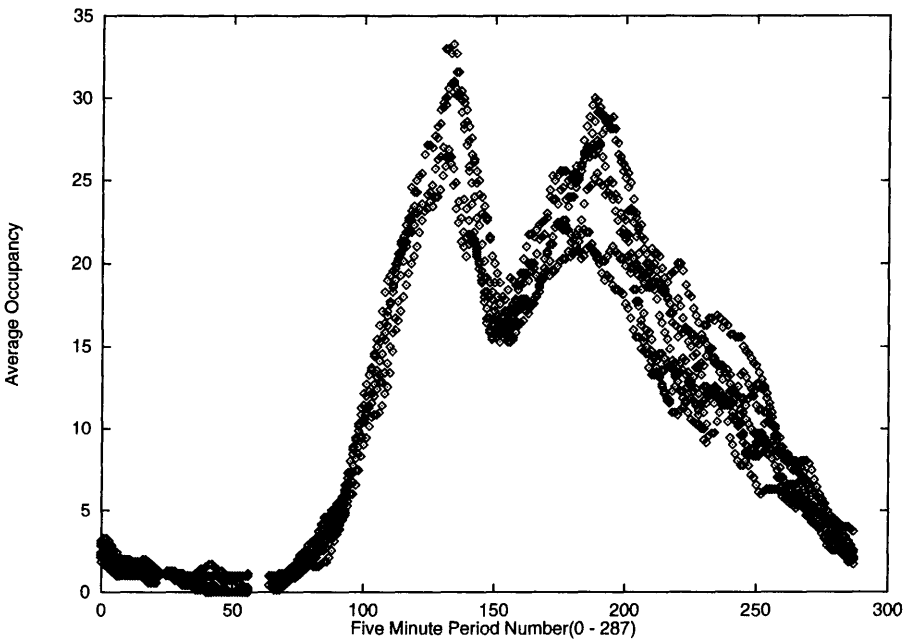

Figure 10.2: Wednesday Occupancy Moving Average

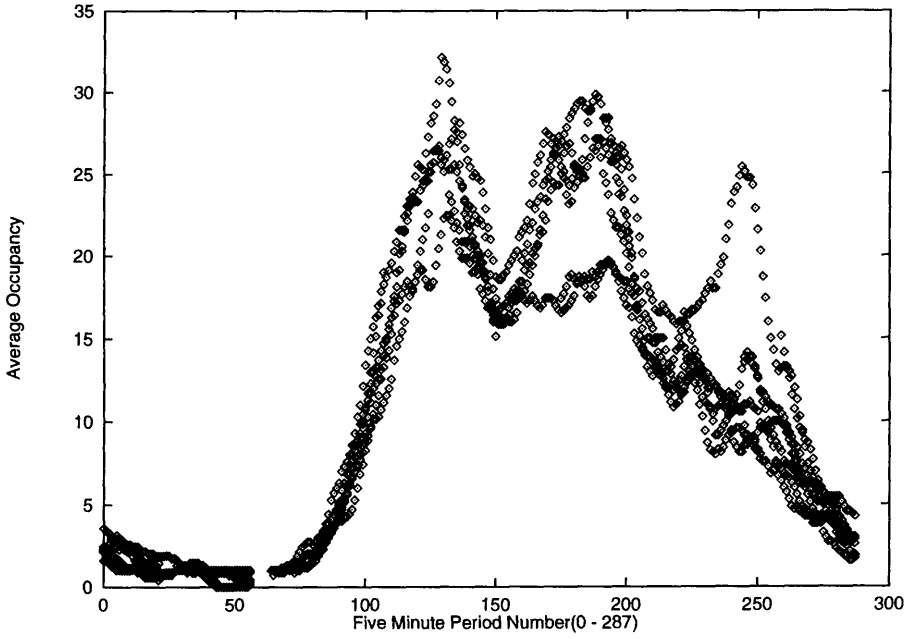

Figure 10.3: Thursday Occupancy Moving Average

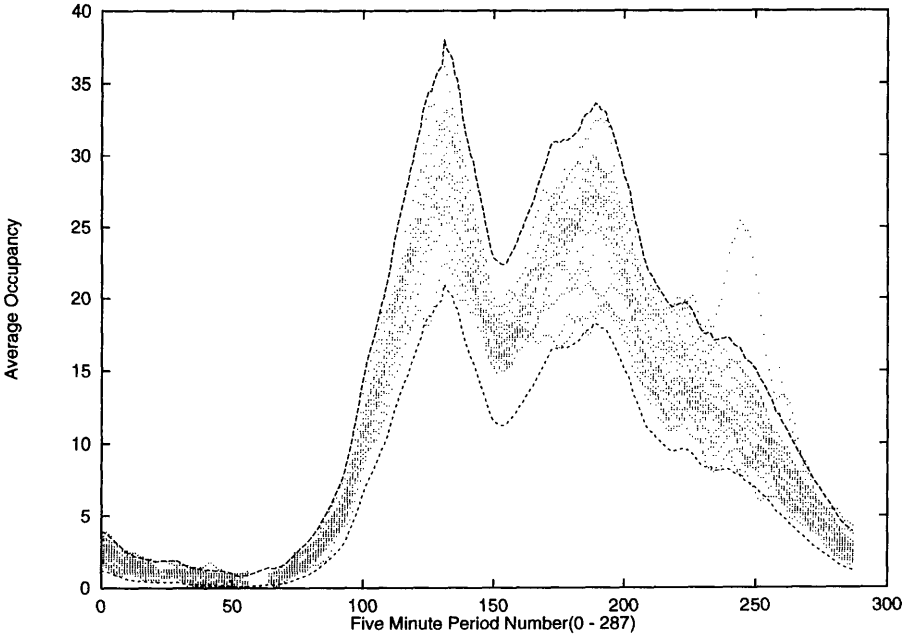

Figure 10.4: Tuesday to Thursday Occupancy Moving Average

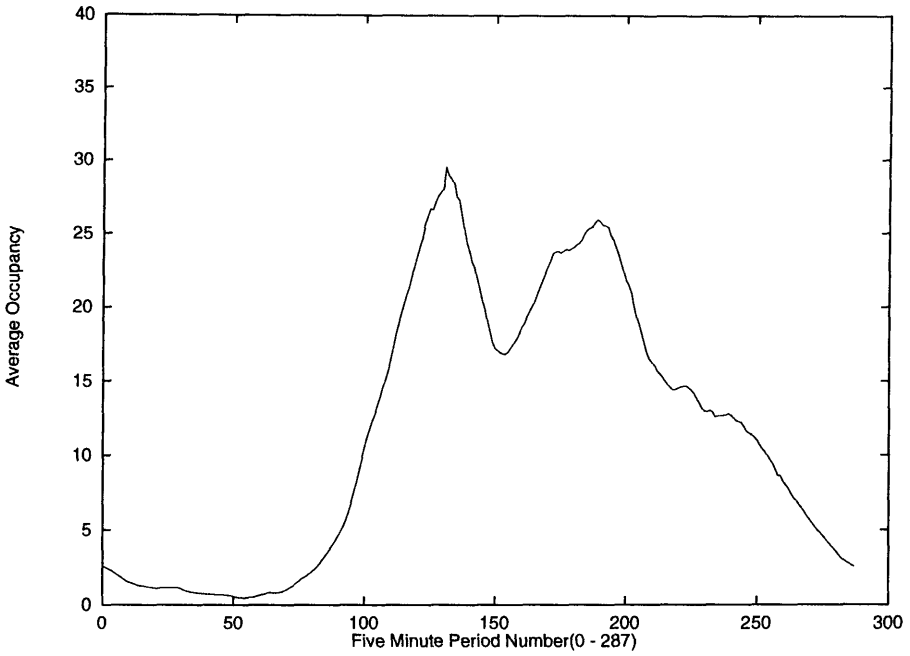

Figure 10.5: Daily Traffic Profile (Tue-Thu)

10.5 Characterizing the Variance

With a large number of trunks, the theoretical traffic distribution will be close to a Poisson distribution, with mean and variance equal to the offered traffic a. In other words, the traffic variance is proportional to the mean traffic level. The moving average operations used to calculate occupancy from traffic samples and to smooth the occupancy readings will not cause this property to be lost. If the applied traffic level is constant then one would expect to see a variance that is proportional to the traffic level. A plot of the bounds suggested by such a variance is given in Figure 10.6.

Clearly this can be improved upon. The fit is good at low traffic levels but is poor at higher traffic levels. If it is postulated that a loading factor that is proportional to the applied traffic is added to the profile, then making the variance proportional to traffic raised to a higher exponent than 1.0 may be better. Figure 10.7 shows the effect of bounds suggested by a variance that is proportional to traffic raised to the power of 1.5. The fit is good for all data periods except the lunch hour.

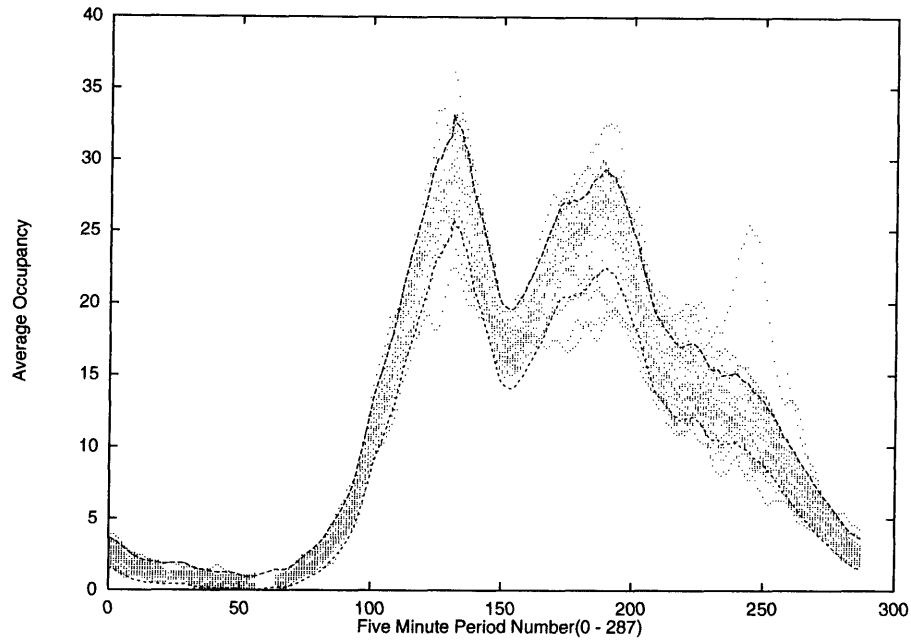

Figure 10.6: Traffic Upper and Lower Bounds with Variance(1)

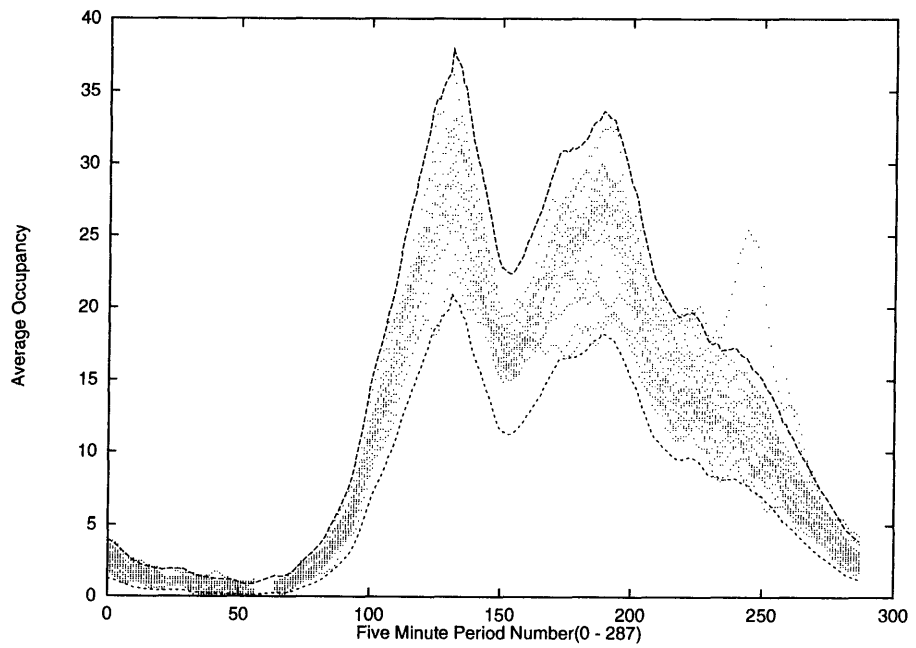

Figure 10.7: Traffic Upper and Lower Bounds with Variance(2)

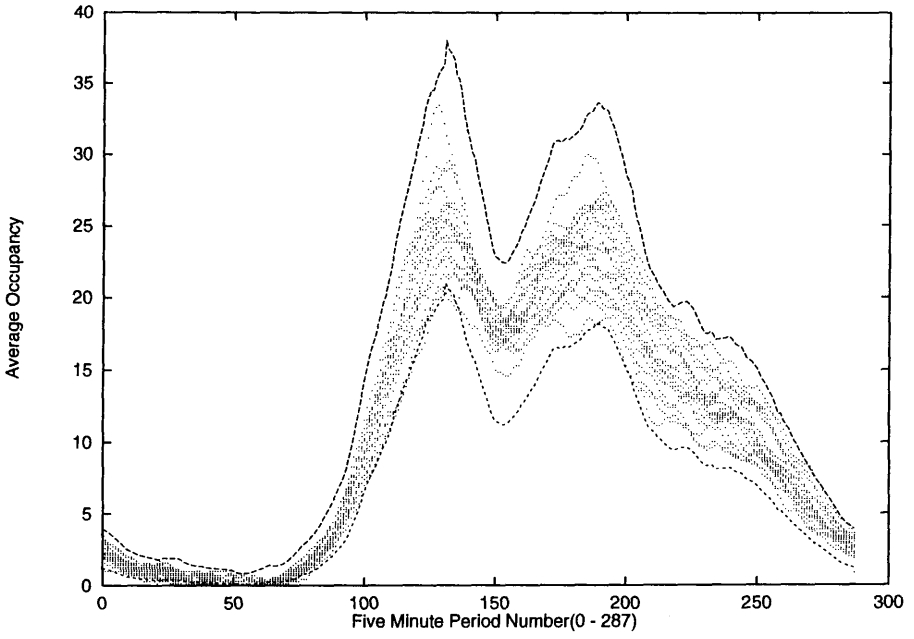

Figure 10.8: Simulation using Occupancy Model

10.6 Model

This gives rise to the proposed occupancy model. The model uses the traffic profile from the data set, adds a loading factor for the day, which is a multiplicative factor of the form

$$M = (1 + \sigma z) \tag{10.1}$$

where z is a normal random variable with mean 0 and variance 1, and σ is a loading factor. Good results are obtained with σ equal to 0.13. Finally this loading factor is removed during lunch hour to agree with the reduced variance observed during this time.

Thus, to simulate occupancy using this model, the following steps are needed:

1. Decide on day type.
2. Use the observed profile for that day type.
3. Scale the profile by a multiplicative factor M given in equation 10.1.
4. Simulate call arrivals according to that profile.

The visual agreement between the observed data in Figure 10.7 and the simulation of the model in Figure 10.8 is good.

Data Period	09:25	09:30	09:35	09:40	09:45	09:50	09:55
Occupancy	18.16	18.91	19.50	20.15	20.82	21.32	22.14
Realization	18.00	16.00	16.00	18.00	18.00	18.00	

Table 10.2: Occupancy Profile and One Realization Prior to 09:55

	Occupancy
Linear Predictor	17.8151
Neural Network	17.9051
Local Approximation	17.9715
Indexed Linear Predictor	19.1335
Simulated	19.4300

Table 10.3: Predictions for 9:55 Occupancy by Various Methods

Although this trunk group carries overflow traffic, for simulation purposes it is treated as first routed traffic without appreciable error.

10.7 Limits of Prediction

It was mentioned in the introduction that one of the benefits of a model of the occupancy process would be to find the limits of predictability of occupancy. It is possible to use the model of occupancy to simulate a large number of future evolutions for occupancy from a given point in time. Since each of these evolutions is equally likely, this will give an indication of the limits of predictability.

An example of this procedure for the time period ending in 9:55 (period 119) is shown in Table 10.2, and the results for the various prediction methods are shown in Table 10.3. It is possible to carry out 10000 evolutions of the realization shown in Table 10.2. Based on these evolutions, the best possible prediction of the next traffic sample at 09:55 would be 19.43 ± 0.84. Table 10.3 shows how close each of the prediction methods is able to get to this best result.

Clearly the use of indexing gives a major improvement over the other methods, coming closest to the expected value of the next occupancy reading.

The important lesson to be learned is that there is a certain randomness in call arrivals that it is impossible to predict. In this case, it is the standard deviation figure of 0.84 above. Unless one tries to model every subscriber in the network, it would be impossible to reduce the expected prediction error for the 09:55 prediction below this, assuming the

model of the occupancy process is accurate.

10.8 Simulated Dataset

It is possible to carry this technique a bit further. It is possible to generate a simulated dataset using the model of the occupancy process developed in this chapter. The difference between the simulated dataset and a real dataset is that for each future value of occupancy that is to be estimated, a large number of evolutions can be carried out, in order to estimate the expected value of the occupancy reading, and the standard deviation of the occupancy reading.

This technique is used to generate a dataset of 3000 examples of weekday occupancy. For each of the examples, 1000 evolutions are carried out. In this way, the expected value μ_i of the future reading of occupancy and the minimum prediction error σ_i, that is, the standard deviation of future reading of occupancy, are known for each of the 3000 examples in the dataset.

The following equation is used to calculate the error associated with the different prediction methods, E_m. Here $\hat{y}_i$ is the predicted value of the test set for example i for this prediction method, found using 50-fold cross-validation on the 3000-element dataset.

$$E_m = \sqrt{\sum_{i=0}^{3000} (\sigma_i^2 + e_i^2)} \tag{10.2}$$

with the error, e_i, being defined as:

$$e_i = \hat{y}_i - \mu_i \tag{10.3}$$

The error, E_{opt}, associated with a hypothetical optimum prediction method is simply:

$$E_{\text{opt}} = \sqrt{\sum_{i=0}^{3000} \sigma_i^2} \tag{10.4}$$

This results for the different prediction methods are reported in Table 10.4. The results are scaled by 10,000 to get rid of decimal points. The indexed methods use an extra input which is the first order difference of the daily profile for that particular time of day. This gives 7 inputs instead of the usual six.

	Error
Last Sample	8281
Linear Predictor	7889
Local Approximation	7416
Indexed Linear Predictor	7030
Indexed Local Approximation	6997
Optimum	5782

Table 10.4: Scaled RMS Prediction Error on Simulated Dataset

The local approximation method gives a sizeable gain over a linear predictor. However the indexed local approximation method is the best method, coming close to the performance given by the optimum predictor for this model. The indexed linear predictor is not a lot worse.

The figure of 6997 in Table 10.4 seems to be the smallest figure obtainable, using the given inputs. This is indicated by the fact that the *training error* from a neural network with the number of hidden units varying from 1 to 10 can show no improvement.

The results in Table 10.4 give an indication of how close to the limit of predictability the various prediction methods can come.

10.9 Effect of Dataset Size

Equation 9.12 seems to suggest that as the dataset size increases, the RMS error should decrease according to the formula:

$$\mathrm{E}(e_{test}) \approx \mathrm{E}(e_{train}) + 2\sigma^2 \frac{p}{n} \tag{10.5}$$

where p is the number of parameters (weights) being estimated.

It is possible to test the effect of using a smaller dataset size by simply breaking the 3000 element data set into n groups of size $3000/n$. For the indexed linear predictor the results are given in Figure 10.9. A good fit is obtained using the $1/n$ term, however it is clear that the fit could be improved. A better fit includes $1/(n^2)$. The fitting formula is:

$$\mathrm{E}(e_{test}) \approx 0.701499 + \frac{4.32672}{n} + \frac{43.7092}{n^2} \tag{10.6}$$

This would indicate that the minimum test error for very large n would be 0.7015. It would also indicate that a 3000 point dataset would be large enough to come very close

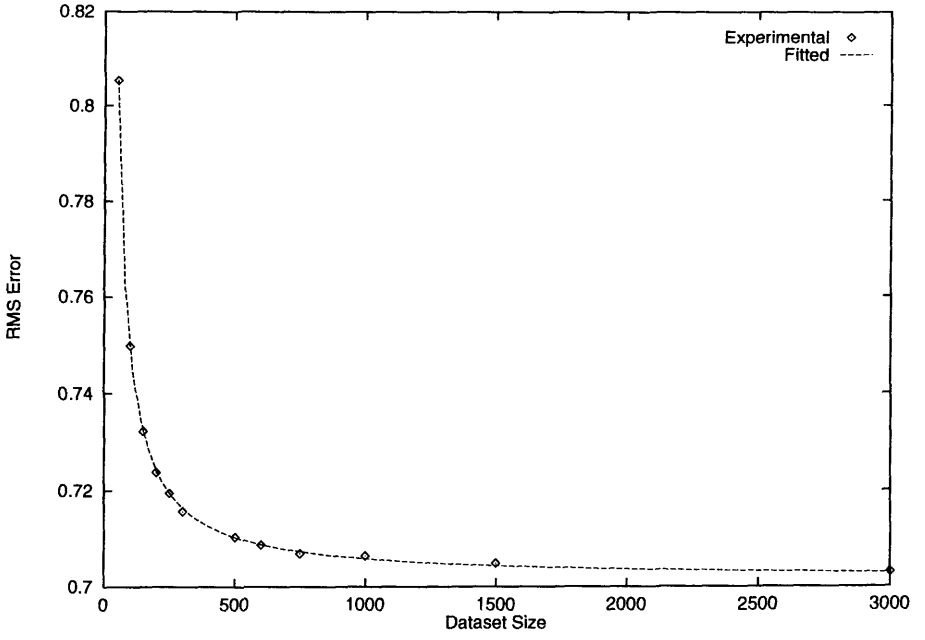

Figure 10.9: Decrease in Error for Indexed Linear Predictor as Dataset Size Increases

to this minimum test error. In fact, from the graph, it is apparent that dataset sizes of greater than 500 points are sufficient for coming close to the minimum error.

10.10 Conclusions

The model of occupancy proposed in this chapter is similar to time series models of power load put forward by utility companies. The aim of the model is to explicitly incorporate all of the sources of variation in the occupancy readings. There are a number of sources of variation in the occupancy readings. The occupancy readings depend on:

- Day of Week
- Time of Day
- A Loading Factor
- Exact Arrival Instants of Calls

The model of occupancy that is proposed includes all of these sources of variation. It successfully describes the observed occupancy during weekdays for one trunk group.

It has been used to investigate the limits of predictability of trunk group occupancy. A model based approach should also prove useful for the detection of traffic spikes.

Chapter 11
Conclusions

11.1 Introduction

In this thesis, some of the issues that arise from considering dynamic routing based on 5 minute data are examined. It is foreseeable in the future that all telephone networks automatically adjust the routing tables in the switches in the network to compensate for changing network conditions and provide the maximum of service quality with the minimum of equipment.

11.2 Dynamic Routing

In Chapter 3, simulations are carried out to test the benefits of dynamic routing based on 5 minute data. The factor with the largest influence seems to be network connectivity. With a typical network connectivity of 0.25, the equipment savings from implementing dynamic routing are approximately 2%. In a typical metropolitan network, with higher connectivity, the equipment savings may be of the order of 3.5%.

The benefit of dynamic routing are also seen to decrease as trunk group sizes increase. This is explained qualitatively as a consequence of larger trunk groups containing relatively fewer free trunks for a given level of blocking. This means that there is less spare capacity for the dynamic routing algorithm to use.

11.3 Fixed Point Models

A fast method for the calculation of network blocking probabilities is described in Chapter 4. For symmetric networks and simple routing schemes, FPMs are instantaneous. However two drawbacks are apparent.

First, the use of FPMs usually implies an assumption of route independence. Thus FPM blocking results are approximate. Without resorting to simulation, it is difficult to write down an expression for the error introduced by the fixed point method.

Second, for non-symmetric networks, the run-time for the FPM will be proportional to the number of routes to be modeled, which rises according to $O(N^2)$ where N is the number

of network nodes. Thus runtimes can rise very quickly as the network size increases.

Despite these limitations, FPMs have been useful for calculating estimates of network blocking associated with various routing techniques.

11.4 Network Stability

In Chapter 5, simulations and analysis are used to examine the dynamics of dynamic routing, when the routing information experiences a delay in feedback. The lesson is that tight control of the feedback time of routing information is necessary in order to avoid network instability.

11.5 Traffic Patterns

A model-based approach to the detection of normal and abnormal traffic patterns is put forward in Chapter 7. An example of the use of this approach to characterize a stormy day in Los Angeles as being an abnormal day with a 99.7% certainty level is given. It is seen that the first step in detecting departures from normal operation has to be the formulation of a model that describes normal operation.

11.6 Learning Techniques

In Chapter 9 the application of many different learning techniques to time series prediction of telephone traffic occupancy is considered. It emerges that local approximation and neural networks are two techniques that generalize well. Using a long data set with about 18000 elements, it is possible to be confident in the comparative performance of the different prediction techniques.

It should be noted that in network management, function approximation is important. There are many functions that would be difficult to solve for analytically, that can be obtained by fitting a function to observed network data. For example, the optimum trunk reservation parameters are considered in Chapter 8.

The linear predictor and the neural network are good function approximators and difficult to beat. The linear predictor has advantages over the neural network in that it is quick to train and there is little or no danger of overfitting. On the other hand, the

neural network can provide a better fit for functions that are a non-linear function of their inputs.

Cross-validation is the key technique for testing learning ability. The cross-validation technique allows maximum use of the dataset, by dividing it into many small test sets. In this way, the significance of the difference in performance between the various prediction methods can be judged. For this reason, cross-validation is the perfect method for testing generalization capability.

Using the model of the occupancy process in Chapter 10, it is possible to estimate that the linear predictor came to within 36% of optimal prediction, and that the local approximation technique came to within 28% of optimal prediction. Both of these figures assume that the model of the occupancy process is an accurate description of the process. As an engineering principle, it is useful to know what the bounds of performance are for a particular problem to see whether it is worthwhile to improve the standard techniques to come closer to those bounds.

The presence of spikes in the traffic data mean that methods for robust time series prediction must be considered. The HMM technique described in Chapter 9 automatically detects when a time series is going out of bounds, and stops training the linear predictor when this happens. This results in good prediction performance. Alternatively a model based approach allows one to simulate the evolution of occupancy over the course of a day, and detect any significant departures from normal operations.

11.7 Future Work

The following are some ideas for future work. In all cases, the future work would require further data collection steps, and hence is not included in this thesis.

More Fine Grained Traffic Patterns In Chapter 7 the work on traffic patterns is based on observing network exceptions and controls placed in the network. This is a very coarse grained view of what is happening in the network. If instead, daily traffic profiles are available for all the trunk groups, then it would be possible to study smaller departures from normal operations. Much more information would be available to support traffic pattern classification.

Inputs for Trunk Reservation In Chapter 8 the inputs for the trunk reservation study are chosen in an ad hoc fashion. The model of trunk reservation as a bet should lead to the determination of what inputs are necessary and sufficient to make a good trunk reservation decision. In carrying out any prediction operation, the resulting prediction is only as good as the input information supplied for the prediction.

Occupancy Model for Different Types of Trunk Groups In Chapter 10 the occupancy model proposed is a good fit for the trunk group for which data had been collected. It remains to be seen whether the same model will be equally good for other trunk groups carrying different traffic mixes.

Stability of Trunk Reservation In Chapter 8 the trunk reservation policy is learned by neural networks to provide reduced network blocking. In this thesis, the question of whether such a policy of learning the network control algorithm can ever be unstable is not answered. Intuitively, one would suspect that the policy can be unstable if the learning rate is such that the neural network weights can vary a lot with each new call arriving. There may be a limit on the learning rate, below which the control algorithm is always stable. This should be an interesting area for further study.

Appendix A
Network Operations Analyzer and Assistant (NOAA): A Real-Time Traffic Rerouting Expert System

A.1 Introduction

The research in this thesis was inspired by a system called NOAA (Network Operations Analyzer and Assistant). NOAA is a set of programs that runs in the Pacific Bell Network Management Center in Sherman Oaks. The objective of NOAA is to duplicate the actions of the network management staff in responding to emergencies and putting controls into the network.

Typical emergencies may be:

- Random overloads of trunk groups.
- Focussed overloads caused by phone-ins.
- Unusual calling patterns as may occur on Mother's day.
- Earthquakes.
- Other major events.

In each of these cases, the network manager must diagnose what the problem is, based on observable symptoms and place controls in the network to reduce the impact of the network event.

The NOAA project was started in September 1990 as a joint project between Caltech and Pacific Bell to develop an expert system to aid in real-time network management. It has since been developed to the extent that it is placing controls in the Southern California telephone network without any manual intervention, and has become a product currently offered for sale to other regional Bell telephone companies.

A.1.1 Network Concepts

In order to gain some appreciation of the network management tasks, one must have a description of the network to be managed. The Pacific Bell Southern Californian telephone

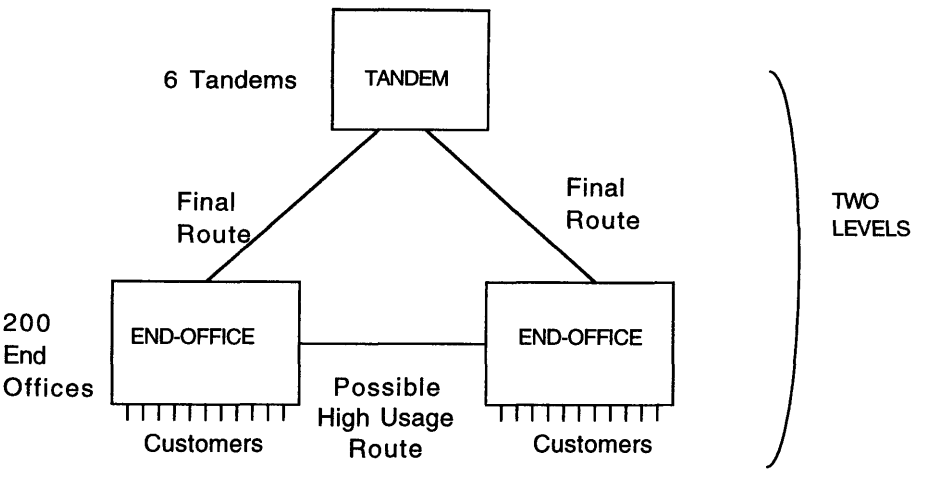

Figure A.1: Pacific Bell Southern California Telephone Network

network provides service to at least 1 million subscribers. For administrative and legal purposes it is divided in Local Access and Transport Areas (LATAs) and the network management center in Sherman Oaks manages LATAs 5 and 6. Geographically LATAs 5 and 6 extend to San Diego in the south and include all of the Los Angeles urban area.

The network is hierarchical. *Endoffices* are the exchanges that serve customers and *tandems* are the exchanges used for traffic between endoffices that are not directly connected. This is illustrated in Figure A.1. For Southern California, there are 6 tandems to be managed and over 200 endoffices. Each endoffice is connected to one or more tandems and the tandems are fully interconnected.

There are two types of trunk groups. *High usage* trunk groups are dimensioned to be lossy, i.e. during the busy hour they are not guaranteed to have enough capacity to carry all the offered traffic. Traffic will therefore overflow on the *Final* trunk groups which are dimensioned to provide enough trunks so that calls are rarely blocked. In general there will be a final route between each endoffice and its parent tandem. It is these final routes that provide the backbone of the network. The final routes are therefore closely monitored by the network managers.

A.1.2 Network Management Concepts

Network management staff may reroute traffic elsewhere (*expansive controls*) or cut the traffic off at its source (*restrictive controls*).

Expansive controls are appropriate for a single overload situation where due to sta-

tistical fluctuations in the offered traffic, a trunk group does not have enough capacity to handle its offered load. Typical expansive controls might be (i) `orr` (overflow reroute) which reroutes calls to another trunk group after they have overflowed or (ii) `irr` (immediate reroute) which reroutes a certain percentage of calls before they even try the "problem route." Other controls may have to be put in at the same time as the main control to avoid routing loops.

`Irr` controls provide an advantage when there is a two way trunk group and the office at the far end cannot effect controls. By using an `irr`, enough traffic can be lifted to prevent overflow at either end.

Restrictive controls are appropriate for call-in conditions, where most of the traffic has a low probability of completion, but its presence is interfering with the normal network operations. This traffic is characterized by a large number of call attempts per trunk and low holding time. Currently NOAA handles expansive controls and restrictive controls to a limited extent.

A.1.3 Netminder

Netminder is an AT&T product used in the Pacific Bell network management computer system. It provides a display to the network operators of the state of trunk groups in the network. It highlights overflow conditions on final trunk groups and alarm conditions in the telephone exchanges.

Netminder information is polled from the offices in Southern California every 5 minutes for trunk group data, and every thirty seconds for switch alarm data. Netminder stores this information in its database. NOAA can then query this information using SQL database interrogation commands. SQL is a database query language.

A.2 CUBE

CUBE is the Caltech / U S Geological Survey Broadcast of Earthquakes system. It provides epicenter and magnitude information of any earthquake occurring in California. Although CUBE only applies to California, the same type of system could conceivably be used to access information about other types of natural disaster, such as the National Hurricane Center's early warning system and tornado watch data.

Indications of an earthquake are first received on sensors distributed throughout California. This data is relayed to Caltech in Pasadena, where it is processed to provide epicenter location and magnitude information. Pager messages are then sent on the standard paging system to NOAA, and a data interface to the CUBE pager allows the message to be read and processed by NOAA.

Earthquake information received in real-time is displayed on NOAA's map in the form of a circle around the epicenter along with a numerical indication of the magnitude of the quake on the Richter scale. The map interface allows the operator an immediate identification of quake location and magnitude as well as identification of end-offices that may be impacted by the quake.

A.3 Architecture of NOAA

The Architecture of NOAA is shown in Figure A.2. The Pacific Bell network management system is called Netminder. NOAA is connected over an Ethernet data link and appears as an ordinary operators terminal to Netminder. NOAA runs on a Sun workstation under UNIX.

A.4 Statistics

NOAA makes its diagnosis on the basis of available information about the trunk groups. This information is available to NOAA in the form of counts of certain events for each trunk group during a 5 minute period. These events are, for instance, (i) a trunk being seized in order to carry a telephone call (ii) a seizure that is not successful i.e. overflow (iii) a survey of trunk occupancy at a given instant. On the basis of these counts, the following standard statistics are calculated:

ACH Attempts per circuit per hour

CCH Connections per circuit per hour

OFL Percentage of attempts that overflowed

OCC Percentage of trunks occupied on average

HT Holding Time of Calls

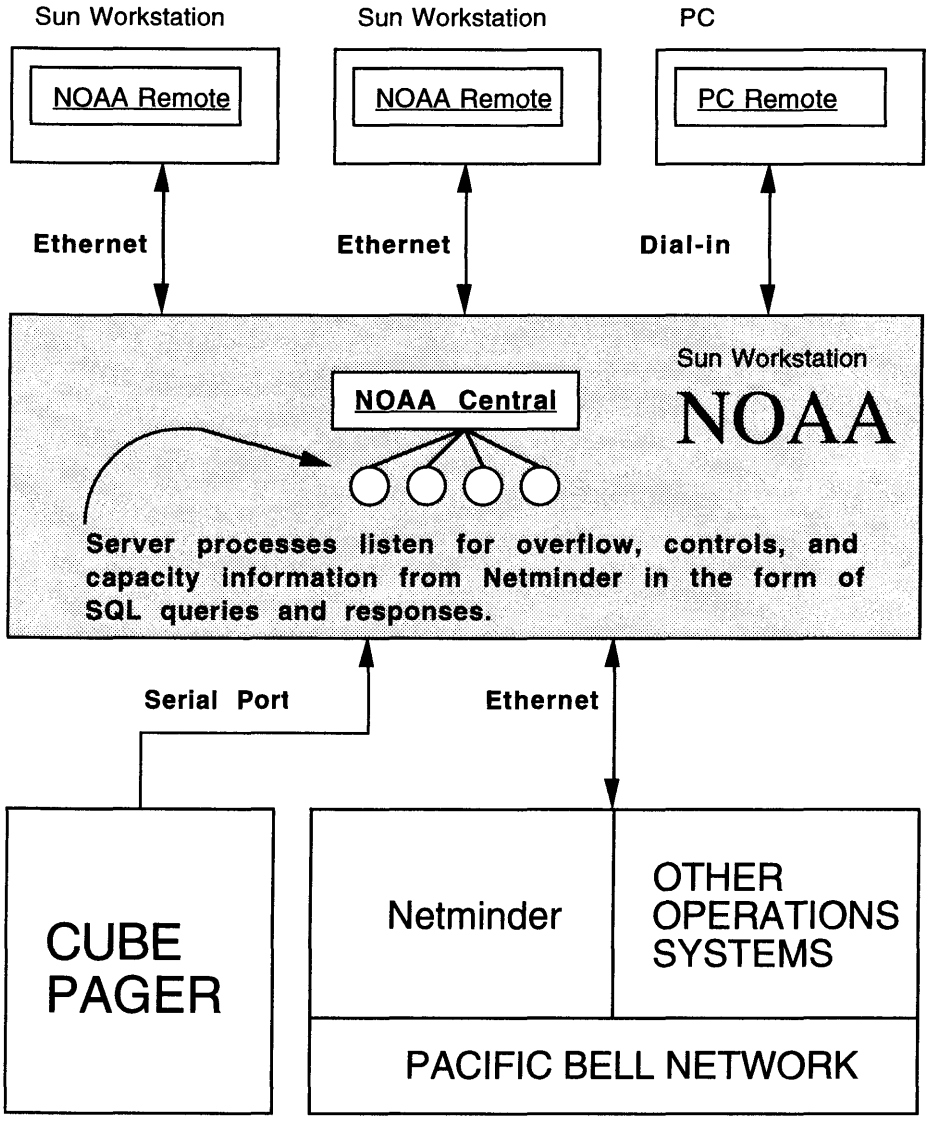

Figure A.2: Architecture of NOAA

The 5 minute period is a compromise. The shorter the period between data collections, the faster the response to any potential problems, but the larger the amount of information to be processed, requiring more expensive hardware. The 5 minute period means that problems can be cleared relatively rapidly.

Since a large amount of information is available, filtering is carried out. First, exceptions on high usage routes are ignored and only final route data are examined. Second, exceptions on special purpose final routes are ignored. For example, a trunk group for the time announcement can be expected to have a short holding time. Thirdly overflows are treated as having higher priority than other exceptions, since an overflow on a final may represent lost traffic.

A.5 Decisions

When an exception condition has been noted on a trunk route, there are many possible explanations for it. Typically phone-ins to radio stations and TV stations may generate excess call attempts. Facilities failures may mean that overflow shoots up on related trunk groups. Occasionally the data gathering may interfere with maintenance operations and unreliable data is returned. Finally random overflows can occur on individual trunk groups.

For any reroutes, a search for spare capacity is carried out. If there is an overflow problem between A and B where A is an endoffice and B is a tandem, then the program first looks for capacity on other trunk groups going from A to B, and then carries out a search for A to B via C possibilities. Here C has to be one of the 6 tandem exchanges, because of the hierarchical nature of the network. The candidate reroutes are then sorted according to available capacity and controls are suggested that make use of the minimum number of reroutes that achieves the required capacity.

The rules used in the program are of three separate types:

- rules that indicate which exceptions can be safely ignored. For example overflow on high usage routes is ignored;
- rules that indicate which routes can be used as candidate reroutes;
- rules that map a suggested reroute into a list of controls to effect the reroutes. For example, certain other routes may have to be finalized first to prevent a round-robin

DISREGARD ANY EXCEPTIONS ON TRUNK GROUP COMMON LANGUAGES (CLLIS) ENDING WITH "MD" (EX: LSANCA02AMD; LSANFDRCCMD). EXCLUDE SAME WHEN SEARCHING FOR VIA ROUTE CANDIDATES.
DISREGARD ANY EXCEPTIONS ON CLLIS INDICATING "PB" IN THE STATE DESIGNATION (EX: OKLDPB0349T). EXCLUDE SAME WHEN SEARCHING FOR VIA ROUTE CANDIDATES.
DISREGARD ANY EXCEPTIONS ON THE FOLLOWING HIGH VOLUME CALL-IN CLLIS: HLWDCA01520, SNANCA01977, COTNCA1143A, SIMICA11629, SNDGCA0157X

Table A.1: Typical Network Management Rules

situation, where two switches pass a call back and forward because of the network routing rules.

Some of the above rules were already written down in operators handbooks. Others were supplied by the network management staff. Examples of the rules are given in Table A.1.

A.6 Controls

It was mentioned above that there are many causes of exceptions on the trunk groups. Similarly there are many possible controls that can be put in to improve the network throughput. In general, controls can be divided into protective controls and expansive controls [Fil90, Bel88].

Protective controls are used to cut down the amount of call attempts entering the network. For example in the event of a phone-in, controls can be put into each endoffice to block call-attempts to that telephone number. Alternatively a certain percentage can be let through. This limits the amount of congestion in the network. Clearly the calls destined for a phone-in are bad traffic as there is a high possibility of non completion, and therefore they should be blocked at source and not rerouted.

On the other hand, *expansive* controls allow trunk groups that are normally separate to share their capacity. If there is a single random overload, then expansive controls are called for. Two alternatives here are overflow rerouting, which reroutes overflow traffic only, and an immediate reroute which puts all call attempts onto the alternative route

first. Each is appropriate for certain cases. Finally spray rerouting is possible, whereby a number of alternatives is given and each is tried for a given call.

A.7 Graphics Interface

A clear record of the actions that NOAA carried out was also essential to gaining user acceptance. This was achieved through a well designed graphics interface. In this case, two scrolling lists were provided. The first was a list of exceptions in the network, the second a list of controls currently in the network. Beside each entry, NOAA's recommended action was given in summary form. Verbose advice was available by selection of an exception or control and pressing the "details" button. Color was used to supply an indication of the severity of the exception or the recommended action for the control. For example, a red lamp next to the control entry signified that NOAA recommended a modification, whereas a yellow lamp next to the control entry indicated that NOAA recommended removing the control. Exceptions were filtered before being presented to the operator. The aim was to give the operator an at-a-glance indication of the status of the network and the recommendations that NOAA produced. A screen capture is shown in Figure A.3.

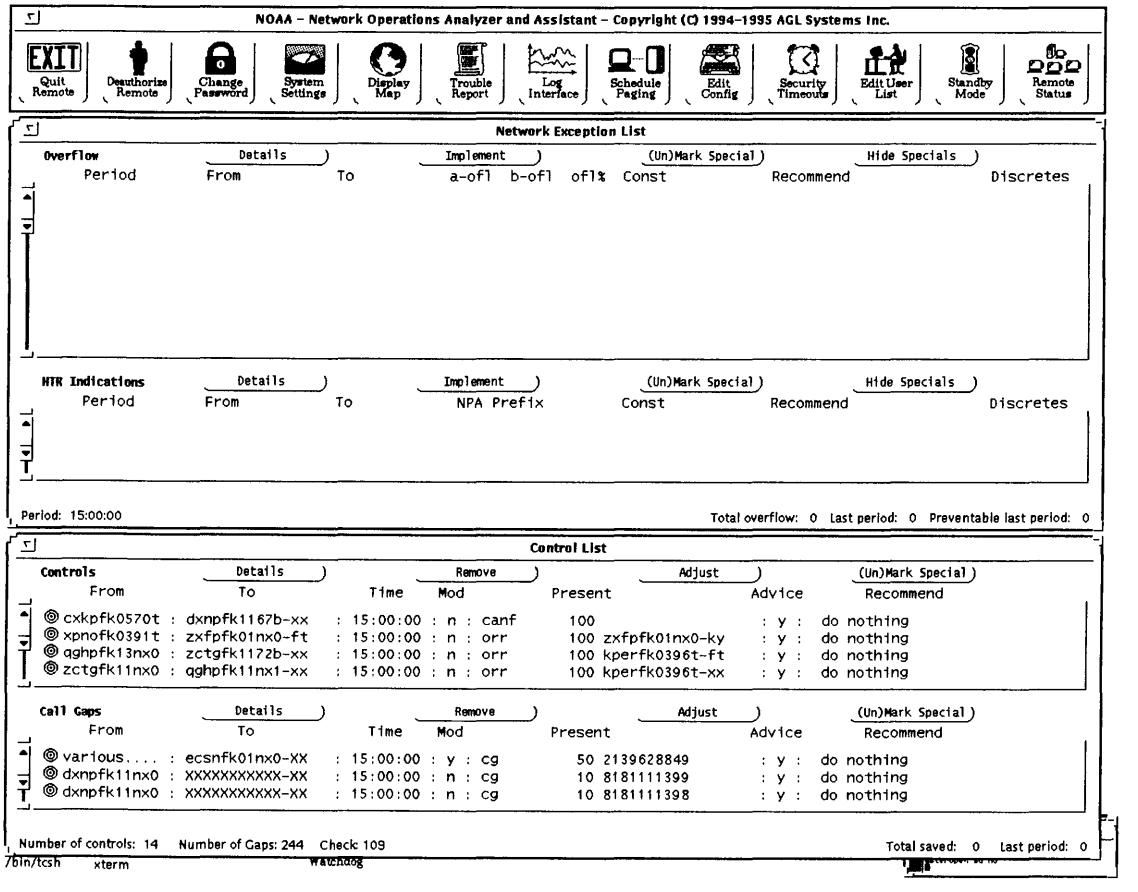

Figure A.3: Graphical Interface for NOAA

References

[AC93a] G. R. Ash and Fu Chang, "Management, control and design of integrated networks with real-time dynamic routing," *Journal of Network and Systems Management*, 1(3):237–253, September 1993.

[AC93b] G. R. Ash and Fu Chang, "Transport network design of integrated networks with real-time dynamic routing," *Journal of Network and Systems Management*, 1(4):319–335, December 1993.

[ACF92] Gerald R. Ash, Jin-Shi Chen, and Alan E. Frey, "Real-time network routing in the AT&T network - improved service quality at lower cost," In *GLOBECOM '92*, pages 802–809, Orlando Florida, December 1992.

[ACL94] G. R. Ash, K. K. Chan, and J. F. Labourdette, "Analysis and design of fully shared networks," In *International Teletraffic Congress - 14*, pages 1311–1320. Elsevier Science Publishers B. V. (North Holland), 1994.

[Aki84] J. M. Akinpelu, "The overload performance of engineered networks with non-hierarchical and hierarchical routing," *Bell System Technical Journal*, pages 1261–1280, September 1984.

[Bar93] Andrew R. Barron, "Universal approximation bounds for superpositions of a sigmoidal function," *IEEE Transactions on Information Theory*, 39(3):930–945, May 1993.

[Bel88] Bellcore, "Network management control descriptions," Technical Report BR 780–150–128, Bellcore, New Jersey, 1988.

[BJR94] George E. P. Box, Gwilym M. Jenkins, and Gregory C. Reinsel, *Time Series Analysis: Forecasting and Control*, Prentice Hall, New Jersey, 3 edition, 1994.

[Car89] France Caron, "Results of the Telecom Canada high performance routing trial," In *International Teletraffic Congress - 12*, pages 86–95. Elsevier Science Publishers B. V. (North Holland), 1989.

[Chr91] Ronald Christensen, *Linear Models for Multivariate, Time Series, and Spatial Data*, Springer Texts in Statistics. Springer-Verlag, New York, 1991.

[CKP91] V. P. Chaudhary, K. R. Krishnan, and C. D. Pack, "Implementing dynamic routing in the local telephone companies of the USA," In *International Teletraffic Congress - 13*, pages 87–91. Elsevier Science Publishers B. V. (North Holland), 1991.

[CMA94] Jerome T. Connor, R. Douglas Martin, and L. E. Atlas, "Recurrent neural networks and robust time series prediction," *IEEE Transactions on Neural Networks*, 5(2):240–254, March 1994.

[Coo81] Robert B. Cooper, *Introduction to Queueing Theory*, Elsevier Science Publishers B. V. (North Holland), New York, 2 edition, 1981.

[Cyb89] G. Cybenko, "Approximation by superposition of a sigmoidal function," *Math. Contr. Signals Syst.*, 2:303–314, 1989.

[DH87] Richard A. Damon, Jr. and Walter R. Harvey, *Experimental Design, ANOVA and Regression*, Harper & Row, New York, 1987.

[DS84] J. Dunlop and D. G. Smith, *Telecommunications Engineering*, Van Nostrald Reinhold, Berkshire, England, 1984.

[DSW73] J. J. Di Stefano III, A. R. Stubberud, and I. J. Williams, *Feedback and Control Systems*, Schaum Outline Series, New York, 1973.

[Fah88] S. E. Fahlman, "Faster-learning variations on back-propagation: An empirical study," In *1988 Connectionist Models Summer School.* Morgan Kaufmann, 1988.

[FC87] Janusz Filipiak and Prosper Chemouil, "Modeling and prediction of traffic fluctuations in telephone network," *IEEE Transactions on Communications*, 35(9):931–941, September 1987.

[FCC89] Janusz Filipiak, Prosper Chemouil, and Edward Chlebus, "Modelling and control of time-varying telephone traffic," In *International Teletraffic Congress - 12*, pages 96–103. Elsevier Science Publishers B. V. (North Holland), 1989.

[Fel50] William Feller, *An Introduction to Probability Theory and Its Applications*, John Wiley & Sons, New York, 1950.

[Fil90] J. Filipiak, *Real Time Network Management*, Draft Copy, 1990.

[FS87] J. Doyne Farmer and John J. Sidorowich, "Predicting chaotic time series," *Physical Review Letters*, 59(8):845–848, August 1987.

[GA93] Rodney M. Goodman and Barry Ambrose, "Applications of learning techniques to network management," In *International Workshop on Applications of Neural Networks to Telecommunications*, pages 34–44, November 1993.

[Hay91] Simon Haykin, *Adaptive Filter Theory*, Prentice-Hall, New York, 3 edition, 1991.

[HKP91] John Hertz, Anders Krogh, and Richard G. Palmer, *Introduction to the Theory of Neural Computation*, volume I of *Santa Fe Institute Studies in the Sciences of Complexity*, Addison-Wesley, New York, 1991.

[HSS87] B. R. Hurley, C. J. R. Seidl, and W. F. Sewell, "A survey of dynamic routing methods for circuit-switched traffic," *IEEE Communications Magazine*, 25(9):13–21, September 1987.

[HSW89] K. Hornik, M. Stinchcombe, and H. White, "Multilayer feed-forward networks are universal approximators," *Neural Networks*, 2:359–366, 1989.

[HY91] Yuan-Yih Hsu and Chien-Chuen Yang, "Design of artificial neural networks for short-term load forecasting. part ii: Multilayer feedforward networks for peak load and valley load forecasting," *IEE Proceedings-C*, 138(5), September 1991.

[Kel94] F. P. Kelly, "Bounds on the performance of dynamic routing schemes for highly connected networks," *Mathematics of Operations Research*, 19(1):1–21, February 1994.

[KO89] K. R. Krishnan and T. J. Ott, "Forward-looking routing: A new state-dependent routing scheme," In *International Teletraffic Congress - 12*, pages 1026–1032. Elsevier Science Publishers B. V. (North Holland), 1989.

[KON91] Jaidev Kaniyil, Yoshikuni Onozato, and Shoichi Noguchi, "A unified approach towards characterization of structural stabilities in telecommunications networks," In *International Teletraffic Congress - 13*, pages 253–260. Elsevier Science Publishers B. V. (North Holland), 1991.

[Kou93] Nick T. Koussoulas, "Performance analysis of circuit-switched networks with state-dependent routing," *IEEE Transactions on Communications*, 41(11):1647–1655, November 1993.

[Kri89] K. R. Krishnan, "Network control with state-dependent routing," In *ITC Specialists Seminar*, pages 1–6, 1989.

[Lat89] B. P. Lathi, *Modern Digital and Analog Communication Systems*, Holt, Rinehart and Winston, New York, 2 edition, 1989.

[LR91] F. Langlois and J. Régnier, "Dynamic congestion control in circuit-switched telecommunications networks," In *International Teletraffic Congress - 13*, pages 127–132. Elsevier Science Publishers B. V. (North Holland), 1991.

[LRS83] S. E. Levinson, L. R. Rabiner, and M. M. Sondhi, "An introduction to the application of the theory of probabilistic functions of a Markov process to automatic speech recognition," *Bell System Technical Journal*, 62(4):1035–1074, April 1983.

[MG92] Debasis Mitra and Richard J. Gibbens, "State-dependent routing on symmetric loss networks with trunk reservations-ii: Asymptotics, optimal design," *Annals of Operations Research*, 35:3–30, 1992.

[MGH93] Debasis Mitra, Richard J. Gibbens, and Baosheng D. Huang, "State-dependent routing on symmetric loss networks with trunk reservations-i," *IEEE Transactions on Communications*, 41(2):400–411, February 1993.

[MS91] Debasis Mitra and Judith B. Seery, "Comparative evaluations of randomized and dynamic routing strategies for circuit-switched networks," *IEEE Transactions on Communications*, 39(1):102–116, January 1991.

[Oht91] Masataka Ohta, "Fluid model for a traffic congestion prediction," In *International Teletraffic Congress - 13*, pages 665–670. Elsevier Science Publishers B. V. (North Holland), 1991.

[OK85] T. J. Ott and K. R. Krishnan, "State dependent routing of telephone traffic and the use of separable routing schemes," In *International Teletraffic Congress - 11*, pages 867–872. Elsevier Science Publishers B. V. (North Holland), 1985.

[Pan91] Alan Pankratz, *Forecasting with Dynamic Regression Models*, John Wiley & Sons, New York, 1991.

[Pea89] Barak A. Pearlmutter, "Learning state space trajectories in recurrent neural networks," *Neural Computation*, 1:263–269, 1989.

[PHK92] T. M. Peng, N. F. Hubele, and G. G. Karady, "Advancement in the application of neural networks for short-term load forecasting," *IEEE Transactions on Power Systems*, 7(1):250–257, 1992.

[Pin89] Fernando J. Pineda, "Recurrent backpropagation and the dynamical approach to adaptive neural computation," *Neural Computation*, 1:161–172, 1989.

[PP80] John R. Pierce and Edward C. Posner, *Introduction to Communication Science and Systems*, Plenum, New York, 1980.

[QSM92] Si-Zhao Qin, Hong-Te Su, and Thomas J. McAvoy, "Comparison of four neural net learning methods for dynamic system identification," *IEEE Transactions on Neural Networks*, 3(1):122–130, January 1992.

[RLS83] L. R. Rabiner, S. E. Levinson, and M. M. Sondhi, "On the application of vector quantization and hidden Markov models to speaker- independent, isolated word recognition," *Bell System Technical Journal*, 62(4):1075–1105, April 1983.

[Ros87] Sheldon M. Ross, *Introduction to Probability and Statistics for Engineers and Scientists*, Wiley, New York, 1987.

[SB94] Arvind Srinivasan and Celal Batur, "Hopfield/ART-1 neural network-based fault detection and isolation," *IEEE Transactions on Neural Networks*, 5(6), November 1994.

[Sch87] Mischa Schwartz, *Telecommunications Networks*, Addison-Wesley, New York, 1987.

[SM92] Padhraic Smyth and Jeff Mellstrom, "Detecting novel classes with applications to fault diagnosis," In *International Conference on Machine Learning 9.* Morgan Kaufmann, 1992.

[Smy93] Padhraic Smyth, "Hidden Markov model and neural networks for fault detection in dynamic systems," In *1993 Workshop on Neural Networks and Signal Processing*, 1993.

[Smy94] Padhraic Smyth, "Markov monitoring with unknown states," *IEEE Journal on Selected Areas in Communications*, December 1994.

[SP90] R. Sharda and R. Patil, "Neural networks as forecasting experts: an empirical test," In *International Joint Conference on Neural Networks*, volume 1, pages 491–494, Washington D. C., 1990.

[TdAF91] Zaiyong Tang, Chris de Almeida, and Paul A. Fishwick, "Time series forecasting using neural networks vs. Box-Jenkins methodology," *Simulation*, 57(5):303–310, November 1991.

[UM91] Joachim Utans and John Moody, "Selecting neural network architecture via the prediction risk: Application to corporate bond rating prediction," In *First International Conference on Artificial Intelligence Applications on Wall Street.* IEEE Computer Society Press, 1991.

[vLW92] J. H. van Lint and R. M. Wilson, *A Course in Combinatorics*, Cambridge University Press, New York, 1992.

[WG94] Andreas S. Weigand and Neil A. Gershenfeld, editors, *Time Series Prediction: Forecasting the Future and Understanding the Past*, Santa Fe Institute Studies in the Sciences of Complexity. Addison-Wesley, New York, 1994.

[Wil56] Roger I. Wilkinson, "Theories for toll traffic engineering in the USA," *Bell System Technical Journal*, pages 422–514, March 1956.

[WZ89] Ronald J. Williams and David Zipser, "A learning algorithm for continually running fully recurrent neural networks," *Neural Computation*, I:270–280, 1989.