PH.D. THESIS

Fault Tolerant Rerouting in Broadband Multiclass Networks

by A.I. Vakhutinsky
Advisor: M.O. Ball

CSHCN Ph.D. 96-3
(ISR Ph.D. 96-12)

The Center for Satellite and Hybrid Communication Networks is a NASA-sponsored Commercial Space Center also supported by the Department of Defense (DOD), industry, the State of Maryland, the University of Maryland and the Institute for Systems Research. This document is a technical report in the CSHCN series originating at the University of Maryland.

Web site http://www.isr.umd.edu/CSHCN/

Fault Tolerant Rerouting in Broadband Multiclass Networks

by

Andrew I. Vakhutinsky

Dissertation submitted to the Faculty of the Graduate School
of The University of Maryland in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
1996

Advisory Committee:

Professor Michael O. Ball, Chairman/Advisor
Professor John Baras
Professor Bruce L. Golden
Assistant Professor Bharat Kaku
Professor Prakash Narayan

Abstract

Title of Dissertation: Fault Tolerant Rerouting in Broadband Multiclass Networks

Andrew I. Vakhutinsky, Doctor of Philosophy, 1996

Dissertation directed by: Professor Michael O. Ball
Institute for Systems Research and
College of Business and Management

Modern broadband integrated service digital networks (B-ISDN) must handle multiclass traffic with diverse quality of service (QOS) requirements. The main purpose of our research is to design call rerouting mechanisms which provide rapid restoration of network services in case of link failures. We suggest two approaches: virtual circuit (VC) and virtual path (VP) reroutings. The first approach is more reactive while the latter is more proactive. The applicability conditions for the first approach include the availability of a layered network structure similar to VC/VP architecture which is widely accepted in asynchronous transfer mode (ATM) networks. Another applicability condition is the extent of network failure: VP level restoration is designed for single link failures – the most common in the telecommunication networks. On the other hand, in case of less predictable multiple link failures, VC-level rerouting is appropriate. These two rerouting approaches vary in the amount of time required to carry them out. Though both schemes are designed to work in real time, VP-level rerouting tends to be faster and can be performed in an on-line mode using pre-computed paths. VC-level rerouting requires real-time computation of routes which may result in a noticeable impact on some services. On the other

hand, VP-level rerouting requires a substantial amount of off-line computation to design the VP layout and the backup routes.

In this dissertation we propose a new model and associated algorithms to solve a VC-rerouting problem in real time. This model takes advantage of the distributed network data and computational resources by decomposing the problem at an early stage and then performing the computations in a decentralized mode.

In order to solve the fault tolerant VP layout problem, we formulate a bi-criteria optimization model reflecting the tradeoff between throughput and certain QOS requirements. The model involves a piece-wise linear approximation to the capacity allocation rule for variable bit rate connections statistically multiplexed over a VP.

Both models are formulated as integer programs. The solution method developed employ relaxation and aggregation of variables, feasible solution heuristics and valid inequalities. The results of the computational experiments presented indicate that the methods developed are efficient and produce accurate solutions.

© Copyright by

Andrew I. Vakhutinsky

1996

Dedication

To my parents

Acknowledgements

I express my gratitude to professors at the University of Maryland, College Park: Arjang Assad, John Baras, Michael Fu, Saul Gass, Bruce Golden, Bharat Kaku and Armand Makowski, who not only taught me in their courses but were always ready to discuss various research matters with me. I am deeply obliged to my academic advisor and mentor Michael Ball who is the architect of my dissertation. I appreciate Mike's efforts for improving my writing style. I am grateful to my committee members for investing their time in reading my dissertation and attending my thesis defense. I acknowledge Institute for Systems Research (ISR) for providing a creative environment of cross-disciplinary research.

I also thank my fellow students for that special atmosphere of comradeship which kept my spirits up during the years at the university. My special thanks to Anindya Datta and Sridhar Bashyam for their advice and encouragement. Finally, I thank Tam Thompson for her constant moral support and understanding.

This work was carried out as part of the network management group project in the ISR. The group received funding from the ISR's NSF core grants CDR-8803012 and EEC-9402384. The initial concept of this work was developed as part of a project sponsored by IBM. Later portions of the work were funded by the NASA Center for Satellite and Hybrid Communication Networks under NASA contract No. NAGW-2777S, the Hughes Network Systems/Maryland Industrial Partnership contract No. 182918 and the Army Federated Laboratory – ARL grant No. DAAL01-96-2-0002.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

The research described in this dissertation involves modeling a broadband multi-class telecommunication network which is tolerant to the failures of its elements. Fault tolerance is the ability of a telecommunication network to restore its operations with minimal impact on the provided services after a network element, such as link or node, fails.

The networks under consideration are broadband integrated services digital networks (B-ISDN) which are designed to support a wide variety of service classes including but not limited to voice, data and video in the digital form. These classes are different in their required bandwidth (varying from 64Kb/s for voice service to several Mb/s for some video applications and data transmission), traffic speed variation (they can be constant bit rate (CBR) or variable bit rate (VBR), the latter may have extremely high peak to mean bit rate ratio) and delay tolerance.

Modern telecommunication networks providing these services represent a variety of systems. Among this variety we specifically consider large, geographically dispersed networks. These networks generally use fiber-optic high capacity

links. In fact, a physical link in the modern telecommunication network may be capable of transmitting at speeds in the Gb/s range. Several types of network architectures have been proposed, or are under a development, to support B-ISDN. Probably the most prominent is Asynchronous Transfer Mode (ATM). We use some ATM specific assumptions in our work but our models apply to other cases as well. The first assumption is a connection-oriented fixed length packet switching mode of operation which means that the information transmission flow between the end users is implemented in the form of packets, or cells, having the same size and following the same route, or sequence of links. The second assumption is the availability of a layered network structure similar to Virtual Channel/Virtual Path (VC/VP) architecture which is widely accepted in ATM networks. This type of architecture allows for the construction of what can be considered as additional logical links in the network. Our modeling assumptions concerning telecommunication networks are described in more detail in chapter 2.

In order to create a testbed for the proposed call rerouting algorithms and compare performance measures of different VP layouts, we perform an on-line call routing simulation which provides us with a meaningful distribution of call routes in a network. This simulation method is described in chapter 3. During the simulation process, incoming calls of a certain type, origin and destination are generated according to Poisson distribution. Call holding times are generated as exponentially distributed. We consider calls of different types (data, voice and video) arriving with different rates between different node pairs. Each call type may have different bandwidth demand and mean holding time. This simulation stops when the network reaches a steady state. An embedded routing algorithm

computes a call path between origin and destination taking into account call parameters and current network status. We ran our simulation for different parameter settings varying the steady state network utilization level and call type mixtures. The distribution of the calls in the network is used as input data for computational experiments conducted to evaluate the quality of the VC-level rerouting algorithms proposed in chapter 4. It is also used in chapter 6 to measure an adaptability of a VP layout by calculating call blocking probability when the offered traffic is increased as compared with the projected level.

The layered architectures of ATM networks make it possible to reroute calls on different levels. We consider two alternatives: virtual circuit (VC) level and VP level rerouting. In the first case each call is treated as a separate entity, in the second case we reroute entire VPs to pre-established backup routes bypassing the failure. Generally, VP level rerouting is preferred since it can be carried out much faster than VC level rerouting. This can be explained by the fact that the number of VPs is normally much lower than the number of VCs and once a backup VP is pre-established, a required amount of capacity can be allocated to it in real time. However, VP level rerouting requires that certain conditions are satisfied. First, the backup VPs with zero capacities are to be established together with the primaries before a failure occurs. Second, backups should not be affected by the same failure which disconnects the corresponding primaries. The latter fact makes the VP rerouting non-applicable in case of multiple link failures affecting large parts of the network. Also, VP level rerouting is less adaptive and uses bandwidth less efficiently. Thus it is less advantageous when the call demand is not steady or not predictable. Finally, VP level rerouting can be performed only in those networks which support the layered architecture.

On the other hand, VC level rerouting has broader applicability but the delays required to find the VCs and set them up will be greater than the delays for VP level rerouting.

In chapter 4 we present an approach to real-time VC rerouting. We assume that there are several priority classes of calls and lower priority services can be preempted to provide bandwidth for the higher priority traffic. We formulate the problem as an integer program (IP), considering it as a multicommodity flow problem with side constraints and zero-one variables, and develop a real time approach to solve it. Our model takes advantage of the distributed network data and computational resources by decomposing the problem at an early stage and then performing the computations in a decentralized mode. Specifically, a two-phase process is proposed. In the first phase, a centralized algorithm calculates a capacity allocation solving a linear relaxation of the problem. This allocation is then distributed to each node. In the second phase, each node calculates paths, in parallel, for the calls it originated. In the process of rerouting calls, our approach allows for the possible preemption of existing calls based on call priorities. Computational experiments demonstrate the validity of our approach.

Chapter 5 is devoted to VP level rerouting. There we consider four different scenarios for a reliable, fault tolerant VP layout. These scenarios vary in two characteristics: (1) whether the backup VP is end-to-end link-disjoint with the primary VP or, instead, a link-specific backup VP is created for each link on the primary VP, and (2) whether the backup has the same or lower capacity relative to the primary VP. For each scenario we give an IP formulation. We also consider the VP layout problem as bi-criteria optimization problem which involves trading off between two performance parameters:

1. delay and control overhead minimization

   and

2. throughput maximization.

In Chapter 6 we develop various IP techniques to solve the problems formulated in Chapter 5. Since these IP problems are quite large and extensive, it appears impractical to attempt to solve them to optimality. Instead, we develop procedure to find high quality feasible solutions. Our approach is similar to that described in Chapter 4 and [6] and involves aggregation of variables, linear relaxation, valid inequalities and rounding heuristics. However, as opposed to the VC rerouting problem applications which are essentially real-time, VP layout is normally performed off-line. Consequently, when using this approach in practice, therewould be more computation time available. The output to the solution process is a Pareto-optimal solution set, i.e. the set of solutions in which neither of the parameters can be improved without worsening the other.

We performed a set of computational experiments having a two-fold goal:

1. to demonstrate a trade-off between VP layout parameters;

2. to estimate the quality of our approximate IP solving methods.

In the first case, we described approximate Pareto-optimal solution sets for model networks by providing upper and lower curves for the set of exact optimal solutions. In the second case, we presented an integrality gap (i.e. gap between feasible and linear relaxation solutions) and computation times.

Another performance measure which we consider in our research is adaptability of a particular VP layout to changes in traffic demand. Quantitatively, we measure adaptability in terms of call blocking probability. Our conjecture

was that longer, lower capacity VPs have lower adaptability than shorter, higher capacity VPs. In order to validate this conjecture, we simulated random generation of calls which were routed through the network using previously computed VPs. The results validated our conjecture and quantified the extent of this effect.

Chapter 2

Foundations of Broadband Multiclass Networks and Modeling Assumptions

In this chapter we give a brief review of the concepts underlying Broadband Multiclass networks, or using more common terminology integrated services digital networks (B-ISDN). Recent monographes and reviews [35, 25, 41] give an excellent introduction into this topic.

The main purpose of B-ISDN design is to support a wide range of voice and non-voice applications in the same network. These networks extend the concepts of telephone networks by incorporating additional functions and features of current circuit and packet-switching networks for data to provide both existing and new services in an integrated manner. The first set of ISDN recommendations was adopted by CCITT in 1984. The primary rate access interfaces were defined with total bit rates of 1.544Mb/s (i.e. T1 bandwidth) and 2.048Mb/s (i.e. E1 bandwidth) including a 64Kb/s signaling channel. However, it was soon realized that higher bit rates are required for applications such as interconnection of local area networks (LAN), video, image, etc., bringing the standardization process to the introduction of of broadband concepts. B-ISDN is conceived as an all-

purpose digital network. Activities currently under way are leading to creating a worldwide network facilitating multimedia information exchange between any two subscribers. CCITT Recommendation I.113 defines "broadband" as "a service or system requiring transmission channels capable of supporting rates that are greater than the primary access rate". Currently, B-ISDN interfaces support up to 622 Mb/s with the possibility of higher rates in the future.

The network transport mode defines how information supplied by the network users is eventually mapped onto the physical network. Asynchronous Transfer Mode (ATM) is the transport mode of choice for B-ISDN. Though our research network model is not limited to ATM, we find it useful to highlight the main features of this mode since this is a typical and probably the best defined example of the multiclass telecommunication network. Below we highlight some of the most prominent features of ATM networks.

- **Connection-oriented network.** This means that once a connection is established, the same route through the network is used for the entire holding time. However, the path used by the connection depends on the state of the network and usually has to be determined before the connection is established.

- **Packet-switched network.** The call traffic is transmitted as a stream of fixed-size packets, referred to as ATM cells. An ATM cell is 53 bytes long, consisting of a 48-byte information field and a 5-byte header. Each header contains routing information applied to switching at the cross-connect nodes.

- **Hierarchical routing.** Two types of routing concepts are used in ATM

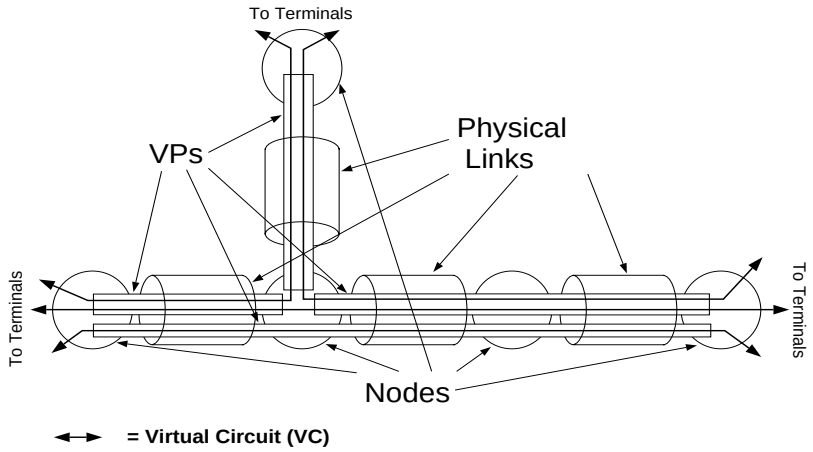

Figure 2.1: Layered Architecture of ATM Networks (as it appears in [9])

networks: virtual circuits (VC) and virtual paths (VP). Before transmission can start, a VC has to be established between end nodes. The VP is a more permanent structure. It can be viewed as a logical link in the network in the sense that it provides a direct connection between a pair of nodes using a pre-established route and has its own fixed bandwidth. VPs normally carry several VCs multiplexed together whose total bandwidth should not exceed the total bandwidth of the VP. Figure 2.1 depicts this hierarchy. Notice that a VC can generally traverse more than one VP. The VP and VC being traversed by a cell are defined by fields in the header called virtual path identifier (VPI) and virtual channel identifier (VCI), respectively. These fields are used for switching each cell between links upon the cell's arrival at an intermediate switching node.

We now consider how features of ATM networks affect our model.

2.1 Quality of Service and Statistical Multiplexing

The networks under consideration carry different classes of calls. Each call by its nature may be constant bit rate (CBR) or variable bit rate (VBR). As an example, consider three main call types: voice, data and video. Voice calls are generally CBR (unless sound compression is involved), may tolerate comparatively high cell loss (usually 10^{-6}) and require rather low bandwidth ($64Kbit/s$), but delay more than $0.1s$ would make necessary installation of sophisticated echo suppression equipment and it is not desirable. On the other hand, data calls can tolerate rather high ($10s$ or more) delays but cell losses require retransmission of the cells. Also, data calls are usually VBR with sometimes very high (10^6 or greater) peak/mean rate ratio; bandwidth requirements can vary substantially for different applications. Video calls are, similar to voice calls, real-time services which implies low delay tolerance and no retransmission of lost or corrupted cells, but unlike voice calls they are VBR and have very high bandwidth requirements (up to $8Mbits/s$). Each VBR call is characterized by its mean and peak transmission rates. These two parameters are negotiated by the call admission mechanism and can not be exceeded for the entire holding time of the call.

ATM networks provide certain Quality of Service (QOS) parameters for each type of calls. We consider two most important QOS parameters:

1. **Cell Loss Ratio (CLR)**. Each fixed capacity VP normally carries several VBR VCs statistically multiplexed together (i.e. sharing the VP capacity). In the event when total VC transmission rate exceeds VP capacity some

cells are dropped (lost). CLR is defined as a ratio of the lost cells number to the total number of cells transmitted. In ATM networks CLR is usually a very small parameter ranging from 10^{-12} to 10^{-6}.

2. Cell Transmission Delay consists of propagation, and switching/queuing delay. The first is negligibly low unless satellite links are involved. The second part can be reduced by eliminating switching in intermediate nodes. It can be accomplished by low hop routing and/or using end-to-end VPs.

Let us consider how the two QOS parameters are influenced by the buffer size and output link rate. CLR is approximately equal to the probability that the buffer is full upon arrival of a cell. Call transmission delay is basically the sum of a relatively small propagation delay and a queuing delay which is determined by the average length of the buffer queue found upon the cell arrival. Thus increasing the buffer size would reduce the CLR but could result in the queuing delay becoming prohibitively large. On the other hand, increasing the output link rate reduces both CLR and call delay but may lead to inefficient use of the network resources. Clearly, the output link rate can not be less than total mean arrival rate. Consider two marginal cases. If the output link rate is set exactly to mean rate and buffer size is extremely large, then CLR is close to zero but transmission delay can be rather large, sometimes reaching hours as is the case with the Internet. If the output link rate is set to the total peak rate, then even a zero length buffer would provide zero CLR but the link would be idle most of the time, since for majority of the services, peak to mean rate ratio is high. In order to optimize ATM network design, satisfying both low CLR and a transmission delay criteria, and giving an acceptable utilization of network resources, the choice is made for sufficiently small buffer size and an output link

rate which is between total mean and peak arrival rates taking advantage of statistical multiplexing gain. This approach results in low CLR (less than 10^{-6}) and provides real-time services with transmission delay of about 0.1 s.

Statistical multiplexing is one of the main advantages of ATM networks over more traditional networks like circuit-switched and synchronous transfer mode (STM) networks. The result of statistical multiplexing is lower required bandwidth per call without QOS degradation when several non-correlated VBR calls are multiplexed together on a link. Since the transmission rate distribution of a general real source is not well understood, there is no complete analytical model for the statistical multiplexing. Several methods were designed to determine the amount of capacity to be allocated in order to satisfy QOS requirements. Among them we mention equivalent capacity approach by Gun et al [16, 17] and application of the Shannon noiseless coding theorem to its calculation in [44].

In chapter 4, we develop a call rerouting model where the total capacity allocated to a group of calls is computed by adding the calls' effective bandwidths ([16]). Later, in chapters 5 and 6 we consider a more complex model which involves a capacity allocation function that models statistical multiplexing effect using a piece-wise linear approximation.

2.2 Rerouting Mechanisms

There are various rerouting protocols in modern telecommunication networks. In this section we outline the assumptions we make about a rerouting procedure on both the VP and VC levels. Table 2.1 gives a brief comparison between the VC- and VP-level in terms of various issues such as approach to rerouting,

applicability, performance time, amount of spare capacity required and likelihood of call refusal.

Issue	VC-level	VP-level
Type of approach	reactive	proactive
Applicability	wide	two-layer architecture
	multiple failures	single link failures
Performance time:	quasi-real	real-time
on-line computations	large	minimal
off-line computations	low	large
Spare capacity	lower	higher
Call refusal	possible	unlikely

Table 2.1: Comparison between VC- and VP-level Rerouting

Chapter 5 addresses the problem of laying out primary and backup VPs such that the failure restoration is restricted to the VP level. It is known [21, 22] that in ATM networks it is possible to establish a zero bandwidth VP between a pair of nodes. This VP is used as a backup VP for a corresponding primary VP. We assume that each end node of the backup VP can be a terminating node of the primary VP as well as its intermediate node. The key assumption of the VP rerouting scheme is a simple but effective restoration procedure which can be performed with a high degree of rapidness and reliability as soon as a failure occurs. In this case the failed primary VP is switched to its backup as described below.

Consider a link failure in a network as shown in Figure 2.2a. We assume

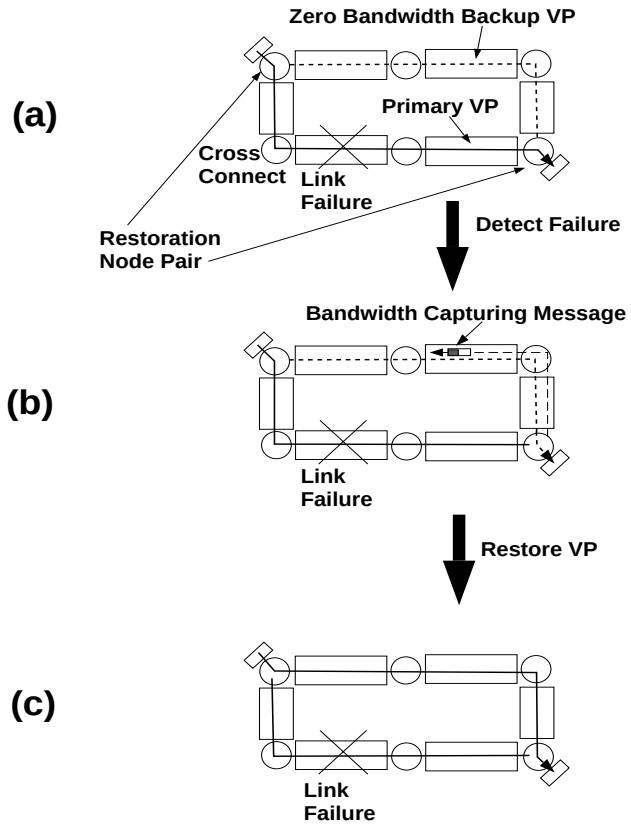

Figure 2.2: Failure Restoration Process (as it appears in [21])

that each network element (NE) maintains a local data base for managing the information of spare resource utilization of the links connected to the NE, and that terminating NEs hold the pairing of occupied VPs and their backup VPs. If a node receives a restoration message, it performs the following procedure as a transit node (Figure 2.2b):

if spare bandwidth $\geq$ required bandwidth
then {
    capture appropriate bandwidth in the data base;
    retransmit the message to the next NE;
}
else {
    return "Insufficient Bandwidth" message
    to terminate the process.
}

When the upstream side node of the restoration pair receives the restoration message, it switches traffic from the failed VP to the backup VP; this completes failure restoration for this VP (Figure 2.2c). This restoration algorithm should be considered as very basic. In chapter 5 we propose two enhancements:

1. If there is not sufficient bandwidth to reroute an entire VP, some protocols may be capable of taking down some of the VCs they are currently carrying.

2. Additional routing flexibility can be achieved by initially routing traffic on both primary and backup VPs. Then, when failure occurs, traffic is switched from one VP to the other. In this case there is practically no

distinction between primal and backup VPs and they can be thought of as "mirror" VPs.

In any of the preceding variants, once the backup VP is established with appropriate capacity, it is a relatively simple matter to switch traffic to the backup VP. There would be a simple entry modification to the routing table at the beginning VP node so that all cells whose outgoing VPI was the primary VP would have the outgoing VPI be the backup VP after the primary VP fails. The first proposed enhancement would require the installation of a sort of filter which dropped all cells with particular VCIs (or setting their cell loss priority bit to one) after switching to lower capacity backup VP. The second proposed enhancement can be considered as part of the basic algorithm where the new outgoing VPI of the cells using failed VP would be the same as VPI already in use by the cells routed into the "mirror" VP.

In lower speed private networks (our primary focus in chapter 4) which might use ATM or fast packet switching (where VPs do not exist), we envision an environment in which backup bandwidth will not be reserved necessitating the preemption of lower priority calls in the event of element failures. To carry out the appropriate restoration, VC/call level rerouting together with call preemption will be required. However, even in the case of ATM networks, where full use is made of the VP concept, this routing scheme makes sense also (for VP rerouting), since a single trunk failure can disrupt up to 4096 VPs, all of which would require rerouting with very high probability, and all of which will be difficult to place on other trunks because of the (in all probability) large capacity of each. In such cases, dynamic rerouting, rather than a simple backup scheme, could produce substantially improved results.

Chapter 3

Testbed for the Evaluation of Rerouting Algorithms and Strategies

In this chapter we describe an aggregate level model for generating multi-media calls. This model serves two intended purposes. The first is to generate call data to be used to compare routing algorithms and routing strategies. The second is to generate a sample list of calls affected by a link failure which is used as an input to VC-level rerouting algorithm.

3.1 Model Description

We consider a connection-oriented telecommunication network where call requests arrive according to a certain distribution. The call is admitted if there is a route between its origin and destination nodes satisfying its bandwidth requirements. Each session (or call) uses the same route for its entire holding time.

The simulation procedure consists of generating random call request parameters and applying a routing algorithm to establish a connection. A call request

may be rejected or blocked if the routing algorithm can not find an available, feasible route.

Arriving calls belong to different classes (e.g. voice, data, or video) and each given node pair may have its own mixture of different classes. For each call the following parameters are randomly generated:

1. origin and destination;

2. required bandwidth;

3. arrival and holding times.

We assume there is a set H of different call types. These types may differ in quality of service (QOS) requirements including bandwidth, maximum tolerable delay, maximum cell loss probability. For the sake of simplicity, we assume that both service request interarrival times and holding times are exponentially distributed, though it is not a binding factor in our model. Different pairs of nodes have different arrival rates for each of the services, denoted by λ_{ij}^h for node pair (i, j) and call type h. Each call type h has mean holding time $1/\mu_h$, the same for each node pair.

Each network node i is assumed to have an associated parameter called node population denoted by T_i. The origin and destination of an arriving call are chosen randomly based on the following probabilities:

$$\begin{aligned} Prob\{\text{Origin} = i\} &= \frac{T_i}{\sum_{\text{all } k} T_k} \\ Prob\{\text{Destination} = j \mid \text{Origin} = i\} &= \frac{T_j}{\sum_{\text{all } k, k \neq i} T_k} \end{aligned}$$

This produces a mix of calls such that number of calls outgoing/incoming from/to a particular node proportional to its population.

For each call, its origin and destination and required bandwidth are used as an input into the routing algorithm. It is assumed that the routing algorithm has access to network status information such as the load on each link. Based on that information a decision on call routing is made. If the call is routed, the network information is updated by subtracting the call bandwidth from the capacity available on links carrying the call. If there is not enough capacity to route the call, the call is blocked. A routed call stays in the network for its holding time. After that it is released from the network and the network status is updated.

Each call is routed through the network by an on-line algorithm. The on-line algorithm deals with the call requests one-by-one, accesses only current network status information, and can not use any a priori knowledge of the entire request sequence. The theory of the on-line routing algorithms is well studied in the literature (c.f. [40, 36]). We adopt one of the algorithms proposed in [13]. Essentially, this algorithm is a type of shortest path algorithm where link length is chosen by exponential scaling of its congestion. The algorithm can be described briefly as follows:

A call between origin O and destination D is routed over O—D path P, **if**

1. P is a min-hop path;
2. $u_e + r/B_e < 1$ for all $e \in P$;
3. $\sum_{e \in P} \kappa^{u_e} \leq \kappa$.

where u_e is a relative congestion of link e, B_e – its capacity and $1 < \kappa < 2$ is a parameter.

The idea behind this approach is to introduce exponential penalties for high link utilization and not to allow an excess of the link capacity.

It was proved in [4] that this algorithm performance (in terms of the throughput) is at least $1/O(\log n)$ of the best off-line algorithm performance, where n is the network size. Though this ratio does not seem sufficiently small from a practical point of view, simulations show that this strategy outperforms traditional minimum-hop and reservation-based strategies.

We evaluate the network performance by the call blocking ratio. Thus the goals mentioned in the introduction to this chapter are accomplished in the following ways: In order to compare different routing strategies and algorithms, the same stream of calls is generated for all simulation runs but on each run a different routing algorithm is used. The quality of the routing algorithm is evaluated by the call blocking probability. In order to obtain a sample set of VCs affected by a link failure for the VC-level call rerouting algorithm, the simulation is run until it reaches a steady state and then we extract a list of calls carried by a particular link which is considered as failed.

3.2 Implementation

The simulation is event driven. That is, we maintain a list of events and keep extracting the minimum time event from the list. We also keep updating the list. The only events stored in the list are calls departures, since each arriving call is either routed immediately or blocked. Thus, there is no queue of arrivals and we have to keep track only of the next call arrival. If the next event is call arrival, we try to route the call. If the call is blocked, we do not consider it

anymore; otherwise, we keep its route and store its departure time in the list of events. If the next event is call departure, the call is extracted from the list and its route is removed from the network. The simulation stops after a prespecified number of calls is generated.

Below, we give more formal description of the algorithm as a pseudocode.

Inputs to the Algorithm

1. simulation length
2. network information
   (a) node list
   for each node:
      i. node population
      ii. call type mix
   (b) link list
   for each link:
      i. end nodes
      ii. link capacity
3. call type list
   for each call type:
   (a) required bandwidth
   (b) mean holding times

Outputs from the Algorithm

1. call list

   for each call:

   (a) *node_or* (origin node) and *node_dest* (destination node)

   (b) call type

   (c) $traff$ (mean traffic rate for call)

   (d) VC – given by $link_1, link_2, \ldots, link_n$

The Algorithm

```
1.   CallArrival = 0;
2.   for Counter = 0 to SimLen
3.       if (EventList is empty OR CallArrival<MinTime(EventList))
4.           GenerateCall()
5.           if (FindRoute(CallParameters) == SUCCESS)
6.               CallDeparture = CallArrival + HoldTime(μ);
7.               AddCallToList(EventList);
8.               UpdateLinkInfo(Network, CallParameters);
9.           else block the call;
10.          CallArrival = CallArrival + Interarrival(λ);
11.      else
12.          CallRelease(Network, CallParameters);
13.          RemoveCallFromList(EventList);
```

VARIABLES:

`Counter`	event counter;
`SimLen`	simulation length (number of events);
`CallArrival`	arrival time of the next call;
`CallDeparture`	departure time of the next call;
`CallParameters`	call parameters (type, bandwidth);
`Network`	network data;
`EventList`	list of currently active calls.

FUNCTIONS:

`Interarrival()`	generates interarrival time;
`GenerateCall()`	generates call request;
`HoldTime()`	generates holding time of the call;
`FindRoute()`	finds a route for the call;
`AddCallToList()`	adds call to the list of currently active calls;
`RemoveCallFromList()`	removes call from the event list;
`MinTime()`	finds the minimum departure time;
`CallRelease()`	releases the bandwidth capacity on the links;
`UpdateLinkInfo()`	adds call to each carrying link.

Call arrival times are computed by adding a randomly generated interarrival time to the last arrival time (line 10). The first arrival is set to 0 (line 1). Interarrival times are generated as exponentially distributed random variates using the inverse-transform method within function `Interarrival`(λ). The departure time of a call is computed as its arrival time plus a randomly generated holding time. The holding time is generated as an exponentially distributed random variate using the inverse-transform method within function `HoldTime`(μ).

In the beginning of any simulation loop (line 2), we determine the next event by comparing the next arrival time and the minimum departure time from the event list (line 3). If the next event is arrival, the call request is generated (line 4) and the routing function is called to compute its route (line 5). If the call gets through (i.e. a feasible route is found), the departure time of the call is computed (line 6), the call departure time is added to the list of events (line 7), and its parameters are added to the lists of calls routed through the links carrying the call (line 8). If the call is blocked, it is disregarded by the algorithm.

If the next event turns out to be call departure, then the call is taken down (line 11). That means releasing capacity bandwidth on the links used by the call (line 12) and removing the call from the event list (line 13).

The simulation ends when the number of events defined by the event counter exceeds the simulation length.

Chapter 4

Distributed VC-level Call Rerouting in Multiclass Broadband Networks

In this chapter we present mechanisms for rerouting around faults in multiclass traffic environments which will be encountered in most of the future ubiquitous broadband integrated service digital networks (B-ISDN). These networks should deliver service for calls with widely varying bandwidth and quality of service (QOS) requirements. In order to provide high service availability, it is necessary for the networks to react in real-time to network faults, the topic of our research.

The rerouting problems treated in this chapter involve the simultaneous calculation of routes for several calls. Our approach differs from classical which is routing in circuit-switched networks [23] in that we are most concerned with satisfying a large group of calls and the variety of bandwidth requirements giving the problem a "packing" dimension. That is, calls with varying bandwidth must be packed into bandwidth limited links. The "bandwidth packing problem" has been studied in the literature [24, 37, 38]. Previous research has concentrated on finding an optimum or near-optimum solution by applying off-line algorithms. In this chapter we develop a model and associated algorithms designed to oper-

ate in a distributed environment. Specifically, a two-phase process is described. In the first phase, a centralized algorithm calculates a capacity allocation. This allocation is then distributed to each node. In the second phase, each node calculates paths, in parallel, for the calls it originated. In the process of rerouting calls, our approach allows for the possible preemption of existing calls based on call priorities.

Routing standards and schemes typically address large-network issues by network aggregation techniques. The two-phase routing scheme proposed in this chapter (which does not explicitly consider network aggregation) has immediate applicability within subnetworks. These "subnetworks" themselves may be quite large–for example, one could view a corporate network (such as IBM's) as a subnetwork. In such cases, the "local graph" of such a scheme as the Private Network-to-Network Interface (PNNI) is large enough to present the challenging problems that we are trying to solve here. For interworking between subnetworks (i.e. where other subnetworks are seen only in an aggregated sense) this scheme would have to be adapted in two ways: First, it would have to be adapted to multilayered rerouting, since as it stands in this chapter it addresses only a single network layer. Second, we would have to find an appropriate way of representing the allocations in an aggregate form for higher layer clusters. While we have not yet done this work, we expect that multilayer schemes, like the PNNI, can be accommodated with reasonable effort.

Another unique characteristic of the problem we consider is allowing the possibility of preempting existing calls. Preemption is used in private networks in order to ensure that service to the highest paying customers is disrupted least and established or restored most quickly. For virtual path networks (VPN)

by public carriers, preemption can be an effective mechanism for sharing links between casual customers and customers to whom service is guaranteed as part of a VPN contract. Such a priority mechanism can ensure that the capacity guaranteed to a VPN customer is always available.

The chapter is structured as follows: section 4.1 gives a model description and general approach. The algorithms are described in section 4.2, computational experiments are presented in section 4.3 and section 4.4 summarizes the chapter contributions.

4.1 Problem Formulation

4.1.1 Assumptions and Fault Scenario

The basic scenario we are presented with is the failure of a network link, e, in an environment in which we may assume all network elements, including e, are currently handling significant amounts of traffic. In general, at the time of its failure, several calls will be routed over e. We denote the set of these calls by $\mathcal{K}$. The problem to be addressed is the determination of new routes for all calls in $\mathcal{K}$. Due to the dynamics of the network information used by the originating node when setting up the call, it is possible that when the call setup is actually carried out, it will be refused by an intermediate node due to the unavailability of sufficient resources on a link. We call this phenomenon a *call refusal*. Alternatively, it is possible that in order to set up the incoming call it is necessary to take down the route of an existing lower priority call. We call this phenomenon *call preemption*. In such cases, the originating node of the preempted call would try to find a new route for that call. We note that each

intermediate node along a call's path contains a limited amount of information concerning the call, including a call id, the originating node, call bandwidth and outgoing link. Certain rerouting strategies might require that this information be augmented.

We consider two failure scenarios for a network link, e. Under the first scenario, the network management system anticipates the failure of e and the system has sufficient time to plan route adjustments accordingly. Under the second scenario, link e experiences a sudden failure. In the first case, we could assume that the network management system has a "reasonable amount" of time to react to the failure and in the second, the network must react much more quickly. In either case, we assume that several calls are routed over e.

The call model is as follows: Each (single connection) call is composed of two paths: A forward path (relative to the call originator) and a return path. When the call is initially set up, both the paths are established by the originator, though the two paths may indeed be different. The two paths then may be rerouted independently of one another and may be preempted independently of one another. If one of the paths, however, fails to be rerouted when necessary, the call fails (unless, of course, the communication is only in one direction). We think that this gives the network great flexibility because it allows the network to find feasible routes for connections that might otherwise not be found, thereby enhancing perceived network availability for customers. In addition, in case of link failure, communication is disrupted only in one direction. Finally, in case of dynamic bandwidth adjustment, failure to find the necessary bandwidth for a path in one direction does not necessitate the rerouting of the entire call (i.e. both directions). To simplify the formulation and analysis given, we do not

explicitly address the correlation of the forward and reverse paths in our model, i.e. we proceed as if each call requires a path in one direction only. Our overall scheme can address this correlation with some modest modifications which we do not include here.

We assume routing computations are distributed throughout the network so that the node at which the call originates calculates a route for the call. Specifically, under normal conditions, the call-origination node selects a route for each call as each individual call request is received by the node. The route calculations are based on link load information passed to the node periodically by other nodes. We associate a fixed bandwidth requirement with each call. Thus our model most directly applies to circuit switched networks where each call may have an arbitrary bandwidth requirement. However, under appropriate conditions, it is also applicable to ATM networks and fast packet switching networks with variable length packets. We make a key assumption with respect to additivity of bandwidth; that is, the total bandwidth used by the calls on each link is sum of all bandwidths associated with calls routed over the link. In order to apply the model to ATM and fast packet switching networks at the call/virtual circuit (VC) level, one would use the call's effective bandwidth ([16]) as its bandwidth requirement. The additivity assumption would imply that effective bandwidths over a single link can be added to test the capacity constraint. There is currently a large amount of active research (e.g. [17]) related to this issue and the potential error that might be introduced by doing so. We will not discuss details here but only point out that the applicability of our model in this context clearly is impacted by the results of this research. It is also possible to apply our model to the determination of routes for virtual paths

(VPs) in ATM networks. Since fixed bandwidths are associated with virtual paths, the additivity assumption appears to be valid.

Our approach makes use of a certain capacity allocation scheme: Each call origination node is given its own capacity allocation for each link in the network. The capacity allocated to a particular call origination node will typically be substantially less then the total capacity available on the link. The path selection algorithms that the call origination node uses will penalize any violation of the capacity allocation. When a link fails, a designated node will compute the capacity allocation for all call origination nodes that have routed calls over the failed link. This capacity allocation will be based on an aggregate rerouting of all affected traffic. The capacity allocation will then be transmitted to each origination node which will use the allocation when determining new routes for all calls previously routed over the failed link. The motivation for this approach is to reduce the incidence of call preemption and call refusal that would normally take place when failures occur. Specifically, it is anticipated that, when failures occur, since the origination nodes are trying to carry out "bulk" call setup, the information they would have on the traffic load on links would be inaccurate. Thus, it would be much more likely that call setups would be attempted on links that did not have sufficient capacity. The intent is to reduce the occurrence of this phenomenon by effectively allocating the link capacity ahead of time.

In this chapter, we describe the underlying optimization problems, propose basic solution approaches for each and evaluate the merits of the two-phase approach in determining high quality routes. We understand that several additional issues must be addressed in order to make this approach practical. These include:

- determination of what additional information must be stored by each node;
- thorough analysis of capacity allocation step, specifically – evaluation of alternative locations for computing the allocation, protocols for distributing the allocation information and determination of time required to carry out distribution;
- consideration of pre-calculation and storage-for-fast-retrieval of the bandwidth allocation;
- the development of a hybrid approach which reroutes as many calls as possible by switching to backup VP's, and then executes the algorithms presented here to carry out VC level rerouting.

In sections 4.1.3 and 4.1.2 we present formulations of the rerouting problems. The purpose of these is to describe the fundamental problem structure. These formulations can not be used directly, since for the target environment distributed, real-time algorithms are required. In Section 4.2, when specific algorithms are presented, the telecommunications environment is taken into account.

4.1.2 Notation

Let $\mathcal{G} = (\mathcal{N}, \mathcal{L})$ be the undirected graph representing a telecommunication network; $\mathcal{N}$ is the set of nodes, $\mathcal{L}$ is the set of links. Since each link carries different calls in each of its two directions, we consider each link $[i, j] \in \mathcal{L}$ consisting of two directed arcs: (i, j) and (j, i). The set of all arcs is denoted by $\mathcal{A}$, $\mathcal{A} \subset \mathcal{N} \times \mathcal{N}$. Residual (free) capacity of the arc is defined as the difference between the total link capacity and the total bandwidth of the calls carried by the arc. In general, the arcs (i, j) and (j, i) have different residual capacities B_{ij} and B_{ji} since

different calls are routed over these arcs. We will also introduce the following notation for the network: $IN(i)$ = set of arcs directed into node i; $OUT(i)$ = set of arcs directed out of node i. Everywhere further the network is considered consisting of directed arcs.

As indicated previously, we let $\mathcal{K}$ denote the set of calls to be rerouted. For each call $k \in \mathcal{K}$, the following set of parameters is defined: origin $O(k) \in \mathcal{N}$, destination $D(k) \in \mathcal{N}$, bandwidth b_k and priority class $h(k) \in H$. Additionally, we associate a revenue $\hat{c}_{h(k)}b_k$ with call k. Thus $\hat{c}_\ell$ can be considered as a revenue obtained from routing each unit bandwidth of class ℓ (We note that the requirement that revenue is a linear function of bandwidth is *not* essential to the overall approach). Notice, that the revenue is received only after routing the "entire" call.

For the purposes of the exact problem formulation, we assume the set of all active calls $\mathcal{M}$ is known explicitly. This assumption will be replaced with a reasonable, practical approximation in section 4.2. Let

$$\text{for all } m \in \mathcal{M},\, e \in \mathcal{A}: \quad \delta_{em} = \begin{cases} 1 & \text{if call } m \text{ is carried by arc } e \\ 0 & \text{otherwise} \end{cases} \tag{4.1}$$

and let $\tilde{c}_{h(m)}b_m$ be the cost of preempting active call m with priority class $h(m)$ and bandwidth b_m. Similarly to routing revenue, $\tilde{c}_\ell$ can be considered as a cost of preempting each unit of class ℓ bandwidth.

A path through the network for the call k is defined as a sequence of arcs $(i_0, i_1), (i_1, i_2), \ldots, (i_{m-1}, i_m)$ where $i_0 = O(k)$ and $i_m = D(k)$. We will denote the path by p_k. The set of all feasible paths for call $k \in \mathcal{K}$ will be denoted as $\mathcal{P}(k)$. In order to "penalize" long call routes, we introduce heuristic coefficients s_e^k representing the penalty we pay by rerouting call k over arc e.

4.1.3 Problem Formulation

In this section, we present an integer programming (IP) formulation of the problem. This formulation is probably the most natural. It includes embedded "flow" problems for each call to be routed. The formulation uses the following 0/1 variables:

$$\text{for all } k \in \mathcal{K}: \quad y_k = \begin{cases} 1 & \text{if call } k \text{ is rerouted} \\ 0 & \text{otherwise} \end{cases}$$

$$\text{for all } k \in \mathcal{K} \text{ and } e \in \mathcal{A}: \quad x_e^k = \begin{cases} 1 & \text{if call } k \text{ is rerouted over the arc } e \\ 0 & \text{otherwise} \end{cases}$$

$$\text{for all } m \in \mathcal{M}: \quad z_m = \begin{cases} 1 & \text{if call } m \text{ is preempted} \\ 0 & \text{otherwise} \end{cases}$$

The exact IP formulation is given by:

(IPR):

$$\text{Maximize} \quad \sum_{k \in \mathcal{K}} \hat{c}_{h(k)} b_k y_k - \sum_{m \in \mathcal{M}} \tilde{c}_{h(m)} b_m z_m - \sum_{k \in \mathcal{K}} \sum_{e \in \mathcal{A}} s_e^k x_e^k \tag{4.2}$$

subject to

$$\sum_{e \in OUT(i)} x_e^k - \sum_{e \in IN(i)} x_e^k = \begin{cases} -y_k & \text{if } D(k) = i \\ y_k & \text{if } O(k) = i \\ 0 & \text{otherwise} \end{cases} \quad \text{for all } i \in \mathcal{N} \text{ and } k \in \mathcal{K} \tag{4.3}$$

$$\sum_{k \in \mathcal{K}} b_k x_e^k \leq B_e + \sum_{m \in \mathcal{M}} \delta_{em} b_m z_m \qquad \text{for all } e \in \mathcal{A}, \tag{4.4}$$

$$0 \leq y_k \leq 1 \quad \text{and integer} \qquad \text{for all } k \in \mathcal{K}, \tag{4.5}$$

$$0 \leq z_m \leq 1 \quad \text{and integer} \qquad \text{for all } m \in \mathcal{M}, \tag{4.6}$$

$$x_e^k \geq 0 \quad \text{and integer} \qquad \text{for all } e \in \mathcal{A} \text{ and } k \in \mathcal{K} \tag{4.7}$$

Objective function (4.2) maximizes the revenue received by routing the calls (the first term), minimizes the capacity violation (the second term) and a cost

which encourages shorter routes to be chosen (the last term). Coefficients s_e^k are chosen to be relatively small in comparison to cost/revenue coefficients. Constraints (4.3) are imposed upon each call k are the usual network flow conservation equations. Since variables x_e^k are integral and the flow value for each call is 0 or 1 (the y variable restriction), the call can be rerouted over at most one, non-split path. Hence $x_e^k \in \{0, 1\}$. Constraints (4.4) enforce the capacity restriction on each arc where the upper bound on the capacity is a "soft" constraint – it may be violated by preempting the calls.

We now present an alternate formulation, which uses a 0/1 variable for each feasible path. Thus, conceptually it potentially requires a very large number of variables/data. In general, this should not deter one from considering it since formulations of this type can be quite practical *if one generates variables and their associated columns dynamically.* Specifically, for problems of this type "column generation" approaches are used, which generate columns of the constraint matrix only as they are required. It is typical that the number of columns generated is of a very manageable size and is much less then the total number which appear in the problem statement. Although, initially we do not intend to use formal column generation approaches, our approximate algorithms will be similar in spirit to column generation and will use many of the same concepts.

The new variables for this formulation are path variables defined as follows:

$$f_{kp} = \begin{cases} 1 & \text{if call } k \text{ is rerouted over the path } p \\ 0 & \text{otherwise} \end{cases} \qquad \text{for all } k \in \mathcal{K} \text{ and } p \in \mathcal{P}(k)$$

Flow-Path Formulation with Aggregate Call Preemption Costs:

(FPAB):

$$\text{Maximize:} \quad \sum_{k \in \mathcal{K}_r} \hat{c}_{h(k)} b_k y_k - \sum_{m \in \mathcal{M}} \tilde{c}_{h(m)} b_m z_m - \sum_{k \in \mathcal{K}} \sum_{p \in \mathcal{P}(k)} s_p^k f_{kp} \tag{4.8}$$

subject to

$$\sum_{p \in \mathcal{P}(k)} f_{kp} = y_k \qquad \text{for all } k \in \mathcal{K}, \tag{4.9}$$

$$\sum_{k \in \mathcal{K}} b_k \sum_{p : e \in p \in \mathcal{P}(k)} f_{kp} \leq B_e + \sum_{m \in \mathcal{M}} \delta_{em} b_m z_m \qquad \text{for all } e \in \mathcal{A} \tag{4.10}$$

$$0 \leq y^k \leq 1 \qquad \text{for all } k \in \mathcal{K}, \tag{4.11}$$

$$f_{kp} \geq 0 \text{ and integer for all } k \in \mathcal{K} \text{ and } p \in \mathcal{P}(k) \tag{4.12}$$

$$0 \leq z_m \leq 1 \text{ and integer} \qquad \text{for all } m \in \mathcal{M}, \tag{4.13}$$

Constraints (4.9) state there is exactly one path for the call if it is rerouted and no paths if it is not. Constraints (4.10) are similar to those of (4.4).

The path formulation and associated column generation approaches provide a general framework for the development of a class of approximate and exact algorithms. Specifically, the process can be viewed as starting with a set of calls to be rerouted. A list of paths is built up dynamically over time. At any given time there exists:

- a list of calls
- a list of paths
- a "matrix" of path/call compatibilities and assignment costs (e.g. s_k^p)
- a (possibly partial) assignment of calls to paths

An any given iteration the algorithm could improve the assignment of calls to paths or alternatively, generate one or more new paths. Of course, these two options are clearly linked. If it did not seem possible to generate a good assignment

of calls to paths using the current set of paths, then this would signal to need to generate additional paths. Furthermore, paths could be generated specifically to try to accommodate certain unassigned or poorly assigned calls.

4.2 Algorithms

In the previous section, we presented exact IP formulations of the problem. Theoretically, we could have plugged those formulations into an IP solver and obtained an optimal solution. Such an approach would not be appropriate in a real-time distributed setting. We propose a two-phase approach in which the final decision on, and calculation of, a call's path remains with the call origination node.

In this section we outline the design of the algorithms and overall problem structure and derive some theoretical running time bounds for the algorithm. The next section gives an experimental evaluation of the quality of these algorithms.

Subsection 4.2.2 gives a formulation of the Phase I capacity allocation problem in which aggregated link capacity is allocated to the call originating nodes. An example solution to the capacity allocation problem is sketched in Figure 4.1 where two different shading colors show capacities allocated to call originating nodes A and G. The capacity allocation problem is a linear program which can be solved using standard commercial software. The solution provides a global capacity allocation. In calculating this allocation, the total bandwidth allocated to all call origination nodes on a given link might include both bandwidth currently available and bandwidth that must be freed by preempting existing calls.

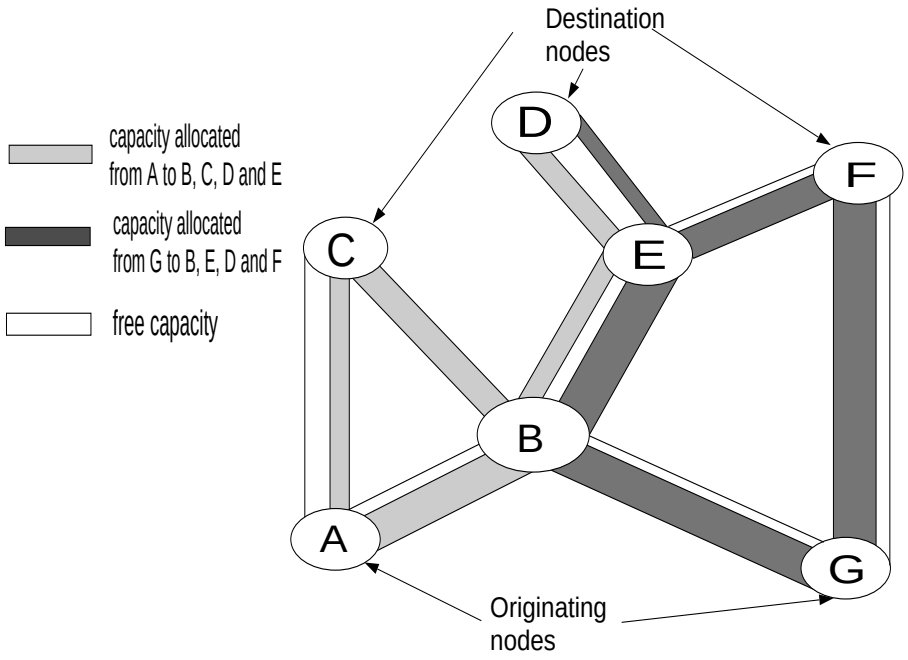

Figure 4.1: Phase I: Capacity Allocation by Solving a Linear Program

In the second phase, each call origination node sets up calls "assuming" all allocated bandwidth is available. When each individual call is actually set up, the nodes along the call's path will preempt existing calls if necessary. The Phase I problem must be solved once by a single central node. Possible candidate locations for solving this problem include one of the two nodes adjacent to the failed link and the network control/management center.

Subsection 4.2.3 presents an approximate algorithm for the bulk path selection problem which is solved, in parallel, by each call origination node. Figure 4.2 demonstrates the way by which an aggregated link capacity allocated to nodes A and G is assigned to individual calls. The bulk path selection problem is a variant of the bandwidth packing problem. The algorithm presented involves the iterative solution of shortest path problems and is motivated by first-fit-decreasing bin packing heuristics.

4.2.1 Network Environment

One of the basic notions underlying this chapter is to have network management functions and network control functions cooperate on this problem. Our paradigm is that each node has a copy of the network topology database which has such information as: the set of links and nodes in the network, current loading of the links (or a moving average of this) perhaps broken down by preemption priority or by QOS class. This topology database is updated via network control protocols and is used as input to the route computation algorithms.

In addition, each link tracks the bandwidth currently associated with the connections crossing it. When a link fails, the node on each side of the failing link may access this information and may use network management (or other)

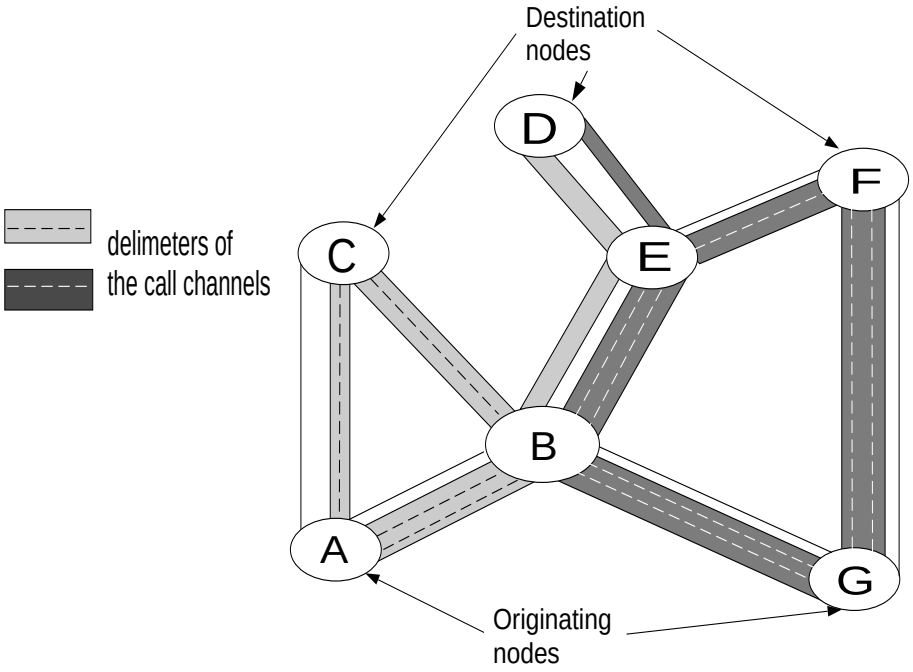

Figure 4.2: Phase II: Bulk Rerouting

protocols to send the list of affected calls to the node that is performing the Phase I computation (as mentioned previously, the node performing the Phase I calculation could be one of the nodes adjacent to the failed link). Note that some of this information may have to be kept for network accounting and charge-back anyway, but it may also have some real-time network control uses. It is also possible to aggregate this information somewhat as suggested later in this chapter. In any case, it is perfectly reasonable for a node to keep such information (IBM's NBBS does keep much of this information in the switches [1, 43]).

The time required to manipulate and possibly transmit the information for the Phase I computation should be reasonable. Specifically, if a link carrying 4096 connections in each direction fails, then the nodes adjacent to it each send 4096 $\times$ (calling party id, preemption priority class, and traffic parameters). Certainly this is not an outrageous amount of information and probably less than 100K bytes. The node performing the Phase I computation distributes (possibly via multicast directly, or possibly through the topology database update mechanism) the results of Phase I to the rest of the network for the Phase II computation.

4.2.2 Phase I: Capacity Allocation

Below, we give a linear programming formulation of the capacity allocation problem. We introduce the following variables:

for all $(i,j) \in \mathcal{N} \times \mathcal{N}$, $\ell \in H$: $\tilde{y}_{ij}^{\ell}$ is total amount of priority ℓ capacity assigned to calls whose origin, destination pair is (i,j);

for all $e \in \mathcal{A}$, $i \in \mathcal{N}$: $\tilde{x}_e^i$ is total amount of capacity assigned on arc e to calls whose origination node is i.

The $\tilde{x}_e^i$ variable values are the capacity allocations that are passed from Phase I to Phase II. As before s_e^i is the cost coefficient which discourages long paths.

The LP problem is obtained from problem (**IPR**) by aggregating x and y variables and relaxing the integrality constraints.

(CA):

$$\text{Maximize} \quad \sum_{\ell \in H} \hat{c}_\ell \sum_{(i,j) \in \mathcal{N} \times \mathcal{N}} \tilde{y}_{ij}^\ell - \sum_{m \in \mathcal{M}} \tilde{c}_{h(m)} b_m z_m - \sum_{i \in \mathcal{N}} \sum_{e \in \mathcal{A}} s_e^i \tilde{x}_e^i \tag{4.14}$$

subject to

$$\sum_{e \in OUT(j)} \tilde{x}_e^i - \sum_{e \in IN(j)} \tilde{x}_e^i = \begin{cases} -\sum_{\ell \in H} \tilde{y}_{ij}^\ell & \text{if } j \neq i \\ \sum_{m \in \mathcal{N}} \sum_{\ell \in H} \tilde{y}_{im}^\ell & \text{if } j = i \end{cases} \quad \text{for all } i, j \in \mathcal{N}, \tag{4.15}$$

$$\sum_{i \in \mathcal{N}} \tilde{x}_e^i \leq B_e + \sum_{m \in \mathcal{M}} \delta_{em} b_m z_m \qquad \text{for all } e \in \mathcal{A}, \tag{4.16}$$

$$\tilde{y}_{ij}^\ell \leq \sum_{k:O(k)=i, D(k)=j, h(k)=\ell} b_k \qquad \text{for all } i, j \in \mathcal{N},\ \ell \in H, \tag{4.17}$$

$$\tilde{x}_e^i, \tilde{y}_{ij}^\ell \geq 0 \qquad \text{for all } i, j \in \mathcal{N},\ e \in \mathcal{A},\ \ell \in H, \tag{4.18}$$

$$0 \leq z_m \leq 1 \qquad \text{for all } m \in \mathcal{M}. \tag{4.19}$$

It can be easily seen that the formulation **(CA)** immediately follows from **(IPR)** if we perform the substitutions:

$$\tilde{x}_e^i = \sum_{k:O(k)=i} b_k x_e^k \ , \tag{4.20}$$

$$\tilde{y}_{ij}^\ell = \sum_{k:O(k)=i, D(k)=j, h(k)=\ell} b_k y_k \tag{4.21}$$

and relax the integrality constraints.

Because of the discrete nature of calls (they are not allowed to be split), it is likely that the Phase II algorithm will not be able to "pack" all calls within the capacity allocated, if the amount of the capacity allocated is exactly equal to the

total bandwidth of the calls to be packed. To compensate for this effect, at least partially, we introduce an additional, "virtual" priority class ℓ'. Thus, we update the set of priority classes: $H' = H \bigcup \{\ell'\}$, and define $\hat{c}_{\ell'} \ll \min\{\hat{c}_\ell : \ell \in H\}$. Then the objective function (4.14) becomes:

$$\text{Maximize} \quad \sum_{\ell \in H'} \hat{c}_\ell \sum_{(i,j) \in \mathcal{N} \times \mathcal{N}} \tilde{y}_{ij}^\ell - \sum_{m \in \mathcal{M}} \tilde{c}_{h(m)} b_m z_m - \sum_{i \in \mathcal{N}} \sum_{e \in \mathcal{A}} s_e^i \tilde{x}_e^i \tag{4.22}$$

A new heuristic upper bound for the y-variables is added:

$$\tilde{y}_{ij}^{\ell'} \leq \varepsilon \sum_{k: O(k)=i, D(k)=j} b_k \qquad \text{for all } i, j \in \mathcal{N}, \tag{4.23}$$

($\varepsilon = .1$ is a heuristically set parameter),

and the flow conservation constraints (4.15) are changed to:

$$\sum_{e \in OUT(j)} \tilde{x}_e^i - \sum_{e \in IN(j)} \tilde{x}_e^i = \begin{cases} -\sum_{\ell \in H'} \tilde{y}_{ij}^\ell & \text{if } j \neq i \\ \sum_{m \in \mathcal{N}} \sum_{\ell \in H'} \tilde{y}_{im}^\ell & \text{if } j = i \end{cases} \quad \text{for all } i, j \in \mathcal{N}, \tag{4.24}$$

(i.e. H is replaced by H') and we obtain a new formulation $(\mathbf{CA}')$ which is a slight modification of $(\mathbf{CA})$.

Till now, it was assumed the set of currently active calls to be known explicitly. It might not be the case in the real-life situations since

1. this set of calls is enormously large;

2. the algorithm will be executed by each individual source node and these nodes generally will not have global call information available.

Instead, we will assume that for each link, in addition to the residual capacity, the amount of bandwidth allocated to each priority class of calls is known. Specifically, we have for each link, $e \in \mathcal{A}$, and priority class $\ell \in H$,

B_e^ℓ = bandwidth amount along link e occupied by calls of priority class ℓ.

We then define variables for each $e \in \mathcal{A}$ and $\ell \in H$,

$$\tilde{z}_e^\ell \quad = \quad \text{the amount of bandwidth preempted from priority class } \ell \text{ on link } e,$$

Note that we have made a rather significant simplifying assumption that the cost of preempted traffic is *a linear function of the total bandwidth freed on each link*. This assumption represents an approximation for two reasons:

1. bandwidth is freed in discrete increments corresponding to the bandwidth associated with the calls that must be preempted.

2. preempting of a single call will, in general, free bandwidth on several links.

While there are certainly circumstances under which this assumption clearly leads to significant cost distortions, we feel that this simplification (or another similar one) is necessary in order to derive a practical approach to routing.

Using definitions for z variables and δ_{em}, one can express B_e^ℓ and $\tilde{z}_e^\ell$ as follows:

$$\begin{aligned} B_e^\ell &= \sum_{m:\ m \in \mathcal{M},\, h(m)=\ell} \delta_{em} b_m \\ \tilde{z}_e^\ell &= \sum_{m:\ m \in \mathcal{M},\, h(m)=\ell} \delta_{em} b_m z_m \end{aligned}$$

Next, we use $\tilde{z}$ variables to formulate the LP which can be used in practical computations.

(CAP):

$$\text{Maximize} \quad \sum_{\ell \in H} \hat{c}_\ell \sum_{(i,j) \in \mathcal{N} \times \mathcal{N}} \tilde{y}_{ij}^\ell - \rho \sum_{e \in \mathcal{A}} \sum_{\ell \in H} \tilde{c}_\ell \tilde{z}_e^\ell - \sum_{i \in \mathcal{N}} \sum_{e \in \mathcal{A}} s_e^i \tilde{x}_e^i \tag{4.25}$$

subject to

$$\sum_{e \in OUT(j)} \tilde{x}_e^i - \sum_{e \in IN(j)} \tilde{x}_e^i = \begin{cases} -\sum_{\ell \in H} \tilde{y}_{ij}^\ell & \text{if } j \neq i \\ \sum_{m \in \mathcal{N}} \sum_{\ell \in H} \tilde{y}_{im}^\ell & \text{if } j = i \end{cases} \quad \text{for all } i, j \in \mathcal{N}, \tag{4.26}$$

$$\sum_{i \in \mathcal{N}} \tilde{x}_e^i \leq B_e + \sum_{\ell \in H} \tilde{z}_e^\ell \quad \text{for all } e \in \mathcal{A}, \tag{4.27}$$

$$\tilde{y}_{ij}^\ell \leq \sum_{k:O(k)=i, D(k)=j, h(k)=\ell} b_k \quad \text{for all } i, j \in \mathcal{N},\ \ell \in H, \tag{4.28}$$

$$\tilde{z}_e^\ell \leq B_e^\ell \quad \text{for all } e \in \mathcal{A},\ \ell \in H, \tag{4.29}$$

$$\tilde{x}_e^i, \tilde{y}_{ij}^\ell, \tilde{z}_e^\ell \geq 0 \quad \text{for all } i, j \in \mathcal{N},\ e \in \mathcal{A},\ \ell \in H. \tag{4.30}$$

Notice that the cost of call preemption in this formulation may be over-counted since it is summed up over several links the call's bandwidth was freed. To compensate this effect, a corrective coefficient ρ was applied. Theoretically,

$$\frac{1}{\text{number of hops in the max hop route of the existing call}} < \rho < 1.$$

After some experimentation, we choose $\rho = 0.5$.

Similar to defining problem **(CA′)**, substitution of priority set H with H' can be performed and then we obtain formulation **(CAP′)**

Problems **(CA)** (or **(CA′)**) and **(CAP)** (or **(CAP′)**) were solved with a linear programming solver for the variety of instances. The solutions obtained indicated that the approximation was sufficiently accurate.

Observe that, since problems formulated in this section are essentially multi-commodity flow problems, specialized algorithms could have been used. Exploration of such approaches lies beyond the scope of our current study.

As a result of solving the problem **(CAP′)**, we assign arc capacities to each

origination node. Those capacities are input into the algorithm described in the next section.

4.2.3 Phase II: Bulk Routing

In this section, we consider an algorithm which works independently for all origination nodes. Each node must receive a capacity allocation from the Phase I algorithm. That capacity allocation for call origination node i_0, for each link e, is denoted by $B_e(i_0)$. This link capacity is set equal to the value of variable $\tilde{x}_e^i$ computed in Phase I. The problem that must be solved in Phase II is a variant of FPAB (given in Section 4.1.3) in which the $\tilde{z}_e^\ell$ (call preemption) variables are not present. That is, each call origination must carry out path selection using only the capacity allocated in Phase I. The paths calculated by each call origination node will automatically lead to appropriate call preemption since this was planned in determining the Phase I allocations.

Two considerations are critical in designing the routing algorithm: first, calls should be routed over shortest paths and, second, care should be taken in "packing" the calls into the constructed routes as tightly as possible. In addition, the running time should be small enough to allow for real-time use. To address these issues, the algorithm designed makes use of shortest path algorithms (see e.g. [2]) and handles the calls in the spirit of the so-called *first fit decreasing* (FFD) heuristic which is known to be quite effective for the bin packing problem (see e.g. [32]).

We arrange the calls within the same priority class in descending order of their bandwidth. As the algorithm proceeds, we keep a pointer to the first unrouted call, $\hat{k}$. Focusing on $\hat{k}$, we execute a special shortest path algorithm

which constructs a shortest path tree (SPT) rooted at node i_0 using only those arcs which have enough capacity to handle $\hat{k}$. Afterwards, we try to route as many calls as possible over that SPT taking them in the order they appear in the list (descending by bandwidth). The process iterates using the new first unrouted call. When we can not route anymore calls within the same priority class, we proceed to the next priority class. Clearly, this approach is similar to the FFD algorithm for the bin packing problem. Indeed, the computational results presented in the next section indicate that it is quite effective in practice.

Input parameters of the algorithm are network $(\mathcal{N}, \mathcal{A})$, the origination node i_0, allocated capacities on the arcs $B_e(i_0),\ e \in \mathcal{A}$ and set of calls $\mathcal{K}_{i_0} = \{k \in \mathcal{K} | O(k) = i_0\}$, output is the set of routes. We suppose the priority levels are ordered in the descending order: $l_1 \succ l_2 \succ \ldots \succ l_{|H|}$ and denote $\mathcal{A}(w) = \{e \mid e \in \mathcal{A}, B_e \geq w\}$. To better access the largest bandwidth calls, all calls are grouped by their destinations and stored at corresponding binary trees. Another set of internal data consists of capacity availability labels w_j stored at each node j which show the maximum amount of capacity that can be used to route a call to the node or, equivalently, it is the capacity of the "narrowest" arc on the path from i_0 to j in the SPT. In general the arc lengths defining the shortest path are call delays. However, in this case we consider the min-hop path and hence we have unit arc lengths.

1.**for** priority level $\ell = \ell_1,\ \ell_2, \ldots, \ell_{|H|}$ **do**

2. Initialize binary trees $\mathcal{T}_j,\ \forall j \in \mathcal{N}$:

3. $\mathcal{T}_j$ contains items $\{k \in \mathcal{K}_{i_0} \mid h(k) = \ell,\ D(k) = j\}$ with keys b_k;

4. **while** there is a non-empty tree $\mathcal{T}_j$ **do**

5. find call $\hat{k} : b_{\hat{k}} = \max_{j \in \mathcal{N}} \max_{k \in \mathcal{T}_j} b_k$;

6. Construct an SPT rooted at i_0 for the network $(\mathcal{N}, \mathcal{A}(b_{\hat{k}}))$;

7. **if** there is a path from i_0 to $D(\hat{k})$, route the call $\hat{k}$ and update the labels w_j;

8. Remove the call $\hat{k}$ from its tree: $\mathcal{T}_{\hat{k}} \leftarrow \mathcal{T}_{D(\hat{k})} - \{\hat{k}\}$;

9. **do**

10. find call $\tilde{k}$: $b_{\tilde{k}} = \max_{j \in \mathcal{N}} \max_{k \in \mathcal{T}_j} \{b_k \mid b_k \leq w_j\}$;

11. if $\tilde{k}$ was found,

12. route the call $\tilde{k}$ in the current SPT;

13. remove the call $\tilde{k}$ from its tree: $\mathcal{T}_{\tilde{k}} \leftarrow \mathcal{T}_{D(\tilde{k})} - \{\tilde{k}\}$;

14. update the labels $w_j \; \forall j \in \mathcal{N}$;

15. **while** there is a call $\tilde{k}$;

The next issue we want to discuss here is the running time of the algorithm. All operations we perform with binary trees require time logarithmic on the size of the tree. These operations are: removing the items (lines 8 and 13) and finding the max key item (lines 5 and 10). Clearly, each call is considered only once. Indeed, if there is enough bandwidth in the existing SPT for the call (line 11), it is routed and removed (lines 12–13); if there are no calls which can be routed over the existing SPT (line 15), a shortest path algorithm is run to route the largest bandwidth unrouted call (line 6). If there is a path for that call, it is routed and removed, otherwise, it is removed anyway, since there is no chance for the call to be routed later. Capacity availability labels w_j are updated in time $O(|\mathcal{N}|)$ (lines 7 and 14). The time required to extract the largest bandwidth call is $O(|\mathcal{N}| \log |\mathcal{K}|)$. If we denote the time required to construct an SPT by T_{SPT}, the worst case running time of the algorithm is $O(|\mathcal{K}|(|\mathcal{N}| \log |\mathcal{K}| + T_{SPT}))$. The average should be significantly better since, in general, several calls will be routed over each SPT constructed. The running time for the shortest path

Call Type	Bandwidth Distribution	Priority Distribution	Percentage		
			Mix 1	Mix 2	Mix 3
voice	all are .064 Mb/s	10% are 1, 90% are 2	54%	47%	41%
data	50% are .0096 Mb/s, 50% are uniformly from .01 to 1. Mb/s	uniformly, 1–4	43%	47%	53%
video	uniformly, 1 to 3 Mb/s	uniformly, 1–3	3%	6 %	6%

Table 4.1: Call Types

algorithm can vary depending on the approach used and the exact nature of the cost function. In the specific context we considered, since the objective was a pure "min-hop", a breadth first search algorithm could be used which runs in time $O(|\mathcal{A}|)$.

4.3 Experimental Results

The purposes of our experiments were:

1. estimate the quality of our solution

2. estimate the time required to perform the computations

Our sample data were obtained by following a general simulation routine described in chapter 3. In our experiments we generated calls of three different types: voice, data and video. Table 4.1 gives bandwidth and priority characteristics for the calls. Each call of priority class 1, 2, 3 and 4 was assumed to have revenue 1.0, 0.1, 0.01 and 0.001 respectively. Table 4.1 also presents three different mixes of calls which were considered.

The network used for our simulation had a topology similar to nation-wide ARPA net with 59 nodes, 71 undirected links (respectively, 142 directed arcs), as it appears in [10], p. 135. Each link had a full duplex capacity of 154 Mb/s. The overall parameter setting was chosen so that after a link failure the total number of calls to be rerouted would be between 300 and 1,600.

We evaluate our approach in two general areas. First, there are the revenue/cost components reflected in the objective functions previously described, i.e. the total revenue of all calls rerouted minus lost revenue of calls preempted. The second criterion to be considered involves call refusal. A call refusal occurs when the initial setup attempt fails. A call refusal causes a substantial cost in overall network overhead as well as delay associated with reestablishing the call. To accurately evaluate these concerns, a detailed simulation is required. In lieu of such an analysis, we carried out certain experiments which indicate how alternate approaches would perform under "extreme" conditions. These were sufficient to derive definitive conclusions.

The experimental evaluation consisted of two parts. In the first part the quality of the call routing algorithm was estimated without considering call preemption. In the second part the evaluation of call preempting capabilities was performed. These parts of the experiment are described below in sections 4.3.1 and 4.3.2 respectively.

4.3.1 Evaluation of the Call Rerouting Algorithm

In lieu of simulation, we use two methods to establish a baseline solution for which to compare the quality of our two-phase approach to call rerouting without preemption. In both approaches, each call is rerouted by an independent

execution of a path selection (shortest path) algorithm. For the first approach, which can be considered a "worst case" scenario, we assume that each call origination node reroutes its calls but receives no updated information about link status during the rerouting process. Thus, all nodes assume that they have full access to all available bandwidth capacity. This is called "fast" rerouting. This approach will necessarily lead to high levels of call refusal in that competing call origination nodes will try to setup calls over the same sets of links. If a call is refused, we do not attempt to reroute it – another very pessimistic assumption.

Under the second approach, we assume that a central node with perfect information reroutes all calls. That is, whenever a route is calculated, the precise status of all links is known and as a result, there is no call refusal. The central node uses a first fit decreasing heuristic approach, by ordering calls in decreasing order of bandwidth. We call this method "slow" rerouting since time would be required to propagate the information about the state of the network and since the actual implementation of this approach would require the distribution of all routes from the central node. The two approaches constitute two extremes in that

- the fast approach requires the minimal amount of information exchange, but generally would achieve low quality routes;
- the slow approach should achieve the highest quality results (subject to the limitations of the first fit decreasing approach to packing) but is impractical due to the extreme information exchange requirements.

Our approach can be viewed as a compromise which executes an approximate global rerouting with modest information exchange requirements while keeping

final path selection in the hands of the call origination nodes.

Two sets of results are presented. The first is aimed at evaluating the effectiveness of the overall two phase approach and the second specifically addresses the Phase II packing algorithm. Table 4.2 gives the characteristics of the calls to be rerouted. For each utilization and call mix combination the number of calls to be rerouted and their total revenue are given. The results presented in Table 4.3 address the overall approach. The first four rows give the results of the fast and slow rerouting algorithms respectively. The fifth and sixth rows give the results of the two phase approach applied to the same problems. Note that it shows significant improvement over the fast approach and comes close to achieving the quality levels of the slow approach. Note that the results do degrade somewhat for the highest utilization level. This can be expected since for these levels packing becomes more difficult. In the next two rows results are given when preemption is allowed. This gives an indication of the rather significant potential improvements provided by preemption. The summary of the results is plotted in Figure 4.3. Each of the three parts in this figure corresponds to a particular distribution of the call types. Pie diagrams on the right show the fractions of the number of calls in each class. Plots on the left show the fraction of the rerouted calls.

Evaluation of the call preemption algorithm will be presented in section 4.3.2. The final row in Table 4.3 gives the running times in seconds. We temper the discussion given below by noting that the experiments were run on a Sun Sparc 10 which may or may not be similar to the hardware environment anticipated in an actual telecommunications network. The running time of the first phase is always lies between 20 seconds and 1 minute. This would certainly not be fast

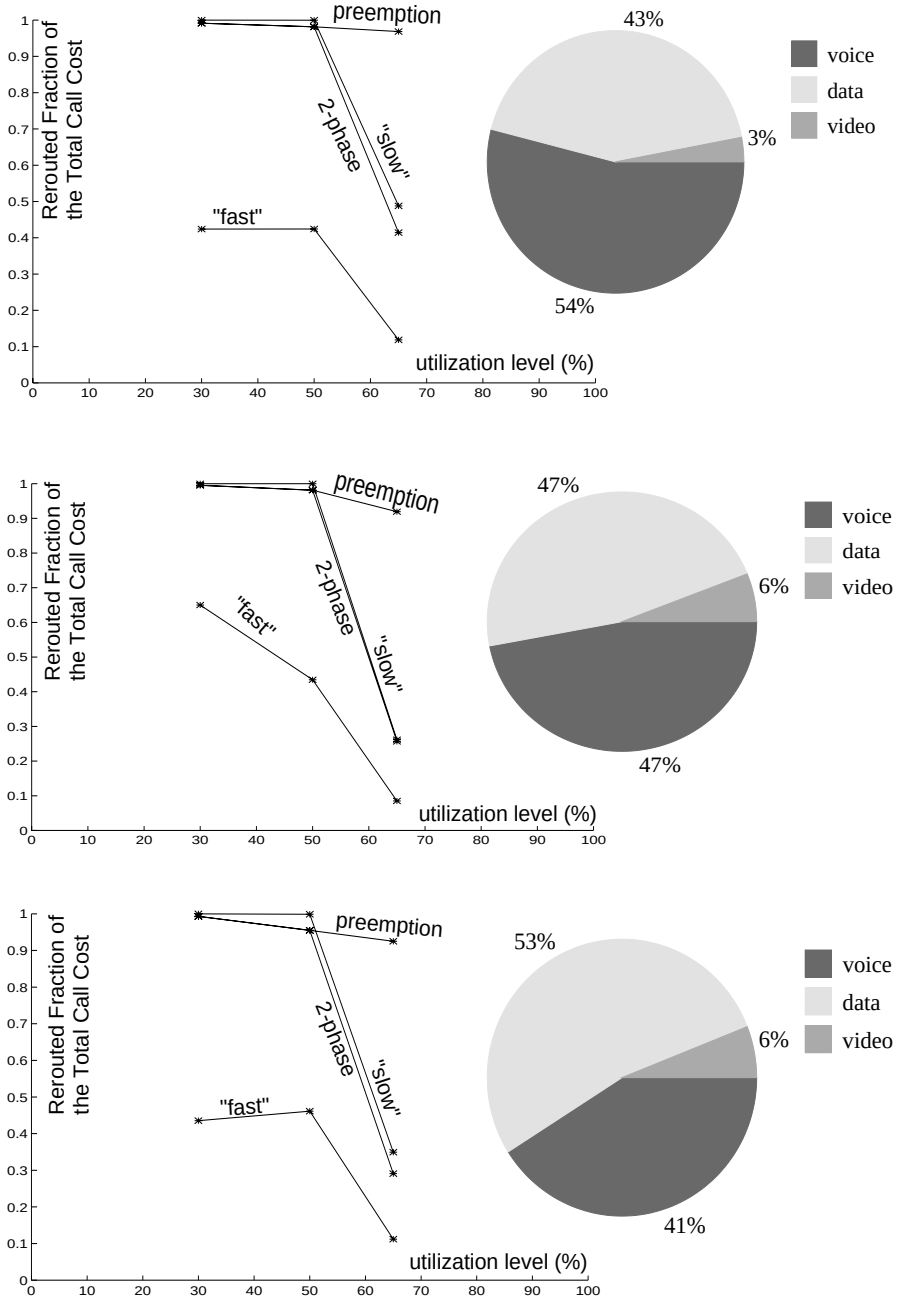

Figure 4.3: Summary of the Experimental Results

	Mix 1			Mix 2			Mix 3		
Utilization	30%	50%	65%	30%	50%	65%	30%	50%	65%
Total call revenue	32.46	79.50	115.28	49.79	97.65	124.99	38.29	114.40	139.38
Number of calls	357	897	1566	305	646	1414	287	670	1375

Table 4.2: Call Characteristics

Total Rev. and Number of the Rerouted Calls		Mix 1			Mix 2			Mix 3		
		30%	50%	65%	30%	50%	65%	30%	50%	65%
"Fast"	revenue	13.76	33.73	13.67	32.37	42.44	10.66	16.67	52.78	15.61
Scenario	number	237	503	227	229	360	162	191	359	227
"Slow"	revenue	32.46	79.48	56.25	49.79	97.64	32.72	38.29	114.26	48.72
Scenario	number	357	869	148	305	630	171	287	636	179
2-Phase w/out	revenue	32.18	78.03	47.76	49.56	95.83	32.07	38.03	109.25	40.54
preemption	number	355	882	306	304	620	254	284	648	280
2-Phase with	revenue	32.18	79.16	110.05	49.56	95.83	119.53	38.28	110.80	135.02
preemption	number	355	882	1287	304	620	1118	286	653	1076
Running	Phase I	26.8	28.5	30.6	25.1	67.1	34.7	24.5	55.2	39.7
Time	Phase II	0.25	0.60	0.96	0.25	0.53	0.88	0.25	0.48	0.88

Table 4.3: Evaluation of the Two-Phase Approach

enough for real time operation but is not unreasonable for the scenario where the network management system had provided advance warning of a failure. In order to achieve real-time response, alternate approaches, including heuristics, for solving the LP could be considered. The running times for the second phase algorithm were always less than 1 second. The times reported consisted of the time used to execute **sequentially** the Phase II algorithm at each call origination node. Of course, in an actual implementation these calculations would be carried out in parallel leading to much smaller times. We feel these times are quite encouraging in that they would seem to allow for real-time operation.

Table 4.4 presents results concerning the quality of the Phase II algorithm. Three sets of experiments were carried out. In the first two, the fast and slow rerouting algorithms were each run. The capacity taken up on each link for calls associated with each call origination node were then allocated to that node. The Phase II algorithm was then given the job of routing calls using only the capacity allocated. This tested the ability of the phase II algorithm to choose a good set of calls and also to pack them into the bandwidth restricted links. The results for this experiment are given in the first 8 rows of Table 4.4. Note that the phase II algorithm generally did significantly better than the fast algorithm and generally came close to achieving the results of the phase II approach. In the 9th and 10th rows, the LP value is compared with the value achieved in Phase II (the Phase I value is an upper bound on the Phase II value). Note that for the medium and low utilization cases, the Phase II algorithm came very close to achieving the LP value. For the high utilization it was not as close. This, again, can be explained by the fact that for these levels packing becomes more difficult. Of course, further investigations would be required to determine how

Total Revenue and Number of the Rerouted Calls		Mix 1			Mix 2			Mix 3		
		30%	50%	65%	30%	50%	65%	30%	50%	65%
"Fast" Scenario	revenue	13.76	33.73	13.67	32.37	42.44	10.66	16.67	52.78	15.61
	number	237	503	227	229	360	162	191	359	227
Phase II Algorithm	revenue	14.90	37.67	16.54	32.19	46.35	10.68	18.14	60.64	20.53
	number	228	496	234	220	353	167	172	388	226
"Slow" Scenario	revenue	32.46	79.48	56.25	49.79	97.64	32.72	38.29	114.26	48.72
	number	357	869	148	305	630	171	287	636	179
Phase II Algorithm	revenue	32.20	76.41	56.21	48.95	96.33	32.07	38.03	109.25	48.42
	number	312	802	161	271	582	176	252	597	183
Phase I revenue (LP value)		32.46	79.50	56.29	49.79	97.65	32.77	38.29	114.40	50.27
Phase II revenue		32.18	78.03	47.76	49.56	95.83	26.21	37.35	106.80	40.54

Table 4.4: Effectiveness of the Phase II Packing Algorithm

much of the gap is due to the quality of the relaxation and how much is due to the quality of the heuristic.

4.3.2 Evaluation of the Call Preemption

In this section we present an evaluation of call preemption algorithm. Suppose after the work of the two-phase algorithm was completed a set of rerouted calls $\mathcal{K}_r$ was obtained together with their routes. Clearly, $\mathcal{K}_r \subset \mathcal{K}$. The set of the new routes can be described by introducing notation similar to (4.1):

$$\text{for all } k \in \mathcal{K}_r,\, e \in \mathcal{A}\text{:} \qquad \sigma_{ek} = \begin{cases} 1 & \text{if call } k \text{ is carried by arc } e \\ 0 & \text{otherwise} \end{cases} \tag{4.31}$$

Now, consider the following IP formulation:

$$\text{Maximize} \quad \sum_{k \in \mathcal{K}_r} \hat{c}_{h(k)} b_k y_k - \sum_{m \in \mathcal{M}} \tilde{c}_{h(m)} b_m z_m \tag{4.32}$$

subject to

$$\sum_{k \in \mathcal{K}_r} \sigma_{ek} b_k y_k - \sum_{m \in \mathcal{M}} \delta_{em} b_m z_m \leq B_e \quad \text{for all } e \in \mathcal{A}, \tag{4.33}$$

$$0 \leq y_k \leq 1 \quad \text{and integer} \quad \text{for all } k \in \mathcal{K}_r, \tag{4.34}$$

$$0 \leq z_m \leq 1 \quad \text{and integer} \quad \text{for all } m \in \mathcal{M}, \tag{4.35}$$

It can be easily seen that solution to (4.32–4.35) is a feasible suboptimal solution to **(IPR)** corresponding to the call rerouting and preemption given by the two-phase algorithm. The output of this integer program (4.32–4.35) consists of decisions regarding call refusal (the y_k variables) and call preemption (the z_m variables). Thus, in a certain sense, it gives the optimal solution to to the call setup process which is carried in a distributed fashion by the network nodes. Then the quality of the two-phase algorithm can be estimated by how close solution of (4.32–4.35) is to that of **(IPR)**. As it was noted above, the latter problem can not be solved explicitly but it is possible to estimate the value of its solution from above by solving its LP relaxation which is essentially **(CA)**. The results of the comparison for the previously described set of calls are presented in the table 4.5. We consider these results to be quite encouraging.

4.4 Conclusions

In this chapter we have presented models and algorithms which address certain rerouting problems which arise in multi-class traffic environments. The approach has potential applicability in several contexts. Our computational experiments

Calls Rerouted		Mix 1			Mix 2			Mix 3		
		30%	50%	65%	30%	50%	65%	30%	50%	65%
Upper Bound		32.46	79.50	113.24	49.79	97.65	121.64	38.29	114.40	136.27
Lower Bound	value	31.58	74.94	109.34	47.49	93.89	112.59	37.54	106.40	132.93
	number	318	835	1245	276	588	1084	254	604	1033
Gap		2.8%	6.1%	3.6%	4.8%	4.0%	8.0%	2.0%	7.5%	2.5%

Table 4.5: Preemption Evaluation

have demonstrated that these methods can greatly improve the response of a telecommunications system to link failures over more direct approaches. In order to carry out the implementation our methods, specific protocols must be designed to transfer the required information. Furthermore, refinements must be made to insure appropriate response in a real-time setting.

Chapter 5

Virtual Path Layout in an ATM Environment

Introduction

The Virtual Path (VP) is an important concept in ATM multiservice networks. In our research framework we consider VPs as logical macrolinks in telecommunication networks. That is, in general, a VP may traverse several physical links in the network to provide a faster connection between a pair of remote network nodes. This efficiency results from a greatly reduced processing time in the intermediate switching nodes of the VP. However, there is a tradeoff involved since capacity is allocated for each VP individually. In various ATM network models, the allocated capacity is treated differently but most often it is considered not to be available for use by the other VPs on the link. This in turn reduces the gain of statistical call multiplexing.

In the literature, there exist several different approaches to modeling and solving the VP layout problem. In [5, 18], the routing problem in VP-based ATM networks is considered. Dynamic routing policies suggested by the authors

attempt to minimize the VC blocking rate. Simulations were used to evaluate the proposed routing policies.

References [9, 30] consider the problem of minimizing the total VC blocking rate where VC demand between each node pair is assumed randomly arriving with a Poisson distribution. There are two special cases of the problem solved by the authors: (1) each VC uses exactly one VP and (2) each VP involves exactly one physical link. The problem was formulated as a nonlinear non-differentiable combinatorial optimization problem. Suggested approaches include Lagrangean relaxation, penalty functions and subgradient optimization techniques.

Reference [3] explored various issues of ATM network restoration, particularly rerouting strategies emphasizing restoration speed and capacity allocation and redundant capacity allocation methods that take into account that a cut of a single physical link may result in multiple service failures. However, analytical models for the call rerouting or spare capacity allocation were not proposed.

References [15, 14] deal with optimizing dynamic virtual path bandwidth allocation and restoration schemes. It is assumed that VP-level protocols provide the optimal logical assignment of VP bandwidth for an upcoming control period. The objective was to minimize the total rejected bandwidth demand subject to QOS constraints (cell loss and delay) and 100% restorability. The latter requirement implied that there should be a certain amount of spare capacity kept to respond to link/node failures. The authors did not consider explicit backup routing algorithms.

In reference [33], a proposed approach was described to minimize the amount of expected traffic flow loss when a link failure occurs. It was assumed that the traffic is restored using a set of routes bypassing the failed link. In this

approach the bypassing routes begin and terminate at the end nodes of the failed link. Statistical multiplexing effects which would affect the required amount of capacity when the VPs are split were not considered.

The VP-layout methods proposed in our paper allow fast restoration for ATM networks based on the concept of backup VPs. This approach was suggested in series of papers [21, 20, 22]. We follow the proposed framework and build an optimization model for VP layout satisfying certain reliability constraints. Essentially, we take advantage of the following ATM feature. It is known that the flexibility of the ATM architecture makes it possible to establish a zero capacity backup VP and then, in the event of failure, the full capacity can be captured sufficiently fast to provide a real-time service restoration. This approach has the advantage that the time-consuming process of setting routing tables at the switches and defining VP identifiers (VPI) can be carried out in an off-line mode and, when a failure occurs, an affected primary VP can be switched to a backup without noticeable interruption of real-time services. The restoration algorithm is very simple [22]. Basically, it consists of failure detection followed by a restoration message sent along the VP to reserve a proper amount of capacity (this algorithm is outlined in more detail in section 2.2).

We consider four variants of the fault-tolerant VP layout based on combinations of the following two alternatives:

1. (a) Exactly one link-disjoint backup VP is established for any primary VP;

   or

   (b) a set of backup VPs is established for a primary VP so that each backup VP provides a backup in the event of the failure of a corre-

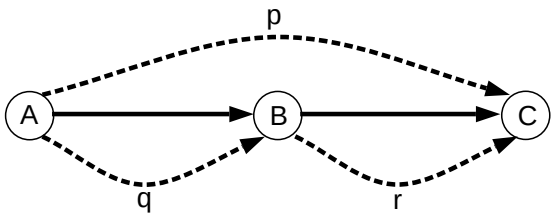

Figure 5.1: An Example of Two Rerouting Strategies

sponding link of the primary VP.

2. (a) The amount of bandwidth captured on the backup VP during the restoration is exactly the same as that of the primary VP;

   or

   (b) the backup VP is allocated less than the total primary VP bandwidth.

   (In the latter case a set of VCs must be dropped when the failure occurs.)

The approach which is the simplest, at least conceptually, is to establish one backup VP for each primary VP (1(a)). This backup VP would be used if any link in the primary VP fails. Thus, it should be link disjoint from the primary VP. Alternatively, the backup route may be set up to detour a certain link in a primary VP (1(b)). Then there are several backups for each primary VP, generally as many as there are links traversed by the primary VP. Backups of this kind serve as bypasses around faults. An example of these two strategies is shown in Figure 5.1. Here, a primary VP is set over the route ABC. Under the first strategy, a link-disjoint backup VP p is used if any of the links AB or BC fails. Under the second strategy, a backup VP consisting of path q and link BC

is used when link AB fails and a backup VP consisting of links AB and path r is used when link BC fails.

Naturally, one should expect less redundant capacity to be required in the network when implementing the second strategy. However, it has a certain disadvantage. Each link in the ATM network may carry a number of VPs not exceeding 4096 ($= 2^{12}$) which is limited by the 12 bits reserved for the VPI in the cell header. In the case of large networks and a large number of long VPs, there might not be enough VPI values to set up all link-dependent backup VPs. Also in this case the control overhead could be significantly higher.

In this chapter we consider a mathematical model for case 1a/2a. Our basic approach was motivated by the solution to a similar problem for packet networks proposed by Gavish et al [12]. However, their model contains a group of non-linear constraints which causes certain difficulties when a linear relaxation technique is applied. We show how these constraints can be linearized.

Considering the second pair of alternatives, a backup VP need not necessarily have the same capacity as the corresponding primary VP. Lesser capacity could be allocated in the case of limited network resources. When such a backup VP is used, some lower-priority calls would have to be taken down or rerouted separately. In order to accomplish this, the network protocols should have the ability to take down or reroute individual calls or VCs.

As was stated in chapter 2, we also suggest a more general model for fault-tolerant, link-disjoint VP layout where a VP can serve as both primary and backup VP. In this case we allow traffic flow to both VPs and, when failure occurs, traffic is switched from one VP to the other. The advantage of this model is additional routing flexibility and higher throughput as will be shown by the

example of section 5.2.5. In chapter 6, we develop a two-step heuristic solution approach for this model. On the first step, a VP pair is routed between every node pair (using link-disjoint routes) based on the solution of a multicommodity flow problem; on the second step, a sort of dimensioning problem is solved assigning appropriate capacities to VPs and reserving backup capacities on the network physical links.

Another performance measure which we consider in our research is adaptability of a particular VP layout to changes in traffic demand. Quantitatively, we measure adaptability in terms of call blocking probability. Our conjecture is that longer, lower capacity VPs have lower adaptability than shorter, higher capacity VPs. In order to validate this conjecture, we simulate the random generation of calls which are routed through the network using previously computed VPs. Our results show an increase in call blocking probability when VPs become longer but more narrow.

5.1 Statistical Multiplexing

We start our study of the VP layout by building a very simple analytical model for statistical multiplexing of VBR calls which captures the main properties of statistical multiplexing effect and gives a linear or piece-wise linear approximation suitable for our purposes.

We consider the two most common VBR traffic types in telecommunication networks: video and data. From the literature [35, 45], it is known that the distribution of video traffic transmission rates can be approximated by a normal distribution. Data traffic is usually considered as generated by on/off sources

with alternating periods of silence and transmission at peak rate.

We assume that buffer sizes in the network are minimal. This is the case for most high speed networks aimed at providing real-time service. In fact, queues are used only to avoid cell loss when arrivals from several multiplexed sources occur simultaneously. For example, even when streams from CBR sources are multiplexed together at a node and the outgoing link transmission rate is greater than the total cell arrival rate, there is a chance that instantaneous incoming rate may exceed the transmission rate when current cells on all links arrive within a short time interval. However, even in this case, there is no real cell queuing delay since the cells in the queue are transmitted before the arrival of the next group of cells.

Notice that this model presents an essential simplification when compared with other models (c.f. [39]) where buffer size is an essential parameter. Taking buffer size out of consideration makes our model insensitive to the distribution of the active period lengths. For example, adopting this model we can not distinguish between Poisson and self-similar types of traffic. Though it is an essential simplification of the model, it allows us to restrict our consideration to the probability distribution of the transmission rate.

We define our capacity allocation rule as providing the lowest capacity sufficient to yield a predefined CLR, ε, when a set of VBR connections $\mathcal{K}$ is multiplexed together. Let A_k be a random variable describing transmission rate of call $k \in \mathcal{K}$ and $A = \sum_{k \in \mathcal{K}} A_k$ be the total transmission rate. Then capacity allocation rule providing CLR, ε, can be formally defined as:

$$C(\mathcal{K}) = \inf\{C : \ Pr(A > C) < \varepsilon\} \tag{5.1}$$

In other words, we want the probability of exceeding the allocated capacity to

be less than predefined CLR. Based on this definition, we will derive piece-wise linear approximations for typical transmission rate distributions encountered in telecommunication networks.

5.1.1 Video Traffic Capacity Allocation

In this section we derive a piece-wise linear approximation for the capacity allocation rule in the case of video traffic which is assumed to have a normally distributed transmission rate [45].

Suppose n video connections are multiplexed together. Then their total transmission rate A is a normally distributed r.v. with mean $\mu = \sum_{i=1}^{n} \mu_i$ and variance $\sigma^2 = \sum_{i=1}^{n} \sigma_i^2$ where μ_i and σ_i^2 are mean and variance, respectively, of the i-th connection. Taking into account properties of the normal distribution, we obtain:

$$Pr(A > C) = Pr(Z > (C - \mu)/\sigma) \tag{5.2}$$

where Z is a random variable having a standard normal distribution.

From (5.1) and (5.2)

$$C = \mu + z_\varepsilon \sigma \tag{5.3}$$

Here, z_ε is $1-\varepsilon$ quantile for standard normal distribution. It can be approximated [35] as $z_\varepsilon \approx \sqrt{2\ln(1/\varepsilon) - \ln(2\pi)}$.

Thus, in case of n video calls with independent identically distributed transmission rates with $\mu_i = \mu_0$ and $\sigma_i = \sigma_0$, we have the following capacity allocation rule

$$C(n) = \mu_0\, n + z_\varepsilon \sigma_0 \sqrt{n}. \tag{5.4}$$

For the purposes of our approximation we assume that

$$\mu_0 = \kappa\, \sigma_0$$

where κ is a constant between 3 and 4 chosen such that the probability to have zero traffic would be low but positive. Then, it is natural to assume that peak rate $R = \mu_0 + \kappa\,\sigma_0$, or $R = 2\mu_0$.

The function on the right-hand side of (5.4) is concave but with very low curvature. We are going to implement a conservative over-dimensioning approach by approximating the concave function by its tangent line. It is easy to see that the linear approximation using the line tangent at n' is the following:

$$C(n) \approx \frac{1}{2} z_\varepsilon \sigma_0 \sqrt{n'} + (\mu_0 + \frac{z_\varepsilon \sigma_0}{2\sqrt{n'}})\, n \tag{5.5}$$

or introducing notation:

$$R' = \frac{1}{2} z_\varepsilon \sigma_0 \sqrt{n'}$$
$$r' = \mu_0 + \frac{z_\varepsilon \sigma_0}{2\sqrt{n'}}$$

we obtain the following fixed charge linear approximation:

$$C(n) \approx \begin{cases} 0 & \text{if } n = 0, \\ R' + r'\, n & \text{if } n \geq 1 \end{cases} \tag{5.6}$$

We will call R' the "fixed charge"; r' will be called the "slope coefficient".

For example, if $n' = 20$ and $\kappa = 4$,

$$R' = 1.17R$$
$$r' = 1.23r$$

Here, we assume that for a typical normal approximation $R = 2\mu$ is the peak rate, $r = \mu$ is the mean rate where μ is the mean of the normal distribution.

This approximation is plotted in Figure 5.2. In this figure '$\circ$' marks correspond to quantiles of n-fold convolution of normal distribution which are taken

at the highest permitted CLR value ($\varepsilon = 10^{-8}$); '$\times$' marks show required bandwidth per call; solid line plots formula (5.4); dashed line plots approximation (5.6) with $n' = 20$.

Figure 5.3 plots relative approximation error (expressed in per cent) for different choices of tangent point $n' = 3, 10, 20, 50$. When implementing this approach, the actual choice of the tangent point would be chosen based on the number of video calls that are typically multiplexed together on a link and/or the required precision.

Our approximation method can be easily generalized for the case when there is a mix of independent video calls $h \in \mathcal{H}$ having different distribution types. In this case expression (5.3) becomes:

$$C = \sum_{h \in \mathcal{H}} \mu_h \, n_h + z_\varepsilon \sqrt{\sum_{h \in \mathcal{H}} \sigma_h^2 \, n_h} \tag{5.7}$$

where n_h is the number of calls of type h. Next, we make another approximation by neglecting terms with smaller standard deviation values in the summation under square root in (5.7). This gives us the following linear approximation similar to (5.6):

$$C \approx \begin{cases} 0 & \text{if } \sum_{h \in \mathcal{H}} n_h = 0, \\ \max_{h \in L} R'_h + \sum_{h \in L} r'_h \, n_h & \text{if } \sum_{h \in \mathcal{H}} n_h \geq 1 \end{cases} \tag{5.8}$$

where R'_h and r'_h are parameters of call type h approximation.

Using alternative notation, let $\mathcal{K}_p$ be the set of calls multiplexed on the link (or VP) p then:

$$C(\mathcal{K}_p) \approx \begin{cases} 0 & \text{if } \mathcal{K}_p = \emptyset, \\ \max_{k \in \mathcal{K}_p} R'_k + \sum_{k \in \mathcal{K}_p} r'_k & \text{otherwise.} \end{cases} \tag{5.9}$$

We call the value $\max_{k \in \mathcal{K}_p} R'_k$ in (5.9) the "variable fixed" cost for its analogy to the fixed cost term and its variable nature.

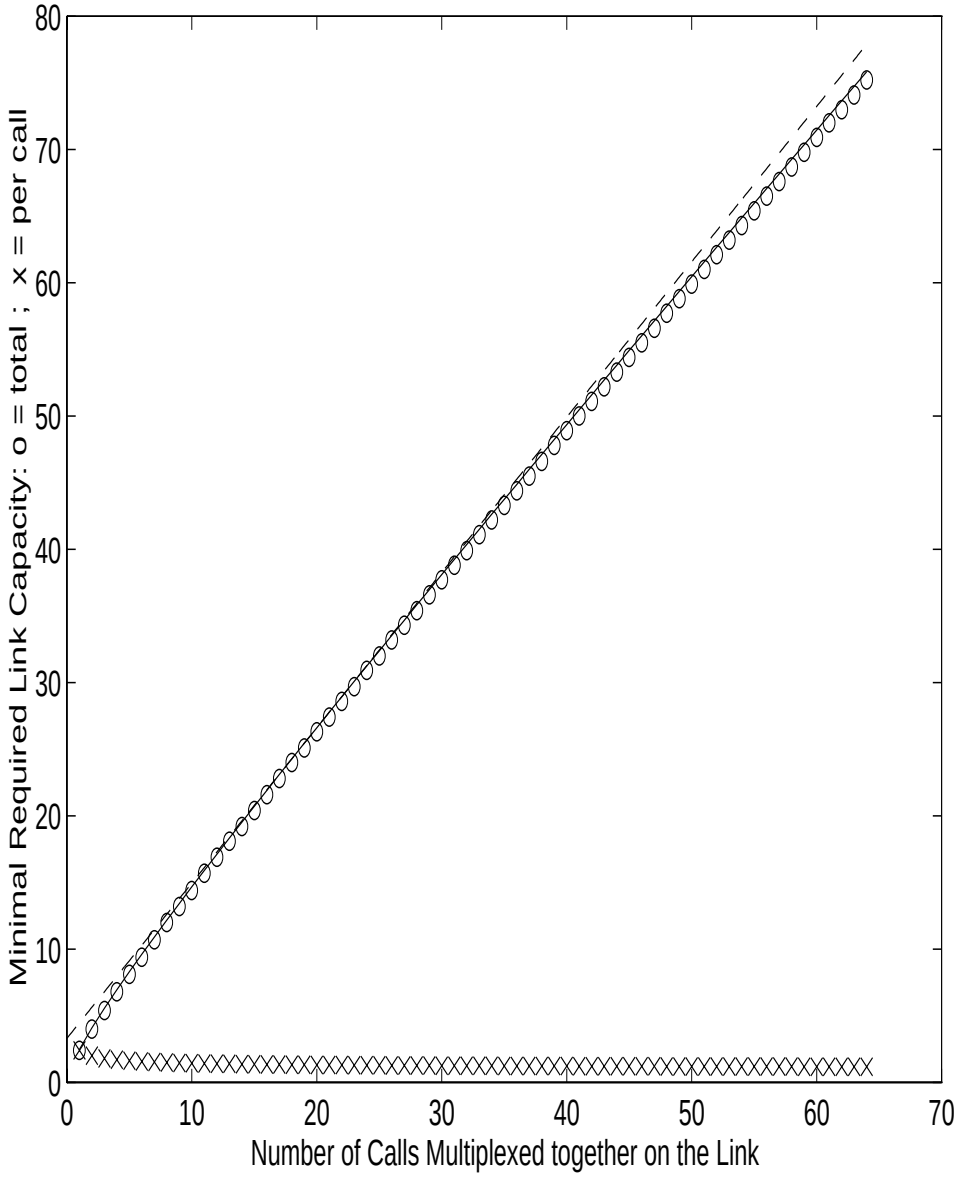

Figure 5.2: Linear Approximation for the Video Traffic Statistical Multiplexing

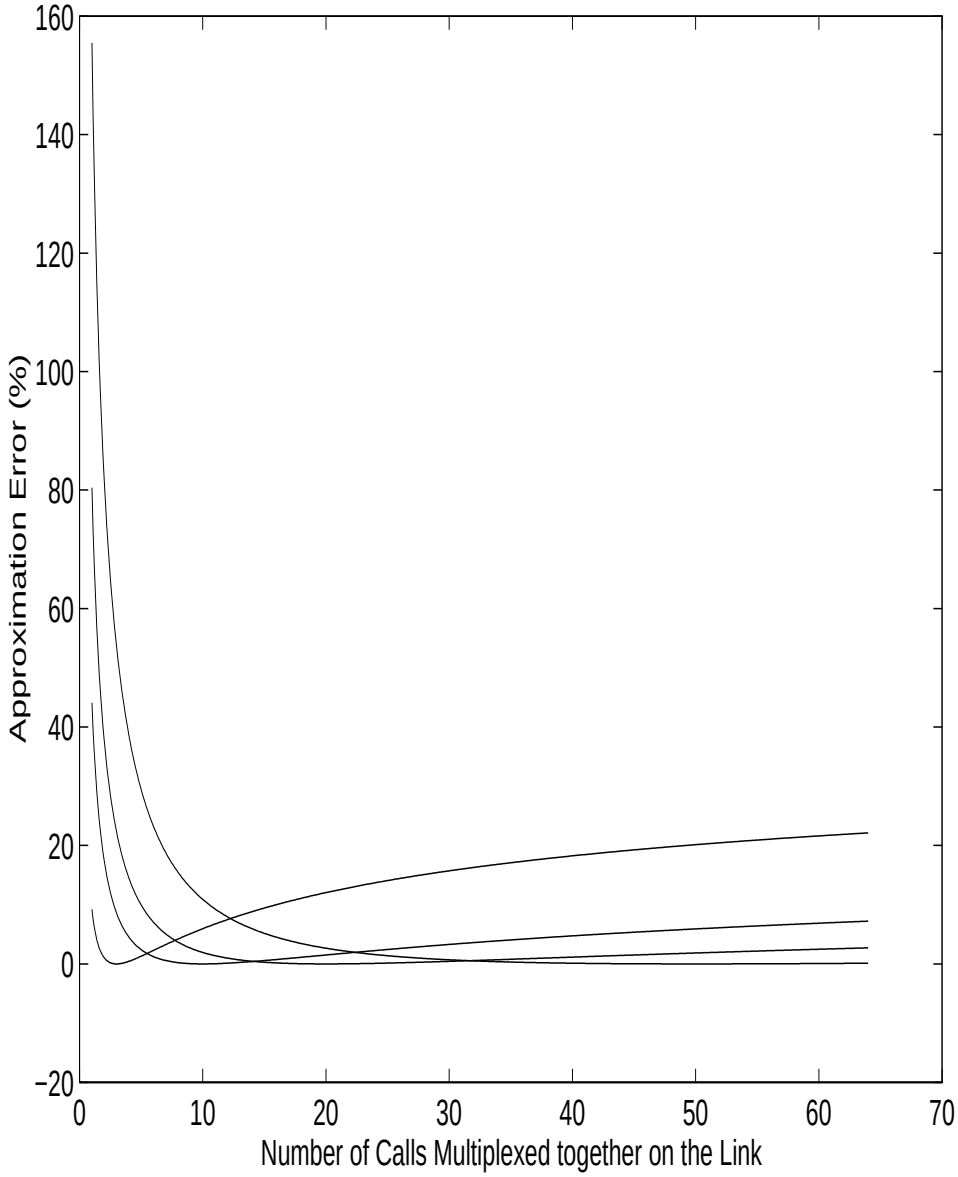

Figure 5.3: Relative Approximation Error for $n' = 3,\ 10,\ 20,\ 50$.

This approximation is illustrated in Figure 5.4. In this example every forth call has mean and standard deviation 3 times higher than the rest of the calls. We have the same notation as in Figure 5.2 for quantiles of n-fold convolutions and per call bandwidths. Solid line plots formula (5.7); dotted line plots the same formula but with averaged mean ($1.5\mu n = \frac{1}{4}(3\mu + (3\mu))n$) and standard deviation ($\sqrt{3}\sigma n = \sqrt{\frac{1}{4}(3\sigma^2 + (3\sigma)^2)}n$) where μ and σ are parameters of the lower bandwidth calls; dashed line plots linear approximation (5.8) (or, its equivalent (5.9)).

5.1.2 Data Traffic Capacity Allocation

In this section we derive a piece-wise linear approximation for the capacity allocation rule in case of data traffic. We assume that the distribution of the data transmission rate is discrete, with only two non-zero probabilities: at zero and peak traffic values. If several non-correlated sources are multiplexed together on a link, their traffic distribution is calculated as convolution of their original distributions. The minimal required bandwidth is determined as the quantile of the negative cumulative density function (c.d.f.) taken at maximal permitted CLR.

Let p (q) be the probability the source is transmitting (silent). Since the source is considered as "on/off", $p + q = 1$. Suppose all sources are i.i.d. If statistical multiplexing is not employed, the bandwidth required to transmit data from n sources with peak rate R is simply nR. If statistical multiplexing is used, the required bandwidth will be $(n - g)R$. We will call g statistical

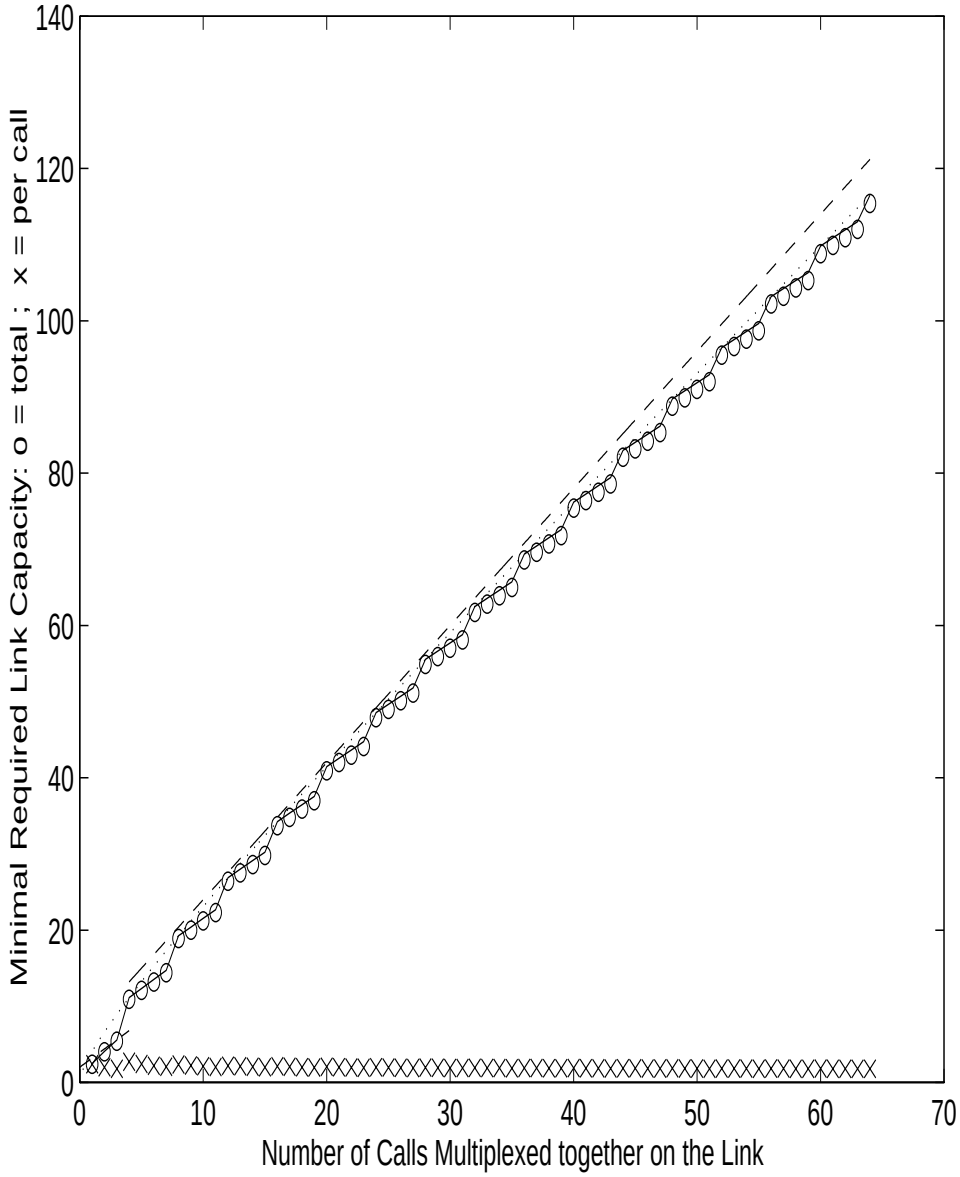

Figure 5.4: Linear Approximation for the Video Traffic Statistical Multiplexing

multiplexing gain for data transmission. Clearly,

$$g(n) = \sup\{g : \sum_{k=0}^{g} \binom{n}{k} p^{n-k} q^k < \varepsilon\} \tag{5.10}$$

where ε is the highest permitted CLR.

It is easy to see that if n is small enough so that $p^n > \varepsilon$, there is no statistical multiplexing gain, i.e. $g(n) = 0$. In this case the minimal required bandwidth is equal to the total peak rate: $b_0(n) = R\,n$. On the other hand, as n gets larger, the binomial distribution in (5.10) can be approximated by normal with mean np and standard deviation $\sqrt{npq}$. Then, following procedure in section 5.1.1, we approximate minimal required bandwidth as:

$$C(n) \approx npR + z_\varepsilon R\sqrt{npq} \tag{5.11}$$

As we did in section 5.1.1, we further approximate 5.11 with a linear function $b_0 \approx \hat{R} + r'\,n$ where $\hat{R} = \lceil \ln\varepsilon / \ln p \rceil qR$. Thus the approximating function is piece-wise linear and consists of two pieces:

$$C(n) \approx \begin{cases} R\,n & \text{if } n \le \lceil \ln\varepsilon / \ln p \rceil, \\ \hat{R} + r'\,n & \text{otherwise.} \end{cases} \tag{5.12}$$

Notice, that the first equation in (5.12) is exact. This approximation is illustrated in Figure 5.5. Here, we consider negative c.d.f. for the n-fold convolutions of discrete distribution with $Prob\{transmission\ rate = 0Mb/s\} = 0.9$ and $Prob\{transmission\ rate = 1Mb/s\} = 0.1$ where $n = 1, \ldots, 64$. Quantiles, taken at the highest permitted CLR value ($\varepsilon = 10^{-8}$), are plotted in Figure 5.5 where the '$\circ$' marks correspond to the total required bandwidth, '$\times$' marks show required bandwidth per call, solid line plots approximation (5.11), and dashed line plots approximation (5.12). As it is seen in the figure, the approximated

function for the minimal required bandwidth is step-wise. This is explained by the fact that the minimal required bandwidth is a multiple of peak rate.

Now consider statistical multiplexing of data calls of different types $h \in \mathcal{H}$ with different "on/off" probabilities $p_h + q_h = 1$ and different peak rates R_h. Then following our models for i.i.d. data calls and video calls of different types, we obtain that:

$$C \approx \begin{cases} \sum_{h\in\mathcal{H}} R_h n_h & \text{if } -\sum_{h\in\mathcal{H}} n_h \ln p_h \le -\ln \varepsilon, \\ \max_{h\in\mathcal{H}} \hat{R}_h + \sum_{h\in\mathcal{H}} r'_h n_h & \text{otherwise.} \end{cases} \tag{5.13}$$

Notice, that the first equation in (5.13) is exact.

5.1.3 Mixing Heterogeneous Calls

The following model is proposed for the case when there are calls of different types multiplexed together. Let $\mathcal{H}_{video}$ and $\mathcal{H}_{data}$ be the sets of different video and data call types, respectively; $\mathcal{H} = \mathcal{H}_{video} \bigcup \mathcal{H}_{data}$. Then

$$C_{mix} \approx \begin{cases} C(\mathcal{K}_{video}) + \sum_{h\in\mathcal{H}_{data}} R_h n_h & \text{if } -\sum_{h\in\mathcal{H}} n_h \ln p_h \le -\ln \varepsilon, \\ \max(\max_{h\in\mathcal{H}_{data}} \hat{R}_h, \max_{h\in\mathcal{H}_{video}} R'_h) + \sum_{h\in\mathcal{H}} r'_h n_h & \text{otherwise.} \end{cases} \tag{5.14}$$

where $C(\mathcal{K}_{video})$ is obtained from (5.9).

Obviously, the second part of the formula (5.14) is a slight modification of (5.8) where the mix of data calls is approximated as normally distributed. Concerning the first part, it is easy to see that this implements a conservative approach by ignoring statistical multiplexing between data and video calls. However, since in this case both bounds are rather tight, the loss of accuracy is not very high. In Figure 5.6, we plot the minimal bandwidth to meet CLR requirements. In this example we consider several data calls multiplexed together with one video call. Data calls are i.i.d. having the same parameters as in the example

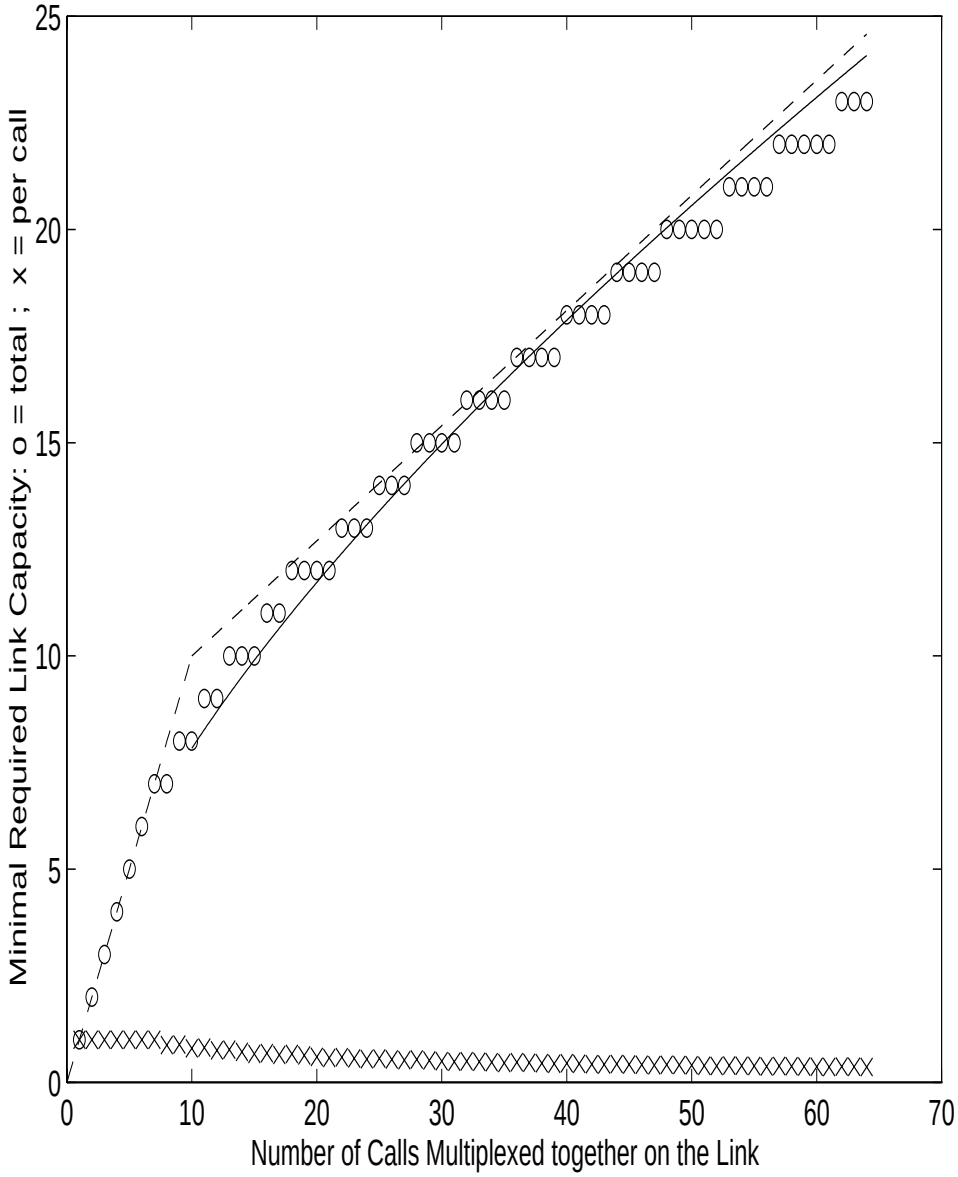

Figure 5.5: Illustration of the Statistical Multiplexing

pictured in Figure 5.5, section 5.1.2 ($R = 1$, $p = 0.1$) and the video call has peak rate 10 and mean rate 5. The numbers on the horizontal axis show the number of data calls and the dashed lines are a linear approximation (5.13). As can be seen from the plot, relative error of the approximation does not exceed 20%.

5.1.4 A Comparison with Related Results

Gaussian Self-Similar Traffic

The network traffic stochastic time dependence is widely regarded to be statistically self-similar meaning that it shows fractal behavior over various time scales. This concept is based on traffic measures taken at Bellcore [27, 26]. It implies the following capacity allocation rule [8]:

$$C = m + \Phi^{-1}(\varepsilon)\alpha^{1/(2H)}x^{-(1-H)/H}m^{1/(2H)} \tag{5.15}$$

where

m is total mean transmission rate;

α is transmission rate variance;

x is buffer size;

H is Hurst parameter, $1/2 < H < 1$.

It can be seen that expression for C in (5.15) consists of two terms: linear and sublinear on mean transmission rate m. The latter is dominated by the first term. Thus, applying steps outlined in section 5.1.1, we can approximate C as a function of m and derive a fixed charge linear approximation similar to (5.4).

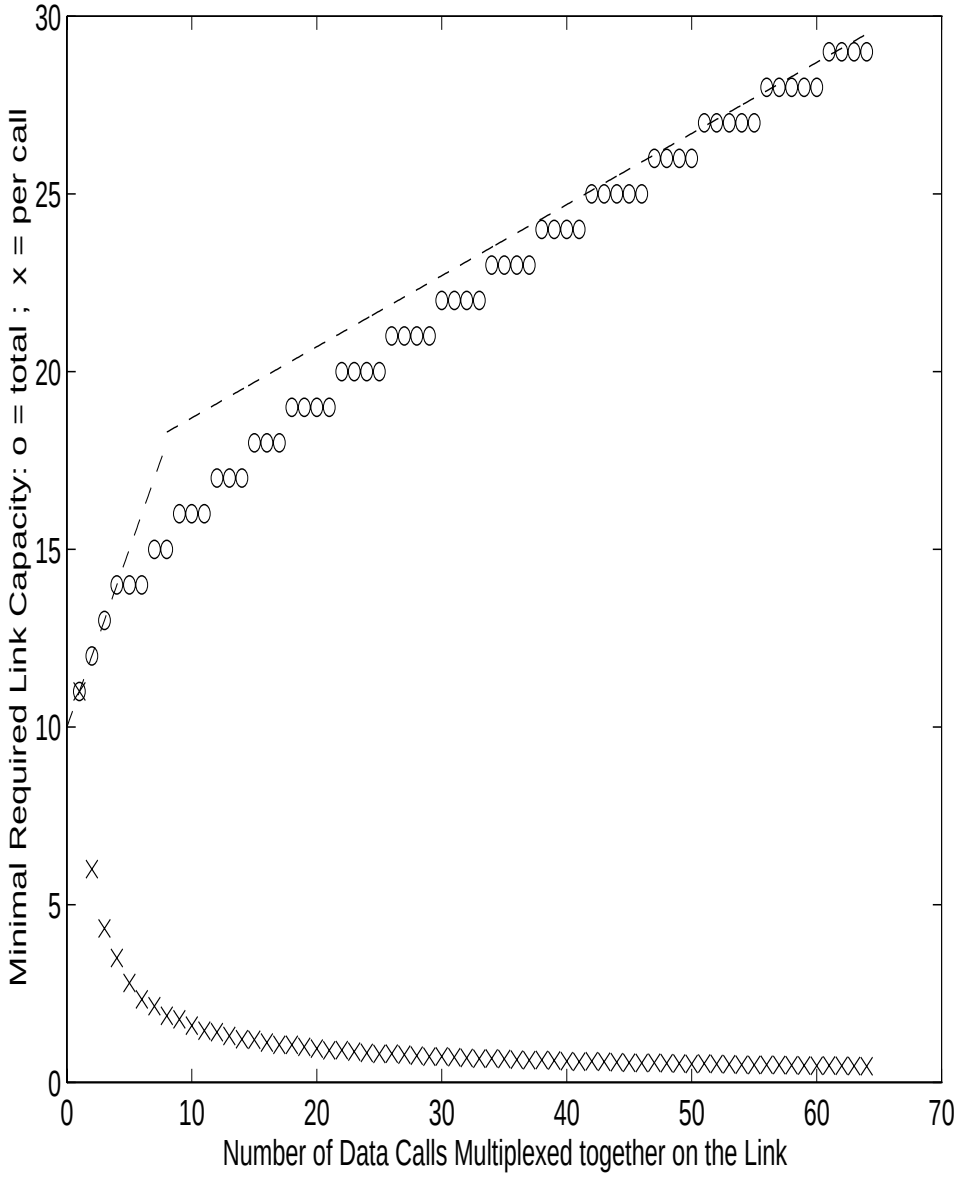

Figure 5.6: Relative Approximation Error for the Mixes of Heterogeneous Calls

Two-Piece Linear Approximation

In references [28, 29], the authors showed that the minimum capacity required for a number of "on–off" VBR connections multiplexed together can be approximated as a two-piece linear function similar to (5.12). In [28], the authors proposed a simple analytical model which results in a two-piece linear approximation. In [29], in addition to two linear pieces, the authors considered a "transient" curve between the two approximating lines.

In both of these approaches, the first linear piece of the approximation corresponded to the total peak rate capacity allocation for all multiplexed calls; the second piece had the slope approximately equal to the mean rate of the calls. Thus, this approximation is similar to (5.12) derived in section 5.1.2.

5.2 Problem Formulation

The problem formulated in this section can be described as follows. Suppose there is a certain number of various call connections to be set up between each node pair in a network. Each call is characterized by its mean and peak bandwidths and its largest tolerable delay. In order to simplify routing and reduce call delay and network control overhead, we wish to establish multilink VPs between some pairs of nodes. These VPs are considered as logical macro-links. The problem is to choose which node pairs are to be connected by VPs, what route each VP should take and the amount of capacity to be allocated to each VP.

The problem constraints are dictated by the following two QOS requirements:

1. maximum call delay;

2. minimum cell loss probability.

The essential property of cell switching in ATM networks is that queuing delay is practically completely eliminated at the intermediate switching nodes of a VP through which the cell is routed [25, 35, 41]. Thus, in our model, we assume an end-to-end delay proportional to the number of traversed VPs.

A cell loss probability constraint translates into a capacity allocation rule which provides a guaranteed cell loss probability. As was shown earlier, the capacity allocation rule is essentially non-linear. For the purposes of our model we will be using its piece-wise linear approximations derived in section 5.1.

5.2.1 Bi-Criteria Objective Function

A good VP layout involves a fundamental tradeoff between the network throughput and its grade of service. Consider the quality of service parameter, transmission delay. It is the sum of propagation and queuing/switching delays. The first part is negligibly small in an ATM network unless it contains satellite links. The second part is determined by the number of VP-to-VP switching points since, as was assumed earlier, delay at the intermediate VP nodes is relatively small. Hence, transmission delay is substantially decreased by reducing the number of VP-to-VP switching nodes.

Thus, establishing "long" VPs decreases transmission delay and network control overhead and simplifies routing. On the other hand, establishing too many low-capacity VPs may take away statistical multiplexing gain obtained by sharing VP capacity among multiple variable-bit rate (VBR) calls.

However, using too many "short" VPs in order to take advantage of statistical multiplexing generally results in an unacceptably high call delay and network

	Short, High-capacity VPs	Long, Low-capacity VPs
Throughput	larger	smaller
Control Overhead	larger	smaller
Delay	larger	smaller

Table 5.1: VP Layout: Fundamental Tradeoff

control overhead.

This concept is summarized in the Table 5.1.

In order to capture the VP layout tradeoff, we formulate the problem as bi-criteria optimization:

- **maximize** the minimum over all Origin–Destination (O–D) pairs and traffic classes of the ratio of admitted traffic to traffic demanded;
- **minimize** the total mean traffic flow through the VP switching nodes (except at origin and destination).

The ratio in the first part of the objective function will be called "relative throughput". Maximizing the minimum of the relative throughput corresponds to the concept of "fairness" in the ATM network management. In our model, relative throughput may take values greater than one which means that network can accommodate higher throughput than projected demand.

The second part of the objective function represents the amount of traffic flow going through VP-to-VP switches. Since these nodes are intermediate for the virtual channels, this part will be called "flow through intermediate nodes"

or "intermediate flow", for short. Based on our assumption, this flow amount is a measure of overall transmission delay and control overhead. Note that if there was a direct VP between every O–D pair then this value would be zero.

The solution to this type of optimization problem is a set of individual solutions to optimization problems with objective function composed of the two criteria taken with certain weights. The whole set (also known as Pareto-optimal solution set) is obtained by varying these weights. It is convenient to represent the solution set as points on the relative throughput – intermediate flow plane. It will be also called a tradeoff curve.

5.2.2 Basic VP Layout Formulation

In this section, we give a formulation of the bi-criteria optimal VP layout problem which we consider to be basic in the sense that we do not include fault tolerant constraints into this formulation.

Let us represent a telecommunication network as a graph $\mathcal{G} = (\mathcal{N}, \mathcal{A})$ where $\mathcal{N}$ and $\mathcal{A}$ are the sets of nodes and one-directional links, respectively. Denote the set of all possible routes between nodes i and j by $\mathcal{P}_{ij}$ and let $\mathcal{P} = \bigcup_{i,j \in \mathcal{N}} \mathcal{P}_{ij}$ be the set of all possible virtual paths in the network. Introduce the following notation:

V_p , capacity assigned to VP p (unknown variable);

B_e , capacity of physical link e (fixed).

$$\delta_{pe} = \begin{cases} 1 & \text{if VP } p \text{ passes through the link } e, \\ 0 & \text{otherwise;} \end{cases}$$

$OUT(i)$ $(IN(i))$, set of candidate VPs beginning (ending) at node $i \in \mathcal{N}$.

Let $\mathcal{K}$ be the set of calls to be (re)routed. This call set can be viewed as a known network traffic pattern or a set of calls which exists in the network at a particular time, i.e. a "snapshot". With each call k we have associated its origin and destination nodes $O(k)$ and $D(k)$, respectively, its maximum tolerable delay d_k and mean and peak bandwidths r_k and R_k, respectively. We introduce two groups of integer zero-one call variables:

$$x_{kp} = \begin{cases} 1 & \text{if call } k \text{ uses VP } p, \\ 0 & \text{otherwise;} \end{cases}$$

$$y_k = \begin{cases} 1 & \text{if call } k \text{ is accepted,} \\ 0 & \text{if it is blocked.} \end{cases}$$

Thus, any subset of "core" call set $\mathcal{K}_{core} = \{k \in \mathcal{K} : y_k = 1\}$ is guaranteed to be routed.

We assume that call set $\mathcal{K}$ consists of two subsets: $\mathcal{K}^{video}$ and $\mathcal{K}^{data}$, containing video and data calls, respectively. Since voice calls are CBR, for the purposes of our modeling, we consider them as a special case of video calls with zero fixed charge. In general, we consider a set $\mathcal{H}$ of all call classes.

Let $\mathcal{K}_{ij}^h$ be the set of class $h \in \mathcal{H}$ calls with the origin i and destination j and r_k be the mean transmission rate of call k. Then the ratio of the total traffic routed between nodes i and j to the total traffic demand between these nodes is

$$\sum_{k \in \mathcal{K}_{ij}^h} r_k y_k / \sum_{k \in \mathcal{K}_{ij}^h} r_k \ .$$

We would like to maximize, ρ, the lowest of these ratios over all traffic classes and node pairs and at the same time to minimize the traffic flow through the VP-to-VP switches. This gives us the following expression for the objective

function:

$$\text{Maximize} \qquad \rho - \eta \sum_{k \in \mathcal{K}} r_k \sum_{p \in \mathcal{P} - OUT(O(k))} x_{kp} \tag{5.16}$$

subject to

$$\sum_{k \in \mathcal{K}_{ij}^h} r_k y_k \geq \rho \sum_{k \in \mathcal{K}_{ij}^h} r_k \quad \text{for all } i,\, j \in \mathcal{N},\, h \in \mathcal{H} \tag{5.17}$$

Here, coefficient η is used to put certain weights on the two criteria which are optimized at the same time.

As was shown in section 5.1.2, the amount of capacity required to carry data traffic is a concave function of the number of calls carried. We approximate this function by a two-piece linear function. The first piece corresponds to the case when no statistical multiplexing gain is observed and the bandwidth allocated per call is exactly its peak rate. The second linear piece corresponds to the case when statistical multiplexing gain is in effect. We will call these cases "non-saturated" and "saturated", respectively. Since the linear approximation is concave, we need one additional zero-one variable per link and two continuous variables per call and link pair to provide an adequate linear programming formulation.

$$s_p = \begin{cases} 1 & \text{if the data traffic is saturated on VP } p; \\ 0 & \text{otherwise;} \end{cases}$$

ζ_{kp} is the slope of the "non-saturated" linear piece, when traffic is not saturated, zero, otherwise.

ξ_{kp} is the slope of the "saturated" linear piece, when traffic is saturated, zero, otherwise.

The next group of constraints forces s_p to one when traffic is saturated, that is, according to (5.14), when $\sum_{k \in \mathcal{K}^{data}} - \ln p_k x_{kp} \geq - \ln \varepsilon$, otherwise s_p is forced to

zero.

$$(1 - s_p)\,(-\ln \varepsilon) \leq \sum_{k \in \mathcal{K}^{data}} -\ln\, p_k x_{kp} - (-\ln \varepsilon) \leq$$
$$s_p \sum_{k \in \mathcal{K}^{data}} -\ln p_k \quad \text{for all } p \in \mathcal{P}, \tag{5.18}$$

Coefficient $(-\ln \varepsilon)$ in the left hand side of (5.18) is chosen as a lower bound on the expression in the middle part of (5.18); coefficient $\sum_{k \in \mathcal{K}^{data}} -\ln p_k$ in the right hand side part is chosen as an upper bound on the expression in the middle part.

The next three groups of constraints define a fixed charge, Δ_p^{data}, and slopes for two-piece linear approximation to capacity allocation for data traffic.

$$\Delta_p^{data} \geq R_k(x_{kp} - (1 - s_p)) \quad \text{for all } p \in \mathcal{P},\ k \in \mathcal{K}^{data}, \tag{5.19}$$

$$R_k\,(x_{kp} - s_p) \leq \zeta_{kp} \leq (1 - s_p)R_k \quad \text{for all } p \in \mathcal{P},\ k \in \mathcal{K}^{data}, \tag{5.20}$$

$$r_k(x_{kp} - (1 - s_p)) \leq \xi_{kp} \leq r_k s_p \quad \text{for all } p \in \mathcal{P},\ k \in \mathcal{K}^{data}, \tag{5.21}$$

$$0 \leq s_p \leq 1 \qquad \text{and integer} \quad \text{for all } p \in \mathcal{P}, \tag{5.22}$$

$$\zeta_{kp},\ \xi_{kp} \geq 0 \quad \text{for all } p \in \mathcal{P},\ k \in \mathcal{K}, \tag{5.23}$$

The following two groups of constraints define the total fixed charge as the maximum of fixed charges over all call classes.

$$\Delta_p \geq R_k x_{kp} \qquad \text{for all } p \in \mathcal{P},\ k \in \mathcal{K}^{video}, \tag{5.24}$$

$$\Delta_p \geq \Delta_p^{data} \qquad \text{for all } p \in \mathcal{P}, \tag{5.25}$$

The next two constraint groups define capacity allocated to VP p and place capacity constraints on physical link e, respectively.

$$V_p \geq \Delta_p + \sum_{k \in \mathcal{K}^{video}} r_k x_{kp} + \sum_{k \in \mathcal{K}^{data}} (\zeta_{kp} + \xi_{kp}) \qquad \text{for all } p \in \mathcal{P}, \tag{5.26}$$

$$\sum_{p \in \mathcal{P}} \delta_{pe} V_p \leq B_e \qquad \text{for all } e \in \mathcal{A}, \tag{5.27}$$

$$V_p \geq 0 \qquad \text{for all } p \in \mathcal{P}. \tag{5.28}$$

Flow conservation constraints introduced next and integrality constraints provide a single path between origin and destination of the accepted calls.

$$\sum_{p \in OUT(i)} x_{kp} - \sum_{p \in IN(i)} x_{kp} = \begin{cases} -y_k & \text{if } D(k) = i \\ y_k & \text{if } O(k) = i \\ 0 & \text{otherwise} \end{cases} \quad \text{for all } i \in \mathcal{N},\ k \in \mathcal{K}, \quad (5.29)$$

$$0 \leq y_k \leq 1 \ \text{ and integer} \quad \text{for all } k \in \mathcal{K}, \quad (5.30)$$

$$x_{kp} \geq 0 \ \text{ and integer} \quad \text{for all } p \in \mathcal{P},\ k \in \mathcal{K}, \quad (5.31)$$

Finally, the basic formulation of the VP layout problem is given by

BAS: Maximize (5.16) subject to (5.17–5.31)

It can be seen from the problem formulation that the best way to reduce the intermediate flow is to establish VPs between each pair of nodes. However, if this is done, then we might not get enough statistical multiplexing gain and, as a result, would not have enough capacity to route all calls. On the other hand, establishing VPs using exactly one physical link maximizes statistical multiplexing gain but may make intermediate flow rather high. Short VPs also tend to increase the network control overhead. In order to favor longer VPs, objective function coefficient η should be made larger.

Below, we study the properties of the formulation by considering two marginal cases.

Suppose, we perform a pure relative throughput maximization ignoring the second part of the objective function (5.16), or equivalently, coefficient η is set to zero. Then, it is easy to show that in the optimal solution each VP will traverse one physical link and one physical link will contain at most one VP. Indeed, if there are VPs traversing more than one physical link, then they can be partitioned into sequences of VPs associated with the traversed physical links

and the objective function will not be affected. After that, any existing VP will share exactly one physical link with some other VPs. If, now, all VPs sharing the same link are merged together it will loosen constraint (5.27) and the solution will stay feasible. We can formulate this statement as the following

Proposition 1 *If in the problem* **BAS** $\eta = 0$*, then there is an optimal solution having VPs equivalent to physical links.*

Now, assume there exists a VP layout connecting each O–D pair by a set of VPs and admitting at least one call in each class between this pair. Clearly, with this layout the second part of the objective function becomes zero and solution value becomes positive. On the other hand, for any feasible solution with non-zero second criterion (i.e. non-zero intermediate flow), coefficient η can be chosen large enough so that the solution value be negative. If such end-to-end layout does not exist, then zero solution will still be feasible. This argument implies the following

Proposition 2 *If coefficient η in objective function (5.16) is chosen sufficiently large, for any optimal solution the second term of the objective function will equal zero.*

5.2.3 Fault Tolerant Routing Constraints

In this section we expand our model to capture link failure case. In this case we wish to reroute calls as fast as possible taking advantage of the two level VC/VP ATM network architecture. That is, together with primary VPs, we want to establish backup VPs so that in the event of failure only VP level rerouting is required. Since this kind of rerouting does not touch the VC level, particularly

VC identifier tables, it is expected to work very fast, changing only VP identifier tables. We propose two different strategies:

1. Backup VPs use paths which are link-disjoint from the primary VPs. When a link fails, all affected primary VPs are replaced by their backups.

2. A backup VP is designed for each VP–link pair, so that potentially, there is a different replacement for each link on the primary VP.

We start with an assumption that between each node pair there are at most one primary and one backup VP established. If there is a need to establish more than one VP between the same end-points (e.g., to accommodate multiple traffic types), then our problem formulation can be generalized quite easily.

Link-Disjoint Backup Paths

In addition to the variables and notation introduced earlier in this chapter, we define the following zero-one and continuous decision variables for backup VPs.

$$v_p = \begin{cases} 1 & \text{if route } p \in \mathcal{P}_{ij} \text{ is used as a primary VP,} \\ 0 & \text{otherwise;} \end{cases}$$

$$u_r = \begin{cases} 1 & \text{if route } r \in \mathcal{P}_{ij} \text{ is used as a backup VP,} \\ 0 & \text{otherwise;} \end{cases}$$

U_r = capacity assigned to backup VP r.

If v_p (u_r) is equal to zero, then there is no capacity assigned to the primary (backup) VP, which means that $V_p = 0$ ($U_r = 0$). If any of the variables v_p or u_r is one, then capacity of the corresponding VP is non-zero but does not exceed the smallest link capacity on the VP route. This relation can be formulated as

follows:

$$V_p \leq v_p \min_{e \in p} B_e \quad \text{for all } p \in \mathcal{P} \tag{5.32}$$

$$U_r \leq u_r \min_{e \in r} B_e \quad \text{for all } r \in \mathcal{P} \tag{5.33}$$

Since there is at most one primary and backup VP between each pair of nodes,

$$\sum_{p \in \mathcal{P}_{ij}} v_p = \sum_{r \in \mathcal{P}_{ij}} u_r \leq 1 \quad \text{for all } \mathcal{P}_{ij}. \tag{5.34}$$

Since the backup VP should have the same assigned capacity,

$$\sum_{p \in \mathcal{P}_{ij}} V_p = \sum_{r \in \mathcal{P}_{ij}} U_r \quad \text{for all } \mathcal{P}_{ij}, \tag{5.35}$$

Finally, the link-disjoint condition is:

$$\sum_{p \in \mathcal{P}_{ij}} v_p \delta_{pe} + \sum_{r \in \mathcal{P}_{ij}} u_r \delta_{re} \leq 1 \quad \text{for all } \mathcal{P}_{ij},\, e \in \mathcal{A} \tag{5.36}$$

Before we formulate the link capacity constraint, let us introduce one more group of variables:

Z_{pr}, the amount of flow transferred from the primary VP p to the backup VP r, when the primary VP is affected by the link failure.

In our model, we assume all flow on the primary VP is transferred to the backup VP so that $Z_{pr} = V_p u_r = v_p U_r$ since only in case of $v_p = u_r = 1$ is capacity transferred from a primary VP p to its backup r. However, expressing Z_{pr} in this way makes the problem non-linear which is not desirable. Instead, the following group of linear constraints gives the equivalent expression:

$$\sum_{r \in \mathcal{P}_{ij}} Z_{pr} = V_p \quad \text{for all } p \in \mathcal{P}_{ij} \text{ and all } \mathcal{P}_{ij}, \tag{5.37}$$

$$\sum_{p \in \mathcal{P}_{ij}} Z_{pr} = U_r \quad \text{for all } r \in \mathcal{P}_{ij} \text{ and all } \mathcal{P}_{ij}, \tag{5.38}$$

Suppose link e fails. Consider the change of flow through an arbitrary link $e' \neq e$. The initial flow through that link was $\sum_{p \in \mathcal{P}} \delta_{pe'} V_p$. Since link e fails, all VPs p satisfying $\delta_{pe} = 1$ are taken down and the total flow decrease on link e' becomes $\sum_{p \in \mathcal{P}} \delta_{pe} \delta_{pe'} V_p$. The increase in the flow on link e' caused by new backup VPs is $\sum_{\text{all } \mathcal{P}_{ij}} \sum_{p,r \in \mathcal{P}_{ij}} \delta_{re'} \delta_{pe} Z_{pr}$. Finally, capacity constraints for the links become:

$$\sum_{p \in \mathcal{P}} (1 - \delta_{pe}) \delta_{pe'} V_p + \sum_{\text{all } \mathcal{P}_{ij}} \sum_{p,r \in \mathcal{P}_{ij}} \delta_{re'} \delta_{pe} Z_{pr} \leq B_{e'} \quad \text{for all } e' \neq e \in \mathcal{A} \tag{5.39}$$

The problem for the link-disjoint path is given by (5.16–5.39) and zero-one constraints for v_p and u_r as well as non-negativity constraints for U_r and Z_{pr}:

$$0 \leq v_p \leq 1 \quad \text{and integer} \quad \text{for all } p \in \mathcal{P}, \tag{5.40}$$

$$0 \leq u_r \leq 1 \quad \text{and integer} \quad \text{for all } r \in \mathcal{P}, \tag{5.41}$$

$$U_r \geq 0 \quad \text{for all } r \in \mathcal{P}, \tag{5.42}$$

$$Z_{pr} \geq 0 \quad \text{for all } p, r \in \mathcal{P}. \tag{5.43}$$

Link-Dependent Backup Paths

In order to implement the second strategy, i.e. link-dependent rerouting, we treat backup route r as paired with a potentially failed link e. Then, variables u_r and U_r are replaced with link-specific variables u_{re} and U_{re}, respectively, which correspond to the routes chosen as backup VPs and activated in the case of link e failure.

The problem formulation for the link dependent rerouting is similar to link disjoint formulation. Here constraints (5.36) forcing the link-disjoint paths become

$$\delta_{re} u_{re} = 0 \quad \text{for all } r \in \mathcal{P}, \text{ all } e \in \mathcal{A}, \tag{5.44}$$

which bar backup VPs from passing through the link e.

If primary VP does not pass through the link e, then the corresponding backup VP u_{re} is not setup. This requirement, together with a condition limiting the number of VPs with the same end points to one, replaces constraints (5.34–5.35) as follows:

$$\sum_{p \in \mathcal{P}_{ij}} v_p \leq 1 \qquad \text{for all } \mathcal{P}_{ij} \tag{5.45}$$

$$\sum_{r \in \mathcal{P}_{ij}} u_{re} = \sum_{p \in \mathcal{P}_{ij}} \delta_{pe} v_p \quad \text{for all } \mathcal{P}_{ij}, \text{ all } e \in \mathcal{A} \tag{5.46}$$

$$\sum_{r \in \mathcal{P}_{ij}} U_{re} = \sum_{p \in \mathcal{P}_{ij}} \delta_{pe} V_p \quad \text{for all } \mathcal{P}_{ij}, \text{ all } e \in \mathcal{A} \tag{5.47}$$

Constraints (5.46–5.47) do not let backup VP u_{re} be set up if the primary VP with the same end points does not go through the link e. Constraints (5.38) become:

$$U_{re} = \sum_{p \in \mathcal{P}_{ij}} Z_{pr} \delta_{re} \quad \text{for all } r \in \mathcal{P}_{ij}, \text{ all } \mathcal{P}_{ij}, \text{ and all } e \in \mathcal{A}, \tag{5.48}$$

Finally, we have the following formulation of the problem.

(5.16)

subject to

(5.17–5.33), (5.44–5.47), (5.37), (5.48), (5.39–5.43)

The solution to the problem is a set of link specific backup VPs which are activated when the corresponding link fails.

5.2.4 VP contraction

In the previous section it was assumed that any primary VP should be backed up by a backup VP with the same capacity since the currently routed calls should

not be affected by a link failure. This may cause a substantial reduction in the number of calls routed over a VP in order to guarantee their complete tolerance to the link failure. Alternately, if it is possible to take down some calls in case of link failure, the number of calls routed under normal conditions may be much higher. In this case the backup VP may have lower assigned capacity than the primary. Notice that we do not try to reroute calls to different paths due to large potential number of the affected calls and, in many cases, unreasonable complications in the network management protocol. We assume that it is quite easy to take down some calls. We study this model by modifying the formulation presented in section 5.2.3.

Link-Disjoint Backup Paths

In the case of VP contraction, in addition to call-to-VP assignment variables x_{kp}, we must introduce variables indicating which calls (VCs) are rerouted together with a VP when it is affected by a failure.

$$x'_{kp} = \begin{cases} 1 & \text{if call } k \text{ uses VP } p \text{ and is not dropped when failure occurs} \\ 0 & \text{otherwise;} \end{cases}$$

Variables x_{kp} and x'_{kp} define two VP sets $\mathcal{P}_k$ and $\mathcal{P}'_k$, respectively. The first set is the set of VPs used for call k routing, the second is the set of VPs where call k is backed up. Clearly, $\mathcal{P}'_k \subset \mathcal{P}_k$, or

$$x'_{kp} \leq x_{kp} \quad \text{for all } k \in \mathcal{K} \text{ and } p \in \mathcal{P} \tag{5.49}$$

Let f_p be the probability that VP p fails during the next time period. Since any two VPs in the set $\mathcal{P}_k$ use different links ($p_1,\ p_2 \in \mathcal{P}_k,\ p_1 \bigcap p_2 = \emptyset$) and links fail independently, failures of VPs p_1 and p_2 are also independent. Since call k

is taken down if and only if VP $p \in \mathcal{P}_k \backslash \mathcal{P}'_k$ fails, then

$$Prob\{\text{call } k \text{ is taken down}\} = 1 - \prod_{p \in \mathcal{P}_k \backslash \mathcal{P}'_k} (1 - f_p) =$$

$$1 - \prod_{p\,:\, x_{kp} - x'_{kp} = 1} (1 - f_p) \approx \sum_{p\,:\, x_{kp} - x'_{kp} = 1} f_p = \sum_{p \in \mathcal{P}} f_p (x_{kp} - x'_{kp}).$$

The above approximation is valid when we neglect the multiple failures case and assume f_p is small.

The expected loss of traffic flow due to a link failure then becomes

$$\sum_{k \in \mathcal{K}} r_k \sum_{p \in \mathcal{P}} f_p (x_{kp} - x'_{kp}) \tag{5.50}$$

Since the backup VP in the current model may have a bandwidth lower than the primary, the corresponding constraints are to be modified. Constraints (5.35) become

$$\sum_{p \in \mathcal{P}_{ij}} V_p \geq \sum_{r \in \mathcal{P}_{ij}} U_r \quad \text{for all } \mathcal{P}_{ij}, \tag{5.51}$$

and (5.37) becomes

$$\sum_{r \in \mathcal{P}_{ij}} Z_{pr} \leq V_p \quad \text{for all } p \in \mathcal{P}_{ij} \text{ and all } \mathcal{P}_{ij}, \tag{5.52}$$

whereas (5.38) stays the same. A failed primary VP is replaced by a lower capacity VP with new capacity $\sum_{r \in \mathcal{P}(p)} Z_{pr}$ where $\mathcal{P}(p)$ is a group of routes between nodes i and j which are the end points for VP p. Hence the new constraints similar to (5.26-5.24) are added for the remaining calls with guaranteed rerouting.

$$\sum_{r \in \mathcal{P}(p)} Z_{pr} \geq \Delta_p + \sum_{k \in \mathcal{K}} r_k x'_{kp} \quad \text{for all } p \in \mathcal{P}, \tag{5.53}$$

$$\Delta_p \geq R_k x'_{kp} \qquad \text{for all } p \in \mathcal{P},\ k \in \mathcal{K}, \tag{5.54}$$

The entire problem becomes:

$$\text{Minimize} \qquad \rho - \eta \sum_{k \in \mathcal{K}} r_k \Big(\sum_{p \in \mathcal{P} - OUT(O(k))} x_{kp} + \sum_{p \in \mathcal{P}} f_p(x_{kp} - x'_{kp}) \Big) \quad (5.55)$$

subject to

$$(5.17\text{–}5.34),\ (5.51),\ (5.36),\ (5.52),\ (5.38\text{–}5.43),\ (5.49),\ (5.53)$$

$$x'_{kp} \geq 0 \ \text{ and integer} \qquad \text{for all } p \in \mathcal{P} \text{ and } k \in \mathcal{K}. \quad (5.56)$$

In this problem, the objective function (5.55) is obtained by composing the objective function (5.16) and expression (5.50) for the expected traffic loss.

Link-Dependent Backup Paths

In order to formulate the problem for the second strategy, i.e. link dependent rerouting, it has to be assumed that lower amount of the capacity transferred to the backup VP is the same for the failure of any link. Otherwise the variables Z_{pr} would depend on failed link e and the problem would become intractable.

The problem formulation can be obtained by replacing constraints (5.51) with

$$\sum_{r \in \mathcal{P}_{ij}} U_{re} \leq \sum_{p \in \mathcal{P}_{ij}} \delta_{pe} V_p \quad \text{for all } \mathcal{P}_{ij}, \text{ all } e \in \mathcal{A} \qquad (5.57)$$

which is similar to (5.47). Then the problem formulation becomes:

$$(5.55)$$

subject to

$$(5.17\text{–}5.33),\ (5.44\text{–}5.46),\ (5.57),\ (5.52),$$

$$(5.48),\ (5.39\text{–}5.43),\ (5.49),\ (5.53),\ (5.56)$$

5.2.5 Network Dimensioning Subproblem for the Generalized Fault–Tolerant VP Layout Model

In this section we discuss a dimensioning subproblem of the generalized fault-tolerant VP layout model which was stated in the introduction to this chapter. Here we make two essential modifications:

- the set of VP routes between a pair of end nodes is replaced by a set of link-disjoint VP pairs with the same end nodes;
- if the decision is made to set up both VPs from a link-disjoint pair, then under normal conditions call traffic is admitted to both VPs.

The first modification does not change the model. Its advantage is that we are able to omit constraints (5.36) and get rid of zero-one variables associated with VP routes which dramatically speeds up solution process. Although, it comes for the price of potential increase in the number of VP variables.

We argue that the second modification provides a better solution to the VP layout problem than the original approach when a backup VP has zero capacity. Indeed, a set of feasible solutions to this problem includes the "zero backup VP" solution as a special case.

On the other hand, the second modification implies that in case of a VP failure, VP traffic can be switched to the other VP which may be already carrying some non-zero traffic of its own. Thus, the network protocol should be able to increase VP capacity from an arbitrary value. This procedure may require more advanced network signaling than was assumed earlier when the VP capacity had to be increased from a zero value.

It is not practically possible to consider all link-disjoint pairs of VP routes (without resorting to a column generation approach). We propose an approach where a limited number of VP route pairs are selected in advance using special routing heuristics. After that the problem becomes a type of dimensioning problem which allocates capacity to the VPs. The solution obtained is in general approximate but, as will be shown by our computational experiments, its quality does not degrade substantially.

An exact formulation of this model will be presented in the next chapter. The advantage of the simplified model is demonstrated by the following example.

Consider a network instance shown in Figure 5.7. Suppose that links e_1, e_2, e_3 are of unit capacity, there is no fixed charge for the traffic between node pairs (A, A'), (B, B'), (C, C'), and all node pairs have unit mean traffic demand. Let us compare the two strategies:

- If traffic between each node pair is restricted to one VP, then in an optimal solution each link carries one primary VP and has an equal capacity reserved for a backup VP. Hence, capacity assigned to each VP is equal to $1/2$ and the total traffic routed between any node pair is $1/2$.

- If traffic between each node pair is allowed to be spread between both primary and backup VPs, then there is a solution providing better throughput. Indeed, allocate $1/3$ capacity units to each VP and reserve $1/3$ capacity capacity units on each link for backup purposes. Then, it is easy to see that, first, link capacities are not exceeded and, second, there is sufficient amount of capacity to reroute traffic affected by any single link failure. The total traffic between any node in this case is $2/3$ which is higher than $1/2$ in the previous case.

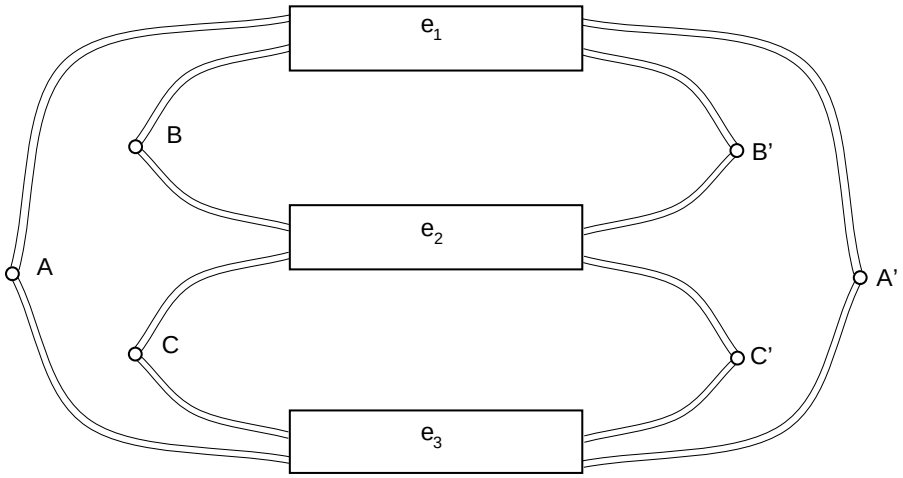

Figure 5.7: Example of VP pairs

Chapter 6

Solution Methods for the VP Layout Problem

In the previous chapter we gave an exact integer programming formulation of the fault tolerant VP layout problem and considered different versions. Theoretically, if this formulation is used as input into a general IP solver, then an exact solution can be obtained. However, because of the complex problem structure, the running time of this solution process would be prohibitively long. In this section, we propose an approach to solving the problem which would provide us with a high quality approximate solution.

First, let us outline the main ideas.

- **Linear Relaxation.** In order to speed up computations, we drop the integrality constraints on variables x_{kp} and y_k.

- **Aggregation.** Similar to the approach presented in Chapter 4, we suggest aggregating y_k and x_{kp} variables by merging the same class calls originating at the same node together. This approach is motivated by the fact that we anticipate a large number of calls between any pair of nodes and we are

more interested in overall demand than in the particular call routing.

- **Heuristics.** We obtain integral zero-one values for z-variables by rounding off their values obtained via linear programming relaxation using various thresholds.

- **Valid Inequalities.** In order to tighten the linear relaxation of the problem, we introduce several groups of valid inequalities. This enables us to improve the quality of our approximate solution.

6.1 Aggregation of the Call Variables

The first step in obtaining an approximate to the solution of the original problem is to aggregate call variables. Earlier, we defined a traffic class as a group of calls having the same peak and mean bit rates. Here we also assume that video and data calls belong to different classes since, as it follows from our earlier discussion, they require different statistical multiplexing models. Similar to the approach used in Chapter 4, call variables are aggregated within the groups corresponding to common originating node and to the same call class. Let $h(k)$ denote a class call k belongs to. Then the new aggregated variables are defined as follows:

$$\tilde{x}^h_{ip} = \sum_{k:\, O(k)=i,\ h(k)=h} r_k x_{kp} \tag{6.1}$$

$$\tilde{y}^h_{ij} = \sum_{k:\, O(k)=i, D(k)=j,\ h(k)=h} r_k y_k \tag{6.2}$$

Next, we define another set of zero-one variables which would indicate admission or rejection of a call class to a VP.

$$z_{ph} = \begin{cases} 1 & \text{if call class } l \text{ is admitted to VP } p \\ 0 & \text{otherwise;} \end{cases}$$

Obviously,

$$\sum_{i\in\mathcal{N}} \tilde{x}_{ip}^h \le \min\{\tilde{M}_i^h, B(p)\} z_{ph} \quad \text{for all} \quad p\in\mathcal{P},\ h\in\mathcal{H} \tag{6.3}$$

where $\tilde{M}_i^h = \sum_{k:\ O(k)=i,\ h(k)=h} r_k$ is the maximal amount of class h traffic which can originate at node i, $B(p) = \min_{e\in p} B_e$ is minimal capacity of a link traversed by VP p.

Let $\tilde{R}_h$ and $\tilde{r}_h$ denote peak and mean bit rate, respectively, of calls within class h. Also, let $\mathcal{H}^{video}$ and $\mathcal{H}^{data}$ be subsets of $\mathcal{H}$ consisting of video and data call classes, respectively. As was mentioned in the discussion of statistical multiplexing mechanism in section 5.1, data traffic is approximated by a two-piece linear function. The first part corresponds to what is called the "non-saturated" case and it expresses direct proportionality dependence on the number of calls. The second part implies a fixed charge cost. Similar to model (**BAS**), we introduce zero-one variable s_p indicating whether the data traffic on VP p is saturated. After we replace $\sum x_{kp}$ in (5.18) by $\sum \frac{\tilde{x}_{ip}^h}{\tilde{r}_h}$ and make appropriate adjustments, we have:

$$\begin{aligned}
(1-s_p)\ LB\{\sum_{i\in\mathcal{H}^{data}}\sum_{i\in\mathcal{N}}(\lfloor -\ln\ p_h\rfloor \frac{\tilde{x}_{ip}^h}{\tilde{r}_h} - \lceil -\ln\varepsilon\rceil)\} \le \\
\sum_{i\in\mathcal{H}^{data}}\sum_{i\in\mathcal{N}}(\lfloor -\ln\ p_h\rfloor \frac{\tilde{x}_{ip}^h}{\tilde{r}_h} - \lceil -\ln\varepsilon\rceil) \le \\
s_p\ UB\{\sum_{i\in\mathcal{H}^{data}}\sum_{i\in\mathcal{N}}(\lfloor -\ln\ p_h\rfloor \frac{\tilde{x}_{ip}^h}{\tilde{r}_h} - \lceil -\ln\varepsilon\rceil)\} \quad \text{for all } p\in\mathcal{P},
\end{aligned} \tag{6.4}$$

where $LB\{\ \}$ and $UB\{\ \}$ are lower and upper bounds, respectively.

If we also use two continuous variables $\tilde{\zeta}_{ph}$ and $\tilde{\xi}_{ph}$ for each data class h and VP p to describe the slopes of the two-piece linear function, then the expression

for the data traffic becomes:

$$0 \leq \tilde{\zeta}_{ph} \leq (1 - s_p)t_h \tag{6.5}$$

$$0 \leq \tilde{\xi}_{ph} \leq s_p T_{ph} \tag{6.6}$$

$$\sum_{i \in \mathcal{N}} \frac{\tilde{x}^h_{ip}}{\tilde{r}_h} = \tilde{\zeta}_{ph} + \tilde{\xi}_{ph} \tag{6.7}$$

$$V_p^{data} = \sum_{h \in \mathcal{H}^{data}} (\tilde{R}_h \tilde{\zeta}_{ph} + \tilde{r}_h \tilde{\xi}_{ph}) \tag{6.8}$$

Here, $t_h = \tilde{r}_h \lceil \ln \varepsilon / \ln p_h \rceil$ is the saturation threshold and T_{ph} is the highest class h traffic level on link p. Notice that only one of the variables $\tilde{\zeta}_{ph}$ and $\tilde{\xi}_{ph}$ is non-zero at a time.

The following two propositions show that the inequalities chosen above are quite "tight" meaning that LP relaxation extreme points are expected to be close to the IP solutions.

Proposition 3 *If variable x has negative lower and positive upper bounds: $LB(x) < 0 < UB(x)$, then the two-dimensional polyhedron $\{(x, y) : (1 - y)\, LB(x) \leq x \leq y\, UB(x);\ 0 \leq y \leq 1\}$ has integer y-coordinates of its extreme points.*

Proof:

As can be seen from Figure 6.1(a), the polyhedron has the following extreme points: $(LB(x), 0)$, $(0, 1)$, $(UB(x), 1)$, $(0, 0)$, all of which have integer y-coordinates. □

Corollary 1 *Constraints (6.4) together with $0 \leq s_p \leq 1$ define feasible sets with zero-one s_p coordinates of their extreme points.*

Proposition 4 *The three-dimensional polyhedron defined by the constraints (6.5–6.6) and $0 \leq s_p \leq 1$ in the space $(\tilde{\zeta}_{ph}, \tilde{\xi}_{ph}, s_p)$ has integer s_p-coordinates of its extreme points.*

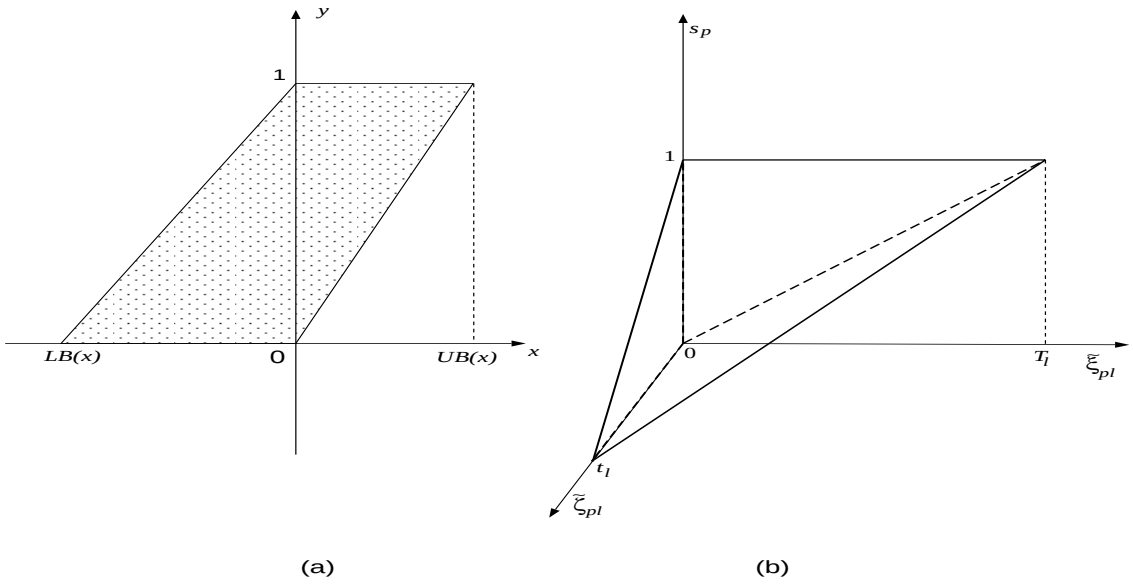

Figure 6.1: Polyhedra

Proof:

As can be seen from Figure 6.1(b), the polyhedron is a tetrahedron with extreme points: $(0,0,0)$, $(t_h,0,0)$, $(0,T_h,1)$, $(0,0,1)$, all of which have integer s_p-coordinates. □

Next, we introduce an expression for the fixed charge Δ_p corresponding to VP p:

$$\Delta_p \geq \begin{cases} \tilde{R}_h z_{ph} & \text{if } h \in \mathcal{H}^{video} \\ \tilde{R}_h(z_{ph} - (1 - s_p)) & \text{if } h \in \mathcal{H}^{data} \end{cases} \qquad \text{for all } h \in \mathcal{H},\ p \in \mathcal{P} \quad (6.9)$$

$$\Delta_p \geq 0 \qquad \text{for all } p \in \mathcal{P} \quad (6.10)$$

Suppose now the traffic classes are ordered such that,

$$0 = \tilde{R}_0 \leq \tilde{R}_1 \leq \tilde{R}_2 \leq \ldots \leq \tilde{R}_{|\mathcal{H}|} \leq 1$$

That is, we arrange all (video and data) traffic classes in increasing order of the fixed charge term in their linear approximation. Class 0 with no fixed charge is

used for CBR traffic. Impose the following "order" constraints on z-variables:

$$1 \geq z_{p1} \geq z_{p2} \geq \ldots \geq z_{p|\mathcal{H}|} \geq 0 \quad \text{for all } p \in \mathcal{P} \tag{6.11}$$

Conceptually, this means that if a call class is admitted to VP, then all call classes with a lower fixed charge are admitted as well.

Now, consider an expression for the slope of the linear approximation for video traffic function:

$$V_p^{video} = \sum_{h \in \mathcal{H}^{video}} \sum_{i \in \mathcal{N}} \tilde{x}_{ip}^h \tag{6.12}$$

Then the entire aggregated approximation of the problem becomes:

AGGR:

$$\text{Maximize} \qquad \tilde{\rho} - \eta \sum_{i \in \mathcal{N}} \sum_{p \in \mathcal{P} - OUT(i)} \sum_{h \in \mathcal{H}} \tilde{x}_{ip}^h \tag{6.13}$$

subject to

$$\tilde{y}_{ij}^h \geq \tilde{\rho} \sum_{k \in \mathcal{K}_{ij}^h} r_k \qquad \text{for all } i \neq j \in \mathcal{N},\ h \in \mathcal{H} \tag{6.14}$$

$$\sum_{p \in OUT(j)} \tilde{x}_{ip}^h - \sum_{p \in IN(j)} \tilde{x}_{ip}^h = \begin{cases} -\tilde{y}_{ij}^h & \text{if } j \neq i \\ \sum_{m \in \mathcal{N} - \{i\}} \tilde{y}_{im}^h & \text{if } j = i \end{cases} \qquad \forall\ i, j \in \mathcal{N},\ h \in \mathcal{H} \tag{6.15}$$

$$(6.3 - 6.12)$$

$$\sum_{p \in \mathcal{P}} (V_p^{video} + V_p^{data} + \Delta_p) \delta_{pe} \leq B_e \qquad \text{for all } e \in \mathcal{A}, \tag{6.16}$$

$$z_{ph} \in \{0,\ 1\} \qquad \text{for all } p \in \mathcal{P},\ h \in \mathcal{H} \tag{6.17}$$

$$s_p \in \{0,\ 1\} \qquad \text{for all } p \in \mathcal{P} \tag{6.18}$$

$$\tilde{x}_{ip}^h,\ \tilde{y}_{ij}^h,\ \tilde{\rho} \geq 0 \qquad \text{for all } i, j \in \mathcal{N},\ p \in \mathcal{P},\ h \in \mathcal{H} \tag{6.19}$$

In this formulation, the objective function (6.13) and constraints (6.14), (6.15) are obtained from the corresponding objective function (5.16) and constraints (5.17), (5.29) by substituting the expressions for x_{kp}- and y_k-variables

from (6.1) and (6.2), respectively. Notice, that since constraint (5.30) from the basic MIP formulation is omitted in the aggregated formulation, the value of $\tilde{\rho}$ may exceed one. If this is the case, then the network is capable of carrying an amount of traffic increased by the factor of $\tilde{\rho}$. Thus $\tilde{\rho}$ can be interpreted as the measure of network over-dimensioning.

Notice that if any constraint from the group (6.14) is satisfied as a strict inequality, then the corresponding variable y_{ij}^h can be decreased which would cause the decrease of flow between the nodes i, j and, as a consequence, non-decrease of the objective function (6.13). This observation can be formulated as the following:

Proposition 5 *There is always an optimal solution to (***AGGR***) in which constraints (6.14) are satisfied as equations.*

Consequently, in order to simplify the formulation of the problem, we can eliminate y-variables by a substitution:

$$\tilde{y}_{ij}^h := d_{ij}^h \tilde{\rho} \quad \text{for all } i \neq j \in \mathcal{N},\ h \in \mathcal{H}$$

where $d_{ij}^h = \sum_{k \in \mathcal{K}_{ij}^h} r_k$ is the class h demand between nodes i and j.

After the substitution the problem is reformulated equivalently as:

AGGR:

Maximize (6.13)

subject to

$$\sum_{p \in OUT(j)} \tilde{x}_{ip}^h - \sum_{p \in IN(j)} \tilde{x}_{ip}^h = \begin{cases} -d_{ij}^h \tilde{\rho} & \text{if } j \neq i \\ \sum_{m \in \mathcal{N} - \{i\}} d_{im}^h \tilde{\rho} & \text{if } j = i \end{cases} \quad \forall\ i, j \in \mathcal{N},\ h \in \mathcal{H} \qquad (6.20)$$

$$(6.3 - 6.12) \text{ and } (6.16 - 6.18)$$

$$\tilde{x}_{ip}^h,\ \tilde{\rho} \geq 0 \qquad \text{for all } i \in \mathcal{N},\ p \in \mathcal{P},\ h \in \mathcal{H} \qquad (6.21)$$

6.1.1 Computational Model for the Fault-Tolerant VP Rerouting

In this section we continue our work on defining a fault-tolerant model which was started in section 5.2.5.

One more comment is to be made before we add fault-tolerant constraints to the problem formulation (**AGGR**). According to our linear approximation for the capacity allocation rule, when multiclass traffic is switched from one VP to another, the fixed charge is equal to the highest fixed charge of all classes. In other words, it is the maximal fixed charge of the two VPs.

We first introduce necessary notation:

$r(p)$, backup for the VP p. (Since each VP in a pair is backup for the other, $r(r(p)) = p$)

Q_e , amount of the capacity reserved on link e to be used for traffic restoration in case of any link failure;

Z_p , increase in fixed charge on the backup VP $r(p)$ when the VP p fails.

It is easy to see that

$$Z_p = \max\{\sum_{h \in \mathcal{H}} \Delta R_h \, (z_p^h - z_{r(p)}^h), 0\} \tag{6.22}$$

where we define $\Delta R_h = \tilde{R}_h - \tilde{R}_{h-1}$ for all $h \in \mathcal{H} - \{0\}$ and $\Delta R_0 = 0$.

Finally, the aggregated model with fault tolerant constraints is formulated as follows:

AGGR-FT:

Maximize (6.13)

subject to

(6.20)

$$\sum_{p\in\mathcal{P}:e\in p}\Big(\sum_{i\in\mathcal{N}}\sum_{h\in\mathcal{H}} x_{ip}^{h}+\sum_{h\in\mathcal{H}}\Delta R_h z_p^h\Big)+Q_e\le B_e \qquad \text{for all } e\in\mathcal{A} \tag{6.23}$$

$$Q_e \ge \sum_{p:e'\in p,\ e\in r(p)}\ \sum_{i\in\mathcal{N}}\sum_{h\in\mathcal{H}} x_{ip}^{h}+Z_p \qquad \text{for all } e\ne e'\in\mathcal{A} \tag{6.24}$$

$$Z_p \ge \sum_{h\in\mathcal{H}}\Delta R_h\,(z_p^h - z_{r(p)}^h) \qquad \text{for all } p\in\mathcal{P} \tag{6.25}$$

$$Z_p \ge 0 \qquad \text{for all } p\in\mathcal{P} \tag{6.26}$$

(6.11)

$(6.17 - 6.21)$

In this formulation, we reserve amount Q_e of link e capacity (constraint (6.23)) sufficient to restore failed VPs whose backups traverse link e (constraint (6.24)). The fixed charge increment Z_p for VP p backup is determined by constraints (6.25–6.26) which are equivalent to expression (6.22).

6.2 Valid Inequalities for the Fixed-Charge Linear Approximation

In this section we present a polyhedral approach to solving the VP layout problem in which the capacity allocation rule for all types of traffic is assumed to be approximated by a fixed-charge model. As was pointed out earlier, this is a case

when all traffic is video or saturated data. Then the constraint set is as follows:

(6.20)

$$\sum_{p\in\mathcal{P}:e\in p}(\sum_{i\in\mathcal{N}}\sum_{h\in\mathcal{H}}x_{ip}^{h}+\sum_{h\in\mathcal{H}}\Delta R_h z_p^h)\leq B_e \quad \text{for all } e\in\mathcal{A} \tag{6.27}$$

(6.11)

$$(6.17-6.21)$$

6.2.1 Band Inequalities

Our approach to formulating valid inequalities in this section follows methodology proposed in [42, 11].

We start by assuming that lower bound on ρ is known a priori. That is

$$\rho \geq \rho_0$$

This constraint can be either explicitly present in the problem formulation or a lower bound might be known in advance.

Let us introduce some notation:

$\vec{\mu} = (\mu_e \geq 0,\ e\in\mathcal{E})$, vector of arbitrarily assigned link lengths;

$\pi_{ij}^{\vec{\mu}}$, shortest path between nodes i and j with respect to link lengths $\vec{\mu}$;

$\mathcal{P}^{\vec{\mu}} = \{p:\ \sum_{e\in p}\mu_e > 0\}$, set of positive length VPs with respect to link lengths $\vec{\mu}$.

In addition, let $\bar{B}_e$ be a link e capacity and $\bar{D}_{ij}$ be a flow demand between nodes i and j. Then a multicommodity flow satisfying the demand can be routed through the network if and only if

$$\sum_{e\in\mathcal{E}}\mu_e\bar{B}_e \geq \sum_{i,j\in\mathcal{N}}\pi_{ij}^{\vec{\mu}}\bar{D}_{ij} \quad \text{for all } \vec{\mu}\geq 0 \tag{6.28}$$

This characterization of the capacity/demand for a network carrying multicommodity flow was first stated in [19, 34] and can be proved using linear programming duality.

Notice that in the trivial case when $\mu_e = 1$ for links e forming a (multi-)cut and zero otherwise, condition (6.28) becomes equivalent to minimum (multi-)cut necessary conditions on multicommodity flow feasibility.

In our case after substituting

$$\begin{aligned} \bar{B}_e &= B_e - \sum_{p:\, e \in p} \sum_{h \in \mathcal{H}} \Delta R_h z_p^h \\ \bar{D}_{ij} &= \sum_{h \in \mathcal{H}} \rho_0 d_{ij}^h \end{aligned}$$

into (6.28) and regrouping the parts, we obtain

$$\sum_{p \in \mathcal{P}^{\vec{\mu}}} \sum_{h \in \mathcal{H}} (\sum_{e \in p} \mu_e) \Delta R_h z_p^h \leq \sum_{e \in \mathcal{E}} B_e - \sum_{i,j \in \mathcal{N}} \pi_{ij}^{\vec{\mu}} (\sum_{h \in \mathcal{H}} \rho_0 d_{ij}^h) \quad \text{for all } \vec{\mu} \geq 0 \quad (6.29)$$

In Figure 6.2 we picture our approach to deriving a valid inequality based on (6.29) for a fixed $\vec{\mu}$. Each column of the grid corresponds to a VP, $p \in \mathcal{P}^{\vec{\mu}}$, and

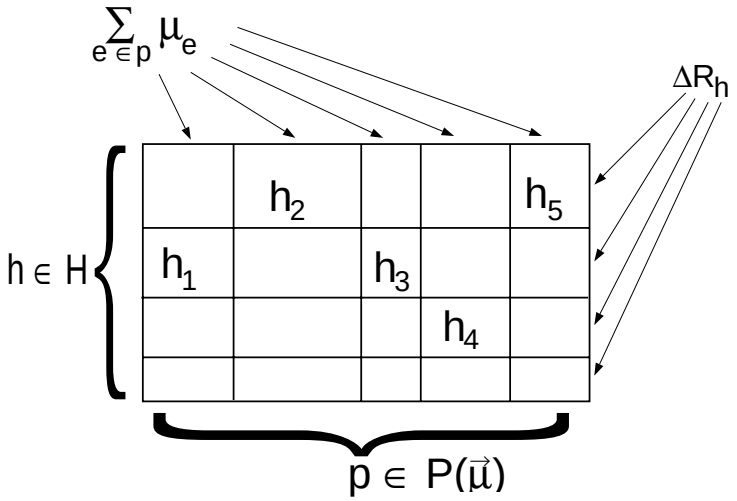

Figure 6.2: Band Inequality

is $\sum_{e \in p} \mu_e$ wide; each row corresponds to traffic class $h \in \mathcal{H}$ and is ΔR_h high. The shaded area is equal to $\sum_{e \in p} \mu_e R_h z_p^{h_p}$ and chosen to exceed the right-hand side of (6.29) (denoted further as RHS($\vec{\mu}$)). This means that in order to satisfy flow feasibility conditions (6.28), at least one of the variables $z_p^{h_p}$ has to be zero. We call this valid inequality "band inequality" following [11] and formulate it as a proposition:

Proposition 6 *If for any $p \in \mathcal{P}^{\vec{\mu}}$ h_p is chosen such that*

$$\sum_{p \in \mathcal{P}^{\vec{\mu}}} \sum_{h \in \mathcal{H}} (\sum_{e \in p} \mu_e) R_{h_p} > \sum_{e \in \mathcal{E}} B_e - \sum_{i,j \in \mathcal{N}} \pi_{ij}^{\vec{\mu}} (\sum_{h \in \mathcal{H}} \rho_0 d_{ij}^h)$$

then

$$\sum_{p \in \mathcal{P}^{\vec{\mu}}} z_p^{h_p} \leq |\mathcal{P}^{\vec{\mu}}| - 1 \tag{6.30}$$

*is a valid inequality for (***AGGR***).*

Separation Problem

We formulate a separation problem for generating violated constraints (6.30) as a multi-constraint knapsack problem finding the most violated constraint if there is any. Though the knapsack problem is NP-hard, it allows pseudo-polynomial solution algorithms. Also, the rounding heuristics based on LP-relaxations are normally quite effective.

Variables:

$$t_p^h = \begin{cases} 1 & \text{if } h = h_p \\ 0, & \text{otherwise} \end{cases} \quad \text{for all } p \in \mathcal{P}^{\vec{\mu}} \text{ and } h \in \mathcal{H}$$

SP:

$$\text{Maximize} \qquad \sum_{p\in\mathcal{P}^{\vec{\mu}}} \sum_{h\in\mathcal{H}} z_p^h t_p^h \tag{6.31}$$

Subject to

$$\sum_{h\in\mathcal{H}} t_p^h = 1 \qquad \text{for all } p \in \mathcal{P}^{\vec{\mu}} \tag{6.32}$$

$$\sum_{p\in\mathcal{P}} (\sum_{e\in p} \mu_e) R_h t_p^h \geq \text{RHS}^{\vec{\mu}} \tag{6.33}$$

$$t_p^h \in \{0,1\} \qquad \text{for all } p \in \mathcal{P}^{\vec{\mu}} \text{ and } h \in \mathcal{H} \tag{6.34}$$

If the problem (**SP**) is not feasible or the value of the optimal solution does not exceed $|\mathcal{P}^{\vec{\mu}}| - 1$ (right-hand side of (6.30)), then all valid inequalities for the given values of ρ_0 and $\vec{\mu}$ are satisfied; if the optimal solution value exceeds $|\mathcal{P}^{\vec{\mu}}|-1$, a new valid inequality is added to the formulation. The problem solution method can be easily modified to allow for generating more than one violated constraint.

Choice of $\vec{\mu}$

As was mentioned earlier, any cut induces a vector $\vec{\mu}$ such that

$$\mu_e = \begin{cases} 1 & \text{if arc } e \text{ belongs to the cut} \\ 0 & \text{otherwise} \end{cases} \tag{6.35}$$

In our computational experiments, we consider a set of cuts $\mathcal{C} = \{(\mathcal{N} - \{i\}, \{i\})|\ i \in \mathcal{N}\}$. For each cut in $\mathcal{C}$, vector $\vec{\mu}$ is defined by (6.35) and used to solve the separation problem (**SP**) which generates violated band inequalities in the form (6.30).

6.2.2 Connectivity Inequalities

Consider a cut $\mathcal{C} = (\mathcal{V}, \mathcal{W} = \mathcal{N} - \mathcal{V})$ in the network $\mathcal{G} = (\mathcal{N}, \mathcal{P})$ where $\mathcal{V} \subset \mathcal{N}$. Suppose the total demand between the node subsets defining the cut is non-zero. That is, $\exists h \in \mathcal{H} : \ D^h_{\mathcal{V}\mathcal{W}} = \sum_{ij:\, i\in\mathcal{V}, j\in\mathcal{W}} d^h_{ij} > 0$. Let h_0 be the highest among such h. (Here, we omit indices ij at h_0). Then, in order to provide non-zero network throughput, at least one arc from the cut arc set should be open for class h_0 traffic. If we denote this arc cut set by $\mathcal{P}_{\mathcal{V}\mathcal{W}} = \bigcup_{i\in\mathcal{V}, j\in\mathcal{W}} \mathcal{P}_{ij}$, then the above statement can be formulated as the following proposition.

Proposition 7 *If for a cut $(\mathcal{V}, \mathcal{W})$ there exists h_0 such that*

$$h_0 = \max\{h : \ D^h_{\mathcal{V}\mathcal{W}} > 0\}, \tag{6.36}$$

then

$$\sum_{p\in\mathcal{P}_{\mathcal{V}\mathcal{W}}} z_p^{h_0} \geq 1 \tag{6.37}$$

*is a valid inequality for (***AGGR***).*

Separation Problem

Clearly, there are in general exponentially many inequalities (6.37) for a given problem instance. However, once the linear programming relaxation of (**AGGR**) is solved, the violated connectivity constraints (6.37) can be found in polynomial time using the following approach.

For every node pair (i, j) such that there exists $h_0 = \max\{h : \ d^h_{ij} > 0\}$, solve a single commodity flow problem sending a unit amount of flow from i to j over arc set $\mathcal{P}$ with capacities $z_p^{h_0}$. If the flow problem is not feasible, add valid inequalities (6.37) for the min cut $(\mathcal{V}\mathcal{W})$ to the constraint set of problem (**AGGR**).

6.2.3 Capacity Inequalities

Capacity inequalities can be considered as strengthened connectivity inequalities introduced in Section 6.2.2 in a special case of equal capacity links. This type of valid inequality was considered in the literature [7, 31] as applied to a network design problem.

Similar to the way it was introduced in section 6.2.1, we assume that a lower bound on variable ρ, denoted by ρ_0, is known.

Consider a cut $(\mathcal{V}, \mathcal{W})$. If all physical links associated with the cut are of the same capacity B and a fixed charge associated with class h is R_h, then in order to accommodate demand $\rho_0 D^h_{\mathcal{V}\mathcal{W}}$ across the cut, at least $\lceil \rho_0 D^h_{\mathcal{V}\mathcal{W}}/(B - R_h) \rceil$ links/VPs should be open. This can be formulated as the following proposition.

Proposition 8 *If all links associated with cut* $(\mathcal{V}, \mathcal{W})$ *have the same capacity* B,

$$\sum_{p \in \mathcal{P}_{\mathcal{V}\mathcal{W}}} z^h_p \geq \lceil \frac{\rho_0 D^h_{\mathcal{V}\mathcal{W}}}{B - R_h} \rceil \tag{6.38}$$

*is a valid inequality for (***AGGR***).*

Separation Problem

Using approach outlined in [31], we propose the following heuristic to detect some of the violated capacity inequalities (6.38).

STEP 1. Test valid inequality (6.38) for all $h \in \mathcal{H}$ and all $\mathcal{V} = \mathcal{V}_i = \{i\}$, $i \in \mathcal{N}$. If a violated inequality is detected, add it to the problem and stop, otherwise goto STEP 2.

STEP 2. Grow node sets $\mathcal{V}_i$:

$$\mathcal{V}_i := \mathcal{V}_i \bigcup \{j^* = \arg \min_{j \in \mathcal{N}, h \in \mathcal{H}} \frac{\rho_0 D^h_{\mathcal{V}_i \mathcal{W}_i}}{B - R_h}\}$$

where $\mathcal{W}_i = \mathcal{N} - \mathcal{V}_i$. Goto STEP 3.

STEP 3. If there are any violated inequalities found in STEP 2, add them to the problem and stop, otherwise repeat STEP 2 while $|\mathcal{V}_i| < |\mathcal{N}|$.

The idea here is to apply a kind of "greedy" approach in order to find as many violated inequalities as possible.

6.2.4 Bundle Inequalities

As it was discussed earlier, the problem under the scope is essentially a bi-criteria optimization problem. In the marginal case, we wish to maximize the throughput when the flow through intermediate nodes is equal to zero. That means that at least one direct end-to-end VP is set up between each pair of nodes with non-zero traffic demand.

Proposition 9 *If flow through intermediate nodes is constrained to be zero, that is,*

$$\sum_{i \in \mathcal{N}} \sum_{p \in \mathcal{P},\, p \notin OUT(i)} \sum_{h \in \mathcal{H}} \tilde{x}_{ip}^{h} = 0$$

and there exists $h_0 = \max\{h : \ d_{ij}^{h} > 0\}$

then

$$\sum_{p \in \mathcal{P}_{ij}} z_p^{h_0} \geq 1 \tag{6.39}$$

Notice that in case of a single VP candidate between each node pair, constraint (6.39) uniquely defines the solution.

6.3 Algorithms and Heuristics

Linear programming relaxation of the problem **(AGGR-FT)** together with various valid inequalities presented in sections 6.1–6.2 allows us to obtain an upper bound on the solution value. However, it does not provide us with a feasible solution. Also, the number of VP candidate paths between node pair (i, j) contained in a VP candidate set $\mathcal{P}_{ij}$ can be exponentially large. In this section we discuss various aspects of obtaining feasible solutions using a limited number of VP candidate routes and outline a general procedure for solving the problem.

6.3.1 Feasible Solution Heuristic

Problem formulation **(AGGR-FT)** contains zero-one z-variables. Since the problem size is quite large, it does not appear practically possible to find an exact optimal solution. Instead, we propose a heuristic method to obtain a feasible solution from a solution to the linear programming (LP) relaxation with little additional computations.

Consider solution values $\hat{z}_p^h$ of the LP relaxation variables z_p^h. Suppose, we are given a threshold T such that $0 < T < 1$. If variables z_p^h are fixed to their rounded values $\bar{z}_p^h$:

$$\bar{z}_p^h = \begin{cases} 0 & \text{if } \hat{z}_p^h < T \\ 1 & \text{otherwise} \end{cases} \tag{6.40}$$

then it is easy to see that the rounded values satisfy order constraints (6.11). Now, if problem **(AGGR-FT)** is solved again as an LP with its z-variables fixed to the $\bar{z}_p^h$ values and has a feasible solution, this new solution will be IP-feasible.

The remaining question is how to choose a rounding threshold T. If the

chosen threshold is too large, too few z-variables will be fixed to one and the network may become disconnected. On the other hand, if the chosen threshold is too small, too many z-variables will be fixed to one and the throughput will be affected. Our computational experiments showed that the solution value is not very sensitive to the particular choice of the threshold. In order to find the best solution by rounding, we try several thresholds starting with initial threshold value T_1 and sequentially decreasing the threshold value by the constant factor.

6.3.2 Selection of Initial VP Route Set

As was stated in Proposition 1, the optimal solution to VP layout problem with objective function weight coefficient $\eta = 0$ (pure throughput maximization) and without fault-tolerant constraints, does not involve VPs longer than one hop. Thus, the optimal solution can be obtained without creating VP candidate route sets $\mathcal{P}_{ij}$. We use this property to obtain an initial solution which is later used to generate an initial set of VP routes. Note that for every VP we must find a link-disjoint backup VP.

The heuristic can be outlined as follows. Consider a solution to the VP layout problem with pure throughput maximization and with the set $\mathcal{P}_{ij}$ defined as follows:

$$\mathcal{P}_{ij} = \begin{cases} (i,j) & \text{if } (i,j) \in \mathcal{A} \\ \emptyset & \text{otherwise} \end{cases} \tag{6.41}$$

This solution provides flow values from any node i obtained by summing the variables x_{ie}^h over all traffic classes. For any node pair (i,j) consider a highest capacity flow path p_{ij}. This path is defined as a path maximizing the lowest flow value on its arcs and can be found by a simple modification of the shortest

path algorithm. In our implementation we use the Bellman-Ford algorithm since it guarantees that the resulting path will have the lowest number of hops. The second link-disjoint backup path is obtained by deleting the links on the first path and reapplying the method.

6.3.3 Outline of the Solution Process

In this section we give a general outline of our algorithm by providing an approximate solution to **(AGGR-FT)**.

1. Solve problem **(AGGR)** with $\eta = 0$
2. Use VP routing heuristic to obtain a set of link-disjoint path pairs
3. **for** $\eta = \eta_0, \eta_1, \ldots, \eta_N$ **do**
4. Solve an LP relaxation of **(AGGR-FT)** with valid inequalities
5. **for** $T = T_1, T_2, \ldots, T_M$ **do**
6. Round z-variables values using threshold T
7. Solve problem **(AGGR-FT)** with z-variables fixed
8. Choose the best solution to **(AGGR-FT)** with fixed variables
9. Choose a subset of non-dominated solutions to **(AGGR-FT)**

In this algorithm, values of the η coefficients were chosen empirically such that $\eta_0 = 0$, η_N was large enough to provide end-to-end VP layout (Proposition 2). As was mentioned earlier, different threshold values $T_1, T_2, \ldots, T_M$ are obtained by sequentially decreasing its initial value T_1 by a constant factor. Our particular choice was the initial value of $1/2$ and the decrement factor of 2. Thus, $T_m = 2^{-m}$. Details of the LP computations will be described in the next section.

6.4 Computational Experiments

In this section we present results of the computational experiments and software implementation of our approach to solving problem **(AGGR-FT)**. Our computational experiments were designed to demonstrate telecommunication as well as combinatorial optimization aspects of the problem. We also present instances of the test problems.

6.4.1 Implementation

The algorithm was implemented on SUN SPARC 10 workstation, coded in C and used CPLEX 3.0 LP solver callable library. Below we give some detail of the implementation.

Since our problem is a generalization of a multicommodity flow problem, we developed an approach which would take advantage of the embedded network structure. An initial basis was obtained by solving a problem with single-commodity network flow constraints (i.e. a subproblem of the original problem obtained by omitting coupling constraints) by using CPLEX's built-in network optimizer. The resulting basis was not feasible but nevertheless provided a good starting point for solving the LP relaxation. At this stage we used the primal simplex-method providing, as was empirically shown, a faster way to solve the problem than dual simplex. The resulting optimal solution was used to obtain rounded z-variable values and its optimal basis was used as initial dual-feasible basis to solve the problem with fixed z-variables. The latter problem was solved several times for various rounding thresholds. Since its initial basis was dual-feasible we applied the dual simplex-method. To initiate a solution process for

a linear programming relaxation with a new η objective function coefficient, the optimal basis from the previous iteration was used. Since this basis was primal-feasible, the primal simplex was used.

6.4.2 Test Data

Our computational experiments were conducted for two network configurations presented in Figures 6.3 and 6.4. The first is an example of a grid network with 8 nodes and 10 undirected links, the other represents a geographically distributed network with 12 nodes and 20 undirected links. All links in these two networks have the same capacity of 250 Mb/s. Traffic demand between each node pair was generated randomly using the same uniform distribution.

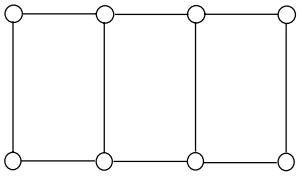

Figure 6.3: Grid Network Configuration

6.4.3 Telecommunication Aspects

We present solutions to the bi-criteria optimization VP layout problem in the form of a curve connecting approximate Pareto-optimal solution points. This curve demonstrates a tradeoff between relative throughput and VP-to-VP switching (intermediate flow). We present these results in Figures 6.5 and 6.6 for the "grid" and "distributed" networks, respectively. The curves are shown by solid

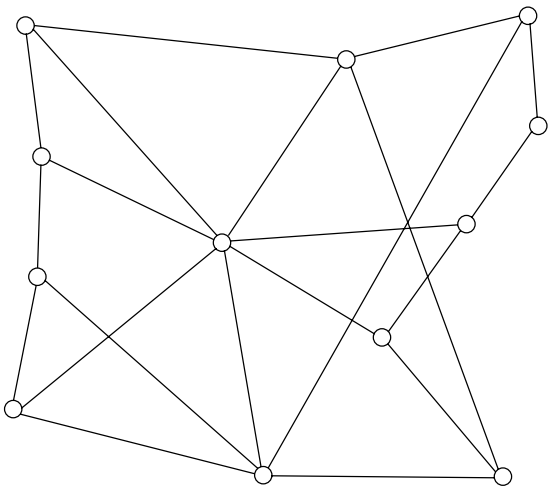

Figure 6.4: Geographically Distributed Network Configuration

lines. As can be seen, the throughput increase can be achieved only at the price of increased flow through intermediate nodes. In the accompanying Tables 6.1, we provide additional statistics for the VP layout such as the number of VPs, average hop length and allocated capacity. In accordance with the theory, the VPs tend to become "slimmer", "longer" and more numerous as we increase the weight coefficient η.

VP Layout Adaptability

As was mentioned in the introduction to Chapter 5, we define adaptability as the ability of a particular telecommunication network (and its VP layout) to function well when the traffic demand varies significantly from the projected traffic (under which its design was based).

Our conjecture was that layout using "fat" and "short" VPs should have better adaptability than a layout using narrower and longer VPs.

"Grid" Network				
η	Number of VPs	Average Number of Hops on VP	Average VP Capacity, Mb/s	Relative Gap, %
0.0000	47	2.15	39.784	2.17
0.0001	85	2.48	16.093	7.84
0.0002	88	2.55	15.941	8.28
0.0004	93	2.59	12.862	10.53
0.0006	97	2.59	11.545	1.86

"Distributed" Network				
η	Number of VPs	Average Number of Hops on VP	Average VP Capacity, Mb/s	Relative Gap, %
0.000	153	1.95	29.608	3.18
0.00005	164	1.97	26.449	4.38
0.001	190	2.16	18.259	7.11
0.002	188	2.14	17.418	8.35
0.003	200	2.18	16.181	9.74
0.004	208	2.21	14.941	1.90

Table 6.1: VP layout Statistics

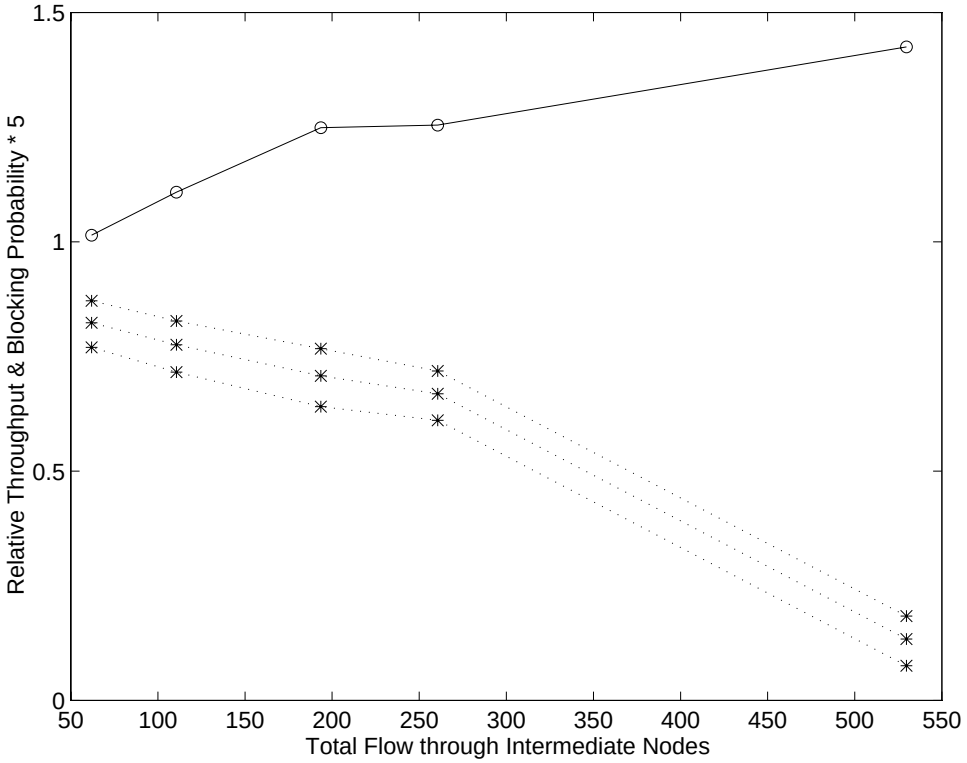

Figure 6.5: Results for "Grid" Network – Tradeoff Curve (solid line) and Bandwidth Blocking Probabilities for Various VP Layouts (dashed line).

We used call blocking probability as a measure of network adaptability. Since in our model different calls have different bandwidth, we are using weighted call blocking probability, or bandwidth blocking probability defined as the ratio of the total rejected bandwidth to the total bandwidth requested.

In order to confirm our conjecture, we conducted a series of computational experiments simulating call requests – setup/blocking – takedown using a precomputed VP layout. We considered several sets of call requests obtained by increasing projected traffic demand in the VP layout by 1, 10 and 30 %. After that the requested connections were routed/rejected using methods described in Chapter 3. The networks used consisted of VPs laid out according to the

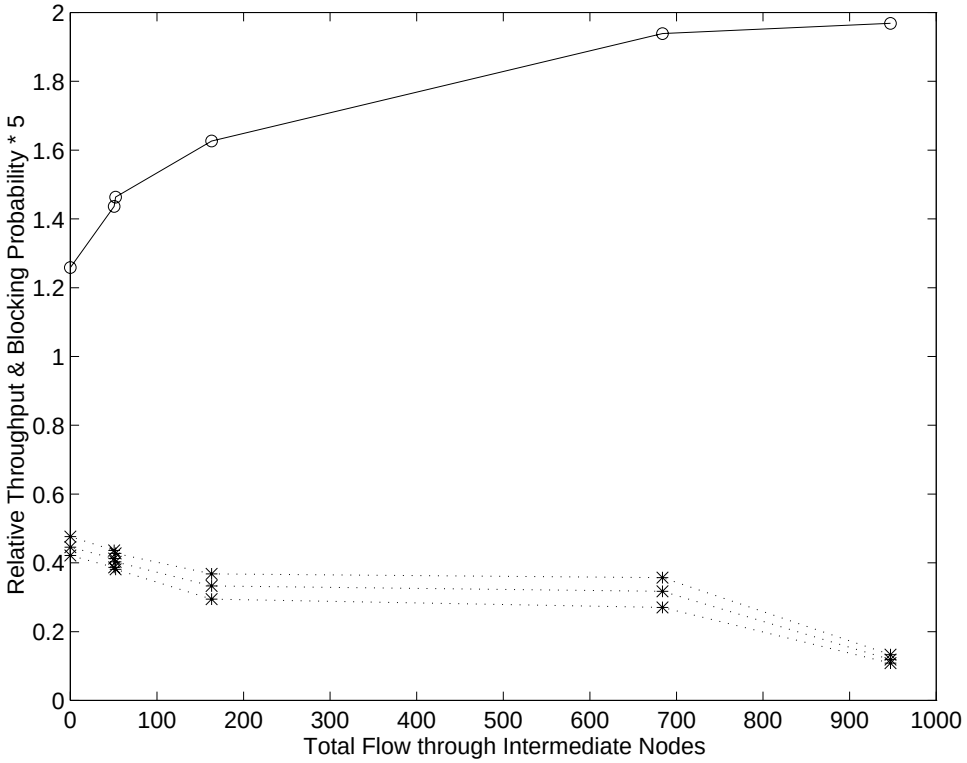

Figure 6.6: Results for "Distributed" Network – Tradeoff Curve (solid line) and Bandwidth Blocking Probabilities for Various VP Layouts (dashed line).

solution of problem (**AGGR**) with varying bi-criteria objective function weight coefficients for the throughput – intermediate flow criteria.

The results are shown in Figures 6.5 and 6.6 for the "grid" and "distributed" networks, respectively. In these figures, dotted lines connect points representing different bandwidth blocking probabilities (multiplied by 5 for better representation purposes) for different VP layouts. The three lines correspond to three traffic demand increments mentioned above.

It can be seen from the figures that "fat" and "short" VPs have better adaptability than "slim" and "long" VPs.

6.4.4 Combinatorial Optimization Aspects

As was mentioned earlier, it practically impossible to find an optimal solution to the problem. Instead, we produce an approximate solution and estimate quality of the approximation.

From the previous discussion of the solution process, it can be concluded that there are two factors which may cause suboptimality of the solution:

1. instead of generating all paths between each node pair within the linear program, we obtained a single pair using a multicommodity flow heuristic;

2. instead of solving IP to optimality, we chose a feasible solution using a certain rounding heuristic.

Since it is difficult to give a quantitative estimation to the loss of accuracy caused by the first factor, we argue that potential loss of accuracy is not substantially high by comparing layouts for one- and two-route per node pair VP candidate sets. The comparison results obtained for the solution to the **(AGGR)** problem are shown in Figures 6.7 and 6.8 for "grid" and "distributed" networks, respectively.

It can be seen that the curve shapes are very similar and adding the second path to a VP candidate set does not substantially improve the solution.

An accuracy loss caused by the second factor which is also known as integrality gap can be estimated by comparing solution values of the LP relaxation and feasible solutions. Quantitative results are provided in Tables 6.1 (last column) which contains the ratio (per cent) of the integrality gap (difference in value of integer and LP solutions) and the value of the linear relaxation solution.

It can be seen from the Tables 6.1 that the integrality gap does not exceed

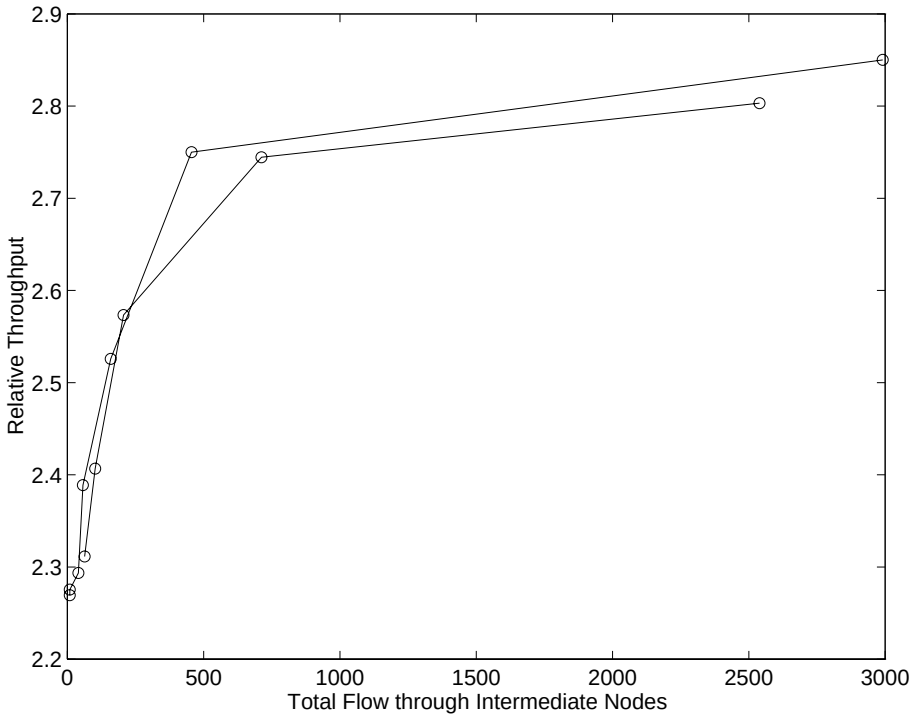

Figure 6.7: Pareto-optimal Solution Sets for One- and Two- VP Candidate Route Approaches ("Grid" Network)

10% and it becomes quite low for the last solution corresponding to end-to-end VP layout where we can apply rather strong "bundle" valid inequality (6.39).

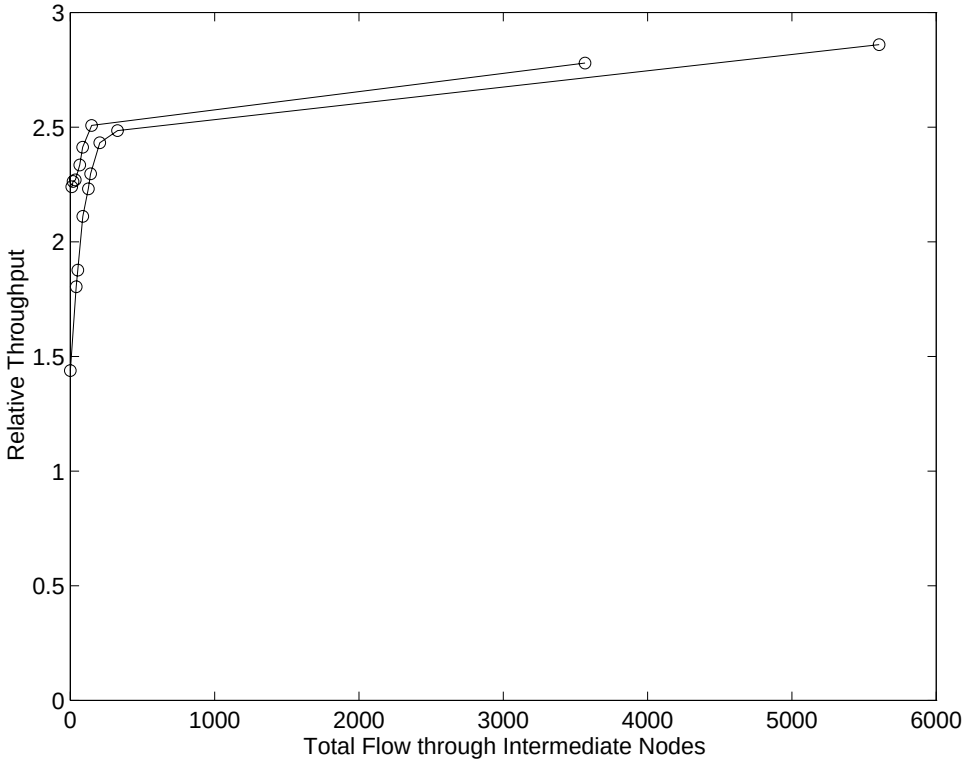

Figure 6.8: Pareto-optimal Solution Sets for One- and Two- VP Candidate Route Approaches ("Distributed" Network)

6.5 Conclusions

In this chapter we have presented models and algorithms which address the fault-tolerant VP layout problem. Using VPs as a backup mechanism enables ATM networks to respond in real time to single-link failures by switching traffic to pre-computed backup routes. The amount of time required to perform such rerouting is negligibly low and determined by the speed with which VPI switching table is changed in the beginning node of each affected VP.

Our problem formulation involves bi-criteria optimization for throughput maximization and control overhead/delay minimization objectives. This ap-

proach results in output consisting of a solution set also called a tradeoff curve which provides a network manager with a useful decision making tool.

In order to carry out the implementation of our methods, specific protocols must be designed enabling VP switching as soon as link failures are detected. Furthermore, refinements to our algorithms should be made to insure efficient interaction with network operation and management.

Chapter 7

Conclusions

We, now, summarize the contributions of the dissertation, and describe potential directions for the future research. Since our work was essentially interdisciplinary, involving telecommunication as well as mathematical programming issues, we consider our contributions to both of these areas.

Modern telecommunication networks are designed to carry multiclass traffic and often have a geographically distributed, complex structure. In this situation it becomes very important to develop reliable mechanisms working in both reactive and proactive modes for responding to network failures without interruption of real-time network services. The primary contribution of our work in the area of fault management of telecommunication networks is the development of tools capable of restoring traffic under a wide variety of conditions and network architectures. In chapters 4 and 5 we considered two approaches to rerouting: reactive and proactive, performed on VC and VP levels, respectively. The first approach was designed to respond to multiple link failures and/or for use in telecommunication networks lacking two layer VC/VP architecture. The other approach which is less universal but more effective was designed to respond

to single-link failures – the most common failure type in ATM networks. It is proactive and explicitly predefines a set of VP switching operations to fix any potential link failure.

Certain multiclass telecommunication networks allow for preemption of lower priority services when higher priority services are affected by a network failure and need to be rerouted. This feature was incorporated in to our model to compute VC-level rerouting. Another feature of modern telecommunication networks – their ability to perform distributed computations – was also used in our model.

Our approach to VP layout can be considered as an advanced approach to resource allocation. The novel features here are enhancement of existing VP backup schemes allowing for non-zero traffic over the backup VP and bi-criteria optimization involving throughput maximization and control overhead/delay minimization objectives. This approach results in output consisting of a solution set also called a tradeoff curve which provides a network manager with a useful decision making tool. It requires a minimal set of on-line operations and because of that can be implemented over broad spectrum of network protocols.

Development of these two approaches can be considered as a first step towards an integrated fault management system. The future research directions would involve refinements of the proposed models and design of the protocols which could take advantage of the suggested fault recovery techniques.

Our contribution in the area of mathematical programming can be highlighted as follows:

- Generalization of the multicommodity flow problem with zero/one variables by including preemption variables (chapter 4).

- Design of new solution methods oriented toward obtaining a high-quality approximate solution in real time.
- Development of a two-phase approach taking advantage of distributed computational resources.
- Piece-wise linear approximation for capacity allocation rule in case of statistical multiplexing.
- Integer programming formulation for VP layout problem.
- Design of solution methods for VP layout problem based on valid inequalities and rounding schemes.

Finally, we list the most relevant future research directions. The first two deal with telecommunication aspects of the VP layout problem; the last two are related to mathematical programming issues.

1. VP layout for point-to-multipoint and multipoint-to-multipoint connections. This type of connection is expected to be quite common since it supports video conferencing and data broadcasting through the network. The challenging issue here is large amount of required bandwidth and high burstiness.
2. VP layout capable of supporting rare high-bandwidth connections. This connection type can add a lot of variance to the traffic pattern and require high adaptability of the VP layout.
3. Dynamic VP route generation. Since our VP route set is limited to only one pair of link-disjoint routes between any node pair, there is a need

to develop techniques capable of dynamically generating additional link-disjoint VP route couples during the solution process. Conceptually, this process should be similar to column generation.

4. Narrowing the integrality gap for the approximate solutions. This is a classical problem faced by everyone designing approach to solving an IP. Since the formulated problem is quite large and has a complex structure, it does not seem promising to apply any sort of branching technique. Instead, this problem should be solved by improving the quality of the approximate solution and tightening the LP relaxation bounds.

Bibliography

[1] H. Ahmadi, P. Chimento, R. Guérin, L. Gün adn B. Lin, R. Onvural, and T. Tedijanto. NBBS traffic management overview. *IBM Systems Journal*, 34(4):604–628, 1995.

[2] R. Ahuja, T. Magnanti, and J. Orlin. *Network Flows: Theory, Algorithms and Applications*. Prentice-Hall, Englewood Cliffs, NJ, 1993.

[3] J. Anderson, B. Doshi, S. Dravida, and P. Harshavardhana. Fast restoration of ATM networks. *IEEE Journal on Selected Areas in Communications*, 12(1):128–138, 1993.

[4] B. Awerbuch, Y. Azar, and S. Plotkin. Throughput competitive on-line routing. In *Proceedings of 34th IEEE Annual Simposium on Foundations of Computer Science*, pages 32–40, November 1993.

[5] S. Bahk and M. Zarki. Dynamic multi-path routing and how it compares with other dynamic routing algorithms for high speed wide area networks. In *SIGCOMM '92*, pages 53–64, 1992.

[6] M. Ball, A. Vakhutinsky, P. Chimento, L. Gün, and T. Tedijanto. Distributed call rerouting in multiclass broadband networks. *Journal of Network and Systems Management*, 3(4):381–404, 1995.

[7] F. Barahona. Network design using cut inequalities. *SIAM Journal on Optimization*, 6(3):823–837, August 1996.

[8] J. Baras. ATM networks. Lecture Course, 1996.

[9] K.-T. Cheng and F. Y.-S. Lin. On the joint virtual path assignment and virtual circuit routing problem in ATM networks. In *IEEE GLOBECOM '94*, pages 777–782, December 1994.

[10] C. Colbourn. *The Combinatorics of Network Reliability*. Oxford University Press, 1987.

[11] G Dahl and M. Stoer. A cutting plane algorithm for multicommodity survivable network design problems. Technical Report 3, Institute of Informatics, University of Oslo, May 1995.

[12] B. Gavish and I. Neuman. Routing in a network with unreliable components. *IEEE Transactions on Communications*, 40(7):94–110, 1992.

[13] R. Gawlick, A. Kamath, S. Plotkin, and K. Ramakrishnan. Routing and admission control in general topology networks. Technical Report STAN-CS-TR-95-1548, Stanford University, 1994.

[14] A. Gersht and A. Shulman. Optimal dynamic virtual path bandwidth allocation and restoration in ATM networks. In *IEEE GLOBECOM '94*, pages 770–776, December 1994.

[15] A. Gersht, A. Shulman, J. Vucetic, and J. Keilson. Dynamic bandwidth allocation, routing, and access control in ATM networks. In *IEEE Workshop*, pages 94–110, 1993.

[16] L. Gün and R. Guérin. A unified approach to bandwidth allocation and access control in fast packet switching networks. In *INFOCOM '92*, pages 1–12, Italy, 1992.

[17] L. Gün and R. Guérin. A framework for bandwidth management and congestion control in high-speed networks. *Computer Networks and ISDN Systems*, 1993.

[18] S. Gupta, K. Ross, and M. Zarki. Routing in virtual path based ATM networks. In *IEEE GLOBECOM '92*, pages 571–575, December 1992.

[19] M. Iri. On an extension of the maximum-flow minimum-cut theorem to multicommodity flows. *Journal of the Operation Research Society of Japan*, 13:129–135, 1971.

[20] R. Kawamura, H.Hadama, and I. Tokizawa. Implementation of self-healing function in ATM networks based on virtual path concept. In *IEEE INFOCOM '95*, April 1995.

[21] R. Kawamura, K. Sato, and I. Tokizawa. Self-healing ATM networks based on virtual path concept. *IEEE Journal on Selected Areas in Communications*, 12(1):120–127, 1994.

[22] R. Kawamura and I. Tokizawa. Self-healing virtual path architecture in ATM networks. *IEEE Communications Magazine*, 33(9):72–79, 1995.

[23] F. Kelly. Routing in circuit switched networks: Optimization, shadow prices and decentralization. *Advances in Applied Probability*, 20:492–500, 1988.

[24] M. Laguna and F. Glover. Bandwidth packing: A tabu search approach. *Management Science*, 39:492–500, 1993.

[25] J.Y. Le Boudec. The asynchronous transfer mode: a tutorial. *Computer Networks and ISDN Systems*, 24:279–309, 1992.

[26] W. Leland, M. Taqqu, W. Willinger, and D. Wilson. On the self-similar nature of ethernet traffic. In *SIGCOMM '93*, pages 183–193, 1993.

[27] W. Leland and D. Wilson. High time resolution measurement and analysis of LAN traffic: Implications for LAN interconnection. In *INFOCOM '91*, pages 1360–1366, 1991.

[28] B. Liang and K. Ross. Loss models for ATM networks with separable statistical multiplexing. Technical report, Department of Systems Engineering, University of Philadelphia, September 1995.

[29] B. Liang and K. Ross. Tutorial: Stochastic network models for asynchronous transfer modes. In *INFORMS National Meeting*, Washington, D.C., May 1996. Institute for Operations Research and Management Science.

[30] F. Y.-S. Lin and K.-T. Cheng. Virtual path assignment and virtual circuit routing in ATM networks. In *IEEE GLOBECOM '93*, pages 436–441, December 1993.

[31] T. Magnanti, P. Mirchandani, and R. Vachani. Modeling and solving the two-facility capacitated network loading problem. *Operations Research*, 43(1):142–157, January-February 1995.

[32] S. Martello and P. Toth. *Knapsack Problems*. John Wiley & Sons, 1990.

[33] K. Murakami and H. Kim. Virtual path routing for survivable ATM networks. *IEEE/ACM Transactions on Networking*, 4(1):22–39, February 1996.

[34] K. Onaga and O. Kakusho. On feasibility conditions of multicommodity flows in networks. *Transactions on Circuit Theory*, 18(4):425–429, 1971.

[35] R.O. Onvural. *Asynchronous Transfer Mode Networks: Performance Issues*. Artech House, Inc., 1994.

[36] O. Palmon, A. Kamath, and S. Plotkin. Routing and admission control in general topology networks with poisson arrivals. In *7th Symposium on Discrete Algorithms*. ACM-SIAM, 1996. to appear.

[37] K. Park, S. Kang, and S. Park. An integer programming approach to the bandwidth packing problem. Manuscript, 1994.

[38] M. Parker and J. Ryan. A column generation algorithm for bandwidth packing. *Telecommunication Systems*, 2:185–195, 1994.

[39] M. Parulekar and A. Makowski. Tail probabilities for a multiplexer with self-similar traffic. Manuscript, 1995.

[40] S. Plotkin. Competitive routing in ATM networks. *IEEE Journal on Selected Areas in Communications*, pages 1128–1138, August 1995. Special issue on Advances in the Foundations of Networking.

[41] M. Prycker. *Asynchronous Transfer Mode: Solution for Broadband ISDN*. Ellis Horwood, 1991.

[42] M. Stoer and G. Dahl. A polyhedral approach to multicommodity survivable network design. *Numerische Mathematik*, 68:149–167, 1994.

[43] T. Tedijanto, R. Onvural, D. Verma, L. Gün, and R. Guérin. NBBS path selection framework. *IBM Systems Journal*, 34(4):604–628, 1995.

[44] F. Vakil. A capacity allocation rule for ATM networks. In *IEEE GLOBECOM '93*, pages 406–416, December 1993.

[45] W. Verbiest and L. Pinnoo. A variable bit rate video code for asynchronous transfer mode networks. *IEEE Journal on Selected Areas in Communications*, 7(5):1253–1265, June 1989.