Link and Path Metrics for Broadband Networks: Simulation Studies using the Encourager Program

Bill Lloyd-Smith[1], Richard Harris[2] and Sanjay Bose[3]
[1,2]CATT Centre, RMIT University, Australia, [3] I.I.T. Kanpur, India

Abstract: We consider the implementation of a broadband system using digital cross-connect switches or ATM where a variety of paths may be established between the source and destination nodes. A heuristic path selection approach based on encouragement factors [2] is considered for implementation. We suggest a mathematical basis for this heuristic and study its performance using a PC-based tool, the Encourager program. The design of this tool and the results obtained from it for typical networks are presented. Future work will permit the testing of a number of extension strategies..

1. INTRODUCTION

In broadband networks and ISDN systems, digital cross-connect switches (DCS) or ATM switches may be used to connect links to allow a variety of paths between the nodes. The bandwidths along paths used to carry traffic may be varied to reflect changes in traffic loading. The performance measures of special interest will normally include the overall efficiency with which the network resources are used and the quality of service (QoS) provided to the individual traffic streams. The QoS parameter of interest will depend on the nature of the traffic stream and may include parameters such as the probability of congestion (i.e. grade of service or GOS), lost cells/packets and delay and/or delay jitter.

We consider networks where a virtual path (VP) is set up as a logical path between every active origin-destination (OD) node pair. Several VPs may be set up simultaneously between an OD pair. Every traffic stream gives rise to an OD pair. Indeed, one OD pair may correspond to several traffic streams (usually traffic classes). The VPs span one or more physical

The original version of this chapter was revised: The copyright line was incorrect. This has been corrected. The Erratum to this chapter is available at DOI: 10.1007/978-0-387-35673-0_28
C. McDonald (ed.), *Converged Networking*

links in the network with the *spare capacity* of a link is defined as the unused capacity not assigned to VPs going through that link. The *bandwidth capacity* of the individual VPs may be dynamically modified based on changes in the traffic demands of the OD pairs.

We assume that the links in a multi-node network are bi-directional. The topology (i.e. the links and nodes) is assumed to be known at each node. Each node maintains a VP database to keep track of total assigned bandwidth; the proportion of bandwidth currently used by ongoing calls and a link encouragement factor and related data on traffic passing through that link. Given a knowledge of mean arrival rates and mean service times for each call type, together with the target GOS we can estimate the bandwidth required.

Previous work of Bose et al [2], [3] investigated the use of encouragement parameters for each link to preferentially route traffic along certain preferred paths whenever possible. Moreover, certain paths may be declared in advance to be primary paths (PPs), which are preferred to other paths. Whenever a PP is defined, the Call Bandwidth Allocation algorithm (CBA) and the Virtual Path Allocation algorithm (VPA) will encourage the traffic stream to use the PP whenever possible. If no PP is defined for a traffic stream, the CBA and VPA do not have any explicit path preference.

Performance problems in teletraffic engineering can be tackled using analytic methods, simulation methods or both techniques in combination. For this application, a simulation tool (called Encourager) is proposed that will enable users to determine the performance of individual traffic streams in an ATM network presented with multi-slot traffic.

The Encourager program implements link encouragement factors, which are used to preferentially route traffic on a telecommunications network along certain paths rather than alternative paths that may also be available. The program is designed to simulate traffic in a network where the routing is achieved with the aid of these link encouragement factors. The link encouragement factors depend on available bandwidth and current traffic loading, subject to certain requirements to ensure that the traffic load is "fairly" distributed among available links. Path encouragement factors for a given path are obtained by multiplying the link encouragement factors over all constituent links making up that path. These path encouragement factors are equivalent to an additive metric, which is well known in the literature (see, for instance, Guérin and Orda [1]). We shall refer to this metric as the *encouragement metric*, which is obtained by taking the negative logarithm of the link encouragement factors.

2. ENCOURAGEMENT FACTORS

2.1 Definition of Encouragement Factors

The following description is based on that of [2]. A primary path (PP) is a declared path to which traffic routes and VPs are preferentially assigned. Often, the shortest path will be declared as the PP but other choices are allowed. An OD pair trying to set up a VP is encouraged to use the PP if possible. However, other paths may be used as well. Routing is done via a Call Bandwidth Allocation algorithm (CBA). Bandwidth requirements for an OD pair can be estimated from current traffic patterns. The VP Allocation algorithm (VPA) is used to calculate the required bandwidths that should be assigned to each OD pair. The CBA and the VPA algorithms are described later in this paper.

The CBA proposed in this paper uses encouragement factors for both the link and the path levels. It is possible for a given link l to have several PPs going through it. These OD pairs are called the *primary set* for that link. Each PP corresponds to a different OD pair. An encouragement factor is attached to each link. An OD pair in the primary set of l will receive full encouragement to use that link, so the link encouragement factor for l will be set equal to $1+\varepsilon$ where ε is a small positive number. Other OD pairs that attempt to set up a path using the link l will receive a link level encouragement somewhere in the range from 0 to 1 inclusive, depending on bandwidth requirements and the availability of bandwidth on l.

The encouragement factors reflect the link loading and the ability of the link to meet the demands of the OD pairs in the primary set. If spare capacity is available after the demands of the primary set are met, other OD pairs are encouraged to use this link. Path encouragement factors are obtained by multiplying the link encouragement factors for each constituent link on that path. In symbols, we compute the path encouragement Ω_p for a path p via

$$\Omega_p = \prod_{l \in p} \alpha_l$$

where α_l is the link encouragement factor for link l. The rules for calculating α_l are given next. Let C_l be the capacity of link l, let F_l be the spare capacity on this link and let P_l be the set of OD pairs in the primary set for this link. For OD pair x, let BT_x be the current target bandwidth requirement and let BP_x the bandwidth currently assigned to this OD pair

on its primary path. We set ε to be a small positive number, say 0.0001. Also $\varnothing$ denotes the empty set. We calculate α_l via

$$\alpha_l = \begin{cases} 0.0 & \text{if } F_l = 0 \\ 1-\varepsilon & \text{if } F_l > 0 \text{ and } P_l = \varnothing \\ 0.0 & \text{if } F_l > 0 \text{ and } P_l \neq \varnothing \text{ and } \sum_{i \in P_l} \mathrm{BT}_i \geq C_l \\ \left[\left(\sum_{i \in P_l} \min\{\mathrm{BT}_i, \mathrm{BP}_i\} \Big/ \sum_{i \in P_l} \mathrm{BT}_l\right)\right] & \text{otherwise} \end{cases}$$

In general, these link encouragement factors can be calculated in $O(mn)$ steps. This follows from the observation that $\sum_k \text{card } P_k = \sum_x h_x$ where h_x is the number of links on the (unique) PP for traffic stream x if a PP has been specified and $h_x = 0$ otherwise.

If the set P_l has more than one OD pair, fairness between them at the link level is ensured by logically partitioning the total link capacity in proportion of their bandwidth requirements and setting this as the maximum bandwidth that they will get. This limit is called the logical-link-share for this OD pair. The OD pair can find its logical share on its primary path, if any, as the minimum of its logical-link-shares for every link on its PP. Hence the relevant formulae are:

$$(\text{Logical - link - share})_l^x = \left(\mathrm{BT}_x \Big/ \sum_{i \in P_l} \mathrm{BT}_i\right) * C_l$$

$$(\text{Logical - primary - path - share})^x = \min_{l \in p} (\text{Logical - link - share})_l^x$$

2.2 Theoretical discussion

Some heuristic arguments for the use of encouragement factors are considered which formed the motivation behind their use in [2], [3]. If we consider exponentially distributed service times then the mean delay normalized to the service time (for a link) will be of the form $A/(1-\rho)$ for some $A > 0$ where ρ is the current traffic loading on the link. Of course,

$1-\rho$ will be proportional to available bandwidth. If we form link encouragement factors of the type $\exp(-A/(1-\rho))$ then we can still compute a path encouragement factor, which is the product of the link encouragements. These link encouragements lie in the range 0 to 1 and will be approximately proportional to available bandwidth when $\rho << 1$. The implied delay metric will then also be additive in the exponent. However, delay is only implicit in [2] and [3]. The actual formulae for the encouragement metric are expressed in terms of available and target bandwidths.

In the context of QoS routing with networks with inaccurate link state information Guérin and Orda [1] suggest a Most Reliable Path (MRP) algorithm for finding a route that has the maximum probability of satisfying the required bandwidth. Let $p_l(w)$ be the probability that w units of bandwidth are indeed available on link l. The MRP algorithm finds the path that maximises the product $\prod_l p_l(w)$. This reduces to a shortest path problem for a metric defined by $w_l = -\log p_l(w)$, i.e. the probability that w units of bandwidth are indeed available on link l.

This is similar to the idea in Gupta et al [2] of using encouragement factors for the preferential routing of traffic along certain preferred routes. After all, a path with high probability of accommodating a call requiring w units of bandwidth would be preferred to a path with low probability of meeting this requirement and should be given higher encouragement for this call. However, we do not treat encouragement factors in [2] as probabilities, even though they assign a weight α_l between 0 and 1 to a link l for non-primary paths. Indeed, in the event of a primary path being chosen for a call, we will have $\alpha_l > 1$. In analogy to the MRP algorithm, we maximise the product $\prod_l \alpha_l$ and we can define an additive metric for link l via $w_l = -\log \alpha_l$ after pruning links with insufficient available bandwidth. Using the Bellman-Ford algorithm, the required VPs can be found in at most $O(mn)$ steps. This is an improvement on the original version in [2] and [3].

Guérin, Orda and Williams [8] take these ideas further. They have described additions to the Bellman-Ford algorithm so that it also obtains paths of maximal available bandwidth for all hop counts. Links with insufficient available capacity are pruned before running this algorithm. Specifically, the h^{th} iteration of the Bellman-Ford algorithm is used to identify the maximal bandwidth path among all paths of at most h hops. Details are available in [9]. If we wish to use a modified Dijkstra algorithm for this purpose, we can modify the encouragement factor α_l for a link l on

a primary path by redefining it to equal 0. With this idea, a modified Dijkstra algorithm as in [8] can be used. This begins by pruning edges with insufficient available capacity. The list of equal cost next (previous) hops may be sorted according to available bandwidth so that the minimum hop count path with maximum available capacity is selected. An alternative idea is that, at each step, we only retain paths with maximum available bandwidth among all paths with equal hop count.

Now we consider the computational complexity of the methods in [8] and [9]. For the Bellman-Ford algorithm suggested in [8] the computational complexity is $O(H \cdot m)$, where m is the number of edges in the graph and H is an upper bound on the number of hops. Although H can be as large as $n-1$, in practice H is often much less than n, the number of nodes in the graph. Using the Dijkstra algorithm leads to a solution of complexity $O(m \log n)$. Our approach allows one to use a standard Bellman-Ford algorithm while the approach of Guérin et al requires the specially modified algorithms of [8] and [9] in order to take account of available bandwidth.

2.3 Computational Algorithms

We only sketch the CBA and VPA algorithms. Detailed descriptions are available in [2] and [3] so we only need to explain the modifications.

In the case of the CBA algorithm the main difficulty is that the number of VPs will be very large in a dense network with many nodes. To overcome this limitation we can convert this problem to a Bellman-Ford algorithm as described in the previous section.

In the case of the VPA algorithm, an obvious bottleneck is the number of VPs to examine, since P will be very large for a dense network. To get around this obstacle, we suggest the following modifications to the VPA algorithm. For each traffic stream, we could specify a limited set of VPs for which we allocate bandwidth according to the VPA algorithm. These VPs should preferably have relatively large values of Ω_p as they will tend to be selected by the CBA algorithm in preference to other VPs. The most promising candidates for alternate VPs are likely to be the ones with lowest hop count. As for the VPs not in the limited set described above, one might simply allow calls to use them if sufficient bandwidth exists but not make any special provision otherwise.

The technical details follow. Imagine a call from a particular traffic stream a arriving at a source node s in the network requiring x units of bandwidth. When we run the modified Bellman-Ford algorithm we obtain a tree of shortest paths from s to every other node. For each value of h in turn for $h = 1, \ldots, H$ we insist that, at the j^{th} iteration of the modified Bellman-Ford algorithm, we identify the next to maximal bandwidth path among all

paths of at most j hops when $j = h$ and otherwise we use the maximal bandwidth path. This gives rise to H_s possible shortest path trees from s to every other node and hence we get $H_{s,d} \leq H_s$ possible paths from s to a given destination d. These H_s shortest path trees can form the initial set V_0 of VPs to which we apply the VPA algorithm when we initialize the VPs from s to d.

When we update the VPs we can and probably should rely on the observed frequencies with which these VPs are selected. For each destination d only the $H_{s,d}$ most commonly selected VPs and going from s to d should be updated when VPA is rerun, giving a set V_1 of VPs to be updated. We will have no problems updating VPs belonging to $V_0 \cap V_1$ since we know what the current bandwidth allocation will be, thanks to Roberts' formula. This will give H_s preferred paths for input to VPA. The selected VPs not belonging to $V_0 \cap V_1$, i.e. those belonging to $V_0 \setminus V_1$, can be updated using the average allocation of bandwidth to calls from our traffic stream a as the "current bandwidth allocation" and proceed as before. Other paths need not be updated.

3. THE SIMULATION APPROACH

3.1 Overview of Simulator

One approach to describing traffic using an ATM network is to use the concept of traffic calls requiring multiples of a base capacity unit. For example, a voice call may require the equivalent of one 64kbit/s connection while another service may require an end-to-end connection of 128kbit/s. The latter service is regarded as needing 2 units of capacity if we regard one basic unit as being 64kbit/s. The simulator assumes that bandwidths are measured in terms of slot sizes, one slot being equal to 64kbit/s.

The program is designed as a real-time simulator to investigate traffic performance in a network where the routing is achieved with the aid of these link encouragement factors. Individual warm-up periods can be specified in order to allow the traffic load to attain statistical equilibrium. For a given traffic stream only one path can be declared as a primary path (PP). Such paths have higher path encouragements and incoming calls are encouraged to use these paths when possible. However, alternate paths may be used if the PP is not available.

The links are assumed to be bi-directional. Each node has a database to keep track of the topology, total assigned bandwidth, the proportion of bandwidth currently used by ongoing calls and a link encouragement factor and related data on traffic passing through that link. This data, together with mean arrival rates and mean service times, enables us to calculate the bandwidth required for a given GOS.

The Virtual Path Algorithm (VPA) is run at "roughly periodic" intervals (i.e. with a periodic and a random delay component) at each node to manage the assignment of VPs for every traffic stream. The Call Bandwidth Algorithm (CBA) uses the bandwidth allocation determined by the VPA to seek a path for an incoming call. If no path can be found the call is rejected.

The size of the buffer area needed for the event list is found as follows. Let m be the number of links in the network. For departures we can have at most the number of slots available on each link, since departing calls have to be already in progress. Hence for departures we reserve the amount

$$\Delta = \sum_{i=1}^{m} \text{Slot\_Capacity}(\text{link}\_i)$$

For arrivals we can only have one arrival per traffic stream. When a call arrives we immediately generate a new arrival time and insert it in the event list. Hence we have to reserve the amount $s+1$ where s is the number of traffic streams. A new VPA time must be generated immediately a VPA signal arrives. This new event is then inserted in the event list. This implies that one unit is to be available for VPA events. Similarly we should reserve one unit for network events, which correspond to a failure of a link component. Hence the total capacity of the buffer should be $\Delta + s + 3$.

A simulation study consists of a number of runs with individual warm-up periods, where statistics on the performance indicators are taken after each run and accumulated for each traffic stream in the network, or for the network as a whole (if required). At the conclusion of the simulation, the accumulated statistics are used to find the average performance of the streams and/or network. Confidence limits are obtained using standard statistical theory based on the Central Limit Theorem.

The program is limited to a maximum of 30 nodes, 30 links, 30 traffic streams, 30 chains (VPs) and each chain may have up to 7 links. The limits just described may be altered if necessary. At least two but not more than 29 runs are allowed.

3.2 Program Attributes

A schematic overview of the operation and concept of the program is shown as follows:

Key features of the program include the ability to perform real-time simulations with graphical entry of the network description. The user enters data for the simulation model through simple interfaces and dialog boxes. The network description is drawn upon an extendible canvas. The program provides a detailed report of the simulation results.

Objects specified in the model include nodes, links, chains and traffic streams. With each traffic stream, we store end-to-end demands, slot size, arrival rate and service rates. Traffic is expressed in erlangs. Capacity is expressed in terms of slot size. A number of chains are associated with each traffic stream. These serve as VPs. Each chain consists of one or more links. With each link, we store its encouragement factor and capacity. Each chain is a path from origin to destination. One of these chains is chosen by the user as the primary path. With each chain, we store mean offered traffic, mean carried traffic and end-to-end congestion. The user environment is Windows 95 or Windows NT.

The output from this program includes details of the simulation parameters used in the runs. Given these parameters, detailed results are collected as the simulations progress. The results of the simulations include detailed summary statistics such as:

The number of runs.

The number of calls offered per run.

The number of calls in the warm up phase.

The number of calls lost during each simulation run.

Offered traffic for each simulation run.

The average probability of loss is calculated, together with 95% confidence limits.

Total simulated time for each run.

In addition, the program provides a summary, including size of event queue area, total traffic offered and total traffic load offered

4. SIMULATION RESULTS

This section describes some results from the Encourager program. They are intended to approximately replicate some of the results given in Gupta's thesis (Chapter 5). Three topologies were considered in Gupta's work. It is necessary to also specify mean arrival rates and mean service times. In her simulations, the GOS was taken to be 0.1 (somewhat larger than one might often seek in practice [6], [7]). Because of this constraint on GOS the effective bandwidth is taken as given. In reality, it has to be determined from the characteristics of the traffic classes as well as the requirement that the current bandwidth may exceed the effective bandwidth with probability no more than the GOS. The target factor (see [2], [3]) was always taken to be 1.

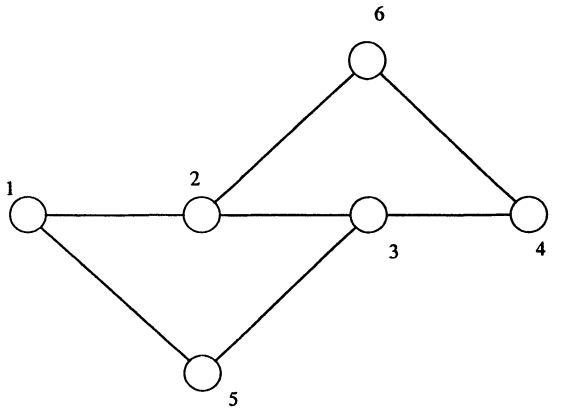

Figure 1. Trap topology

Topology I (often called the trap topology) in Gupta's thesis is illustrated below. It is fairly richly interconnected and allows a number of possible routes. This topology was the only case where Gupta considered multiclass traffic. We reproduce the effects of multiclass traffic by using two or more traffic streams for the same OD pair, where each traffic stream corresponds to a different traffic class. Traffic may enter at any of nodes 1, 2, 3, 4 and 5. Node 6 is only used for switching.

Arrivals are assumed to follow a Poisson process. Service times are assumed to follow an exponential distribution with mean equal to 1. The mean arrival rates were apparently not recorded in Gupta's thesis. Actually, we can use the inverse of the Erlang Loss formula to estimate these arrival rates. Indeed, we can reasonably assume that the arrival rates are constrained to positive integer values. This works well for single class traffic. For multiclass traffic we do not have enough equations to decide the arrival rates for the traffic classes. However we can make a plausible guess if we restrict the arrival rates to have integer values. Further, we treat the links as being partitioned for different traffic types so that we can apply the inverse Erlang approach. In practice this method yielded some useful arrival rates that could be tested.

Simulations are based on a number of runs (at least two but no more than 30). Each run consisted of 5500 offered calls, of which the first 500 were treated as warm-up calls and discarded from further analysis. In some cases the number of runs was increased in order to obtain meaningful confidence intervals. This arose because of the nature of the normal approximation used to estimate confidence intervals. Multiclass traffic is only considered for Topology I as in Gupta's work. In all cases we specified a primary path for each traffic stream.

Topology I.

The OD pairs of interest are (1,3), (2,4) and (2,5). The PPs were, respectively, 1-2-3,

2-6-4 and 2-1-5. Several cases were considered for this set of OD pairs.

Case 1.

Let us suppose that all links have capacity 16 slots each. We try arrival rates of 13 for each traffic stream. The slot size of each call is 1. We found that 1596 calls were lost out of 20000 offered calls. This gives a mean congestion of 0.0798 with a 95% CI for congestion of (0.0674079,0.0921921).

Case 2.

Let us suppose that all links now have capacity increased to 160 slots each but the slot size is still 1. We try arrival rates of 170 each. Out of 20000 offered calls 1623 were lost. We find the mean congestion to be 0.08115 with a 95% CI of (0.0589741,0.103326).

Case 3.

This example has multiclass traffic. Here we have six traffic streams, two for each OD pair. All links are supposed to have capacity 16 slots each. For each OD pair one class has slot size 1 and arrival rate 5, while the other class has slot size 5 and arrival rate 1. Out of 20000 offered calls 1253 were lost. Estimated congestion is 0.06265 with a 95% CI of (0.0531772,0.0721228).

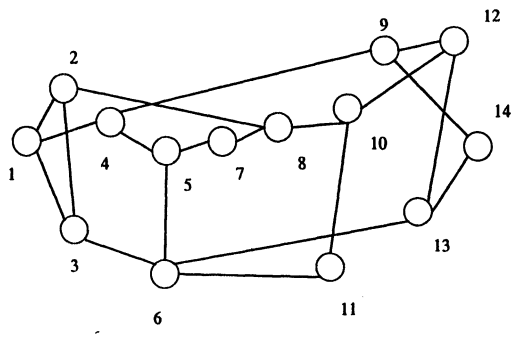

Figure 2. NSFNET topology

Case 4.

Here is a second example of multiclass traffic where the link capacity has been increased to 160 slots. Arrival rates for calls of slot size 1 were set to 101, while arrival rates for calls of slot size 5 were set to 15. This time 15 runs were made with a total of 75000 offered calls with 5445 calls being lost. The mean congestion is 0.0726 with a 95% CI of (0.0683782,0.0768218).

We have also experimented with three larger networks with more complex topologies. One of them is NSFNET with 14 nodes and 21 links. Two others are described in [5]. Two larger networks are illustrated in [4]. The purpose of these trials was to test the Encourager program on these networks with a view to assessing the scalability of the algorithms based on encouragement factors. Details of OD pairs, arrival rates and service times were not available for these networks so artificial values were developed for our simulations. Diagrams for these topologies will be reproduced below.

It was found that the program ran quite fast when these networks were tried. It is apparent that larger networks can be simulated successfully. It is clear that limiting the number of VPs is effective in keeping the running time within acceptable limits. At present there is an upper limit of 30 on the

number of nodes, links, traffic streams and chains but this can be increased if desired.

Our results are summarised in the following discussion.

The mean service time is assumed to be equal to 1 throughout. A number of traffic streams with their associated arrival rates and slot sizes were devised for these simulations. Similarly suitable values for the link capacities were devised. These decisions were required as we only had the network diagrams without any information on the traffic demands and capacities.

Results for NSFNET.

OD pairs (traffic streams) are:

(1,6) Arrival rate = 18 Slot size = 1
(2,6) Arrival rate = 18 Slot size = 1
(3,9) Arrival rate = 12 Slot size = 2
(7,11) Arrival rate = 13 Slot size = 1
(11,12)Arrival rate = 18 Slot size = 1
(4,10) Arrival rate = 20 Slot size = 1
All links have capacity 20.
Calls offered in simulation 20000
Calls lost in simulation 1991
Average congestion 0.09955
95% CI for congestion (0.0801035,0.118996)

5. CONCLUSIONS

This program is a useful tool for simulating broadband networks. However, it is of a specialised nature. It is expected that this tool will be mainly used for research purposes unless its scope can be broadened to allow for other more comprehensive simulation facilities. This simulator specifically investigates the use of, and extensions to, the Bose models for dynamic reconfiguration of networks. It is envisaged that further extensions will be developed with the capacity to include delay in the dynamic routing of calls.

We developed the program as follows:

The tool was developed in a Windows environment using an appropriate GUI package to limit the development time.

The package should be capable of easy extension in order to accommodate new facilities and ideas for dynamic reconfiguration of networks.

REFERENCES

[1] Guérin, R.A. and Orda, A. (1999), "QoS Routing in Networks with Inaccurate Information: Theory and algorithms", *IEEE/ACM Trans. Networking*, Vol. 7, No. 3, 350-364.

[2] Gupta, S., Bose, S.K., Harris, R. and Berry, L. (1998), "Distributed dynamic bandwidth allocation for self-healing broadband networks with multi-class traffic", *Proc. Globecom* '98, Sydney, Nov. 1998, Session 102.4.

[3] Gupta, S. (1998), "Distributed bandwidth allocation and call control for VP based ATM networks with multi-class traffic", M.Tech. Thesis, Dept. of Elect. Engg., I.I.T., Kanpur, India.

[4] Lee, H. et al. (2000), "Preplanned rerouting optimisation and dynamic path rerouting for ATM VP restoration", Telecommunications Systems, Vol. 14, 243-257.

[5] Zaumen, W.T. (1991), "Dynamics of distributed shortest-path routing algorithms", Proceedings, ACM, SIGCOMM '91.

[6] Freeman, Roger L. (1993) *Reference manual for telecommunications engineering*, 2nd ed. Wiley, New York.

[7] Telecom Australia (1978) *A course in teletraffic engineering*, Prepared by the Staff of Traffic Engineering Section, Planning Services Branch, Telecom Headquarters.

[8] Roch A. Guérin, Ariel Orda and Douglas Williams (1997) "QoS Routing Mechanisms and OSPF Extensions", *Proc. GLOBECOM*, Phoenix, AZ, 1903-1908.

[9] G. Apostopoulos, R. Guérin, S. Kamat, A. Orda, T. Przygienda and D. Williams (1997), "QoS routing mechanisms and OSPF extensions", Internet Draft, RFC 2676.

[10] Sanjay K. Bose, Richard Harris and Les Berry (1997) "A Simple Distributed Algorithm for Dynamic Bandwidth Allocation in Virtual Path Based ATM Networks Carrying Multi-Class Traffic" SPCOM'97, July 16-19, Bangalore, India, 81-88.