International Journal of Research Studies in Computer Science and Engineering (IJRSCSE)
Volume 2, Issue 2, February 2015, PP 1-12
ISSN 2349-4840 (Print) & ISSN 2349-4859 (Online)
www.arcjournals.org

Optimal Virtual Path Routing Control for Survivable ATM Networks

Dr. M.Usharani

Professor,
SPMVV, Tirupati

K. Sailaja

Research Scholar, SPMVV, Tirupati
sailaja.k20@gmail.com

Abstract: *In order to offer a better network survivability, it is crucial that a network manager can realize restorable traffic assignment in response to traffic dynamics and facility network reconfiguration. In this paper, I presented an optimal virtual path routing control for survivable ATM networks to minimize the expected amount of lost flow upon restoration from a network failure. The amount of lost flow is calculated based on the fast restoration algorithm such that it gives an optimal traffic distribution with the minimum service interruption. The fast restoration and optimal flow assignment have two contradicting requirements. In order to accommodate these contradicting requirements, the VP manager uses two-step restoration approach. This paper focuses on the problem of the long-term reconfiguration, which is also used for the network-wide restoration. It has been found through numerical experiments that convergence to a near-optimum is possible by properly choosing parameters.*

Keywords: *ATM, virtual path routing, network reconfiguration, Fast restoration, optimal flow assignment.*

1. INTRODUCTION

Asynchronous transfer mode (ATM) is now well recognized as the fundamental switching and multiplexing technique for future broadband ISDN's.As these networks will be increasingly relied upon for providing a multitude of integrated voice, data and video services, network reliability is a key concern.In high-speed networks, a network or even a node failure can cause a large loss of data even in a short outage. It is hence imperative to make the service interruption time as short as possible. Self-healing algorithms have been proposed to achieve fast restoration from a failure, but their success greatly depends on how traffic is distributed and how spare capacity is dimensioned over the network when the failure happens. In order to offer better network survivability, it is crucial that a network manager realizes a restorable traffic assignment in response to changing traffic demand and facility network configuration.

ATM network resource management requires highly complicated procedures since resource allocation requests from several levels of traffic entities (i.e. ATM cells, calls and virtual paths) must be handled effectively to meet the objectives regarding quality of service, already set. In order to reduce the complexity a layered switching architecture has been proposed for ATM networks. This architecture simplifies the network management process by classifying different types of network resources and traffic entities into layers. The network manager at each level can concentrate on resource allocation of its traffic entity to promote the quality of service (QoS) of its own layer. The four layers of this survivable ATM network management architecture are illustrated in fig.1 below:

The survivability functions are embedded at the VP and higher layers , considering the fact that path level recovery enables rapid and efficient restoration and considerably reduces the complexity of traffic management. Given a VP-level traffic demand satisfying call level QoS, the

VP manager configures virtual paths so the survivability measure is optimally enhanced. The VP manager also performs fast VP restoration when a network failure happens. If the VP manager cannot maintain a survivability measure at a desired level due to a growth of traffic demand, the FN layer must initiate a facility network planning process.

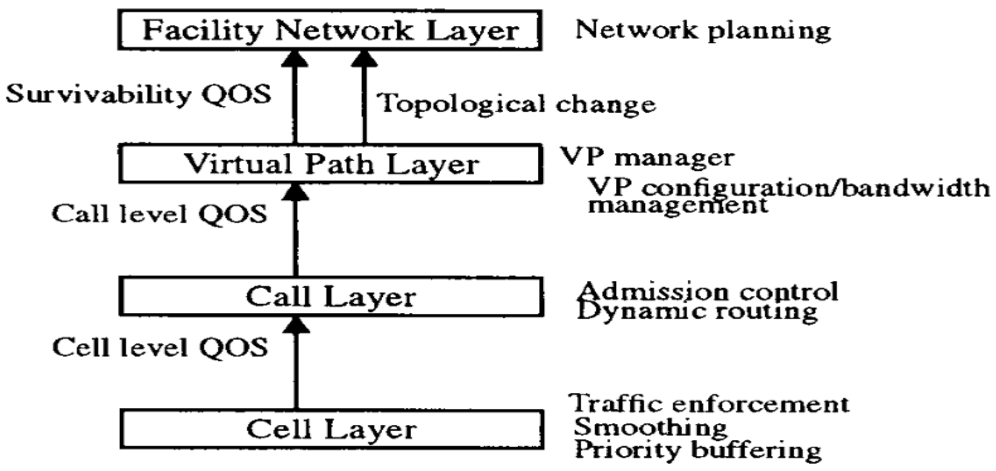


Fig1. *Layered architecture of survivable ATM networks*

When a failure happens, restoration messages are broadcast from one end of a failed facility to the other to find a set of available restoration paths. This procedure usually results in finding a set of successively shortest paths with spare bandwidth until enough restoration capacity is found. This scheme is very simple, robust to topological changes [3], and quick to achieve fast restoration 1121. However, the success of fast restoration greatly depends on how traffic is distributed and spare capacity is dimensioned over the network when a failure occurs. Thus, in order to offer a better network survivability, it is crucial that a network manager *can* realize restorable traffic assignment in response to traffic dynamics and facility network reconfiguration.

A logical transport path, called a Virtual Path (VP), **has** been proposed for the Asynchronous Transfer Mode (ATM) networks [201 [21] and adopted by CCIlT [l]. The VP is a bundle of virtual circuits and logically connects a pair of nodes which is not necessarily linked by a single cable. In ATM networks, the virtual path is considered **as** a restoration path [22] instead of a synchronized channel such **as** DS-3 in Synchronous Transfer Mode (STM) networks. In a VP-based restoration system, the spare capacity may be allocated logically as a virtual path bandwidth. Thus, the spare bandwidth can also alleviate temporal traffic congestion since the band-width is not reserved only for the restoration purpose.

2. The Functions of the VP Manager for Survivable ATM Networks

A virtual path route is established by the virtual path identifier (VPI) and the path routing table (RT).The VPI is a number contained in the cell header that identifies the assigned path of the cell. Path restoration in ATM networks is realized by redirecting cells on the failed VP to a backupVP. The VP has some unique characteristics. The most striking characteristic is the independence of route and bandwidth establishment, allowing a VP route to be established without assigning its bandwidth along the path. This is not the case in STM networks where a digital path is established by assigning a time slot of the TDM frame at each cross-connect on the path, allowing only fixed bandwidth digital paths to be established.

The objective is to find a VP configuration and bandwidth assignment in response to a dynamic change of network environment so that a self-healing algorithm can succeed. A higher level of survivability is achieved by minimizing the expected amount of lost flow upon restoration from a link failure. Since the amount of lost flow is calculated based on distributed self-healing algorithm, the solution gives an optimal traffic distribution with the minimum service interruption. Two contradicting requirements should be satisfied upon virtual path restoration. After a failure it is desirable to realize an optimal VP configuration that incurs the least service interruption upon a possible subsequent failure. The optimal flow calculation, though, introduces a computational delay which is clearly undesirable in high-speed networks where fast restoration is essential. In order to accommodate these contradicting requirements, the VP manager uses a two-step restoration approach, as shown in Fig.2 below. Upon failure, the VP restoration manager executes the fast restoration procedure to accelerate a recovery from the failure. After the restoration is completed, the VP planner module computes an optimal VP assignment for the new network topology and a VP configuration is again changed to the newly calculated optimal solution (network wide restoration) . Although this scheme temporarily produces a flow that is not optimal from the survivability viewpoint, it is acceptable in practice since the probability of more than one failure in a short time is very small.

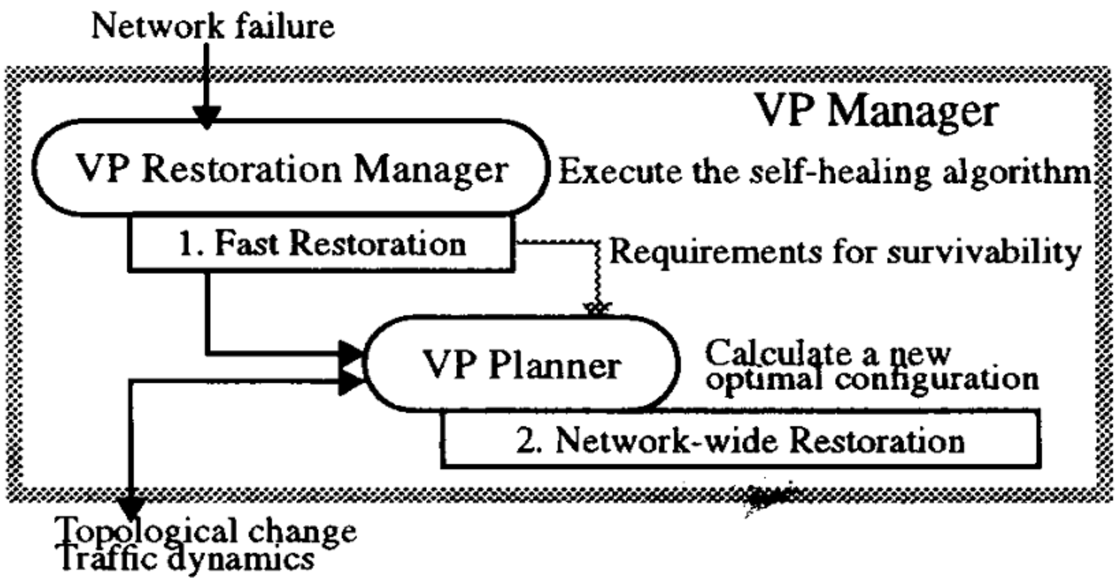

Network failure
VP Manager
VP Restoration Manager
Execute the self-healing algorithm
1. Fast Restoration
Requirements for survivability
VP Planner
Calculate a new optimal configuration
2. Network-wide Restoration
Topological change
Traffic dynamics

Fig2. *Two-step restoration approach*

Survivability quality of service (SQoS) gives a quantitative measure for the survivability level of the networks. It is used as a decision criterion for resource management control in a survivable ATM network. A good measure of survivability should express the actual amount of damage experienced in the network, instead of using traditional reliability criteria such as global availability. A number of lost calls at the call layer of a telecommunications network or the amount of lost flow are measures that have been proposed to express the damage to the VP layer.

3. SURVIVABLE VIRTUAL PATH ROUTING PROBLEM

In the formulation of the survivable virtual path routing (SVPR) problem, the end-to-end flow requirement is assumed to be given by the equivalent bandwidth of the call layer which takes cell and call level QOS into account. The aggregated traffic load can be obtained by simply adding the equivalent bandwidth of each connection [7] Using this property, we can define the problem as a multicommodity flow problem (2] with linear constraints.

An ATM exchange network is modeled as a directed graph G = (V, A, C) where $G = v_1, v_2, \ldots v_N$ is a node set representing ATM switches, $A = a_1, a_2, \ldots, a_M$ is an arc set representing optical trunks, and $\underline{c} = C_a$ is a vector representing capacity for each arc $a \in A$. We assume that a network is bidirectional with a link set $E = e_1, e_2, \ldots, e_m$ where $e_i = e_{2i-1}, e_{2i}$ and m=M/2. Suppose some amount of bandwidth is to be routed from v, to v, through direct virtual paths. Let $\Pi = \pi_1, \pi_2, \ldots, \pi_K$ be a set of such commodities in the network and $Q = q^\pi \quad \forall \pi \in \Pi$ be the requested bandwidth for each commodity. Also, let $f^\pi = f_1^\pi, f_2^\pi, \ldots f_M^\pi$ denote a flow assignment foreach commodity $\pi \in \Pi$ and $\underline{f} = f_1, f_2, \ldots f_M$ denote thetotal flowwhere $f_a = \sum_{\pi \in \Pi} f_a^\pi \; for \; \forall \alpha \in A$.

A lost flow due to a failure is used to assess the SQOS of a given traffic flow assignment. Let *S*be the set of failure patterns considered in the network design, and w_s be a weight of a failure event $s \in S$ which is determined by its likelihood or its importance. Then, the weighted lost flow *L* can beexpressed by $L = \sum_{s \in S} w_s \times L_s$ we define $w_s$as theme probability of the events, *L* becomes the expected amount of lost flow upon the failure. In the following, only the events of a single link failure are considered (namely S=E) where each failure is equally weighted $w_s = 1/|S| = 1/m$. Note that although a traffic assignment is computed assuming a single link failure, the system can work against any failure as long as the fast restoration algorithm supports it.

Now, consider the restoration process against a failure of a link e_i. Let $RP^{\alpha,k}$ be the *k*-th shortest restoration path for an arc a. Based on the fast restoration scheme, this path is the *k*-th candidate for restoring the arc flow. Let $r_\alpha^{e_i,k}$ denote the residual capacity of an arc $\alpha \in G \backslash \alpha_{2i} - 1, \alpha_{2i}$ after the link restoration using the first *k* candidate paths for the arcs α_{2i-1} and α_{2i} . Then,

$$r_\alpha^{e_i,0} \ \underline{f} \ = c_\alpha - f_\alpha$$

$$r_\alpha^{e_i,k+1} \ \underline{f} \ = r_\alpha^{e_i,k} \ \underline{f} \quad if\, \alpha \in RP^{\alpha_{2-1,i}k} \cup RP^{\alpha_{2i},k} = r_\alpha^{e_i,k} \ \underline{f} \quad - \min \ r_\beta^{e_i,k} \ : \beta \in RP^{\alpha_{2i-1},k}$$

$$if\, \alpha \in RP^{\alpha_{2i-1},k} = r_\alpha^{e_i,k} \ \underline{f} \quad - \min \ r_\beta^{e_i,k} \ : \beta \in RP^{\alpha_{2i},k} \quad if\, \alpha \in RP^{\alpha_{2i},k}$$

Using this recursive formula, the restorable arc follow, $R_\alpha \ \ \alpha \in \ a_{2i-1,} \alpha_{2i}$,if asobtained by, $R_\alpha \ \underline{f} \ = \sum \min \ r_\beta^{e_i,k} \ : \beta \in RP^{\alpha_{2i},k}$ where k_α is the number of the restoration paths for the arc α. Now a lost flow due to a failure of a link e_i, denoted by L_{e_i} is expressed as follows;

$L_{e_i} \ \underline{f} \ = \max \ 0, f_{\alpha 2i-1} - R_{\alpha 2i-1} \ \underline{f} \ + \max \ 0, f_{\alpha 2i} - R_{\alpha 2i} \ \underline{f}$ Using *L* as an objective function, the SVPR problem is formulated as follows;

Minimize $L \ \underline{f} \ = \frac{1}{m} . \sum_{e \in E} L_e \ \underline{f}$ (1)

Over $\underline{f}^\pi = f_1^\pi, f_2^\pi, \ldots, f_m^\pi \quad \forall \pi \in \Pi$

Subject to $\sum_{\alpha \in A} \phi_{v,\alpha} . f_\alpha^\pi = \eta_v^\pi q^\pi \qquad \forall \pi \in \Pi \ \forall v \in V$ (2)

$f_\alpha = \sum f_\alpha^\pi \le c_\alpha \quad \forall \alpha \in A$ (3)

$\underline{f}^\pi \ge 0 \qquad \forall \pi \in \Pi$ (4)

Where $\phi_{v,\alpha}$ and $\phi_{v,\alpha}$ are,

$$\phi_{v.\alpha} = \begin{cases} -1 & If\ the\ arc\ \alpha\ leaves\ the\ node\ v \\ 1 & (If\ the\ arc\ \alpha\ enters\ the\ node\ v) \\ 0 & (Otherwise) \end{cases}$$

$$\eta_v^\pi = \begin{cases} -1 & If\ v\ is\ an\ originating\ node\ of\ VP\ \pi \\ 1 & (If\ v\ a\ destination\ node\ of\ VP\ \pi) \\ 0 & (Otherwise) \end{cases}$$

4. SOLUTION APPROACH

The SVPR problem can be solved by an iterative method where a new feasible flow is found at each iteration and the value of the objective function monotonically decreases. In order to maintain the feasibility, not only a flow must be a legal multi-commodity flow [8] but also the capacity constraints (3) must be checked at each iteration. The method also needs to rind an initial feasible flow, which may be obtained through a linear programming algorithm although it is very time-consuming for a large network. Instead of solving the above problem directly, we relax the capacity constraints as in [9], If the total flow of each link is allowed to exceed its capacity, the capacity constraints can be removed from the formulation and the iterative procedure is greatly simplified. Moreover, an initial solution is now obtained through a well-known shortest path algorithm which is less computationally intensive than a simplex method especially for a large network. Let

$\Omega_A = \underline{f} : \underline{f}\, is\, a\, multicommodity\, flow\;\; \underline{f}\, satisfies\;\; 2\;\; and\;\; 4$ and $\Omega_B = \Omega_A \cap \;\; \underline{f} : \underline{f}\, satisfies\;\; 3$ .The removal of the capacity constraints means expanding a feasible region from Ω_B to Ω_A . This is only possible if an iterative algorithm converges to Ω_B even if it starts from $\Omega_A \setminus \Omega_B$. As shown in the Lemma below, this is accomplished by modifying the definition of *L* as follows;

$$L^*\;\; \underline{f} \;\; = \frac{1}{m}.\sum_{e\in E} L_e^*\;\; \underline{f} \tag{5}$$

Where $L_e^*\;\; \underline{f} \;\; = L_e \underline{f} + \omega.\sum_{\alpha\in A} \max\;\; 0, f_\alpha - c_\alpha$ Thus, if there is an arc ($a \in A$) where the capacity constraint is violated (max $0, f_\alpha - c\alpha \;\; > 0$,the excess flow $f_\alpha - c_\alpha$ is treated as a big lost flow by adding $\omega.\;\; f_\alpha - c_\alpha \;\; to L_e$ for all $e \in E$.

In summary, the survivable virtual path routing (SVPR) problem is reformulated as follows, which we call the relaxed SVPR (RSVPR) problem.

Minimize $L^* = \sum_{e\in E} L_e^*$ Over $\underline{f}^\pi = f_1^\pi, f_2^\pi, ..., f_M^\pi \;\; \forall \pi \in \Pi$ subject to the Equations (2) and (4). An alternative way to find a feasible decent direction is to minimize the directional derivative over all feasible directions. Since any feasible direction at $\underline{f}$ can be expressed by $\underline{v}$ -$\underline{f}$ for some $\underline{v} \in \Omega_A \setminus \;\; \underline{f}$, the problem is formulated as a minimization of DD $\underline{v}, \underline{f}$ over $\underline{v} \in \Omega_A \setminus \;\; \underline{f}$ where D $\underline{v}, \underline{f}$ is the directional derivative of $L^* \underline{f}$ along $\underline{v}$ - $\underline{f}$. Due to non-smoothness of L^*, however, the directional derivative cannot be calculated just through a gradient In order to overcome the difficulty, the following approximation, DD* $\underline{v}, \underline{f}$, is employed instead of DD. $DD^* g^*\;\; \underline{f} \bullet \frac{\underline{v} - \underline{f}}{\|\underline{v} - \underline{f}\|}$ where $g^*\;\; \underline{f} \;\; = \left[g_j^*\;\; \underline{f} \;\; \right] \; j \in A$

$$g_j^+\;\; \underline{f} \;\; = \begin{cases} g_j^+\;\; \underline{f} & if\, v_j - f_j > 0 \\ g_j^-\;\; \underline{f} & if\, v_j - f_j < 0 \end{cases}$$

$g_j^+\;\; \underline{f} \;\; = \lim_{h\to 0^+} \;\; L^*\;\; \underline{f} + h.e_i \;\; - L^*\;\; \underline{f} \;\; / h$ Where e_i, is a unit vector with the *i*-th component equal to one. Note that DD* $\underline{v}, \underline{f}$ is a directional derivative of a piece-wise linear approximation of *L** using one-sided partial derivatives, $g_i^+\;\; \underline{f} \;\; and\; g_i^-\;\; \underline{f}$. Now, the problem is to minimize DD* $\underline{v}, \underline{f}$ over $\underline{v} \in \Omega_A \setminus \;\; \underline{f}$. Consider applying the flow deviation method for this new minimization problem. Again, the objective function DD* $\underline{v}, \underline{f}$ is not differentiable everywhere. However, the only non-differentiable region is $A = \;\; \underline{v} : v_i = f_i \;\; \exists i \in A$. The partial derivative of DD* $\underline{v}, \underline{f}$ (except at the points in A) can be obtained as

$$\frac{\partial}{\partial v_i} DD^* \ \underline{v},\underline{f} \ = \frac{g_i^* \ \underline{f}}{\sqrt{\sum_j \ v_j - f_j \ ^2}} - \frac{\left[\sum_j g_j^* \ \underline{f} \ . \ v_j - f_j \ \right]. \ v_i - f_i}{\left(\sum_j \ v_i - f_i \ ^2\right)^{\frac{3}{2}}} \tag{8}$$

When $v_j - f_j = 0$, we heuristically define $g_j^* = \ g_j^+ + g_j^- \ / 2$ in (8) in order to compute the partial derivative. According to our numerical experimentations, $v_j - f_j = 0$ seldom happens during the iterations.DD* $\underline{v},\underline{f}$ is not convex nor concave, so the flow deviation method gives only a stationary point of DD* $\underline{v},\underline{f}$. With a good initial point $\underline{v}^0$, however, it has a good chance to converge to a near-optimal point. If L^* is differentiable at $\underline{f}$, it is known that $\underline{v} - \underline{f}$ gives a feasible decent direction [8] where y is a shortest route flow obtained under the metric $\partial L^* / \partial f_i$,which is equal to $g_i^+ = g_i^- = \ g_i^+ + g_i^- \ / 2$. Although L^* is not usually differentiable at the convergence point of each iteration, $\underline{v}^0 - \underline{f}$ would give a good approximation to a decent direction if $\underline{v}^0$ is a shortest route flow under the metric $g_i^+ + g_i^- \ / 2$. Therefore, using the $\underline{v}^0$ as a starting point, the method could at least reach a local minimum with negative directional derivative and possibly reach the minimum of DD* $\underline{v},\underline{f}$. The proposed algorithm is summarized as follows;

0. Find a feasible starting point $\underline{f}^0 \in \Omega_A$ using a shortest path algorithm and let n=0.

1. Through the following procedures, obtain $\underline{v}^\pi$ such that $\underline{v}^\pi - \underline{f}^\pi$ would be the steepest decent direction.

a) Find a feasible starting flow $\underline{v}^0 \neq \underline{f}^n \ in \ \Omega_A$ and letm=0. $\underline{v}^0$ is a shortest route flow under the metric $g_i^+ + g_i^- \ / 2$.

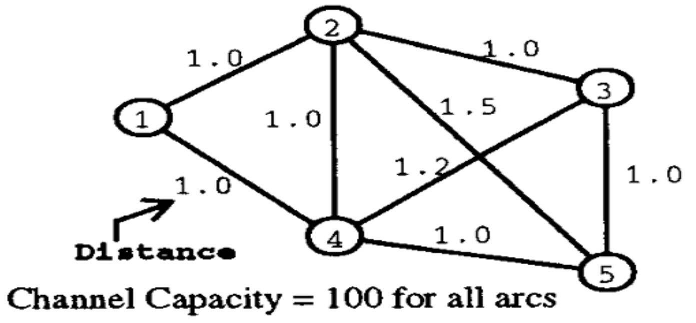


Fig3. *Sample network 1.5 node network*

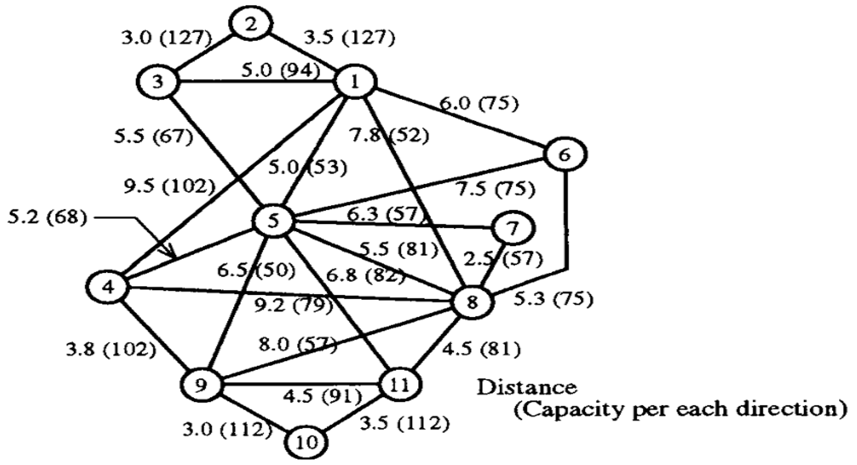


Fig4. *Sample network 2. Metropolitan case study*

b) Obtain $\underline{w}^m$ as a shortest route flow taking $\partial DD^* \; \underline{v}^m, \underline{f}^m \; / \partial v_i$ as a distance.

c) $\underline{v}^{m+1} = 1-\mu^m \because \underline{v}^m + \mu^m \underline{w}^m$ where μ^m is the minimizer of $DD^* \; 1-\mu^m \; \underline{v}^m + \mu^m \underline{w}^m \; ,\underline{f}^m) \; 0 \le \mu^m \le 1$

d) If $DD^* \; \underline{v}^m, \underline{f}^n \; - DD^* \; \underline{v}^{m+1}, \underline{f}^n \; < \delta$,then $\underline{v}^n - \underline{v}^{m+1}$ and go to the step 2. Otherwise, let m=m+l and go back to the step 1-b).

2. $\underline{f}^{n+1} = 1-\lambda^n \; \underline{f}^n + \lambda^n \underline{v}^{n,}$ where λ^n is the minimizer of $L^* \; 1-\lambda^n \; \underline{f}^n + \lambda^n \underline{v}^n$

$0 \le \lambda^n \le 1$.

3. If $L^* \; \underline{f}^n \; L^* \; \underline{f}^{n+1} \; < \varepsilon,$ then stop.Otherwise, let n=n+l and go back to the step 1.

5. NUMERICAL EXAMPLES

The proposed algorithm was coded in C and the program was executed on a DEC 5000/25 workstation. It was examined in a small test network with 5 nodes and 16 directed arcs as shown in Figure 3 as well as in two sample networks which appeared in the literature; the metropolitan case-study network from [23] with 11 nodes and 46 directed arcs (Figure 4), and the US long-haul case-study network used in [12] with 28 nodes and 90 directed arcs (Figure 5). In the experiment, it is assumed that there is a requirement to establish direct virtual paths for each direction of every node pair. Thus, the total number of commodities, *K,* is equal to *N(N-1)* and this amounts to 20, 110 and 756 for 5, 11, and28 node models, respectively. The flow requirement is assumed to be given in terms of an equivalent bandwidth at the call layer.

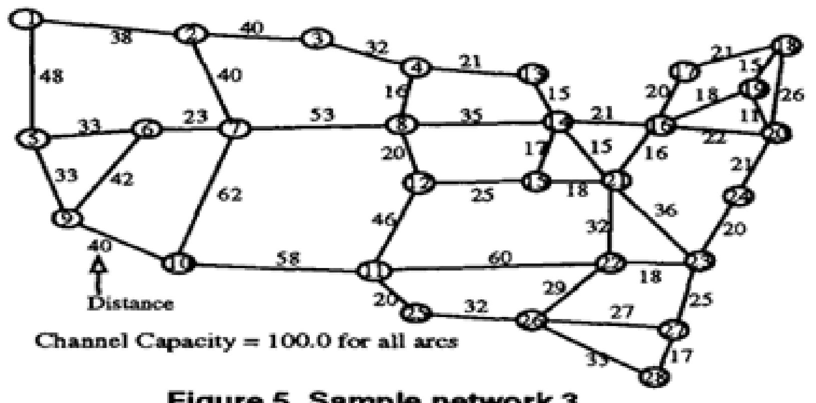
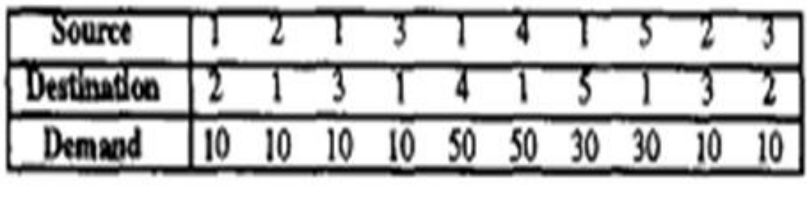
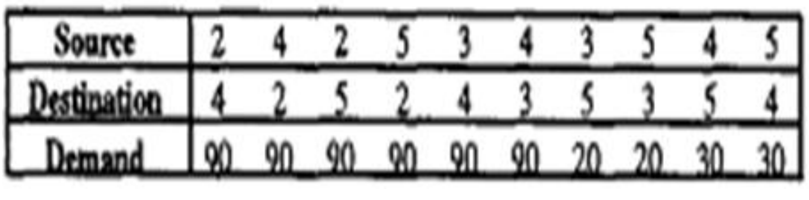


Figure 5. Sample network 3. US long-haul case study

Table1. *Traffic Demands*



Source	1	2	1	3	1	4	1	5	2	3
Destination	2	1	3	1	4	1	5	1	3	2
Demand	10	10	10	10	50	50	30	30	10	10



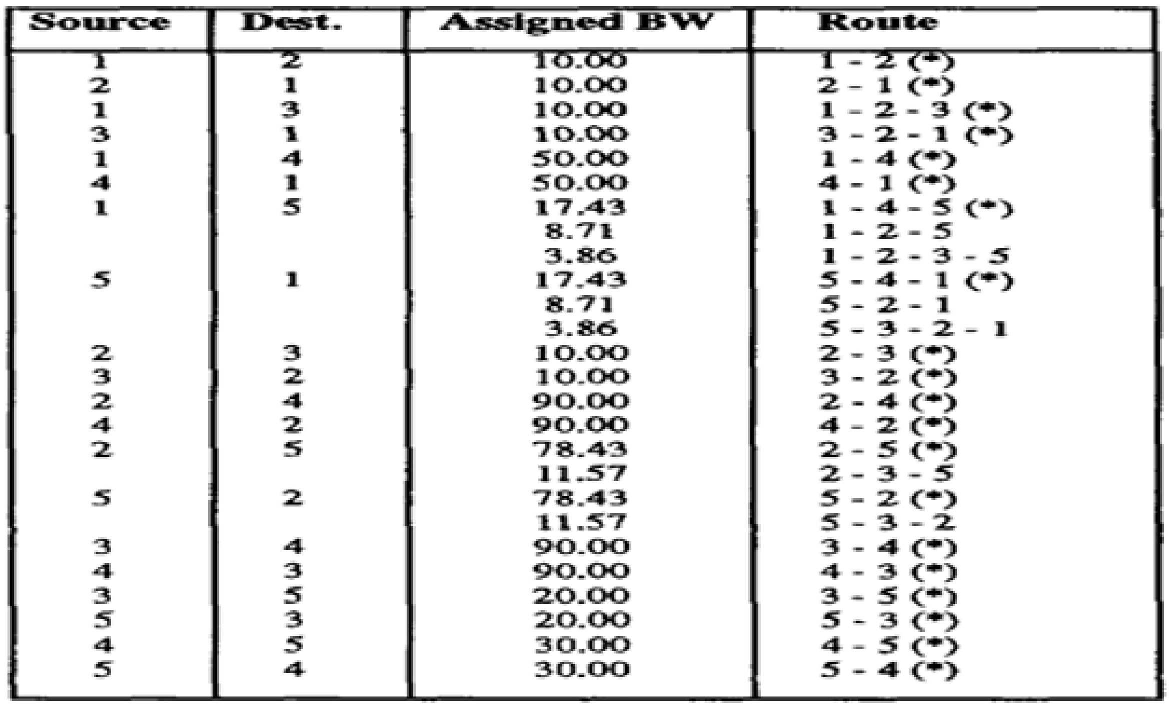
Source	2	4	2	5	3	4	3	5	4	5
Destination	4	2	5	2	4	3	5	3	5	4
Demand	90	90	90	90	90	90	20	20	30	30

Since $L^* \; \underline{f}$ and $L^* \; \underline{g}$ have no closed form solutions, they can be calculated only numerically. The calculation of $L_e^* \; \underline{f}$ involves repeated application of the shortest path algorithm to find successively shortest restoration paths over the residual network. Assuming the Dijkstra's algorithm with binary heap [5], it requires $0 \; A^3 \log V$ time in the worst case since at least one arc is removed from the residual network at each application of the shortest path algorithm. Then, the total running time for $L^* \; \underline{f}$ and $g^* \; \underline{f}$ goes up to 0(A^3log V) and 0(A^4logV), respectively. Although their complexities are polynomial, they slow down the algorithm since these operations are invoked every outer iteration. In order to reduce the computational time, a pre-calculated table of the restoration paths is used in our implementation. Although the size of such table grows exponentially as the size of network increases, our seminary study shows that it is enough to consider the first 30 shortest restoration paths to obtain a very close approximation of $L^* \; \underline{f}$. This strategy reduces the computational complexities of $L^* \; \underline{f}$ and $g^* \; \underline{f}$ down to 0(A) and 0(A^2), respectively. A golden section search method (17) is used to implement the line search in step l-c)andstep2).

First, the proposed algorithm has been examined on the small test network with traffic demands as listed in Table 1. Table 2 shows the result of virtual path assignment. The algorithm has reached to the optimum solution in two iterations. Although the fast restoration scheme cannot recover all the flow against some link failures *(L* =2)* under the initial flow (the shortest route flow), the algorithm could find apoint where *L* =0.* Most of the flow is still routed over the shortest route, but some additional virtual paths are introduced to achieve

L =0.*

Table2. *Virtual path flow assignment(* indicates the shortest route)*



Source	Dest.	Assigned BW	Route
1	2	10.00	1 - 2 (*)
2	1	10.00	2 - 1 (*)
1	3	10.00	1 - 2 - 3 (*)
3	1	10.00	3 - 2 - 1 (*)
1	4	50.00	1 - 4 (*)
4	1	50.00	4 - 1 (*)
1	5	17.43	1 - 4 - 5 (*)
		8.71	1 - 2 - 5
		3.86	1 - 2 - 3 - 5
5	1	17.43	5 - 4 - 1 (*)
		8.71	5 - 2 - 1
		3.86	5 - 3 - 2 - 1
2	3	10.00	2 - 3 (*)
3	2	10.00	3 - 2 (*)
2	4	90.00	2 - 4 (*)
4	2	90.00	4 - 2 (*)
2	5	78.43	2 - 5 (*)
		11.57	2 - 3 - 5
5	2	78.43	5 - 2 (*)
		11.57	5 - 3 - 2
3	4	90.00	3 - 4 (*)
4	3	90.00	4 - 3 (*)
3	5	20.00	3 - 5 (*)
5	3	20.00	5 - 3 (*)
4	5	30.00	4 - 5 (*)
5	4	30.00	5 - 4 (*)

There are three parameters which might affect the performance of the algorithm; the stopping conditions, ε and δ , and the step size, *h*, which is used to numerically calculate the partial derivatives, $g_i^* \; \underline{f}$ The stopping condition,δ, has influence not only on the required number of the inner iterations but also on that of the outer iterations. Less inner iterations are necessary for larger δ, while less outer iterations are required for smaller δ since it could find a steeper downhill at each iteration. Considering the fact that the outer iteration is much more computationally expensive than the inner iteration, it is better to use a smaller value for δ when longer iterations are expected. The stopping condition ε does not have much to do with the convergence as long as it is sufficiently small. Otherwise, premature termination may result.

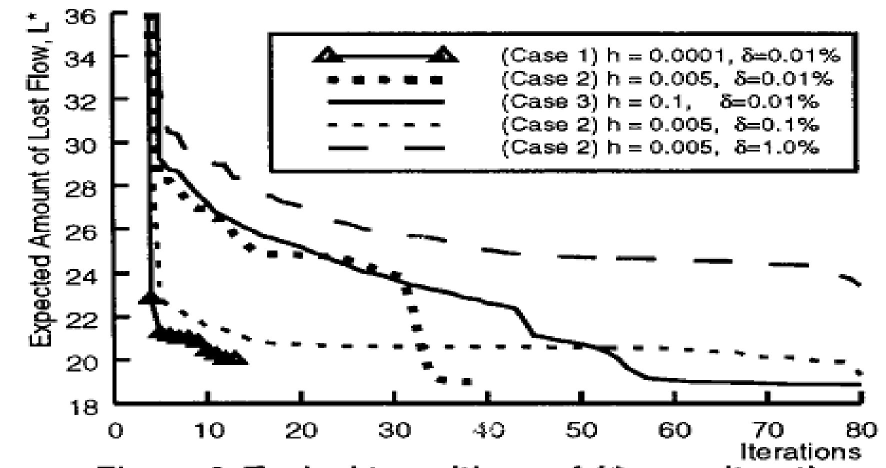


Figure 6. Typical transitions of *L over iterations (11 node network)**

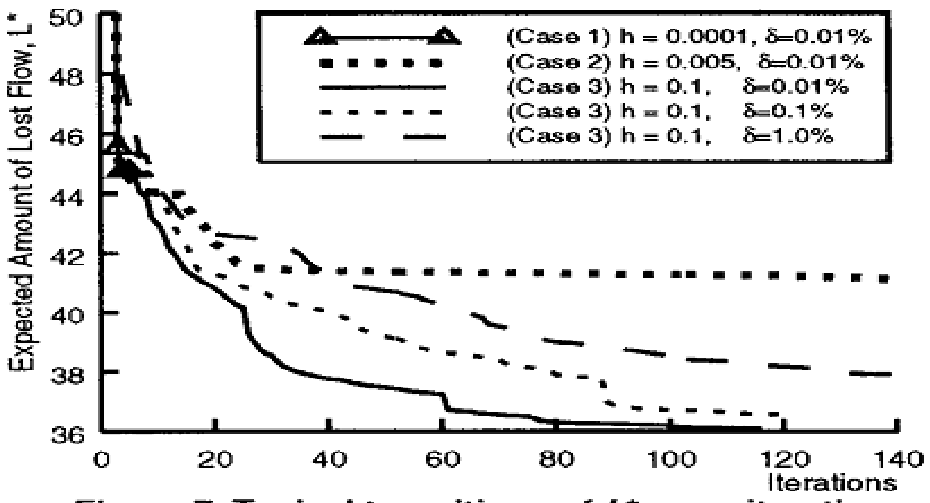


Figure 7. Typical transitions of *L over iterations (28 node network)**

When the load is light, the procedure converges to the point with $L^* = 0$, usually in a few iterations. The selection of the above parameters does not make any significant difference in the convergence speed as long as the step size is reasonably small. Since die region $\Omega_0 = \underline{f} : L^* \underline{f} = 0$ is large in a lightly loaded situation, the procedure can find a point in ii0 rather easily. In general, the convergence point $\underline{f}^n$ depends on the choice of die parameters, but it is not a problem from the survivability point of view since $L^* = 0$ at any convergence point.An increase in the traffic demand makes Ω_0 empty where L^* is nonzero for some link*e* no matter how traffic is distributed. Although higher load results in longer iterations, the necessary number of outer iterations is still small at this stage, typically less than 10. With any reasonable step size the algorithm converges to a near-optimal point, although faster convergence is possible for smaller *h*.

When the load increases further, possibly due to a network failure, L_e^* becomes nonzero for all links. At this stage, the VP planner tries to find a single minimum point, which further prolongs the iterations. The convergence speed gready depends on the selection of the two parameters, *h* and δ, in heavily loaded situations. Figures δ and δ illustrate typical transitions of L^* over iterations for various values of *h* and δ. These figures show the results obtained from the 11-node network model and 28-node network model, respectively. The latter case has heavier load than the former. Three curves are plotted for each case representing small (case 1), medium (case 2) and relatively large (case 3) values of *h* as well as two curves for different values of δ,. For all cases, rapid decrease in L^* is observed in the first few iterations. Since the starting point of the algorithm is a shortest route flow, it generally violates capacity constraints in many arcs for a heavily loaded network and the flow is far away from the feasible region Ω_B. Due to the definition of L^*, great improvement per iteration is possible at this region regardless of the step size.

Although die fastest decrease is usually observed for small h (case 1), the procedure often stops prematurely in a heavily loaded situation (Figure 6). This is especially true when load becomes too heavy (Figure 7). The premature convergence is due to lack of global information on the change of L^*. A search direction obtained through a smaller step size might be decent only in a neighborhood around $\underline{f}$. This area is usually small for heavily loaded networks since a small flow change at one link can readily affect die restorability of some other links. Since the minimum point along the search direction often falls near f, enough progress cannot be made at each iteration. As a result, a decrease of $L*$ is too small to pass the termination test and premature termination follows. When h is relatively large (case 3), the procedure converges very slowly since the resulting search direction is not truly steepest decent. However, when the load grows, this is the only case to approach to a near-optimal point. Since the next direction is obtained by seeking a broader range around $\underline{f}$, the minimum point along the search direction tends to reside away from $\underline{f}$. Namely, die procedure can find a longer decent slope although it might not be the steepest decent direction at $\underline{f}$. Consequently, it usually produces sufficient decrease to prevent premature convergence even in a heavily loaded network although the attained improvement rate might be low. In other words, the linear approximation of $L*$ with a large step size is not precise locally but gives its global view. It works well in heavily loaded situations since die effect of dense non-smooth points is smoothed out over a relatively large neighborhood. In addition, a search direction obtained through larger step size might find a direction with very long downhill and this results in occasional large reduction of $L*$.

In summary, the best choice of the step size gready depends on the traffic load. When a load is light, small h produces die best result as expected. When load becomes heavier, however, h should be increased to avoid premature convergence. In case of Figure 6, a medium value of h (case 2) gives the best result, while small h yields a quick improvement but results in premature termination with a slightly higher $L*$. A near optimal point can be also attained with large h, but it takes longer iterations. When load further increases, the algorithm can reach a near optimal point only through large step size. In this case (Figure 7), the procedure terminates shortly after early improvement with a small value of h, and stays at a higher level with a medium value of h. This phenomenon is very typical for a large network. Since non-smooth points become much denser than a small network even with the same load, larger step size is more helpful to obtain the global view of $L*$ and to avoid a locally steep but short slope. On the other hand, a search direction based on a local view causes premature termination if h is small or slow convergence if it's not large enough to find a long downhill.

As for δ, a smaller value is preferable since a higher convergence rate is expected. With small δ, a steep decent direction can be found and the number of outer loops is greatly reduced (see Figures 6 and 7). Therefore, it requires less CPU time for heavily loaded networks in spite of the increased number of inner iterations. On the other hand, large δ may cause premature convergence since the minimization process of DD* can terminate before the direction becomes a decent one. The value of δ, however, should not be unnecessarily small. The objective function of the inner iteration is just an approximation and fine tuning on such function might not always lead to better solution unless the approximation is very precise. In fact, smaller 8 occasionally fails to find the better direction as in the case of Figure 6, where the curve with δ=0.01 shows slower improvement than that with δ=0.1 at the early stage of the iterations. According to our experiment, δ less than 0.1% usually works well.

6. CONCLUSION

I have proposed an optimal virtual path routing control for survivable ATM networks which minimizes the expected amount of lost flow upon restoration from a network failure. The concept of two-step restoration has been introduced to realize both fast restoration and optimal reconfiguration. The proposed VP flow reconfiguration can be used not only to achieve the best network survivability after a change in the facility networks, but also to guarantee the quality of service of the lower layers in response to varying demand while maintaining the survivability level. The problem has been formulated as a nonlinear, non-smooth multi-commodity flow problem with linear constraints. The definition of the objective function has been extended to simplify the procedure, and the modified flow deviation approach has been developed to obtain the steepest decent direction of the non-smooth objective function. It has been found through numerical experiments that convergence to a near-optimum is possible by properly choosing parameters. Although the scheme has been proposed for the ATM networks, it is also applicable to the STM networks.

REFERENCES

[1] CCITT Recommendation, I Series (B-ISDN), Nov. 1990.

[2] A.A.Assad, "Multicommodity Network Flows - A Survey", Networks, vol.8, pp.31-91, 1978.

[3] J.E.Baker, "A Distributed Link Restoration Algorithm with Robust Preplanning,** IEEEGlobecom '91, pp.306-311, Dec. 1991.

[4] D.Bertsekas and R.Gallager, "Data Networks," Prentice-Hall, 1987.

[5] D.Bertsekas, "Linear Network Optimization: Algorithms and Codes," MIT Press, 1991.

[6] J.A.Bondy and U.S.R.Murty, "Graph Theory with Applica¬tions," McGraw-Hill, 1976.

[7] A.I.Etwalid and D.Mitra. "Effective Bandwidth of General Markovian Traffic Sources and Admission Control of High Speed Networks (Extended Abstract),*' IEEE LNFOCOM '93, pp.256-265, 1993.

[8] LPratta, M.Gerla and K.Kleinrock, "The Flow Deviation Method: An Approach to Store-and-Forward Communication Network Design," Networks, vol.3, pp.97-133,1973.

[9] A.Gersht and A.Shulman, "Optimal Routing in Circuit Switched Network," IEEE Trans, on Comm., vol.37,no.ll, pp.I203-I211,Nov. 1989.

[10] A.Gersht and S.Kheradpir, "Real-Time Bandwidth Allocation and Path Restorations in SONET-Based Self-Healing Mesh Networks," IEEE ICC '93. pp.250-255. May 1993.

[11] P.E.Green Jr., "Fiber Optic Networks," Prentice Hall Inc., Englewood Cliffs, N.J,1993.

[12] Wayne D. Grover, "The Selfhealing Network: A Fast Distrib¬uted Restoration Technique for Networks Using Digital Cross-connect Machines," IEEE Globecom '87, pp. 1090-1095, Dec. 1987.

[13] W.D.Grover, TD.Bilodeau and B.D.Venables, "Near Optimal Spare Capacity Planning in a Mesh Restorable Network," IEEE Globecom '91, pp.2007-2012, Dec. 1991.

[14] R.Guerin, H.Ahmadi and M.Naghshineh, "Equivalent Capac¬ity and Its Application to Bandwidth Allocation in High-Speed Networks," IEEE JSAC, vol.9, no.7, pp.968-981. Sep. 1991.

[15] J.Y.Hui, M.B.Gursoy, N.Moayeri and R.D.Yates. "A Layered Broadband Switching Architecture with Physical and Virtual Path Configurations," IEEE JSAC.vol.9, no.9, pp.1416-1426, Dec. 1991.

[16] C.G.Kang and H.H.Tan, Fault-Tolerant Capacity and Flow Assignment in Packet Switched Networks," IEEE MILCOM '92,pp.l65-171, 1992.

[17] D.G.Luenberger, "Linear and Nonlinear Programming," 2nded..Addison-Wesley. Reading, M.A., 1984.

[18] S.E.Minzer. Broadband ISDN and Asynchronous Transfer Mode (ATM)." IEEE Comm. Magazine, vol.27, no.9. pp.17-24, Sep. 1989.

[19] H.Sakauchi, Y.Nishimuraand S.Hasegawa, "A Self-Healing Network with an Economical Spare-Channel Assignment," IEEE Globecom '90, pp.438-443, Dec. 1990.

[20] K.Sato, S.Ohta and I.Tokizawa, "Broad-Band ATM Network Architecture Based on Virtual Paths," IEEE T^ans. on Comm., vol.38, no.8. pp.1212-1222, Aug. 1990.

[21] K.Sato and I.Tokizawa, "Flexible Asynchronous Transfer Mode Networks Utilizing Virtual Paths," IEEE ICC '90. pp.831-838, 1990.

[22] K.Sato, H.Hadama and I.Tokizawa, "Network Reliability Enhancement with Virtual Path Strategy," IEEE Globecom'90, pp.464-469. Dec. 1990.

[23] C.H.Yang and S.Hasegawa, "FITNESS: Failure Immunization Technology for Network Service Survivability," IEEE Globe¬com '89, pp.1549-1554, Dec. 1989.

AUTHORS' BIOGRAPHY

Dr.M.Usha Rani is Professor in the Department of Computer Science, Sri Padmavati Mahila Visvavidyalayam (SPMVV Womens' University), Tirupati. She did her Ph.D. in Computer Science in the area of Artificial Intelligence and Expert Systems. She is in teaching since 1992. She presented many papers at National and Internal Conferences and published articles in national & international journals. She also written 4 books like Data Mining - Applications: Opportunities and Challenges, Superficial Overview of Data Mining Tools, Data Warehousing & Data Mining and Intelligent Systems & Communications. She is guiding M.Phil. and Ph.D. in the areas like Artificial Intelligence, DataWarehousing and Data Mining, Computer Networks and Network Security etc.

K.Sailaja is a Research scholar in the Department of Computer Science, Sri Padmavati Mahila Visvavidyalayam (SPMVV Womens' University), Tirupati. She did her M.Phil in computer science in the area of WDM networks. She presented papers at National and Internal Conferences and published articles in national & international journals.