Virtual Path Assignment and Virtual Circuit Routing in ATM Networks

Frank Yeong-Sung Lin
Bellcore
Piscataway, NJ 08854

Kwang-Ting Cheng
AT&T Bell Laboratories
Murray Hill, NJ 07974

Abstract — The use of virtual paths in ATM networks reduces the call set-up delays, simplifies the hardware in the transit nodes and provides simple virtual circuit admission control. However, it also reduces the degree of capacity sharing and, thus, increases the call blocking rate. We consider the following problem: given a network topology, link capacity of each physical link and traffic requirement of each origin-destination pair, we want to jointly determine the following four design variables: (1) the pairs of nodes that should have virtual paths, (2) the route of each virtual path, (3) the bandwidth assigned to each virtual path, and (4) the routing assignment for each virtual circuit (call), to minimize the expected call blocking rate subject to call set-up time constraints, or, alternately, to minimize the call set-up delay subject to the expected call blocking rate constraints. The problem is formulated as a nonlinear nondifferentiable combinatorial optimization problem. We also present simplified formulations for networks with abundant capacity or with limited capacity (two special cases of the general problem). Algorithms for solving the two special cases are proposed and implemented. We present computational results and make comparisons of different schemes.

1. Introduction

An important concept associated with ATM is the use of virtual paths [1, 2]. A virtual path (VP) is a logical connection for a node pair by means of a label in the header of an ATM cell named Virtual Path Identifier (VPI). Each VP is considered as a logical link for a certain service. Thus VP subnetworks for different services can be built within an ATM network. Each VP is assigned a number of physical links and an effective capacity (in terms of the maximum number of virtual circuits allowed) to assure quality of service (QOS) requirements. Several VPs may be multiplexed on the same physical link. Figure 1 (from [3]) shows the concept of virtual paths. There are three virtual paths, VP_1, VP_2 and VP_3, shown in the figure. VP_2 is a virtual path between *end nodes* N_1 and N_5. Nodes N_3 and N_4 are *transit nodes* of virtual path VP_2.

The advantages of using virtual paths include:

- Decrease of call set-up delays: At call set-up, the routing tables of the transit nodes need not be updated. Routing procedure is also avoided at the transit nodes.
- Simple hardware: Because the call set-up functions are eliminated, the processing load decreases. This leads to low cost node construction.
- Logic service separation on network service access.
- Simple virtual circuit admission control: This point will be elaborated later in this section.

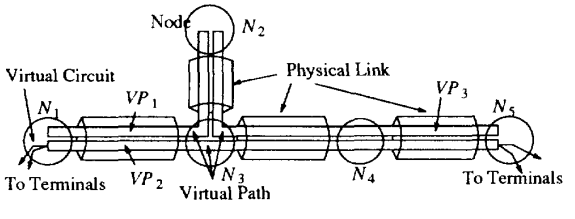


Fig. 1: Schematic illustration of virtual paths

Whereas, the disadvantage of using virtual paths is that the network throughput decreases (the total call blocking rate increases) because the degree of capacity sharing decreases.

In [3], the effect of VPs in ATM networks was investigated. A number of advantages of using VPs are highlighted. In addition, simple adaptive VP bandwidth control schemes are proposed. In the proposed schemes, VPs are assumed to be given and the assigned capacities are adjusted according to the loads. The trade-offs between transmission link utilization (throughput) and processing load (as a function of the stepsize and frequency of changing the bandwidth of each VP by the dynamic bandwidth allocation schemes) were investigated. In their analysis, only one link (i.e. single hop) is considered and, thus, routing was not taken into account. The effect of VPI translation delay in a mutli-hop (on the VP level) environment as well as the statistical multiplexing effect are not considered. In [4], the routing problem in VP-based ATM networks is considered. Dynamic routing policies are proposed attempting to minimize the VC blocking rate. Simulations are used to evaluate the proposed routing policies.

To the best of our knowledge, there has not been published work considering the virtual path assignment and virtual circuit routing assignment problem jointly. In this paper, we address the problem of assigning virtual paths and determining virtual circuit (call) routing assignments for each pair of communicating nodes in an ATM network. Given a network topology, link capacity of each physical link and traffic requirement of each origin-destination pair, we want to jointly determine the following design variables: (1) the pairs of nodes that should have virtual paths, (2) the route of each virtual path, (3) the bandwidth assigned to each virtual path, and (4) the routing assignment for each virtual circuit (call), to minimize the expected call blocking rate subject to call set-up time constraints, or, alternately, to minimize the call set-up delay subject to the expected call blocking rate constraints.

The routing policy considered in this paper is described below.

- The virtual paths that (i) connect the same origin-destination (O-D) pair and (ii) involve the same number of physical links are jointly considered as a *macro logical link*.
- On the arrival of a virtual circuit, a path, which may contain multiple macro logical links, is selected from a given set of candidate paths with a given probability.
- If each macro logical link on the path has sufficient spare capacity to accommodate this new virtual circuit, this new virtual circuit is admitted and routed over the selected path; otherwise, the virtual circuit is rejected.

Under this routing policy, the problem described above is formulated as a nonlinear nondifferentiable combinatorial optimization problem. In the formulation, we implicitly use the concept of *effective channels* of physical links. In ATM networks, the cell loss requirement is considered the most stringent and usually dominates other performance requirements. Therefore, in our study, we allocate the cell loss requirement to each physical link. This allocated requirement at each physical link l is then converted into effective capacity of the physical link, C_l. The allocated requirement for each link and, thus, the effective capacity, is a function of the maximum number of physical links used by each O-D pair, M, which is another design variable. Under

0-7803-0917-0/93$03.00 © 1993 IEEE

this principle, the quality of service (QOS) for each session can then be guaranteed by limiting the number of established sessions (assigned channels) on each link l by C_l, which is computed from a given M. A more detailed discussion of this point is given in Section 6.

One clear advantage of the above performance objective allocation scheme is that the virtual circuit admission control and routing process becomes simple, mainly due to the avoidance of usually complicated calculations of the QOS of existing sessions that share one or more of the physical links the new virtual circuit will use.

Due to the hardness of the general problem and usually the abundant capacity of an ATM network, we consider a special case of the general problem where each call is allowed to use exactly one VP (single-hop on the VP level) so that the call set-up delay is kept minimum. A simplified problem formation is then considered.

The basic approach to the algorithm development for this simplified problem is Lagrangean relaxation, penalty function and subgradient optimization techniques. In computational experiments, the proposed algorithm facilitates solutions that are within 8% of an optimal solution for all five networks we have experimented with.

To illustrate the trade-off between the call set-up delay and the call blocking rate, another special case of the general problem is also considered where each physical link is considered a virtual path. This scheme is recommended for networks with very limited capacities and capacity sharing is highly desirable. In this formulation the call blocking rate is minimized without considering the call set-up delay constraints. This problem is formulated as a standard multicommodity flow problem so that standard solution procedures, e.g. the Frank-Wolfe method, can be applied.

The two special schemes are compared. Computational results indicate that the first scheme (i.e. each call is allowed to use exactly one virtual path) incurs higher blocking probability. Given the topology of the network, the link capacity of each physical link and traffic requirement for each O-D pair, if the blocking probability derived by the algorithms given in Section 5 is less than the specified threshold, it will be a preferable scheme because of the simplification in the call set-up process. On the other hand, if the computed blocking rate is not acceptable, the second scheme (i.e. each physical link is considered as a virtual path) or a hybrid scheme as formulated by Formulation 1 should be used to reduce the blocking probability with minimally incurred extra delay in call set-up.

The remainder of this paper is organized as follows. In Section 2, a general problem formulation is presented. In Section 3, a special case of the general formulation where each virtual circuit is allowed to use exactly one virtual path (to minimize the call set-up delay) is considered and a simpler problem formulation (than the general formulation) is presented. In Section 4, another special case of the general formulation where each physical link is considered as a virtual path (i.e. the call set-up time constraint in the general formulation is ignored) is considered. A simple surrogate problem formulation is presented. In Section 5, solution procedures for the two special cases are proposed. In Section 6, we report the computational results.

2. Problem Formulation

An ATM network is modeled as a graph $G_l(N,L)$ where the ATM switches are represented by the node set $N = \{1,2,...,n\}$ and the optical trunks (physical links) are represented by the link set L. Let V be the set of virtual paths. The logical topology (on the virtual path level) is represented by $G_v(N,V)$ where each virtual path in V is considered as a logical link in $G_v(N,V)$. Since a node pair in the network may be directly connected by multiple virtual paths, $G_v(N,V)$ is a multigraph. Let V_{ijk} be the set of logical links that connect node pair (i,j) and involve k physical links. Each of the logical links which involves k physical links is referred to as a type-k logical link. Let V_l be the set of logical links that use physical link l.

Let W be the set of O-D pairs in the network. For each O-D pair $w \in W$, sessions (virtual circuits) are established (if admitted) and terminated with time. Throughout this paper, sessions and virtual circuits are used interchangeably. In this study, we consider one class of sessions, e.g. video, where all the sessions have the same inter-cell arrival time distribution (hence the same mean traffic rate as well) and holding time distribution (similar assumptions were made in [3]). A bandwidth which is equal to the mean traffic rate of a session is referred to as a channel. The capacity of each physical link then can be translated into a number of channels. We specify the link capacity by effective channels as described in the last section.

The average arrival rate of new sessions for O-D pair w is λ_w (sessions/sec). We assume that the session arrivals to each O-D pair can be modeled as a Poisson process. We also assume that the holding time of each session is generally distributed with mean $1/\mu$ sec. Let Q_w be a given set of simple directed paths in $G_v(N,V)$ for O-D pair w. To satisfy the QOS, each path in Q_w should involve no greater than M physical links. Let Q be $\bigcup_{w \in W} Q_w$. For each path $q \in Q$ and node pair (i,j), let $\delta_{q,ijk}$ be 1 if path q passes a direct type-k logical link that connects node pair (i,j). Let y_q, $q \in Q_w$ be the amount of flow that O-D pair w routes over path q. The aggregate flow (virtual circuit set-up demand) on type-k logical links in V_{ijk}, denoted by g_{ijk}, then can be expressed as $\sum_{q \in Q_w} \sum_{w \in W} \delta_{q,ijk}\, y_q$. Let c_v be the number of channels in logical link v. Note that exactly c_v channels are allocated on every physical link that is involved in logical link v. For each physical link l, the total number of channels assigned is $\sum_{v \in V_l} c_v$. Let $v_{ijk} = \sum_{v \in V_{ijk}} c_v$ be the total number of channels in the type-k logical links connecting node i to node j. In the proposed virtual circuit routing scheme, all the v_{ijk} channels are considered jointly as a macro logical link, denoted by m_{ijk}, with capacity v_{ijk}. Let B_{ijk} be the blocking probability for node pair (i,j) and type-k logical links, i.e. for macro logical link m_{ijk}. Let H_q be the set of all macro logical links that path q uses. Let $H_{q,ijk}$ be the set of all macro logical links prior to m_{ijk} that path q uses. Under the assumptions of Poisson arrival processes, $B_{ijk}(g_{ijk}, v_{ijk})$ can be calculated by the Erlang's B formula [5]:

$$B_{ijk}(g_{ijk}, v_{ijk}) = \frac{\dfrac{(\frac{g_{ijk}}{\mu})^{v_{ijk}}}{v_{ijk}!}}{\sum_{a=0}^{v_{ijk}} \dfrac{(\frac{g_{ijk}}{\mu})^{a}}{a!}}. \qquad (2.1)$$

The problem of determining virtual path assignments and virtual circuit routing assignments in an ATM network is formulated as the following nondifferentiable nonconvex combinatorial optimization problem.

$$Z_{IP1} = \min \sum_{q \in Q_w} \sum_{w \in W} y_q \left(1 - \prod_{m_{abd} \in H_q} (1 - B_{abd})\right) \quad \text{(IP1)}$$

subject to:

$$g_{ijk} = \sum_{q \in Q_w} \sum_{w \in W} \delta_{q,ijk}\, y_q \prod_{m_{abd} \in H_{q,ijk}} (1 - B_{abd})$$

$$\forall\ i,j \in N,\ i \neq j,\ k \in \{1,2,..,n-1\} \qquad (2.2)$$

$$\sum_{q \in Q_w} y_q = \lambda_w \quad \forall\ w \in W \qquad (2.3)$$

$$\sum_{v \in V_l} c_v \leq C_l \quad \forall\ l \in L \qquad (2.4)$$

$$\sum_{v \in V_{ijk}} c_v = v_{ijk} \quad \forall\ i,j \in N,\ i \neq j,\ k \in \{1,..,n-1\} \qquad (2.5)$$

$$y_q \geq 0 \quad \forall\ q \in Q_w,\ w \in W \qquad (2.6)$$

$$\frac{\sum_{i,j \in N,\, i \neq j} \sum_{k=1}^{n-1} g_{ijk}}{\sum_{w \in W} \lambda_w} \leq K \qquad (2.7)$$

c_v *is a nonnegative integer* $\forall\ v \in V$. (2.8)

The objective function is to minimize the total virtual circuit blocking rate. The right hand side of Constraint (2.2) is the aggregate flow on the macro logical link m_{ijk}, taking into account the traffic loss along the path. Constraint (2.3) requires that all traffic for each O-D pair be serviced. The left hand side of Constraint (2.4) is the total number of channels allocated (to virtual paths) on each physical link l. Constraint (2.4) requires that the worst-case (all the allocated channels are used at the same time) aggregate link flow not exceed the effective link capacity (to satisfy the QOS). The left hand side of Constraint (2.5) is the total number of channels allocated to macro logical link m_{ijk}. Constraint (2.6) requires that the amount of flow routed over each path nonnegative. The left hand side of Constraint (2.7) is the average number of hops (in terms of virtual paths/logical links) for a virtual circuit. Constraint (2.7) requires that the average number of hops for a virtual circuit be no greater than a given constant K. It is clear that K must be greater than or equal to 1 for (IP1) to be feasible. Constraint (2.8) requires that the number of channels allocated to each virtual path be a nonnegative integer.

Another possible formulation of the joint virtual path assignment and virtual circuit routing assignment problem is (i) to replace the objective function of (IP1) by the minimization of the average number of hops (in terms of virtual paths) for a virtual circuit (the left hand side of Constraint (2.7)) and (ii) to replace Constraint (2.7) by

$$\sum_{q \in Q_w} \sum_{w \in W} y_q \left(1 - \left(1 - \prod_{m_{abd} \in H_q} B_{abd}\right)\right) \le J \qquad (2.9)$$

where J is a given constant. With this change, we try to minimize the call set-up delay subject to a given call block rate constraint.

Below we briefly discuss the hardness of (IP1). Due to Constraint (2.8), (IP1) is an integer programming problem which is known to be hard to solve exactly. In addition, B_{ijk} is a discrete and highly nonlinear function. One may use linear interpolation (details are given in the next section) to make B_{ijk} continuous and convex with respect to v_{ijk}. However, this modified B_{ijk} is nondifferentiable with respect to v_{ijk} and nonconvex with respect to g_{ijk}. Even with the simplifying assumption made in Section 5 which changes the problem so that one needs to consider $g_{ijk}\, B_{ijk}$ (which is a convex function of g_{ijk} [6]) rather than B_{ijk} alone, however, numerical examples show that $g_{ijk}\, B_{ijk}$ is not jointly convex with respect to g_{ijk} and v_{ijk}. From an inspection of the objective function and Constraint (2.2), it is clear that (IP1) is not a convex programming problem.

Under the circumstances where the network capacity is abundant (the nature of ATM networks), one may consider to limit the number of hops (in terms of virtual paths) for a virtual circuit (to choose a small value of K in Constraint (2.7)) at the cost of throughput (the degree of virtual path sharing among O-D pairs is reduced, which causes the increase of the total blocking rate). The extreme case is to make each virtual circuit use exactly one virtual path (K is set to 1). In this implementation, it is clear that the call set-up time is minimized. Another advantage associated with this scheme is that the switch hardware may be significantly simplified, since no intermediate virtual path ID translation is required. In the next section, a formulation for this virtual path assignment and "single-hop" virtual circuit routing assignment problem is provided. As compared to (IP1), the extreme problem ($K = 1$) formulation is much simpler and possesses convexity which facilitates efficient and effective solution procedures as proposed in Section 5.

3. Formulation for Networks with Abundant Capacities

In this section, a special case of (IP1) where $K = 1$ is considered. This requires that each virtual circuit use exactly one virtual path so that the call set-up delay is kept minimum. This implementation is favorable when the call set-up time is considered as a critical performance measure and the network has abundant capacity so that the achievable total call blocking rate is within an acceptable range.

We assume that each session uses exactly one virtual path (single-hop routing on the virtual path level). Let P_w be a given set of simple directed paths for O-D pair w where each path involves at most M hops. Let P be $\bigcup_{w \in W} P_w$. Let x_p be the number of channels allocated (for the virtual path that connects the same O-D pair as path p) on path p. In this case, exactly x_p channels are allocated on every link on path p. For each path $p \in P$ and link $l \in L$, let δ_{pl} be 1 if link l is on path p and 0 otherwise. The number of channels allocated on link l can then be expressed as $\sum_{p \in P_w} \sum_{w \in W} \delta_{pl}\, x_p$. For each O-D pair w, let the total number of channels assigned be $v_w = \sum_{p \in P_w} x_p$, which is the capacity of the virtual path that directly connects O-D pair w. Let B_w be the blocking probability for O-D pair w. We denote λ_w/μ as γ_w. Similar to Equation (2.1), $B_w(v_w)$ can be calculated by the Erlang's B formula:

$$B_w(v_w) = \frac{\dfrac{\gamma_w^{v_w}}{v_w!}}{\displaystyle\sum_{i=0}^{v_w} \frac{\gamma_w^{i}}{i!}} \qquad (3.1)$$

where γ_w is considered given. $B_w(v_w)$ is defined only on nonnegative integer points. For the purpose of algorithm development, linear interpolation is used to make $B_w(v_w)$ a continuous function of v_w. More precisely, for all nonnegative integer i, points $(i, B_w(i))$ and $(i + 1, B_w(i + 1))$ are connected by a segment. From [7] this new (continuous) blocking function is convex with respect to v_w.

The special case of (IP1) where $K = 1$ can be formulated as the following nondifferentiable combinatorial optimization problem.

$$Z_{IP2} = \min \sum_{w \in W} \lambda_w\, B_w(v_w) \qquad \text{(IP2)}$$

subject to:

$$v_w = \sum_{p \in P_w} x_p \qquad \forall\ w \in W \qquad (3.2)$$

$$\sum_{p \in P_w} \sum_{w \in W} \delta_{pl}\, x_p \le C_l \qquad \forall\ l \in L \qquad (3.3)$$

$$x_p \text{ is a nonnegative integer} \qquad \forall\ p \in P_w,\ w \in W. \qquad (3.4)$$

The objective function is to minimize the total call blocking rate. The right hand side of Constraint (3.2) is the total number of channels allocated to each O-D pair w (the capacity of the virtual path directly connecting each O-D pair w). The left hand side of Constraint (3.3) is the total number of channels allocated (to virtual paths) on each link l. Constraint (3.3) requires that the worst-case (all the allocated channels are used at the same time) aggregate link flow not exceed the effective link capacity (to satisfy the QOS requirements). Constraints (3.4) requires that the number of channels allocated on each path be a nonnegative integer.

As compared to (IP1), (IP2) is simpler in terms of the number of constraint sets (3 versus 7) and the number of decision variables (the virtual circuit routing decision variables are omitted). It should be pointed out that (IP2) is not exactly equivalent to the special case of (IP1) where $K = 1$. Indeed, the optimal objective function value of (IP2) is no greater than that of the special case of (IP1) due to the consolidation of all types of macro logical links for each O-D pair as a super macro logical link. This is valid in terms of assuring the QOS requirements since all macro logical links for each O-D pair involve no more than M physical links. Therefore, all macro logical links for each O-D pair can be jointly considered when a virtual circuit admission control and routing decision is made (a higher degree of capacity sharing is achieved).

4. Formulation for Networks with Limited Capacities

For the purpose of showing the trade-off between the average call set-up delay and the total call blocking rate, we consider (IP1) where: (1) Constraint (2.7) is assumed to be inactive by setting a sufficiently large number to K (one may consider K to be infinity in this case), and (2) Each VP involves exactly one physical link. In this

special case, the total call blocking rate is minimized without considering the call set-up delay constraints. In general, the total blocking rate does not increase when the degree of channel sharing among O-D pairs increases. This special case is to consider each physical link as a virtual path to achieve a high degree of channel sharing and hence a low total call blocking rate.

The graph model and notation used in Section 3 are used in this section with the following changes. Let z_p be the amount of flow routed over path p. Let B_l be the call blocking probability of link l. Let g_l be the aggregate flow (call set-up requests) on link l.

Two approximations are made to construct a convex programming problem formulation. The approximations basically lead to overestimation of the call blocking rate for each O-D pair and it can easily be seen that the surrogate formulation is an accurate approximation when the blocking probability on each virtual path (physical link in this case) is small.

The first approximation is to replace the end-to-end call blocking probability $1 - \prod_{l \in h_p} (1 - B_l)$ for each path p by $\sum_{l \in h_p} B_l$, where h_p is the set of virtual paths (physical links in this case) on path p. It is obvious that $\sum_{l \in h_p} B_l$ is an upper bound on $1 - \prod_{l \in h_p} (1 - B_l)$. In addition, if each B_l is small, the bound is tight.

The second approximation is to ignore the traffic (call set-up requests) losses when aggregate link flows are calculated. This has an effect of overestimating the aggregate link flows and thus the call blocking probability for each link (since each B_l is a monotonically increasing function of the aggregate link flow). It is also clear that when each B_l is small, this approximation leads to tight upper bounds on the link flows and link-level call blocking probabilities.

Combining the above two approximations, the total call blocking rate can be expressed as $\sum_{l \in L} g_l B_l(g_l)$, which, as discussed above, is an upper bound on the exact total call blocking rate as given in (IP1). From [6], $g_l B_l(g_l)$ is a monotonically increasing and convex function of g_l.

The special case of (IP1) (K is infinite and each physical link is considered as a VP) with the aforementioned approximations is formulated as the following convex programming problem.

$$Z_{IP3} = \min \sum_{l \in L} g_l B_l(g_l) \qquad \text{(IP3)}$$

subject to:

$$g_l = \sum_{p \in P_w} \sum_{w \in W} \delta_{pl} z_p \le C_l \quad \forall\, l \in L \qquad (4.1)$$

$$\sum_{p \in P_w} z_p = \lambda_w \quad \forall\, w \in W \qquad (4.2)$$

$$z_p \ge 0 \quad \forall\, p \in P_w,\ w \in W. \qquad (4.3)$$

The performance measure is an approximate expression (upper bound) of the total call blocking rate. The left hand side of Constraint (4.1) is the aggregate flow on link l (ignoring call losses in upstream nodes). Constraint (4.1) requires that the aggregate flow not exceed the effective link capacity (to satisfy the QOS). Constraint (4.2) requires that all traffic for each O-D pair be serviced. Constraints (4.3) requires that the flow routed on each path be nonnegative.

It is clear that (IP3) is a standard multicommodity flow problem where standard solution procedures are available to solve the problem. For simplicity, we apply the Frank-Wolfe method.

5. Algorithms

In this section, we describe methods for solving Problem (IP2). For Problem (IP3), we simply apply the well-known Frank-Wolfe (Flow Deviation) method [8]. The details will not be given in this paper. For Problem (IP2), we use Lagrangean relaxation to obtain tight lower bounds and use several different heuristics to obtain sub-optimal feasible solutions and upper bounds.

Lower bounds (Lagrangean relaxation)

We dualize Constraint (3.3) and obtain the following Lagrangean Relaxation problem:

$$Z_D(u) = \min \sum_{w \in W} \lambda_w B_w(v_w) + \sum_{l \in L} u_l \Big(\sum_{p \in P_w} \sum_{w \in W} \delta_{pl} x_p - C_l \Big) \qquad \text{(LR2)}$$

subject to Constraints (3.2) and (3.4).

By using the weak Lagrangean duality theorem, the optimal objective function value of (LR2), $Z_D(u)$, is a lower bound on Z_{IP2}. We would like to determine the greatest lower bound by

$$Z_D = \max_{u \ge 0} Z_D(u). \qquad \text{(D)}$$

Subgradient method for Problem (D)

There are several methods for solving the dual problem (D). We use the subgradient method.

The solution to problem (LR2) at iteration k of the subgradient optimization procedure, $Z_D(u^k)$, is given below. The problem can be solved separately for each O-D pair. For each O-D pair $w \in W$, we find a shortest path $p_w{}^1 \in P_w$, where $u_l{}^k$ is the arc weight for link l. For $p \ne p_w{}^1$, x_p is 0. We then assign an integer value between 0 and C_l to $x_{p_w{}^1}$ that minimizes the following convex function:

$$\lambda_w B_w(x_{p_w{}^1}) + \sum_{l \in L} u_l{}^k \delta_{p_w{}^1 l} x_{p_w{}^1}.$$

Let an $|L|$ vector y be a subgradient of $Z_D(u)$. In iteration k of the subgradient optimization procedure, the multiplier for each link $l \in L$ is updated by

$$u_l{}^{k+1} = u_l{}^k + t^k y_l{}^k.$$

The step size t^k is determined by

$$t^k = \delta \frac{Z_{IP2}{}^h - Z_D(u^k)}{\|y^k\|^2} \qquad (5.1)$$

where $Z_{IP2}{}^h$ is an objective function value for a heuristic solution (upper bound on Z_{IP2}) and δ is a constant, $0 < \delta \le 2$.

Upper bounds (heuristic solutions)

The set of $\{x_p\}$ values obtained by solving Problem (LR2) described above may not be a valid solution to Problem (IP2) because the capacity constraints may be violated for certain links. Even if there is no violation to the link capacity constraints, it may not be a "good" solution. This is because for each O-D pair, only one path is assigned channels. Therefore, we need additional heuristics to derive valid and good values for $\{x_p\}$ and, thus, a tight upper bound on Z_{IP2}. In this section, we describe the details of these heuristics. We first describe the "drop and add" heuristics that gradually modify a given set of $\{x_p\}$ to satisfy the link capacity constraints and to minimize the total blocking rate.

Drop and Add Heuristics

Given an initial set of values for $\{x_p\}$, the *drop heuristic* trys to decrease the values of certain x_p's to satisfy the link capacity constraints, if the initial set of values violates the constraints. The drop heuristic also attempts to minimize the increase of the blocking rate when twisting the values of x_p's to satisfy the link capacity constraints. After the drop heuristic is applied, a valid set of $\{x_p\}$ values will be obtained. The *add heuristic* is then applied to increase the values of certain x_p's to utilize the unassigned link channels to reduce the blocking rate. The heuristic find an O-D pair w and a path p_w which has at least one unassigned channel for each of the links in p_w. The value of x_{p_w} is then increased by one. If there are more than one path having free channels in each of its links, the path with the largest blocking rate

will be selected. Assigning an additional channel to such a path tends to maximize the reduction of the blocking rate.

Drop heuristic - Given x_p, for all $p \in P_w$, we first compute the total number of channels allocated (to virtual paths) on each link l, denoted as g_l. If $g_l \le C_l$, the given set of x_p forms valid solution and the "drop" process will not be performed. Otherwise, a link with a largest ratio of g_l/C_l, denoted as l^1, is identified. Then, we identify a path, p^1, which satisfies the following properties: (1) x_{p^1} is positive (i.e. the path has been assigned some channels). (2) Path p^1 contains link l^1. (3) The O-D pair w^1 which contains p^1 has the lowest blocking rate among all O-D pairs. Once p^1 is identified, we then decrease the value of x_{p^1} by 1. This step will reduce the amount of overflow by one in the congested link l^1 (Requirement (2)). However, it also increases the blocking rate of O-D pair w^1. Requirement (3) above is intended to minimize the increase of the overall blocking rate. We then recompute the total number of channels allocated on each link. This updated information is then used to select the next link and path for dropping a channel. The iterative process will continue until a valid solution is found (i.e. the number of allocated channels is less than or equal to the link capacity for all links).

Add heuristic - Given a valid set of x_p, $p \in P_w$, if there are unassigned channels in any VP, the "add" heuristic is applied to assign those channels for further reducing the blocking rate. Similar to the "drop" heuristic, the "add" heuristic adds one channel for a selected path at each iteration. At each iteration, a path, p^2, that satisfies the following properties is identified: (1) There exists at least one unassigned channel for all links contained in p^2. (2) The O-D pair w^2 which contains p^2 has the highest blocking rate, among all O-D pairs that contain at least one path satisfying property (1). Once p^2 is identified, we then increase the value of x_{p^2} by 1. Assigning one channel to a path having Property (1) guarantees that it will not violate the link capacity constraints. Assigning one channel to a path having Property (2) attempts to maximize the amount of reduction in the overall blocking rate for such an assignment. The process continues until no free channel is left. We applied the drop and add heuristic at selective iterations of the subgradient method. Since the set of x_p values generated by solving Problem (LR2) may not be a "good" initial point for the add and drop heuristics. We devise an alternative method to derive a better initial solution for the add and drop heuristic.

Penalty Function Method

We convert the link capacity constraint (3.3) into a penalty function and add it to the objective function. The integer constraints (part of Constraint 3.4) are relaxed. Let k be a large constant. The original integer programming problem is then converted into the following:

$$Z_{PF2} = \min \sum_{w \in W} \lambda_w B_w(v_w) + \sum_{l \in L} f_l(\mathbf{x}) \qquad \text{(PF2)}$$

subject to $x_p \ge 0$, $\forall\ p \in P_w$, $w \in W$, where

$$f_l(\mathbf{x}) = \begin{cases} k \cdot \left(\sum_{w \in W} \sum_{p \in P_w} \delta_{pl} x_p - C_l \right)^2 & \text{if } \sum_{w \in W} \sum_{p \in P_w} \delta_{pl} x_p > C_l \\ 0 & \textit{otherwise.} \end{cases}$$

Subgradient Method for Problem (PF2)

We use the subgradient method to solve the problem. As mentioned in Section 3, linear interpolation is used to make $B_w(v_w)$ a continuous function of v_w.

The routing and number of channels assigned for each O-D pair by this formulation/method may not form a feasible solution to Problem (IP2) due to the following two reasons: (1) the assigned channel numbers may not be integers, and (2) the link capacity constraints may still be violated because k is a finite number. We can convert it into a feasible solution by (1) rounding the assigned channel numbers to integers and (2) postprocessing the assigned channel numbers using the add/drop heuristics described above. In general, the combination of this formulation and the subgradient method produce better initial solution for the add/drop heuristics to generate a tighter upper bound. Therefore, this combination is adopted in our experiments for deriving upper bounds.

6. Computational Results

The algorithms described in Section 5 were coded in the C language. The algorithms were tested on five networks - ARPA2, ARPA1, OCT, SWIFT and PSS with 21, 61, 26, 15 and 14 nodes, respectively. Their topologies can be found in [9]. For each of the five networks, we try different combinations of link bandwidths and traffic requirements for each O-D pair. For each test case, it is assumed that the traffic requirements for all O-D pairs are the same.

Formulation 2 - Table 1 shows the results for Formulation 2 - minimizing the expected call blocking rate using "single-hop" (on the virtual path level) virtual circuit routing. The bandwidths and traffic requirements are chosen in such a way that the resulting average blocking probability for each O-D pair is within our interested window (i.e. the average blocking probability is around 1.0e-3). The columns of Lower and Upper bounds give the lower and upper bounds of the expected blocking rate for each O-D pair computed by the algorithms/heuristics given in Section 5. The last column gives the difference of these two bounds divided by the lower bound.

Formulation 3 - Table 2 shows the results for Formulation 3 where the same objective as in Formulation 2 is used and the call set-up delay constraint as considered in Formulation 1 is omitted. In addition, every physical link is considered as a virtual path. The delays are estimated and measured by the average number of hops used for each call. For all cases, the expected blocking probability is negligible ($< 10^{-8}$) and is not shown in the table. However, as explained above, routing needs to be computed for every call which incurs extra delays.

Table 1 - Experimental results for Formulation 2.

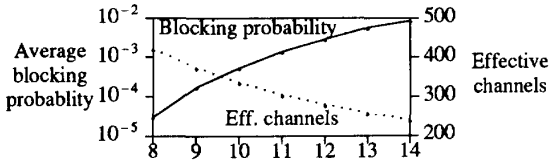
Network ID	Link capacity	Traffic req./ OD pair	Lower bound	Upper bound	Difference (%)
ARPA2	240	1.0	2.38e-4	2.57e-4	7.98
ARPA2	320	2.0	7.50e-4	8.11e-4	8.13
ARPA2	480	3.0	6.83e-4	7.40e-4	8.34
ARPA1	1000	1.0	6.62e-3	6.87e-3	3.78
ARPA1	1500	2.0	9.26e-3	9.95e-3	7.45
OCT	120	1.0	1.41e-1	1.43e-1	1.42
OCT	240	1.0	8.74e-3	8.88e-3	1.60
OCT	240	2.0	1.18e-1	1.22e-1	3.39
OCT	480	1.0	3.49e-6	3.69e-6	5.73
OCT	480	2.0	7.58e-4	8.21e-4	8.31
SWIFT	120	1.0	7.67e-4	7.95e-4	3.65
SWIFT	180	2.0	5.81e-4	6.14e-4	5.68
PSS	120	2.0	1.10e-4	1.12e-4	1.82
PSS	160	3.0	4.06e-5	4.42e-5	8.86

We have presented results for two special schemes: (1) every O-D pair is assigned a virtual path, and (2) every physical link is considered as a virtual path. As predicted and also shown in Tables 1 and 2, the first scheme incurs higher blocking probability due to the lower degree of channel sharing among O-D pairs. Given the link capacity and traffic requirement, if the blocking probability derived by the algorithms given in Section 5 is less than the specified constraint, it will be a preferable scheme because of the simplification in the call set-up process. On the other hand, if the computed blocking rate is not acceptable, a hybrid scheme as formulated by Formulation 1 should be used to reduce the blocking probability with minimally incurred extra delay in call set-up.

Effects of limiting the maximum number of physical links for each path - As discussed in Section 2, we use the concept of effective capacity to guarantee that the rate of traffic loss at each physical link be below a certain threshold. Suppose the maximum number of physical links used by each O-D pair is M and the threshold of the

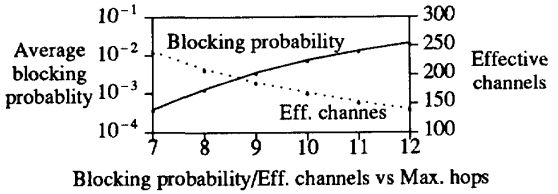
Table 2 - Experimental results for Formulation 3.			
Network ID	Link capacity	Traffic requirement/ OD pair	Average # of hops
ARPA2	240	1.0	3.42
ARPA2	320	2.0	3.42
ARPA2	480	3.0	3.42
ARPA1	1000	1.0	5.55
ARPA1	1500	2.0	5.55
OCT	120	1.0	4.20
OCT	240	1.0	4.20
OCT	240	2.0	4.20
OCT	480	1.0	4.20
OCT	480	2.0	4.20
SWIFT	120	1.0	2.31
SWIFT	180	2.0	2.31
PSS	120	2.0	2.05
PSS	160	3.0	2.05

end-to-end cell loss probability for each O-D pair is D. It can easily be shown that, if the cell loss probability at each link is no greater than $1 - (1 - D)^{1/M}$, then the end-to-end cell loss probability should be below D. Here we assume the cell loss probability for an O-D pair is evenly distributed to M links. This cell loss probability requirement can then be converted into the effective capacity of a physical link. The larger the value of M, the smaller the effective capacity of a physical link. Fig. 2 shows the effects of limiting the maximum number of physical links used by each O-D pair for network OCT. The dotted line shows the changes of the effective channels as a function of the number of maximum hops. For network OCT, the minimum number of physical links needed to have each O-D pair connected (i.e. the diameter of the network) is 8. When the value of M increases, there will be more alternative paths for each O-D pair. One effect of this is that it gives more flexibility in routing and, thus, tends to reduce the blocking probability for each O-D pair. Another effect of increasing M is the reduction of the effective capacities of the physical links as explained above. This effect tends to increase the blocking probabilities of the O-D pairs. Fig. 2 shows that the blocking probability (computed using algorithms given in Section 5) increases when the value of M increases. This result implies that the effect of the increase of routing flexibility is overweighted by the effect of the effective capacity reduction. Therefore, we should simply use the diameter of the network as the limit of the physical links for each O-D pair. Fig. 3 shows the results for network ARPA2. Similar effects are observed.



Blocking probability/Eff. channels vs Max. hops

Fig. 2: The effects of limiting the maximum hops for network OCT.



Blocking probability/Eff. channels vs Max. hops

Fig. 3: The effects of limiting the maximum hops for network ARPA2.

7. Summary and Conclusions

In this paper, we address the problem of assigning virtual paths and determining virtual circuit (call) routing assignments for each pair of communicating nodes in an ATM network. The performance measure considered is the call setup delay and blocking probability. The problem is formulated as a nonlinear nondifferentiable combinatorial optimization problem where the objective is to minimize the expected call blocking rate subject to call setup time constraints, or, alternately, the objective is to minimize the call setup delay subject to the expected call blocking rate constraints.

The formulated problem is not a convex programming problem and, thus, cannot be easily solved using existing mathematical techniques. Therefore, we simplify the formulation for two different assumptions: (1) The network capacity is abundant (which is the nature of ATM networks). In this case, we assume each O-D pair has a virtual path and each call uses exactly one virtual path in routing. We provide a mathematical formulation, along with several solution procedures, to determine the bandwidth and route for each virtual path to minimize the blocking rate. If the resulting blocking rate is within a specified limit, the solution computed for the simplified problem will also be a near-optimal solution to the original, general problem. (2) If the network capacity is very limited and capacity sharing is highly desirable, the blocking rate will be too high and not acceptable if we use the scheme of (1). In this case, we assume every physical link is a virtual path. This tends to provide a high degree of channel sharing and hence a low total call blocking rate. We devise a convex programming formulation, which can be solved by the Frank-Wolfe method, for minimizing the blocking rate for this case.

We are continuing to research methods for the hybrid scheme formulated in Problem (IP1). We are also extending the formulations for multiple classes of traffic.

REFERENCES

1. K.-I. Sato and I. Tokizawa, "Flexible Asynchronous Transfer Mode Networks Utilizing Virtual Paths," *GLOBECOM'90*, pp. 831-838.
2. J. L. Adams, "The Virtual Path Identifier and Its Applications for Routing and Priority of Connectionless and Connection-Oriented Services," *Int'l Journal of Digital and Analog Cabled Systems* **1**, pp. 257-262 (1988).
3. S. Ohta and K.-I. Sato, "Dynamic Bandwidth Control of the Virtual Path in an Asynchronous Transfer Mode Network," *IEEE Trans. on Communications* **40-7**, pp. 1239-1247 (July 1992).
4. S. Gupta, K. W. Ross, and M. E. Zarki, "Routing in Virtual Path Based ATM Networks," *GLOBECOM'92*, pp. 571-575 (Dec. 1992).
5. L. Kleinrock, "Queueing Systems," *Wiley-Interscience*, New York, **Volumes 1 and 2** (1975-76).
6. K. R. Krishnan, "The Convexity of Loss Rate in an Erlang Loss System and Sojourn in an Erlang Delay System with Respect to Arrival and Service Rates," *IEEE Trans. on Communications* **38-9**, pp. 1314-1316 (Sept. 1990).
7. E.J. Messerli, "Proof of a Convexity Property of the Erlang B Formula," *Bell System Technical Journal* **51-4**, pp. 951-953 (1972).
8. D. Bertsekas and R. Gallager, "Data Networks," *Prentice Hall*, 2nd Edition (1992).
9. F.Y.S Lin and J.R. Yee , "A New Multiplier Adjustment Procedure for the Distributed Computation of Routing Assignments in Virtual Circuit Data Networks," *ORSA Journal on Computing, Summer Issue* (1992.).