On the Joint Virtual Path Assignment and Virtual Circuit Routing Problem in ATM Networks

Kwang-Ting Cheng
Dept. of ECE
University of California
Santa Barbara, CA 93106

Frank Yeong-Sung Lin
Bellcore
444 Hoes Lane
Piscataway, NJ 08854

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 for most cases. In this paper, we consider the following problem: given a network topology, link capacity of each physical link, cell-level performance objectives and traffic requirement of each origin-destination pair, we want to jointly determine the following four design variables: (1) the node pairs 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. The problem is formulated as a nonlinear nondifferentiable combinatorial optimization problem and an efficient two-phase solution procedure is proposed. Experimental results are presented to show the trade-offs between throughput and call set-up time thresholds.

1. Introduction

Broadband ISDN (BISDN) has been intensively studied and would dominate data networking in the future due to its identified capability to service a wide variety of traffic such as video, voice and data. Among different transport techniques proposed to implement BISDN, Asynchronous Transfer Mode (ATM) is considered to be the most promising one and has been standardized by CCITT due to its efficiency and flexibility.

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, and therefore 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, routing tables in transit nodes do not need to be updated. Routing procedures are also avoided at 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.

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

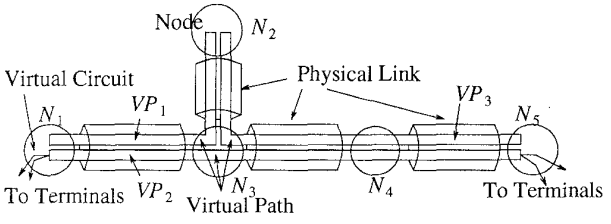

Fig. 1: Schematic illustration of virtual paths

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 step size 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 multi-hop (on the VP level) environment and the effect of statistical multiplexing 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. In [5], for the first time, the virtual path assignment problem and the virtual circuit routing assignment problem are jointly considered. A general problem formulation is presented in [5] and two special cases of the general problem are considered. Algorithms for solving the two special cases are proposed and implemented.

In this paper, we address and propose a solution procedure to the joint (general) problem of assigning virtual paths and determining virtual circuit (call) routing assignments for ATM networks which was not solved in [5]. Given a network topology, link capacity of each physical link, cell-level performance objectives and traffic requirement of each origin-destination (O-D) pair, we want to jointly determine the following design variables: (1) the node pairs 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.

The routing policy considered in this paper is described below.

- Each virtual path is considered as a logical link. The virtual paths (logical links) that (i) connect the same node 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 certain probability.
- If each macro logical link on the path has sufficient free capacity to

0-7803-1820-X/94 $4.00 © 1994 IEEE

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 and impose physical-level hop constraints to satisfy cell-level performance objectives (e.g. end-to-end cell loss probabilities and end-to-end cell delays shall be within given thresholds). For mean and percentile-type end-to-end delay objectives, allocation schemes are discussed and compared in [6]. Similar approaches can be used to allocate end-to-end cell loss objectives. For each physical link l, the allocated requirement (the most stringent one) is converted into an effective capacity C_l. It is clear that 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 this principle, the quality of service (QOS) for each session then can 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. 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 sharing common physical link(s) with the new virtual circuit.

The joint virtual path assignment and circuit routing problem is formulated as a combinatorial optimization problem where the objective is to minimize the total blocking rate (to maximize the total throughput) subject to call set-up time constraints (the number of virtual paths that a virtual circuit traverse shall not exceed a given value).

A two-phase algorithm is proposed which iteratively improves the virtual circuit routing and the virtual path assignment. In computational experiments, we attempt to find all Pareto optimum solutions by applying the proposed algorithm sequentially with different call set-up constraints. In a special case where each virtual circuit is allowed to use exactly one virtual path, theoretical lower bounds on the optimal objective function value can be obtained by the solution approach proposed in [5]. It is shown by computational experiments that the algorithm proposed in this paper provides satisfactory results in this special case when compared with the theoretical lower bounds.

The remainder of this paper is organized as follows. In Section 2, a problem formulation is presented. In Section 3, a solution procedure is proposed. In Section 4, 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 $G_v(N,V)$. (Throughout this paper, virtual paths and logical links are used interchangeably.) 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, calls 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. In addition, to satisfy the call set-up time constraints, each path in Q_w should involve no greater than K (a given number) virtual paths. 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 each 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}, which is referred to as a type-k macro logical link. 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 [7]:

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

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:

$$g_{ijk} = \sum_{q \in Q_w} \sum_{w \in W} \delta_{q,ijk}\, y_q \prod_{m_{abd} \in H_{q,ijk}} (1 - B_{abd})$$
$$\forall\ i,j \in N,\ i \neq j,\ k \in \{1,2,...,n-1\} \qquad (2.2)$$

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

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

$$\sum_{v \in V_{ijk}} c_v = v_{ijk} \quad \forall\ i,j \in N,\ i \neq j,\ k \in \{1,2,...,n-1\} \qquad (2.5)$$

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

$$c_v \text{ is a nonnegative integer } \forall\ v \in V. \qquad (2.7)$$

The objective function is to minimize the total virtual circuit blocking rate (to maximize the total throughput). 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. Constraint (2.7) requires that the number of channels allocated to each virtual path be a nonnegative integer. Note that the physical level and the virtual path level hop constraints are enforced by properly choosing $\{Q_w\}$. It may be impractical/intractable to explicitly manage the set Q_w. In Section 4, a mechanism to implicitly consider all possible simple paths satisfying the two sets of hop constraints (physical level and virtual path level) is discussed.

Below we briefly discuss the complexity of (IP1). Due to Constraint (2.7), (IP1) is an integer programming problem, which is typically hard to solve exactly. In addition, B_{ijk} is a discrete and highly nonlinear function. One may use linear interpolation to make B_{ijk} continuous and convex with respect to v_{ijk}. More precisely, for all nonnegative integer i, points $(i,B_{ijk}(i))$ and $(i+1,B_{ijk}(i+1))$ are connected by a segment. From [8] this new (continuous) blocking function is 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}. From an inspection of the objective function and Constraint (2.2), it is clear that (IP1) is not a convex programming problem without considering the integrality constraints.

In a previous study [5], we consider two special cases of the general problem. Under the circumstances where the network capacity is abundant (the nature of ATM networks), we consider to limit the number of hops (on the virtual path level) for a virtual circuit at the cost of throughput (the degree of virtual path sharing among O-D pairs is reduced, which tends to increase the total blocking rate). This special case is to make each virtual circuit use exactly one virtual path. 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. A formulation for this virtual path assignment and "single-hop" virtual circuit routing assignment problem is provided in [5]. As compared to (IP1), this problem ($K = 1$) formulation is much simpler and possesses convexity which facilitates efficient and effective solution procedures.

In [5] we also consider another special case of the general problem where (i) the hop constraint on the VP level is assumed to be inactive (one may consider K to be infinity in this case) and (ii) each VP involves exactly one physical link (only type-1 logical links are allowed). In this special case, the total call blocking rate is minimized without considering the call set-up delay constraints. In addition, VP assignments are fixed. 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.

Under the circumstances where neither scheme is considered satisfactory, the general problem formulation shall be considered. We first construct a surrogate problem formulation to (IP1) in the next section so that the degree of nonconvexity of (IP1) is reduced, which has an effect of simplifying the solution procedures as will be discussed in Section 4. In addition, the optimal objective function value of the surrogate problem is an upper bound on the optimal objective function value of (IP1).

3. A Simplified Formulation

Two approximations are made to reduce the degree of nonconvexity of Problem (IP1). 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 macro logical link is small.

The first approximation is to replace the end-to-end call blocking probability $1 - \prod_{m_{ijk} \in H_q} (1 - B_{ijk})$ for each path q by $\sum_{m_{ijk} \in H_q} B_{ijk}$. It can easily be shown that $\sum_{m_{ijk} \in H_q} B_{ijk}$ is an upper bound on $1 - \prod_{m_{ijk} \in H_q} (1 - B_{ijk})$. In addition, if each B_{ijk} is small, the bound is tight.

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

Combining the above two approximations, the total call blocking rate can be expressed as $\sum_{i,j \in N, i \neq j} \sum_{k=1}^{n-1} g_{ijk} B_{ijk}$, which, as discussed above, is an upper bound on the exact total call blocking rate as given in (IP1). From [9], $g_{ijk} B_{ijk}$ is a monotonically increasing and convex function of g_{ijk}.

A surrogate formulation of (IP1) with the aforementioned approximations is given below.

$$Z_{IP2} = \min \sum_{i,j \in N, i \neq j} \sum_{k=1}^{n-1} g_{ijk} B_{ijk} \qquad \text{(IP2)}$$

subject to:

$$g_{ijk} = \sum_{q \in Q_w} \sum_{w \in W} \delta_{q,ijk} y_q \qquad \forall\ i,j \in N,\ i \neq j,\ k \in \{1,2,\ldots,n-1\} \qquad (3.1)$$

$$\sum_{q \in Q_w} y_q = \lambda_w \quad \forall\ w \in W \qquad (3.2)$$

$$\sum_{v \in V_l} c_v \leq C_l \quad \forall\ l \in L \qquad (3.3)$$

$$\sum_{v \in V_{ijk}} c_v = v_{ijk} \quad \forall\ i,j \in N,\ i \neq j,\ k \in \{1,2,\ldots,n-1\} \qquad (3.4)$$

$$y_q \geq 0 \quad \forall\ q \in Q_w,\ w \in W \qquad (3.5)$$

$$c_v \text{ is a nonnegative integer } \forall\ v \in V. \qquad (3.6)$$

The performance measure is an approximate expression (upper bound) of the total call blocking rate. The right hand side of Constraint (3.1) is the aggregate flow on link l (ignoring call losses in upstream nodes). Constraints (3.2) to (3.6) are the same as Constraints (2.3) to (2.7).

It can be verified that, without considering the integrality constraints, Problem (IP2) is convex with respect to either $\{g_{ijk}\}$ or $\{v_{ijk}\}$. This property is helpful in the algorithm development as will be discussed in the next section. Unfortunately, (IP2) is not jointly convex with respect to $\{g_{ijk}\}$ and $\{v_{ijk}\}$. The following example shows this claim.

Example: Let $F(g,v)$ be $g\,B(g,v)$. Consider the following three points $F(0,0)$, $F(a,1)$ and $F(2a,2)$. Then,

$$\frac{1}{2}[F(0,0)+F(2a,2)]-F(a,1) = \frac{1}{2}\left[0+\frac{4a^3}{1+2a+2a^2}\right]-\frac{a^2}{1+a}$$

$$= \frac{-a^2}{(1+a)(1+2a+2a^2)} \leq 0.$$

Therefore, $F(g,v)$ is not jointly convex with respect to g and v.

4. Algorithm

In this section, we propose an algorithm to solve the joint virtual path assignment and virtual circuit routing problem. The algorithm consists of two phases. In the first phase, the capacity of each virtual path is fixed and the routing assignments are adjusted to reduce the overall call blocking rate. In the second phase, the virtual path capacity assignments are adjusted to reduce the overall call blocking rate, using the fixed virtual circuit routing assignments derived in the first phase. We iterate this two-phase process until no improvement to the overall call blocking rate can be made. Figure 2 shows the algorithm in the abstraction level.

```
VP_Assignment_&_VC_Routing()
{
    G_m(N,E) = Initial_capacity_assignment( G_l(N,L) );
    Routing = Initial_routing_assignment(G_m(N,E) );
    while (blocking rate reduction is possible)
    {
        Routing = Routing_adjustment(G_m(N,E));
        G_m(N,E) = Capacity_adjustment(Routing);
    }
}

Initial_capacity_assignment(G_l(N,L) )
{
    Set E to be an empty set;
    For every O-D pair with a direct physical link in G_l
        construct a type-1 macro logical link in G_m;
        set the capacity of the macro logical link to be the
          effective capacity of the corresponding physical link;
    Otherwise
        construct a type-a macro logical link where a is the
          shortest distance in physical hops for the O-D pair;
        set the capacity of the macro logical link to be 0;
}

Initial_routing_assignment(G_m(N,E))
{
    For every O-D pair with a direct physical link
        use the type-1 macro logical link;
    Otherwise
        use the direct (type-a) macro logical link;
}
```

Figure 2: Two-phase algorithm

Initialization:

Consider a graph $G_m(N,E)$ where N is the set of nodes and E is the set of macro logical links. Initially, set E to be an empty set. Then, for every O-D pair with a direct physical link in G_l, which represents the physical network, construct a type-1 logical link with the capacity equal to the effective capacity of the corresponding physical link. Consider this type-1 logical link as a type-1 macro logical link (containing only one logical link) and add it to E. For each O-D pair without a direct physical link in G_l, construct a type-a logical link where a is the shortest distance in terms of the number of physical hops for the O-D pair. The physical route involved for this logical link is the minimum-hop path. (The reason to choose a minimum-hop physical path for each logical link is to satisfy the physical level hop constraint.) Set the capacity of this logical link to 0. Then, consider this type-a logical link as a type-a macro logical link (containing only one logical link) and add it to E. An initial feasible routing assignment is to use the direct (single-hop) macro logical link for each O-D pair.

Phase I:

In this phase, the virtual path assignments are fixed and the virtual circuit routing assignments are adjusted to reduce the overall call blocking rate. Consider (IP2) where the virtual path assignments are fixed. As discussed earlier, this virtual circuit routing problem in Phase I is a convex programming problem. However, we need to consider the following two hop constraints (enforced by properly choosing Q_w): (i) to limit the maximum number of hops in the virtual path level to K for each virtual circuit and (ii) to limit the maximum number of hops in the physical level to M for each virtual circuit. When all simple paths satisfying the above two hop constraints are considered, it is intractable to explicitly manage Q_w. Below are a number of approaches to circumventing this difficulty.

For simplicity, we applied the Frank-Wolfe method [10] to solve the routing problem where Q_w is considered to be the set of all simple paths on the VP level for O-D pair w and the hop constraints are considered explicitly (shown in Equations (4.1) and (4.2)). In the first phase of the Frank-Wolfe algorithm, a minimum first derivative length (MFDL) path is found for each O-D pair under the following two hop constraints:

$$\sum_{q \in Q_w} y_q \sum_{i,j \in N \,\&\, i \neq j} \sum_{k=1}^{n-1} \delta_{q,ijk} \leq K \qquad \forall\ w \in W \qquad (4.1)$$

$$\sum_{q \in Q_w} y_q \sum_{i,j \in N \,\&\, i \neq j} \sum_{k=1}^{n-1} \delta_{q,ijk}\ k \leq M \qquad \forall\ w \in W. \qquad (4.2)$$

Constraint (4.1) (together with the flow conservation constraint and the single (shortest) path constraint) requires that the selected shortest path not involve more than K hops on the VP level. Constraint (4.2) (together with the flow conservation constraint and the single path constraint) requires that the selected shortest path not involve more than M hops on the physical level. If Constraint (4.2) is relaxed, then the hop-constrained shortest path problem can be solved by the Bellman-Ford algorithm [10] where for each node pair with multiple parallel links (different types of macro logical links) the shortest link is used. On the other hand, if Constraint (4.1) is relaxed, the following modification on the graph model (which is also illustrated in Figures 2 and 3) can make the Bellman-Ford algorithm applicable.

Consider the graph $G_m(N,E)$, which has been considered in the initialization phase, where each macro logical link $e \in E$ is associated with a weight w_e, i.e. the first derivative of the blocking rate of the macro logical link with respect to the offered load. Let $type(e)$ be the type of macro logical link e, i.e. the number of physical links that each of the logical link(s) in the macro logical link uses. Then, construct a new (augmented) graph $G_a(N',E')$ by the following changes on $G_m(N,E)$. For each edge e in $G_m(N,E)$, introduce $[type(e) - 1]$ new nodes. Then replace edge e by $type(e)$ new edges which form a simple path connecting the originating and the terminating nodes for edge e in $G_m(N,E)$ and traverses all the new nodes just introduced. The arc weight of the last edge in the path is assigned the weight of e, while the weights of all the other edges in the path are assigned 0. This is illustrated in Figures 2 and 3.

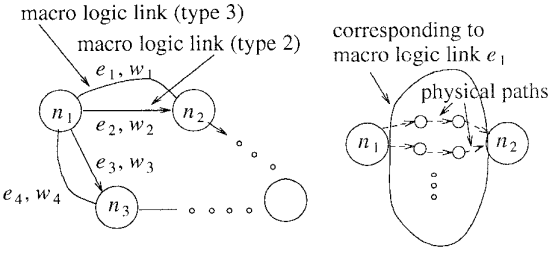

Figure 3: Graph representation for macro logical links and physical paths

Once the new graph is constructed, we find a shortest path for each O-D pair in the new graph which involves no greater than M hops. This can be achieved again by the Bellman-Ford algorithm. If the shortest path found terminates at a node in N' but not in N, then

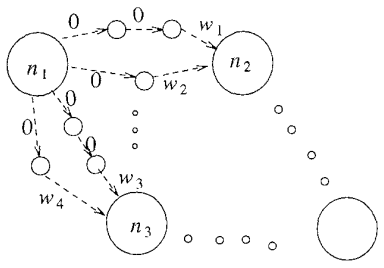

Figure 4: Edge weight assignments in the new graph

backtrack the path until a node in N is reached.

However, when the aforementioned two hop constraints are jointly considered, it is not clear to us that the general constrained shortest path problem can be solved in polynomial time. The following three heuristics are then proposed.

Method I:

1. Relax Constraint (4.2).
2. Solve the shortest path problem with the K-hop constraint on the VP level.
3. If the solution also satisfies the M-hop constraint on the physical level (the problem is optimally solved), add the path to the set of active paths; otherwise, apply Method II.

Method II:

1. Relax Constraint (4.1).
2. Solve the shortest path problem with the M-hop constraint on the physical level.
3. If the solution also satisfies the K-hop constraint on the VP level (the problem is optimally solved), add the path to the set of active paths and return; otherwise, apply Method III.

Method III:

1. Combine Constraints (4.1) and (4.2) as follows:

$$\sum_{q \in Q_w} y_q \sum_{i,j \in N \,\&\, i \neq j} \sum_{k=1}^{n-1} \delta_{q,ijk} (k+1) \leq M+K \quad \forall\, w \in W. \quad (4.3)$$

2. Solve the shortest path problem with the above combined $(M + K)$-hop constraint by introducing one more artificial link with weight 0 to each edge in $G_m(N,E)$.
3. If the solution satisfies both the K-hop constraint on the VP level and the M-hop constraint on the physical level (the problem is optimally solved), add the path to the set of active paths and return; otherwise, find the shortest path in the current set of active paths (each path in this set satisfies the two hop constraints).

Once an efficient routing is found, we move onto the second phase: virtual path capacity adjustment.

Phase II:

In the virtual path capacity adjustment phase, the virtual circuit routing assignments are fixed and the virtual path assignments are adjusted to reduce the total call blocking rate. As discussed in the previous section, when the virtual path assignments are fixed and the integrality constraints are relaxed, (IP2) becomes a nondifferentiable convex programming problem. One possible approach to solving the virtual path assignment problem in Phase II is linear programming relaxation in conjunction with a rounding heuristic. However, in this paper, we propose a greedy heuristic which has a lower computational complexity. More specifically, at each iteration of the heuristic, we attempt to move one channel from a virtual path to another so that the total call blocking rate can be reduced the most. Suppose we move one channel from VP p to VP q. It will decrease the blocking rate of VP p, while increasing the blocking rate of VP p.

There may be many routes (through physical links) for each virtual path. The possible underlining physical paths for each VP were identified and stored (it is done in the preprocessing phase). For a virtual path between an O-D pair that has a direct physical link between them, the underlining physical path is simply a one-hop physical path. For virtual paths between O-D pairs without a direct physical link, we record a number of simple (i.e. no cycles) physical paths as the underlining physical paths. If there are m underlining physical paths for virtual path p and each of them allocates n_i channels for VP p, the capacity of virtual path p will be equal to $\sum_{i=1}^{m} n_i$. To increase the capacity of a virtual path by one channel, every physical link in a underlining path of the virtual path must allocate one extra channel for this VP. This added channel must be taken from a number of other virtual paths and thus increase the blocking rates for those virtual paths using these links as underlining physical links. For each virtual path p and for each of its underlining physical path q, we compute the changes of overall blocking rate if one channel is added to p and one channel is deducted from every type-1 VP that lies on physical path q. We then choose the one that causes the maximum blocking rate reduction and adjust the corresponding virtual path capacities accordingly. If there exists no such adjustment that can reduce the overall blocking rate, the algorithm stops and the final virtual path capacities will be the solution.

Once the capacity adjustment is made, we re-adjust the routing under the new capacity assignment by calling the modified Frank-Wolfe algorithm again. The process iterates until no blocking rate reduction is possible.

This is a greedy algorithm. At each iteration, the algorithm finds the best way (in terms of overall blocking rate reduction) to add one channel to a virtual path. It is therefore possible that the final solution is a local minimum. To get some idea of how good the algorithm is, we compare a theoretical lower bound on the overall blocking rate with the one derived by our algorithm for $K = 1$. In [5], a mathematical formulation is given for $K = 1$. Mathematical programming techniques were used to compute lower and upper bounds on the overall blocking rate. Comparisons will be made in the next later section.

5. Experimental Results

We have implemented the algorithm in the C programming language and run the program in a SUN Sparc 10 workstation. We tested the algorithm for 2 networks -- OCT and ARPA2 with 26 and 21 nodes, respectively. The topologies of both networks can be found in In our first experiment, we attempt to show the trade-offs between the overall blocking rate and K, the hop limit in routing. In general, the larger the value of K, the more routing flexibility, the lower overall blocking rate and the longer call set-up time. Figure 5 shows the trade-off for network OCT. The traffic requirement between every O-D pair is assumed 1 erlang. The capacity is assumed 120 channels for each physical link. The diameter of this network is 8. Figure 5 shows the overall blocking rate for K=1 to 8. In the figure we also show the number of O-D pairs with direct virtual paths. Figure 6 shows the results for network ARPA2. Again, the traffic requirement between every O-D pair is assumed 1 erlang. The capacity is assumed 120 channels for each physical link. The diameter of this network is 7.

The capability of exploiting the trade-off between the overall call blocking rate and K enables the application to several scenarios. First, if there are hard constraints on both K and overall call blocking rate to a specific ATM network, this trade-off curves can show all Pareto optimum solutions for selection. Or the algorithm will report that no feasible solution can be obtained. The trade-off curve will illustrate the sensitivity of either constraint. Second, if there is a hard constraint on blocking rate, from the curve we can obtain the smallest K that satisfies the overall call blocking rate constraint to minimize the call set-up delay.

To illustrate the efficiency of the algorithm given in Section 4,

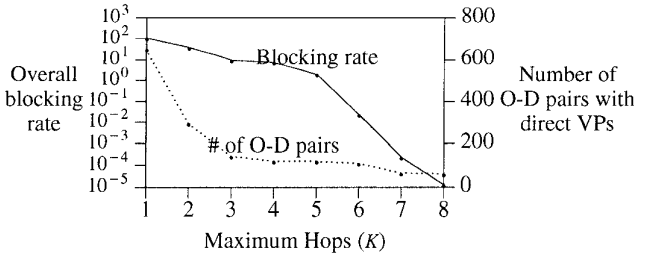

Figure 5: Results of OCT (overall blocking rate/number of O-D pairs with direct VPs vs. max. hops)

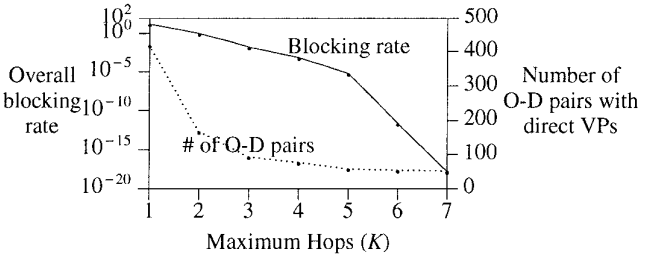

Figure 6: Results of ARPA2 (overall blocking rate/number of O-D pairs with direct vs. max. hops)

Table 1 compares the blocking rate for $K = 1$ derived by our algorithm with a theoretical lower bound. The theoretical lower bound is obtained using the mathematical programming techniques described in [5]. Our algorithm produces satisfactory results. This comparison shows that the greedy heuristic works well for the test cases.

Table 1 - Summary of results when $K = 1$

network	traffic requirement per OD pair	physical link capacity	overall blocking rate	
			our algorithm	lower bound
OCT	1.0	120	104.5	91.7
ARPA2	1.0	120	18.1	15.8

It should be pointed out that the problem formulation considered in [5] for the case where each virtual circuit uses exactly one virtual path is not truly a special case of (IP2) with $K = 1$. Indeed, the formulation considered in [5] has a larger feasible region than that of Problem (IP2) with $K = 1$. This is because in the formulation considered in [5] all types of macro logical links satisfying the M-hop constraint on the physical level are jointly considered when routing decisions are made. Whereas, in Problem (IP2) with $K = 1$ macro logical links involving different numbers of physical links are separately considered. As such, lower bounds obtained by the procedure provided in [5] may be loose in some cases. Therefore, lower bounds obtained by the procedure provided in [5] may potentially be loose.

6. 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 measures considered are the call setup delay and the overall call blocking rate. 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.

The formulated problem is difficult to solve due to nonconvexity and the combinatorial nature. We then construct a surrogate (restricted) problem formulation so that the degree of nonconvexity is reduced. An efficient two-phase algorithm is proposed to solve this restricted problem.

In the first phase of the proposed algorithm, the capacity of each virtual path is fixed and the routing assignments are adjusted to reduce the overall call blocking rate. In the second phase, the virtual path capacity assignments are adjusted to reduce the overall blocking rate, using the fixed virtual circuit routing assignments derived in the first phase. We iterate this two-phase process until no improvement to the overall call blocking rate can be made.

Two sets of computational experiments are performed to investigate the efficiency and effectiveness of the proposed algorithm. The first set of experimental results shows the trade-offs between throughput and call set-up time thresholds. The second set of results shows that our proposed algorithm achieves satisfactory results when compared with legitimate lower bounds.

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 (December 1992).
5. F. Y.S. Lin and K.T. Cheng, "Virtual Path Assignment and Virtual Circuit Routing in ATM Networks," *Proc. GLOBECOM'93* (November 1993).
6. F. Y.S. Lin, "Allocation for End-to-end Delay Objectives for Networks Supportinh SMDS," *in Proc. GLOBECOM'93* (November 1993).
7. L. Kleinrock, "Queueing Systems," *Wiley-Interscience*, New York, **Volumes 1 and 2** (1975-76).
8. E.J. Messerli, "Proof of a Convexity Property of the Erlang B Formula," *Bell System Technical Journal* **51-4**, pp. 951-953 (1972).
9. 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 (September 1990).
10. D. Bertsekas, D. Bertsekas, and R. Gallager, "Data Networks," *Prentice Hall*, 2nd Edition (1992).
11. 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).
12. M.-J. Lee and J. R. Yee, "Optimal Minimax Routing in ATM Networks," *Proc. GLOBECOM'90* (December 1990).
13. F. Y.S. Lin and J. R. Yee, "A Real-time Distributed Routing and Admission Control Algorithm for ATM Networks," *INFOCOM'93*, pp. 792-801 (March 1993).