

# MULTICAST ROUTING ALGORITHMS FOR 3-STAGE CLOS ATM SWITCHING NETWORKS

*Soung C. Liew*

Bellcore  
445 South Street  
Morristown, NJ 07962-1910, U.S.A.

**ABSTRACT** An approach to building a large ATM switch is to simply set up a regularly-structured network in which smaller switch modules are interconnected. Routing is an issue if there are multiple paths from any input to any output in such a network. We focus on the 3-stage Clos network. An optimal and a heuristic algorithms have been designed and tested. Our results show that the heuristic algorithm can find multicast routes that are close to optimal within a response time that is significantly lower than that of the optimal algorithm. Further analysis of the experimental data suggests a hybrid implementation in which the optimal and heuristic algorithms are run in parallel with a set time limit. The algorithms and the discussion here also apply to other networks, including wide-area communications networks, with a two-hop structure.

## I. Introduction

Asynchronous Transfer Mode (ATM) has emerged as a very promising transport technique for supporting services of diverse bit-rate and performance requirements in future broadband networks [1]. High-speed packet switches are essential elements for successful implementation of ATM networks. If a significant population of network users are potential broadband-service subscribers, high-capacity packet switches with a large number of input and output ports will be indispensable.

Two basic philosophies in large packet switch designs emerge as a result of recent research activities. Both approaches concentrate on scalable designs that construct a large switch using smaller switch modules. The first approach strives to avoid internal buffering of packets in order to simplify traffic management. Examples in this category are the Modular switch [2], the generalized Knockout switch [3], and the 3-stage generalized dilated-banyan switch [4] (with no buffering at the center stage). The second approach attempts to build a large switch by simply interconnecting switch modules as nodes in a regularly-structured network, with each switch module having its own buffer for temporary storage of packets. A notable example in this category is the proposal by [5, 6, 7] (Fig. 1(a)) in which output-buffered switch modules are connected together as in the 3-stage Clos circuit-switch architectures [8]. Typically, a packet must pass through several queues before reaching its desired output in these switch architectures.

Because of the simplicity of switches in the second category, they have been the focus of several potential switch vendors [5, 6, 7]. However, these switches necessitate more complicated network control mechanisms, since more queues must then be managed. In addition, for the Clos architecture, routing within the switching network becomes an issue because there are multiple paths from any input port to any output port (see Fig. 1(b)). Things become even more complicated if multicast (point-to-multipoint) connections, an important class of future broadband services, are to be

supported. For communications networks that use these switching networks for switching in their nodes, each node should be treated as a "micronetwork" rather than an abstract entity with queues at the output links only [9], as is done traditionally.

An open question is to what extent the internal buffers in the micronetwork would complicate traffic management and whether routing algorithms for call setups would require unacceptably long execution times. To answer this question, this paper investigates the general problem of multicast routing in the 3-stage Clos switching network, with point-to-point routing as a special case. We will assume all switch modules in the micronetwork to have multicast capability. Before proceeding further, it is worth comparing our multicast routing problem with the multicast routing problem in a general network [10]. Three features associated with routing in the Clos network come into mind immediately:

1. Necessity for a very fast setup algorithm;
2. Large numbers of switch modules and links;
3. Regularity and symmetry of the network topology.

It is necessary to have an algorithm that is faster and more efficient than those used in a general network because the Clos switching network is only a subnetwork within an overall communications network. From the viewpoint of the overall network, the algorithm performed at each Clos switching network is only part of the whole routing algorithm. Adding to the complexity is the highly connected structure of the Clos network, which dictates the examination of a large number of different routing alternatives. In fact, the Clos network is stage-wise fully connected in that each switch module is connected to all other switch modules at the adjacent stage. As an example, for a modest Clos network with 1024 input and output ports made of  $32 \text{ inputs} \times 32 \text{ outputs}$  switch modules (with  $n = 32, m = 32, p = 32$  in Fig. 1(a)), the numbers of nodes and links are 96 and 3072, respectively. Thus, algorithms tailored for a general network [10] are likely to run longer than the allotted call-setup time. Both features 1 and 2 above argue for the need for a more efficient algorithm, and feature 3, regularity of the network topology, may lend itself to such an algorithm. To address these issues, this paper investigates the extent to which specialized algorithms can expedite the call setup process.

The rest of this paper is organized as follows. Section II provides some background materials and discusses our specific problem formulation and definition with respect to other possibilities. Section III presents our designs of an optimal and a heuristic algorithms. Section IV discusses computation results which show that the heuristic algorithm can have much faster response time than the optimal algorithm while achieving near-optimal routing. Implications of our results for actual real-time implementation of the routing schemes in switching networks are also discussed. Finally, we summarize the main results and conclusions of this work

## 45.3.1

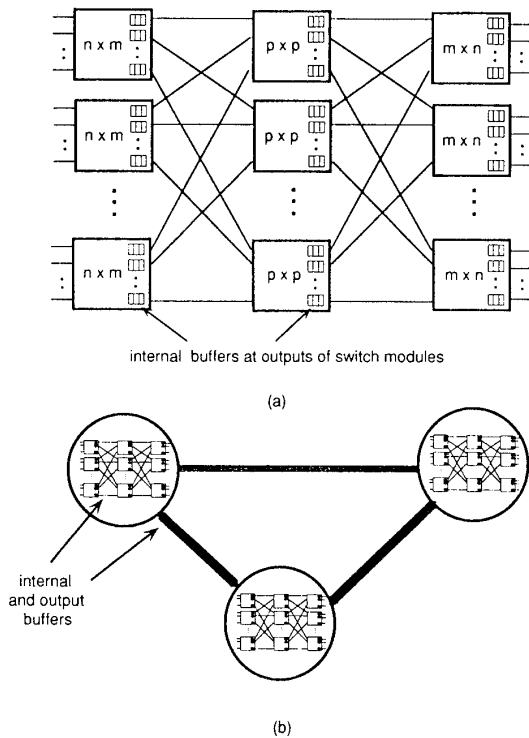


Figure 1: (a) 3-stage Clos switching network; (b) Clos switches as micronetworks within communications network.

in Section V.

## II. Background and Assumptions

To put things in the proper context, we now discuss some issues relevant to the Clos network. Melen and Turner derived in their recent work [11] the relationship between various switch parameters that guarantee an ATM Clos network to be *nonblocking*. In the ATM setting, each input and output link in a switch contains traffic originating from different connections with varying bandwidth requirements. An ATM switch is said to be nonblocking if a connection request can find a path from its input to its targeted output as long as the bandwidth required by the connection does not exceed the remaining bandwidths on both the input and output. What was not addressed in [11] is the issue of routing. Even though the switch used may be nonblocking as defined, a connection may still suffer unacceptable performance in terms of delay and packet loss if the wrong path is chosen. This is due to contention among packets for common routes in the ATM setting where packet arrivals on different inputs are not coordinated. Consequently, regardless of whether the switch is nonblocking, some routes will be preferable because they are less congested. The choice of routes is the focus of this paper.

Routing in any network of switch modules can be posed as a graph problem in which the switch modules correspond to nodes and the links correspond to directed arcs [9] in the graph. A weight is assigned to each arc, and its value corresponds to the congestion level on the associated link. For instance, the weight assigned could be traffic load, packet mean delay, packet loss rate, mean buffer occupancy, or other traffic measures. Alternatively, it could be a weighted function of all these parameters. In either case, the weight of an arc corresponds to the "undesirability" of choosing the arc as part of the overall route. One may argue that more than one parameter is needed to capture the traffic characteristics on each link. Although such "multiobjective optimization" problem is beyond the scope of this paper, our treatment here provides a basis for extension along this line.

This paper also assumes that the undesirability of a route is the sum of all the weights of the arcs in the route. For instance, if the weights are taken to be the mean delays of the links, this approach aims to minimize the mean delay of the overall route. As far as point-to-point connections are concerned, the routing problem in this formulation becomes a shortest-path routing problem [9]. It is well known that there are good algorithms that can solve this problem within a short time [9, 12].

The situation is not as clear-cut in multicast routing, which involves multiple paths from one source to several destinations. If we aim to optimize the local performance or grade-of-service perceived by each path, then the shortest-path formulation is still valid, simply because this approach assigns the least congested path to each input-output pair. On the other hand, if we aim to minimize the global congestion level (e.g., the total buffer occupancies of all queues) of the overall switching network, then we are faced with a Steiner-tree problem [10, 13], in which the sum-total of the weights of all the arcs in the multicast connection is to be optimized.

The global viewpoint has the advantage that it can accommodate more connection requests and that it reserves more capacity for future connection requests. So, this paper will focus on this approach. Unfortunately, the general Steiner tree problem is a hard problem without a known fast algorithm [10, 13]. It can be shown that the two-hop structure of the Clos Network allows us to pose the multicast routing problem as a warehouse location problem [14]. Although this problem is simpler than the Steiner-tree problem, it is still a hard problem if one aims for the optimal solution. Therefore, a heuristic algorithm that finds a close-to-optimal solution within a short time is desirable. The next section considers optimal as well as heuristic routing algorithms.

## III. Multicast Routing Algorithms for Clos Networks

Let the three sets of nodes in stage 1, stage 2, and stage 3 of the Clos network be  $I$ ,  $J$ , and  $K$ , respectively. Furthermore, denote the weight of the arc from node  $i \in I$  to node  $j \in J$  by  $c_{ij}$ , and that from node  $j \in J$  to node  $k \in K$  by  $d_{jk}$ . Suppose that we want to multicast from input link  $p$  of switch module  $i$  to the set of output links  $Q$  (see Fig. 2) of switch modules  $K'$ . Then, the problem is basically to select a set of second-stage switch modules  $J'$  to be included in the multicast tree. If we only knew the second-stage nodes  $J'$  that are used in the optimal multicast tree, then the links in the tree can be easily found using the minimal-link selection process below:

### *Minimal-link Selection Process based on Node Set $J'$*

1. Certainly, the links from node  $i$  to all  $j \in J'$  will be included.
2. In addition to these links, for each third-stage node  $k \in K'$ ,

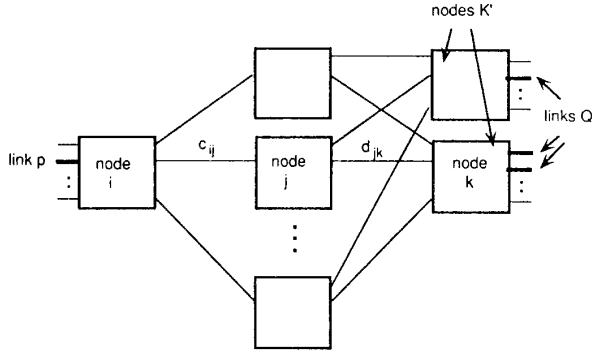


Figure 2: Multicast connection to be considered.

we choose the minimal link  $(j_k, k)$  for connection from from the second stage to node  $k$ ; i.e., the second-stage node  $j_k$  chosen for the connection is such that  $j_k \in J'$  and  $d_{j_k k} \leq d_{jk}$  for all  $j \in J'$ .

The problem, of course, is that we do not know the second-stage nodes used in the optimal multicast tree, and finding them is not so easy.

Given a subset of second-stage switch modules that is proposed for use as intermediate nodes, not necessarily those used in the optimal solution, we can easily compute the best solution based on that proposal using the minimal-link selection process described above. If number of intermediate nodes,  $m \leq |K'|$ , then there are  $2^m - 1$  possible proposals, ranging from those with only one intermediate module to that with all  $m$  intermediate modules. If  $m > |K'|$ , there are  $\sum_{i=1}^{|K'|} \binom{m}{i}$  possible proposals, ranging from those with one intermediate module to those with  $|K'|$  intermediate modules; proposals with more than  $|K'|$  intermediate modules need not be considered because at most  $|K'|$  intermediate modules will be used in any multicast tree. Thus, there are  $\min(2^m - 1, \sum_{i=1}^{|K'|} \binom{m}{i})$  possible proposals. A brute-force method for the overall algorithm is to go through all the proposals, calculate the best solution associated with each proposal, and choose the one with the lowest cost. With this exhaustive enumeration method, the run time of the algorithm grows exponentially with  $m$ . Assuming  $m \leq |K'|$  and a modest  $m$  value of 32, for instance, there are more than four billion proposals that must be considered! Fortunately, there are ways to eliminate some of the non-optimal alternatives without computing their solutions. Reference [14] provides one such algorithm. We believe the algorithm presented in this paper is more efficient because it can eliminate more non-optimal alternatives at the outset.

To understand our algorithm, for simplicity, let us for the time being consider all the  $2^m$  subsets of intermediate nodes and attempt to devise a method for enumerating the proposals. The fact that some of the proposals need not be considered will be taken into account later to further improve the algorithm. Figure 3 shows one possible enumeration scheme depicted as a tree in which the leaf nodes on the right are the  $2^m$  alternative proposals. Each node in the enumeration tree, whether it is a leaf node or not, is rep-

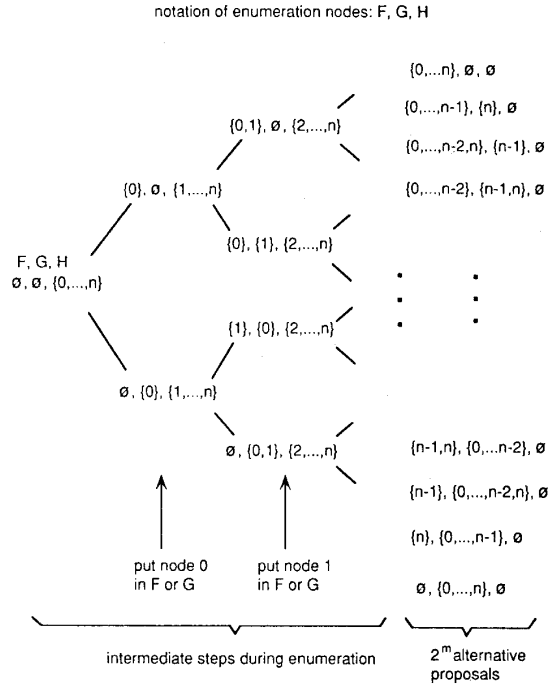


Figure 3: Enumeration tree for listing all alternative solutions.

resented by three disjoint subsets of intermediate switch modules,  $F$ ,  $G$ , and  $H$ . Here,  $F$  denote the proposed second-stage switch modules,  $G$  denote the excluded switch modules, and  $H$  denote the switch modules that have neither been proposed nor excluded so far in the enumeration process. The enumeration process starts with the root node on the left with all modules being in  $H$  originally. At each node of the enumeration tree, a new module is taken from  $H$ , and the tree branches off in two directions with the chosen module being assigned to  $F$  and  $G$ , respectively. After  $m$  levels of branching, we end up with each module either being assigned to  $F$  or  $G$  for the  $2^m$  leaf nodes, thus completing the enumeration process.

The basis of our multicast routing algorithm is as follows: if we can determine during the enumeration process that the best solution given by the leaf nodes of one branch is inferior to the solution given by some leaf node of the other branch, then we need to branch in the latter direction only, since the formal direction will not yield the optimal solution anyway. This can potentially save a lot of computation. In the following, a theorem is adapted from [14] for such a trimming process.

Let  $C(F)$  be the cost of the particular solution with node set  $F$  being the proposed intermediate nodes. Specifically,

$$C(F) = \sum_{j \in F} c_{ij} + \sum_{k \in K'} d_{j_k k} \quad (1)$$

where  $j_k = \arg \min_{j \in F} d_{jk}$  can be found by the minimal-link selection process described above (i.e., node  $k$  in stage 3 will be connected to node  $j_k$  in stage 2 via the link that has the smallest

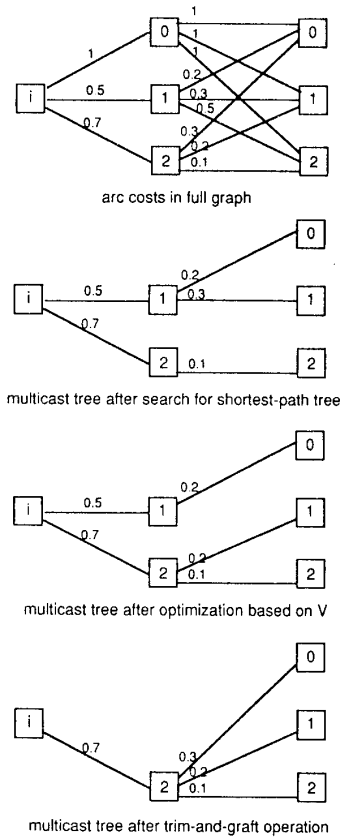


Figure 4: Example for illustration of heuristic algorithm.

The original cost associated with the subtree of node  $v$  is  $c_{iv} + \sum_{w \in W_v} d_{vw}$ , and the cost associated with attaching the nodes in  $W_v$  to other nodes in  $V$  is  $\sum_{w \in W_v} \min_{j \in V - \{v\}} d_{jw}$ . If  $c_{iv} + \sum_{w \in W_v} d_{vw} - \sum_{w \in W_v} \min_{j \in V - \{v\}} d_{jw} > 0$ , then saving can be achieved; remove  $v$  from  $V$  and attach nodes in  $W_v$  to the other nodes in  $V$ .

The 3-step augmentation scheme can be very good if the shortest-path solution in the first step already yields very good results, or the intermediate modules used in the shortest-path solution overlap substantially with those used in the optimal solution.

#### IV. Computation Results

The optimal and heuristic algorithms have been coded in C language and implemented on a SPARC 2 work station, a RISC (reduced-instruction set computing) machine with 28.5 MIPS (million instructions per second) processing power. We will assume in our discussion that a response time of no more than 0.1 second is required. From the viewpoint of the end-users, a call setup time of less than a few seconds is probably desirable. Since our algorithm resides in only one switching node, and is one of the many functions that must be performed by the overall network, it is a sound engineering practice to have a more stringent requirement on the

run time. A more conservative approach is also necessary to compensate for the communication overhead between different layers of functionalities and the computation of other network algorithms that share the same computing resources.

To test the algorithms, we have conducted experiments in which we considered the problem of multicasting from node  $i$  in stage 1 to  $d$  ( $d \leq p$ ) nodes in stage 3, assuming there are  $m$  stage-2 nodes in the Clos switching network. The details are as follows.

#### Experimental Setup

- The arc costs were created with a pseudo-random number generator which generates numbers uniformly distributed from 0 to 1. Five sets of random arc costs were generated to run five independent experiments for each multicast connection. Based on these data points, we studied the sensitivity of the algorithms to arc costs.
- The run time and the cost of the solution given by each algorithm were taken. The ratio of the heuristic cost to the optimal cost was calculated to measure the "goodness" of the heuristic algorithms.

To compare the heuristic algorithm with the optimal algorithm, Fig. 5 plots run time (the left  $y$ -axis) and heuristic-to-optimal cost ratio (the right  $y$ -axis) versus number of end nodes,  $d$ , for four  $m$  values ( $m = 8, 16, 32$ , and  $64$ ). Both the individual run times ( $\bullet$  for the optimal algorithm and  $\circ$  for heuristic algorithm) and the average run time of five data points (solid line) are shown. Only the average cost ratio is plotted (dashed line). From the graphs, we can make the following observations and recommendations about the Clos network.

#### Observations and Recommendations

- Run time of the optimal algorithm - For  $m \leq 8$ , the optimal algorithm satisfies our criterion of 0.1s response time. The optimal algorithm is very sensitive to the  $m$  value. For  $m \geq 16$ , the average run time of the optimal algorithm is not satisfactory, although individual run times in certain cases of  $m = 16$  fall within our limit. The run times of different data points (with different arc costs) of the same multicast parameter values can differ significantly. For instance, for  $m = 16$ , the difference can be close to three orders of magnitude. This is attributed to the solution-trimming process of our algorithm. Trimming is most effective in the early stage of enumeration. If the arc costs are such that a larger number of branches can be eliminated in the beginning, then a significant fraction of alternative solutions can be eliminated from consideration. On the other hand, if the arc costs do not allow for substantial trimming at the outset, even if many branches are cut later, chances are the algorithm will still take a long time. The graphs also show that for each  $m$ , run time generally increases with the number of end nodes,  $d$ , although it tends to taper off after a certain point. Overall, the run time is much more sensitive to  $m$  than to  $d$ .
- Run time of the heuristic algorithm - The run time of the heuristic algorithm in all cases satisfy our criterion of 0.1 response time. Furthermore, it is much less sensitive to  $m$  than the optimal algorithm is. Consequently, for large  $m \geq 16$ , the run time of the heuristic algorithm can be several orders of magnitude better than the optimal algorithm. In addition, the heuristic algorithm is also much less sensitive

cost among all possible links). Note that the first summation of  $C(F)$  includes all links from node  $i$  to nodes in  $F$ . However, it is possible for some nodes in  $F$  not to be used, because the arcs from them to nodes in  $K'$  are not minimal. Therefore,  $C(F)$  is the unadjusted cost: the adjusted cost will have  $c_i$  deducted from the unadjusted cost for any node  $j$  that is not used. This distinction, however, is not important if we are considering all  $2^m$  subsets of intermediate nodes as candidates for  $F$  in our optimization process, since there is an optimal candidate in which all nodes in  $F$  are used. Therefore, when comparing different solutions in our optimization process, one needs to concentrate only on the unadjusted cost.

Given the above definition, we have the following theorem [14] relating the unadjusted costs of four alternative proposals:

**Theorem 1** Consider two subsets of the second-stage nodes  $S$  and  $T$ , where  $S \subset T$ , and a node  $h \notin T$ . Then,  $C(S) - C(S \cup \{h\}) \geq C(T) - C(T \cup \{h\})$ .

*Comment:* It is not difficult to see the plausibility of the theorem in simple and intuitive terms. The cost savings due to the inclusion of node  $h$  in node set  $S$  and node set  $T$  are  $C(S) - C(S \cup \{h\})$  and  $C(T) - C(T \cup \{h\})$ , respectively. Since  $S \subset T$ , as far as the costs of the arcs from stage 2 to stage 3 are concerned, the solution with  $S$  as the proposed nodes is less optimized than the solution with  $T$  as the proposed nodes. Therefore, adding node  $h$  to  $S$  is likely to achieve more cost saving than adding it to  $T$ .

**Proof:** The left-hand side of the inequality is

$$C(S) - C(S \cup \{h\}) = -c_{ih} + \sum_{k \in K'} (d_{j_k k} - d_{hk})^+, \quad (2)$$

where  $j_k = \arg \min_{j \in S} d_{jk}$  and  $(x)^+ = \max(0, x)$ . Similarly, the right hand side of the inequality is

$$C(T) - C(T \cup \{h\}) = -c_{ih} + \sum_{k \in K'} (d_{j'_k k} - d_{hk})^+, \quad (3)$$

where  $j'_k = \arg \min_{j \in T} d_{jk}$ . Clearly,  $d_{j'_k k} \leq d_{j_k k}$  since  $S \subset T$ . Hence (2)  $\geq$  (3). □

We now use the above theorem as the basis of our solution-trimming process.

#### Optimal Algorithm: Enumeration-tree Trimming Scheme

Consider an arbitrary node in the enumeration tree in Fig. 3 where the switch modules are distributed into the three sets  $F$ ,  $G$ ,  $H$  defined above. A module will be selected from  $H$  and put into  $F$  and  $G$ , and the enumeration process will branch off in two different directions. In the figure, the particular module in  $H$  that will be selected is fixed at each level. For instance, the figure considers module 0 and 1 at the first and second level, respectively. We will modify the enumeration process slightly by letting the module chosen be a variable. The following test, which will be explained shortly, can be used to determine whether given the current status of  $F$  and  $G$ , we can eliminate one of the two branches without missing the optimal solution.

#### Tests for Trimming Enumeration Tree

1. Choose each module  $h \in H$  successively until all modules in  $H$  have been considered. For each  $h$ , compute  $C(F \cup H)$  and  $C(F \cup H - \{h\})$ . If  $C(F \cup H - \{h\}) > C(F \cup H)$ , move  $h$  from  $H$  to  $F$ ; we will not miss the optimal solution by not considering the branch with  $h$  in  $G$ .
2. Choose each module in  $h \in H$  successively until all modules in  $H$  have been considered. For each  $h$ , compute  $C(F)$  and  $C(F \cup \{h\})$ . If  $C(F \cup \{h\}) > C(F)$ , move  $h$  from  $H$  to  $G$ ; we will not miss the optimal solution by not considering the branch with  $h$  in  $F$ .

If both the tests above do not succeed in moving any module from  $H$  to  $F$  or  $G$ , then trimming is not possible, and we must branch off in two directions by moving a module from  $H$  to both  $F$  and  $G$ .

To see how the first test works, substitute  $T$  in Theorem 1 with  $F \cup H - \{h\}$ . If  $C(F \cup H - \{h\}) - C(F \cup H) = C(T) - C(T \cup \{h\}) > 0$ , then  $C(S) - C(S \cup \{h\}) > 0$  for all  $S \subset T$  according to Theorem 1. We can interpret  $S$  as the proposed modules of an arbitrary leaf node belonging to the branch of the enumeration node where  $h$  is put into  $G$ . The above result says that there is a leaf node in the other branch which achieves lower cost by having  $h$  in addition to  $S$  as the proposed nodes. Thus, given the current status of  $F$ ,  $G$ , and  $H$ , we will not miss enumerating the optimal solution if we only branch in the direction where  $h$  is in  $F$ . Similar reasoning applies to the second test by substituting  $S$  in Theorem 1 with  $F$ .

Finally, recall that if  $|K'| < m$  (i.e., there are fewer than  $m$  stage-3 switch modules in the multicast connection), at most  $|K'|$  stage-2 switch modules will be used. We can incorporate another test at each node of the enumeration tree: if  $F = |K'|$ , branch no more; this node will be taken as one of the proposals to be examined. This test can substantially reduce the computation needed if  $|K'| \ll m$ .

#### Heuristic Algorithm: 3-step Augmentation Scheme

The run time of the optimal algorithm can be excessive in the worst case. We now consider a heuristic algorithm that attempts to find a solution that is close to optimal but within a shorter time. It consists of three procedures running in sequence, each improving on the solution given by the previous procedure. Figure 4 is a simple example for illustration of the algorithm. Also, since we are not considering all  $2^m$  subsets of intermediate nodes in our optimization process here, we will be concentrating on the adjusted cost when comparing different solutions.

##### 3-step Augmentation Algorithm

1. Find the shortest-path solution. That is, for each node  $k \in K'$ , find  $j_k = \arg \min_{j \in J} c_{ij} + d_{jk}$ , and make links  $(i, j_k)$ ,  $(j_k, k)$  and node  $j_k$  part of the multicast tree.
2. Find a new multicast tree as follows. Denote the intermediate switch modules used in the shortest-path solution by  $V$ . Find the set of minimal links from node set  $V$  to node set  $K'$  using the minimal-link selection process based on  $V$ . That is, for each  $k \in K'$ , find  $j_k = \arg \min_{j \in V} d_{jk}$ , and make links  $(i, j_k)$ ,  $(j_k, k)$  and node  $j_k$  part of the multicast tree. Remove nodes from  $V$  that are not part of the resulting multicast tree.
3. For each node  $v \in V$ , denote the set of third-stage nodes attached to it in the multicast tree by  $W_v$ . See if these nodes can be attached to other nodes in  $V$  at a net cost saving.

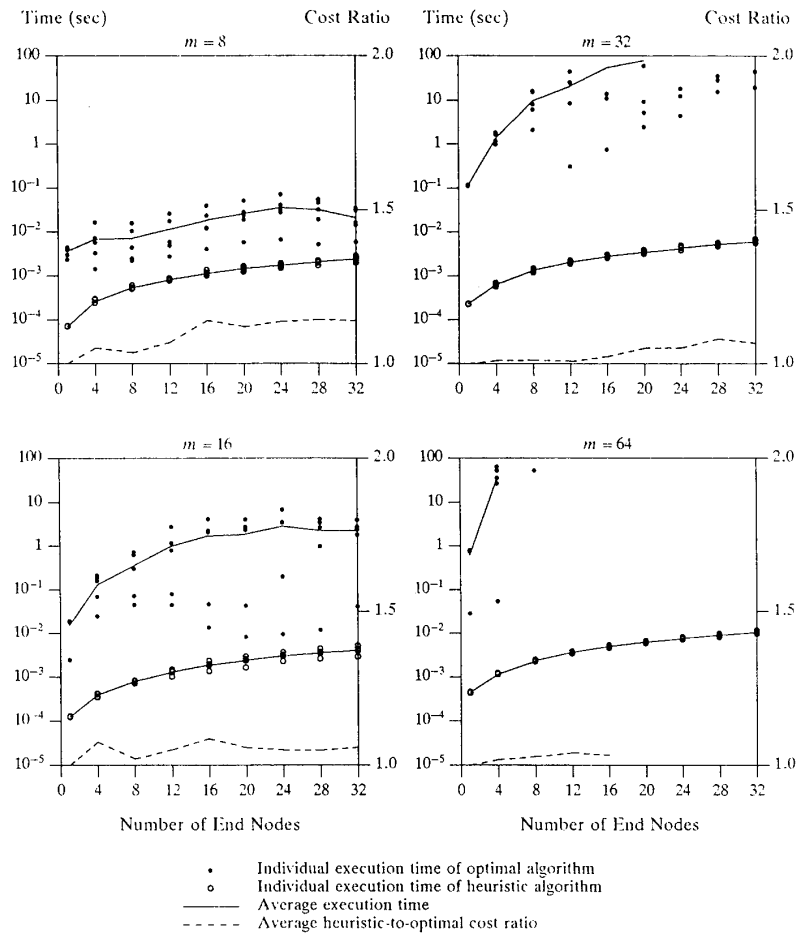


Figure 5: Run time (left y-axis) and heuristic-to-optimal cost ratio (right y-axis) versus number of end nodes for  $m = 8, 16, 32$  and  $64$ .

to the arc costs, and it is highly dependable as far as meeting the response-time limit is concerned.

- Cost ratio – The *average* heuristic-to-optimal cost ratio is very close to 1.0 on the whole, and never exceeds 1.15. What makes the heuristic algorithm even more interesting is that for large  $m$ , when the run time of the optimal algorithm is long, the average cost ratio quite timely becomes closer to 1.0.
- Implication of parallel computing – Our optimal algorithm can be parallelized quite easily. Each time the enumeration process branches off in two directions, computation on each branch can be assigned to a separate processor. Nevertheless, even with 100 processors, the reduction in run time is at most two orders of magnitude. From Fig. 5, although parallel computing may help when  $m$  is small, it will not solve the problem for  $m \geq 64$ .
- Implication of time-limit interrupts – The optimal algorithm can easily be modified to store the best solution computed

so far. With this change, the algorithm can be interrupted when the time limit of 0.1s is reached. This gives us a feasible, albeit possibly non-optimal, solution.

- Implementation strategy – For small networks (say networks with less than 32 intermediate nodes) the response time of the optimal algorithm in some cases is no more than an order of magnitude larger than that of the heuristic algorithm. The use of the optimal algorithm should be considered for these cases. We can adopt a strategy in which the optimal and heuristic algorithms are run in parallel with a set time limit. When time is up, the better solution offered by the two algorithms is chosen.

Based on our further experimentation, we found that the optimal algorithm can usually meet the response time limit if  $m \leq 12$ .

## V. Conclusions

This paper has investigated multicast routing in 3-stage Clos

## 45.3.6

networks to find out if routing will be a bottleneck to call setup. An optimal and a heuristic algorithms for multicast routing have been designed and tested. The optimal algorithm is centered on a procedure which eliminates a large number of non-optimal solutions from consideration without computing them, thereby achieving a substantial reduction in run time. The heuristic algorithm is based on a three-step optimization process in which each step attempts to improve on the solutions found by the previous steps. Major observations and implications of the work are summarized below.

1. Computation experiments show that the heuristic algorithm can find multicast routes that are close to optimal within an average response time that is several orders of magnitude lower than that of the optimal algorithm. Compared with the optimal algorithm, the response time of the heuristic algorithm does not increase as much with the network size. In addition, the response time of the heuristic algorithm is also relatively insensitive to the values of arc costs.
2. For large networks (say networks with more than 32 nodes at stage 2) the response time of the optimal algorithm can exceed the targeted 0.1s by several orders of magnitude. Even with a more powerful processor (say 100 MIPS) than the one used in our experiments, the response time will still not be satisfactory. For small networks (say networks with less than 32 intermediate nodes) the run time of the optimal algorithm in some cases is no more than an order of magnitude larger than that of the heuristic algorithm. The use of the optimal algorithm should be considered in these cases.
3. By modifying the optimal algorithm to store the best solution computed so far, we can have a hybrid strategy in which the optimal and heuristic algorithms are run in parallel. When a set time limit is reached, the better solution offered by the algorithms is chosen.
4. The need for sophisticated routing procedure in itself does not rule out Clos network as a viable choice for a switch architecture. If the network can also be designed to meet other requirements, such as grade-of-service and fault tolerance requirements, without complex control mechanisms, then it is a serious candidate for a future broadband switch.

As a final note, although this paper is motivated by the Clos switching network, the algorithms and the discussion here also apply to large-scale communications networks with a two-hop structure. It is likely that facility cross-connects will be used to configure future ATM networks into very simple logical network structures in order to facilitate control and increase reliability. It is undesirable from a control standpoint to have too many stages of queues between two nodes. This work is especially relevant to logical networks in which two nodes are directly connected via a set of logical paths, and indirectly connected via another set of two-hop logical paths, with each involving only one intermediate switching node.

## References

- [1] CCITT, New Draft Recom. I. 150, B-ISDN ATM Layer Functionality and Specifications, *Committee XVIII*, Jan. 1990.
- [2] T. T. Lee, "A Modular Architecture for Very Large Packet Switch," *IEEE Trans. on Commun.*, vol. 6, pp. 1455-1467, July 1990.

- [3] K. Y. Eng, M. J. Karol, and Y. S. Yeh, "A Growable Packet (ATM) Switch Architecture: Design Principles and Applications," *Conf. Record, IEEE Globecom '89*, pp. 1159-1165, Nov. 1989.
- [4] S. C. Liew and K. W. Lu, "A 3-stage Interconnection Structure for Very Large Packet Switches," *International J. of Digital and Analog Communications*, vol. 2, pp. 303-316, 1989.
- [5] K. Hajikano, K. Murakami, E. Iwabuchi, O. Isono, and T. Kobayashi, "Asynchronous Transfer Mode Switching Architecture for Broadband ISDN," *Conf. Record, IEEE ICC '88*, pp. 911-915, June 1988.
- [6] H. Suzuki, H. Nagano, T. Suzuki, T. Takeuchi, and S. Iwasaki, "Output-buffer Switch Architecture for Asynchronous Transfer Mode," *Conf. Record, IEEE ICC '89*, pp. 99-103, June 1989.
- [7] Y. Sakurai, N. Ido, S. Gohara, and N. Endo, "Large Scale ATM Multi-stage Switching Network with Shared Buffer Memory Switches," *Proc. ISS '90*, vol. 4, paper A6.3, pp. 121-126, 1990.
- [8] C. Clos, "A Study of Non-blocking Switch Network," *Bell System Tech. J.*, vol. 32, pp. 406-424, 1953.
- [9] D. Bertsekas and R. Gallager, *Data Networks*, Prentice-Hall, 1987.
- [10] C-H Chow, "On Multicast Path Finding Algorithm," to appear in *Conf. Record, IEEE Infocom '91*, 1991.
- [11] R. Melen and J. S. Turner, "Nonblocking Networks for Fast Packet Switching," *Conf. Record, IEEE Infocom '89*, pp. 548-557, April 1989.
- [12] C. H. Papadimitriou and K. Steiglitz, *Combinatorial Optimization: Algorithms and Complexity*, Prentice-Hall, 1982.
- [13] P. Winter, "Steiner Problem in Networks: A Survey," *Networks*, vol. 17, pp. 129-167, 1987.
- [14] A. Alcouffe and G. Muratet, "Optimal Location of Plants," *Management Science*, vol. 23, pp. 267-274, Nov. 1976.