Traffic Routing for Multicomputer Networks with Virtual Cut-Through Capability

Dilip D. Kandlur, *Member, IEEE*, and Kang G. Shin, *Fellow, IEEE*

Abstract—**A point-to-point interconnection network can be a good choice for use in distributed real-time systems, since it can provide multiple disjoint paths between nodes to tolerate link and node failures. The major drawback of this type of network has been that full connectivity becomes prohibitively expensive as the number of nodes increases, and partial connectivity implies long store-and-forward delays in the prevalent packet-switching mode of operation. However, this drawback of partially connected point-to-point networks can now be overcome with virtual cut-through, a scheme which is feasible to implement using current VLSI technology.**

This paper addresses the problem of selecting routes for interprocess communication in a network with virtual cut-through capability, while balancing the network load and minimizing the number of times that a message gets buffered. The problem is important because the message delivery delay depends upon the number of times that a message gets buffered at intermediate nodes on the route. The approach taken here is to formulate the route selection problem as a minimization problem, with a link cost function that depends upon the traffic through the link. The form of this cost function is derived based on the probability of establishing a virtual cut-through route.

It is shown that this route selection problem is $\mathcal{NP}$-Hard, so an approximate algorithm is developed which tries to incrementally reduce the cost by re-routing traffic. The performance of this algorithm is evaluated for two popular network topologies: the hypercube and the C-wrapped hexagonal mesh—example networks for which virtual cut-through switching support has been developed.

Index Terms— **Fault-tolerant networks, $\mathcal{NP}$-Hard, real-time communications, routing, virtual cut-through.**

I. Introduction

THE availability of inexpensive and powerful microprocessors has led to the use of multicomputers in an increasing number of critical real-time applications. Many of these applications have stringent reliability requirements, since a failure of the computer system can lead to a disaster. The network interconnecting processors is an important component in these multicomputer systems. One of the goals in the design of the interconnection network is to provide reliable communication in the presence of component failures. The approach taken in multicomputers like SIFT [1] and MAFT [2] is to provide a fully-connected network. Although this method is extremely reliable, it does not scale and it is difficult to use in large systems.

Another approach to providing fault-tolerant communications is the AIPS *virtual bus* scheme [3], [4]. The AIPS network controllers are connected by multiple redundant point-to-point links, but they are configured under software control to act as a single virtual bus. However, there is a significant delay incurred in traversing through the repeater stages in a controller node, so the end-to-end latency is substantial when multiple links are to be traversed. This results in long bit-holding times for the contention resolution protocol, which can adversely affect the throughput. Also, the virtual bus configuration does not permit simultaneous communication over disjoint paths in the network.

A point-to-point interconnection network with a regular structure is proposed as a good candidate for use in real-time applications. Examples of such networks include hypercubes and meshes [5], [6]. The existence of multiple disjoint paths between nodes in these networks make them robust to link and node failures. Also, since it is possible to support multiple simultaneous conversations in different links, the total throughput achievable is higher. Until recently, one major drawback of such networks was the large delay in communication which was associated with store-and-forward packet switching. Advances in VLSI technology have now made it possible to implement sophisticated switching schemes, like the *virtual cut-through* scheme [7], which reduce the message delivery time significantly. It is therefore feasible to use a multicomputer system based on a point-to-point network with virtual cut-through for many real-time applications. Virtual cut-through, which is pivotal in the discussion that follows, is described below.

Virtual cut-through is a switching technique introduced by Kermani and Kleinrock [7] which has flavors of both circuit switching and packet switching. In this method, messages arriving at an intermediate node are forwarded to the next node in the route without buffering if a circuit can be established to the next node. This differs from conventional packet-switching schemes in the sense that messages do not always get buffered at an intermediate node. It also differs from circuit switching schemes since messages do not wait for the entire circuit to the destination to be established before proceeding along the route. With current VLSI technology, it is now possible to implement this switching scheme for multicomputer interconnection net-

Manuscript received September 11, 1990; revised October 6, 1991. This work was supported in part by the Office of Naval Research under Contract N00014-85-K-0122, NASA under Grant NAG-1-296, and an IBM Graduate Fellowship. Any opinions, findings, and conclusions or recommendations expressed in this paper are those of the authors and do not necessarily reflect the views of the funding agencies.

D. D. Kandlur is with IBM Research Division, T. J. Watson Research Center, Yorktown Heights, NY 10598.

K. G. Shin is with the Real-Time Computing Laboratory, Department of Electrical Engineering and Computer Science, The University of Michigan, Ann Arbor, MI 48109-2122.

IEEE Log Number 9200312.

works, and network controllers like the Torus routing chip [8], [9] and the HARTS routing controller [10] are examples of such implementations.

The main advantages of a virtual cut-through scheme over a circuit switching scheme are that circuit setup and tear-down delays are not incurred, and network channels are not tied up. On the other hand, since messages do not necessarily get buffered at intermediate nodes, the delays encountered are smaller than those for a packet switching scheme. The automatic forwarding also means that if a cut-through switch is established, the packet is not seen by the processor at that intermediate node. Hence, the load imposed on the processors at the intermediate nodes is much smaller than that for a packet switching scheme.

The problem addressed in this paper is the selection of routes for messages in a multicomputer system of this type. It is assumed that the assignment of tasks to different nodes has already been done and the communication patterns between tasks are known. The reason for this is that, in a real-time system, the placement of tasks depends not only on the communication pattern but also on the peripherals and resources available at the different nodes. The traffic between nodes is specified in terms of mean data rates, which can be determined from the length of the messages and the periodicity of the communication. Given the network topology and this traffic pattern, the objective is to select a route for each pair of communicating processes such that the network load is balanced and the probability of establishing virtual cut-through is enhanced. Route selection is essential in order to reduce message delivery delays and to make full use of the network's cut-through capabilities. It can be considered to be a high-level approach to message scheduling, where the units considered are inter-process traffic patterns and not individual packets.

In the type of network considered here, it is assumed that link capacities are large and the overheads in handling packets at intermediate nodes is a significant factor. This assumption is reasonable considering the Gigabit range bandwidth possible with fiber-optic links. Also, we restrict our attention to finding a single fixed route for each pair of communicating processes. Fixed routing is preferable in real-time systems because it can simplify the computation of bounds for the message delivery time, an issue which is important when messages with delivery time constraints are considered [11].

Variations of this problem have been studied for general wide-area networks, mainly as techniques for building routing tables in packet switched networks [12]. However, most of these studies are based on link capacities, since these are limited for wide-area networks. In the TRANSPAC network, the routing algorithm uses link costs depending upon utilization [12]. The cost function is piecewise continuous with hysteresis. The routing is computed to minimize costs depending upon the current utilization or queue lengths, but future traffic is not considered.

One study relating to multiprocessor networks is by Bianchini and Shen [13], in which a traffic scheduling algorithm is presented. However, they assume that the switching nodes in the network have the ability to implement traffic splitting, which makes the network behave like a fluid-flow pipeline. They further assume that the delivery cost is not depended on the length of the path and arrive at a link cost which is an exponential function of the link traffic. The properties of the exponential link cost function result in a major simplification for their traffic scheduling algorithm. On the other hand, the message delivery delays in a multicomputer network have a strong dependence on the length of the path and this is taken into account in the cost function derived here.

In this paper, the traffic scheduling problem is formulated as an optimization problem in Section II. The link cost function used in this formulation is derived in Section III. In Section IV it is shown that the optimization problem is $\mathcal{NP}$-Hard and the associated decision problem is $\mathcal{NP}$-Complete. The heuristic algorithms developed to solve this problem and their basis are presented in Section V. The performance of these algorithms is then analyzed using simulation in Section VI. The paper concludes with Section VII.

II. Notation and Problem Formulation

The environment under consideration is a multicomputer system with a point-to-point interconnection structure and a set of assigned tasks. It is assumed that the message communication patterns between the tasks are known in terms of average length of messages and the frequency of communication. From this, one can determine the unidirectional task-to-task communication volume. We call this volume a *flow*, which is measured in bytes/second. The problem is then to determine a route through the network for each flow to meet a global "goodness" criterion. The notation used to describe this problem formally is given below.

DG: A digraph (N, L) representing the multicomputer system which consists of a set, N, of n nodes and a set, L, of m links.

e_{ij}: The directed link/edge from node i to node j.

Q: The set of q flows $\{(s_i, d_i, r_i) :$ flow r_i required from node s_i to node d_i in DG, $\{i = 1, \cdots, q\}$.

P_k: The set of all simple directed paths from s_k to d_k in DG, where $(s_k, d_k, r_k) \in Q$.

p_k: An element of P_k, which can be considered to be an ordered subset of L.

S: $\{p_k : k = 1, \cdots, q\}$, an element of $P_1 \times P_2 \cdots \times P_q$.

f_{ij}: The flow in link e_{ij}. $f_{ij} = \sum_{k=1}^{q} I_{ij}^k r_k$, where $I_{ij}^k = 1$ if $e_{ij} \in p_k$ and 0 otherwise.

C_{ij}: The capacity of link e_{ij}.

c_{ij}: The cost of routing traffic onto link e_{ij}, which can be a function of f_{ij} and/or C_{ij}.

For brevity, the term *path* is used to denote a *simple directed path* in the digraph DG. Also, the terms *path* and *route* are used interchangeably in this paper. Given these definitions, and the condition that there is a unique path for each flow, the routing problem can be stated as follows.

Path Selection Problem: Given N, L, Q, the capacities and the costs of the links, find a set, S, of paths such that for each link $e_{ij} \in L$, $f_{ij} \leq C_{ij}$ and total cost $T = \sum_{k=1}^{q} r_k \cdot (\sum_{e_{ij} \in p_k} c_{ij})$ is minimized.

By changing the order of summation, T can be rewritten as $T = \sum_{e_{ij} \in L} c_{ij} \cdot (\sum_{k=1}^{q} I_{ij}^k r_k)$. The inner summation is seen

to be the link flow f_{ij}, so $T = \sum_{e_{ij} \in L} c_{ij} \cdot f_{ij}$.

As formulated above, the objective of route selection is to minimize the total cost over all the flows in the network, while adhering to the capacity constraints for each link. Although this formulation is general, the characteristics of a particular type of network can be incorporated by a suitable choice of the link cost function c_{ij}. The features of virtual cut-through switching make the flow through a link critical to the cost function, since it affects the probability of establishing a cut-through route. Intuitively, it would be better to choose longer than optimal routes so as to avoid heavily used links. However, path lengths remain an important factor because long paths use up more network resources. It will be shown in the next section that the link cost function can be chosen such that minimizing the total cost is equivalent to maximizing the weighted probability of achieving virtual cut-through in the network. The total cost function T then represents a tradeoff between using longer than optimal paths to avoid busy links versus the penalty incurred by the increased path length.

III. Derivation of the Link Cost Function

It is our intention to minimize the probability of messages getting buffered at intermediate nodes in the path. When a message gets buffered, it has to be examined by the processor on the communication adapter and then scheduled for onward transmission. Thus, in addition to the message store-and-forward delay, an additional processing cost is incurred. Hence, it is preferable to use a long path with a lower probability of buffering. The link cost function is derived based on this probability.

The network $DG = (N, L)$ is modeled as a network of queues with one (single server) queue for each link. The problem of analyzing a network of queues with arbitrary arrival and service patterns is known to be intractable. To make the analysis tractable, the following assumptions are made in this analysis. It is noted that these assumptions may not be valid for a real-time system where periodic message traffic is predominant. However, their use is justified, since the results obtained are used mainly as a guideline for the choice of the link cost function.

- A0: Poisson message generation at the source nodes.
- A1: Exponentially distributed message lengths.
- A2: Infinite nodal capacity.
- A3: A packet loses its identity when it arrives at a node and a new length is chosen for it at random, i.e., the Independence Assumption [14].

This model, and the Independence Assumption, was first used by Kleinrock [14], and simulations and actual measurements have later demonstrated the validity of the model. The model has also been used by other researchers for the analysis of networks [7], [15]. Under these assumptions, the arrival process for a link is independent of the departure process from the link and Jackson's result [16] can be applied to the network of queues. That is, in the steady state the network behaves as if each node were stochastically independent of the other nodes and similar to an M/M/1 system.

Consequently, the probability that a message gets buffered at a particular intermediate node is independent of the probability of it getting buffered at any other intermediate node. Hence, the probability of establishing a source-destination cut-through route can be written as a product form expression. Consider a path $P = \{e_{n_0 n_1}, e_{n_1 n_2}, \cdots, e_{n_{\ell-1} n_\ell}\}$ from node $n_0 = i$ to node $n_\ell = j$. The probability that a message from node i to node j will be delivered without buffering at intermediate nodes is given by

$$\begin{aligned}
&\text{Prob(cut-through on } P) \\
&\quad = \prod_{r=1}^{\ell-1} (1 - \text{Prob(buffering at node } n_r)) \\
&\quad = \prod_{r=1}^{\ell-1} (1 - \text{Prob(link} e_{n_r n_{r+1}} \text{ is busy)}) \\
&\quad = \prod_{r=1}^{\ell-1} (1 - \rho_{n_r n_{r+1}}) \\
&\quad = 1 - \sum_{r=1}^{\ell-1} \rho_{n_r n_{r+1}} + \text{higher order terms in } \rho
\end{aligned}$$

where $\rho_{n_r n_{r+1}} = f_{n_r n_{r+1}} / C_{n_r n_{r+1}}$ is the mean utilization of link $e_{n_r n_{r+1}}$.

In the type of high-bandwidth network considered here, the mean utilization of the links would tend to be small. Hence, to a good approximation, the higher order terms can be dropped from the equation. Therefore,

$$\text{Prob(cut-through on } P) \approx 1 - \sum_{r=1}^{\ell-1} f_{n_r n_{r+1}} / C_{n_r n_{r+1}}.$$

Prob(cut-through on P) is maximum when $\sum_{r=1}^{\ell-1} f_{n_r n_{r+1}} / C_{n_r n_{r+1}}$ is minimum.

To consider all flows in Q we have to form a weighted sum over all q flows and maximize this:

$$\sum_{k=1}^{q} r_k \cdot \text{Prob(cut-through on } p_k).$$

In this sum, the utilization of the first link in the path will not be included because the expression for cut-through on p_k includes only intermediate nodes. However, for this analysis, an approximation is used in which the utilization of the first link is also considered, since a lower utilization for the first link would ensure that the delay experienced before the message is transmitted from the source node is small. This then corresponds to finding the minimum of $\sum_{k=1}^{q} r_k \cdot \left(\sum_{e_{ij} \in p_k} (f_{ij} / C_{ij}) \right)$.

The form of this function matches the cost function of the Path Selection Problem, with the link cost $c_{ij} \equiv f_{ij} / C_{ij}$. If we restrict our attention to homogeneous networks, the capacity C_{ij} is the same for all links and the cost function can be simplified to $c_{ij} = f_{ij}$. This function can be interpreted as follows. As the number of flows through a link increases, f_{ij}

increases and so does the cost of using the link. This would present a bias toward the choice of longer paths to circumvent heavily used links. On the other hand, the total cost for a flow depends upon the length of the path and so, there is an opposing bias to reduce the length of the paths. With $c_{ij} = f_{ij}$ as the link cost function, the total cost $T = \sum_{e_{ij} \in L} f_{ij}^2$. This is the form of the total cost function used in subsequent sections.

IV. Problem Characterization

In this section it is shown that the Path Selection Problem is $\mathcal{NP}$-Hard and the decision problem associated with it is $\mathcal{NP}$-Complete. The decision problem, shown below as Decision Problem 1, is $\mathcal{NP}$-Complete because it contains the subproblem of finding a feasible set of paths. It can be shown that SATISFIABILITY (SAT) [17] is reducible to this subproblem of finding feasible paths. The reduction relies on capacity constraints which force exclusive use of links. However, even when the capacity constraints are not imposed, the problem remains $\mathcal{NP}$-Complete when the link cost c_{ij} is a function of the link utilization. In particular, it is shown that Decision Problem 2 given below, which has the cost function $c_{ij} = f_{ij}$ and no link capacity constraints, is $\mathcal{NP}$-Complete. The definition of SAT, adapted from [18], is given in Appendix A.

Decision Problem 1: Given N, L, Q, and the link capacities C_{ij}. Is there a feasible set, S, of paths such that for each edge $e_{ij} \in L$, $f_{ij} \leq C_{ij}$ where $f_{ij} = \sum_{e_{ij} \in p_k} r_k$?

Decision Problem 2: Given N, L, Q, link cost function $c_{ij} = f_{ij}$, and a bound B. Is there a feasible set, S, of paths such that total cost $= \sum_{k=1}^{k=q} r_k \cdot (\sum_{e_{ij} \in p_k} c_{ij}) = \sum_{e_{ij} \in L} f_{ij}^2 \leq B$?

The reduction from SAT is similar for the two problems and it is shown only for the second problem. It is based on the method used in [19] for the Multicommodity Integral Flow problem. The proof is constructive, using graph components shown in Fig. 3, and it shows a polynomial time transformation from an instance of SAT to an instance of Decision Problem 2.

Theorem 1: Decision Problem 2 (DP2) is $\mathcal{NP}$-Complete.

Proof: The proof of this theorem can be found in Appendix B.

Corollay 1: The Path Selection Problem with cost function $c_{ij} = f_{ij}$, and no capacity constraints, is $\mathcal{NP}$-Hard.

The Path Selection Problem is a search problem which has a finite upper bound, so it can be easily shown that it is Turing reducible to DP2 using the "binary search" technique described in [18]. Hence, it follows from the theorem that the Path Selection problem is $\mathcal{NP}$-Hard. Also, from the proof of Theorem 5 it can be seen that a bound, B, can be suitably chosen to accommodate other cost functions where the link cost is a monotone increasing function of the link utilization.

V. Solution Algorithm

The problem of finding a minimum cost solution has been shown to be $\mathcal{NP}$-Hard when the link cost function is a function of the utilization. It is observed that the search space grows as $\prod_{k=1}^{q} |P_k|$, where $|P_k|$ is usually large for the types of regular interconnection networks that are under consideration. This implies that it is impractical to determine the optimal solution by brute-force techniques, even for a small number of flows. Thus, we need to find a good heuristic solution for the problem. We have developed an algorithm which selects paths one at a time, while keeping the other paths fixed. The algorithm begins with an initial assignment of paths to the flows. It then successively tries to improve the paths, considering one flow at a time, until no further cost improvement is possible. An outline of this algorithm, which is called Route_All, is given in Fig. 2. The procedure for determining the path for a single flow, called Route_One, is described in Fig. 1. The theoretical basis for this procedure is given below.

The algorithm required for selection of a single route is one in which we are given the current utilization of the network and the characteristics of the new flow to be established. The objective here is to find a route such that, under the given cost function, the incremental cost is minimized. It is shown that a version of the *shortest path* algorithm with appropriately defined link lengths will yield a solution.

Consider a path p_k for this new flow. When this flow is added to the system, the new flows in the links are given by

$$f'_{ij} = \begin{cases} f_{ij} + r_k & \text{if } e_{ij} \in p_k \\ f_{ij} & \text{otherwise.} \end{cases}$$

The additional cost incurred is then

$$\begin{aligned} I &= \sum_{e_{ij} \in L} [(f'_{ij})^2 - f_{ij}^2] \\ &= \sum_{e_{ij} \in p_k} (2 f_{ij} r_k + r_k^2) \\ &= r_k \cdot \sum_{e_{ij} \in p_k} (2 f_{ij} + r_k). \end{aligned}$$

This incremental cost, I, is mininum when $\sum_{e_{ij} \in p_k} (2f_{ij} + r_k)$ is minimum. This minimum can be obtained by using $2f_{ij} + r_k$ as the *length* of link e_{ij} and finding the shortest path from s_k to d_k. This is the approach used in Algorithm Route_One, in which a greedy algorithm is used to find the shortest path. Since all the link lengths are nonnegative, a greedy algorithm would yield the shortest path. From this discussion, it is clear that Algorithm Route_One is optimal under the condition that routes that are already established cannot be disturbed. This is the case when requests for creation of new flows arrive dynamically and are serviced in order.

Algorithm Route_All starts with an initial flow assignment, F. It then selects each flow in turn, removes it from F, and checks whether a better route is available using Route_One. The function *Incremental_cost* computes I as shown in the derivation above. The solution obtained is not guaranteed to be optimal because it is possible that a better solution may be obtained by the simultaneous re-routing of multiple paths. The algorithm, however, is guaranteed to terminate since the cost function is monotone decreasing with successive re-routing. Algorithm Route_One is essentially a shortest path algorithm and its complexity is $O(N^2)$ for an adjacency

Algorithm Route_One

```
*/
    Given N, L, F={f_ij : e_ij ∈ L} and a new flow (s_k,d_k,r_k),
    find a route p_k with minimum incremental cost.
*/
    /* Initialize the distance matrix D_ij */
    for each i, j ∈ N
        if e_ij ∈ L
            D_ij := 2 * f_ij + r_k
        else
            D_ij := 0
        endif
    end for
    Use the Greedy Algorithm to find a path p_k from s_k to d_k
    with the distance matrix D_ij.
end Route_One
```

Fig. 1. Single path selection.

Algorithm Route_All

```
/*
    Given N, L, and Q, find a set of paths for each flow in Q
    with the objective of minimizing the cost function.
*/
    Assign an initial path for each flow, initialize the flow matrix F.
    new_route_found := TRUE
    while (new_route_found)
        new_route_found := FALSE
        for each (s_i, d_i, r_i) in Q
            remove p_i from F
            old_cost = Incremental_cost(p_i)

            Route_One (s_i, d_i, r_i) returns path p_i'

            new_cost = Incremental_cost(p_i')
            if (new_cost < old_cost)
                add p_i' to F
                new_route_found := TRUE
            else
                restore p_i to F
            end if
        end for
    end while
end Route_All
```

Fig. 2. The route selection algorithm.

matrix representation of the network. Algorithm Route_All uses Route_One repeatedly, but the number of iterations is data dependent and cannot be easily determined. The running time of Route_All is bounded below by $q \cdot N^2$, that is, it is $\Omega(qN^2)$. However, the results of experiments with the algorithm on several flow patterns show that the convergence is rapid.

VI. Performance Evaluation

In this section, we evaluate the performance of the algorithms developed in Section V, and study the effectiveness of the cost function derived in Section III. To evaluate the performance of Algorithm Route_All, termed ALLP, it is compared with two other path selection algorithms. One of them, called INC, is a simple extension of Algorithm Route_One in which routes are assigned to the flows in the order of arrival and no further re-routing is employed. The other is a "pure" shortest path algorithm (SP) in which a path of minimum length from source to destination is selected for each flow. The purpose of this comparison is to measure the improvement obtained using the re-routing technique. The

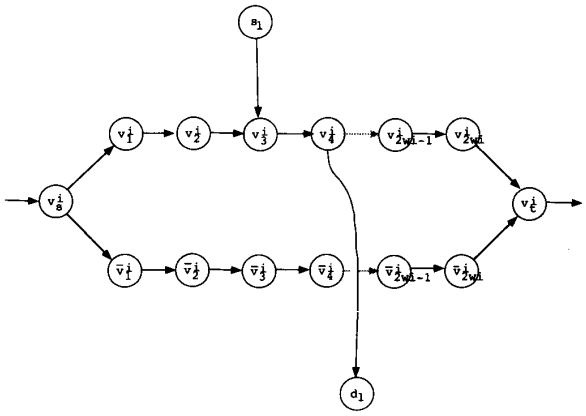


Fig. 3. Graph component corresponding to a variable.

comparison between SP and INC demonstrates the utility of Algorithm Route_One as compared to a simple shortest path algorithm. The complexity of INC is the same as that of the shortest path algorithm, $O(N^2)$ for an adjacency matrix representation of the network. The execution time is slightly higher because it has to update the flow f_{ij} in the links which constitute the route after the route has been selected.

To evaluate the effectiveness of the cost function, we used a discrete-event simulator which models a network with virtual cut-through switching. This simulator was originally developed by the authors of [20] to study the performance of the HARTS network. HARTS is a multicomputer with a C-wrapped hexagonal mesh topology [6], currently being built in the Real-Time Computing Laboratory, The University of Michigan. The C-wrapped hexagonal mesh is a regular, homogeneous graph in which each node is connected to six other nodes (see Fig. 4). Each HARTS node has a network processor which is connected to the network through the routing controller. The simulator models the front-end routing controller [10], which implements virtual cut-through, and its interface to the network processor of each HARTS node. It accurately models the delivery of each packet by emulating the routing hardware along the route of a packet at the microcode level. It also captures the internal bus access overheads in the routing controller experienced by packets as they pass through an intermediate node. We modified the simulator to generate traffic based on each connection (flow), and to collect statistics on the number of *bufferings*, i.e., the number of times packets failed to cut-through and were buffered at intermediate nodes. We chose to measure the number of bufferings because that is what the cost function tries to capture. We did not use the average delivery time as a measure because the overhead cost incurred when a packet is buffered depends upon the processor, and the load on the processor. The simulator did not model this load.

The simulator allows us to relax many of the modeling assumptions used in the derivation of the cost function. In particular, we do not use the Independence assumption, nor do we have exponentially distributed message lengths. In our

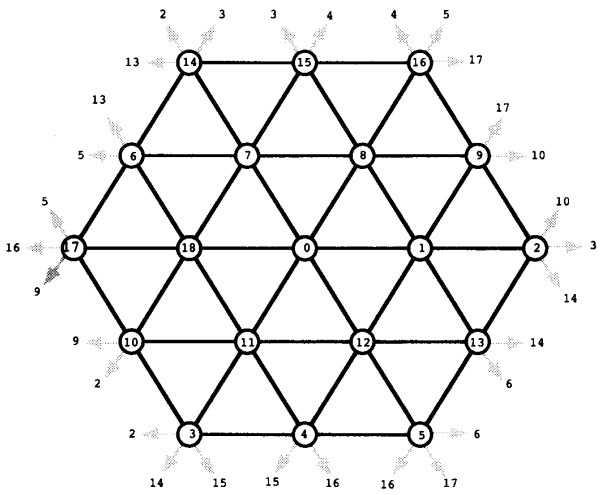


Fig. 4. A hexagonal mesh of dimension 3 (E-3).

experiments, we configured the simulator for a hexagonal mesh of size 5 (denoted by E-5), which has 61 nodes. We generated sets of flows by selecting the source, destination, and quantity of each flow using independent random number generators. The source node was chosen using a uniform random number generator. For a fixed source, the destination node was chosen using an independent random number generator, with two different types of distributions in different experiments. One was a uniform distribution, where the destination node is chosen to be any other node in the system with equal probability. The second distribution tried to capture the principle of locality, that is, the destination node is more likely to be close to the source. The value of each flow was selected in the range [1, 10] using a third uniform random number generator. Note that in the cost function, it is the *relative* values of the flows which are significant, the units do not affect the results.

We used the SP, INC, and ALLP algorithms to produce 3 different sets of route assignments for each set of flows. These algorithms also returned the "cost" of the route assignments that they produced. We then used the mesh simulator to simulate traffic on these flows and measured the number of bufferings for each of the three sets of routes. The value of a flow was interpreted to be a rate of packet generation, and the packet size was fixed at 128 bytes. On each flow, packets were generated using a Poisson distribution with a mean inter-arrival time specified by the value of the flow. The actual rate of packet generation and the service time for the packets are a function of the link transmission rate. The effective link transmission rate in our simulation was 1.5 μs per byte, and a flow value of 1 was converted into an arrival rate of 1 packet per 180 ms. This value was chosen to ensure that the average link utilization remained low (it was below 0.4 in all our experiments). We note that the number of bufferings observed is a function of link utilization and it would change if we assigned a different rate for the flow value. We could make this arbitrary choice of generation rate because, in our experiments, we are interested in the relative values of the number of bufferings for the three route assignments. In each simulation, we collected statistics over a period in which 100 000 packets were delivered. This number was chosen by looking at the convergence of the statistic of interest (number of bufferings) in some sample simulations.

Experiment 1: In this experiment, for any source, the destination of the flow was chosen using a uniform distribution. The number of flows was varied from 50 to 500, and this resulted in a change in the observed average link utilization from 0.035 to 0.33. The results of the simulation are shown in Fig. 5, which plots the number of bufferings observed for a particular number of flows with each of the three algorithms. Each data point in the graph represents an average over 50 different data sets. The results here can be compared with the cost figures shown in Fig. 6 for the same sets of flows. This experiment also allows us to compare the performance of the three algorithms. The comparison shows that the computed cost and the observed number of bufferings have similar behavior, except when the number of flows is very small. In this case, INC and ALLP perform better than what the cost function shows. It is seen that there is a modest improvement for INC over SP, and for ALLP over INC. The improvement for INC over SP is higher when there are fewer flows, that is, when the network load is light, because INC is able to locate routes through links that are otherwise unused. Since INC performs quite well in this situation, the additional improvement attainable by using ALLP is small. On the other hand, when the number of flows is very large, because of the uniform communication pattern, most of the links are almost evenly loaded and INC is not able to find much improvement.

Experiment 2: This experiment differs from Experiment 1 in that the destination node was picked by first selecting a hexagonal ring and then selecting the particular node within the ring, considering the source node as the center of the mesh. Since there are more nodes in the outer rings, the probability of selecting a node in an outer hexagon is smaller than that of selecting a node from an inner hexagon. With this distribution, the average source-destination separation in a mesh of size e is $e/2$ as compared to $(2e-1)/3$ in the case of a uniform distribution. When the number of flows was varied from 50 to 400, the observed average link utilization changed from 0.035 to 0.26. The number of flows could not be increased beyond 400 for the simulation because the routes generated by SP resulted in severe network congestion. The results of the simulation are shown in Fig. 7, while those of the cost function are shown in Fig. 8. Here again, the results for the simulation match well with those given by the cost function. There is an apparent anomaly seen here, in that, the cost for the SP algorithm is vastly increased even though the average distance for a route is reduced as compared to the uniform distribution. The reason for this is that, although the route length is longer in Experiment 1, the number of different paths to nodes in an outer hexagon is substantially larger. Hence, because of the uniform distributions for sources and destinations, the SP algorithm performed quite well. However, for the nonuniform distribution, where the average number of possible shortest paths for a flow is smaller, the effects

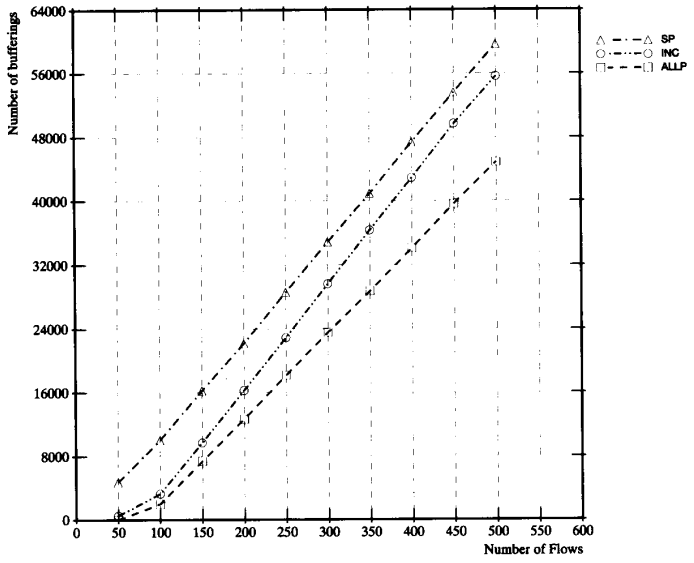


Fig. 5. Comparison using number of bufferings: E-5 mesh, uniform distribution.

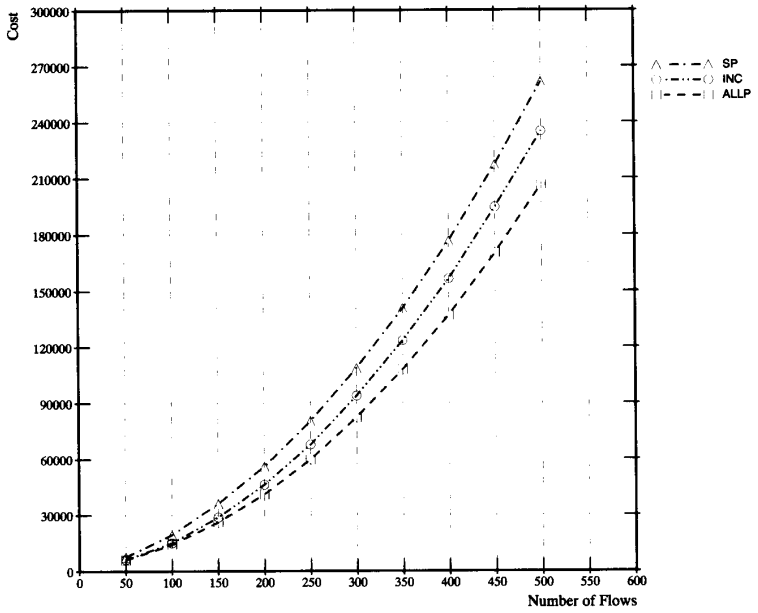


Fig. 6. Comparison using the cost function: E-5 mesh, uniform distribution.

of congestion are more pronounced and so the performance for SP deteriorates. The INC and ALLP algorithms are better equipped to cope with congestion and this is reflected in their vastly better performance than SP.

From these experiments, we can see that a route assignment which reduces the cost function also results in a reduction of the number of bufferings in a hexagonal mesh network.

Other Experiments: To evaluate the performance of the routing algorithms, we have to test them on other topologies and other mesh sizes. However, the computing resources required for comparing the performance using the number of bufferings is prohibitive. The mesh simulator ran on a SUN Sparcstation 1, and the time required for the simulation of a single set of routes was several minutes. Consequently, we

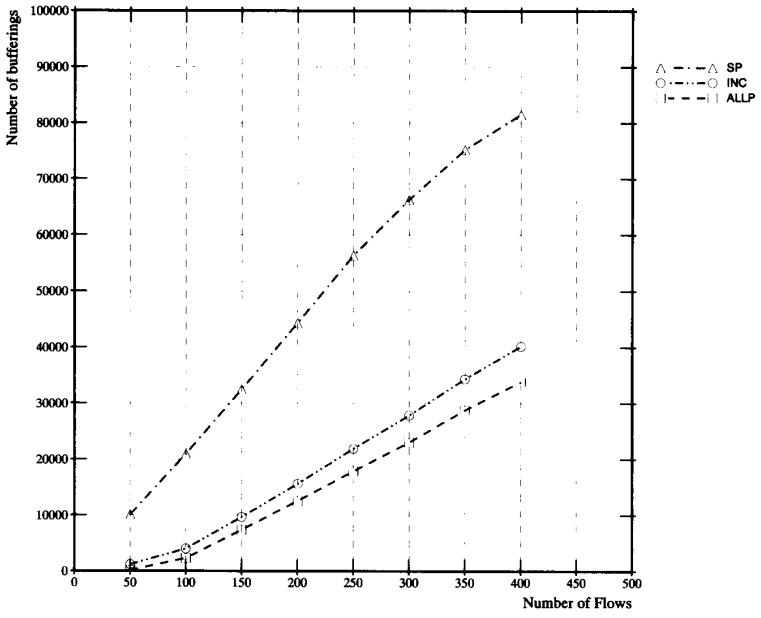


Fig. 7. Comparison using number of bufferings: E-5 mesh, nonuniform distribution.

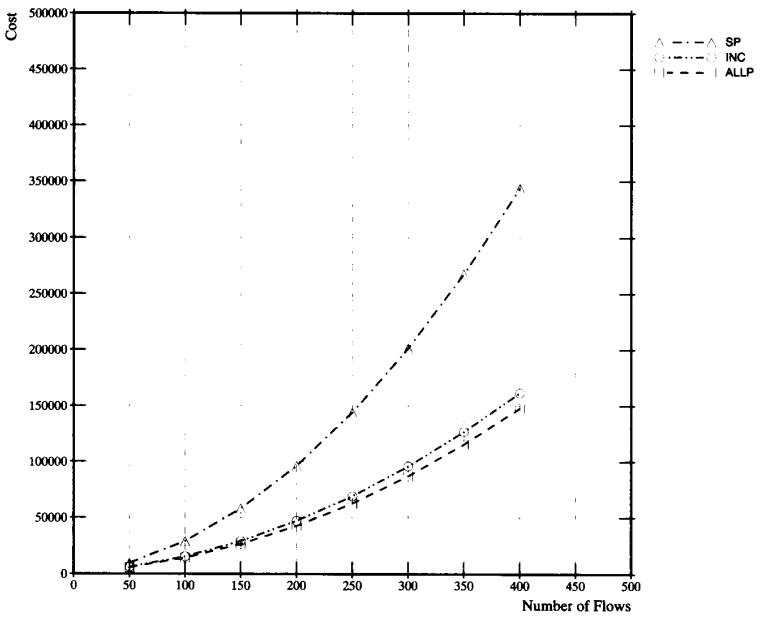


Fig. 8. Comparison using the cost function: E-5 mesh, nonuniform distribution.

evaluate the algorithms based only on the cost function. We have tested the routing algorithms on two network topologies: binary hypercubes and hexagonal meshes, and for different network sizes. This choice was motivated by the fact that variations of virtual cut-through switching have been implemented for these topologies. The binary hypercube is a well-studied topology [5] and is used in many commercially available multicomputer systems, hence its description is omitted.

The performance of these algorithms has been studied using simulated traffic patterns which were generated as described

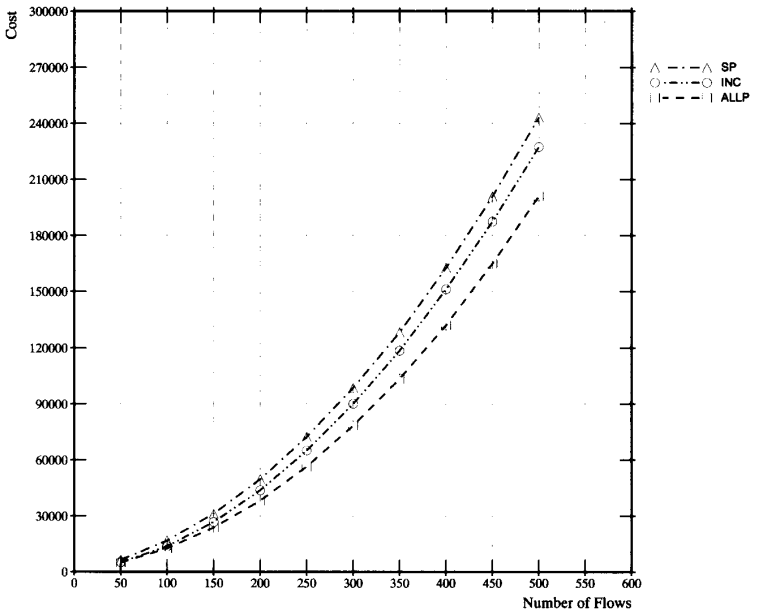


Fig. 9. Cost comparison: E-4 mesh, uniform distribution.

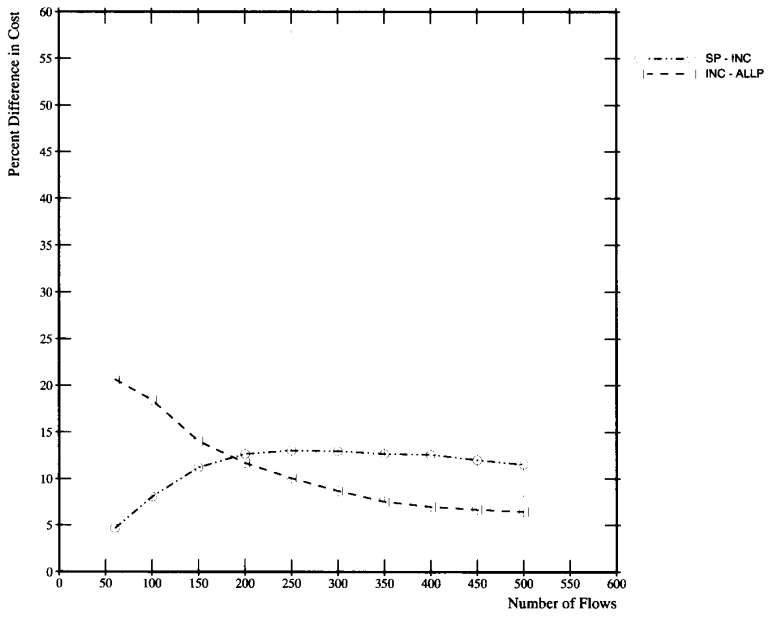


Fig. 10. Performance improvement: E-4 mesh, uniform distribution.

earlier. For each data point, the experiment was repeated with 100 different sets of flows and an average value was obtained. The standard deviation of the mean was found to be less than 5% of the mean for all data points, and less than 3% for most data points.

Figs. 9 and 10 show the results of the simulation on a hexagonal mesh of size 4 (denoted by E-4), having 37 nodes, using a uniform distribution for the selection of the destination node. A comparison of the total cost for the solutions obtained by the three algorithms is shown in Fig. 9, and the percent

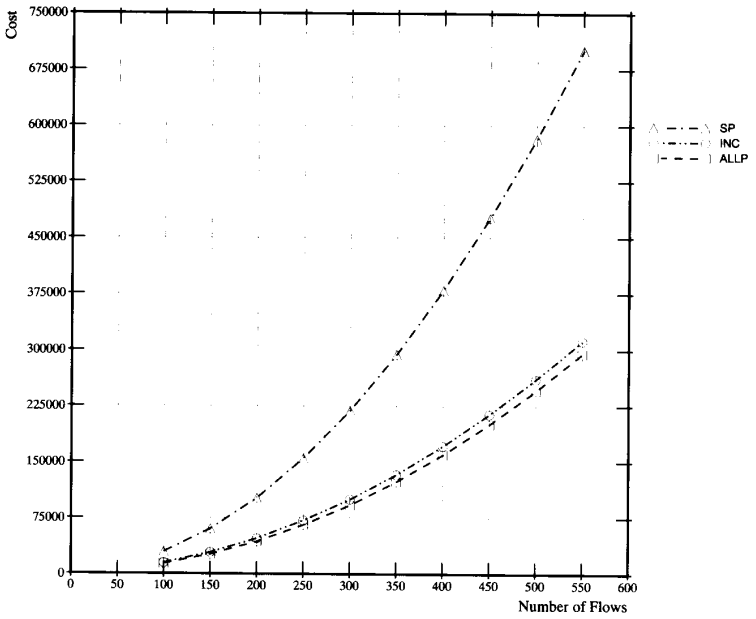


Fig. 11. Cost comparison: E-4 mesh, nonuniform distribution.

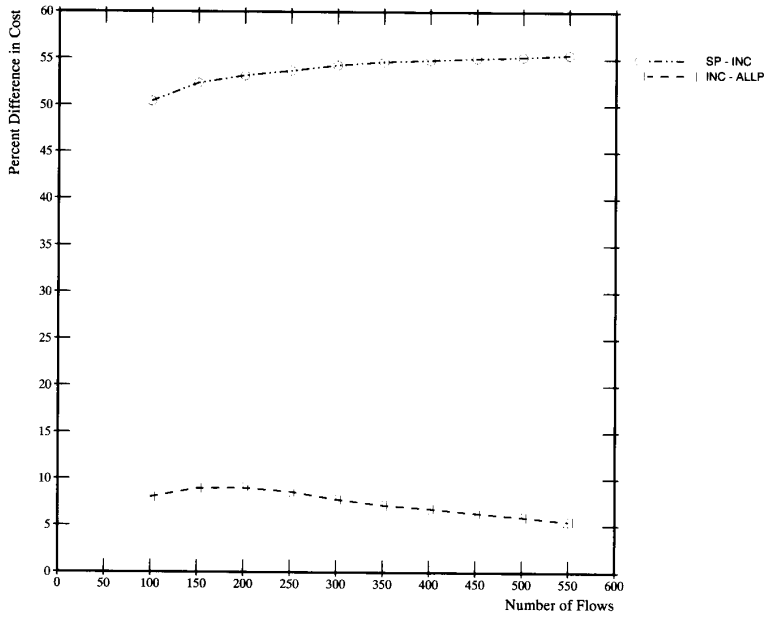


Fig. 12. Performance improvement: E-4 mesh, nonuniform distribution.

improvement in cost between SP and INC, and between INC and ALLP, is shown in Fig. 10. These results are similar to those obtained for the E-5 mesh in Experiment 1. In the second set of experiments for the hexagonal mesh, the destination node was selected using a nonuniform distribution as in Experiment 2. The simulation results obtained with this nonuniform distribution for an E-4 mesh are shown in Figs. 11 and 12, and they are similar to the results obtained for the E-5 mesh in Experiment 2. We also performed experiments on an E-6 mesh, with similar results.

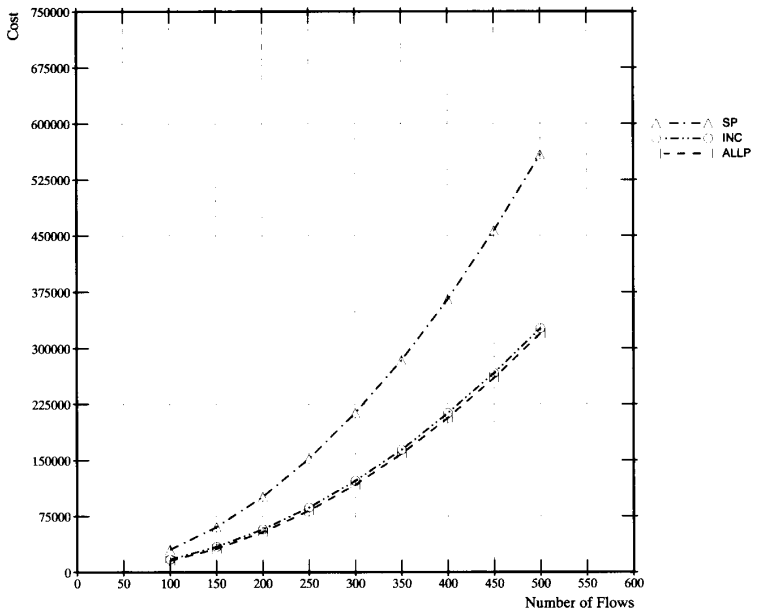


Fig. 13. Cost comparison: Q-5 hypercube, uniform distribution.

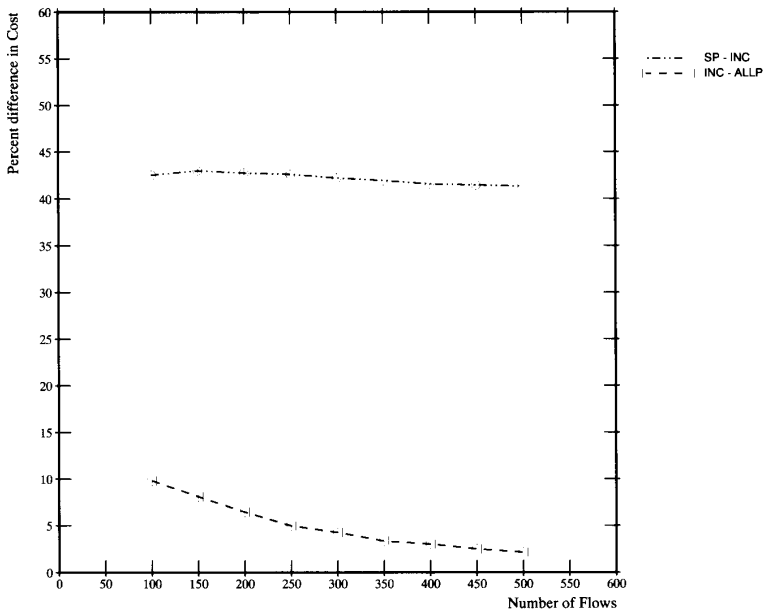


Fig. 14. Performance improvement: Q-5 hypercube, uniform distribution.

The experiments with the binary hypercube used a uniform distribution for the selection of source and destination nodes, but in this case, the average distance between source and destination is $n/2$ for a hypercube of dimension n. The results obtained for hypercubes of dimension 5 (Q-5, 32 nodes) and 6 (Q-6, 64 nodes) are shown in Figs. 13 to 16. These are similar to the results for the hexagonal mesh with a nonuniform source-destination distribution. They also show that the INC and ALLP algorithms perform much better than the SP algorithm. ALLP again shows a modest improvement over INC.

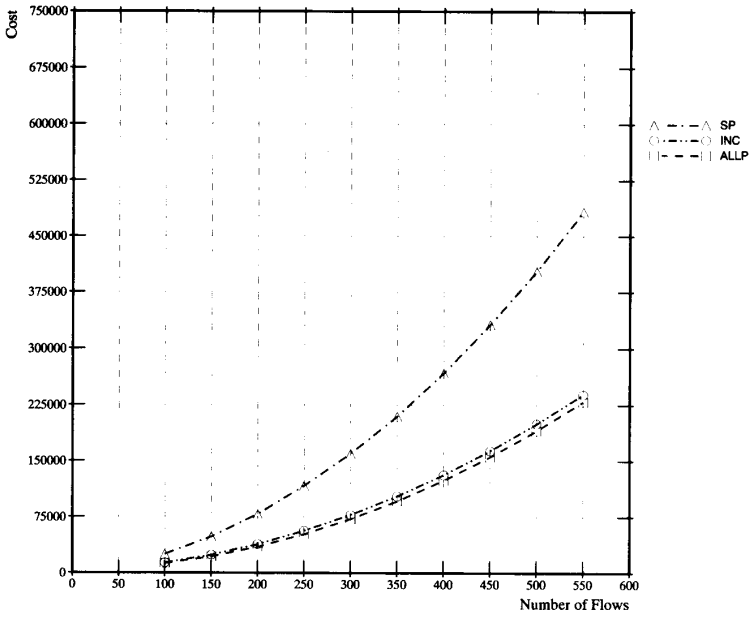


Fig. 15. Cost comparison: Q-6 hypercube, uniform distribution.

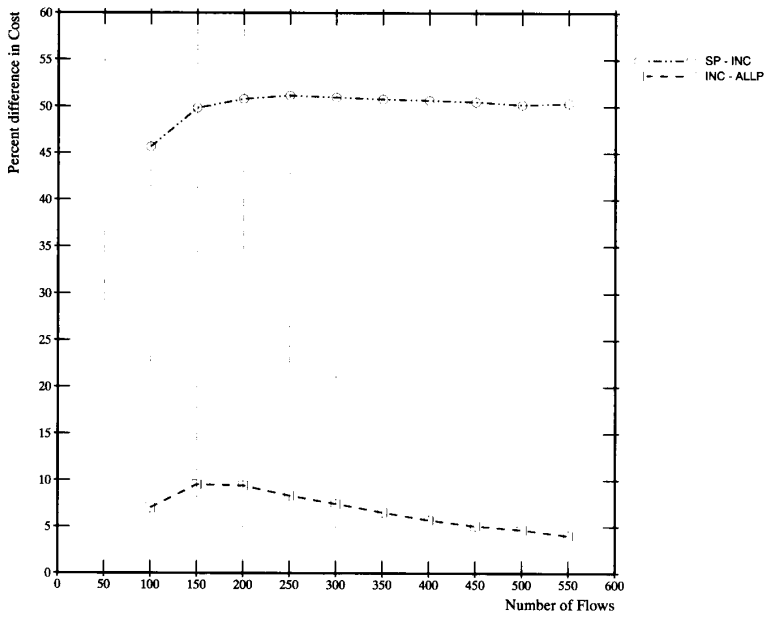


Fig. 16. Performance improvement: Q-6 hypercube, uniform distribution.

In Algorithm Route_All, the final results can depend on the initial state of the network. To study the sensitivity of the results to the initial flow assignment, three different strategies were used to select this assignment. The first method was to select a route using the shortest path algorithm without considering the other routes in the network. The second method used the Route_One algorithm, with routes chosen in the order in which the flows were generated. In the third strategy, the flows were sorted in the order of cost before using the Route_One algorithm for assignment. Both the

ascending and the descending order of cost were considered. A comparison of these strategies was made for an E-4 mesh using the same set of flows as input, and the experiment was repeated several times with different sets of flows. The results indicated that, on the average, there was very little difference in the final cost.

VII. Conclusions

The problem of routing inter-processor message traffic in a point-to-point interconnection network has been formulated as an optimization problem for the total cost, where the cost of a route depends upon the links that are used. The link cost function $c_{ij} = f_{ij}$ was chosen with the objective of maximizing the probability of establishing virtual cut-through routes in the network using analysis based on a queueing model for the network. It is noted that although some of the assumptions made in the queueing analysis may not be valid in a real-time system, the cost function obtained intuitively captures the notion of congestion avoidance. This was verified in our experiments with the hexagonal mesh simulator.

The optimization problem was shown to be $\mathcal{NP}$-Hard, and thus, a heuristic algorithm was developed for the solution. The performance of this algorithm was studied for the binary hypercube and the hexagonal mesh network topologies using simulation. It was found that the extent of the performance improvement depends upon the network topology and the distribution of the source-destination pairs in the network.

The ALLP algorithm can be used for selecting routes off-line in conjunction with an algorithm for assignment of tasks to the nodes. The cost function of ALLP can then be incorporated into the cost of the task assignment to reflect the cost of communication. On the other hand, the INC algorithm can be used to select routes for flows in a dynamic environment when requests are serviced as they arrive. It is optimal for the cost function derived here under the condition that routes that are already established cannot be disturbed. We have used this algorithm to select routes for real-time channels, as described in [21]. One possible extension of this work is to consider routing for multicast communication and to integrate this with the current algorithm. This problem is significantly different from the one treated so far because we are now trying to increase the overlap between paths from a node to different destinations.

Appendix

A. The Satisfiability Problem

Let $U = \{x_1, x_2, \cdots, x_m\}$ be a set of Boolean variables. If u is a variable in U, then u and $\overline{u}$ are *literals* over U. A truth assignment for U is a function $t : U \rightarrow \{T, F\}$, which corresponds to *true* and *false*, respectively. The literal u is true under t if and only if the variable u is true under t. Similarly, the literal $\overline{u}$ is true if and only if the variable u is false.

A *clause* over U is a set of literals over U, and represents the disjunction of those literals. It is *satisfied* by a truth assignment if and only if at least one of its members is true under that assignment. A collection C of clauses over U is *satisfiable* if and only if there exists some truth assignment for U that simultaneously satisfies all the clauses in C. With this definitions, the SATISFIABILITY problem can be stated as follows. It was shown to be $\mathcal{NP}$-Complete by Cook [17].

SATISFIABILITY: Given a set U of variables and a collection C of clauses over U. Is there a truth assignment for U such that C is satisfiable?

B. Proof of Theorem 1

Theorem: Decision Problem 2 (DP2) is $\mathcal{NP}$-Complete.

Proof: It is easy to see that DP2 is in $\mathcal{NP}$ because given a guess for the set of paths, it is possible to compute and verify that the total cost is within the bound B, in polynomial time. We now show that SAT is reducible to DP2.

Given an instance of SAT with collection $C = \{C_1, C_2, \cdots, C_\ell\}$ of clauses on a finite set U of variables. For each variable x_i, let t_i be the number of occurrences of x_i in the clauses, and u_i be the number of occurrences of $\overline{x_i}$. Let $w_i = \max(t_i, u_i)$. Construct a graph component G_i corresponding to this variable as shown in Fig. 3. The component consists of a start node v_s^i, an end node v_t^i, and two chains of nodes, each with $2w_i$ nodes. The chains represent a choice of truth assignment for the variable. These components are then connected in series with the end node of component G_i connected to the start node of component G_{i+1}, that is, v_t^i connected to v_s^{i+1}. Create special nodes s and d and links from s to v_s^1, $v_t^{|U|}$ to d.

For each clause C_i create two nodes, s_i and d_i. For the jth occurrence of literal x_i, say in clause C_ℓ, create links from from s_ℓ to v_{2j-1}^i and from v_{2j}^i to d_ℓ. A similar construction is used for an occurrence of $\overline{x_i}$, but in that case, the nodes v_{2j-1}^i and v_{2j}^i would be used. The graph components and the additional links described above, together form the sets N and L. The set of flows is chosen to be $Q = \{(s, d, 1), (s_1, d_1, 1), \cdots, (s_{|C|}, d_{|C|}, 1)\}$. Finally, the bound B is selected to be $3|C| + 1 + \sum_{i=1}^{|U|}(2w_i + 2)$. This bound is in fact the total cost of routing the set of flows Q, where each link has *exactly one* flow. The $3|C|$ part corresponds to the flows associated with the clauses, and the $(1 + \sum_{i=1}^{|U|}(2w_i + 2))$ part corresponds to the $(s, d, 1)$ flow. This gives an instance of DP2.

If the instance of SAT has a feasible solution, there is a truth assignment to the variables which satisfies all the clauses. In the corresponding instance of DP2, the following solution is feasible. The path from s to d is chosen such that if x_i is assigned a FALSE value, the upper trail $(v_1^i, v_2^i, \cdots)$ is chosen through the component G_i, and vice versa. In each clause C_j, there exists a term u_k^j which evaluates to TRUE and the link in the graph component corresponding to this term, say e_j, would be unused. The path for (s_j, d_j) can be chosen to go through this link. It can be verified that this choice of paths gives a feasible solution for DP2.

Similarly, given a solution to DP2, the path from s to d through the component G_i determines the truth assignment for the corresponding variable. The choice of B forces the paths selected to be disjoint. Otherwise, if a link were used in more than one path, its cost will be greater than 1 and the

total cost will exceed B. This ensures that a variable cannot be used with conflicting values in different clauses, and the path (s_j, d_j) identifies a term in C_j which has a TRUE value. Thus, it is clear that the instance of SAT has a feasible truth assignment *iff* there is a feasible set of paths for the instance of DP2. □

REFERENCES

[1] J. Goldberg *et al.*, "Development and analysis of SIFT," NASA contractor rep. 172146, NASA Langley Research Center, Feb. 1984.

[2] R. M. Kieckhafer, C. J. Walter, A. M. Finn, and P. M. Thambidurai, "The MAFT architecture for distributed fault tolerance," *IEEE Trans. Comput.*, vol. C-37, pp. 398–405, Apr. 1988.

[3] J. Lala, "AIPS tutorial," tech. rep., The Charles Stark Draper Laboratory, Inc., Jan. 1987.

[4] "Completion of the advanced information processing system," The Charles Stark Draper Laboratory, report in response to NASA Langley Research Center CBD announcement Ref SS017, Issue PSA-9214.

[5] C. L. Seitz, "The cosmic cube," *Commun. ACM*, vol. 28, pp. 22–33, Jan. 1985.

[6] M.-S. Chen, K. G. Shin, and D. D. Kandlur, "Addressing, routing, and broadcasting in hexagonal mesh multiprocessors," *IEEE Trans. Comput.*, vol. C-39, pp. 10–18, Jan. 1990.

[7] P. Kermani and L. Kleinrock, "Virtual cut-through: A new computer communication switching technique," *Comput. Networks*, vol. 3, pp. 267–286, 1979.

[8] W. J. Dally and C. L. Seitz, "The torus routing chip," *J. Distributed Syst.*, vol. 1, no. 3, pp. 187–196, 1986.

[9] W. J. Dally and P. Song, "Design of a self-timed VLSI multicomputer communication controller," in *Proc. IEEE Int. Conf. Comput. Design: VLSI in Comput.*, 1987, pp. 230–234.

[10] J. W. Dolter, P. Ramanathan, and K. G. Shin, "A microprogrammable VLSI routing controller for HARTS," in *Proc. IEEE Int. Conf. Comput. Design: VLSI in Comput.*, IEEE, Oct. 1989, pp. 160–163.

[11] D. Ferrari, "Guaranteeing performance for real-time communication in wide-area networks," Tech. Rep. UCB/CSD 89/485, U.C.B. Computer Science Division, EECS, Berkeley, CA, Jan. 1989.

[12] M. Schwartz and T. E. Stern, "Routing techniques used in computer communication networks," *IEEE Trans. Commun.*, vol. 28, pp. 539–552, Apr. 1980.

[13] R. P. Bianchini and J. P. Shen, "Interprocessor traffic scheduling algorithm for multiple-processor networks," *IEEE Trans. Comput.*, vol. C-36, pp. 396–409, Apr. 1987.

[14] L. Kleinrock, *Communication Nets: Stochastic Message Flow and Delay*. New York: McGraw-Hill, 1964.

[15] M. Ilyas and H. T. Mouftah, "Toward performance improvement of cut-through switching in computer networks," *Perform. Eval.*, vol. 6, pp. 125–133, July 1986.

[16] J. R. Jackson, "Networks of waiting lines," *Oper. Res.*, vol. 5, pp. 518–521, Aug. 1957.

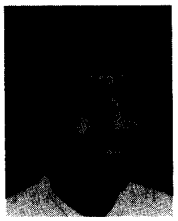
[17] S. A. Cook, "The complexity of theorem-proving procedures," in *Proc. 3rd Annu. ACM Symp. Theory of Comput.*, 1971, pp. 151–158.

[18] M. R. Garey and D. S. Johnson, *Computers and Intractability*. San Francisco, CA: Freeman, 1979.

[19] S. Even, A. Itai, and A. Shamir, "On the complexity of timetable and multicommodity flow problems," *SIAM J. Comput.*, vol. 5, pp. 691–703, Dec. 1976.

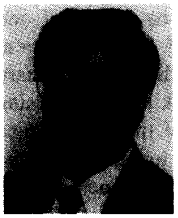
[20] J. W. Dolter, P. Ramanathan, and K. G. Shin, "Performance analysis of virtual cut-through switching in HARTS: A hexagonal mesh multicomputer," *IEEE Trans. Comput.*, vol. 40, pp. 669–680, June 1991.

[21] D. D. Kandlur and K. G. Shin, "A communication subsystem for HARTS: An experimental distributed real-time system," Tech. Rep., CSE-TR-109-91, CSE Div., EECS Dep., Unv. of Michigan, Ann Arbor, Oct. 1991.



Dilip D. Kandlur (S'90–M'91) received the B.Tech. degree in computer science and engineering from the Indian Institute of Technology, Bombay, in 1985 and the M.S.E. and Ph.D. degrees, also in computer science and engineering, from the University of Michigan, Ann Arbor, in 1987 and 1991 respectively.

From 1987 to 1991 he was a member of the Real-Time Computing Laboratory at the University of Michigan, involved in the development of the HARTS experimental real-time system. He was also the recipient of an IBM Graduate Fellowship. Currently, he is a research staff member at the IBM Thomas J. Watson Research Center, Yorktown Heights, NY. His research interests include operating systems, real-time systems, and computer networks.



Kang G. Shin (S'75–M'78–SM'83–F'92) received the B.S. degree in electronics engineering from Seoul National University, Seoul, Korea, in 1970, and both the M.S. and Ph.D degrees in electrical engineering from Cornell University, Ithaca, NY in 1976 and 1978, respectively.

He is Professor and Associate Chair of Electrical Engineering and Computer Science (EECS) for Computer Science and Engineering, The University of Michigan, Ann Arbor. He has authored/coauthored over 230 technical papers (about 100 of these in archival journals) and several book chapters in the areas of distributed real-time computing and control, fault-tolerant computing, computer architecture, and robotics and automation. In 1985, he founded the Real-Time Computing Laboratory, where he and his colleagues are currently building a 19-node hexagonal mesh multicomputer, called **HARTS**, to validate various architectures and analytic results in the area of distributed real-time computing. From 1978 to 1982 he was on the faculty of Rensselaer Polytechnic Institute, Troy, NY. He has held visiting positions at the U.S. Airforce Flight Dynamics Laboratory, AT&T Bell Laboratories, Computer Science Division within the Department of Electrical Engineering and Computer Science at U.C. Berkeley, and International Computer Science Institute, Berkeley, CA.

Dr. Shin was the Program Chairman of the 1986 IEEE Real-Time Systems Symposium (RTSS), the General Chairman of the 1987 RTSS, the Guest Editor of the 1987 August special issue of IEEE TRANSACTIONS ON COMPUTERS on Real-Time Systems, and is a Program Co-Chair for the 1992 International Conference on Parallel Processing. He currently chairs the IEEE Technical Committee on Real-Time Systems, is a Distinguished Visitor of the Computer Society of the IEEE, an Editor of IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS, and an Area Editor of *International Journal of Time-Critical Computing Systems*. In 1987, he received the Outstanding IEEE Transactions on Automatic Control Paper Award for a paper on robot trajectory planning. In 1989, he also received the Research Excellence Award from The University of Michigan.