A Distributed Routing Algorithm for Virtual Circuit Data Networks

Yeong-Sung Lin[1] and James R. Yee[1,2]
Department of Electrical Engineering
University of Southern California
Los Angeles, CA 90089-0781
(213)-743-3919

Abstract

In this paper, the routing problem in virtual circuit networks is considered. In virtual circuit networks, all of the packets in a session are transmitted over exactly one path established between the origin and the destination. We consider the problem of choosing a path for each origin-destination pair so as to minimize the average number of packets in the network. We consider the usual formulation of this problem as a nonlinear multicommodity flow problem with integer decision variables.

The emphasis of this work is to develop a distributed algorithm to solve this optimization problem. The basic approach in this work is Lagrangean Relaxation which has been a common and successful technique in solving many difficult combinatorial optimization problems. We develop a multiplier update technique which facilitates the solution of the nonlinear integer programming problem using distributed computation. In computational experiments, our distributed algorithm determines solutions that are within 1% of an optimal solution in minutes of CPU time for networks with 26 to 61 nodes.

1 Introduction

Computer communications networks play an important role in satisfying our communication and computational needs. The applications of computer communications networks include but are not limited to electronic mail, telephone networks, cellular phone systems, airline reservation systems, automated teller machines, tactical military C^3 systems, etc.. In order for a computer network to operate efficiently and reliably, it is essential that the routing algorithm be carefully designed. For this reason, the routing problem has been studied intensively [22,23].

An overwhelming majority of the research literature on routing assumes datagram service. Two primary reasons for this are (i) the ARPANET [18], which is a datagram network, inspired a large amount of the research and (ii) the routing problem in datagram networks can be solved by utilizing many standard convex programming techniques [1,5]. Whereas, the routing problem in virtual circuit involves solving a nonlinear combinatorial optimization problem which is more difficult. In this paper, we focus on virtual circuit networks. Our motivation is that many networks of today (e.g., SNA [11,15], TYMNET [20,27], TELENET [12], and TRANSPAC [3]) are virtual circuit networks. Furthermore, networks of the future such as ISDN will provide virtual circuit service [10]. The major advantage of virtual circuit switching over datagram switching is that all packets in a particular session arrive in the same order in which they are sent. It is well-known that a distributed algorithm is better than a centralized algorithm since the use of a distributed algorithm results in greater reliability.

Although virtual circuit switching techniques are used so widely, the number of papers on routing in virtual circuit networks is small relative to the number of papers on routing in datagram networks. Segall [24] formulated the routing problem in virtual circuit networks as a convex programming problem and extended Gallager's algorithm [6] to develop a distributed routing algorithm. Courtois and Semal [2] modified Fratta, Gerla and Kleinrock's Flow Deviation method [5] to develop a heuristic routing algorithm for virtual circuit networks. They obtained solutions that are within 3% (on the average) of the optimal solutions for lightly loaded networks. Gavish and Hantler [8] formulated the problem of optimal route selection in virtual circuit networks to minimize average delay as a nonlinear multicommodity flow problem with 0-1 decision variables. They applied Lagrangean Relaxation and the subgradient optimization method to develop a centralized algorithm. Their computational results showed that this algorithm is effective in finding good feasible solutions and determining tight lower bounds on the minimal expected delay.

Lagrangean Relaxation has been a common and successful technique in solving many difficult combinatorial optimization problems [4,9]. In this technique, the dual problem is solved to provide a lower bound (for minimization problems) on the optimal objective function value. This dual problem is usually solved by the subgradient method.

[1]Supported in part by Contract No. N00228-87-R-4196 from the Naval Postgraduate School.

[2]Supported in part by a summer faculty member fellowship at the Jet Propulsion Laboratory at the California Institute of Technology.

The standard update rule [14] for the dual variables was applied in [8]. However, a distributed implementation of this multiplier update rule results in more overhead messages than desired. In this paper, we introduce a new multiplier adjustment procedure which requires far fewer overhead messages in a distributed implementation. Moreover, the distributed routing algorithm with our multiplier update finds solutions that are closer to an optimal solution than the usual multipler update rule.

The remainder of this paper is organized as follows. In Section 2, the routing problem is formulated as a nonlinear combinatorial optimization problem. In Section 3, a Lagrangean Relaxation approach to the problem is presented. In Section 4, a distributed multiplier adjustment procedure is described. In Section 5, a heuristic algorithm called K(0)-ordering is decentralized to provide good initial primal and dual solutions. In Section 6, the computational results are reported.

2 Problem Formulation

A virtual circuit communications network is modeled as a graph where the processors are represented by nodes and the communication channels are represented by arcs. Let $V = \{1, 2, ..., N\}$ be the set of nodes in the graph and let L denote the set of communication links in the network. Let W be the set of origin-destination (O-D) pairs (commodities) in the network. For each O-D pair $w \in W$, the arrival of new traffic is modeled as a Poisson process with rate γ_w (packets/sec). Then the arrival of new traffic to the network is a Poisson process with rate $\Gamma = \sum_w \gamma_w$. For O-D pair w, the traffic for a particular session is transmitted over one path in the set P_w, a given set of simple directed paths from the origin to the destination of O-D pair w. Let P be the set of all simple directed paths in the network. For each link $l \in L$, the capacity is C_l packets/sec.

For each O-D pair $w \in W$, let x_p be 1 when path $p \in P_w$ is used to transmit the packets in a session and 0 otherwise. In a virtual circuit network, all of the packets in a session are transmitted over one path from the origin to the destination. Thus $\sum_{p \in P_w} x_p = 1$. For each path p and link $l \in L$, let δ_{pl} denote an indicator function which is one if link l is on path p and zero otherwise. Then, the aggregate flow of packets over link l is given by the left hand side of (1).

In the network, there is a buffer for each outbound link. Using Kleinrock's independence assumption [16], the arrival of packets to each buffer is a Poisson process where the rate is the aggregate flow over the outbound link. It is assumed that the transmission time for each packet is exponentially distributed with mean C_l^{-1}. Thus, each buffer is modeled as an M/M/1 queue.

The problem of determining a path for each O-D pair to minimize the average delay is formulated as the following nonlinear combinatorial optimization problem.

$$Z_{\overline{IP}} = \min \frac{1}{\Gamma} \sum_{l \in L} \frac{\sum_{w \in W} \sum_{p \in P_w} x_p \gamma_w \delta_{pl}}{C_l - \sum_{w \in W} \sum_{p \in P_w} x_p \gamma_w \delta_{pl}} \qquad (\overline{IP})$$

s.t.

$$\sum_{w \in W} \sum_{p \in P_w} x_p \gamma_w \delta_{pl} \leq C_l \qquad \forall l \in L \quad (1)$$

$$\sum_{p \in P_w} x_p = 1 \qquad \forall w \in W \quad (2)$$

$$x_p = 0 \text{ or } 1 \qquad \forall p \in P_w \quad (3)$$

The objective function represents the average delay for a packet in the network. Constraint (1) requires that the aggregate flow not exceed the capacity for each link. Constraints (2) and (3) require that all of the traffic in each session be transmitted over exactly one path.

An equivalent formulation of the above problem is given by (IP) below. We redefine x_p to be rate at which packets in a session for O-D pair $w \in W$ are transmitted over path $p \in P_w$. (IP) is better suited for the development of a distributed algorithm and the application of the Lagrangean Relaxation method. We will elaborate further on this statement at the end of the next section.

$$Z_{IP} = \min \sum_{l \in L} \frac{f_l}{C_l - f_l} \qquad (IP)$$

s.t.

$$\sum_{w \in W} \sum_{p \in P_w} x_p \delta_{pl} \leq f_l \qquad \forall l \in L \quad (4)$$

$$0 \leq f_l \leq C_l \qquad \forall l \in L \quad (5)$$

$$\sum_{p \in P_w} x_p = \gamma_w \qquad \forall w \in W \quad (6)$$

$$x_p = 0 \text{ or } \gamma_w \qquad \forall p \in P_w, \; w \in W \quad (7)$$

For each link l, a variable f_l is introduced. We interpret these variables to be "estimates" of the aggregate flows. Since the objective function is strictly increasing with f_l and (IP) is a minimization problem, each f_l will equal the aggregate flow in an optimal solution. As the reader will see in the next section, the introduction of $\{f_l\}$ decouples the problem into two subproblems in the Lagrangean Relaxation.

3 Lagrangean Relaxation and Dual Problem

Lagrangean relaxation is a method used for obtaining lower bounds (for minimization problems) as well as good primal solutions in integer programming problems. A Lagrangean relaxation (LR) is obtained by identifying in the primal problem a set of complicating constraints whose removal will simplify the solution of the primal problem. Each of the complicating constraints is multiplied by a multiplier and added to the objective function. This mechanism is referred to as dualizing the complicating constraints. Since our ob-

jective is to develop a distributed algorithm, the choice of constraints to dualize will be made so that the resulting Lagrangean relaxation can be solved by distributed computation.

Lagrangean relaxation has been applied to obtain excellent heuristic solutions and tight lower bounds for the traveling salesman problem [13], the concentrator location problem [19], a topological design problem in centralized computer networks [7] and many other NP-hard problems. In addition, Gavish and Hantler [8] have successfully applied this technique to develop a centralized routing algorithm for virtual circuit networks. As in [8], we dualize constraint (4) to obtain the following relaxation

$$Z_D(u) = \min \sum_{l \in L} \frac{f_l}{C_l - f_l} + \sum_{l \in L} u_l \left\{ \sum_{w \in W} \sum_{p \in P_w} x_p \delta_{pl} - f_l \right\} \quad (LR)$$

s.t.

$$0 \leq f_l \leq C_l \qquad \forall l \in L \qquad (8)$$

$$\sum_{p \in P_w} x_p = \gamma_w \qquad \forall w \in W \qquad (9)$$

$$x_p = 0 \text{ or } \gamma_w \qquad \forall p \in P_w, \; w \in W. \qquad (10)$$

Note that (LR) is composed of the following two independent subproblems:

Subproblem 1:

$$Z_D^1(u) = \min \sum_{l \in L} \left\{ \frac{f_l}{C_l - f_l} - u_l f_l \right\}$$

s.t.

$$0 \leq f_l \leq C_l \qquad \forall l \in L \qquad (11)$$

and

Subproblem 2:

$$Z_D^2(u) = \min \left\{ \sum_{w \in W} \sum_{p \in P_w} x_p \left\{ \sum_{l \in L} u_l \delta_{pl} \right\} \right\}$$

s.t.

$$\sum_{p \in P_w} x_p = \gamma_w \qquad \forall w \in W \qquad (12)$$

$$x_p = 0 \text{ or } \gamma_w \qquad \forall p \in P_w, \; w \in W. \qquad (13)$$

Subproblem 1 is composed of $|L|$ (one for each link) simpler problems. Each of these simple problems is the minimization of a convex univariate function over a simple interval. For each link $l \in L$, the solution is

$$\bar{f}_l = \begin{cases} C_l(1 - \sqrt{\frac{1}{u_l C_l}}) & \text{if } u_l > \frac{1}{C_l} \\ 0 & \text{otherwise.} \end{cases} \qquad (14)$$

It is clear that the $\{\bar{f}_l\}$ can be computed using a distributed algorithm. If link l connects node i to node k, then we will refer to node i as the tail and node k as the head of link l, i.e. $i = \text{tail}(l)$ and $k = \text{head}(l)$. In the distributed routing protocol (to be described), tail(l) will compute a multiplier u_l associated with link l. From this and knowing C_l, tail(l) can compute $\bar{f}_l$ from the above equation.

Subproblem 2 consists of $|W|$ (one for each commodity) shortest path problems where u_l is the arc weight for link l. For each origin–destination pair $w \in W$, a shortest path $p_w^1 \in P_w$ is found. Then,

$$\bar{x}_p = \begin{cases} \gamma_w & \text{if } p = p_w^1 \\ 0 & \text{if } p \neq p_w^1. \end{cases} \qquad (15)$$

There are a number of distributed shortest path algorithms that can be used to solve this problem [18,25,26].

At this point, it would be useful to point out the differences between our formulation (IP) and the formulation in [14, equation (4)]. From a mathematical point of view, the differences are trivial. However, the differences are worth noting with respect to developing a distributed algorithm. First, we omitted the term $\frac{1}{\Gamma}$. Of course, this means that we are minimizing the average number of packets in the network as compared to minimizing the average delay in [8]. By including the term $\frac{1}{\Gamma}$, $\bar{f}_l$ would be a function of $\Gamma = \sum_w \gamma_w$. This would then require an extra number of overhead messages for tail(l) to estimate Γ and to compute $\bar{f}_l$. Second, we specified the feasible values of x_p, $p \in P_w$, as $\{0, \gamma_w\}$ instead of $\{0, 1\}$ as in [8]. Note in subproblem 2 above all commodities solve a shortest path problem with respect to one weighted graph where the arc weights are $\{u_l\}$. By dualizing constraint (1), the Lagrangean Relaxation "appears" to include $|W|$ different shortest path problems (for each $w \in W$, there is a weighted graph with arc weights $\{u_l \gamma_w\}$). In reality, one would make an adjustment in the implementation so there is just one weighted graph. But, the formulation (IP) above is cleaner. Third, we interpret f_l as a link flow, whereas in [8], it is interpreted as a link utilization. We prefer to be consistent with the routing literature where routing variables are either flows or probabilities.

For any $u \geq 0$, the optimal objective function value of (LR), $Z_D(u)$, is a lower bound on Z_{IP} [9]. Naturally, one wants to determine the greatest lower bound by

$$Z_D = \max_{u \geq 0} Z_D(u) = \max_{u \geq 0} Z_D^1(u) + Z_D^2(u) \qquad (D)$$

There are a number of properties of $Z_D(u)$ which should be noted. First, $Z_D(u)$ is a concave, nondifferentiable function over the convex set $\{u | u \geq 0\}$. Second, the vector with the l^{th} element equal to $(\sum_{w \in W} \sum_{p \in P_w} \bar{x}_p \delta_{pl} - \bar{f}_l)$ is a subgradient of $Z_D(u)$ at the point u, where $\bar{f}_l$ and $\bar{x}_p$ are the solutions to (LR) [14]. Third, $Z_D(u)$ is not a piecewise linear function due to the nonlinearity of the objective function of the original problem (IP). In most combinatorial optimization problems addressed in the literature, the objective function is linear. Consequently, the corresponding $Z_D(u)$ is a piecewise linear, nondifferentiable concave function.

There are several methods for solving the dual problem

(D). The most popular method is the subgradient method [4,9,14]. Let an $|L|$ vector y be a subgradient of $Z_D(u)$. In iteration k of the subgradient optimization procedure, the multiplier for each link $l \in L$ is updated by

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

The step size t^k is determined by

$$t^k = \delta \frac{Z_{IP}^h - Z_D(u^k)}{\|y^k\|^2} \quad (16)$$

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

In Section 5, a distributed algorithm is given to compute Z_{IP}^h. Both $Z_D(u^k)$ and $\|y^k\|^2$ can be expressed in the form $\sum_{l \in L} \beta_l$. In [28], an efficient distributed protocol (based upon using an arborescence) is given. Thus, the subgradient method can be distributed. However, in the next section, we present a distributed algorithm to solve (D) based upon a multiplier adjustment procedure.

4 A Multiplier Adjustment Procedure

In this section, a new multiplier adjustment procedure is introduced. The standard multiplier update rule [14] given in (16) was used by Gavish and Hantler [8] to develop a centralized algorithm for virtual circuit networks. For the purposes of reliability, we focus on developing a distributed algorithm. The multiplier adjustment rule presented below (i) has a lower communication complexity than (16) and (ii) results in a distributed algorithm that finds solutions that are closer to an optimal solution than an algorithm using (16).

For iteration k, let u_l^k be the multiplier for link l and let f_l^k be the solution to subproblem 1. Let g_l^k be the aggregate flow on link l determined from $\{x_p\}$, the solution to subproblem 2. The l^{th} component of the subgradient in iteration k is denoted as y_l^k. The basic idea in our multiplier adjustment scheme is to choose u^{k+1} so that $|y^{k+1}| \leq |y^k|$. In other words, we choose u^{k+1} so that the distance between f_l^{k+1} and g_l^{k+1} is less than or equal to f_l^k and g_l^k. Let $m_k \geq 1$ be a parameter used to control the stepsize. We choose f_l^{k+1} to be the following convex combination of f_l^k and g_l^k

$$f_l^{k+1} = \frac{(m_k - 1) f_l^k + g_l^k}{m_k} \quad (17)$$

$$= f_l^k + \frac{1}{m_k}(g_l^k - f_l^k). \quad (18)$$

Repeating equation (14) for iteration $k+1$,

$$f_l^{k+1} = C_l\,(1 - \sqrt{\frac{1}{u_l^{k+1} C_l}}). \quad (19)$$

Consequently, from (18) and (19) the new multiplier is

$$u_l^{k+1} = C_l^{-1}\,[1 - \frac{(m_k - 1) f_l^k + g_l^k}{m_k C_l}]^{-2}. \quad (20)$$

It can be shown that the magnitude of the subgradient tends to decrease from iteration to iteration.

Equation (20) can also be expressed in the following iterative form:

$$u_l^{k+1} = u_l^k + t_l^k (g_l^k - f_l^k). \quad (21)$$

where

$$t_l^k = \frac{C_l[2m_k C_l - (2m_k - 1) f_l^k + g_l^k]}{(C_l - f_l^k)^2 [m_k C_l - (m_k - 1) f_l^k + g_l^k]^2}. \quad (22)$$

From equation (22), it is clear that the parameter m_k will affect the stepsizes. Note that in (22) there is a stepsize for each link. In the usual multiplier adjustment method in (16), there is one stepsize. This is analogous to the distributed routing algorithm for datagram networks in [28] where there is one stepsize for each node.

The above multiplier adjustment method can be implemented either in a one-at-a-time manner, i.e., only one multiplier is adjusted in each iteration, or in an all-at-once manner, i.e., all multipliers are adjusted in each iteration. Also, instead of using a fixed m, we use a monotonically increasing sequence $\{m_k \mid k = 1, 2, ...\}$ where m_k tends to infinity as k approaches infinity. Then from equation (18), f_l^{k+1} will be equal to f_l^k as k (or m_k) approaches infinity. By equation (14), the convergence of $\{f_l^k\}$ also implies that $\{u_l^k\}$ will converge to a limit.

The overall algorithm is

0. Initialize

- 0.a Generate a candidate route set for each O-D pair.
- 0.b Assign a nonnegative value to each multiplier.
- 0.c Set the iteration counter, k, to zero.

1. Test stopping criteria

If the number of iterations has reached the predetermined limit, stop; otherwise, go to step (2).

2. Solve the Lagrangean Relaxation

- 2.a For each link l, each tail(l) calculates f_l^k by using (14).
- 2.b A distributed shortest path algorithm is used to solve subproblem 2 for each O-D pair.
- 2.c For each link l, each tail(l) estimates g_l^k.

3. Adjust the multipliers

- 3.a For each link l, each tail(l) uses (20) to calculate u_l^{k+1}.
- 3.b $k \leftarrow k + 1$.
- 3.c Go to step (1).

Note that steps 2 and 3 can be performed using distributed computation.

5 Initial Solutions

A distributed algorithm is developed to calculate a good initial primal solution. Also, a transformation is designed to construct a mapping of the initial primal solution to the initial dual solution.

First, we introduce a distributed version of the K(0)-ordering [2] to find a starting nonbifurcated flow pattern. All of the commodities are ordered according to their *inertia*, defined by the right hand side of equation (24). Then, according to the ordering, each commodity routes its requirement over the current shortest path. An intuitive interpretation of inertia is the following. If a commodity with a higher inertia is deviated from the original shortest path, a larger increase in the objective function will be incurred. Therefore, a route should be selected for this commodity earlier.

Next, a mapping of the primal solutions to the dual ones is constructed. Since the objective of the dual problem is continuous and concave, if differentiability is assumed, then there is a one-to-one correspondence between the optimal dual solution u^* and the optimal primal solution (x^*, f^*) by the duality theorem [17]. It is also assumed that (x^0, f^0), the initial primal solution, is equal to (x^*, f^*), where f^0 can be expressed in terms of u^0 by using equation (14) and x^0 is obtained by the K(0)-ordering. Finally, by setting the "gradient" at the point u^0 to zero, we can obtain the following equation

$$u_l^0 = \frac{1}{C_l(1 - \sum_{w \in W} \sum_{p \in P_w} \frac{x_p^0 \delta_{pl}}{C_l})^2} \qquad \forall l \in L \qquad (23)$$

to compute a good initial set of multipliers. Note that the above transformation is performed by using distributed computation.

A distributed version of the K(0)-ordering is presented as follows:

1. **Find the shortest paths under zero flows**
   Define the arc weight of link l, $cost(l)$, to be the first derivative of the objective function of (IP) with respect to the aggregate link flow. Then for each $w \in W$, compute $\prod_w^{min}$, the shortest path for commodity w under zero flows.

2. **Calculate the new shortest paths for all commodities when some arc weight is set to infinity**
   Under a cyclic order control protocol, all links take turns to perform the following steps:

   2.a On the turn of link l, tail(l) sets $cost(l)$ to infinity and sends this message to every $CO(w)$, the origin of commodity w, for which $l \in \prod_w^{min}$.

   2.b After receiving this message from tail(l), each $CO(w)$ determines the new shortest path and the associated path cost.

3. **Calculate inertia for each commodity**
   At the end of step (2), every $CO(w)$ can compute its inertia $K_w(0)$ by the following equation:

$$K_w(0) = \gamma_w a_w^{-1} \sum_{j=1}^{a_w} \left\{ \sum_{l \in \prod_w^j} cost(l) - \sum_{l \in \prod_w^{min}} cost(l) \right\} \qquad (24)$$

   where a_w is the number of links along the path $\prod_w^{min}$, $cost(l)$ is evaluated under zero flow, and $\prod_w^j$ is the shortest path from the origin to the destination of commodity w when the j^{th} link of $\prod_w^{min}$ is assigned an infinite cost.

4. **Order the commodities**
   Use some distributed ranking algorithm, e.g., [21,29], to sort the commodities into a decreasing order according to $\{K_w(0)\}$.

5. **Construct an initial flow pattern**
   Under the same cyclic order control protocol as in step (2), each $CO(w)$ computes its routing decision according to the order given in step (4) by using the following algorithm:

   5.a The same as step (1) except that arc weights are evaluated at current flows.

   5.b Route the traffic requirement on the shortest path obtained in step (5.a).

Note that the routing decisions obtained by this distributed K(0)-ordering can serve as a good approximate solution to (IP). However, compared with our distributed routing algorithm, K(0)-ordering has higher complexity and determines higher average delay. Therefore, K(0)-ordering is suggested only for initialization. If more accuracy is needed, then some distributed version of the nonbifurcated FD (flow deviation) algorithm [2,5] can also be developed. The nonbifurcated FD algorithm takes the solution obtained by the K(0)-ordering as an initial solution and can provide a better flow pattern.

For a lightly loaded network, a nonnegative identical initial value for every u_l^0 is a fairly good choice. Another possible choice is $1/C_l$, which is the first derivative of the link cost function evaluated at zero flow. Such strategies might well lead to over-saturated links. However, this problem can be coped with by a carefully designed flow control scheme or a slightly modified objective function [1] to allow the routing algorithm to gradually adapt.

6 Computational Results

The distributed routing algorithm for virtual circuit networks described in Section 4 was coded in FORTRAN 77 and run on a SUN 3/50 workstation. In the multiplier adjustment procedure in section 4, the all-at-once method (all multipliers are adjusted in each iteration) was implemented. Recall that the parameter in the multiplier adjustment procedure that effects the stepsize is m_k. We chose $m_k = (\log_2(k+3))^2$. The maximum number of iterations allowed is 200 iterations. The choice of the initial values of the multipliers was $\{\frac{1}{C_l}\}$ (however we found in our computational experiments that the initial values had little effect on the results).

The algorithm was tested on three networks –ARPA, RING and OCT with 61, 32 and 26 nodes respectively. Their topologies are shown in Figures 1, 2 and 3. For each of the three networks, it is assumed that there is one session per O-D pair and that the rate at which packets are generated is 1 packet/sec for each session. Other characteristics of the test problems are given in Table 1. In the third column, the number of candidate paths for each O-D pair is given. The fourth column specifies the total number of candidates paths (the number of integer decision variables) in the network. The fifth column specifies the capacity of each link in each network. The method of generating the candidate paths for each network is generated as follows. For each O-D pair, shortest paths were found with respect to several sets of randomly generated arc weights.

Table 1 summarizes the results of our computational experiments. The sixth column is largest lower bound on the optimal objective function value in 200 iterations. Recall that this is the best objective function value of the dual problem. The seventh column gives the best objective function value for (IP) in 200 iterations. The percentage difference [(upper-bound – lower-bound)·100/lower-bound] is an upper bound on how far the best feasible solution found is from an optimal solution. The ninth column provides the CPU times which consists of the time to compute a solution plus the time used to input the problem parameters. Furthermore, these reported CPU times measure the total CPU time used when every operation in the algorithm is performed sequentially. When implementing the algorithm in an actual virtual circuit network, the computations will be performed on N separate computers and most of the computations will be done in parallel. Thus, the CPU times reported grossly overestimate the real time needed to obtain a near-optimal solution.

From an inspection of Table 1, it is clear that the distributed routing algorithm is efficient and very effective in finding near-optimal solutions. For every test problem (networks with up to 61 nodes), the distributed algorithm determines a solution that is within 1% of an optimal solution in minutes of CPU time on a SUN 3/50 workstation. Also our routing algorithm worked well for a heavily loaded network (test problem number 8). In the final solution, the link utilization factor of one of the links was 0.997. For this test problem, every solution generated by the algorithm after the fourth iteration was feasible. Furthermore, a near-optimal solution was obtained in less than 4 minutes of CPU time.

7 Summary and Conclusions

This papers focuses on the development of a distributed routing algorithm for virtual circuit networks. We first consider the formulation of this problem as the nonlinear combinatorial optimization problem by Gavish and Hantler [8]. We modified their formulation so that fewer status messages are required in a distributed protocol. As in [8], we applied Lagrangean relaxation to develop an algorithm. We found that the standard update rule used in [8] for the multipliers required the transmission of too many overhead messages. We discovered a new way of updating the multipliers that requires much fewer overhead messages. Moreover, the resulting distributed algorithm finds solutions that are closer to an optimal solution with the same amount of computation time. In section 5, we presented a distributed version of a centralized heuristic developed by Courtois and Semal [2].

We are continuing our work in two directions. First, we are improving the formulation of the problem so that the resulting distributed algorithm can be made more reliable and adaptive. Second, we are continuing our computational experiments to investigate how the algorithms adapts to changes in the network structure and to changes in the traffic requirements.

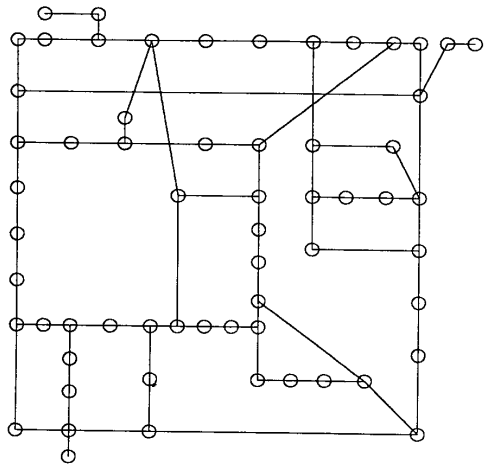

Fig. 1. 61-node 148-link ARPA net.

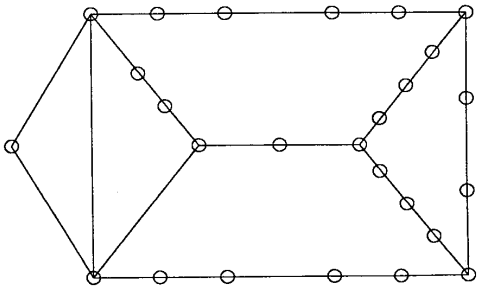

Fig. 3. 26-node 60-link OCT net.

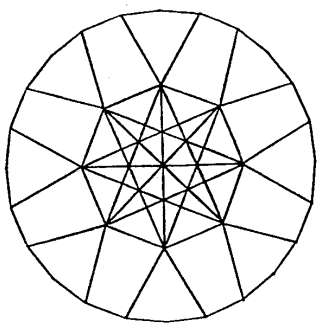

Fig. 2. 32-node 120 link RING net.

Case No.	Network ID	Routes /Pair	Total Routes Generated	Link Capacities	Lower Bounds (msec)	Upper Bounds (msec)	Percentage Difference (%)	CPU Time (sec)	No. of Iter.
1	ARPA	3	6178	500.0	17.4840	17.4851	0.006	244.8	200
2	"	3	6178	375.0	29.8636	29.8674	0.013	250.7	200
3	"	3	6178	300.0	55.4858	55.5511	0.118	243.8	200
4	"	3	6178	272.7	90.7174	90.8069	0.099	245.9	200
5	"	2	5290	500.0	17.5500	17.5514	0.008	198.1	200
6	"	2	5290	375.0	30.1053	30.1101	0.016	198.6	200
7	"	2	5290	300.0	57.2807	57.2998	0.033	198.5	200
8	"	2	5290	272.7	189.692	189.795	0.054	198.4	200
9	OCT	3	999	125.0	56.9099	56.9850	0.132	21.3	200
10	"	3	999	100.0	86.5846	86.6951	0.128	20.1	200
11	"	3	999	83.33	133.946	134.278	0.234	18.4	200
12	"	3	999	71.43	225.917	226.655	0.327	20.1	200
13	"	2	794	125.0	57.3806	57.3882	0.013	16.3	200
14	"	2	794	100.0	87.9979	88.0489	0.058	16.6	200
15	"	2	794	83.33	138.763	138.816	0.038	16.5	200
16	"	2	794	71.43	249.558	249.975	0.167	16.5	200
17	RING	3	2422	150.0	24.0110	24.0270	0.067	39.0	200
18	"	3	2422	100.0	40.4423	40.4931	0.126	38.8	200
19	"	3	2422	75.00	61.6932	61.8494	0.253	36.6	200
20	"	3	2422	60.00	90.6315	91.2678	0.480	37.6	200
21	"	2	1646	150.0	24.1919	24.2004	0.035	27.2	200
22	"	2	1646	100.0	41.0024	41.0335	0.076	27.6	200
23	"	2	1646	75.00	63.0824	63.1706	0.140	27.9	200
24	"	2	1646	60.00	93.8871	94.1109	0.238	27.7	200

Table 1: Summary of some computational results

References

[1] D.G. Cantor and M. Gerla. Optimal routing in a packet switched computer network. *IEEE Transactions on Computers*, C-23:1062–1069, 1974.

[2] P.J. Courtois and P. Semal. An algorithm for the optimization of nonbifurcated flows in computer communication networks. *Performance Evaluation*, 1:139–152, 1981.

[3] A. Danet, R. Despres, A.L. Rest, G. Pichon, and S. Ritzenthaler. The French public packet switching service: The transpac network. In *Proceeding Third International Computer Communication Conference*, pages 251–260, 1976.

[4] M.L. Fisher. The lagrangian relaxation method for solving integer programming problems. *Management Science*, 27(1):1–18, January 1981.

[5] L. Fratta, M. Gerla, and L. Kleinrock. The flow deviation method: An approach to store-and-forward communication network design. *Networks*, 3:97–133, 1973.

[6] R.G. Gallager. A minimum delay routing algorithm using distributed computation. *IEEE Transactions on Communications*, COM-25(1):73–85, January 1977.

[7] B. Gavish. Topological design of centralized computer networks: formulations and algorithms. *Networks*, 12:355–377, 1982.

[8] B. Gavish and S.L. Hantler. An algorithm for optimal route selection in SNA networks. *IEEE Transactions on Computers*, COM-31(10):1154–1160, October 1983.

[9] A.M. Geoffrion. Lagrangean relaxation and its uses in integer programming. *Math. Programming Study*, 2:82–114, 1974.

[10] M. Gerla. Routing and flow control ISDN's. In *Proc. 1986 ICCC*, pages 643–647, 1986.

[11] J.P. Gray and T.B. McNeill. SNA multiple-system networking. *IBM System Journal*, 18:263–297, 1979.

[12] GTE Telenet Communications Corp., Vienna, VA. *Functional Description of GTE Telenet Packet Switching Networks*, May 1982.

[13] M. Held and R.M. Karp. The traveling salesman problem and minimum spanning trees. *Operations Res.*, 18:1138–1162, 1970.

[14] M. Held, P. Wolfe, and H.D. Crowder. Validation of subgradient optimization. *Math. Programming*, 6:62–88, 1974.

[15] V.L. Hoberecht. SNA function management. *IEEE Transactions on Communications*, COM-28:594–603, 1980.

[16] L. Kleinrock. *Queueing Systems*, volume 1 and 2. New York: Wiley-Interscience, 1975 and 1976.

[17] D.G. Luenberger. *Linear and Nonlinear Programming.* Addison Wesley, 1984.

[18] J. McQuillan. *Adaptive Routing Algorithms for Distributed Computer Networks.* PhD thesis, Harvard University, May 1974.

[19] A. Mirzaian. Lagrangean relaxation for the star-star concentrator location problem: Approximation algorithm and bounds. *Networks*, 15:1–20, 1985.

[20] A. Rajaraman. Routing in TYMNET. In *Proceeding European Comput. Conf.*, 1978.

[21] D. Rotem, N. Santoro, and J.B. Sidney. Distributed sorting. *IEEE Transactions on Communications*, C-34(4):372–376, April 1985.

[22] H. Rudin. On routing and 'delta routing': A taxonomy and performance comparison of techniques for packet-switched networks. *IEEE Transactions on Communications*, COM-24(1):43–58, January 1976.

[23] M. Schwartz and T.E. Stern. Routing techniques used in computer communication networks. *IEEE Transactions on Communications*, COM-28:539–552, 1980.

[24] A. Segall. Optimal routing for virtual line-switched data networks. *IEEE Transactions on Communications*, COM-26, 1977.

[25] A. Segall. Advances in verifiable fail-safe routing procedures. *IEEE Transactions on Communications*, COM-29:491–497, 1981.

[26] A. Segall. Distributed network protocols. *IEEE Transactions on Information Theory*, IT-29(1):23–35, January 1983.

[27] L.R. Tymes. Routing and flow control in TYMNET. *IEEE Transactions on Communications*, COM-29:392–398, 1981.

[28] J.R. Yee. *Distributed Routing and Flow Control Algorithms for Communication Networks.* PhD thesis, Massachusetts Institute of Technology, December 1985.

[29] S. Zaks. Optimal distributed algorithms for sorting and ranking. *IEEE Transactions on Computers*, C-34(4):376–379, April 1985.