ROUTING WITH PACKET DUPLICATION AND ELIMINATION IN COMPUTER NETWORKS

Ariel Orda
Raphael Rom

Electrical Engineering Department
Technion - Israel Institute of Technology
Haifa, Israel

July 1985

Revised October 1987

ABSTRACT

Packet duplication is discussed as a means of increasing network reliability in an environment where packet loss exists. Several methods of routing the duplicates are presented, one of which--the *st*-numbering--is shown to have the combined advantage of using disjoint paths and more even utilization of network resources.

An additional mechanism, deliberate packet elimination, is introduced as a means of controlling congestion that may result, in part, from the duplication.

A comprehensive model is defined encompassing the process of packet duplication together with both forms of packet elimination. Within this model a cost function, based on average packet delay, is defined. A quasi-static distributed algorithm is developed that is optimal, deadlock-free and loop-free. Extension of the model to include packet retransmission is considered.

I. INTRODUCTION

In some computer networks or internetworks a phenomenon of packet loss takes place. This phenomenon may be due to a node or a link becoming inoperative, in which case *all* packets passing through it are lost. Another cause of packet loss may be statistical, such as shortage of resources. For example, gateways in an internetwork may destroy packets at will [1,2,3], in which case not all packets passing through that node are destroyed. This paper addresses the phenomenon of the second kind.

In such an environment we propose to improve the network's reliability by resorting to packet duplication, allowed both at the source node and at any node along the route. Of course, such duplication is not free: each duplicate consumes resources that must be conserved. The increased load may create congestion, a problem that can be alleviated by deliberate elimination of packets.

The main questions associated with the process are thus-when and where packets should be deliberately eliminated; when and where they should be duplicated; and how the duplicates should be routed.

One method of routing the duplicates consists of simply sending all duplicates along the ‘‘best" route. This is probably inefficient, as the circumstances that lead to packet loss are not likely to change quickly (e.g., buffer shortage typically lasts longer than a packet's lifetime). A better way would be to send the duplicates along separate routes. Since all members of a duplicate set traverse the network at almost the same time, it is advantageous to ensure to some extent that their routes do not cross, say for a certain number of hops. While this localizes the duplication process, the nodes are required to remember all messages that pass through them. A preferable way is to use completely disjoint routes. Here, messages have to be recorded only at the endpoints, which is done anyway for the purpose of windowing and sequencing. We shall return to disjoint-path routing in Section II.

In considering the question whether to duplicate a packet, we must take into account the trade-off between the additional load caused by the duplication (which may be the cause of increased packet loss) versus the chances that a nonduplicated packet may be lost. As this additional load should be accounted for, the duplication-routing algorithm presented here should be one of a class in which each node considers the effects of its decisions on the future performance of the entire network. Typical examples of such routing algorithms are given in the literature [4,5,6]. A similar trade-off between load and reliability exists when deliberate elimination is considered. To quantify these trade-offs, we consider the packet cost as a combination of the delay incurred by the copies of a packet that arrive at the destination and a penalty for each packet lost. Two approaches are presented: in one retransmission of lost packets is not considered; in the other it is.

Section II is devoted exclusively to the problem of routing through disjoint paths. A general approach for modeling the duplication-routing environment is discussed in Section III while the exact model and analysis are presented in Section IV. Section V presents a duplication-routing algorithm. Section VI discusses the case in which retransmissions take place.

II. ROUTING THROUGH DISJOINT PATHS

To achieve the purpose in question we need an algorithm which not only ensures that the duplicates of a given packet traverse different paths, but identifies the best paths for them (e.g., by attaching a certain cost, not necessarily fixed, to each link). We propose three different schemes. The first permits us to define a formal problem to which an optimal solution can be found; the solution, however, is rather complicated. The second scheme is easier to implement but may result in inefficient use of network resources. The third scheme is limited to two disjoint paths but is overcomes the deficiencies of the other two schemes.

A. Max Flow Min Cost Approach

The following is one way tpo define the problem formally. Given: (1) a directed graph $G(V,E,\Gamma)$ representing a network in which V is the set of nodes, E the set of links, and $\Gamma=\{\gamma_e \mid e\in E\}$ the set of link costs; (2) a source node $s\in V$ and a termination node $t\in V$ whose node connectivity is m. Find a set of k ($k\le m$) node-disjoint paths between s and t, such that the sum of the link costs of all paths will be minimal.

A quite simple solution to this problem is based on the path augmentation algorithm (PAA) for finding a minimum-cost maximum flow in a graph [7]. Our algorithm is composed of three parts: first, expand the given graph G into a new graph $\tilde{G}$ (since the PAA deals with link connectivity, while we are concerned with node connectivity), then execute the PAA on $\tilde{G}$, and finally map the paths found in $\tilde{G}$ into paths in G.

Part A--Graph expansion:

Define a new graph $\tilde{G}(\tilde{V},\tilde{E},\tilde{\Gamma})$ in the following way: with each node $v\in V$ associate a new node v', thus creating a set of nodes $V'=\{v' \mid v\in V\}$.* $\tilde{V}$ is now defined as

$$\tilde{V}=V\bigcup V'.$$

The set of links $\tilde{E}$ is constructed by connecting each node $v'\in V'$ to the node $v\in V$ with which it is associated, and each $w\in V$ to each $v'\in V'$ iff $(w,v)\in E$. The cost $\tilde{\gamma}_{\tilde{e}}$ of a link $\tilde{e}=(w,v')$ equals that of link (w,v), and the cost $\tilde{\gamma}_{\tilde{e}}$ of all links (v',v) is 0.

Obviously, an m node connectivity between any two nodes x and y in G implies an m link connectivity between x and y in $\tilde{G}$.

Part B--Modified Path-Augmentation Algorithm:

Assume that every link in $\tilde{G}$ has unit capacity. Apply the PAA of [7] to nodes x and y in $\tilde{G}$ for k iterations. Let $\tilde{\Phi}$ be the set of paths computed by the PAA.

---

*Existence of a one-to-one and onto function $f: V\to V'$ such that for every $v\in V$, $f(v)$ describes the node 'associated with' v is implicitly assumed. This assumption holds throughout the rest of this work.

Part C--Reverse Mapping:

Define a set of paths Φ between x and y in G by joining directly every $v, w \in V$ that is connected through a node $v' \in V'$ in $\tilde{\Phi}$.

Lemma: Φ is a set of k node-disjoint paths in G between x and y with minimal cost.

Proof: It was shown [7] that at its i-th iteration ($i \le m$), the path-augmentation algorithm finds a set of i link-disjoint paths between s and t at minimum cost. Because $k \le m$, $\tilde{\Phi}$ is a set of k such paths. Suppose now that the paths of Φ are not node-disjoint. This implies that there exist paths p_1 and p_2 in Φ and nodes w_1, w_2, v in V such that $(w_1, v) \in p_1, (w_2, v) \in p_2$; but this implies that there exist paths $\tilde{p}_1$ and $\tilde{p}_2$ in $\tilde{\Phi}$, for which: $(w_1, v), (v, v') \in \tilde{p}_1, (w_2, v), (v, v') \in \tilde{p}_2$. Thus $\tilde{\Phi}$ is not link-disjoint, which is a contradiction. □

This algorithm has a severe disadvantage: the routing is source-dependent--i.e., routing decisions to the same destination may vary depending on the packet source. Figure 1 demonstrates such a situation: node x_1 sends two duplicates destined for node y along the paths $x_1 \to l \to y$ and $x_1 \to k \to r \to y$; node x_2 does the same along the paths $x_2 \to k \to l \to y$ and $x_2 \to r \to y$; node k is thus seen to route packets destined for y in two different ways. This feature makes the routing algorithm excessively complex, in terms of both information and communications.

B. Multiple Tree Approach

To overcome the complexity of the solution presented in the previous subsection, we seek an algorithm that will find a set of node-disjoint paths from which source-independent routing can be derived. A tempting approach is to define for each destination node a set of directed spanning trees, all of which have the destination node as their root, thereby making all paths from any node to the root along the trees node-disjoint. Routing of duplicates along node-disjoint paths in such a network is obviously source-independent, as a node need only know along which tree (among those having the destination as their root) each duplicate should be routed.

Figure 1: Source Dependence of Routing Using PAA

Edmonds' theorem [8] defines the conditions for the existence of such trees. It states that

> If v is a vertex of a digraph $G(V,E)$ in which the link connectivity between any node $w \in V$ $(w \neq v)$ and v is not less than k, then there are k link-disjoint directed spanning trees of G rooted at v.

An algorithm for finding such trees is given in [9]. Note that the theorem deals with link connectivity, so we use the same expansion method as we did in the PAA. The complexity of the algorithm is $O(k^2nm^2)$ for building k trees in an n-node, m-link graph; as the algorithm is to be executed for each destination, we have an overall complexity $O(k^2n^2m^2)$ and for a typical communication network $O(k^2n^4)$, which is acceptable. The amount of memory required by each node to perform the routing (per destination) is $O(kn)$, which is also acceptable.

The tree approach gives rise to several problems, such as building the best trees in a reasonable number of steps. Moreover, it is deficient in its use of network resources since certain links may remain unused for routing duplicates to certain nodes.

C. *st*-Numbering Approach

If we restrict ourselves a maximum of two duplicates for each packet and to bidirectional links, a better solution to the node-disjoint-path problem can be found. This solution, based on the *st*-numbering algorithm (or *st* algorithm), is described in [10,11]. As it is used throughout the rest of this paper, we describe it here in more detail.*

Given two adjacent nodes s,t of a graph $G(V,E)$ for which each $v \in V$ is nonseparable from s (i.e., has node connectivity of at least two with respect to s), a one-to-one function $g: V \rightarrow \{1,2,..,|V|\}$ is called an *st* numbering if the following conditions are satisfied:

1. $g(s)=1$.
2. $g(t)=|V|$.
3. For every $v \in V - \{s,t\}$ there are adjacent nodes u and w such that $g(u)<g(v)<g(w)$.

The *st*-numbering offers an excellent solution for routing duplicates along node-disjoint paths. Suppose *st*-numbers are given to all nodes in the network with respect to some destination node s. To handle duplicates destined for s, one duplicate is routed through nodes with decreasing *st*-numbers and the other through nodes with increasing *st*-numbers (using the convention that node s has a higher *st*-number relative to those of its neighbors that do not have any higher numbered neighbor). Such routing is clearly disjoint-path and source independent. Moreover, the possibility of some links remaining unused (as in the tree approach) is obviated.

The *st*-numbering algorithm referred to above is also applicable for our more general case, in which we demand nonseparability only with respect to s.

We now proceed to describe the exact way in which the *st*-numbering is used for duplicate routing in a network $G(V,E)$.

---

*Use of the *st* numbering scheme in routing is discussed in [12], in the context of routing through link-disjoint paths in trees.

We concentrate on a certain destination $j \in V$. As there may be nodes in V that are separable from j, we identify first the subnetwork $G_j(V_j, E_j)$ of all nodes that are nonseparable from j (including j), and the links connecting these nodes (i.e., $E_j = \{(v,w) \mid v,w \in V_j, (v,w) \in E\}$). The st-numbering algorithm is applied to G_j (having j play the role of the s node, and any of its neighbors the role of the t node), and for each $i \in V_j$, $i \neq j$, two nonempty sets $H_i(j)$ and $L_i(j)$ are defined as follows:

$$H_i'(j) \stackrel{\Delta}{=} \left\{ k \;\middle|\; (i,k) \in E_j, st_j(k) > st_j(i) \right\},$$

$$H_i(j) \stackrel{\Delta}{=} \begin{cases} H_i'(j) & H_i'(j) \neq \varnothing \\ \{j\} & H_i'(j) = \varnothing \end{cases}, \quad L_i(j) \stackrel{\Delta}{=} \left\{ k \;\middle|\; (i,k) \in E_j, k \notin H_i(j) \right\}$$

where $st_j(l)$ stands for the st number given to l having j as the s node. Thus $H_i(j)$ groups neighbors of node i with number higher than that of i with respect to j. $L_i(j)$ similarly groups neighbors with lower numbers.

When a packet is duplicated, each duplicate is marked H or L. Node i $(i \neq j)$ routes H duplicates only to nodes in $H_i(j)$ and L ones only to nodes in $L_i(j)$.

For a node $i \in V$, $i \notin V_j$, we set $H_i(j) = L_i(j) \equiv \varnothing$; such a node cannot duplicate packets destined for j. However, packets passing through i en route for j might be duplicated later on their way; this happens if a path from i to j passes through node $k \in V_j$ and the packet is duplicated at k. Figure 2 depicts such a situation.

Figure 2: Duplication of Packets Originated Outside V_j

Finally, a word about optimization. Generally, there might exist several *st* numberings for a given network, each resulting in a different network performance. In these circumstances, choosing a numbering and simultaneously securing optimal performance is a complicated task. Our approach consists in arbitrary choices of a numbering and optimization of the performance with respect to it. (A heuristic approach to selecting a "good" *st* numbering is presented in [13]).

III. THE APPROACH

We present a model that describes the effects both of packet elimination and packet duplication. Within this model, we define a cost function, minimization of which reflects our intent to minimize the average delay while simultaneously increasing the network's reliability. In this section we present our approach qualitatively. A quantitative insight is given in Section IV.

Our approach follows closely Gallager's model and algorithm presented in [5] (to which we refer as the "basic" algorithm) which is a distributed, quasi-static algorithm for minimum average delay routing. Essentially, every node maintains--for each destination--a set of routing variables, specifying the corresponding fraction of the node's outflow routed towards each of the node's neighbors. In each iteration of the algorithm each node adjusts its routing variables so that part of its out-flow is diverted from "expensive" links (i.e., with high delay), to "cheaper" ones. It is shown that under the prescribed conditions this leads to minimization of the cost function.

We extend the basic model in several directions so as to include all processes (duplication, elimination, and routing of originals and duplicates). An important part of the extension is mapping of these processes into flows and link costs, which is described subsequently.

Routing

We assume that an *st* numbering procedure was previously applied to every node in our network with respect to every destination, and that every node i recognizes its sets $H_i(j)$ and $L_i(j)$ with respect to every destination node j. Every packet carries a label: a packet that has not been duplicated is labeled and referred to as a normal (or N) packet, and the two duplicates generated from it are referred to as an H-packet and an L-packet. Consequently, three types of flow are observed, one for each of the packet types (N, H, L).

Routing is performed in a way similar to that of the basic algorithm. Each node maintains three sets of routing variables, one for each type of flow. The routing variables at node i are computed so that H-packets destined for node j are forwarded only to neighbors belonging to $H_i(j)$, and L-packets, similarly, to neighbors belonging to $L_i(j)$.

Elimination

Packets may be eliminated in two kinds of circumstances: (1) randomly (or incidently), due to situations only partially under the node's control, or (2) whenever the node deems it useful. We thus distinguish between *randomly eliminated packets* (*re*-packets), and *deliberately eliminated packets* (*de*-packets). Apart from the above distinction we are also interested in distinguishing between eliminated packets having a duplicate which arrives at the destination and packets that do not have one. The latter are termed *costly* packets because of the cost involved (we pay a penalty for them), while the former are termed *free*.

We thus have a total of four types of eliminated packets: randomly eliminated costly and free, and deliberately eliminated costly and free (respectively rc, rf, dc, and df packets).

It should be noted that deliberate elimination is in fact necessitated by the presence of random elimination in order to avoid undesired phenomena as exemplified in Figure 3. In this figure node s sends packets to node t via nodes i and k. Let w_1,w_2,w_3 be the costs per packet for traversing the links and w_4 the penalty for eliminating a packet at node i. We assume that no deliberate elimination is allowed and that $w_2 \gg w_1,w_3,w_4$. Under these circumstances, node's k best policy would be to route packets back toward node i in the "hope" that they will be eliminated. This is clearly an unacceptable policy; it would be preferable to allow node i to eliminate packets deliberately (note the routing loops formed!).

In a manner similar to that of [14] we model packet elimination by ‘‘routing" eliminated packets toward their destinations through fictitious links added for that purpose. This enables us to consider elimination as a regular case of routing (see Section IV.A).

Duplication

To compensate for packet elimination, nodes are allowed to duplicate packets. Only a single duplication is allowed along a packet's path. The duplication process is modeled by defining for each node i -- per destination node j --a duplication variable $d_i(j)$ which specifies the fraction of the N-packets destined for j that is duplicated in node i. As part of the algorithm, each node will determine a proper value of $d_i(j)$. Clearly, no duplication can take place at a node unless both $H_i(j)$ and $L_i(j)$ are nonempty; the algorithm will ensure that $d_i(j) \equiv 0$ empty $H_i(j)$ or $L_i(j)$.

Cost function

Our cost function is composed of the total average delay incurred by packets arriving at the destination and by the penalties paid for lost packets. When a single duplicate arrives at the destination, the cost is its delay; when two duplicates arrive, the cost is some combination of their delays. When all duplicates are eliminated, the cost is the penalty paid for that lost packet.

Figure 3: The Need for Deliberate Elimination

Thus, to every link, real and fictitious, a weight is attached signifying the cost of flow through it. To calculate the cost of eliminated packets, a mapping is needed between the packet type (N,H,L) and its cost. Eliminated N-packets are clearly all costly; eliminated H- and L-packets are harder to deal with, as determination of their cost requires knowledge of the fate of the sibling. This difficulty is overcome by defining a quantity $\beta(j)$, related to the elimination probability of any duplicate on its way from any source to destination j. Now, each node regards a portion $\beta(j)$ of its eliminated H- and L- packets destined toward j as costly, and the rest as free.

Algorithm

Following the basic algorithm (and also [6]) we find necessary and sufficient conditions for a set of routing and duplication variables to minimize the cost function. The algorithm is then derived directly from the conditions for optimality.

The algorithm consists of a protocol to calculate the marginal costs and exchange messages among neighbors, and a duplication routing (DR) algorithm to update the routing and duplication variables at each node. In each iteration, and for each type of flow, the DR algorithm reduces the traffic on expensive links and increases it on the cheap ones. Furthermore, it levels the amount of duplication and of deliberate elimination according to the marginal gain in the cost function involved in each of these processes.

Note that extensions of the basic algorithm effected with a view to increasing its speed of convergence (such as that in [15]), or for other purposes (as in [16]) are applicable to our model. For the sake of clarity, however, our presentation is confined to the basic algorithm.

IV. THE MODEL

A. General

Let our network be of fixed topology and composed of the set of nodes V_0 and the set of links E_0. We assume that each link $(i,k) \in E_0$ has finite capacity $C_{ik} > 0$ and consider full duplex links, i.e., $(i,k) \in E_0$ implies $(k,i) \in E_0$. We denote by $r_i(j)$ the expected traffic (in bits per second) entering the network at node i and destined for node j, and assume that this input traffic forms an ergodic process. The random elimination process is modeled by $e_i(j)$, the probability of a packet on its way to node j being (randomly) eliminated while in transit through node i.

To facilitate the analysis we construct an augmented network in which all flows (of both real and eliminated packets) are conserved. We model packet elimination by adding fictitious links carrying the flow of eliminated packets directly to their destinations--each type of eliminated packet through a different fictitious link. However, although we originally defined four types of eliminated packets, only three need be considered since (as will be shown subsequently see Section V.C) there is no need for deliberate elimination of duplicated packets (df flow is nonexistent).

To avoid having more than one link between two nodes, fictitious nodes are also added. Thus, an eliminated packet is routed from the node in which it is eliminated through a fictitious link toward a fictitious node associated with the intended destination of the packet, and then toward the destination itself. Figure 4 shows the fictitious nodes and links associated with some node j, as well as the fictitious links connecting some node i to them. The total sets of nodes and links in the augmented network are denoted by V and E respectively.

Figure 4: Fictitious Nodes and Links

To complete the description, we must specify what packets are routed along which links and what the associated costs are.

B. Routing and Duplication

From the node's activity standpoint, we distinguish between packets over which the node has control (i.e., noneliminated and deliberately eliminated) and the rest (randomly eliminated). Each of these will be treated separately. Noncontrollable packets are obviously routed directly toward the destination over the appropriate fictitious links. Controllable packets are routed according to a routing function that diverts a portion of the traffic to each eligible neighbor. Formally, let Z_i be the set of node i's real neighbors (from V_0) plus the fictitious nodes toward which deliberately eliminated packets are routed (j_{de} in Figure 4); node i routes a fraction $\phi_{Nik}(j)$ of the outflow of N-packets at node i and destined for node j over link (i,k) where $k \in Z_i$. The amount of deliberately eliminated packets is therefore controlled by $\phi_{Nij_{de}}(j)$. A similar approach is adopted for H- and L-packets, except (as noted before) that these packets will not be deliberately eliminated and the eligible neighbors are those belonging to $H_i(j)$ and $L_i(j)$ respectively.

To sum up, for each node i, we have three sets of numbers $\{\phi_{Nik}(j)\}$, $\{\phi_{Hik}(j)\}$, and $\{\phi_{Lik}(j)\}$ controlling the routing and the deliberate eliminations, and one set $d_i(j)$ controlling the duplication process.

Figure 5 describes in detail the packet flow through node i. Note the definition of $\tilde{t}_{Ai}(j)$, $\tilde{t}_{Ai}(j)$, and $t_{Ai}(j)$ (where A is one of N, H, L) which represent the flow entering the node, the flow after random elimination, and the flow after duplication, respectively. These flows obey the following relations:

Figure 5: Packet Flow Through Node i

$$
\begin{aligned}
\tilde{t}_{Ni}(j) &= [1-e_i(j)]\sum_l [1-d_l(j)]\tilde{t}_{Nl}(j)\phi_{Nli}(j) + r_i(j) \\
t_{Ni}(j) &= [1-e_i(j)][1-d_i(j)]\sum_l t_{Nl}(j)\phi_{Nli}(j) + [1-d_i(j)]r_i(j) \\
t_{Hi}(j) &= [1-e_i(j)]\sum_l t_{Hl}(j)\phi_{Hli}(j) + d_i(j)\tilde{t}_{Ni}(j) \\
t_{Li}(j) &= [1-e_i(j)]\sum_l t_{Ll}(j)\phi_{Lli}(j) + d_i(j)\tilde{t}_{Ni}(j)
\end{aligned}
\qquad (1)
$$

Equations (1) uniquely define the flows as a function of the duplication and routing variables[5,13].

C. Cost Function

Denote by $f_{Nik}, f_{Hik}, f_{Lik}$ the flow of N-, H-, and L-packets respectively in link (i,k), and use the shorthand $\underline{f_{ik}} \stackrel{\Delta}{=} (f_{Nik}, f_{Hik}, f_{Lik})$. In addition we define $f_{ik} \stackrel{\Delta}{=} f_{Nik} + f_{Hik} + f_{Lik}$. Let $T_{ik}(f_{ik})$ be the delay (per bit) function of a link $(i,k) \in E_0$. We assume that the function $f_{ik} \cdot T_{ik}(f_{ik})$ is nonnegative, continuous, convex, and nondecreasing with respect to f_{ik}, with continuous first and second derivatives, and that T_{ik} becomes unbounded as the flow reaches capacity [6]. We also denote by β the probability of packet elimination in the network irrespective of source and destination.

With these definitions, the cost function of an actual link $(i,k) \in E_0$ is

$$
D_{ik}(\underline{f_{ik}}) \stackrel{\Delta}{=} [f_{Nik} + \frac{(1+\beta)}{2}(f_{Hik} + f_{Lik})]T_{ik}(f_{ik}) \ .
$$

Here the cost for N-packets is the full delay. For H- and L-packets, the full price is paid when only one duplicate arrives (probability β), and half the price for each when both duplicates arrive (probability $1-\beta$).

The cost of deliberate elimination is calculated by assigning the cost function $D_{ij_{de}}(f_{ij_{de}}) \stackrel{\Delta}{=} f_{ij_{de}} D_j$ to the fictitious links (i, j_{de}) carrying these packets, where D_j does not depend on the flow. Similarly, for randomly eliminated flow $D_{ij_{rc}}(f_{ij_{rc}}) = f_{ij_{rc}} D_j$. The cost function of the rest of the fictitious links is zero since all the traffic they carry is either not costly or the cost has already been accounted for.

Putting all this together, the network cost function, D_T, is defined as

$$
D_T(\boldsymbol{f}) = \sum_{(i,k) \in E} D_{ik}(\underline{f_{ik}}).
$$

It is readily verified that $D_{ik}(\underline{f_{ik}})$ is guaranteed to be convex if and only if $\beta=1$. This is a consequence of assigning a different cost to each kind of flow, although all flows equally affect the link delay. We shall therefore restrict ourselves to the case of $\beta=1$. This restriction results in increasing the relative cost of sending duplicates as compared with that of sending an unduplicated packet and that of deliberately eliminating it--effects that can be canceled out by choosing higher values for D_j.

D. Optimality Conditions

Our aim is to find necessary and sufficient conditions for a set of flows to minimize D_T. Having constructed a Lagrangian based on the constraints on network variables and taken derivatives (steps similar to those taken in [6]), we are led to the following four conditions:

$$\forall i,j\in V_0\ i\neq j,\forall k\in Z_i\quad D_{ik}^{N'}+M_{Nk}(j)\begin{cases}=\lambda_{Ni}(j) & f_{Nik}(j)>0\\ \geq\lambda_{Ni}(j) & f_{Nik}(j)=0\end{cases} \tag{2}$$

$$\forall i,j\in V_0\ i\neq j,\forall k\in H_i(j)\quad D_{ik}^{H'}+M_{Hk}(j)\begin{cases}=\lambda_{Hi}(j) & f_{Hik}(j)>0\\ \geq\lambda_{Hi}(j) & f_{Hik}(j)=0\end{cases} \tag{3}$$

$$\forall i,j\in V_0\ i\neq j,\forall k\in L_i(j)\quad D_{ik}^{L'}+M_{Lk}(j)\begin{cases}=\lambda_{Li}(j) & f_{Lik}(j)>0\\ \geq\lambda_{Li}(j) & f_{Lik}(j)=0\end{cases} \tag{4}$$

$$\lambda_{Ni}(j)\begin{cases}=\lambda_{Hi}(j)+\lambda_{Li}(j) & 0<d_i(j)<1\\ \leq\lambda_{Hi}(j)+\lambda_{Li}(j) & 0=d_i(j)\\ \geq\lambda_{Hi}(j)+\lambda_{Li}(j) & d_i(j)=1\end{cases} \tag{5}$$

where $D_{ik}^{N}\overset{\Delta}{=}\dfrac{\partial D_{ik}(\underline{f}_{ik})}{\partial f_{Nik}}$, and similarly for $D_{ik}^{H'}$ and $D_{ik}^{L'}$, and the λs are the Lagrange multipliers, and the M values are defined by

$$\begin{aligned}
&M_{Nk}(j)\overset{\Delta}{=}[1-e_k(j)][1-d_k(j)]\lambda_{Nk}(j)+[1-e_k(j)]d_k(j)[\lambda_{Hk}(j)+\lambda_{Lk}(j)]+e_k(j)D_j\\
&M_{Hk}(j)\overset{\Delta}{=}[1-e_k(j)]\lambda_{Hk}(j)+e_k(j)\beta(j)D_j\\
&M_{Lk}(j)\overset{\Delta}{=}[1-e_k(j)]\lambda_{Lk}(j)+e_k(j)\beta(j)D_j\\
&M_{Nj}(j)=M_{Hj}(j)=M_{Lj}(j)\overset{\Delta}{=}0\ .
\end{aligned} \tag{6}$$

From the above derivation it is clear that expressions (2)-(5) are necessary conditions for minimizing D_T over the set of feasible solutions. For β=1, they are also sufficient [13].

The conditions for optimality have an intuitive significance: The Lagrange multipliers $\lambda_{Ai}(j)$ are the cost of sending an infinitely small incremental flow of type A from node i to destination j, where A is one of N,H,L. The value $M_{Ak}(j)$ is, then, the average cost of sending such a flow via a neighbor k (apart from the incremental cost of transmission to k itself, which is $D_{ik}^{A'}$), bearing in mind the possibilities for duplication and elimination at k. Thus, the conditions simply state that optimality requires that the incremental cost be equal for all neighbors k to which i sends flow of type A, and that i levels the degree of duplication so that the incremental cost of sending the duplicates equals that of sending a single copy.

V. THE ALGORITHM

The algorithm seeks to satisfy the optimality conditions by modifying the flows (through changes in the variables ϕ and d) so that the λs are equalized. This is done by reducing those routing variables $\phi_{Aik}(j)$ for which $D_{ik}^{A'}(\underline{f_{ik}})+M_{Ak}(j)$ is large, increasing those for which it is small, and, in addition, reducing $d_i(j)$ if $\lambda_{Ni}(j)$ is smaller than $\lambda_{Hi}(j)+\lambda_{Li}(j)$ and increasing it if $\lambda_{Ni}(j)$ is larger. As mentioned previously, the algorithm consists of a protocol to calculate the Lagrange multipliers and a duplication routing (DR) algorithm to update the variables in each node.

A. The Protocol to Calculate λ

To calculate the λ each node i estimates, as a time average, the $D_{ik}^{N'}$, $D_{ik}^{L'}$, $D_{ik}^{H'}$ for each outgoing link [13]. Calculation of the quantities $\lambda_{Hi}(j)$ and $\lambda_{Li}(j)$ proceeds as in [5]: node i waits until it has received $M_{Hk}(j)$ or $M_{Lk}(j)$ from all its neighbors k that are H- or L-downstream from it respectively, and then computes $\lambda_{Hi}(j)$ and $\lambda_{Li}(j)$ by the formula

$$\lambda_{Hi}(j)=\sum_{k\in H_i(j)}[D_{ik}^{H'}+M_{Hk}(j)]\phi_{Hik}(j) \qquad \lambda_{Li}(j)=\sum_{k\in L_i(j)}[D_{ik}^{L'}+M_{Lk}(j)]\phi_{Lik}(j). \tag{7}$$

Having calculated $\lambda_{Hi}(j)$, node i calculates $M_{Hi}(j)$ according to (6) and transmits it to all nodes $k\in L_i(j)$. Similarly for $M_{Li}(j)$. Note that we should not be concerned with routing loops because of the constraints imposed by the *st*-numbering scheme.

The calculation of $\lambda_{Ni}(j)$ is somewhat different since the N-stream is affected by the H- and L-streams. Typically, node i waits until it has computed $\lambda_{Ni}(j)$, $\lambda_{Li}(j)$, and $\lambda_{Hi}(j)$ from which it computes $M_{Ni}(j)$ according to (6), and transmits it to all its neighbors (unless $d_i(j)=1$, in which case $M_{Ni}(j)$ can be calculated immediately). $\lambda_{Ni}(j)$ is calculated after receiving $M_{Nk}(j)$ from all N-downstream neighbors by

$$\lambda_{Ni}(j)=\sum_{k\in Z_i}[D_{ik}^{N'}+M_{Nk}(j)]\phi_{Nik}(j). \tag{8}$$

The protocol for the N-variables is problematic because it has deadlocks unless the routing is loop-free. It is known that the basic protocol deadlocks only if there is a circuit of nodes and a destination j such that for each subsequent pair in the circuit (i,k), $\phi_{ik}(j)>0$. In our case, however, this is too strict a requirement since if one of the nodes in such a circuit duplicates all its N-packets ($d_i(j)=1$), routing loops are not formed. Freedom from loops in our case is secured by absence of a circuit of nodes for which $[1-d_i(j)]\phi_{Nik}(j)>0$.

B. DR Algorithm

In order to ensure loop-free routing given disjoint duplicate paths, we define--in the spirit of [5]--for each node and each type of flow (N, H, L) a set of blocked neighbors ($B_{Ni}(j)$, $B_{Hi}(j)$, $B_{Li}(j)$ respectively). For these nodes a flow may not be initiated if it was zero before. Note that in our case a zero flow may

result from either $\phi=0$ or, for N-flow, if $d=1$.

On each iteration the *DR* algorithm maps the current duplication parameter set d into a new set $\hat{d}$, and the routing parameter set ϕ into a new set $\hat{\phi}$. Given our definition of the sets of neighbors and of blocked neighbors, $\hat{\phi}$ can be derived from ϕ in a manner similar to [5]. The duplication variable is similarly derived as follows:

If $d_i(j)=1$ and there exists a $k \in B_{Ni}(j)$ such that $\hat{\phi}_{Nik}(j)\neq 0$, then

$\hat{d}_i(j)=1$

else

a. $\Delta_{\lambda_i}(j)=\lambda_{Ni}(j)-\lambda_{Hi}(j)-\lambda_{Li}(j)$;

b. $$\Delta_{d_i}(j)=\begin{cases}\max\{-d_i(j), \min[1-d_i(j), \dfrac{\Delta_{\lambda_i}(j)\eta}{\tilde{t}_{Ni}(j)}]\} & \tilde{t}_{Ni}(j)\neq 0 \\ -d_i(j) & \tilde{t}_{Ni}(j)=0 \text{ and } \Delta_{\lambda_i}(j)<0 \\ 1-d_i(j) & \tilde{t}_{Ni}(j)=0 \text{ and } \Delta_{\lambda_i}(j)>0 \\ 0 & \tilde{t}_{Ni}(j)=0 \text{ and } \Delta_{\lambda_i}(j)=0 \end{cases}$$

c. $\hat{d}_i(j)=d_i(j)+\Delta_{d_i}(j)$

In the above η is an arbitrary scale parameter that controls the amount of correction to the routing and duplication variables in every iteration.

This algorithm proves to converge to optimum. This convergence does not result directly from that of the basic algorithm, since our case is more general for the following reasons: (1) there are several types of flow; (2) the direct origin of the H- and L- flows is the duplicated N-flow, and thus may be non-stationary; (3) the total amount of flow in the network is not stationary. However, it does seem natural for this algorithm to be optimal given the optimality of the basic algorithm. If we consider the duplication box and each of the elimination boxes in Figure 4 as distinct nodes, we have an almost normal network (except that the duplication box routes each packet toward two different nodes). Although this nonstationarity calls for a rigorous treatment to prove the optimality claim [13], this nonstationarity is not crucial since the flow may vary within a bounded range.

As is expected from [5] and shown in [13], the protocol thus defined is free of deadlocks, the resulting routing is loop-free on every iteration, and the duplication and routing variables converge to values that minimize D_T.

C. DR Algorithm and Flow Control

From the above model, it is readily deduced that, at optimality, deliberate elimination takes place only at the source node--if at all. We note that the cost of deliberate elimination does not depend on the identity of the eliminating node, and recall that D_T increases whenever a packet traverses an actual link.

Therefore, D_T is minimized only if all deliberate eliminations are performed at the source. This also explains why there is no need for deliberate elimination of H- and L- packets, since duplicating a packet at the source and then having the source deliberately eliminate one or both duplicates is meaningless. (Note also that deliberate elimination can take place only at nodes without N-upstream neighbors as this process does not distinguish a node's own packets from those coming from an upstream neighbor.)

In fact, source nodes use deliberate elimination as a flow control mechanism, reducing the input flow in order to reduce the load (and thus the delay) in the network. In [14] a similar approach was used for developing a flow control mechanism, which was braided with a routing algorithm. There, discarded input packets are routed fictitiously toward the destination via a fictitious link whose cost function is a penalty function for discarded packets. However, in [14] this penalty function is not linear in the flow, as it is in our case, but increases asymptotically to infinity as the amount of discarded packets reaches the total input demand of the node. In our case, such an approach would not be appropriate, since packets can be discarded at any node, and thus the penalty cannot be related to some value of the source node. Moreover, it is not obvious that the penalty paid for any discarded packet should depend on the total amount of discarded, and it can be argued that in certain cases it is more reasonable to pay a constant penalty for each lost packet.

VI. A MODEL WITH RETRANSMISSIONS

We consider now the case in which a packet is transmitted repeatedly by the source until it arrives at the destination.

A model that best describes this situation is one in which a fictitious link is added between any node i and every source node s. All packets completely eliminated at i (i.e., all costly lost packets) are routed toward s via this fictitious link, as complete loss of a packet implies its retransmission by the source. (As before, free lost packets are routed toward the destination via fictitious links.) However, a major problem arises when this model is used for a distributed implementation, as the routing decisions are now source-dependent (of complexity $O(n^2)$), thus requiring a large volume of memory at each node and entailing a high communication cost in the links. Another problem with this model arises when an algorithm of the same class as the one described in Section V is considered; routing loops cannot then be avoided.

We thus propose an alternative model in which we add a fictitious link between any node and itself (to avoid the inconvenience of creating a self-loop one can regard each node as composed of two internal nodes). All costly packets eliminated at i are routed again toward i via this fictitious link; they then continue to their destination as normal packets. As before, free lost packets are sent directly to the destination via a fictitious link. Observe that in this model we assume that a retransmitted packet will arrive again at the node where it was lost. The cost function of the fictitious link between the eliminator node i and itself is related to the retransmission price from the source node to i. Obviously, this model is less accurate then the one elaborated on previously, but still retains the main properties of the retransmission scheme; for example, deliberate elimination is ruled out (as it is always more expensive to lose a packet than to go on transmitting it), a reasonable feature where retransmissions are involved.

This model can be analyzed in a very similar way to the one presented in Section III; an algorithm similar to the *DR* algorithm can be derived for it, and all properties proved for the *DR* algorithm can be proved to hold for this algorithm as well.

REFERENCES

1. D.R. Boggs, J.F. Shoch, E.A. Taft, and R.M. Metcalfe, ‘‘PUP: An Internetwork Architecture,’’ *IEEE Trans. on Communications* **COM-28**(4) pp. 612-623 (April 1980).
2. J.B. Postel, ‘‘Internetwork Protocol Approaches,’’ *IEEE Trans. on Communications* **COM-28**(4) pp. 604-611 (April 1980).
3. J.F. Shoch and L. Stewart, ‘‘Interconnecting Local Networks via the Packet Radio Network,’’ pp. 153-158 in *Proc. of the 6th Data Communications Symposium*, Pacific Grove, California (November 1979).
4. C.E. Agnew, ‘‘On Quadratic Adaptive Routing Algorithms,’’ *Communications of the ACM* **19**(1) pp. 18-22 (1976).
5. R.G. Gallager, ‘‘A Minimum Delay Routing Algorithm Using Distributed Computation,’’ *IEEE Trans. on Communications* **COM-25**(1) pp. 73-85 (January 1977).
6. A. Segall, ‘‘Optimal Distributed Routing for Virtual Line-Switched Data Networks,’’ *IEEE Trans. on Communications* **COM-27**(1) pp. 201-209 (January 1979).
7. R.G. Busacker and T.L. Saaty, *Finite Graphs and Networks: An Introduction with Applications*, McGraw Hill, New York (1965).
8. J. Edmonds, ‘‘Edge Disjoint Branchings,’’ pp. 91-96 in *Courant Institute Scientific Symposium 9*, ed. R. Rustin, Algorithmics Press, (1973).
9. S. Even, *Graph Algorithms*, Computer Science Press, New York (1979).
10. A. Lempel, S. Even, and I. Cederbaum, ‘‘An Algorithm for Planarity Testing of Graphs,’’ pp. 215-232 in *Proc. of the International Symposium Theory of Graphs*, ed. P. Rosentiehl, New York (1967).
11. S. Even and R.E. Tarjan, ‘‘Computing an st-numbering,’’ *Theory of Computer Science* **2** pp. 339-344 (1976).
12. A. Itai and M. Rodeh, ‘‘The Multi-Tree Approach to Reliability in Distributed Networks,’’ pp. 137-147 in *Proc. 25th Symposium on Foundations of Computer Science*, Singer Islands, Florida (October 1984).
13. A. Orda and R. Rom, ‘‘Optimal Routing with Packet Duplication in Computer Networks,’’ EE Publication 521, Faculty of Electrical Engineering, Technion, Haifa, Israel (July 1985). (English version).
14. R.G. Gallager and S.J. Golestaany, ‘‘Flow Control and Routing Algorithms for Data Networks,’’ pp. 779-784 in *Proc. 5th International Conference on Computer Communications (ICCC-80)*, Atlanta, Georgia (October 1980).
15. D.P. Bertsekas, E.M. Gafni, and R.G. Gallager, ‘‘Second Derivative Algorithms for Minimum Delay Distributed Routing in Networks,’’ *IEEE Trans. on Communications* **COM-32**(8) pp. 911-919 (August 1984).
16. A. Ephremides, ‘‘Extension of an Adaptive Distributed Routing Algorithm to Mixed Media Networks,’’ *IEEE Trans. on Communications* **COM-26**(8) pp. 1262-1266 (August 1978).