OPTIMAL PACKET FRAGMENTATION IN COMPUTER NETWORKS

Ariel Orda
Raphael Rom

Department of Electrical Engineering
Technion - Israel Institute of Technology
Haifa, Israel 32000

March 1994

ABSTRACT

The packet fragmentation problem in computer networks is that of breaking a packet into smaller pieces (fragments) due to packet size limitations along the packet's route. This is a typical internetworking problem. We show that the commonly used simplistic approach whereby the routing and fragmentation functions operate completely independently is far from being efficient and has adverse effects on network performance. This paper deals with the combined fragmentation-and-routing problem.

We discuss several possible fragmentation machines and indicate their equivalence. This enables the formulation of a comprehensive yet tractable flows model, whose performance measure is total network delay. An analysis of this model leads to necessary and sufficient optimality conditions for the fragmentation-and-routing problem.

The optimality conditions serve as a baseline for devising several optimal algorithms, both centralized and distributed. To deal with minimum first derivative length algorithms, we generalize the concept of minimum first derivative paths in order to accommodate them into our environment.

Several special cases of practical interest are discussed. We show how the problem size and the running time of the algorithms are considerably shortened in networks in which packet sizes come in a limited number of sizes. We also show how our approach can accommodate performance measures other than total delay. The case of networks with virtual circuits is also discussed.

Keywords: Fragmentation, Internetworks, Networks, Packet-size, Routing.

I. INTRODUCTION

The packet fragmentation problem in computer networks is that of breaking a packet into smaller pieces (fragments) due to packet size limitations along the packet's route. The problem arose when network interconnection was first considered[1]. It was then realized that packets traversing an internetwork may have to undergo fragmentation if the packet size and the allowable packet sizes along the internetwork route do not match. With the recent advances of network interconnection [2], the problem becomes ever more prominent. Packet fragmentation is not only a problem of internetworking. It is possible that links in a single network impose packet size constraints, for example to facilitate error correction over noisy links.

Packet fragmentation adversely effects network operation in various ways such as increasing the load on the network and complicating packet handling both at the endpoints and along the way. Therefore, wherever possible, packet fragmentation should be avoided at the design level. Nonetheless, in existing networks it is a necessary evil, and thus its efficient control must be addressed. We shall show that current simplistic approaches for handling fragmentation are far from being efficient. The purpose of this paper is to formulate schemes for the efficient control of networks in which packet fragmentation is necessary.

A major fragmentation-related function is that of reassembly, i.e., where are fragments put together in a manner acceptable to the receiving host. Two possibilities exist: along the way or at the destination. En-route reassembly poses too many constraints such as having to buffer messages in intermediate nodes, limiting the selection of routes, and causing extra delays. Thus, in most existing networks that perform fragmentation, reassembly is done at the destination.

The fragmentation process deals with the questions of where to perform the fragmentation, to what fragment-sizes and how to route the fragments to their destination. These questions, which are non-trivial, are the subject of this paper.

The approach applied in existing networks is a simplistic one, namely a routing function that operates independently of fragmentation requirements and a fragmentation function that operates according to a greedy rule. This approach is far from being efficient. Consider for example the simple network depicted in Figure 1 in which packets of size 12 arrive to a five-node sequence; the maximum packet size of each link is shown in the figure. The greedy approach would fragment a 12-unit packet into two fragments of size 5 and a single fragment of size 2. These fragments will undergo further fragmentation in the next hop resulting in two fragments of size 4, one fragment of size 2 and two fragments of size 1 and so on. Altogether this approach will result in 24 fragment-hops to get a single packet along the path. This is almost the worst approach. A much better approach would be to fragment the packet into 3 fragments of size 4 which will then be refragmented into 6 fragments of size 2 resulting in a total of 18 fragment-hops.

This simple example is quite instructive. It shows that the greedy approach is far from optimal and that the location where fragmentation is performed has a vast impact on performance. Moreover, it easily leads to the realization that routing and fragmentation are tightly coupled. This observation is true for both circuit and datagram networks: In circuit based networks the selection of a route must be constrained by the needs of fragmentation, and in datagram networks (where bifurcated routing may be used) this relation is even more involved since separate (and possibly different) routing decisions must be made for each fragment.

The interrelation between fragmentation and routing is summarized by the following trade-off. On one hand breaking a packet into smaller fragments consumes network resources, due to the increased overhead dedicated to the additional fragments. On the other hand, fragmenting a packet into smaller sizes means more flexible routing with better load sharing. An example of the consequences of the common approach, i.e., making the routing decisions independent of the fragmentation, will be shown in the next section, after a more formal definition of the model.

Earlier attempts at solving the fragmentation problems[3] were at the basic-heuristic level. Later Bennett[4] presented a comparative analysis of the overhead cost at the receiver end due to fragmentation for various strategies. Recently, Bonuccelli [5] considered the problem taking a graph-theoretic approach. The network model there is of a graph with constant link lengths. After formulating several fragmentation problems, he showed that, in general, they are intractable. An optimal approach is presented for a restricted case. An initial treatment along the lines taken in this work was done in[6]. It was observed there that optimal performance can be achieved by a separate yet dependent treatment of the fragmentation and routing functions. Another interesting observation there is that fragmentation can be performed by a recursive scheme, similar to the one proposed in this paper. Based on a fragmentation-routing operators scheme, necessary conditions for achieving optimality were developed. However, no comprehensive analysis of the general fragmentation-routing problem seems to have been published to date.

The aim of this paper is to provide a comprehensive treatment of the general fragmentation-routing problem in a bifurcated routing environment. In Section II we formulate the problem as a network flow problem. In Section III we present various ways to perform fragmentation and show how a recursive approach leads to a tractable representation of the fragmentation-routing problem. With this formulation we obtain optimality conditions that are necessary and sufficient (Section IV). These conditions are the foundation for constructing centralized as well as distributed optimal algorithms (Section V). In Section VI

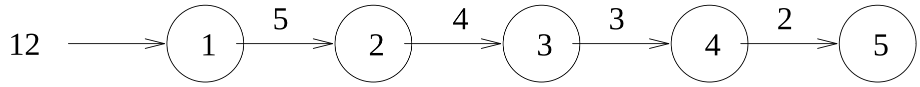

Figure 1: Example of fragmentation

we discuss implementation issues, and show how the size of the problem (and thus the number of operations of the algorithms) can be cut considerably in certain practical cases. The general results are demonstrated for cost functions representing total network delay. In section VII we show how other cost functions, in particular direct penalties for fragmentation, are easily accommodated within our results. The special case of virtual circuit networks is also discussed in that section.

II. THE MODEL

We model both the network and internetwork environments as a graph (V, E) where $V = \{1, 2, \cdots, n\}$ is the set of nodes and $E \subseteq V \times V$ is the set of edges. In a regular network environment the network nodes correspond to the nodes of the graph and the links between nodes correspond to the edges. In an internetwork environment the edges represent the individual networks whereas the nodes represent the gateways between them (a model identical to that of [7]).

We identify the nodes by numbers and refer to the edge (i, k) as the bidirectional link connecting node *i* with node *k*. We denote by N_i the set of node *i*'s neighbors. Each link (i, k) has capacity C_{ik} and a maximum packet size constraint T_{ik} on all packets. We assume the network topology does not change and that all messages sent over the link are correctly received within finite time after having been sent.

The fragmentation mechanism works as follows. Each node receiving a packet can perform one of two actions: forward the packet unfragmented to one of its neighbors or fragment it to several fragments and forward each of them to a neighbor (not necessarily the same neighbor for all fragments). Each fragment is applied a separate header specific to the link over which it is sent. Following the same philosophy as in [8] we do not allow en-route reassembly, i.e., messages are reconstructed from fragments only at the destination. This leads to the distinction between the information-data which is the data sent from source to destination, and overhead (header) data which are the added headers. It is the flow of information-data which must be preserved (headers are attached anew to every packet on every link) whereas the sum of the information and overhead data is the quantity limited by link capacity. For a given packet we refer to *info-size* as the size of the information part of the packet, and to *header-size* as the size of the header. Header size is a quantity unique to every link and is denoted h_{ik}. The sum of the info-size and header-size is referred to as *packet-size* (or *packet length*). The flow generated by the information-data as it traverses the network towards its destination is referred to as *info-flow*, whereas the info-flow over a link with the appropriate header attached is referred to as *link-flow*. The amount of info-flow of *t*-bits packets on link (i, k) destined to node *j* is denoted by $\hat{f}^t_{ik}(j)$. The total amount of info-flow on that link, destined to node *j*, is

$$\hat{f}_{ik}(j) \triangleq \sum_{t=1}^{T_{ik}} \hat{f}^t_{ik}(j),$$

the total amount of flow generated by *t*-packets is

$$\hat{f}^t_{ik} \triangleq \sum_{j=1}^{n} \hat{f}^t_{ik}(j),$$

and the total amount of info-flow is

$$\hat{f}_{ik} \triangleq \sum_{j=1}^{n} \hat{f}_{ik}(j) = \sum_{t=1}^{T_{ik}} \hat{f}^{t}_{ik}.$$

Corresponding to the info flows $\hat{f}^{t}_{ik}(j)$, $\hat{f}_{ik}(j)$ and $\hat{f}_{ik}$ we define the link-flow quantities $f^{t}_{ik}(j)$, $f_{ik}(j)$ and f_{ik}. The link-flows and info-flows are interelated by the quantity h_{ik}, which is the header-size for packets flowing through link (i,k). In other words, $f^{t}_{ik}(j) = \alpha^{t}_{ik} \hat{f}^{t}_{ik}(j)$, where we define $\alpha^{t}_{ik} \triangleq \dfrac{t + h_{ik}}{t}$.

We assume that packets enter the network according to an ergodic process, and denote by $r^{t}_{i}(j)$ the average rate of external (info-)flow of t-bit packets entering node i and whose destination is node j. These external packets may be generated in an any packet size up to some maximal size T which can be arbitrarily large but finite.

Let $D_{ik}(\cdot)$ denote the total cost (delay) per unit time of all traffic passing through link (i,k). We assume the following with regard to D_{ik}[9]:

P1. D_{ik} is a nonnegative continuous increasing function of f_{ik}, with continuous first and second derivatives.

P2. D_{ik} is convex.

P3. $\lim_{f_{ik} \to C_{i,k}} D_{ik}(f_{ik}) = \infty$.

The total cost per unit time is therefore

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

According to the cost function D_T, the price of fragmentation is due to the additional headers, which increase the link-flow. In general, one may consider additional adverse effects of fragmentation, e.g., due to reassembly. In Section VII we shall see that such a generalization can be incorporated into our model and results.

We are now able to show the relation between fragmentation and routing. consider, for example, the network depicted in Figure 2 in which the links are labeled according to the maximum packet sizes. Assume that the only flow in the network enters node 1 at a rate of r units/sec in packets of size 3. We assume that $h_{ik} = 1$ for all links and for convenience assume $D_{ik}(f_{ik}) = k f^{2}_{ik}$ for all links (capacity constraints are ignored). The classical routing process will route a fraction x of the incoming flow through node 2 which requires each packet to be fragmented into three fragments of size 1. The rest of the flow goes through node 3 and requires each packet to be fragmented into two fragments, of sizes 1 and 2. It can be shown that $x = .41$ is the optimal solution yielding a total cost of $3.28kr^2$. Adopting our approach to fragment the packets before a final routing decision is made yields a solution in which 96% of the incoming packets are fragmented into two fragments of sizes 1 and 2 and 4% of the packets are fragmented into three

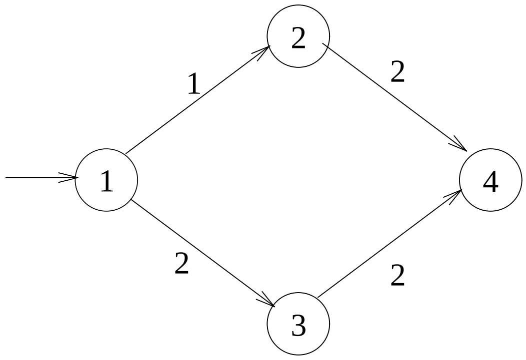

Figure 2: Example for routing and fragmentation relation

fragments of size 1 each. The large fragments will be routed through node 2 while the short ones through node 3. This yields a total cost of $2.88kr^2$. Yet a more striking difference between the two approaches is found when capacity constraints are considered. Suppose that in the above example $D_{ik}(f_{ik}) = \frac{f_{ik}}{C_{ik} - f_{ik}}$ and $C_{ik} = \frac{10}{11} r$ for all links. The classical routing process will fail to find a feasible solution, whereas for our approach feasible solutions do exist, the optimal one being to fragment 86% of the incoming packets into two fragments of sizes 1 and 2 and 14% of the packets into three fragments of size 1 each (as before, the large fragments will be routed through node 2 while the short ones through node 3). It is thus evident that ignoring the dependence between routing and fragmentation has adverse effect on network performance.

III. METHODS OF FRAGMENTATION

We proceed to describe the exact way in which fragmentation and routing are performed. First, we reiterate that since we deal with flows (and with flow-dependent cost functions), we can separate the fragmentation process from the routing process, i.e., a packet is first subjected to fragmentation, and then each resulting fragment (or the original packet, if unfragmented) undergoes a routing decision. The general configuration is depicted in Figure 3. Note that the fragmentation and routing processes work separately but do share information (represented by the dashed arrow in the figure). It is easy to verify that such a two-stage scheme is equivalent to one in which the routing and fragmentation decisions are made together.

The routing process consists of choosing the next neighbor for each fragment. The choice is made through a routing table, that specifies, according to the destination and the packet (or fragment) length, the probability of sending it to each neighbor.

The fragmentation process decides to which sizes should a packet be broken. The straightforward approach is that of the *Complete Fragmentation* (CF) machine: given a packet of size t, the machine decides, in a single step, into which fragments the packet should be broken, if at all (Figure 4). For example, the CF machine may decide to break a 800-bit ($t = 800$) packet into one 300 bits fragment, two 200 bits fragments and one 100 bits fragment.

In general there is an exponential number of ways to break a packet into fragments, leading to an intractable analysis when the straightforward (CF) approach is used. We overcome this problem by introducing alternative machines that perform the fragmentation recursively.

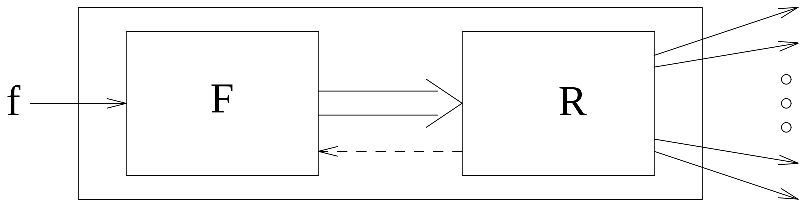

Figure 3: Fragmentation-routing function separation

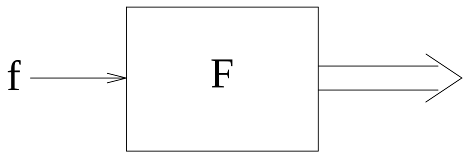

Figure 4: The complete fragmentation machine

The first of the recursive machines is the *Two-Fragment Recursive* (TFR) machine (see Figure 5). This machine handles a packet through several possible stages. In every stage, the machine decides whether the t-bits packet should undergo fragmentation or exit the fragmentation process. If the decision is made to fragment, the packet is broken into two fragments, of sizes, say, τ and $t-\tau$ and each of these two fragments is fed back into the machine and is considered as a new independent packet (hence the recursion). For example, the above mentioned 800 bit packet may be broken by the machine into two fragments of size 300 and 500 bit which are fed back to it; the former is then broken into 200 and 100 bit fragments and the latter to a 200 and 300 bit fragments.

The second recursive machine is the *Two-Fragment Semirecursive* (TFS) machine (see Figure 6). This machine is similar to the TFR machine, except that the smallest of the two fragments, say τ ($\tau \le t-\tau$) is not fed back to the machine, but rather is sent directly to the routing process. (By convention, when the two fragments are of equal size one of them is arbitrarily regarded as the smaller one). Our 800 bit packet will be broken into 300 and 500 bit fragments with only the latter being fed back to the machine to be subsequently fragmented into 200 and 300 bit fragments, etc.

We now show that all three machines are equivalent in terms of fragmentation and routing. The key point in establishing equivalence is the straightforward observation that any set of fragments produced by one fragmentation machine can be produced by any other. In other words, all these machines are equally powerful in terms of fragmentation. Thus, given a set of inputs, any fragmentation-routing solution achieved by one machine can be achieved by the other. One should note, though, that there exist fragmentation strategies that are possible in just part of the machines but not in all of them. For example, the CF machine may decide that packets of length 500 are to be split into fragments of sizes 300, 100 and 100,

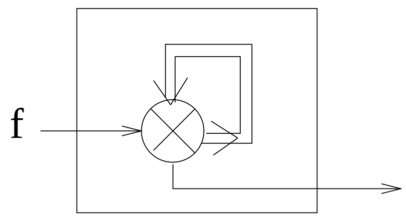

Figure 5: The two-fragment recursive (TFR) fragmentation machine

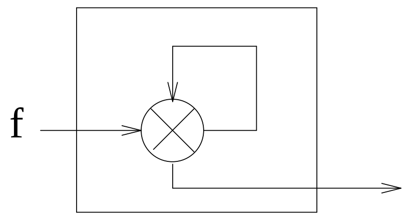

Figure 6: The two-fragment semirecursive (TFS) fragmentation machine

whereas packets of length 400 should not be split at all; such a scheme cannot be implemented in the TFS machine, since one would need to split first the 500 packet into fragments of sizes 400 and 100. However, the observation that comes to rescue is that we only care about external (link) flows. Thus, by properly regulating the rates of fragmentation, one can split part of the 400 packets into fragments of 300 and 100, while leaving the rest unfragmented, and come out with external flows that are identical to those of the CF case. Note that the relative part of 400 packets that are fragmented would correspond to the relative amount of 400 fragments made from 500 packets compared to that of original 400 packets. We thus have:

Lemma 1: The CF, TFR, and TFS machines will all arrive at the same optimal solution for the same fragmentation-routing problem. □

In the next section we show that both the TFR and TFS machines lead to tractable analysis. There are, however, other differences, such as implementation considerations, that make one machine preferable over the other in certain environments.

The TFR machine is especially suitable for parallel architectures. In this machine packets are fragmented and re-fragmented by applying the same procedure over and over again. Once a packet is fragmented into two packets, the order in which these are treated is immaterial, and hence parallel computation is called for to expedite processing.

The TFS machine is more suitable for serial computation. Since a routable fragment is generated in every cycle of the fragmentation process, this fragment can be handed to the routing function for forwarding to the next neighbor. Thus a pipe can be envisioned in which fragments are generated and forwarded immediately--a process that can reduce processing delay in the node. From a more intuitive aspect, the TFS machine seems to lead us in the "right" direction by producing in every cycle a routable fragment.

For the TFR and TFS machines, we denote by $g_i^{t,\tau}(j)$ the amount of info-flow of t-bit packets destined to node j, which is broken into fragments of sizes τ and $t-\tau$. By convention (and without loss of generality), the second (τ) superscript refers to the smaller fragment i.e., $g_i^{t,\tau}(j)$ is defined only for $\tau \le t/2$. It is important to note that the quantity $g_i^{t,\tau}(j)$ stands for the *total* amount of info-flow that undergoes such a fragmentation i.e., it includes both the new τ and $t-\tau$ fragments. In the TFR machine the fragments of sizes τ and $t-\tau$ accounted for in $g_i^{t,\tau}(j)$ may subsequently undergo further fragmentations whereas in the TFS machine only the larger ($t-\tau$) fragment may undergo subsequent fragmentation.

IV. ANALYSIS

In this section we analyze the fragmentation-routing problem and derive conditions for optimal performance. In the previous section we have introduced three fragmentation machines and proved their equivalence. We indicated that the TFR and TFS machines lend themselves to easier analysis than the CF machine. We shall thus not discuss the CF machine and concentrate on the other two.

Starting with the TFR machine our target cost function is

$$D_T = \sum_{ik} D_{ik}(f_{ik}) = \sum_{ik} D_{ik}(\sum_t f_{ik}^t) = \sum_{ik} D_{ik}(\sum_t \sum_j f_{ik}^t(j)) = \sum_{ik} D_{ik}(\sum_t \sum_j \alpha_{ik}^t \cdot \hat{f}_{ik}^t(j)) \ .$$

We organize the free variables as a vector

$$fr = \left[\overline{f}\ ,\ \overline{g}\right]$$

where $\overline{f}$ and $\overline{g}$ are themselves vectors whose components are

$$\overline{f} = \{\hat{f}_{ik}^t(j) \mid \ (i,k) \in E,\ \ j \in V,\ \ 1 \le t \le T_{ik}\}$$

$$\overline{g} = \{\ g_i^{t,\tau}(j) \mid i \in V,\ \ j \in V,\ \ 1 \le t \le T,\ \ 1 \le \tau \le t/2\}\ ,$$

$\overline{f}$ is a vector describing flows between nodes and is defined per link and $\overline{g}$ is a node internal fragmentation vector and is defined per node. Note that from the definition of $\overline{f}$ follows that there is no flow of t-packets on a link (i,k) for $t > T_{ik}$. (We order these components in an arbitrary but predetermined way.)

Let $\boldsymbol{F}$ be the set of all vectors fr. D_T is thus a function over $\boldsymbol{F}$. Since not all members of $\boldsymbol{F}$ are flow vectors we define $\boldsymbol{F_c}$ as the subset of $\boldsymbol{F}$ containing all flow vectors. In other words, to be a member of $\boldsymbol{F_c}$ a vector fr must obey the following constraints:

C1. Nonnegativity of info-flows: $\hat{f}_{ik}^t(j) \ge 0$.

C2. Link capacity constraint: $\sum_t \sum_j f_{ik}^t(j) = \sum_t \sum_j \alpha_{ik}^t \cdot \hat{f}_{ik}^t(j) < C_{ik}$.

C3. Nonnegativity of fragments flow: $g_i^{t,\tau}(j) \ge 0$.

C4. Conservation of flow:

$$\sum_k \hat{f}_{ik}^t(j) \ = \sum_k \hat{f}_{ki}^t(j)\ +\ r_i^t(j)\ -\sum_{0<\tau\le\frac{t}{2}} g_i^{t,\tau}(j) + \sum_{t<\tau\le 2t} \frac{t}{\tau} \cdot g_i^{\tau,\tau-t}(j) + \sum_{2t\le\tau\le T} \frac{t}{\tau} \cdot g_i^{\tau,t}(j)$$

The expression in constraint C4 merits some explanation. The left hand-side is the total info-flow of t-packets out of node i. This flow can be constructed (as appears on the right hand-side of the expression) from t-packet info-flow into node i from its neighbors and similar external input to node i (see Figure 7).

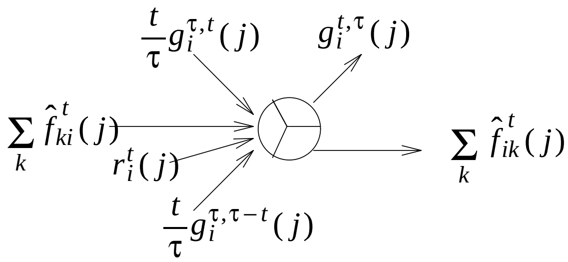

Figure 7: Nodal flows for the TFR machine

From this flow we have to subtract any flow of *t*-packets that was fragmented into smaller fragments (the third term) and add info-flow from packets of sizes larger than *t* that were fragmented into *t*-packets (the fourth and fifth term). By our definition, $g_i^{\tau,t}(j)$ accounts for the entire τ-packet info-flow that was fragmented (and so does $g_i^{\tau,\tau-t}(j)$), so we have to account separately for the case where the *t*-packet is the larger fragment and the case where it is the smaller . Note also the $\frac{t}{\tau}$ factor which adjusts for the fact that $g_i^{\tau,t}(j)$ accounts for τ-packets and our expression balances the flow of *t*-packets.

With the above definitions, the Fragmentation and Routing (FR) problem, for the TFR machine, is formulated as follows:

Problem FR: Minimize D_T over $\boldsymbol{F_c}$.

Before pursuing the solution for problem FR we first establish some convexity properties.

Lemma 2: D_T is a convex function over the convex set $\boldsymbol{F_c}$.

Proof: Being defined by linear constraints, $\boldsymbol{F_c}$ is clearly a convex set. Since D_T is a (finite) sum of D_{ik}'s it suffices to show that each D_{ik} is convex over $\boldsymbol{F_c}$. D_{ik} is a function of a linear combination of the free variables. Since D_{ik} is convex in f_{ik}, it follows that D_T is convex over $\boldsymbol{F_c}$. □

Having established the convexity properties of the target function and the domain region, we can derive the following conditions for optimality.

Theorem 1: A necessary and sufficient condition for a vector $fr^* = \{\hat{f}_{ik}^{t*}(j)\} \cup \{g_i^{t,\tau*}(j)\}$, that is feasible for the TFR machine, to solve Problem FR (with that machine) is the existence of a set of nonnegative numbers (Lagrange multipliers) $\{\lambda_i^t(j) | \quad 1 \le i \le n, \quad 1 \le j \le n, \quad 1 \le t \le T\}$, such that:

(OC1) for all $1 \le j \le n$, for all $1 \le i \le n$ ($i \ne j$), for all $k \in N_i$, for all $1 \le t \le T_{ik}$:

$$\lambda_i^t(j) \begin{cases} = \lambda_k^t(j) + \alpha_{ik}^t \cdot D'_{ik} & \hat{f}_{ik}^{*t}(j) > 0 \\ \le \lambda_k^t(j) + \alpha_{ik}^t \cdot D'_{ik} & \hat{f}_{ik}^{*t}(j) = 0 \end{cases}$$

(OC2) for all $1 \le j \le n$, for all $1 \le i \le n$ ($i \ne j$), for all $1 \le t \le \max_k Tki$, for all $1 \le \tau \le t/2$:

$$\lambda_i^t(j) \begin{cases} = \dfrac{t-\tau}{t}\lambda_i^{t-\tau}(j) + \dfrac{\tau}{t}\lambda_i^{\tau}(j) & g_i^{*t,\tau}(j) > 0 \\ \le \dfrac{t-\tau}{t}\lambda_i^{t-\tau}(j) + \dfrac{\tau}{t}\lambda_i^{\tau}(j) & g_i^{*t,\tau}(j) = 0 \end{cases}$$

(OC3) for all $1 \le j \le n$, for all $1 \le t \le \max_i T_{ij}$:

$$\lambda_j^t(j) = 0$$

Proof: These are a special case of the Kuhn-Tucker conditions for optimality. A direct proof is presented in the Appendix.

□

The optimality conditions (OC1)-(OC3) have an intuitive meaning: the Lagrange multiplier $\lambda_i^t(j)$ is the cost of sending an infinitely small incremental flow of packets of length *t* from node *i* to destination *j*. This incremental *t*-packet flow can either be routed as such to some neighbor of *i* or, alternatively, can be split into two fragments. The right hand-side of condition (OC1) is the incremental cost of routing a *t*-fragment from node *i* through node *k*. Thus, condition (OC1) states that, at optimum, the incremental cost is equal for all neighbors to which node *i* actually sends *t*-packets, and is not lower for any other neighbor. Condition (OC2) states that, at optimum, fragmentation is done only when it is "worthwhile" i.e., when the sum of the incremental costs of handling the two resulting fragments is not more expensive than sending the packet as is. Condition (OC3) simply states that the incremental cost at the destination is zero.

We proceed to analyze the TFS machine. Recall that in the TFS machine whenever a packet is fragmented one of the fragments must be routed, whereas the other may be considered for additional fragmentation. Therefore, one must distinguish between two types of *t*-packets flows. The first *(type-A*) is the flow of packets that may still be fragmented in the current node, while the second *(type-B*) is the flow of packets for which further fragmentation in the current node is prohibited. Type-A flow contains both packets whose original length is *t*, as well as fragments that are the larger of two fragments produced in the current node.

To properly capture the conservation of flow we define the quantity $x_i^t(j)$, as the portion of type-A flow of *t*-packets at node *i* destined to node *j* that *does not* undergo further fragmentation (see Figure 8). The constraints for the TFS machine are therefore as follows:

C1'-C3' As for the TFR machine.

C4' Conservation of fragmented flow: $x_i^t(j) = \sum_{k \in N_i} \hat{f}_{ki}^t(j) + r_i^t(j) - \sum_{1 \le \tau \le \frac{t}{2}} g_i^{t,\tau}(j) + \sum_{t < \tau \le 2t} \frac{t}{\tau} \cdot g_i^{\tau,\tau-t}(j)$.

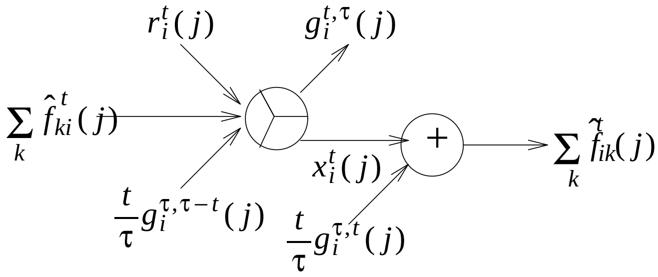

Figure 8: Nodal flows for the TFS machine

C5' General conservation of flow: $\sum_{k\in N_i} \hat{f}_{ik}^t(j) = x_i^t(j) + \sum_{\tau\geq 2t} \frac{t}{\tau}\cdot g_i^{\tau,t}(j)$.

C6' Nonnegativity of unfragmented type-A flow: $x_i^t(j) \geq 0$.

It is easy to verify that here too we face the problem of minimizing a convex function over a convex set. Following a similar process as for the TFR machine, we derive the necessary and sufficient optimality conditions for the TFS machine.

Theorem 2: A necessary and sufficient condition for a vector $fr^* = \{\hat{f}_{ik}^{t*}(j)\} \cup \{g_i^{t,\tau *}(j)\}$, that is feasible for the TFS machine, to solve Problem FR (with that machine) is the existence of a set of nonnegative numbers (Lagrange multipliers)

$$\{\lambda_i^t(j) \mid 1\leq i\leq n,\ 1\leq j\leq n,\ 1\leq t\leq T\} \cup \{\bar{\lambda}_i^t(j) \mid 1\leq i\leq n,\ 1\leq j\leq n,\ 1\leq t\leq T\},$$

such that:

(OC1') for all $1\leq j\leq n$, for all $1\leq i\leq n$ $(i\neq j)$, for all $k\in N_i$, for all $1\leq t\leq T_{ik}$:

$$\bar{\lambda}_i^t(j) \begin{cases} = \lambda_k^t(j) + \alpha_{ik}^t\cdot D'_{ik} & \hat{f}_{ik}^{*t}(j) > 0 \\ \leq \lambda_k^t(j) + \alpha_{ik}^t\cdot D'_{ik} & \hat{f}_{ik}^{*t}(j) = 0 \end{cases}$$

(OC2') for all $1\leq j\leq n$, for all $1\leq i\leq n$ $(i\neq j)$, for all $1\leq t\leq \max_k Tki$, for all $1\leq\tau\leq\frac{t}{2}$:

$$\lambda_i^t(j) \begin{cases} = \frac{t-\tau}{t}\lambda_i^{t-\tau}(j) + \frac{\tau}{t}\bar{\lambda}_i^{\tau}(j) & g_i^{*t,\tau}(j) > 0 \\ \leq \frac{t-\tau}{t}\lambda_i^{t-\tau}(j) + \frac{\tau}{t}\bar{\lambda}_i^{\tau}(j) & g_i^{*t,\tau}(j) = 0 \end{cases}$$

(OC3') for all $1\leq j\leq n$, for all $1\leq i\leq n$ $(i\neq j)$, for all $1\leq t\leq \max_k T_{ik}$:

$$\lambda_i^t(j) \begin{cases} =\overline{\lambda}_i^t(j) & x_i^{*t}(j)>0 \\ \le\overline{\lambda}_i^t(j) & x_i^{*t}(j)=0 \end{cases}$$

(OC4') for all $1\le j\le n$, for all $1\le t\le \max_i T_{ij}$:

$$\lambda_j^t(j)=\overline{\lambda}_j^t(j)=0$$

Proof: Similar to that of Theorem 1.

□

The intuition behind these conditions is as follows. The Lagrange multiplier $\lambda_i^t(j)$ is the cost of handling an infinitely small incremental flow of t-packets from node i to destination j, when subsequent fragmentation at node i is allowed. $\overline{\lambda}_i^t(j)$ is the cost of sending such an incremental flow, but when no further fragmentation may be performed at node i (i.e., routing is required). The right hand-side of condition (OC1') is the cost of handling an incremental packet if it is sent to some neighbor k and thus, condition (OC1') states that, at optimum, the incremental cost of routing without further fragmentations at the present node is equal for all neighbors to which node i actually sends t-packets, and is not lower than for any other neighbor (note that at the next node, k, fragmentation is allowed, and $\lambda_k^t(j)$ is the corresponding cost). Condition (OC2') states that, at optimum, fragmentation is done only when it is "worthwhile". Note that the cost of the smaller fragment of size τ is $\overline{\lambda}_i^\tau(j)$ (and not $\lambda_i^\tau(j)$), since for that fragment further fragmentation is prohibited. Condition (OC3') simply states that the incremental cost of handling a packet when fragmentation is allowed cannot be higher than when fragmentation is prohibited, and the two are equal when part of type-A flow is routed unfragmented.

In the next section we use the optimality conditions as a baseline for an optimal distributed algorithm. It should be noted, though, that each set of optimality conditions was obtained only for the case that all nodes in the network use the same type of fragmentation machine (namely, TFR for OC1-OC3 and TFS for OC1'-OC4'). This means that any algorithm which is based on these conditions would be applicable only to networks that use a homogeneous fragmentation scheme. On the other hand, we mentioned that each fragmentation machine has its particular advantages, thus different nodes may be implemented with different types of fragmentation machine. We therefore seek optimality conditions that are machine-independent, i.e., that hold in a network using any arbitrary combination of TFR and TFS machines. From the standpoint of a distributed algorithm, this means that a node using one machine may assume that all nodes use the same machine, although actually this may not be so. We observe that message exchange among nodes both according to (OC1)-(OC3) and (OC1')-(OC4') is achieved by the exchange of the values of $\lambda_i^t(j)$. Thus, all we need is to show that there exists a set of λ's that fulfills both (OC1)-(OC3) and (OC1')-(OC4') for the same network and at optimality. As a first step, we show that (OC1)-(OC3) the optimality conditions of TFR machines are necessary and sufficient optimality conditions also for TFS machines.

It might be tempting to derive the required result directly from the equivalence between the two machines, as stated by Lemma 1. However, this simple argument is invalid, since the sequences of fragmentation operations (for the same optimal vector of external link flows) may be different for the two machines; indeed, we recall that there are sequences of the TFR machine that are not feasible in the TFS machine. Thus, (OC2) and (OC2') are not comparable, since the values of i, t and τ for which $g_i^{t,\tau}(j) > 0$ may be different for TFR and TFS. A more rigorous argumentation is therefore needed.

Lemma 3: (OC1)-(OC3) are necessary and sufficient conditions for optimality also for the TFS machine, i.e.: A necessary and sufficient condition for a vector $fr^* = \{\hat{f}_{ik}^{t*}(j)\} \cup \{g_i^{t,\tau*}(j)\}$, that is feasible for the TFS machine, to solve Problem FR (with that machine) is the existence of a set $\{\lambda_i^t(j) \mid 1 \le i \le n,\ 1 \le j \le n,\ 1 \le t \le T\} \cup \{\overline{\lambda}_i^t(j) \mid 1 \le i \le n,\ 1 \le j \le n,\ 1 \le t \le T\}$ for which (OC1)-(OC3) hold.

Proof: See the Appendix.

□

We thus have:

Theorem 3: Given a network with any combination of TFR and TFS machines, a necessary and sufficient condition for a vector $fr^* = \{\hat{f}_{ik}^{t*}(j)\} \cup \{g_i^{t,\tau*}(j)\}$ (for which, for each i, $\{g_i^{t,\tau*}(j)\}$ is feasible for the fragmentation machine implemented at i), to solve Problem FR is the existence of a set of nonnegative numbers (Lagrange multipliers) $\{\lambda_i^t(j) \mid 1 \le i \le n,\ 1 \le j \le n,\ 1 \le t \le T\}$ that fulfills (OC1)-(OC3).

Proof: See the Appendix. (Note that the proof relies on some terms defined in the next section.)

□

V. ALGORITHMS

In this section we explore several algorithms to achieve optimal fragmentation and routing. Given the analysis of the previous section, we focus only on the TFR machine model.

Routing in computer networks has been investigated extensively in the last three decades. In terms of optimization, bifurcated routing has been shown to achieve optimal performance and several approaches have been developed[10,11]. These approaches treat the routing problem as a multicommodity minimum cost flow problem. It has been shown that under such circumstances optimal routing results only if traffic flows along *Minimum First Derivative Length (MFDL)* paths i.e., paths in which the sum of the first derivative of the link cost functions is minimal.

The computational approach taken is an iterative one, in which a cost function is reduced in each iteration by diverting some flow from expensive routes to less expensive ones. The classical solutions determine inexpensive routes by means of the first derivative of the cost function with respect to the flows. The flow deviation algorithm[11] determines inexpensive routes by searching for shortest paths according to the MFDL metric. Gallager's algorithm [10] exchanges marginal costs among nodes and diverts traffic to those routes where the marginal cost is the smallest.

The structure of the optimality conditions (OC1)-(OC3) (or, equivalently, (OC1')-(OC4')) hints that such approaches are also well suited for our environment. Indeed, we shall indicate how some classical optimal routing algorithms can be accommodated to solve the FR problem in both centralized and distributed fashions. Since in both cases we rely on well-known and extensively studied algorithms, we concentrate on the aspects that are particular to our environment.

A. Centralized Algorithms

A class of centralized algorithms (e.g., [11]) devised for the classical routing problem rely on the ability to identify at each iteration an MFDL path, through which flow should be improved. In order to accommodate such approaches in our environment, we need to extend the concepts of first derivative lengths and first derivative length paths. This extension is not straightforward since the notion of a path itself (and an MFDL path in particular) is not well defined given the fragmentation that might take place. Our definition includes all the paths traveled by all the fragments of a given packet. In other words, consider an origin node $i \in V$, a destination node $j \in V$ and a packet of info-size t. We define a *generalized t-path* between node i and node j as the collection of simple paths in the network that corresponds to a possible journey of a single t-packet originated at i, and all its descendant fragments, until their arrival in j. Given a set of link flows, the *first derivative length* of a generalized t-path is defined as the weighted sum of the first derivative of the link costs of all links in the path. Note that it is possible that certain links will be counted more than once in a generalized path and their weight will then be added several times appropriately.

We now define those terms in a more formal way. A *t-path* p_{ij}^t between any two nodes i and j is a sequence of nodes $(i = v_0, v_1, \cdots v_r = j)$ such that, for all $0 \le l \le r-1$ $(v_l, v_{l+1}) \in E$ (i.e., the path is topologically feasible) and $T_{v_l, v_{l+1}} \ge t$ (i.e., t-packets are acceptable through the entire path), and for all $0 \le l < m \le r$ $v_l \ne v_m$ (i.e., the path is simple). For a given set of link flows, the first derivative length of a t-path p_{ij}^t is defined as $FDL(p_{ij}^t) \overset{\Delta}{=} \sum_{l=0}^{r-1} \alpha_{v_l, v_{l+1}}^t \cdot D'_{v_l, v_{l+1}}$.

The generalized t-path between nodes i and j, denoted gp_{ij}^t, is defined recursively. For $t = 1$, since fragmentation is impossible, a generalized path gp_{ij}^1 is a 1-path in the network and the FDL of the generalized 1-path is identical to that of the regular 1-path. For $t \ge 2$ we use recursion. Let v be some node (possibly $v = i$ or $v = j$) and $\tau \le t/2$. The generalized t-path gp_{ij}^t is the concatenation of $p_{i,v}^t$, the t-path from i to v (for $v = i$ it is an empty path), together with the two generalized paths $gp_{v,j}^\tau$ and $gp_{v,j}^{t-\tau}$ (for $v = j$ they are empty paths). The first derivative length of a generalized t-path is correspondingly defined as

$$FDL(gp_{ij}^t) \overset{\Delta}{=} FDL(p_{i,v}^t) + \frac{\tau}{t} FDL(gp_{v,j}^\tau) + \frac{t-\tau}{t} FDL(gp_{v,j}^{t-\tau}) \quad .$$

Finally, for an origin node $i \in V$, a destination node j and an integer t, a *t-minimal generalized path* between i and j is a t-generalized path between i and j whose FDL is minimal. The FDL of that generalized path is referred to as the t-MFDL between i and j. A t-minimal generalized path shall also be referred to as a t-MFDL path.

Since we intend to use MFDL-type algorithms, our next step is to devise an algorithm for finding t-MFDL paths. The algorithm works in phases, starting with $t = 1$, where in the t-th phase t-MFDL paths are found for all origin-destination pairs, based on results obtained for previous phases. Considering a destination node j, when the t-th phase begins we look for the best way to fragment a t-packet at each node. This is done by identifying at each node i the value of τ in the range $1 \le \tau \le t/2$, for which the weighted sum of the τ-MFDL and the $(t-\tau)$-MFDL between i and j is minimal. The value of that minimal sum is incorporated in the t-th phase by assigning it to a fictitious link that is added between i and j. Every nonfictitious link (i,k) is assigned the length $\alpha_{ik}^t \cdot D'_{ik}$. The t-th phase then consists of finding shortest paths between all pair of nodes on the extended graph based on the assigned edge lengths. Choosing a fictitious link (say, from k to j) in a shortest path from i to j is interpreted as fragmenting a packet at k and incorporating the corresponding τ- and $(t-\tau)$-MFDL paths (from k to j) in the t-MFDL path from i to j.

Algorithm MFDL

1. Set $t \leftarrow 1$ (first phase).

   a. For all $(i,k) \in E$ set $l_{ik} \leftarrow \alpha_{ik}^{1} \cdot D'_{ik}$.

   b. Run an all-pairs shortest-paths algorithm based on l_{ik} (see e.g.,[12]). For all $i,j \in V$ assign the shortest path between i and j to $MFDLP_{ij}^{1}$ and its length to $MFDL_{ij}^{1}$.

2. $t \leftarrow t+1$ (the t-th phase)

   a. For all $i,j \in V$:

      i. Find a θ, $1 \le \theta \le t/2$, such that for all $1 \le \tau \le t/2$

      $$\frac{\theta}{t} MFDL_{ij}^{\theta} + \frac{t-\theta}{t} MFDL_{ij}^{t-\theta} \le \frac{\tau}{t} MFDL_{ij}^{\tau} + \frac{t-\tau}{t} MFDL_{ij}^{t-\tau}.$$

      ii. Add a fictitious link $(\tilde{ij})$ and assign it the length

      $$l_{\tilde{ij}} \leftarrow \frac{\theta}{t} MFDL_{ij}^{\theta} + \frac{t-\theta}{t} MFDL_{ij}^{t-\theta} \quad .$$

      iii. $\tau_{ij}^{t} \leftarrow \theta$.

   b. For all $(i,k) \in E$ let $l_{ik} \leftarrow \begin{cases} \alpha_{ik}^{t} \cdot D'_{ik} & t \le T_{ik} \\ \infty & t > T_{ik} \end{cases}$.

   c. Run an all-pair shortest-paths algorithm on the augmented network based on l_{ik}. For all $i,j \in V$ assign the shortest path between i and j to $MFDLP_{ij}^{t}$ and its length to $MFDL_{ij}^{t}$.

   d. If $t < T$ then remove fictitious links and repeat step 2; else stop.

When the algorithm stops, a corresponding t-minimal generalized path for i,j can be constructed from $MFDLP_{ij}^{t}$ in the following recursive way. If $MFDLP_{ij}^{t}$ does not contain fictitious links, then it is the required generalized path (which, in this case it is a regular path). Otherwise, $MFDLP_{ij}^{t}$ ends with a fictitious link, say $(\tilde{k,j})$; the required generalized path is constructed by concatenating the simple (possibly empty) path which constitutes $MFDLP_{ij}^{t}$ as a prefix up to node k, together with the τ_{kj}^{t}-MFDL and $(t-\tau_{kj}^{t})$-MFDL generalized paths. It is easy to verify the validity of the algorithm, and thus the formal proof is omitted.

B. Distributed Algorithms

The structure of the necessary and sufficient conditions (OC1)-(OC3) (in fact, also (OC1')-(OC4')) resembles that of the classical network routing problem (see [9,10]). This resemblance enables a straightforward adaptation of the optimal distributed routing algorithm of [10] in order to solve distributedly the FR problem.

Gallager's algorithm [10] is based on the consecutive exchange and updating of λ values, in a way that tries to make these values fulfill the optimality conditions. In addition, each node calculates (or measures) the cost function derivatives of its outgoing links. The algorithm works in iterations and per destination node *j*. In each iteration, each node *i* waits until it receives the value of $\lambda_k(j)$ from each *downstream* neighbor *k* (i.e., a neighbor to which *i* sends flow destined to *j*). Then, node *i* calculates its own $\lambda_i(j)$ by adding to each $\lambda_k(j)$ the value of the derivative D'_{ik}, and averaging the obtained sums appropriately. *i* then sends $\lambda_i(j)$ to all neighbors. This λ-exchange protocol starts with node *j* sending $\lambda_j(j)=0$. A loop-avoidance blocking mechanism is incorporated into the algorithm to assure loop-free routing. After sending its $\lambda_i(j)$, node *i* updates its routing: flow is diverted from neighbors for which $D'_{ik}+\lambda_k(j)$ is not minimal to a neighbor for which the minimum is obtained.

We proceed to discuss the corresponding algorithm for the FR problem. It is assumed that each node *i* is cognizant of the values α_{ik}^t for all $k \in N_i$ and *t*. Also, node *i* either calculates or measures the cost function derivatives D'_{ik} on its outgoing links (see [13]). The new Distributed FR (DFR) algorithm also works iteratively, per destination *j*. However, each iteration is composed of *phases*, one for each value of *t*. The first phase ($t=1$) is identical to an iteration of the original algorithm (using $\alpha_{ik}^1 \cdot D'_{ik}$ and $\lambda_i^1(j)$ for D'_{ik} and $\lambda_i(j)$). A typical *t*-phase ($t>1$) begins at node $i \neq j$ after all prior phases ended at that node (at node *j*, the phase consists of sending $\lambda_j^t(j)=0$ to all neighbors at the beginning of the phase). Upon entering the *t*-phase, node *i* calculates the value of

$$\Lambda_i^{t,\tau}(j) \triangleq \frac{t-\tau}{t} \cdot \lambda_i^{t-\tau}(j) + \frac{\tau}{t} \cdot \lambda_i^{\tau}(j)$$

for every $\tau \leq \frac{t}{2}$. Then, node *i* waits until it has received the value of $\lambda_k^t(j)$ from each *t*-downstream neighbor *k* (i.e., a neighbor to which *i* sends flow of *t* packets destined to *j*). Then, node *i* calculates its $\lambda_i^t(j)$ from the values of $\alpha_{ik}^t \cdot D'_{ik}$ for all downstream neighbors *k and* also the precalculated $\Lambda_i^{t,\tau}(j)$, i.e.: the value of $\lambda_i^t(j)$ incorporates the cost of fragmented flow as well as routed flow of type *t*. Having calculated $\lambda_i^t(j)$, node *i* sends it to all its neighbors, and proceeds to update its actual fragmentation and routing processes. This is done by identifying the "best" neighbor for routing (while using a loop-avoidance mechanism such as that of [10]) and the "best" value of τ for fragmenting; then, the cheapest of the two alternatives is chosen. If routing (through, say, neighbor k^*) is the cheapest option then node *i* decreases the amount of fragmentation of *t*-packets (i.e., each $g_i^{t,\tau}(j)$ is decreased), and decreases the routed flow through all neighbors $k \neq k^*$ (i.e., $\hat{f}_{ik}^t(j)$ is decreased for $k \neq k^*$); these flow reductions are balanced by an increase in the amount of routing through neighbor k^*. Otherwise, if fragmentation is cheaper (into, say, τ^* and $t-\tau^*$), node *i* increases the amount of fragmentation into fragments of sizes $\tau^*, t-\tau^*$ and decreases the amount of fragmentation into all other sizes as well as the amount of routed flow through all neighbors.

Following is a sketch of a typical t-phase ($t > 1$) at a node $i \neq j$.

Algorithm DFR (t-phase, destination j, node $i \neq j$)

(Begins after all previous phases for j end at i)

1. (a) For all $\tau \leq \frac{t}{2}$: $\Lambda_i^{t,\tau}(j) \leftarrow \frac{t-\tau}{t} \cdot \lambda_i^{t-\tau}(j) + \frac{\tau}{t} \cdot \lambda_i^{\tau}(j)$.

   (b) $\lambda_i^t(j)_1 \leftarrow \min_{k \in N_i} [\alpha_{ik}^t \cdot D'_{ik} + \lambda_k^t(j)]$.

   (c) $\lambda_i^t(j)_2 \leftarrow \min_{\tau \leq \frac{t}{2}} [\Lambda_i^{t,\tau}]$.

2. If $\sum_k \hat{f}_{ik}^t(j) + \sum_\tau g_i^{t,\tau}(j) \overset{\Delta}{=} S > 0$ then

$$\lambda_i^t(j) \leftarrow \sum_{k \in N_i} \frac{\hat{f}_{ik}^t(j)}{S} \cdot [\alpha_{ik}^t \cdot D'_{ik} + \lambda_k^t(j)] + \sum_{\tau \leq \frac{t}{2}} \frac{g_i^{t,\tau}(j)}{S} \cdot \Lambda_i^{t,\tau}(j)$$

   else

$$\lambda_i^t(j) \leftarrow \min\{\lambda_i^t(j)_1, \lambda_i^t(j)_2\}$$

3. Send $\lambda_i^t(j)$ to all neighbors.

4. If $\lambda_i^t(j)_1 \leq \lambda_i^t(j)_2$ then

   a. Let k^* be such that $\alpha_{ik^*}^t \cdot D'_{ik^*} + \lambda_{k^*}^t(j) = \lambda_i^t(j)_1$.

   b. Decrease $\hat{f}_{ik}^t(j)$ for all $k \neq k^*$, decrease $g_i^{t,\tau}(j)$ for all $\tau \leq \frac{t}{2}$, increase $\hat{f}_{ik^*}^t(j)$.

   else

   a. Let τ^* be such that $\Lambda_i^{t,\tau^*}(j) = \lambda_i^t(j)_2$.

   b. Decrease $\hat{f}_{ik}^t(j)$ for all $k \in N_i$, decrease $g_i^{t,\tau}(j)$ for all $\tau \neq \tau^*$, increase $g_i^{t,\tau^*}(j)$.

For clarity we have omited in the above description the loop-avoidance mechanism, as well as the specification of the exact amount of flow diverted for each k and τ in the last step. These are performed in a way similar to that described in [10] and[9]. We mention that the amount of flow decreased from a k or a τ is proportional to the relative "badness" of that choice compared to the best choice (k^* or τ^*).

A proof of convergence to optimum can be carried out following the same lines as in [10], and is omited due to the similarity. Note that as is the case with the algorithm of [10], convergence is assured only if the amount of change at each iteration and phase is small enough. Improved versions of this algorithm which result in faster convergence, can be accommodated for handling the FR problem as well.

VI. REDUCING FRAGMENTATION COMPLEXITY

Thus far we consideried all possibilities of fragmenting a packet into smaller pieces. One drawback of this approach is encountered in the algorithms discussed above: in each phase they iterate for every t and $\tau \le \frac{t}{2}$, resulting in a total of $O(T^2)$ operations (per basic iteration). In this section we consider a more efficient way of operation.

In an attempt to free the above algorithms from the need to consider each and every value of t we use a "quantized" approach so that only specific values are considered. In other words, we shall not consider every possible way of fragmenting a packet, but only a limited set of possibilities, corresponding to a constrained set of allowable fragment sizes. For example, if the allowable sizes are $2, 4, 6$ etc., then we shall not consider fragmenting a packet of, say, size $t = 8$ into fragments of sizes $\tau = 3$ and $t - \tau = 5$.

Formally, let $Q = \{\tau_1, \tau_2, \ldots, \tau_q\}$ be a set of fragment sizes ($\tau_1 < \tau_2 < \cdots < \tau_q$). A packet of size t can be fragmented into any set of fragments $(a_{0,} a_1, a_2, \ldots, a_q)$, where, for $1 \le i \le q$, a_i is the number of fragments of size τ_i and $a_0 \in \{0, 1,\}$ is the number of fragments of other sizes. In other words, all but one of the fragments of a packet are of a size that belongs to Q. We are obviously interested in the case $q = \|Q\| < T$.

For a given set Q a fragmentation process that follows these rules is called a *Q-quantized* fragmentation process, and an FR algorithm that uses Q-quantized fragmentation is called a *QFR algorithm*.

We shall confine ourselves to sets Q such that, for some τ and for all $1 \le i \le q$, $\tau_i = i \cdot \tau$ that is, all members of Q are multiples of τ. Such a set will be denoted $Q(\tau)$ and we shall refer to a $Q(\tau) - quantized$ fragmentation process and $Q(\tau) FR$ algorithms. Any value of t can be written as $t = \beta_t \cdot \tau + \gamma_t$, where $\beta_t \stackrel{\Delta}{=} \lfloor \frac{t}{\tau} \rfloor$ and $\gamma_t \stackrel{\Delta}{=} t \ mod\ \tau$. When fragmenting a t-packet, a $Q(\tau)$-quantized fragmentation process generates several packets of sizes $k\tau$ for $1 \le k \le q$ and exactly one packet of size $l\tau + \gamma_t$ for some $l \ge 0$ (obviously, if $\gamma_t = 0$ then $l = 0$ and no such packet is generated).

Given a set Q, any regular FR algorithm can be converted into a QFR one resulting in a considerable decrease in the number of operations. In particular, this is true for the algorithms presented in the previous subsections. However, the question is whether optimality is affected by the conversion. We show below that for a certain class of networks optimality is not affected by quantization.

Consider a network (V,E) in which $T_{ik} = b_{ik} \cdot T_{\min}$ for every $(i,k) \in E$ (where b_{ik} and $T_{\min}$ are integers). We refer to such a network as a $T_{\min} - quantized$ network. Note that, since $T_{\min}$ may equal 1, any network can be regarded as a 1-quantized network. The following results are meaningful for values of $T_{\min}$ larger than 1.

Theorem 4: Given a $T_{\min}$-quantized network with traffic inputs such that $r_i^t(j)=0$ for $t \neq kT_{\min}$ $(k=1,2,\cdots)$, there is a $Q(T_{\min})FR$ algorithm that is optimal i.e., that solves (the nonquantized and general) Problem FR.

Proof: Consider an optimal algorithm FR^*. We associate with FR^* a $Q(T_{\min})FR$ algorithm $QFR*$, that operates in the following way. Each time that FR^* fragments a packet, so does $QFR*$, and the number of fragments is equal in both algorithms. In fact, each fragment of $QFR*$ is associated with one of FR^*. The fragments generated by $QFR*$ are either a rounded up or rounded down (to the nearest multiple of $T_{\min}$) version of the fragments generated by FR^* where the rounding is performed according to a rule which we shall establish later. Observe that since maximum packet sizes T_{ik} are multiples of $T_{\min}$, it follows that a path traversed by a packet of size $K \cdot T_{\min} + \gamma$ $(1 \leq \gamma < T_{\min})$ can also be traversed by a packet of size $(K+1) \cdot T_{\min}$. This allows $QFR*$ to route each fragment along the same route taken by the corresponding fragment of FR^*.

Obviously, if $QFR*$ could present the same total flows on all links as FR^*, then its performance would equal that of FR^*. Due to the quantization requirement, this is impossible to do on a per-fragment basis. However, by the definition of the problem, any fragmentation-routing decision taken by any node is repeated infinitely many times at this node. The idea is thus to alternate between the rounded-up and rounded-down versions of every fragment, in such a way that the average flow generated would be equal to that of the corresponding FR^* fragments.

Consider a packet of size, say, $KT_{\min} + \gamma$, $\gamma < T_{\min}$, fragmented at a certain node by the FR^* algorithm, into m fragments of size $K_i T_{\min} + \gamma_i$ each, $\gamma_i < T_{\min}$, $1 \leq i \leq m$ (note that, although external packets entering the network are quantized, packets inside the network may have any length, due to previous fragmentations). The generation of the output flow is based on the relation

$$K_i \cdot T_{\min} + \gamma_i \;=\; \left[\frac{\gamma_i}{T_{\min}} \right] (K_i + 1)\, T_{\min} + \left[\frac{T_{\min} - \gamma_i}{T_{\min}} \right] (K_i)\, T_{\min} \tag{1}$$

which means that the desired output flow can be achieved by generating rounded up packets (of size $(K_i+1)\,T_{\min}$) at a relative frequency of $\dfrac{\gamma_i}{T_{\min}}$ and rounded down packets at a relative frequency of $\dfrac{T_{\min} - \gamma_i}{T_{\min}}$.

In the network running $QFR*$, the node in question will receive packets of size $KT_{\min}$ and $(K+1)\,T_{\min}$ (in certain relative frequencies) instead of packets of size $KT_{\min} + \gamma$. It has to generate packets of size $(K_i+1)\,T_{\min}$ and $K_i T_{\min}$ in the relative frequencies specified in equation (1). We now describe the rule by which this fragmentation is done.

Let $\tilde{K}$ be such that $\sum_{i=1}^{m} \gamma_i = \tilde{K}T_{\min} + \gamma$ (in other words, $\tilde{K} + \sum_{i=1}^{m} K_i = K$). For each γ, $\tilde{K}$, and m we construct a $0-1$ matrix M with $T_{\min}$ rows and m columns. Each of the first γ rows has exactly $\tilde{K}+1$ "1"-entries (and the rest are "0"). The remaining $T_{\min} - \gamma$ rows have $\tilde{K}$ "1"s per row. The matrix is arranged such that , for $1 \le i \le m$, the i-th column has exactly γ_i "1"s. Such a matrix can be constructed.

*QFR** uses the matrix M in the following way. Each row stands for a possible combination of m fragments, where a "1" in the i-th column means that the i-th fragment is rounded up (i.e., its size is $(K_i+1)\,T_{\min}$), and a "0" means that it is rounded down (i.e., size $K_i\,T_{\min}$). Every time that a rounded up packet arrives (compared to that which would have arrived when FR^* runs), *QFR** chooses according to order one of the first γ rows (this choice can also be done probabilistically and uniformly). Since there are $\tilde{K}+1$ "1"s in each of these rows, it is easy to verify that the required fragmentation is indeed possible. Likewise, when a rounded down version arrives, *QFR** chooses one of the remaining $T_{\min} - \gamma$ rows.

Assume now that rounded-up versions (of the discussed packet) arrive at a relative frequency of $\frac{\gamma}{T_{\min}}$ and rounded-down versions at a relative frequency of $\frac{T_{\min} - \gamma}{T_{\min}}$. Consider any i, $1 \le i \le m$. Suppose that the total number of "1"s in the first γ rows of the i-th column is x_i. Then, according to the definition of the matrix M, the total number of "1"s in the remaining $T_{\min} - \gamma$ rows of the i-th column is $\gamma_i - x_i$. The average number of "1"s chosen in the i-th column is thus:

$$\frac{\gamma}{T_{\min}} \cdot \frac{x_i}{\gamma} \; + \; \frac{T_{\min} - \gamma}{T_{\min}} \cdot \frac{\gamma_i - x_i}{T_{\min} - \gamma} \; = \; \frac{\gamma_i}{T_{\min}}$$

meaning that *QFR** is able to produce a rounded-up version of the i-th fragment with relative frequency of $\frac{\gamma_i}{T_{\min}}$ and a rounded-down version with relative frequency of $\frac{T_{\min} - \gamma_i}{T_{\min}}$ as required by equation (1).

The proof itself is inductive, on the fragmentation that a packet undergoes along its path to the destination. Consider, as before, a packet of size $KT_{\min} + \gamma$ that is fragmented at some node by the FR^* algorithm, into m fragments of size $K_i\,T_{\min} + \gamma_i$. Assume inductively that, due to the operation of *QFR**, instead of the packet of size $KT_{\min} + \gamma$ which undergoes fragmentation by FR^*, we now have either a packet of size $KT_{\min}$ or one of size $(K+1)\,T_{\min}$; furthermore, assume that the larger packet appears with relative frequency of $\frac{\gamma}{T_{\min}}$.

We have shown above that under such circumstances *QFR** is able to maintain, on a per node basis, the number of fragments and the total output flow compared to the operation of FR^* so that the inductive assumption holds for the resulting output flow, which is optimal.

To conclude the inductive proof it remains to check the initial condition, which corresponds to the first time that a packet is fragmented. Since external packets enter the network only in quantized sizes, we have that initially $\gamma = 0$, and thus the inductive assumption holds trivially for this case.

To summarize, we carry the inductive assumption along each generalized path created by FR^*. $QFR*$ fragments and routes packets along the same generalized paths, only that the new fragments appear, alternatively, in either rounded-up or rounded-down forms. The choice of quantized fragments is done in such a way that the total flow in all links is preserved. Thus, $QFR*$ exhibits the same performance as FR^*. Since FR^* is optimal, so is $QFR*$. □

By definition, general $Q(\tau) FR$ algorithms generate, in addition to fragments that are multiple of τ also some odd-sized fragments. From the above theorem and proof it follows that all fragments produced by an optimal $Q(T_{\min}) FR$ algorithm in the setting of Theorem 4 have quantized sizes $K \cdot T_{\min}$,without any odd-sized fragments.

Following the above result, in a $T_{\min}$-quantized network, an optimal $Q(T_{\min}) FR$ algorithm needs to consider only $O(\frac{T}{T_{\min}})$ values of t as opposed to $O(T)$ values in general FR algorithms. Thus the performance of the algorithms presented in the previous subsections will improve by a factor of $T_{\min}^2$ when a quantized approach is used. For example, in a network in which the bounds on packet lengths are $100, 200, 300, \cdots$ (i.e., $T_{\min} = 100$) and the maximal packet length is $T = 1000$, we shall have 100 phases per iteration instead of 10^6.

Theorem 4 deals with networks for which there is a limited number of "natural" packet sizes. One might hope that the above result could be extended to include also quantized networks but with no restriction on the input stream (i.e., on $r_i^t(j)$). More precisely, we could try to treat packets of sizes t that fall in the range $K \cdot T_{\min} < t \le (K+1) \cdot T_{\min}$ in the same fashion as packets of size $(K+1) \cdot T_{\min}$, since all these packets are subject to the same set of packet size restrictions. However, the following simple example shows that this is not the case. In Figure 9 we have: $T_{12} = T_{24} = T_{34} = 400$, $T_{13} = 200$, which corresponds to a quantized network with $T_{\min} = 200$. Suppose that the header size is $h = 100$ for all links, and define the

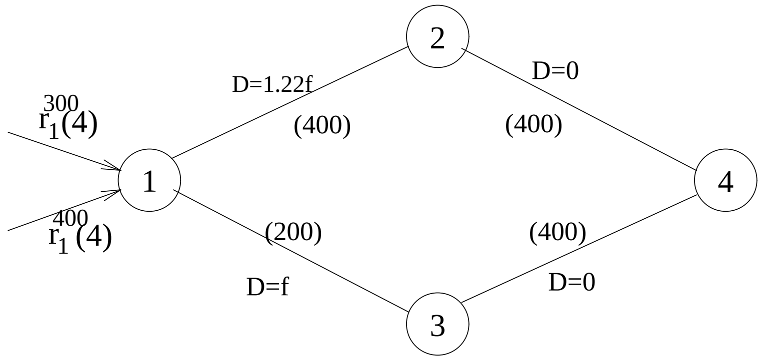

Figure 9: A quantized network with unquantized packets

following link cost functions: $D_{12} = 1.22 \cdot f_{12}$, $D_{13} = f_{13}$, $D_{24} = D_{34} = 0$. Suppose that $r_1^{300}(4) > 0$ and $r_1^{400}(4) > 0$ (note that there is an input flow that violates the assumption of Theorem 4). A quantized approach would attempt to treat 300-bit packets in the same way as 400-bit packets. However, it is easy to verify that the only optimal FR policy is to route 300-bit packets through links $(1,2)$, $(2,4)$, whereas 400-bit packets should be fragmented into pairs of 200-bit packets, which are routed through links $(1,3)$,$(3,4)$. This ill-effect is due to the penalty paid for the header which affects shorter packets more than the long ones.

One observation though comes to rescue. In many cases input packets are themselves the output of a fragmentation process, applied to the original messages (this is the foundation of packet switching). Typically, these original messages are considerably larger than the packets routed through the network. Assume then that we break messages into packets according to a $T_{\min}$-quantized process, i.e., in such a way that all but at most one packet out of each message have proper quantized lengths. Then, the requirement of Theorem 4 is satisfied for "almost" all packets. If the message lengths are large enough (in particular, if they are considerably larger than the maximal allowable packet length T), then the effect of the odd sized packets (at most one per message) on the total network performance becomes negligible.

Yet another quantized approach is possible in the following case. Consider a network for which every maximum packet size divides all maximum packet sizes of greater values. That is, if $T_{ik} < T_{mn}$ then there is some integer q such that $T_{mn} = q \cdot T_{ik}$. In other words, there is some finite sequence $M = \{t_1, t_2, \ldots, t_m, \cdots\}$ such that, for all m, $t_{m+1} = q_m \cdot t_m$ (where q_m are integers) and such that, for all i,k, we have $T_{ik} \in M$. We concentrate on a network with such a property, and for which the input flow is such that $r_i^t(j) = 0$ for $t \notin M$. For example, a network with maximum packet sizes of 5, 25, 50 and 150 belongs to this class. In [14] it is shown that any feasible flow configuration in such a network can be altered such that the new flow preserves the total amount of info-flow ($\hat{f}_{ik}(j)$) on every link, is composed only of packets of size $t \in M$, and does not increase the amount of link flow ($f_{ik}(j)$), i.e., reduces or maintains the number of fragments. Since the link cost function D_{ik} is an increasing function of $f_{ik}(j)$, we conclude that

Proposition 1: Given a set M and a network as described above, there is a *QFR* algorithm with $Q = M$ that is optimal i.e., that solves (the nonquantized and general) Problem FR. □

Thus, in such a network, the performance of the algorithms depend on the value of $|M|$rather than on that of *T*. Since $|M| \le |E|$ the algorithms are affected by the network size rather than by the packet sizes.

VII. SPECIAL CASES

A. Direct Cost for Fragmentation

So far, we considered fragmentation cost that incurred solely due to the increase in flow caused by the additional headers. Quite often other penalties may be associated with fragmentation that are not flow-related but rather are directly related to the total amount of fragments produced. For example, the additional processing at the fragmenting node and performing the reassembly at the destination. A possible way to incorporate such costs is to include an additional penalty function that grows with the total number of fragments in the network. One way to do that is to define for each destination *j* a *fragmentation cost function* F_j, whose argument is the total number of fragments or packets destined to *j* that are received at that node, i.e.:

$$F_j = F_j\left(\sum_k \sum_t \frac{1}{t}\cdot \hat{f}_{kj}^{\,t}(j)\right)$$

We require each F_j to be a convex, nonnegative continuous increasing function, with continuous first and second derivatives. F_j can account for the additional overhead at the end-to-end level due to increased fragmentation, e.g. the increased load on the reassembly process.

The total network cost function is then defined as

$$D_T = \sum_{i,k} D_{ik}(f_{ik}) + \sum_j F_j\left(\sum_k \sum_t \frac{1}{t}\cdot \hat{f}_{kj}^{\,t}(j)\right)$$

It is easy to adapt all our previous results to this new environment. For example, the only change in the optimality conditions is in (OC3), which becomes:

$$\lambda_j^t(j) = \frac{1}{t} F_j'$$

where F_j' is the first derivative of F_j. Note that F_j' is divided by *t* since F_j is a cost per packet, whereas the optimality conditions deal with flows of bits. The intuitive meaning of this change is the existence of a "bias" cost for fragmenting a packet, which should be taken into account whenever fragmentation is considered. This is accomplished in the following way. As the fragmentation of packets destined to *j* grows, so does the value of $F'^{\,j}$ (since *j* receives more fragments). This results in increasing the values of $\lambda_i^t(j)$ throughout the network, in a way that is inversely proportional to the value of *t*, thus causing small fragments to be costlier. (OC2) then forces fewer fragments to be produced.

B. Nodes Incapable of Fragmentation

A possible architecture of a (inter)network can be one in which some nodes do not perform fragmentation. This is easily handled by imposing in the optimality conditions the constraints $g_i^{t,\tau}(j) = 0$ for each node *i* that cannot perform fragmentation.

As a special case of the above, consider a network in which no node is capable of fragmentation. It turns out that if header-sizes are not identical for all links, then it is still necessary to distinguish the different packet sizes. In such networks optimal performance cannot be reached through classical routing algorithms but through optimal FR algorithms (using the above adaptation).

C. Fragmentation-Routing with Packet-Hop Optimization

In many circumstances it may be desired to minimize the number of packets rather than delay. In such circumstances each fragment adds a fixed cost for each link it traverses, that is, each link (i,k) is assigned a weight β_{ik}, and a packet traversing this link adds β_{ik} to the cost function.

The adaptation of our results to this case is immediate, through the definition of the following link cost function:

$$D_{ik}(\hat{f}_{ik}^1, \hat{f}_{ik}^2, \cdots, \hat{f}_{ik}^T) = \beta_{ik} \cdot \sum_t \frac{\hat{f}_{ik}^t}{t}$$

Note that this link cost function can still be interpreted as $D_{ik} = D_{ik}(f_{ik})$, by defining $\alpha_{ik}^t = \frac{1}{t}$ (instead of the definition of section II), since then:

$$D_{ik}(\hat{f}_{ik}^1, \hat{f}_{ik}^2, \ldots, \hat{f}_{ik}^T) = \beta_{ik} \cdot \sum_t \frac{\hat{f}_{ik}^t}{t} = \beta_{ik} \cdot \sum_t \alpha_{ik}^t \cdot \hat{f}_{ik}^t = \beta_{ik} \cdot \sum_t f_{ik}^t = \beta_{ik} \cdot f_{ik} \triangleq D_{ik}(f_{ik}) .$$

The optimality conditions remain unchanged except that (OC1) now reads:

$$\lambda_i^t(j) \begin{cases} = \lambda_k^t(j) + \dfrac{\beta_{ik}}{t} & \hat{f}_{ik}^{*t}(j) > 0 \\ \leq \lambda_k^t(j) + \dfrac{\beta_{ik}}{t} & \hat{f}_{ik}^{*t}(j) = 0 \end{cases}$$

D. Virtual Circuits and Single Path Routing

In some networking configurations, routing is performed through virtual circuits. These circuits are often preestablished in a way that is independent of the packet generation and fragmentation processes, meaning that routing and fragmentation cannot be coupled. However, the example shown in Section I shows that even with preestablished circuits, the places in which packets should be fragmented and their sizes should be carefully calculated.

Our results are general enough to be adapted to this case too. For example, in (OC1) we could use $\alpha_{ik}^{t} \stackrel{\Delta}{=} \infty$ for those links (i,k) which are not part of the circuit. The solution of the problem is, obviously, simpler than for the general case. For example, if the performance measure is that of packet-hops then it is easy to see that the optimality conditions imply a simple integer dynamic programming problem characterized by the following optimality equation:

$$\Lambda_{v_i}^{t} = \min(1 + \Lambda_{v_{i+1}}^{t}, \min_{\tau \le \frac{t}{2}} (\Lambda_{v_i}^{\tau} + \Lambda_{v_i}^{t-\tau}))$$

where $(v_1, v_2, \cdots, v_m)$ is the circuit and $\Lambda_{v_i}^{t}$ is the (least) cost of sending a packet of size t from v_i to v_m. Note that the above dynamic program is solved in $O(T^2 |V|)$ operations, and this complexity can be reduced using the quantized approaches mentioned before.

Suppose now that one can incorporate fragmentation considerations when establishing the virtual circuit, i.e., the choice of the circuit should take into account the effects of the resulting fragmentation. This, in fact, is an instance of a more general problem, namely: the FR problem, but under the restriction that routing between each source-destination pair should be done through a single path (i.e., no bifurcation of flow). This latter problem is known to be very complex in the basic context of routing even with no fragmentation. Obviously, this becomes more complex when fragmentation is added to the picture, However, the tools devised for the general (bifurcated) fragmentation-and-routing problem could serve in devising heuristics for the single-path version, just as is the case with regular routing (e.g., routing in the Codex network[15]).

VIII. CONCLUSION

In this paper we addressed the fragmentation and routing problem in computer networks. We have shown that the most general fragmentation process can be modeled via simple and tractable recursive machines. This observation enabled us to build a comprehensive flows model, on which convexity properties could be established. With that at hand, we were able to obtain optimality conditions that have a simple yet insightful intuitive appeal. In particular, using the optimality conditions we were able to show a strong equivalence between the two recursive fragmentation machines, in the sense that the simpler conditions of the TFR machine apply equally well to the more restricted TFS machine and vice-versa. This means that the fragmentation process in each node can be viewed as a "black box", while the network is free to use the simpler set of optimality conditions in any case. Although we concentrated on the classical performance measure of total packet delay, we showed how our approach can also accommodate other common performance measures.

The derived optimality conditions serve as the foundation for various optimal algorithms. In particular, they indicate how the concept of MFDL paths should be extended in order to be incorporated into our environment.

The general fragmentation-and-routing problem has a natural dependency on the maximal packet length in the network, with a direct impact on the complexity of its solution. However, we showed that if packet sizes in the network are restricted to certain limited sets, then the optimal solution will not need unnatural packet lengths. This results in a convenient and practical reduction of the problem size. In particular, there are cases in which the problem size can be freed from any dependence on packet sizes.

The solution we propose leaves a few questions open for further research. One example is the interference of fragmentation with end-to-end data transfer. Most networks use a variation of window-based protocols on an end-to-end basis. With fragmentation some fragments may arrive while others get lost and the question rises who should perform the retransmission. The easy way out (which is taken by most network designers) is that loss of a fragment constitutes a loss of the entire packet which must be retransmitted from the source. In other words, fragmentation is considered a process performed at a level lower than the Transport. Whether this is an appropriate choice is left open. Another problem is that of reassembly. While we consider only fragmentation at the destination it is unclear whether partial reassembly en route (if possible) is useful.

The general optimality requirements that we derived lead to fairly complex algorithms that may not be practical in all distributed cases. In such cases, a more greedy approach would typically be used. Devising such algorithms and particularly assessing their performance is also a subject for further research.

APPENDIX

A. Proof of Theorem 1:

Theorem 1: A necessary and sufficient condition for a set $fr^* = \{\hat{f}_{ik}^{t*}(j)\} \cup \{g_i^{t,\tau *}(j)\}$ to solve Problem FR (with the TFR machine) is the existence of a set of nonnegative numbers (Lagrange multipliers) $\{\lambda_i^t(j) \mid \ 1 \le i \le n,\ \ 1 \le j \le n,\ \ 1 \le t \le T\}$, such that (OC1)-(OC3) are fulfilled.

Proof: We construct a Lagrangian in the following way:

$$L_1 \triangleq D_T$$

$$L_2 \triangleq \sum_{i,j \ne i,t} \lambda_i^t(j) \cdot [\sum_k \hat{f}_{ki}^t(j) + r_i^t(j) - \sum_{\tau \le \frac{t}{2}} g_i^{t,\tau}(j) + \sum_{2t \le \tau \le T} \frac{t}{\tau} g_i^{\tau,\tau-t}(j) + \sum_{2t \le \tau \le T} \frac{t}{\tau} g_i^{\tau,t}(j)]$$

$$L_3 \triangleq \sum_{i,j \ne i,k \in N_i,t} \eta_{ik}^t(j) \cdot \hat{f}_{ik}^t(j)$$

$$L_4 \triangleq \sum_{i,j \ne i,t,\tau \le \frac{t}{2}} \eta_i^{t,\tau}(j) \cdot g_i^{t,\tau}(j)$$

$$L \triangleq L_1 + L_2 + L_3 + L_4$$

Taking derivatives we get:

$$\forall i,j \ne i,k \in N_i,t: \quad \left. \frac{\partial L}{\partial \hat{f}_{ik}^t(j)} \right|_{fr=fr^*} = \lambda_k^t(j) - \lambda_i^t(j) + \alpha_{ik}^t \cdot D'_{ik} + \eta_{ik}^t(j) \tag{A1}$$

$$\forall i,j \ne i,t,\tau \le \frac{t}{2}: \quad \left. \frac{\partial L}{\partial g_i^{t,\tau}(j)} \right|_{fr=fr^*} = -\lambda_i^t(j) + \frac{t-\tau}{t} \cdot \lambda_i^{t-\tau}(j) + \frac{\tau}{t} \cdot \lambda_i^\tau(j) + \eta_i^{t,\tau}(j) \tag{A2}$$

Since fr^* is optimal we have that, $\forall i,j \ne i,k \in N_i,t$:

$$\eta_{ik}^t(j) \cdot f_{ik}^{t*}(j) = 0, \tag{A3}$$

$$\eta_{ik}^t(j) \le 0, \tag{A4}$$

and also that, $\forall i,j \ne i,t,\tau \le \frac{t}{2}$:

$$\eta_i^{t,\tau}(j) \cdot g_i^{t,\tau *}(j) = 0, \tag{A5}$$

$$\eta_i^{t,\tau}(j) \le 0. \qquad \text{A6}$$

By defining $\lambda_j^t(j) \overset{\Delta}{=} 0$ for all j, t, as in (OC3), we have that A1, A3 and A4 are equivalent to (OC1); A2, A5 and A6 are equivalent to (OC2). Thus, (OC1)-(OC3) are necessary conditions for optimality. Since D_T is a convex function over the convex set $\boldsymbol{F_c}$, it follows that (OC1)-(OC3) are also sufficient conditions for optimality. □

B. Proof of Lemma 3:

Lemma 3: (OC1)-(OC3) are necessary and sufficient conditions for optimality also for the TFS machine, i.e.: A necessary and sufficient condition for a vector $fr^* = \{\hat{f}_{ik}^{t^*}(j)\} \cup \{g_i^{t,\tau^*}(j)\}$, that is feasible for the TFS machine, to solve Problem FR (with that machine) is the existence of a set $\{\lambda_i^t(j) | \ 1 \le i \le n,\ 1 \le j \le n,\ 1 \le t \le T\} \cup \{\bar{\lambda}_i^t(j) | \ 1 \le i \le n,\ 1 \le j \le n,\ 1 \le t \le T\}$ for which (OC1)-(OC3) hold.

Proof: In one direction, if (OC1)-(OC3) hold, then so do (OC1')-(OC4'), by choosing $\bar{\lambda}_i^t(j) = \lambda_i^t(j)$ for all i, t, j.

In the other direction, We are given an optimal flow fr^*, i.e., one that satisfies conditions (OC1')-(OC4'), and a set of numbers $\{\lambda_i^t(j)\ ,\ \bar{\lambda}_i^t(j)\}$ that satisfy the optimality conditions (OC1')- (OC4'). We now show that there is a (possibly new) set that complies also with (OC1)-(OC3).

We are given a network (V,E), that uses exclusively the TFS scheme. Given a vector fr^* that is optimal for that network, we show that there is a set $\{\lambda_i^t(j), \bar{\lambda}_i^t(j)\}$ that fulfills (OC1')-(OC4'), for which it holds that, for all i, t, j, $\lambda_i^t(j) = \bar{\lambda}_i^t(j)$. Note that such a set indeed fulfills also (OC1)-(OC3).

If, for all i, j, t, we have $\lambda_i^t(j) = \bar{\lambda}_i^t(j)$ then we are done. Suppose then that there exist *i*, *j*, and *t* such that $\lambda_i^t(j) \ne \bar{\lambda}_i^t(j)$. We need to show that another set of λs can be constructed such that $\lambda_i^t(j) = \bar{\lambda}_i^t(j)$ for all i, j, t and for which (OC1')-(OC4') still hold.

Let *i*, *j* and *t* be such that $\lambda_i^t(j) \ne \bar{\lambda}_i^t(j)$ and, for these *i* and *j*, let *t* be the smallest size for which this holds.

Since $\lambda_i^t(j) \ne \bar{\lambda}_i^t(j)$ it follows from (OC3') that $\bar{\lambda}_i^t(j) > \lambda_i^t(j)$ and that $x_i^{*t}(j) = 0$. From (OC4') follows that $i \ne j$. We break the proof into subcases and prove each of them separately.

Case 1:

$$\bar{\lambda}_i^t(j) \le \min_\tau \{ \frac{t-\tau}{t} \lambda_i^{t-\tau}(j) + \frac{\tau}{t} \bar{\lambda}_i^\tau(j) \} \qquad (2)$$

Thus, $\forall \tau \le t/2 \quad \lambda_i^t(j) < \frac{t-\tau}{t} \lambda_i^{\tau-t}(j) + \frac{\tau}{t} \bar{\lambda}_i^\tau(j)$, and from (OC2') it follows that $g_i^{*t,\tau}(j) \equiv 0$ which, along with the fact that $x_i^{*t}(j) = 0$ implies (see condition C4' and Figure 8) that $\hat{f}_{ki}^{*t}(j) = 0$ for all *k* and that $g_i^{*\tau,\tau-t}(j)=0$ for all $t<\tau\leq 2t$.

We show that the original set of Lagrange multipliers with $\lambda_i^t(j)$ replaced by $\Lambda_i^t(j)=\bar{\lambda}_i^t(j)$ still satisfies conditions (OC1')-(OC4').

To check for validity of condition (OC1') we must check the case where $\Lambda_i^t(j)$ appears on the right hand-side of the relation, i.e., for some neighbor k of i. For this neighbor, condition (OC1') (on the former set of numbers) implies

$$\bar{\lambda}_k^t(j) \leq \lambda_i^t(j) + \alpha_{ki}^t D_{ki}' < \Lambda_i^t(j) + \alpha_{ki}^t D_{ki}'$$

but since $\hat{f}_{ki}^{*t}(j)=0$ only inequality is needed, and therefore condition (OC1') holds for the new set of numbers.

To check the validity of (OC2') we must consider the cases where $\Lambda_i^t(j)$ appears either on the left or the right hand-side of the relation. Clearly, from the definition of $\Lambda_i^t(j)$ and assumption (2) we have

$$\Lambda_i^t(j) \leq \frac{t-\tau}{t}\lambda_i^{\tau-t}(j) + \frac{\tau}{t}\bar{\lambda}_i^\tau(j)$$

so that (OC2') is satisfied when $\Lambda_i^t(j)$ appears on the left hand-side. Consider now the case where $\Lambda_i^t(j)$ appears on the right hand-side. Rewriting (OC2') and using (OC3') we get (for any θ such that $1\leq\theta\leq t$):

$$\lambda_i^{t+\theta}(j)\leq\frac{t}{t+\theta}\lambda_i^t(j)+\frac{\theta}{t+\theta}\bar{\lambda}_i^\theta(j) < \frac{t}{t+\theta}\Lambda_i^t(j)+\frac{\theta}{t+\theta}\bar{\lambda}_i^\theta(j) .$$

Since $g_i^{*\tau,\tau-t}(j)=0$ for all $t<\tau\leq 2t$, it follows that $g_i^{*t+\theta,\theta}(j)=0$. Thus, only inequality is needed for (OC2') to be satisfied and therefore the condition is satisfied.

We conclude that whenever Case 1 occurs, we can change the set of numbers so that the conditions are not violated.

Case 2:

$$\bar{\lambda}_i^t(j)>\min_\tau\{\frac{t-\tau}{t}\lambda_i^{t-\tau}(j) + \frac{\tau}{t}\bar{\lambda}_i^\tau(j)\} \tag{3}$$

Denote by τ' that value of τ that minimizes the right hand side of inequality (3).

Case 2.1: Assume there exists a $\tau\geq 2t$ for which $g_i^{*\tau,t}(j)>0$.

The above assumption means that, for i, j, τ and t, (OC2') must be satisfied with equality thus:

$$\begin{aligned}\lambda_i^\tau(j) &= \frac{\tau-t}{\tau}\lambda_i^{\tau-t}(j) + \frac{t}{\tau}\bar{\lambda}_i^t(j) > \frac{\tau-t}{\tau}\lambda_i^{\tau-t}(j) + \frac{t}{\tau}(\frac{t-\tau'}{t}\lambda_i^{t-\tau'}(j) + \frac{\tau'}{t}\bar{\lambda}_i^{\tau'}(j)) = \\ &= \frac{\tau-t}{\tau}\lambda_i^{\tau-t}(j) + \frac{t-\tau'}{\tau}\lambda_i^{t-\tau'}(j) + \frac{\tau'}{\tau}\bar{\lambda}_i^{\tau'}(j)\end{aligned} \tag{4}$$

where we made use of equation (3). From the above and from (OC2') we get

$$\frac{\tau-t}{\tau}\lambda_i^{\tau-t}(j) + \frac{t-\tau'}{\tau}\lambda_i^{t-\tau'}(j) + \frac{\tau'}{\tau}\bar{\lambda}_i^{\tau'}(j) < \lambda_i^{\tau}(j) \le \frac{\tau-\tau'}{\tau}\lambda_i^{\tau-\tau'} + \frac{\tau'}{\tau}\bar{\lambda}_i^{\tau'}$$

from which we conclude that

$$\lambda_i^{\tau-\tau'}(j) > \frac{\tau-t}{\tau-\tau'}\lambda_i^{\tau-t}(j) + \frac{t-\tau'}{\tau-\tau'}\lambda_i^{t-\tau'}(j) \tag{5}$$

Since, for i and j, t is the smallest value for which $\lambda_i^t(j) \neq \bar{\lambda}_i^t(j)$, it follows that $\lambda_i^{t-\tau'}(j) = \bar{\lambda}_i^{t-\tau'}(j)$ and substituting this into (5) yields

$$\lambda_i^{\tau-\tau'}(j) > \frac{\tau-t}{\tau-\tau'}\lambda_i^{\tau-t}(j) + \frac{t-\tau'}{\tau-\tau'}\bar{\lambda}_i^{t-\tau'}(j)$$

Since $\tau-t > t-\tau'$ this equation contradicts (OC2') for $i, j, \tau-\tau'$, and $t-\tau'$. Thus, this case cannot exist at all.

Case 2.2: For every $\tau \ge 2t$ holds: $g_i^{*\tau,t}(j) = 0$.

Since $x_i^{*t}(j) = 0$ it follows that $\hat{f}_{ik}^{*}(j) = 0$ for all k. We define $\bar{\Lambda}_i^t(j) = \Lambda_i^t(j) \triangleq \min_{\tau}\{\frac{t-\tau}{t}\lambda_i^{t-\tau}(j) + \frac{\tau}{t}\bar{\lambda}_i^{\tau}(j)\}$, and show that the original set of λs with $\bar{\lambda}_i^t(j)$ and $\lambda_i^t(j)$ replaced (correspondingly) by $\bar{\Lambda}_i^t(j)$ and $\Lambda_i^t(j)$ still satisfies the conditions (OC1')-(OC4').

Condition (OC1') is clearly satisfied for $\bar{\Lambda}_i^t(j)$ since $\hat{f}_{ik}^{*}(j) = 0$ and $\bar{\Lambda}_i^t(j) < \bar{\lambda}_i^t(j)$. If $\Lambda_i^t(j) \neq \lambda_i^t(j)$ then $g_i^{*t,\tau} \equiv 0$, and since $x_i^t(j) = 0$ it follows that $\hat{f}_{ki}^{*t}(j) \equiv 0$; since $\Lambda_i^t(j) \ge \lambda_i^t(j)$, it follows that (OC1') still holds.

It follows immediately from the definition of $\Lambda_i^t(j)$ that replacing $\lambda_i^t(j)$ with $\Lambda_i^t(j)$ does not violate (OC2'). To show that (OC2') for $\bar{\Lambda}_i^t(j)$, assume there exists some $\tau \ge 2t$ for which

$$\lambda_i^{\tau}(j) > \frac{\tau-t}{\tau}\lambda_i^{t-\tau}(j) + \frac{t}{\tau}\bar{\Lambda}_i^t(j) . \tag{6}$$

Replacing $\bar{\Lambda}_i^t(j)$ with $\min_{\tau}\{\frac{t-\tau}{t}\lambda_i^{t-\tau}(j) + \frac{\tau}{t}\bar{\lambda}_i^{\tau}(j)\}$ yields

$$\lambda_i^{\tau}(j) > \frac{\tau-t}{\tau}\lambda_i^{\tau-t}(j) + \frac{t-\tau'}{\tau}\lambda_i^{t-\tau'}(j) + \frac{\tau'}{\tau}\lambda_i^{\tau'}(j)$$

which leads to an inequality like (4) and to the same contradiction.

Condition (OC3') holds trivially.

□

C. Proof of Theorem 3:

Theorem 3: Given a network with any combination of TFR and TFS machines, a necessary and sufficient condition for a vector $fr^* = \{\hat{f}_{ik}^{t*}(j)\} \cup \{g_i^{t,\tau*}(j)\}$ (for which, for each i, $\{g_i^{t,\tau*}(j)\}$ is feasible for the fragmentation machine implemented at i), to solve Problem FR is the existence of a set of nonnegative numbers (Lagrange multipliers) $\{\lambda_i^t(j) \mid 1 \le i \le n,\ 1 \le j \le n,\ 1 \le t \le T\}$ that fulfills (OC1)-(OC3).

Proof: Consider (OC1)-(OC3), which were shown to be optimality conditions for both machines. The dependence between the conditions of different nodes is manifested only in (OC1) and only for neighbors i,k for which, for some t, $\hat{f}_{ik}^{*t}(j) > 0$. It is easy to verify that, if $\hat{f}_{ik}^{*t}(j) > 0$ for some k, then $\lambda_i^t(j)$ (for either machine) is in fact the t-MFDL between i and j. Due to convexity, t-minimal generalized paths are the same for any combination of TFR and TFS machines. This means that the values of the t-MFDL (for all t and between any i and destination j) are independent of the type of machine used at each node. The claim follows. □

REFERENCES

1. V.G. Cerf and R.E. Kahn, ‘‘A Protocol for Packet Network Interconnection,’’ *IEEE Trans. on Communications* **COM-22**(5) pp. 637-648 (May 1974).

2. C. Partridge, *Innovations in internetworking*, Artech House, Norwood, MA (1988).

3. J.F. Shoch, ‘‘Packet Fragmentation in Internetwork Protocols,’’ *Computer Networks* **3**(1) p. 3 (February 1979).

4. C.J. Bennett, ‘‘The Overheads of Transnetwork Fragmentation,’’ *Computer Networks* **6**(1) pp. 21-36 (February 1982).

5. M.A. Bonuccelli, ‘‘Minimum Fragmentation Internetwork Routing,’’ pp. 289-294 in *Proceedings of INFOCOM’91*, Bal Harbor, Florida (April 1991).

6. R. Rom and M. Tal, ‘‘Optimal Fragmentation and Routing in Computer Networks,’’ EE Publication 495, Faculty of Electrical Engineering, Technion, Haifa, Israel (March 1984).

7. J.B. Postel, ‘‘Internetwork Protocol Approaches,’’ *IEEE Trans. on Communications* **COM-28**(4) pp. 604-611 (April 1980).

8. J.B. Postel (ed.), ‘‘Internet Protocol,’’ RFC 791, Defense Advanced Research Projects Agency, Information Processing Techniques Office (September 1981).

9. A. Segall, ‘‘Optimal Distributed Routing for Virtual Line-Switched Data Networks,’’ *IEEE Trans. on Communications* **COM-27**(1) pp. 201-209 (January 1979).

10. R.G. Gallager, ‘‘A Minimum Delay Routing Algorithm Using Distributed Computation,’’ *IEEE Trans. on Communications* **COM-25**(1) pp. 73-85 (January 1977).

11. L. Fratta, M. Gerla, and L. Kleinrock, ‘‘The Flow Deviation Method: An Approach to Store and Forward Communication Network Design,’’ *Networks* **3**(2) pp. 97-133 (1973).

12. N. Deo and C.Y. Pang, ‘‘Shortest Path Algorithms: Taxonomy and Annotation,’’ *Networks* **14** pp. 275-323 (1984).

13. A. Segall, ‘‘The Modeling of Adaptive Routing in Data-Communication Networks,’’ *IEEE Trans. on Communications* **COM-25**(1) pp. 85-95 (January 1977).

14. J. Naor, A. Orda, and R. Rom, *A Combinatorial Approach to Packet Fragmentation*. 1993.

15. D. Bertsekas and R. Gallager, *Data Networks*, Prentice Hall, Inc., New-Jersey (1987).