Università di Pisa

Dipartimento di Informatica

Technical Report : TR-98-06

Routing with Minimum Fragmentation Cost

GalloGiorgio ; Scaparra Maria Paola

June 26, 1998

ADDR: Corso Italia 40,56125 Pisa,Italy. TEL: +39-50-887111. FAX: +39-50-887226

Routing With Minimum Fragmentation Cost

Giorgio Gallo   Maria Paola Scaparra

June 26, 1998

Abstract

One of the greatest difficulties involved in communication networks project is the need to interconnect different networks to form a unique internetwork. A major headache is the differing maximum packet sizes allowed by each network, which causes oversize packets to be broken up into appropriately sized fragments. Unfortunately, the fragmentation process implies additional costs, tightly bounded up with routing and fragmentation strategies in use. The approach commonly used in IP protocols provides two separated functions which operate in a completely indipendent way. This adversely affects network performance measures, both quantitatively (throughput) and qualitatively (average packet delay), whereas some form of cooperation between the two functions could result in a reduction of this efficiency loss. This paper presents a pseudo-polynomial algorithm for solving the combined fragmentation and routing problem. The method of solution is based on the concept of bicriterion shortest path problem, where two distinct objective functions must be minimized: the path length (in accordance with many classic routing algorithm) and the number of fragments received by the destination host. Besides, the paper deals with the problem of the effective fragments-hops minimization. An algorithm aimed at this purpose is provided together with the analysis of the possibility to reduce its running time under appropriate considerations on the generated fragment sizes.

Index Terms

Networks, communication system routing.

1 Introduction

The packet fragmentation problem arises from the need to interconnect computer networks, which differ in the packet-size they can support. The internet model requires a gateway to be able to fragment packets as necessary to match the MPS (*Maximum Packet Size*) of the network to which they are being forwarded. Reassembly of fragmented packets is generally left to the destination host.

The approach applied in existing networks to solve the fragmentation and routing problem is quite simple: a routing algorithm, generally distributed, operates independently of fragmentation requirements. It assigns a length to each internet link, computes the shortest path to a given destination and exchange link lengths and routing information in order to synchronize the distributed database and avoid routing loops. Once the shortest path has been established, a packet will undergo fragmentation only if it must traverse a net that limits packets to a smaller size than its own [1]. Fragmentation schemes usually conform to a greedy rule. The most common ones are the *maximal scheme* and the *balanced scheme*.

The maximal scheme, given an oversize packet of size X, produces:

1. $\lfloor X/MPS \rfloor$ fragments of size MPS
2. One fragment of size $X \bmod MPS$ if $X \bmod MPS > 0$

This greedy scheme generates as many pieces as possible of the maximum size allowed by the network being crossed and, possibly, a small remainder piece.

The other strategy, *balanced* fragmentation, consists of keeping each fragment size as close to the mean as possible. It generates again $\lceil X/MPS \rceil$ fragments, but this time all the fragments differ in size by one unit at most. A possible scheme can be as follows.
Let $y = \lceil X/MPS \rceil$ and $x = \lfloor X/y \rfloor$. Then $X = yx + k$, where $k \geq 0$ is an integer. The scheme produces:

1. $y - k$ fragments of size x

2. k fragments of size $x + 1$

However, there is no guarantee that the greedy schemes, used only when necessary, and independently of routing decisions, are the best ones. This approach to fragmentation may lead to an inefficient use of network resources and to excessively high costs. This additional cost concerns both the final receiver, due to reassembly process and buffer menagement, and the intermediate gateways, because of the additional header processing and fragments bufferization. Further, bandwidth usage by header replication must be included in fragmentation costs [2].

The problem of minimizing these costs is a quite complex one because of many factors interacting and affecting one other. We show what these factors are and how they are related through an example. Consider the network of Fig. 1 in which the links are labeled according to the maximum packet size. Suppose a packet of size 1200 bytes must be forwarded by node 1 to node 7. A traditional routing protocol using the number of hops as path length measure, would choose the shortest path {1, 6, 7}. A greedy fragmentation scheme, for istance the maximal one, would produce along that path 12 fragments of size 100 on the first link, which would not undergo further fragmentation on the second link, resulting in a total of 12 fragments in the destination host. However, the fragmentation costs, in terms of fragments generated, are lower along the other paths.

A more precise measure of the fragmentation costs, including also the costs in intermediate nodes and the bandwidth usage, should be the effective number of hops passed through by the totality of fragments of a single packet. According to this metric, the cost of path {1, 6, 7} is 24, once again superior to the 14 and 15 fragments-hops obtained along the paths {1, 2, 4, 7} and {1, 3, 5, 7}, respectively. At first sight, one could impute the higher cost of path {1, 6, 7} to its bottleneck (i.e. the minimum MPS along the path). However, when greedy fragmentations scheme are used, there's no guarantee that the choice of the path with maximum bottleneck between two nodes will generate the minimum number of fragments. In the above example, path {1, 2, 4, 7}, having bottleneck 200, produces less fragments in node 7 than the path $\{1, 3, 5, 7\}$ with bottleneck 300. This is due to the fact that the number of fragments depends on the initial packet size and on the subpath followed till the bottleneck edge. The same observation still holds when the greedy balanced fragmentation scheme is used, even though in the above example the path $\{1, 3, 5, 7\}$ with the maximum bottleneck results also the best path with respect to the total number of fragments-hops. In fact, it produces 3 fragments of size 400 on the first link, which don't undergo further fragmentation on the second link and are lastly devided in 6 fragments of size 200 on the third link, for a total amount of 12 fragments-hops, against the 14 and 24 obtained along the paths $\{1, 2, 4, 7\}$ and $\{1, 6, 7\}$.

So far, we assumed that the length of a path is measured in terms of the number of hops; however many other metrics are also possible. For example, each link could be labeled with the geographic distance between the two nodes it connects, or by a function of its bandwidth, average traffic, measured delay, communication cost or a combination of those criteria. By changing the weighting function associated to each arc, the fragmentation costs would also change consequently. Suppose, for istance, that in the above example, the graph is weighted, with weights (number in brackets in Fig. 1) representing geographical distances (the figure is not drawn to scale). The fragmentation cost may thus be given by the sum of the distances covered by each fragment generated along the path. In this particular case and with the maximal scheme, we would have a cost of 96 along the path $\{1, 6, 7\}$, 110 for $\{1, 2, 4, 7\}$ and 100 for $\{1, 3, 5, 7\}$. With this metric, the path $\{1, 6, 7\}$ which so far was the worst one, gives the minimum cost, thus showing how the choice of a metric rather than another may deeply affect the "goodness" of a path with respect to the fragmentation cost.

We are now able to point out the factors involved in the analysis of the fragmentation cost minimization, namely:

- the fragmentation schemes adopted;
- the exact point along a path where fragmentation is performed;
- a proper choise of the objective function;

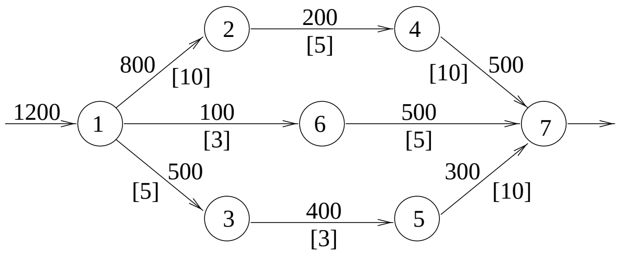

Figure 1: Example of routing and fragmentation.

- the cooperation between fragmentation and routing functions.

The interrelation between fragmentation and routing has been long studied in literature and several approaches proposed. They differ for both the objective functions they intend to minimize and the solution algorithms consequentely derived.

In the next section, we briefly present two former attempts to solve the problem. Then, we propose two different objective functions and give an exact formulation of the problem for each of them. Section 3.1 presents an algorithm for computing the best path according to the criterion which multiplies the length of the path and the number of fragment produced along it, and discusses some implementation issues. In Section 3.2, we study the problem of minimizing the effective fragments-hops and propose an algorithm for its solution. Considerations about further improvements terminate the paper.

2 Former Approaches

Earlier attempts at solving the fragmentation problems were carried out by Bonuccelli, who proposed a solution approach based on graph theory, and by Orda and Rom, who applied techniques borrowed from control theory to a bifurcated routing environment.

In the model proposed by Orda and Rom [3] the cost function denotes the total cost (delay) per unit time of all traffic on the internet and, therefore, is given by the sum of the delay on each link. The total cost in unit time is therefore:

$$C = \sum_{(i,k)\in E} C_{ik}(f_{ik})$$

where:

(i,k) is a graph link representing a generic individual network in the internetwork environment;

E is the set of all the individual networks in the internet;

f_{ik} is the flow (traffic level) on link (i,k);

C_{ik} is the total delay per unit time of all traffic passing through network (i,k).

The partial costs C_{ik} are expressed as increasing convex functions of the total flow f_{ik} on network (i,k) (i.e. the flow generated by the information-data with the appropriate header). Hence, in this model, the price of fragmentation is due to the additional headers, which increase the link-flow and, consequently, the total delay. Since in general, other penalties may be associated with fragmentation that are not flow-related but rather are directly related to the total amount of fragments produced (e.g., performing the reassembly at the destination and the additional processing at the fragmenting network), Orda and Rom provided a way to incorporate such costs in their model by means of an additional penalty function that grows with the total number of fragments in the network. They associate to each destination j a convex, nonnegative continuous increasing *fragmentation cost function* F_j, whose argument is the the total number of fragments or packets destined to j that are received at that node, i.e.:

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

where $\hat{f}^t_{kj}(j)$ denotes the amount of info-flow (i.e., the flow generated by the information-data without the headers) of t-bits packets on link (k,j) destined to node j.

The total network cost function is then defined as:

$$C = \sum_{i,k} C_{ik}(f_{ik}) + \sum_j F_j \left(\sum_k \sum_t 1/t \cdot \hat{f}^t_{kj}(j) \right)$$

In their analysis, they work out a tractable flow model from which deriving necessary and sufficient optimality conditions for the fragmentation problem and use such conditions as a base line for devising several optimal algorithms, both centralized and distributed. A major drawback of the method is that the general optimality requirements lead to fairly too complex algorithms due to the fact that they consider every possible way of fragmenting a packet, not only when it's strictly necessary but also when it is "worthwhile", i.e. when the sum of the incremental costs of handling the resulting fragments is not more expensive than sending the packet as is. The need to consider each value of t (each possible fragment size) makes the problem complexity increase considerably.

A quite different approach was proposed by Bonuccelli [4]. In that case, processing cost and extra delay due to fragmentation are considered proportional to the number of fragments received by the destination host, since, under an internetwork fragmentation strategy, the gateways have just to fragment oversize incoming packets and all the reassembly job is performed by the end host. Hence, the objective function C equals the number of fragments arriving at such host, which we will denote in the following by y. We thus have:

$$C = y = \textit{number of fragments received by the end node.}$$

Then, the problem that Bonuccelli intended to solve was that one of finding a path minimizing C and no longer than a given length. He studied the relation between C and the bottleneck value along a path and proposed a fast routing algorithm, called MAXBOTTLENECK, to find a maximum bottleneck path. This path turns out to be optimum with respect to the objective function C only if some conditions on the network MPS's are met. However, when used with the *bottleneck* fragmentation scheme, the algorithm always yields the path with the minimum value of C. We will go into details about the non-greedy bottleneck scheme in the next sections. Bonuccelli's approach leaves several questions still open. In particular it considers negligible the processing overhead in intermediate node and the overhead incurred by control information transmission.

Work reported in this paper was aimed to study the minimum fragmentation cost routing problem when also these overheads must be minimized. We will proceed by giving different problem formulations from those ones mentioned so far.

3 Problem Formulations

We model the internetwork as a graph $G = (N, E)$, where $N = \{1, \ldots, n\}$ is the set of nodes and $E \subseteq N \times N$ is the set of edges. The edges represent the individual networks and they are labeled by two positive numbers: l_{ij} and MPS_{ij}, to denote their length and the maximum packet size allowed by the corresponding network, respectively. The nodes represent the gateways between individual networks. Link lengths can be assigned in terms of different metrics: geographic distance in kilometres, average delay, link bandwidth, mean queue length, expected congestion or, in the simplest case, unit length for each link. The routing and fragmentation problem can be formulated as follows:

$$
\begin{aligned}
& min \left(\sum_{(i,j)\in E} l_{i,j} x_{ij} \right) y \\
& \sum_{(j,i)\in BS(i)} x_{ji} - \sum_{(i,j)\in FS(i)} x_{ij} = \begin{cases} -1 & \text{if } i = 1 \\ \ \ 1 & \text{if } i = n \\ \ \ 0 & \text{otherwise} \end{cases} \\
& S \le sy \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad (P1) \\
& s \le MPS_{ij} x_{ij} + M(1 - x_{ij}) \quad \forall (i,j) \in E \\
& y \text{ integer} \\
& x_{ij} \in \{0, 1\}
\end{aligned}
$$

where we denote:

x_{ij} : decision variable which takes value 1 if link (i, j) is selected in the path, value 0 otherwise.

y : number of fragments received by the destination host.

s : maximum fragment size.

S : initial packet size.

$BS(i)$: Backward Star of node i.

$FS(i)$: Forward Star of node i.

It should be noted that choosing the above objective function (i.e. the function that multiplies the path length by the number of fragments generated along the path) is the same as supposing that the packet fragmentation is performed by source nodes. This is a quite realistic case, adopted as fragmentation method also in the new IP protocol (IPv6), where no routers along a packet's delivery path is allowed to fragment, except for the source node [5].

The first constraints set in (P1) defines a path in G. The second and third sets state that the generated fragments number is always greater or equal to $\lceil S/s \rceil$ and that if a link is selected in the path, then the largest packet crossing that link must be smaller than the corresponding MPS.

The problem relative to the fragmentation schemes adopted and their effects on the objective funcion will be discussed later.

The particular structure of the above formulation can lead us to think of problem (P1) as a bicriterion optimization problem, that is a problem where two performance criteria, possibly in conflict with each other, must be optimized simultaneously. This observation was used as base idea to develop the solution algorithm.

However, P1 neglects the real number of networks crossed by the fragments originated from a single packet. In order to have a more precise value of the bandwidth usage overhead and of the processing cost for each passing-by fragment, we should change the objective function to include the case in which the fragmentation process is performed by intermediate nodes. We can thus introduce a second version of the problem, referred in the following as problem (P2).

We denote:

$\Phi_{ij} = \{s_1, s_2, \ldots, s_{y_{ij}}\}$: set of fragment sizes originated on link (i, j) from the fragments arriving to node i.

y_{ij} : number of fragments passing through link (i, j), i.e. $y_{ij} = |\Phi_{ij}|$

F_{ij} : function computing the sizes of the fragments crossing (i, j). In the following we will assume that F works according to the maximal fragmentation scheme.

Problem (P2) may be formulated as :

$$min \left(\sum_{(i,j)\in E} l_{i,j} x_{ij}\right) y_{ij}$$

$$\sum_{(j,i)\in BS(i)} x_{ji} - \sum_{(i,j)\in FS(i)} x_{ij} = \begin{cases} -1 \text{ if } i=1 \\ \ \ 1 \text{ if } i=n \\ \ \ 0 \text{ otherwise} \end{cases} \qquad (P2)$$

$$\Phi_{ij} = \begin{cases} F_{ij}(\{S\}) \text{ if } i=1 \ \& \ x_{ij}=1 \\ F_{ij}(\Phi_{ki}) \ \text{ if } x_{ki}x_{ij}=1 \\ \emptyset \qquad\qquad \text{otherwise} \end{cases}$$

$$y_{ij} \text{ integer}$$

$$x_{ij} \in \{0,1\}$$

where the function F_{ij} is defined as follows.

Procedure $F_{ij}(\Phi_{ki})$
begin
  $\Phi := \emptyset$;
  for each $s_r \in \Phi_{ki}$ **do**
    if $s_r \leq MPS_{ij}$ **then** $\Phi := \Phi \cup \{s_r\}$;
    else
    begin
      for $t = 1$ **to** $\lfloor s_r/MPS_{ij} \rfloor$ **do** $\Phi := \Phi \cup \{MPS_{ij}\}$;
      if s_r mod $MPS_{ij} \neq 0$ **then** $\Phi := \Phi \cup \{s_r \text{ mod } MPS_{ij}\}$;
    end {else}
  return Φ;
end $\{F_{ij}\}$.

In the following section we describe two different strategies to solve problem P1, while the discussion of a solution method for the more complex problem P2 is deferred to section 3.2.

3.1 Solution to Problem P1

In order to solve problem P1 we want to perform an ordered examination of the paths from the source node 1 to the destination node n. This ordered examination consists in seeking suboptimal solutions in the network with respect to the function y which counts the number of fragments generated. It means that the first path is the path along which the minimum number of fragments is produced and each subsequent path gives an equal or greater value of y than the previous one. When a new path is found, its length is compared to the length of the paths already generated in order to test its possible candidature to be the best path with respect to the objective function.

We now proceed to analyze how a proper choice of the fragmentation scheme makes it possible to generate the ordered sequence of minimum fragments number paths.

3.1.1 Choice of the Fragmentation Scheme

As we have already noticed, the number of fragments received by the destination host depends on the fragmentation scheme adopted, the initial packet size S and the MPS values encountered along the path followed (particularly the bottleneck value). In order to generate the sequence of suboptimal paths with respect to y, we should be able to relate y itself to the other quantities involved in the problem. Unfortunately, if we reason independently of the fragmentation scheme used, we can give only a lower bound on y. We have the following lemma [4].

Lemma 1 *Let d_j be a path from 1 to n and let μ_j be the minimum MPS in d_j. Then, for any initial packet size S and for any fragmentation scheme, we have*

$$y \geq \lceil S/\mu_j \rceil \tag{1}$$

This relation doesn't guarantee that the path with the largest possible value of μ will yield the minimum number of fragments (as we showed in a previous example). For the achievement of this aim, we would like to find a fragmentation scheme different from the greedy ones commonly used, for which equation (1) holds as equality. This is possible if we choose the *Bottleneck fragmentation scheme* proposed by Bonuccelli [4], which forces each gateway along a path to split the incoming packets into fragments with a size multiple of the bottleneck value, thus generating exactly $\lceil S/\mu_j \rceil$ fragments.

Bottleneck Fragmentation Scheme

Assume the bottleneck value of path d_j is known. Let $A_j = \mu_j \lfloor MPS_{rs}/\mu_j \rfloor$, where $(r,s) \in d_j$ (i.e., A_j is the maximum fragment size not greater than MPS_{rs} and multiple of μ_j). Then, when an oversize packet of size X must cross network (r,s), it is split into $\lfloor X/A_j \rfloor$ fragments of size A_j, plus one fragment of size X mod A_j if X mod $A_j \neq 0$.

The drawback of this scheme is that it doesn't always subdivide a packet into the minimum number of fragments in intermediate edges. Further, the bottleneck value must be inserted in the packet header, increasing the control information transmission. However, this overhead can often be considered negligible and it is guaranteed that from the bottleneck edge to the destination, the number of fragments is minimum.

We can now show how this fragmentation scheme is used in the process of efficient paths generation.

3.1.2 Algorithms

Let us assume to have k different values for the MPS's labeling the edges in G and let such values, denoted by $g_1, \ldots, g_k$, be sorted in decreasing order. This ordering implies that

$$\lceil S/g_1 \rceil \leq \lceil S/g_2 \rceil \leq \ldots \leq \lceil S/g_k \rceil.$$

Suppose the above quantities may have r distinct values, which we denote by $y_1, \ldots, y_r$. We thus have:

$$y_1 < y_2 < \ldots < y_r, \quad r \leq k.$$

The algorithm, starting with $i = 1$, at each iteration, considers the graph G_i defined as the subgraph of G induced by

$$E_i = \{(p,q) \in E \mid \lceil S/g_s \rceil \leq y_i,\ 0 \leq s \leq k\}.$$

Then, it computes the shortest path, d_i, from 1 to n in G_i with respect to the edges length. Clearly, the sequence of graphs considered by the algorithm will be such that:

$$G_1 \subset G_2 \subset \ldots \subset G_r$$

and, hence, the shortest paths returned at each step will result sorted in decreasing order with respect to their length, l_i, i.e.

$$l_1 \geq l_2 \geq \ldots \geq l_r$$

and in increasing order with respect to the number of fragments, y_i, reaching the destination (this is guaranteed by the use of the bottleneck fragmentation scheme).

Thus, problem P1 is equivalent to the following problem:

$$\min\{l_i y_i : i = 1, \ldots, r\}$$

Algorithm EP (Efficient Paths), which is described next, returns the best path d_{opt} with respect to the objective function, whose corresponding value is $F_{\min}$.

Let us suppose that the sorting of the MPS's values in decreasing order has already been performed in a previous step and let us denote by $\text{SP}(G_i)$ the procedure for computing the shortest path in the graph G_i. If node 1 and node n are not connected in G_i, the procedure returns a path of length ∞.

Algorithm EP[S, G, $(y_1, y_2, \ldots, y_r)$]
begin
  $d_{\text{opt}} := 0$; $F_{\min} := +\infty$;
  for $i := 1$ **to** r **do**
  begin
    $(l_i, d_i) := \text{SP}(G_i)$
    if $l_i y_i < F_{\min}$ **then**
    begin
      $F_{\min} := l_i y_i$; $d_{\text{opt}} := d_i$
    end{if}
  end {for}
end { EP}.

The time complexity of EP is $\text{O}(mn \log n + m^2)$ plus the time to carry out the sorting of the MPS's, i.e. $\text{O}(m \log n)$, where $m = |E|$ and $n = |N|$. The overall running time depends on the fact that the *for* loop is performed $r \leq m$ times. The most costly operation in it is the procedure SP. It requires $\text{O}(n \log n + m)$ time by using Dijkstra's algorithm implemented by mean of Fibonacci heaps [7].

If we choose the number of hops as length metric (i.e., $l_{ij} = 1, \ \forall (i,j) \in E$), the time complexity can be further reduced to $O(mn)$. In fact, in this case, problem P1 can be reduced in O(m) time to a bicriterion problem, where the two criteria are the number of edges, l, and the fragmentation, f. The fragmentation of a path d is defined as

$$f(d) = \max\{f_{ij} : (i,j) \in d\},$$

where $f_{ij} = \lceil S/MPS_{ij} \rceil, \ \forall (i,j) \in E$.

A path d is said to be *efficient* (or *Pareto-optimal*) with respect to the given criteria if no path d' exists such that $l(d) \geq l(d')$ and $f(d) \geq f(d')$, with the strict inequality holding in one of the two cases.

The bicriterion problem can be solved by enumerating all the efficient solutions and selecting the one which minimizes the product $l(d) \cdot f(d)$.

In the algorithm CEP (Cardinality Efficient Paths), which is given next, we associate to each node a list of labels, corresponding to tentative efficient subpaths. Each list will contain n elements corresponding to the n possible lengths of the paths reaching that node. The value contained in each position k, $0 \leq k \leq n-1$, represents the minimum number of fragments obtainable along all the paths of length less or equal to k till node i.

We denote:

List$(i) = [y_0^i, \ldots, y_k^i, \ldots, y_{n-1}^i]$: list of solutions associated to node i.

$Pv(i) = [Pv_0^i, \ldots, Pv_k^i, \ldots, Pv_{n-1}^i]$: list of the predecessors of node i. The generic element Pv_k^i denotes the predecessor of i along the path of length less or equal to k with the minimum number of fragments.

Q_{now} : set of nodes under consideration during the generic iteration k. It contains all the nodes reachable through a path of length k and for which a new candidate efficient solution was found in the previous iteration.

Q_{next} : set of nodes for which a new candidate efficient solution with length value $k+1$ is found during the kth iteration.

The algorithm computes the set of all the efficient solutions associated to each node in the graph. The best solution according to the choosen objective function will be given by the path to node n for which the product $k \cdot y_k^n$, $0 \leq k \leq n-1$, is minimum.

Algorithm CEP[G]
begin
  $y_0^1 := 1;\ Pv_0(1) := 0;$
  for $i := 2$ **to** n **do**
    for $k := 0$ **to** $n-1$ **do**
    begin
      $y_k^i := \infty;\ \ Pv_k^i := 0;$
    end {for}
  $Q_{\text{now}} := \{1\};\ Q_{\text{next}} := \{\};$
  for $k = 0$ **to** $n-1$ **do**
  begin
    for each $i \in Q_{\text{now}}$ **do**
    begin
      $y_{k+1}^i := y_k^i;\ Pv_{k+1}^i := i;\ Q_{\text{next}} := Q_{\text{next}} \cup \{i\}$;
      if $Pv_k^i \neq i$ **then**
        for each $(i,j) \in FS(i)$ **do**
        begin
          $\eta := \max\{y_k^i, f_{ij}\};$
          if $\eta < y_{k+1}^j$ **then**
          begin
            $y_{k+1}^j := \eta;\ Pv_{k+1}^j := i;$
            if $j \notin Q_{\text{next}}$ **and** $j \neq n$ **then** $Q_{\text{next}} := Q_{\text{next}} \cup \{j\}$;
          end{if}
        end{for}
    end{for}
    $Q_{\text{now}} := Q_{\text{next}};\ Q_{\text{next}} := \{\};$
  end{for}
end {Procedure}.

The time complexity of the above procedure derives from the fact that the external *for* loop is performed n times. For each of such loops at most m edges are examined and all the operations performed for each edge have time O(1). Hence, the overall time complexity is O(mn).

At termination each element $y_k^n \neq \infty$ of List(n) corresponds to a path with minimum number of fragments generated among those which have exactly k edges. The path can be easily found by backtraking from n to 1. If, for some k, it is $y_k^n = \infty$, then no path of length k exists from 1 to n.

The preference function used in P1 gives a good measure of fragmentation costs when the fragmentation process is performed by the source node. However, in the general case, it provides only an upper approximation of the actual costs. This penalizes those paths where the majority of fragments is generated in proximity to the destination host. In order to have a more exact estimation of the fragmentation costs, including the overhead due to fragments processing in each hop passed through, we should count the effective fragments-hops. In the section 3.2 we analyze how to achieve this purpose.

3.1.3 Implementation Issues

In this section we will discuss some of the issues involved in applying the theoretical ideas presented so far to realistic networks. An important question is how to implement our method in a distributed envinronment, where there is no central node that makes all the routing decisions and that has a complete knowledge of the state of the system.

Our method requires each node running the algorithm EP to have a view of the global state of the network. In particular, it must have a complete information about the MPS's values associated to each net in the internetwork envinronment. This requirements can not be achieved in those Autonomous Systems based on "distance vector" routing protocols [8]. In fact, this kind of protocol operates by having each gateway to maintain a table giving the best known distance to each destination and which link to use to get there. The information exchange is performed only towards neighbor nodes, so that the routing database turns out to be actually distributed among all the nodes and no one has a global view of the Internet.

However, under the link state technology [8], developed to overcome some undesirable characteristics of distance vector protocols and currently supported by many router vendors, the routing database is replicated rather than distributed. Instead of exchanging distances to destinations and trying to compute "best routes" in a distributed fashion, all the nodes maintain a complete copy of the network map and perform a complete computation of the best routes from this local map. To build this replicated database, containing the entire Internet topology, each gateway distributes link state packets listing the distances to each of the gateway's neighbors. The distribution is carried out by a flooding protocol, which guarantees a very fast and reliable information transmission to all the other nodes in the system.

The peculiarities of this technology make it possible to implement our method. This could be achieved by inserting the MPS's values in the link state packets together with the length information and running EP locally in the place of Dijkstra's algorithm, currently used in OSPF (Open Short Path First) protocols to determine the best route [9].

3.2 Solution to Problem P2: Fragments-Hops Minimization

In this section, we will provide an exact algorithm for determining the minimum fragments-hops path and, thus, the minimum global fragmentation overhead.

In the following, we will suppose that the packet fragmentation process is performed according to the maximal scheme.

Unfortunately, this algorithm has an exponential time complexity due to the need of examining a moltitude of subpaths when processing a node. In fact, the optimality property is not applicable to this problem, that is to say we can not state that if d_l is an optimal path from the origin 1 to the destination n and i is one of its intermediate nodes, then the subpath of d_l from 1 to i is an optimal path from 1 to i. However, we can carry out some pruning operations to reduce the number of subpath that must be considered. Such operations allow the removal of those subpaths to an intermediate node that generate a number of fragments-hops greater than the one generated along some source-destination path already computed. In order to carry out some reductions from the very beginning, the algorithm first computes the minimum length path (for semplicity, we will assume each link to have unit length) and calculates its corresponding fragments-hops number. Thus, we don't have to wait for the destination node to be reached the first time before being able to eliminate some cases. Further, the minimum length path should provide a 'good' number of fragments-hops to be used as reference point for some subpaths elimination (i.e., a relatively small number compared to that one provided by many other source-destination paths). As soon as an other path producing a smaller number of fragments-hops is found, it becomes the new reference for further eliminations.

We now go into details about the algorithm's working, showing first the data structures it uses.

3.2.1 Data Structures

We associate a table T^i with each node i in G to keep track of the information relative to all the subpaths leading to that node. Figure 2 shows a generic table T^i.

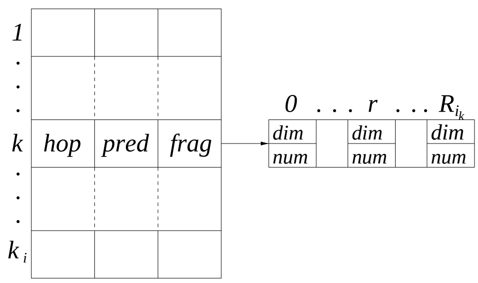

Figure 2: Table associated to node i.

Each row in T^i, denoted by T^i_k ($1 \leq k \leq k_i$), corresponds to a subpath to node i and contains three records. The first record, denoted by $T^i_k.hop$, stores the fragments-hops number produced if the subpath T^i_k is followed to reach node i. The second record, $T^i_k.pred$, indicates the predecessor of i along path T^i_k. In particular, it will contain an entry like T^j_h to indicate to which exact row of the predecessor j's table we must refer since, in turn, node j will be likely reachable from a multitude of subpaths. Finally, the record $T^i_k.frag$ is a pointer to an array, whose generic element r has itself two fields: the field $[T^i_k.frag]_r.dim$, with $0 \leq r \leq R_{i_k}$, stores the R_{i_k} different fragment sizes reaching i through the subpath T^i_k; the corresponding field $[T^i_k.frag]_r.num$ denotes how many fragments of size $[T^i_k.frag]_r.dim$ arrive to node i.

3.2.2 The Algorithm

The algorithm MFH (Minimum Fragment Hop) described next finds the path with minimum number of fragment-hops.

Algorithm MFH($S,\ G =< N,\ E >$)
begin
Let hop_{min} be the fragments-hops number of the minimum length path in G
obtained along d_l.
$Q := 1;\ k_1 := 1;\ T_1^1.hops := 1;\ T_1^1.pred := 0;$
$[T_1^1.frag]_{R_{1_1}}.dim := S;\ [T_1^1.frag]_{R_{1_1}}.num := 1.$
Tables initialization;
while $Q \neq \emptyset$ **do**
begin
Extract the first node from Q, say i;
for each $(i,j) \in FS(i)$ **do**
begin
for $k := 1$ **to** k_i **do**
begin
$hops := T_k^i.hop;$
for $r := 0$ **to** R_{i_k} **do**
$hops := hops + \lceil [T_k^i.frag]_r.dim / MPS_{ij} \rceil \times [T_k^i.frag]_r.num;$
if $hops < hop_{min}$ **then**
begin
if $j = n$ **then** $hops := hop_{min}$;
if $j \notin Q$ **then** $Q := Q \cup \{j\}$;
Add a new row k_j in T^j ($k_j := k_j + 1$);
$T_{k_j}^j.hop := hops;$
$T_{k_j}^j.pred := T_k^i;$
for $r := 0$ **to** R_{i_k} **do**
begin
Compute the fragment sizes generated on (i,j) from fragments stored in $[T_k^i.frag]_r$.
if the newly obtained sizes are already present at some position s in $T_{k_j}^j.frag$
then update only $[T_{k_j}^j.frag]_s num$
else add a new record R_{k_j} and set suitably both the fields dim and num.
end {for}
end {if}
end {for}
end {for}
end {while}

end {MFH}.

When the algorithm stops, hop_{min} yields the minimum fragments-hops number and the path originating that number can be constructed starting from the field $T_k^n.pred$ (where k is the row index whose field *hop* is set to hop_{min}) and following the row and table indexes contained in the fields *pred* backward till node 1.

3.2.3 Tables Reduction through Row Elimination

Thus far, our algorithm considered almost all the possible paths to the destination (except for those ones already exceeding the quantity hop_{min} in intermediate nodes). This can often cause the tables associated to the network nodes to grow up excessively, thus requiring a big amount of processing time and memory space. It could be reasonable to think about the possibility of reducing this resources consumption by eliminating some rows (i.e., some subpaths to a given node). The idea is to find out a way to express the concept of *dominance* among rows on which to base the choice of the rows to be eliminated during the algorithm's running. Meanwhile, we must guarantee that the row elimination process won't compromise the end solution optimality. Observations about the fragment sizes stored in a table's row, allow us to state the following.

Reduction conditions. A sufficient condition for a row k, $1 \leq k \leq k_i$, to be eliminable from table T^i is the existence of a row j in T^i such that:

C.1. $\forall s,\ 0 \leq s \leq R_{i_j}: \quad [T_j^i.frag]_s.dim \bmod [T_k^i.frag]_r.dim = 0 \quad \forall\ 0 \leq r \leq R_{i_k}$

C.2. $T_j^i.hop \leq T_k^i.hop$

If conditions (C.1) and (C.2) are both satisfied, we can say that row j *dominates* row i, i.e. under no circumstances it will be possible to obtain from row i a lower number of fragments-hops than that one achievable along any path to destination starting from row j. It is possible to show through a counter-example that no one of the above conditions can be relaxed without compromising the optimality of the final solution (for istance, by simply stating that all the sizes stored in array T_k^i to be submultiple of some size stored in array

T_j^i) [10].

3.2.4 Tables Reduction through Observations on Path Completion

Another way of reducing the table size can be based on the following property.

Property 1 *Let s be the maximum packet size generated along the optimal path P_{1j} from 1 to j and let μ_{jn} be the minimum MPS of the shortest path, P_{jn}, from j to n. Then, if $s \leq \mu_{jn}$, we can state that $P_{1j} \cup P_{jn}$ is an optimal path from 1 to n.*

This observation allow us to avoid computing the fragments-hops number generated along all the paths reaching the destination starting from the subpath till node j currently considered.

To carry on this reduction, each time a generic node j is reached, we should calculate the shortest path from j to n and check that the condition given in Property 1 is verified. If the test succeeds, rather than to add a new row in T^j which would next processed for each node of $FS(i)$, we compute the fragments-hops number generated along the path resulting from the concatenation of the subpath till node j currently considered and the shortest path from j to n. If that number is smaller than the value hop_{min} computed so far, we update hop_{min} and store the newly found path yielding that value.

4 Conclusions

In this paper we have surveyed the possibility of minimizing the packet fragmentation overheads by inserting the fragmentation cost, in term of number of fragments received by the end node, in the objective function to be minimized when choosing the best routing path from the origin to the destination network. We presented a routing algorithm, called EP, to solve this mathematical programming problem in the general case and the algorithm CEP for the case in which all the edges have unit length.

Further, we faced with the more complex problem of the total fragments-hops minimization. From the theoretical point of view, this approach is attractive because it handles a more precise estimation of the fragmentation cost. However, the solution algorithm could result fairly complex and not practical in many actual cases, due to an excessive growth in the problem size. The solution that we propose can be used as starting point for further improvements. One example could be the analysis of the fragment size sparsity in order to find less strict conditions under which to allow rows elimination. We could also think about the possibility to cancel rows which are likely not to lead to the optimal path and study through heuristics the effects of this "non-optimal" reduction on the final solution worsening, in order to state its effective worth.

Aknowledgement

The authors wish to thank Stefano Pallottino for the fruitful exchange of ideas and for the important hints he provided for the realization of the algorithm "CEP".

References

[1] J. Postel, "Internet Protocol", RFC-791, Network Information Center, SRI International, September 1981.

[2] C. J. Bennet, "The Overheads of Transnetwork Fragmentation", *Computer Networks*, 6, pp. 21-36, 1982.

[3] A. Orda, R. Rom, "Optimal Packet Fragmentation and Routing in Computer Networks", *Networks,* vol.29, pp. 11-28, 1997.

[4] M.A. Bonuccelli, "Minimum Fragmentation Internetwork Routing", *Proceedings of INFOCOM'91,* Bal Harbor, FL, pp. 289-294, April 1991.

[5] W. Stalling, "IPv6: The New Internet Protocol", IEEE *Communications Magazine,* pp. 96-108, July 1996.

[6] A. Osyczka, "Multicriterion Optimization in Engineering with FORTRAN Programs", Ellis Horwood, 1984.

[7] M. L. Fredman and R. E. Tarjan, "Fibonacci Heaps and Their Uses in Improved Network Optimization Algorithms", *Journal of the ACM*, 34(3), pp. 596-615, 1987.

[8] A.S. Tanenbaum, "Computer Networks", Third Edition, Prentice-Hall International, Inc.

[9] C. Huitema, "Routing in the Internet", Prentice Hall PTR, Englewood Cliffs, New Jersey.

[10] M. Scaparra, "Routing con costo minimo di frammentazione", Tesi di Laurea (1997), Universitá degli studi di Pisa.