Efficient Algorithms for Erasure Node Placement on Slotted Dual Bus Networks *

Bhagirath Narahari † Sunil Shende ‡ Rahul Simha §

April 15, 1998

Abstract

We study the problem of placing erasure nodes among passive stations in a slotted dual bus network. Erasure nodes are known to improve throughput by allowing slot reuse. It is also known that choices made in locating erasure nodes significantly impact network congestion and overall throughput – especially when traffic patterns exhibit a high degree of locality. We present algorithms to determine optimal placements of erasure nodes that improve upon prior work on this problem: they are simpler and faster polynomial-time algorithms and are easily modified to handle various useful cost measures. These algorithms can be used to solve related placement problems in which limits on congestion and existing placements are given as input, and the goal is to find the minimum number of erasure nodes required to meet the congestion bound.

1 Introduction

The simplicity of implementation and mostly efficient scheduling of packet transmissions have prompted much interest in slotted dual bus networks (in particular, the DQDB protocol and its variations – see [6] for a survey). However, many have observed that the stations in such a system (whether using DQDB or not) are designed to be electrically passive, thereby precluding destination removal of packets and consequent slot reuse [10]. Accordingly, some people have proposed enhancing a limited number of stations with an 'erasure' capability; this modification would allow slots containing packets that have passed their destination to be declared empty for use by other stations [10], consequently reducing congestion and increasing network throughput [1, 2, 3, 4, 10]. However, it is known that choices made in locating these erasure nodes considerably impact throughput. The question thus arises: what is the best placement of erasure nodes?

*Parts of this paper appeared in the International Conference on Communications (ICC), Seattle, June 1995, pp. 935-939.

†Department of Electrical Engineering and Computer Science, The George Washington University, Washington, DC 20052. E-mail: narahari@seas.gwu.edu.

‡Department of Computer Science and Engineering, University of Nebraska at Lincoln, Lincoln, NE 68588. E-mail: sunil@calypso.unl.edu.

§Department of Computer Science, College of William and Mary, Williamsburg, VA 23185. E-mail: simha@cs.wm.edu.

Erasure nodes are introduced in [10] and their performance benefits are studied in [4, 10, 13] without analysis of placement problems. For any given set of traffic conditions (traffic rates between nodes) the value of any particular placement is typically identified by the congestion (a precise definition of congestion is given in the next section) resulting from that placement. Thus, the lower the congestion, the better the placement. The resulting minimization problem can equivalently be written as a maximization problem, by considering the inverse as a cost measure (see [9], for example). Placement problems are considered in [2, 8, 9, 11, 12]. Uniform traffic has been assumed in [2, 11, 12]. The algorithm presented in [2] for uniform traffic loads is based on a continuous approximation; the method appears to be difficult to implement for non-uniform traffic loads. The algorithm in [8] is pseudo-polynomial: its complexity depends on the magnitude of the cost function. The reader is referred to [9] for additional discussion on the above literature.

Our paper is inspired by the problem formulation and results presented in [9], which include the first polynomial-time algorithm to date for the placement problem. For a network of n stations, k among which are to be erasure nodes, the complexity of their algorithm for a single bus can be outlined as follows. An $O(n^2)$ computation is carried out to determine a congestion matrix (a matrix created as part of the computation); following this, an assignment graph containing $O(n^2)$ vertices and $O(n^2k)$ edges is created with edge weights from the congestion matrix; the edges are then sorted by weight (in $O(n^2k\log n)$ time) and a binary search over edge weights is conducted in $O(n^2k\log n)$ time, with each iteration using a depth-first search of the assignment graph. Thus the complexity is $O(n^2+n^2k+n^2k\log n+n^2k\log n)=O(n^2k\log n)$, in addition to the data structures and code required to manipulate graphs. In contrast, we present a simpler and faster $O(n^2+nk)$ algorithm, $O(n^2)$ for the congestion matrix and $O(nk)$ for the optimal placement, by exploiting some monotonicity properties of the cost measure. Similarly, our dual bus algorithm is faster than the dual bus algorithm in [9]. Other placement problems such as deciding how many erasure nodes will be required to meet a congestion bound, are also studied in this paper.

Note that the problem formulation in this paper (as in other papers on this problem) assumes that traffic rates between nodes are fixed. This is a reasonable assumption when erasure nodes are installed (or placed) only very infrequently. In this case, the long-term traffic rates between nodes would naturally be the data input to the problem. However, a placement based on such long-term averages cannot adapt to temporal variations in the rates. To adapt to short-term variations in load, the capability to turn erasure capability on and off would be needed. If this capability were available, measurements of traffic rates over short periods could be used to update placements adaptively.

The next section introduces notation and formulates the problem. Section 3 contains the key ideas in the paper, described for a single bus. These are extended to the dual bus in Section 4, along with considering a related cost measure and additional problem variations.

2 Preliminaries

Let $S=\{1,2,\ldots,n\}$ denote the set of stations on the network [1]; we assume that the stations are labeled in increasing order along the forward bus. By default, stations 1 and n are considered erasure nodes since they generate slots on the buses. For any positive integer m such that $m\le k+1$ (k is the

[1] We use the words "station" and "node" interchangeably.

number of erasure nodes), consider a vector, $a = (a_1, \ldots, a_m) \in \mathbb{Z}_+^m$, such that $1 < a_i < a_{i+1} \leq n$, for all $1 \leq i < m$. Vector a denotes an *m-placement* of erasure nodes, with the intuitive meaning that node a_i, $1 \leq i \leq m$, is the i^{th} erasure node. For convenience, we will use a_0 to designate node 1 and a_{k+1} for node n (when $m = k+1$). Next, let $S(m,j)$ denote the set of m-placement vectors which satisfy $a_m = j$. Given nodes i and j with $i < j$, the subset of contiguous stations $\{i, i+1, \ldots, j\}$ is denoted as *segment* $[i,j]$. For stations $p < q \leq r < s$, note that r may transmit to s by potentially reusing a slot on the forward bus that was earlier used by a message traveling from p to q, if there is an erasure node that lies within the segment $[q,r]$.

We will assume that average traffic requirements between all possible pairs of distinct stations are provided as input to the problem and that these traffic rates are constant. Specifically, let λ_{ij} be the *average* rate of traffic (slots per unit time) required by station i to transmit information to station j ($\lambda_{ii} = 0$ for all i). Define

$$\phi(i,j) \quad = \quad \begin{cases} \sum_{r=1}^{j-1} \sum_{s=\max\{i+1,r+1\}}^{n} \lambda_{rs} & \text{if } i < j \\ \sum_{r=j+1}^{n} \sum_{s=1}^{\min\{i-1,r-1\}} \lambda_{rs} & \text{if } i > j \\ 0 & \text{if } i = j \end{cases} \tag{1}$$

Intuitively, $\phi(i,j)$ is the total traffic carried on segment $[i,j]$, (or the average bandwidth usage on the segment) given erasure nodes at i and j and *assuming that there are no erasure nodes at stations between i and j*. Next, for any $a \in S(k+1,n)$ and $i \in \{1, \ldots, k\}$, let $u_i(a) = \phi(a_{i-1}, a_i)$ and $v_i(a) = \phi(a_i, a_{i-1})$. We define the following two measures of the (congestion) *cost* resulting from placement a:

$$\begin{aligned} w(a) &= \max\{\max_i u_i(a),\ \max_i v_i(a)\}, \\ z(a) &= \max_i u_i(a) \ + \ \max_i v_i(a). \end{aligned}$$

Our goal is to find placements that minimize the functions w and z. Intuitively, any placement a with k erasure nodes, naturally induces $k+1$ segments on the forward and reverse buses, with each segment beginning and ending at an erasure node. Here, $w(a)$ measures the total bandwidth requirement of the most congested segment (regardless of the bus), while $z(a)$ is the sum of the bandwidth requirements for the most congested segments on each of the buses. Both are useful measures of congestion and have been implicitly studied in [9, 10]. In [9], a scalar multiple of the inverse of $z(a)$, identified as the largest achievable throughput, is maximized. Observing that this maximization corresponds to the minimization of $z(a)$, we see that both versions of the problem are identical.

3 Minimization of $w(a)$ for a single bus

We initially restrict ourselves to analyzing a *single bus* (the forward bus, say), in isolation, which not only develops the main ideas behind our algorithms, but also forms a crucial ingredient of the technique needed to minimize $z(a)$ for the dual bus case. Thus, in this section, we assume that $\lambda_{i,j} = 0$ whenever $i \geq j$. For any m-placement $a = (a_1, a_2, \ldots, a_m) \in S(m,j)$ let α be any integer that satisfies $1 < \alpha \leq n$ and $\alpha \notin \{a_1, a_2, \ldots, a_m\}$. Let b denote the unique $m+1$-placement obtained by augmenting a with α; we say that b is obtained by *extending* placement a with α. For a placement a of any size larger than or equal to m, let $\bar{w}(m,a) = \max_{i \le m} u_i(a)$ and define

$$F(m,j) = \min_{a \in S(m,j)} \bar{w}(m,a) = \min_{a \in S(m,j)} \max_{i \le m} u_i(a). \tag{2}$$

Intuitively, $\bar{w}(m,a)$ is the cost of the placement a, while only considering the first m initial segments induced by the erasure nodes. $F(m,j)$ is the minimum such cost over all placements in $S(m,j)$.

Lemma 3.1

$$F(m,j) = \begin{cases} \phi(1,j) & \textit{if } m = 1 \\ \min_{m-1<i<j} \max\{F(m-1,i), \phi(i,j)\} & \textit{otherwise} \end{cases} \tag{3}$$

for m, j satisfying $1 \le m \le k+1$ and $m < j \le n$.

Proof: By (2) above, $F(1,j) = \phi(1,j)$ for all $j > 1$. Now, let $m > 1$ and consider a, an m-placement which achieves the cost $F(m,j)$. Then, either $u_m(a) = F(m,j)$ or $u_m(a) < F(m,j)$ since by (2), $F(m,j) < u_m(a)$ is not possible.

1. CASE $u_m(a) = F(m,j)$: Then, $F(m,j) = u_m(a) \ge \bar{w}(m-1,a) \ge F(m-1,a_{m-1})$
2. CASE $u_m(a) < F(m,j)$: In this case, $F(m,j) = \bar{w}(m-1,a)$. Now, if the $(m-1)$-placement $(a_1,\ldots,a_{m-1})$ does not achieve the optimal cost $F(m-1,a_{m-1})$ then there must be an $(m-1)$ placement, $b = (a'_1,\ldots,a'_{m-2},a_{m-1})$, of *strictly* lower cost, achieving $F(m-1,a_{m-1})$. Let $a' = (a'_1,\ldots,a'_{m-2},a_{m-1},a_m)$ be the placement obtained by extending b with a_m; then, $\bar{w}(m,a') \ge F(m,j) > \max\{\bar{w}(m-1,a'),u_m(a)\} = \bar{w}(m,a')$, which is an obvious contradiction. Note that the second inequality follows from the better placement implication $\bar{w}(m-1,a') < \bar{w}(m-1,a) = F(m,j)$ and the inequality $F(m,j) > u_m(a)$, the assumption of this second case. ∎

Lemma 3.1 suggests a simple $O(n^2k)$ dynamic programming algorithm, which we henceforth refer to as SB (for Single-Bus), to compute a placement of k erasure nodes that minimizes $w()$ for the forward bus. First, observe that $\phi(i,j)$ can be computed in $O(n^2)$ time. We describe this computation for $i < j$; the case $i > j$ is similar. (1) compute $\lambda(i) \triangleq \sum_{m>i} \lambda_{im}$, the total traffic entering at node i; (2) compute $\gamma(i) \triangleq \sum_{m<i} \lambda_{mi}$, the total traffic leaving at node i; (3) compute $\beta(i) \triangleq \beta(i-1) + \lambda(i) - \gamma(i)$ (here, $\beta(0) = 0$), the traffic carried past node i, if i is the only erasure node; (4) finally, use the recurrence $\phi(i,i+1) \triangleq \beta(i)$, $\phi(i,j) \triangleq \phi(i,j-1) + \lambda(j-1)$ for $j > i+1$. Note that each of $\lambda(i), \gamma(i)$ and $\beta(i)$ can be computed in $O(n^2)$ time, thereby making the straightforward computation of ϕ an $O(n^2)$ computation.

Assuming each $\phi(i,j)$ is stored in a two-dimensional array, the values of $F(m,j)$ can also be stored in an array. From equation (3), it is easy to see that F can be computed in $O(n^2k)$ time; the actual solution is obtained using standard pointer-based tracking in dynamic programming.

Note that we already have a faster algorithm for a single bus than the one in [9]. However, we now develop an even faster algorithm than Algorithm SB by taking advantage of a monotonicity property of the cost measure. Some details need to be established before we present this faster algorithm. First, we observe from Equation (1) that ϕ is monotonically non-decreasing in its second argument and monotonically non-increasing in its first. Next, we show that the matrix F computed by algorithm SB also exhibits monotonicity along each row:

Lemma 3.2 *Let m be a an integer satisfying $1 \le m \le k$. Then for any j such that $m+1 < j \le n$, we have $F(m, j-1) \le F(m, j)$.*

Proof: Fix values of m and j such that $m+1 < j \le n$, and consider an m-placement, $a = (a_1, a_2, ..., a_m)$, with cost $F(m,j)$ and $a_m = j$. Since $a_1 > 1$ and $a_m = j > m+1$, the pigeonhole principle implies that there is a node in the set, $\{2, 3, \ldots, j-1\}$, which is not an erasure node. Let α be the node with the largest label which satisfies this property.

Now, consider the m-placement, b, obtained as follows. First, remove the erasure node at $a_m = j$ in a, and then extend the resulting $(m-1)$-placement with α. It is clear from the choice of α that node $(j-1)$ is an erasure node in placement b. Hence, $b \in S(m, j-1)$ and every segment between consecutive erasure nodes of b is either a segment in a, or is a sub-segment of a segment in a. By the monotonicity of ϕ, it follows that the cost of b is at most the cost of a, yielding the conclusion $F(m, j-1) \le F(m, j)$. ∎

We now fix some $m > 1$ and focus our attention on the m^{th} stage of algorithm SB. Define the function $\pi_{m,j}$ for any $i \in [m,\ j-1]$ as follows:

$$\pi_{m,j}(i) \quad = \quad \max\{F(m-1, i),\ \phi(i, j)\}. \tag{4}$$

Observing Equation 3, we see that $F(m,j)$ is the minimum among the sequence of *candidate* values $\pi_{m,j}(m), \pi_{m,j}(m+1), \ldots, \pi_{m,j}(j-1)$. Now, suppose we observe the above sequence of candidate values, as i is incremented starting from $i = m$. Define $\hat{i}_{m,j}$ to be the smallest i (in the range $[m, j-2]$) for which a *strict* increase is observed going from the candidate value $\pi_{m,j}(i)$ to the candidate value $\pi_{m,j}(i+1)$. If there is no such i, then we let $\hat{i}_{m,j} = j-1$.

Lemma 3.3 *Consider m such that $1 < m \le k$. Then, the candidate values $\pi_{m,j+1}(i)$ for $i < \hat{i}_{m,j}$ are larger than the minimum value for $F(m, j+1)$. Moreover, $\pi_{m,j}(\hat{i}_{m,j})$ is a minimum among all the candidate values for $F(m,j)$. More precisely,*

1. *For every j such that $m < j < n$, $\hat{i}_{m,j+1} \ge \hat{i}_{m,j}$.*
2. *$F(m,j) = \pi_{m,j}(\hat{i}_{m,j})$.*

Proof: Assume to the contrary that for some $j > m$, $\hat{i}_{m,j+1} < \hat{i}_{m,j}$. Consequently, $\hat{i}_{m,j+1} < j-1$, *i.e.* there is a strict increase in $\pi_{m,j+1}(i)$ at $i = \hat{i}_{m,j+1}$. Since for $j+1$ fixed, $\phi(i, j+1)$ is non-increasing in i, the strict increase in $\pi_{m,j+1}(i)$ occurs only because of a strict increase in $F(m-1, i)$ at $i = \hat{i}_{m,j+1}$. Hence, $F(m-1, \hat{i}_{m,j+1}) < F(m-1, \hat{i}_{m,j+1}+1)$ and $\phi(\hat{i}_{m,j+1}, j+1) \le F(m-1, \hat{i}_{m,j+1})$ together yield that at $i = \hat{i}_{m,j+1}$, $\pi_{m,j}(i) = \max\{F(m-1, i),\ \phi(i,\ j)\} < F(m-1, i+1) \le \pi_{m,j}(i+1)$. Thus, there is a strict increase in $\pi_{m,j}(i)$ at $i = \hat{i}_{m,j+1}$, which contradicts the choice of $\hat{i}_{m,j} > \hat{i}_{m,j+1}$ as the smallest index registering such an increase. This proves Assertion 1 of the lemma.

By definition of $\hat{\imath}_{m,j}$, it is clear that the value $\pi_{m,j}(\hat{\imath}_{m,j})$ is minimum for i in the range $[m,\ \hat{\imath}_{m,j}]$. Moreover, as indicated above, the strict increase in $\pi_{m,j}(i)$ at $i = \hat{\imath}_{m,j}$ can only be a consequence of a strict increase in $F(m-1, i)$. Assertion 2 is now immediate from the preceding discussion and the fact that $F(m-1, i)$ is non-decreasing as i increases (from Lemma 3.2). ∎

Thus, while computing $F(m, j+1)$, we can as well commence our search at $i = \hat{\imath}_{m,j}$. As a consequence, in the m^{th} stage, we can compute the values of $F(m, m+1), F(m, m+2), \ldots, F(m, n)$ by making exactly *one sweep* of all the $O(n)$ possible values of i that contribute to finding a minimum for the right hand side of Equation 3. We provide an abbreviated description (in pseudo-code) of the computation of the m^{th} stage: henceforth, we denote by Algorithm Fast-SB, the faster version of SB that is implicit in the description.

Stage $m > 1$ of algorithm Fast-SB

$F(m, m+1) \leftarrow \max\{F(m-1, m), \phi(m, m+1)\}$;
$j \leftarrow m+2$;
$i \leftarrow m$;
while $(j \leq n)$ **do**
  while $(i \leq j-2)$ **and** $(\pi_{m,j}(i) \geq \pi_{m,j}(i+1))$ **do**
    increment i;
  $F(m, j) \leftarrow \pi_{m,j}(i)$;
  increment j;
endwhile;

Since there are $O(k)$ stages in computing $F(k+1, n)$ (the minimum value of w) and each stage takes $O(n)$ steps, the algorithm takes $O(nk)$ steps to compute the optimal placement. Note that only the previous row $F(m-1, \cdot)$ is needed to compute values of $F(m, j)$, hence the algorithm only needs $O(n)$ additional space. Another point about space and time requirements can be made here. We have assumed that the pre-computation of the ϕ matrix takes $O(n^2)$ time; in fact, it takes at least $O(n^2)$ time since every upper-diagonal entry of the traffic matrix must be inspected at least once. Since k is at most as large as n, Algorithm Fast-SB is at most $O(n^2)$ and therefore *optimal* (to within a constant multiple). If $k << n$, then the entire algorithm takes $O(n^2 + nk) = O(n^2)$ time; however, it is easy to show the overall time is $O(nk)$ when flow equations for each node are solved a priori (that is, the flow into and out of each node is known).

4 Generalizations and Problem Variations

4.1 Minimizing $w(a)$ for dual buses

Algorithm Fast-SB can be easily modified to minimize $w(a)$ for dual buses. For any segment between two successive erasure nodes i and j, recall from Section 2 that the segment congestion with respect to the $w(\cdot)$ is given by max $\{\phi(i, j),\ \phi(j, i)\}$. Since this quantity is a function of i and

j, it is convenient to define for all $i < j$, $\phi_w(i,j) = \max\ \{\phi(i,j),\ \phi(j,i)\}$. Then, the cost $w(a)$ for any given k-placement a is given by $w(a) = \max_i\{\phi_w(a_{i-1}, a_i)\}$.

Let $a = (a_1, \ldots, a_m, \ldots, a_j)$ denote a placement of any size larger than or equal to m, and let $\bar{w}(m,a) = \max_{i \leq m}\ \{\phi_w(a_{i-1},\ a_i)\}$. Next, define the term $G(m,j) = \min_{a \in S(m,j)}\{\bar{w}(m,a)\}$, the cost on the dual bus of the optimal placement of m erasure nodes to stations $\{1, \ldots, j\}$. The following result, whose proof is similar to that of Lemma 3.1 [7], establishes the dynamic programming equation for the dual bus.

Lemma 4.1

$$G(m,j) = \begin{cases} \phi_w(1,j) & \textit{if } m = 1 \\ \min_{m-1<i<j} \max\{G(m-1,i),\ \phi_w(i,j)\} & \textit{otherwise} \end{cases} \tag{5}$$

for m, j *satisfying* $1 \leq m \leq k+1$ *and* $m < j \leq n$.

Again, the straightforward dynamic programming algorithm suggested by equation (4.1) can be sped up using similar monotonicity properties as before. In this case, $\phi_w(i,j)$ is non-increasing in i and non-decreasing in j; also, $G(m,j)$ is non-decreasing in j. These may be proved exactly along the lines of Lemma 3.2. Accordingly, it is easy to modify Algorithm FAST-SB to obtain an algorithm (which we call Algorithm FAST-DB) that computes a placement for the dual bus optimizing the congestion measure $w(\cdot)$. The modifications simply consist of replacing ϕ with ϕ_w, and replacing F with G everywhere in Algorithm FAST-SB.

4.2 Minimizing $z(a)$ for dual buses

We now turn our attention to the congestion measure $z(a)$ defined in Section 2 and show that Algorithm SB from Section 3 can be deployed again with some modifications, at the expense of increasing the complexity of the algorithm.

Recall that for placement a, $z(a) = \max_i u_i(a) + \max_i v_i(a)$, where $u_i(a) = \phi(a_{i-1}, a_i)$ and $v_i(a) = \phi(a_i, a_{i-1})$ denote the congestion, respectively on the forward and reverse buses, between consecutive erasure nodes a_{i-1} and a_i. Now, consider two independent runs of Algorithm SB, first to compute an optimal $w(\cdot)$ cost for the forward bus traffic and then to compute an optimal $w(\cdot)$ cost for the reverse bus traffic. Let u^* and v^* denote the optimal $w(\cdot)$ costs found in these two runs. Although the placements which achieve these respective costs will generally differ from each other, it is easy to see that $u^* + v^*$ is a *lower bound* on the optimal $z(a)$ cost for the bus system. To minimize the sum of the two terms in the expression for $z(a)$, we take the approach of minimizing just one term, conditioned on an *upper* bound on the other term.

Formally, given an integer δ, let $S_\delta(m,j)$ be the subset of all legal placements in $S(m,j)$ defined by $S_\delta(m,j) = \{a \in S(m,j) :\ \max_i v_i(a) \leq \delta\}$. Suppose that $a \in S_\delta(k+1, n)$ is an optimal placement found by Algorithm SB by considering only the forward bus traffic in isolation, but subject to the provision that $\phi(i,j)$ is *undefined* for arguments $i < j$ when $\phi(j,i) > \delta$. In other words, δ being a threshold on the tolerable congestion on the reverse bus, automatically prohibits i and j from being considered as candidates for consecutive erasure node placements on the forward bus. It is not hard to see that $Z_\delta = \delta + \max_i u_i(a)$ is an *upper* bound on the optimal $z(\cdot)$ cost for the dual buses. Therefore, the idea is to start with $\delta = v^*$ and compute an optimal forward bus placement in $S_\delta(k+1,n)$. The computation proceeds in stages: if Z_δ is the cost conditioned on some threshold δ from the previous stage, then in the current stage, we take the next higher value for δ from the sub-matrix of ϕ corresponding to the reverse bus, and compute Z_δ as before. If the process is repeated until all the distinct values of $\delta \geq v^*$ are exhausted, then the optimal placement is one which minimizes Z_δ in among the sequence of placements computed above.

The crucial observation is that there are only $n(n-1)/2$ possible distinct values for $\delta \geq v^*$, hence the computation loop that is implicit in the preceding paragraph needs to be executed only $O(n^2)$ times: each pass through the loop involves one execution of Algorithm SB. At first glance, this gives us a time complexity of $O(n^2 \log n \ + \ kn^4)$ to compute a placement minimizing the congestion measure $z(\cdot)$, where the first term counts the time needed to order the candidate values for δ. However, it is again possible to speed up the overall algorithm by using Fast-SB within the loop, with a couple of modifications.

First, for every threshold cost δ, we redefine the corresponding forward bus traffic sub-matrix as follows. Define

$$\phi_\delta(i,j) \quad = \quad \begin{cases} \sum_{r=1}^{j-1} \sum_{s=\max\{i+1,r+1\}}^{n} \lambda_{rs} & \text{if } i < j \textit{ and } \phi(j,i) \leq \delta \\ \infty & \text{otherwise} \end{cases} \tag{6}$$

Simultaneously, we now allow for the possibility that some entries $F(m,j)$ may be *undefined*, *i.e.* of infinite cost. Such entries will reflect the fact that under the threshold δ constraint, it is impossible to find an m-placement on the forward bus ending at station j. Formally, we define

$$F_\delta(m,j) \quad = \quad \begin{cases} \min_{a \in S_\delta(m,j)} w(a) & \text{if } S_\delta(m,j) \text{ is non-empty} \\ \infty & \text{otherwise} \end{cases} \tag{7}$$

The modified algorithm Fast-SB that is used in the computation loop described above, is nothing but the original Fast-SB with the following simple changes: when the threshold weight is δ, we replace ϕ everywhere by ϕ_δ and F by F_δ. The proofs in Section 3 can be modified to show corresponding monotonicity properties of ϕ_δ and F_δ. Thus,

Theorem 4.1 *An optimal placement of k erasure nodes that minimizes the congestion measure $z(\cdot)$ for a dual bus can be computed using the above modification of Algorithm* Fast-SB *as a subroutine, in $O(n^3k)$ time and $O(n)$ additional space.*

4.3 Minimal-placement problems

Consider the case in which we are given the traffic rates λ_{ij} and a *upper bound*, δ_w, on the congestion $w(\cdot)$ that can be tolerated by the system. The goal is to determine a placement that uses a minimum number of erasure nodes while ensuring that the resulting congestion is no more than δ_w.

The solution to the problem is implicit in algorithm Fast-DB discussed earlier. Recall that $G(k+1,n)$ is the optimal $w(\cdot)$ cost if k is the number of erasing nodes placed on the dual buses. Since the matrix G is computed row-by-row in a dynamic fashion, we can simply create new rows until a row, k^*, is fully computed and $G(k^*+1,n) \leq \delta_w$ holds true. From the preceding discussion and the correctness of Algorithm Fast-DB, it follows that k^* is indeed the minimum number of erasing nodes needed. Hence, our algorithm is output-sensitive: it determines k^* in $O(nk^*)$ time steps using $O(n)$ additional space. Note that a different algorithm, without accompanying analysis, is given in [8] for the same problem.

A related placement problem takes a traffic matrix, $(\lambda_{i,j})_{(n\times n)}$, a *fixed* m-placement, $a = (a_1, a_2, \ldots, a_m)$, and a congestion threshold δ_w as inputs. The goal is to determine the placement of a minimum number of *additional* erasing nodes, so that the resulting placement has congestion $w(\cdot) \le \delta_w$. It is easy to see that if any segment, $[i,j]$, induced by the given placement a, already satisfies the congestion constraint, then there is no need to sub-divide the segment further by placing additional erasing nodes between stations i and j. Hence, it suffices to run the algorithm that solves our first problem (in the previous paragraph), but independently on only those segments whose congestion exceeds the threshold δ_w. For every such segment containing n_i stations, we need $O(n_i k_i)$ time to compute the placement of k_i additional erasing nodes within the segment (where k_i is the minimum number of additional nodes required). Since segments are disjoint, it again follows that the entire algorithm runs in $O(nk^*)$ time and uses only $O(n)$ space, where k^* is the minimum number of additional erasing nodes altogether.

5 Summary

We considered the problem of optimally placing erasure nodes in a dual bus network given average traffic rates between pairs of nodes. We have presented significantly faster and simpler algorithms than existing ones and also considered useful variations of the general problem. Future research might include applying similar ideas to other types of networks.

References

[1] F.Borgonovo, A.Lombardo, S.Palazzo and D.Panno. FQDB: A Fair Multisegment MAC Protocol for Dual Bus Networks. *IEEE J.Sel. Areas in Comm.*, Vol. 11, No. 8, October 1993, pp. 1240-1248.

[2] M.W.Garret and S-Q.Li. A Study of Slot Reuse in Dual Bus Multiple Access Networks. *IEEE J. Sel. Areas in Comm.*, Vol. 9, No. 2, February 1991, pp. 248-256.

[3] A.E.Kamal. Efficient Multi-segment Message Transmission with Slot Reuse on DQDB. *Proc. IEEE INFOCOM 91*, Bal Harbour, Florida, April 1991, pp. 869-878.

[4] D.Luciani, R.Pignatelli and L.Susanna. The 'Balanced' Erasure Node: A Mechanism for Slot Reuse in DQDB Protocol. *Proc. IEEE Int. Conf. Comm.*, Denver, June 1991, pp. 1350-1354.

[5] P.B.Mirchandani and R.L.Francis, eds. *Discrete Location Theory. Wiley*, 1990.

[6] B.Mukherjee and C.Bisdikian. A Journey Through the DQDB Network Literature. *Performance Evaluation*, Vol. 16, No. 1, 1992, pp. 129-159.

[7] B.Narahari, S.Shende and R.Simha. Efficient Algorithms for Erasure Node Placement on DQDB Networks. *Technical Report UNL-CSE-94-021*, Department of Computer Science and Engineering, University of Nebraska, Lincoln, NE, 1994.

[8] P.G.Potter and M.Zukerman. Analysis of a DQDB Subnetwork with Erasure Nodes. In A.Jensen and V.B.Iversen (Eds), Teletraffic and Datatraffic in a Period of Change, *North Holland*, 1991, pp. 941-946.

[9] S.Ray and S.Mukherjee. On Optimal Placement of Erasure Nodes on a Dual Bus Network. *Proc. IEEE INFOCOM*, April 1995, Boston, MA.

[10] M.A.Rodrigues. Erasure Node: Performance Improvements for the IEEE 802.6 MAN. *Proc. IEEE INFOCOM 90*, San Francisco, June 1990, pp. 636-643.

[11] O.Sharon and A.Segall. On the Efficiency of Slot Reuse in the Dual Bus Configuration. *IEEE/ACM Trans. on Networking*, Vol. 2, No. 1, February 1994, pp. 89-100.

[12] O.Sharon and A.Segall. Schemes for Slot Reuse in CRMA. *IEEE/ACM Trans. on Networking*, Vol. 2, No. 3, June 1994, pp. 269-278.

[13] M.Zukerman and P.G.Potter. A Protocol for Erasure Node Implementation within the DQDB Framework. Proc. IEEE GLOBECOM 90, San Diego, CA, December 1990, pp. 1400-1404.