On Congestion Problems in Digital Cross-Connect (DCS) Transport Networks

Ramesh Bhandari
AT&T Laboratories
Room 3F-208, 307 Middletown-Lincroft Road
Lincroft, New Jersey 07738, USA
(732) 576-5858
Fax: (732) 576-6075
rbhandari@att.com

Abstract

In *today's environment where traffic demands within a network are increasing rapidly, planning and designing a network are assuming great importance. Rapidly rising traffic within the network is leading to network congestion, requiring solutions to cope with the increasing demand. In this paper, we address the question of network planning and design, focusing in particular on alleviation of congestion at digital-cross connect systems (DCS's) located at the nodes of a broadband transport network. We describe a constrained routing mechanism as part of a new design strategy, employing unique dynamically changing routing constraints. Also critical in network design and planning is the knowledge of DCS capacity lost to tie-trunks, when the requirement for DCS at a given node exceeds unity, and the DCS's need to De tied together to facilitate proper cross-connects. Therefore, this paper also provides a tie-trunk analysis, leading to mathematical expressions, which can act as useful guides in the judicious planning of DCS's at the nodes of the given network.*

1. Introduction

As services are increasing to exploit the large bandwidth offered by fiber, traffic on the broadband transport network is growing rapidly. Thus, incorporating current and expected high future growths in planning and designing of telecommunication networks is receiving attention more than ever before. In today's predominantly mesh-type networks where nodes are connected by fiber cables, traffic is transported from one node to another via a sequence of nodes. At each node, a digital cross-connect (DCS) system resides where the traffic destined for that node terminates, and the transit traffic arriving on one fiber is cross-connected to another fiber, depending upon its destination (see Figure 1).

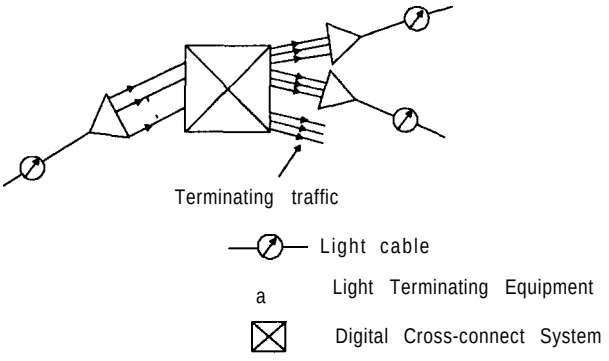

Figure 1 Traffic being cross-connected and terminated at a digital cross-connect system (DCS).

The traffic on a fiber is converted into an electric form and demultiplexed into subunits, e.g., a 2.5 Gbps optical signal on a given fiber may be converted and demultiplexed into 50 electric STS1 signals, and so on [1]. Each such subunit of traffic, whether terminating or transiting a node, utilizes two ports of a DCS - one for entering the DCS and the other for leaving the DCS system. As traffic grows and more fibers in the transport network are opened up to bring the traffic into a node, a resident DCS will experience the exhaustion of its port capacity, leading to the requirement of more than a single DCS. Eventually, depending upon the traffic pattern and the traffic growth rate in the network, several DCS's may be required at a given node to accommodate the increasing traffic and the expected future growth as part of an overall planning process. This type of a requirement leads to congestion at nodes (or central offices) of the networks.

O-8186-8538-7/98 $10.00 © 1998 IEEE

Clearly, as a given DCS at a node begins to experience port exhaustion, any new traffic should be routed around it. Figure 2 illustrates the concept of alternate routing to avoid port exhaustion. Thus, a problem that frequently arises is: ***given the current port utilization at the nodes of the network, how should the routing of transport traffic from one node to the other be altered to avoid nodes with high port utilization?***

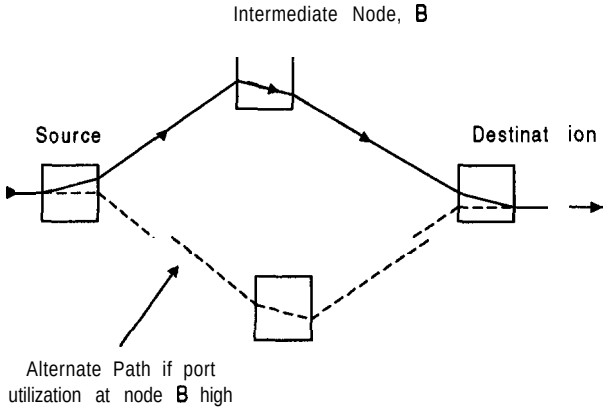

Figure 2 A pair of paths for the same source-destination; the DCS's are shown as squares without the crosses for clarity purposes.

Clearly, the aim of any network design in situations involving potential DCS's congestion is to accommodate the increasing traffic in a way that leads to equitable utilization of the DCS's in the network, thus preventing congestion in the network. In this paper, we provide a routing mechanism to design the network incrementally. This routing mechanism is discussed in Section 2. It invokes a suitable shortest path algorithm like the Dijkstra algorithm [2-3], which uses not only the true cost of the network links and nodes in the determination of the optimal route for the new traffic, but also an extra cost (penalty) assigned to the nodes. This extra penalty assigned to a node depends upon the current state of DCS port utilization at that node. While such constrained routing has been discussed before in a general way [4], the uniqueness of the method employed here is that the penalty assigned is a dynamically-changing number calculated from a special mathematical function that is driven by the current state of DCS port utilization. This method then permits studies of alleviation of DCS congestion versus cost, since any alternate route used to circumvent a congested DCS will most likely be more expensive than the standard route based also on the criterion of least (true) cost.

When the traffic at a node increases to the extent that requirement for DCS's at a node exceeds unity, the DCS's need to be connected to each other. As a result, some fraction of the total DCS's capacity is lost to tie trunks (see Figure 3). This leads to an important problem

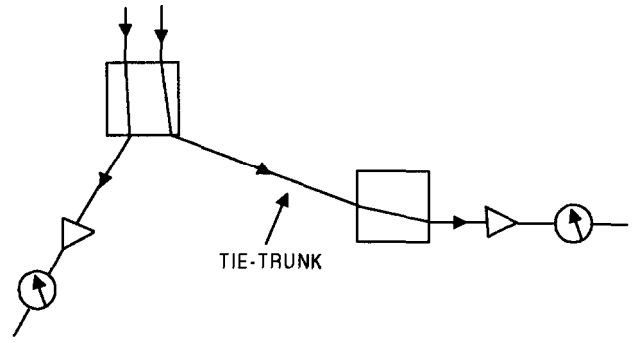

Figure 3 A tie-trunk connecting two DCS's uses up two ports.

in the design and planning of networks: ***given the traffic arriving at a node, what is the total number of DCS's required, taking into account the tie-trunks?*** In Figure 4,

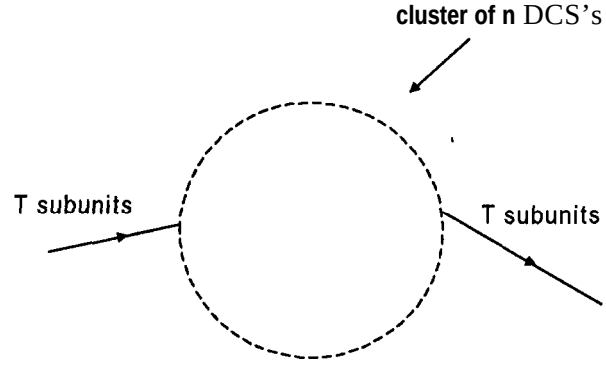

Figure 4 Traffic T entering and exiting a cluster of n DCS's.

we show T subunits of traffic arriving at a node where a cluster of n DCS's cross-connects them. Under the assumption of a symmetric distribution of traffic within the DCS's cluster, we perform an analysis in Section 3, and derive expressions for the number of DCS's required, given the total traffic at the node. The derived formulas (not given before) have proved to be useful in our studies of network planning and design, and can be useful guides to network planners and designers.

2. Constrained routing to alleviate congestion

Commonly, in the analysis and design of transport networks, a length or cost is assigned to each link and

node of the given network, and traffic is routed over the least cost route found by a suitable shortest path algorithm such as the Dijkstra algorithm [2-3]. But as traffic increases at a given node, DCS port utilization increases until a point is reached where it is judicious to divert the traffic to other underutilized nodes (DCS's) within the network. Therefore, a mechanism needs to be built in to automatically divert traffic from a node experiencing high port utilization. This mechanism then results in balancing of port utilization in the network. A special constraint function assigned to a DCS (and described in Section 2.1) facilitates this balancing of the load or port utilization in the network.

Consider the network of Figure 5, where a DCS

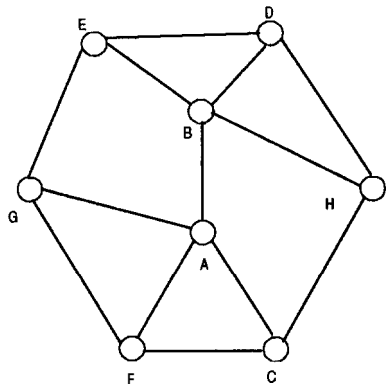

Figure 5 A network of nodes and links

resides at each node. It is convenient to split each node of the network, as in Figure 6a, to facilitate the implementation of the routing mechanism. In Figure 6a, each link incident on node M is split into two oppositely directed arcs of equal length, and the split nodes M and

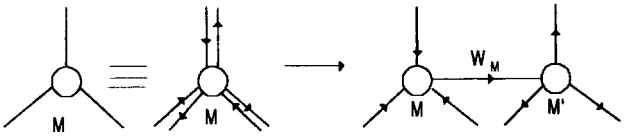

Figure 6a Splitting of a given node M, and assignment of weight W_M to the connecting arc MM'.

M' connected by arc MM' which is assigned a weight W_M. The outgoing arcs originate in the primed node M', while the incoming arcs terminate on the unprimed node M. The length (or cost) W_M assigned to the connecting arc represents the cost W_M of the node in the original network. It is composed of the actual (e.g., dollar) cost for using the node and the extra cost (penalty) w calculated from a special DCS constraint function to be described in Section 2.1. Figure 6b shows a node-split network corresponding to a part of the network of Figure 5. Path CABD in the original network now corresponds to path CC'AA'BB'DD' in the node-split network. Note that routing is always from an unprimed node to a primed node, and the total number of nodes in a given path (in the node-split network) is an even number, being equal to the total number of ports utilized by a subunit of traffic transported over the path. The cost of routing a subunit of traffic along the path CABD in the original network would be $W_C + l_{CA} + W_A + l_{AB} + W_B + l_{BD} + W_D$, where l denotes the link cost and W the cost of using a

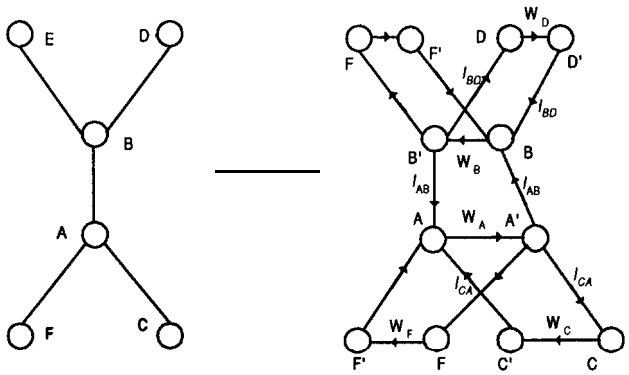

Figure 6b A part of the network of Figure 5 shown with its node-split counterpart.

node; the node cost includes the extra cost (penalty) w.

2.1 Construction of the constraint function, w

Let C denote the capacity of a DCS, which is the maximum number of ports available on the DCS. Further, let T denote the arriving traffic. Then, the number of ports utilized = 2T. Define the fraction x = 2T/C, where $0 \leq x \leq 1$. Clearly, w, the extra cost (penalty) to be assigned to a given node (or equivalently to the arc connecting the split nodes in the node-split network), should be a function of the fraction x, the fractional port utilization. Further, with increasing x, the function w(x) should increase in order to discourage further passage of the traffic through the node. This procedure of employing a constraint function then biases routing of traffic through DCS's with fewer unfilled ports, i.e., with lower values of x. Consequently, the traffic within the network will tend to spread evenly among the DCS's within the network. Clearly, the weighting function w(x) should satisfy the conditions: $w(0) = 0$ and $w(1) =$ inf, where inf is a large number that ensures complete blocking of routing through the given DCS after 2T has reached the value C. If p denotes the number of links and l_{max} the length of the longest link in the given network, we may set inf equal to p times l_{max}.

A number of functional forms of w(x) can be constructed that meet the above conditions. In our construction of a meaningful and useful form, we were guided by the fact that w(x) should be an increasing function of x, starting from the value of 0 at x = 0 and rising to inf at x=1. Further, we required that the function to be constructed be bounded by a step-like function: w(x) = 0 for $0 \le x < 1$ and w(x) = inf at x = 1. The step-like function corresponds to the scenario where routing is completely independent of the current state of port utilization (being determined solely by the network's link and (true) node costs), until the port utilization x reaches its peak value of unity at which point the node is assigned a large extra value equal to inf. We point out two functional forms that we constructed and which meet the above criteria. These are

$$w_1(x) = \inf\ (\exp(x/b) - 1)/(\exp(1/b) - 1)\ (0 \le x \le 1), \quad (1)$$

$$w_2(x) = \inf\ 2/\pi \arctan(bx/(1-x))\ (0 \le x \le 1). \quad (2)$$

Both the functions w_1 and w_2 satisfy the conditions: w_i = 0 at x = 0 and w_i = inf at x= 1, i = 1,2. Parameter b controls the shape of the functions shown in Figures 7a and b. Further, in each case, as b approaches zero, the

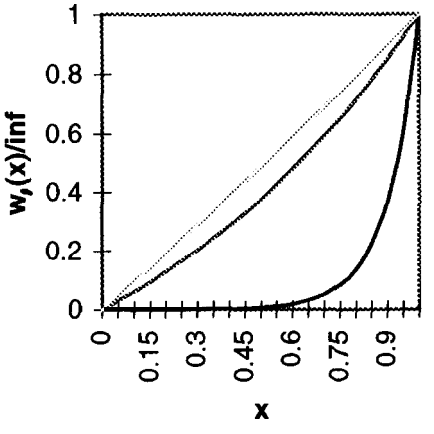

Figure 7a The function w,(x) versus x for b=10.0 (upper curve), b=1.0 (middle curve), and b=0.1 (lower curve).

function approaches the step-like function (w = 0 for $0 \le x < 1$ and w = inf for x = 1). When b increases from zero, the shape deviates from the step-like function, assuming a linear form (= inf x) in the case of $w_1(x)$ as b tends to infinity.

In the case of $w_2(x)$ (Figure 7b), the range of variation of the curve is far greater than that observed in Figure 7a. In fact, the function $w_2(x)$ is bounded by the step-like function at b = 0 and another function which is also a step-like function (equal to 0 at x=0 and equal to inf for $0 < x \le 1$) at b = infinity. As a result, we recommend the use of the function, $w_2(x)$ in network planning and design analyses. While the curve corresponding to low values of b are to be used in situations where the extra cost penalty grows steadily with increasing port utilization, the function corresponding to high values of b (for example, the curve corresponding to b = 10 in Figure 7b) is to be used, for example, for a set of special nodes in the network that need to be avoided, regardless of their state of port utilization. In general, the parameter b

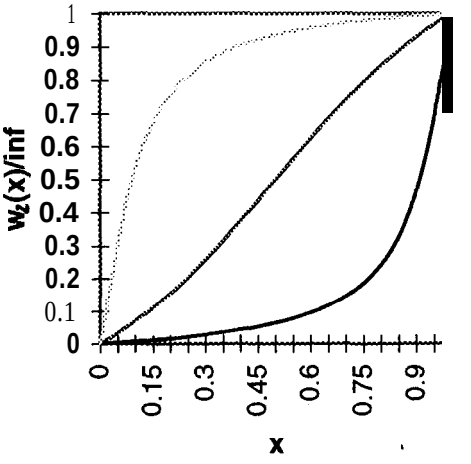

Figure 7b The function w,(x) versus x for b=10.0 (upper curve), b=1.0 (middle curve), and b=0.1 (lower curve).

enables a network designer to control the variation of the constraint function, and thus the network design.

Note that, when more than one DCS can be accommodated at a network node, then C, the capacity of a single DCS, should be replaced with the maximum capacity available in a cluster of DCS's. This maximum capacity is determined in the next section.

3. Tie-trunk analysis for a DCS cluster

Let there be n DCS's in a cluster (see Figure 4). Because some fraction of the ports on a DCS are consumed simply in tying the DCS's together (see Figure 3), the total available capacity for cross-connecting and/or terminating traffic at a node is less than nC, where C is the capacity of a single DCS. Figure 3 shows two subunits of traffic arriving at a node, one being cross-connected to a fiber via one DCS and the other via two DCS's tied together. The number of ports consumed in the latter case is clearly four, two more than in the

former, on account of a tie-trunk. In what follows, we perform an analysis that takes into account tie-trunks. The analysis yields formulas for finding the number of DCS's required, given the total traffic arriving at a node as well as for finding the maximum allowable traffic permitted at a node, given the number of DCS's at a node.

Since no DCS is to be preferred a *priori, we* make the reasonable assumption that all DCS's resident at a node are equivalent, i.e., we treat the DCS's in a cluster equitably. An equal number of subunits of traffic are input to each DCS in a cluster. If T is the total traffic

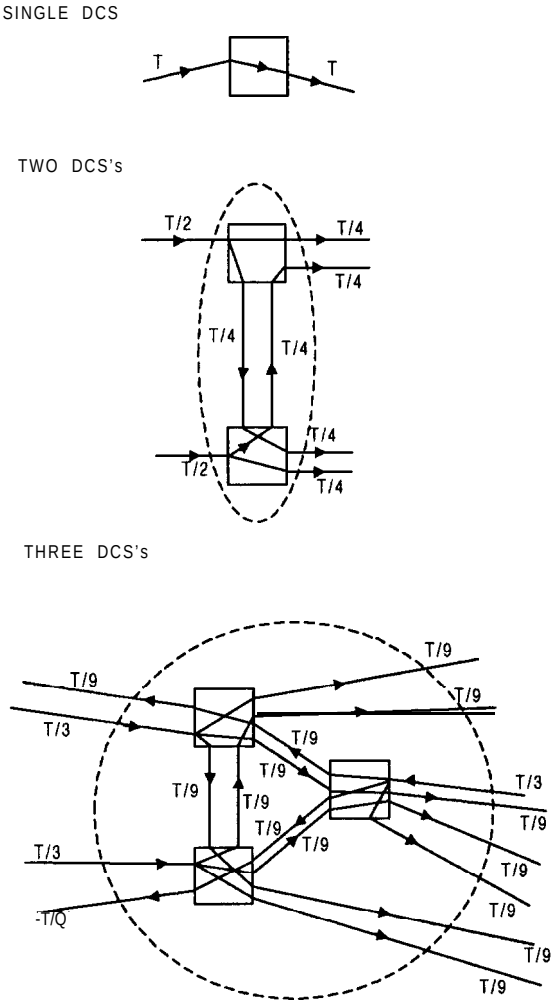

Figure 8 Traffic arrangement in a cluster of DCS's under the assumption of symmetry; T is the total traffic at the cluster.

entering the cluster (see Figure 4), traffic entering each DCS is T/n, where n is the number of DCS's in the cluster. Of the T/n traffic at a DCS, an equal amount is cross-connected through it as well as the remaining DCS's via the tie-trunks. Refer to Figure 8, where we have constructed topologies for symmetric distribution of traffic over a cluster of DCS under the above assumptions. In particular, we show T subunits arriving at a cluster of i) one DCS ii) two DCS's iii) three DCS's.

Single DCS

i) Number of input subunits = T
ii) Total number of ports utilized = input subunits + output subunits = T + T =2T.

Two DCS's

i) Number of input subunits = T/2 + T/2 = T
ii) Input subunits + Output subunits = T/2 + T/2 + T/4 + T/4 + T/4 + T/4 = 2T
iii) Total port utilization = input subunits + output subunits + 2 (T/4 + T/4) (due to tie-trunks) = 2T + T = 3T.
iv) Number of extra ports needed due to tie-trunks = T

Three DCS's

i) Input subunits + output subunits = 2T, as before.
ii) Total port utilization = (2T/3 + 2T/3 + 2T/3) + (4T/9 + 4T/9 +4T/9) (due to tie-trunks) = 2T + 4T/3 = 10T/3
iii) Extra ports needed due to tie-trunks = 4T/3

3.1 Formulas for n DCS's in a cluster

The analysis of the DCS's arrangements for the cluster of two and three DCS's in Figure 8 leads to the arrangement for n (>1) DCS's depicted in Figure 9.

Because of the symmetry, T/n subunits enter each DCS, of which a fraction l/n is cross-connected at the DCS where the traffic first arrives and the same fraction cross-connected through each of the remaining n - 1 DCS's via tie-trunks. In other words, $T/(n^2)$ subunits from each DCS's in the cluster exit to enter a neighboring DCS.

Referring to Figure 9, we see that, between each pair of DCS's, $2(T/n^2 + T/n^2) = 4T/n^2$ ports are utilized in tie-trunking. Since there are $n(n-1)/2$ pairs of DCS's in the cluster, the total number N_t of ports consumed by tie-trunks = $4T/(n^2)$ times $n(n-1)/2$, i.e.,

$$N_t = 2T\,(n-1)/n. \qquad (3)$$

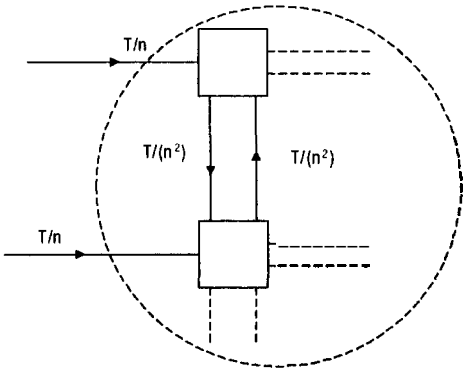

Figure 9 A cluster of n DCS's connected together; T is the total traffic at the cluster.

Adding the 2T ports for the T subunits entering and exiting the cluster, the total number N_d of ports utilized by n DCS's to cross-connect T subunits of traffic is 2T + 2T (n-1)/n = 2T (2n-1)/n, or,

$$N_d = 2T\,(2n-1)/n. \qquad (4)$$

The single, double and triple DCS cases (Figure 8) discussed earlier are verified by the above formula. The fraction $N_t/N_d = (n-1)/(2n-1)$ approaches ½ as n becomes large, i.e, 50% of the DCS capacity is used up by tie-trunks when n is large.

To find the minimum number of DCS's needed to cross-connect the T subunits entering a node, we set $N_d = nC$, where C is the capacity of a single DCS and $n > 1$, i.e.,

$$nC = 2T(2n-1)/n \qquad (5)$$

Eq. (5) is quadratic in n; its solution yields as the root:

$$n' = 2T/C(1 + \mathrm{sqrt}(1 - C/(2T)). \qquad (6)$$

Since the minimum number n_{min} of DCS's required is an integer, we obtain

$$n_{min} = 1 + \mathrm{int}\{2T/C(1 + \mathrm{sqrt}(1 - C/(2T))\}, \qquad (7)$$

where int is an operator yielding the largest integer smaller than or equal to its argument. Figure 10 shows a plot of n_{min} versus T, assuming C = 1728. To depict the effect of tie-trunks, we also show the plot without taking into account the tie-trunk analysis. Without incorporating tie-trunks, n_{min} would be 1 + int{2T/C}. From Eq. (7), one sees that, as T becomes large, n_{min} approaches int{4T/C}, which is twice as large as when the tie-trunks are absent. Again, as before, the tie-trunks consume half of the total DCS capacity available as the number of DCS's becomes large.

When the number of DCS's required at a node (or central office) increases with increasing traffic T, congestion also sets in. As a result, frequently in the network design process, the maximum number of DCS's permissible at a central office (or node) is specified. The question raised then is: what is the maximum number of T subunits permitted to arrive at a node, given the number n of DCS's? The answer is obtained from Eq. (5) above, which leads to

$$T_{max} = n^2 C/(2(2n-1)) \qquad (8)$$

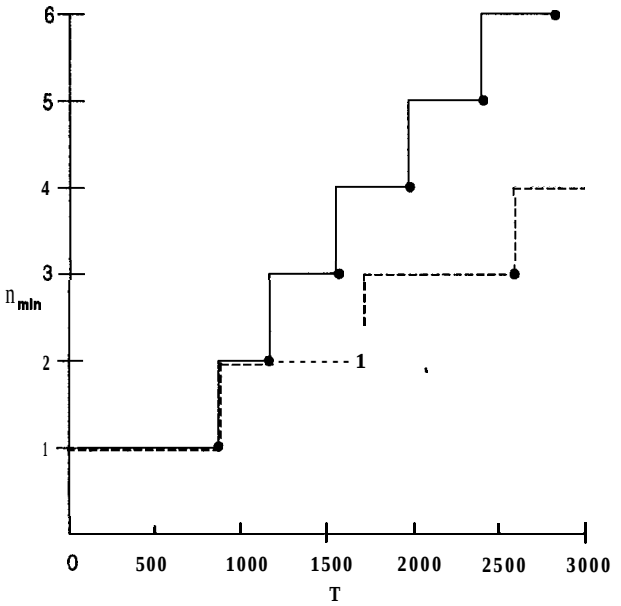

Figure 10 n_{min} versus T, with the tie-trunks (continuous lines) and without the tie-trunks (dashed lines).

Clearly, as n becomes large, $T_{max} \approx nC/4$. In the absence of tie-trunks, $T_{max} = nC/2$. Thus, the above equation indicates, as before, that half of the DCS's capacity is lost to tie-trunks. Figure 11 shows the plot of T_{max} versus n, with and without the tie-trunks.

Further Remarks: Clearly, the maximum useful capacity in a cluster of n DCS's (after deducting for consumption by tie-trunks) is $C_{max} = 2\,T_{max}$, where T_{max} is given by Eq. (8). In constrained routing (Section 2.1), if a cluster of n DCS's exists a node, the constraint function, w(x) may be defined with respect to x= n/C,.

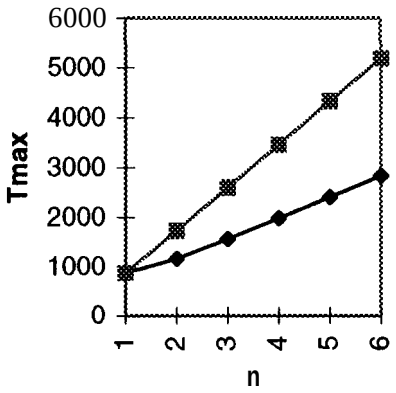

Figure 11 T_{max} versus n, with tie-trunks (lower curve) and without tie-trunks (upper curve).

If the network planning process warrants passage of more traffic than can be accommodated by the cluster of maximum permissible DCS's, i.e., T exceeds T_{max} (Eq. (8)), then the network may be augmented by the construction of express links. The express links are specifically designed to bypass the congested node. For example, in Figure 5, if node A were congested beyond rectification by constrained routing, and a significant amount of traffic was flowing between nodes G and C via node A, then congestion at node A could be alleviated by the construction of a new link GC connecting links G and C directly. Such a link is called an express link.

4. Summary

We have addressed the problems of congestion experienced by network planners and designers in digital cross-connect (DCS) transport networks. Specifically, we have suggested constrained routing using a dynamically-changing penalty factor to contain congestion at nodes. We have constructed and pointed out suitable constraint functions that can be employed in a network design process. These constraint functions are novel and have the unique feature of changing automatically in accordance with the dynamics of the network design. Further, a built-in parameter enables a network practitioner to control the behavior of the constraint function and thus the network design. When the DCS requirement exceeds one, the DCS's need to be tied together to facilitate the cross-connect process at a node. Consequently, some DCS port capacity is lost to tie-trunks. Knowledge of consumption of tie-trunks is crucial in any accurate network design and long-term planning process. We therefore derive expressions (not given before) relating the number of DCS's required with the total traffic arriving at a node and vice versa (the maximum traffic permissible, given the number of DCS's at a node).

REFERENCES

[1] M. Chow, *Understanding SONET/SDH*, Andan Publishers, New Jersey (1995).

[2] E. W. Dijkstra, "A Note on Two Problems in Connexion with Networks", Numer. Math. 1, 269-271 (1959).

[3] M. Gondran and M. Minoux, *Graphs and Algorithms*, John Wiley and Sons (1990).

[4] A. Kershenbaum, *Telecommunications Network Design Algorithms*, McGraw-Hill (1993).