TOPOLOGICAL DESIGN OF INTERCONNECTED LAN-MAN NETWORKS*

Cem Ersoy *Shivendra S. Panwar*

Department of Electrical Engineering
Polytechnic University, Brooklyn, New York 11201

Abstract

This paper describes a methodology for designing interconnected LAN-MAN networks with the objective of minimizing the average network delay. We consider IEEE 802.3-5 LANs interconnected by transparent bridges. These bridges are required to form a spanning tree topology. The optimization algorithm for finding a minimum delay spanning tree topology is based on simulated annealing. In order to measure the quality of the solutions, we find a lower bound for the average network delay. The comparison of our results with this lower bound and several other goodness measures show that the solutions are not very far from the global minimum. We extend our algorithm for finding minimum delay LAN-MAN topologies consisting of FDDI MANs or Switched Multi-Megabit Data Service (SMDS) interconnecting several clusters of bridged LANs.

1 Introduction

This paper describes a heuristic algorithm for designing interconnected LAN-MAN networks with the objective of minimizing the average network delay. We consider IEEE 802.3-5 LANs interconnected by transparent bridges described in IEEE Standard 802.1 Part D [1]. These bridges are widely used for interconnecting CSMA/CD LANs and they can also be used for other types of LANs. The need for covering a large area and many nodes on LANs is fulfilled by these bridges [2]. No participation by an end station is necessary for making use of the services provided by transparent bridges. These bridges are self-learning and self-configuring. They learn the location of end stations by observing source addresses. They route frames by comparing the destination addresses to the table of learned addresses. Given any arbitrary topology, bridges configure themselves to be part of a deterministic active spanning tree topology [3]. They adapt to changes in the topology, such as failures, and the movement of stations without network management intervention.

The performance of bridged CSMA/CD LANs has been studied in [4,5,6,7,8]. The problem of determining which gateways to use to interconnect existing data networks has been discussed in [9,10]. Simulated annealing technique has been used for designing minimum cost interconnected CSMA/CD LANs in [11]. The design of optimally locating bridges and repeaters for minimizing the average delay and fast algorithms for special cases have been studied in [12].

We address the problem of designing minimum delay spanning tree topologies for interconnecting LANs given the traffic requirements. This problem is a difficult combinatorial optimization problem because of many local minima. The optimization method for finding the minimum delay spanning tree topology is based on simulated annealing, since this method works well on problems with many local minima [13]. Although simulated annealing heuristics can find the global optimum, this may require infinite number of transitions [13,14]. Finite-time implementations of the algorithm approximate an optimal solution. In order to measure the quality of our solutions, we find a lower bound for the average network delay corresponding to the given set of traffic requirements. We compare the results of the optimization algorithm with this lower bound. We use other goodness measures for showing the quality of the solutions.

We also consider the LAN-MAN interconnection problem. FDDI LANs are already used as backbone MANs for the interconnection of clusters of bridged LANs. The IEEE 802.6 DQDB (Distributed Queue

*This work was supported by the New York State Science and Technology Foundation's Center for Advanced Technology in Telecommunicatons, Polytechnic University, Brooklyn, New York and by the NSF under grant NCR-8909719

 CH3133-6/92/0000-2260 $3.00 © 1992 IEEE

Dual Bus) MAN is a different alternative backbone network. Another possibility for the interconnection of bridged LAN clusters is the Switched Multi-Megabit Data Service (SMDS) which is to be offered by the Local Exchange Carriers in the early 1990s [15]. We describe an extended version of the algorithm for finding minimum delay LAN-MAN topologies.

The remainder of this paper is organized in the following way: In Section 2, we describe the problem and our assumptions. Section 3 covers the design of interconnected LAN networks. We describe the heuristic algorithm based on simulated annealing and the calculation of the network delay in Sections 3.1 and 3.2. We find a lower bound for the average network delay corresponding to a given set of traffic requirements in Section 3.3. In Section 4, we extend our methodology to LAN-MAN networks. Results and several goodness measures are given in Section 5. Section 6 is the conclusion.

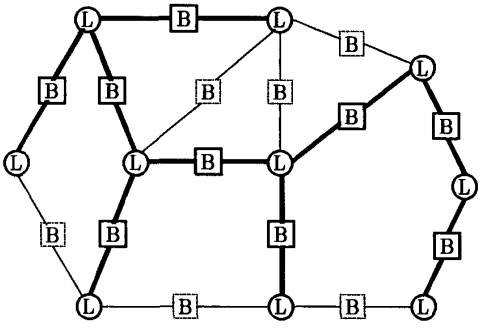

Figure 1: An example of a bridged LAN network.

2 Definition and Formulation of the Problem

An example physical topology for a group of bridged LANs is shown in Fig. 1. In order to avoid looping of packets, transparent bridges require the active topology to be a spanning tree. One of the possible active spanning trees is shown with bold lines. During the normal operation, only the bridges on the active tree forward packets. Other bridges remain idle unless activated to form a new tree in the case of failures. In our problem, we are given N LANs and the traffic requirements between all LAN pairs. Our goal is finding the spanning tree topology with minimum average delay for this set of requirements. We do not have any constraint on where to place the bridges. In other words, we want to find where to place $N-1$ bridges among all possible LAN pairs such that they form a spanning tree with minimum delay. After finding the minimum delay tree, bridges can be set-up to choose this tree as the active topology by adjusting the parameters of the self-configuration algorithm [3].

In Fig. 1, LANs are represented as nodes and bridges are represented as branches of a graph. Since we are more interested in the shape of the minimum delay topology, we assume that all bridges have two ports. Many of the existing bridges are two-port bridges, but we can also model a multi-port bridge as several branches and a central node. In that case, branches in the graph will correspond to bridge ports rather than whole bridges. In that way, both two-port and multi-port bridges can be used for implementing minimum delay spanning tree topologies.

The problem of finding the minimum delay spanning tree topology is formulated as described below:

$$\min_{\{x_{ij}\} \in X} T = \sum_{i=1}^{N} Delay(\text{LAN } i) + \sum_{i=1}^{N} \sum_{j=1}^{N} Delay(\text{Bridge btw } i \text{ and } j)\, x_{ij}$$

$$x_{ij} = \begin{cases} 1, & \text{if there is a bridge between LAN } i \text{ and LAN } j \\ 0, & \text{otherwise} \end{cases}$$

$$X \text{ is the set of all spanning trees.} \qquad (1)$$

In the above formulation, a combination of LAN and bridge delays, T, is minimized over the design parameters x_{ij}'s. N is the total number of LANs. The objective function implies LAN and bridge capacity constraints, since the delay T becomes infinity if the flow values exceed the capacities. In order to keep the problem simple, we do not consider any other constraints. If needed, other constraints, such as cost, maximum number of bridges (i.e. hops) between any two LANs and maximum utilization for LANs and bridges, can easily be incorporated into the formulation and the algorithm. Furthermore, dollar cost constraints may not play an important role in the design of interconnected LANs. For example, in the case of using identical local bridges, every active spanning tree topology will have exactly $N-1$ bridges or bridge ports, resulting in the same dollar cost for different topologies.

Even this simple formulation is a difficult combinatorial optimization problem. Our experiments with greedy local search heuristics showed that the solution space has many local minima. Since simulated annealing algorithms do not get stuck in local minima [13], our searching heuristic is based on this method. During the search for a minimum delay network, we create a sequence of topologies according to the simulated annealing algorithm. We model each topology as a network of queues. We find the average network delay for each topology. The algorithm and the delay analysis will be described in the following section.

3 Design of Minimum Delay Interconnected LANs

3.1 Simulated Annealing

Annealing is known as a thermal process for obtaining low energy states of a melted solid by slow cooling. In 1953, Metropolis *et al.* introduced a simple algorithm for simulating the annealing process [16]. In 1982, Kirkpatrick *et al.* applied the Metropolis algorithm to a combinatorial optimization problem [17]. The simulated annealing algorithm has been applied to many areas of combinatorial optimization including graph theory and networking [13].

Simulated annealing is a local neighborhood search heuristic technique. Basic disadvantages of greedy type local search algorithms are that they may get stuck in local minima because they accept only cost improving solutions and the quality of the final result heavily depends on the initial solution. On the contrary, simulated annealing algorithms occasionally accept deteriorations in cost in a controlled manner besides accepting improvements in cost. This property enables them to escape from local minima while keeping the favourable features of local search algorithms, i.e. simplicity and general applicability.

As in other local search algorithms, we have a neighborhood structure and a generation mechanism for it. During the local neighborhood search for a minimum, if the transition from solution i to solution j is a cost-decreasing one, it is always accepted; if the transition is a cost-increasing one, solution j is accepted as the current solution with a certain probability, p, which is given by

$$p = \exp\left(\frac{-[Cost(j) - Cost(i)]}{c}\right), \qquad (2)$$

$$Cost(j) > Cost(i).$$

Here, c is the control parameter and regulates the probability of accepting a cost-increasing solution. At the beginning, a large value for the control parameter is chosen, resulting in the acceptance of most of the transitions. During the search, we slowly reduce the control parameter towards zero, similar to the behaviour of the temperature in physical annealing. Lower control parameter values make the acceptance of cost-increasing transitions more difficult.

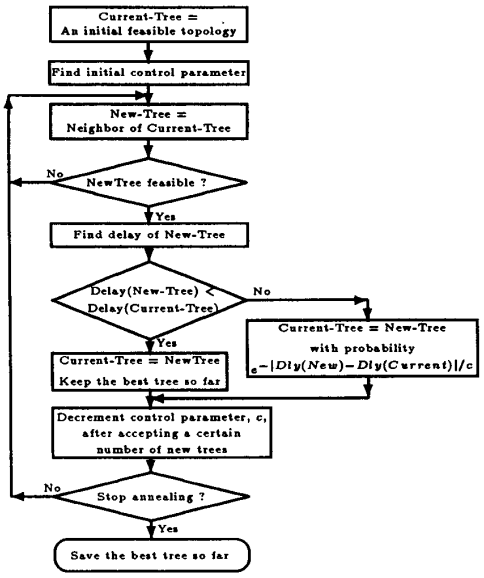

Figure 2: The flowchart for the simulated annealing algorithm.

Fig. 2 shows the flowchart of the annealing algorithm used for finding minimum delay topologies. In general, simulated annealing algorithms are defined by a neighborhood structure and a cooling schedule. The *neighborhood structure* for our problem is defined as follows: Any two spanning tree topologies which have all the branches except one branch common are called neighbor trees. Given a spanning tree, we can create a neighbor tree by removing a branch resulting in two separate sub-trees, and adding another branch which will connect the two sub-trees, but will not create a loop. During the search for a minimum delay topology, we create a sequence of neighbor trees according to the cooling schedule. We check the feasibility of each topology by comparing the flows of LANs and bridge ports with their capacities. If the topology is feasible, we find the average network delay as

described in Section 3.2.

It has been shown in [13,14] that the simulated annealing algorithm finds the global optimum. Unfortunately, this implementation requires an infinite number of transitions. A finite-time simulated annealing algorithm for finding high quality solutions can be implemented with a suitable cooling schedule. A *cooling schedule* consists of choosing an initial value for the control parameter, c_0; the method for decrementing the control parameter; a finite number of transitions at each value of the control parameter; and the stopping criterion. Different cooling schedules have been proposed in [13,14,17]. We experimented with different cooling schedules and chose one similar to the one described in [13], because of its simplicity and efficiency.

The Cooling Schedule

1. **Chosing the Initial Value of the Control Parameter**

   As already mentioned, the initial value of the control parameter, c_0, is chosen so that almost all new topologies will be accepted. In order to achieve this, we create an initial feasible topology and random transitions by finding neighbor topologies. $\overline{\Delta d}$ is the average of the increase in delay in 50 transitions. c_0 is chosen such that

   $$InitialAcceptanceRatio \approx \exp\left(\frac{-\overline{\Delta d}}{c_0}\right) \approx 0.99 \tag{3}$$

2. **Decrementing the Control Parameter**

   The function used for decrementing the control parameter is given by

   $$c_{k+1} = \alpha \cdot c_k \qquad k = 1, 2, \ldots \qquad \alpha \approx 0.8 \tag{4}$$

   The control parameter will be decreased after acceptance of some fixed number of new topologies. However, since transitions are accepted with decreasing probability, the number of topologies examined at each c_k will increase while c_k goes to zero. In order to avoid extremely long iterations at small values of c_k, the total number of topologies examined at each c_k are bounded by a fixed maximum value. This value is comparable to the size of the neighborhood.

3. **Stopping Criterion**

   Simulated annealing is terminated if the value of the delay does not change after decrementing the control parameter a fixed number of times. This fixed number is chosen such that the algorithm has a sufficiently large probability of visiting at least a major part of the neighborhood of a given solution. In order to guarantee that we are not missing any good solutions in the neighborhood of the annealing solution, we terminate the algorithm by comparing the annealing solution with all of its neighbor topologies.

3.2 Delay analysis

We find the average network delay for each topology generated at every iteration of the simulated annealing algorithm. Because of the large number of topologies generated, we need an efficient way of approximating the average network delay. We consider delays due to LANs and bridges. The topology is modelled as a network of queues. Different measurement and performance studies have shown that LAN traffic is bursty and consists of batches or trains of packets because of the existing protocols [18,19]. In order to account for the burstiness of the traffic and to have a computationally inexpensive measure, we use $M^X/M/1$ queues with batch Poisson arrivals. Although it is simple, $M^X/M/1$ queueing model may achieve an acceptable fit for the busiest periods of the network [19] and give us the relative performance of different topologies. More complex delay models can be used with the algorithm, but they will increase the running times.

We are given the mean rate for traffic requirements between each LAN pair, t_{ij}, in terms of batches per unit time. For a given topology, we can find the mean batch flow values of LANs and bridges, λ_i and λ_{ij}, respectively. We know the capacities of LANs and bridges, C_i and C_{ij}, respectively. We assume exponential distribution for packet lengths with mean l and a geometric distribution for the number of packets in a batch with mean X. The mean service rate for LANs, μ_i, is the LAN capacity divided by the mean packet length. The mean service rate for each direction of the bridges, μ_{ij}, is equal to $1/C_{ij}$. The expected number of packets, $E[L]$, in each queue is found in pages 156-160 of [20]. Using Little's formula, $E[T] = E[L]/\lambda X$, the expected value of the delay, $E[T]$, for each $M^X/M/1$ queue is given as

$$E[T] = \frac{\rho}{\lambda(1-\rho)} \qquad \text{where} \quad \rho = \frac{\lambda X}{\mu}. \tag{5}$$

After finding the delays caused by LANs and bridges,

the average network delay, T, is found by using

$$T = \frac{1}{\gamma}\left(\sum_{i=1}^{N}\lambda_i X E[T_i] + \sum_{i=1}^{N}\sum_{j=1}^{N} x_{ij}\lambda_{ij} X E[T_{ij}]\right) \quad (6)$$

as in [21], where $E[T_i]$ is the expected delay in LAN i; $x_{ij} = 1$ if there is a bridge between LAN i and LAN j, and $x_{ij} = 0$ otherwise; $E[T_{ij}]$ is the expected delay on the bridge port from LAN i to LAN j; γ is the total input flow into the network given by $\gamma = X\sum_{i=1}^{N}\sum_{j=1}^{N} t_{ij}$.

3.3 Lower Bound for the Average Network Delay

In order to measure the quality of our solutions, we found a lower bound for the minimum delay corresponding to a given set of traffic requirements. This is not a very tight bound, but is still useful for showing that our solutions are not very far from the global minimum. In this section, we describe the procedure for finding the lower bound for the case where all LANs have equal capacity and all bridges have equal capacity and the traffic requirements are symmetric, i.e. $t_{ij} = t_{ji}$. We will also indicate a procedure for the case of unequal capacities. The procedure for the asymmetric requirements case is more involved and it will not be described here. In order to find a bound for the delay, we start with finding a lower bound for the total flow on all LANs, λ_L^T, and total flow on all bridge ports, λ_B^T. For the equal capacity case, the lower bound for the delay due to LANs is found by distributing λ_L^T as uniformly as possible on all LANs. The bound for the delay due to bridges is found similarly. The overall procedure is as follows:

1. Any traffic requirement which has LAN i as its source or destination has to pass through LAN i. The *mandatory* traffic on each LAN is given by

$$\lambda_M^i = \sum_{k=1}^{N}(t_{ik} + t_{ki}) - t_{ii}, \qquad \forall i. \quad (7)$$

2. We have $N(N-1)$ inter-LAN requirements. Some of these requirements have to be routed over one or more intermediate LANs. This creates transitional traffic. We want to minimize this transitional traffic. Since a spanning tree topology has $(N-1)$ branches, we can directly route *at most* $(N-1)$ requirement pairs, $(t_{ij} + t_{ji})$. We can lower-bound the transitional traffic by assuming that the $(N-1)$ largest requirement pairs are directly connected and they do not create any transitional traffic on LANs.

3. The remaining $(N-1)(N-2)$ smaller requirements have to pass through *at least one* intermediate LAN other than their source and destination (i.e., these requirements will pass through at least 3 LANs). Therefore, these requirements create an extra transitional traffic, t_{TR}.

   In order to minimize the delay due to LANs, we try to distribute this transitional traffic such that the flow on each LAN will be the same or as close as possible. Therefore, total and average LAN flows, λ_L^T and λ_L, and the transitional traffic, t_{TR} for the lower bound are given by

$$\lambda_L^T = 2\cdot\sum_{(i,j)\in S_1}(t_{ij} + t_{ji}) + 3\cdot t_{TR} \quad (8)$$

$$\lambda_L = \frac{1}{N}\lambda_L^T \quad (9)$$

$$t_{TR} = \sum_{(i,j)\in S_2}(t_{ij} + t_{ji}) \quad (10)$$

$$S = \{(i,j)|\ i \neq j\}$$

$$S_1 = \{(i,j)|\ i \neq j \text{ and } (t_{ij} + t_{ji}) \text{ are } (N-1) \text{ largest sum of requirement pairs}\}$$

$$S_2 = S - S_1.$$

   If the largest λ_M^i is less than λ_L, all LANs will be assigned the same flow λ_L. If the LAN with the largest λ_M^i has already more mandatory traffic than λ_L, we do not add any transitional traffic to that LAN. We distribute t_{TR} among the other $(N-1)$ LANs such that all of them will have the same flow. If the second largest λ_M^i is higher than this traffic, we skip the second LAN and try to distribute t_{TR} among the $(N-2)$ LANs. We continue in this way until we spread the combination of mandatory and transitional traffic as uniformly as possible among all LANs.

   We use the following approach if LANs have different capacities: We find the total traffic similarly and distribute this traffic to different LANs as in the well-known capacity assignment problem. However, in our case, capacities are given and the portions of traffic are assigned. We can prove analytically that the described distribution of total LAN flow gives the minimum delay.

4. Bridge flows can be calculated similarly. In order to minimize the effect of transitional traffic,

the $(N-1)$ largest requirement pairs can be carried by bridges directly connecting their sources and destinations. The rest of the requirements have to pass through at least two bridges. Similar to LAN flows, we spread the total bridge flow among $2(N-1)$ bridge ports as uniformly as possible. Therefore, total and average bridge port flows, λ_B^T and λ_B, for the lower bound is given by

$$\lambda_B^T = \sum_{(i,j)\in S_1} (t_{ij} + t_{ji}) + 2 \cdot t_{TR} \quad (11)$$

$$\lambda_B = \frac{1}{2(N-1)} \lambda_B^T \quad (12)$$

If the bridge which was assigned the largest t_{ij} has already higher flow than λ_B, we do not add any transitional traffic to the ports of that bridge. We distribute $2t_{TR}$ among the other $2(N-2)$ bridge ports such that all of them will have the same traffic if possible. If not, and the second largest t_{ij} is also higher than λ_B, we skip the second bridge and try to distribute $2t_{TR}$ among the $2(N-3)$ bridge ports. This procedure is continued until we spread the combination of direct and transitional traffic as uniformly as possible among all bridges. If bridges have different capacities, flows can be found as in the case of LANs.

5. After finding the flow values for LANs and bridge ports, we calculate the delay for N LANs and $2(N-1)$ bridge ports and thus find a lower bound on the average network delay.

4 Design of interconnected LAN-MAN Networks

There is a growing interest in the interconnection of geographically dispersed LANs or bridged LAN clusters by Metropolitan Area Networks (MANs). FDDI and IEEE 802.6 DQDB MANs are promising candidates as backbone MANs interconnecting LAN clusters. There are also proposals of broadband services such as Switched Multi-Megabit Data Service (SMDS) for interconnecting LAN clusters using MAN technology [15].

Fig. 3 shows an example interconnected LAN-MAN network with several clusters. In this section, we will extend the simulated annealing algorithm for the design of minimum-delay LAN-MAN networks. We assume that we know which LAN belongs to which

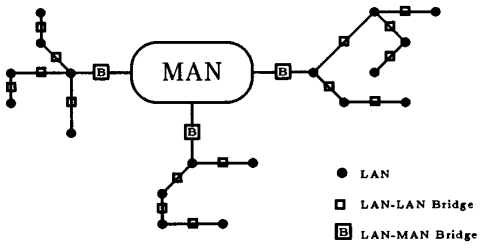

Figure 3: An example LAN-MAN topology.

cluster. These clusters can be formed because of geographical, network management or security reasons. We know the traffic requirements between all LAN pairs. We want to find a minimum-delay LAN-MAN topology. As in the bridged LAN case, we do not have any constraints other than capacity constraints. Again, we consider interconnected LANs with transparent bridges. Overall topology has to be a spanning tree because of the bridges. We assume that we have only one MAN. This MAN can be FDDI, IEEE 802.6 DQDB or instead of the MAN we can have the SMDS as a backbone network. This backbone is modeled as a central node. One approach to find a good LAN-MAN topology is solving the problem as if it were a LAN-LAN problem with many nodes and adding an additional constraint forcing LANs in different clusters to be interconnected through the MAN. Increasing the number of LANs increases the running time of the simulated annealing algorithm quickly. We will use an approach of decomposing the problem into smaller problems for each cluster and hence reduce the running time. In the first phase, we assume that each cluster is connected to the MAN with only one bridge. This is a reasonable assumption because LAN-MAN bridges are more expensive than ordinary bridges. There may also be privacy reasons to use only one LAN-MAN bridge. In the case of using more than one LAN-MAN bridge, some of the intra-cluster traffic has to pass through the MAN, because of the spanning tree requirement. As shown in Fig. 4, there will be seperate spanning trees connected through the MAN in each cluster. This may not be desirable for security reasons.

The problem is suitable for decomposition because the intra-cluster traffic does not effect the other clusters. Therefore, the topology of each cluster can be determined by considering only the intra-cluster traffic and the traffic between the MAN and that cluster. At the begining, we are given an overall traffic requirements matrix, $[t_{ij}]_{N\times N}$, for the whole network.

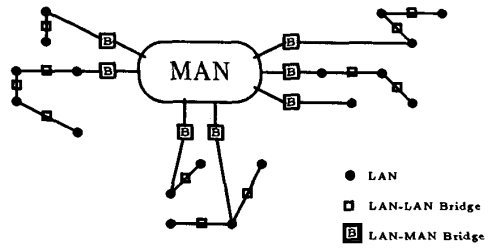

Figure 4: An example topology with more than one LAN-MAN bridge per cluster.

We can calculate smaller traffic requirement matrices $[t^k_{ij}]_{(N_k+1)\times(N_k+1)}$ where N_k is the number of LANs in cluster k. For example, if the original traffic requirements for the topology in Fig. 3 are given by

$$[t_{i,j}] = \left[\begin{array}{c|c|c} \begin{matrix} t_{1,1} & \cdots & t_{1,6} \\ \vdots & \ddots & \\ t_{6,1} & & \end{matrix} & \begin{matrix} t_{1,7} & \cdots & t_{1,11} \\ \vdots & \ddots & \\ & & \end{matrix} & \begin{matrix} t_{1,12} & \cdots & t_{1,18} \\ \vdots & \ddots & \\ & & \end{matrix} \\ \hline \begin{matrix} t_{7,1} & \cdots & \\ \vdots & \ddots & \\ t_{11,1} & & \end{matrix} & \begin{matrix} t_{7,7} & \cdots & \\ \vdots & \ddots & \\ & & \end{matrix} & \begin{matrix} t_{7,12} & \cdots & \\ \vdots & \ddots & \\ & & \end{matrix} \\ \hline \begin{matrix} t_{12,1} & \cdots & \\ \vdots & \ddots & \\ t_{18,1} & & \end{matrix} & \begin{matrix} t_{12,7} & \cdots & \\ \vdots & \ddots & \\ & & \end{matrix} & \begin{matrix} t_{12,12} & \cdots & \\ \vdots & \ddots & \\ & & \end{matrix} \end{array}\right]$$

we can calculate the traffic matrix, $[t^1_{i,j}]$, for cluster 1 as follows:

$$[t^1_{i,j}] = \left[\begin{array}{ccc|c} t^1_{1,1} & \cdots & t^1_{1,6} & t^1_{1,MAN} \\ \vdots & \ddots & & \vdots \\ t^1_{6,1} & & & t^1_{6,MAN} \\ \hline t^1_{MAN,1} & \cdots & t^1_{MAN,6} & 0 \end{array}\right] \quad (13)$$

where

$$t^1_{i,MAN} = \sum_{l=7}^{18} t_{i,l} \quad \text{and} \quad t^1_{MAN,i} = \sum_{l=7}^{18} t_{l,i}, \quad \forall i. \quad (14)$$

After finding the new traffic requirement matrices for each cluster, we can solve each sub-problem independently. At the beginning, we allow only one LAN-MAN bridge per cluster. We use the simulated annealing algorithm as in the LAN-LAN problem with the additional constraint that each cluster will be connected to the MAN node with only one bridge. After finding the overall topology, we calculate the *maximum cluster access delay* T^k_{CA} for each cluster. T^k_{CA} is the maximum delay from any LAN in cluster k to the MAN. The maximum end-to-end delay in the overall network can be calculated as a summation of the two largest T^k_{CA} and the delay due to the MAN, T_{MAN}. The *LAN access delay* between LAN i and the MAN, T^i_{LA}, is the summation of the delays of all LANs p and bridges (r, s) lying on the path from LAN i to the MAN, $P(i, MAN)$, and is given by

$$T^i_{LA} = \sum_{p \in P(i,MAN)} E[T_p] + \sum_{(r,s) \in P(i,MAN)} E[T_{rs}] \quad (15)$$

$$T^k_{CA} = \max_{i \in \text{cluster } k} T^i_{LA}. \quad (16)$$

If T^k_{CA} exceeds a threshold for a cluster, we proceed to the second phase: We allow that cluster to have two LAN-MAN bridges. Because of the spanning tree requirement, that cluster will have two sub-trees as shown in Fig. 4. Since this is likely to reduce the depth of the tree, new T^k_{CA} will be lower. As we have explained before, we can change the topology of one cluster without effecting the other clusters. We find the minimum-delay topology for that cluster as in the first phase, but this time we constrain the maximum number of LAN-MAN bridges to two. If T^k_{CA} drops below the threshold, we stop. Otherwise, we allow the cluster to have one more LAN-MAN bridge. We continue increasing the number of LAN-MAN bridges until T^k_{CA} drops below the threshold. We repeat this procedure for all clusters. As a result, some of the clusters might have one LAN-MAN bridge, some of them might have more depending on their traffic requirements.

Another approach could be using cost constraints to determine the number of LAN-MAN bridges, but this would increase the complexity of the problem and currently high LAN-MAN bridge prices would force us to use as small as possible number of them. In many practical problems, one LAN-MAN bridge per cluster is sufficient and the overall minimum-delay topology is found in one phase.

A variation of the problem occurs in the case of using SMDS for interconnecting the clusters of LANs, since SMDS will provide communications at DS1 (1.5Mbps) or DS3 (45Mbps) rates [15]. During the design process, we have to decide which rate of service and, in the case of DS3, which access class (4, 10, 16, 34Mbps) will be used for each cluster. A possible approach can be described as follows: At first, we choose the lowest and the cheapest rate among all SMDS access classes for all clusters. We find the minimum delay cluster topologies and calculate T^k_{CA}. If it

Table 1: Results for the minimum delay interconnected LAN problem

# of LANs	T_{min} Annealing	T_{max} Annealing	$T_{Lower\ Bound}$	T_{min} Local Search	T_{min} 10000 samples	T_{mean} 10000 samples	CPU Time PC-AT / CONVEX
6	6.406*	6.406*	4.613	—	—	—	8 s / < 1 s
7	7.524*	7.746	5.205	—	—	—	18 s / < 1 s
10	8.323	8.524	5.657	8.639	9.103	11.445	52 s / < 1 s
15	10.470	10.696	6.703	11.117	12.419	15.993	6.2 min / 4.3 s
20	13.910	14.370	8.355	14.832	16.655	21.937	25.8 min / 18 s
30	17.233	18.112	9.029	19.374	22.166	26.263	181 min / 124 s

*: Global minimum in complete enumeration. All delay values are in milliseconds.

is acceptable, we stop; otherwise we choose the next higher SMDS access class for clusters with unacceptable T_{CA}^k. We continue increasing the SMDS rates for each cluster until we are satisfied with T_{CA}^k. If increasing the rate of SMDS does not reduce T_{CA}^k, we can also increase the number of LAN-SMDS bridges connected to a cluster as explained before.

5 Results and Discussion

5.1 Methodology for the Experiments

We performed experiments on networks with 6, 7, 10, 15, 20 and 30 LANs. For 6 and 7 LAN problems, we enumerated all possible spanning tree topologies and found the global minimum. This enabled us to compare the solution of the simulated annealing directly with the global minimum. For larger problems, we used several different goodness measures. In order to estimate the range for the simulated annealing results, we ran the algorithm with different random seeds on the same problem. We compared our solutions with the lower bound described in Section 3.3. The simulated annealing algorithm is a local search heuristic. In order to see its advantages over a standard greedy local search , we implemented a local search heuristic and run it several times with different random initial topologies. We compared the best topology found by the local search algorithm with the simulated annealing results. We also used the goodness measure described in [11]. In this measure, 10,000 random feasible topologies were generated. A histogram corresponding to the delay of these topologies was created. This was then compared with the simulated annealing solutions.

Different patterns used for the traffic requirements are as follows:

- Three of the traffic matrices consist of uniformly distributed random traffic requirements with different average values. These average values correspond to light, medium and heavy loads. In order to consider the unbalanced inter-LAN traffic patterns, some rows and columns of the traffic matrices have larger average values than the others. These rows or columns might correspond to LANs connected to file servers or host computers.
- One of the traffic matrices is such that the traffic between any two LANs decreases linearly with the "distance" between them. For example, the traffic requirements between LAN 1 and LAN 2 are larger than those between LAN 1 and LAN 5. All traffic requirements have deterministic values according to the "distance" measure.
- The last traffic matrix is uniform. All traffic requirements have the same deterministic value.

The mean packet length, l, is equal to 192 bytes as measured in [19]. The average number of packets in a batch, X is 8. The capacity of LANs is 10 Mbits/sec which is the standard for CSMA/CD LANs. The capacity of LAN-LAN bridges is 6,000 packets/sec and the capacity of LAN-MAN bridges are 10,000 packets/sec for these 192-byte long packets. The threshold for T_{CA}^k in the LAN-MAN problem is 20 msec.

5.2 Results for the Interconnected LAN Problem

Table 1 summarizes the results for the interconnected LAN problem in the case of medium load random traffic requirements. As an example for the medium load, the requirement matrix for the 15 LAN problem consists of uniformly distributed random requirements with an average of 3 batches per second. Two rows, corresponding to LANs with file servers, have higher requirements with average 10 batches per second. Corresponding minimum delay topologies found

by the algorithm are such that average utilization of LANs and bridges are 26% and 11%, respectively. The minimum and maximum delay values for the simulated annealing algorithm are found by running the algorithm with 10 different random initial topologies. The small range of the results shows that the final solution is not dependent on the initial topology. For small problems, the algorithm found the global minimum most of the time. It found slightly higher delay topologies the rest of the time. Simulated annealing results are in the range of 39% to 90% of the lower bound. As we have mentioned before, the lower bound is not very tight. It can be observed in Table 1 that the values of the lower bound for 6 and 7 LAN problems are 39% and 45% smaller than the global minimum values, respectively. Therefore, we conjecture that the annealing solutions are not very far from the global minimum values.

The simulated annealing algorithm outperformed multiple runs of the greedy local search algorithm in all cases studied. Comparison of the simulated annealing results with the best solution found by running the greedy local search algorithm with different random initial topologies show that the simulated annealing algorithm does not get stuck in a local minimum.

Related columns of Table 1 show that the simulated algorithm finds better topologies than the best of 10,000 randomly generated topologies. Histograms for the delay of these random topologies show that most of them have significantly higher delays than the simulated annealing solutions.

The last column in Table 1 shows the CPU times for the simulated annealing algorithm on a PC/AT and CONVEX 120 mini-supercomputer. For smaller problems, the algorithm finds high quality solutions very quickly even on a personal computer. For larger problems, running times are still reasonable.

5.3 Results for the Interconnected LAN-MAN Problem

Table 2 shows the results for two LAN-MAN problems. The first problem has three small clusters as in the example shown in Fig. 3. Because of the small size of the clusters, we could enumerate all possible topologies and found the global minimum. As shown in the table, the simulated annealing algorithm found the global optimum for each cluster. Since the overall topology is a combination of these cluster topologies, it is also globally optimal for the first problem. The short overall running time for the first problem shows the advantage of decomposing the problem. If we had approached the problem as an 18 LAN problem without decomposing it, the running time would have been around 20 minutes instead of 67 seconds.

Table 2: Results for the LAN-MAN problem

	# of LANs	T_{min} Annealing	T_{max} Annealing	Cluster Access Delay	CPU Time PC-AT
Clust 1	6	8.421*	8.480	14.643	22 s
Clust 2	5	7.156*	7.352	14.417	9 s
Clust 3	7	8.572*	8.674	15.553	36 s
Overall	18	9.483*	9.667	—	67 s
Clust 1	6	8.375*	8.437	13.948	23 s
Clust 2	12	10.929	11.258	17.663	229 s
Clust 3	7	8.430*	8.768	14.177	34 s
Clust 4	10	9.555	9.780	16.402	86 s
Overall	35	13.069	13.349	—	372 s

*: Global minimum in complete enumeration.
All delay values are in milliseconds.

In the second problem, there are more clusters and some clusters have more LANs. Although we do not know the overall global optimum, we can still check for the global optimum for small clusters. Goodness measures described before show that we have good topologies for larger clusters. Therefore, we conjecture that the simulated annealing algorithm finds high quality solutions for the LAN-MAN problem. Running times for the algorithm are short because of the decomposition of the problem.

6 Conclusion

We have described a method based on simulated annealing for finding minimum-delay topologies of interconnected LANs. For the cases studied, the simulated annealing algorithm finds the best topology much faster than complete enumeration for small problems. For larger problems, several goodness measures show that the algorithm finds good topologies which have average delays in the neighborhood of the global minimum. We have extended the algorithm for finding minimum delay topologies for the LAN-MAN interconnection problem. We have rapidly found good topologies for the case in which a MAN is interconnecting several clusters of LANs.

Although we have considered IEEE 802.3-5 LANs and spanning tree topologies, the same methodology can be used for other LANs and general topologies. We have focused on the minimum delay problem. It is also possible to modify the algorithm to incorporate many constraints, such as cost, maximum number of bridges (i.e., hops) between any two LANs and maximum utilization for LANs and bridge ports.

References

[1] A. N. Standards, "IEEE CSMA/CD Access Method," 1985.

[2] F. Backes, "Transparent Bridges for Interconnection of IEEE 802 LANs," *IEEE Network*, pp. 5–9, January 1988.

[3] R. Perlman, "An Algorithm for Distributed Computation of a Spanning Tree in an Extended LAN," *Computer Commun. Rev.*, pp. 44–53, September 1985.

[4] C. Ersoy, S. S. Panwar, R. Dalias, and D. Segal, "Transient Phenomena in Bridged Local Area Networks," in *Proc. IEEE GLOBECOM*, 1990.

[5] G. M. Exley and L. F. Merakos, "Throughput-Delay Performance of Interconnected CSMA/CD Local Area Networks," *IEEE JSAC*, pp. 1380–1390, December 1987.

[6] C. K. Kwok and B. Mukherjee, "On Transparent Bridging of CSMA/CD Networks," in *Proc. IEEE GLOBECOM*, pp. 185–190, 1989.

[7] L. Merakos and H. Xie, "Interconnection of CSMA/CD LANs via an N-port Bridge," in *Proc. IEEE INFOCOM*, pp. 28–37, 1989.

[8] M. A. Rodrigues and V. R. Saksena, "Performance Analysis of LAN/WAN Bridging Architecture," *IEEE JSAC*, pp. 265–270, February 1991.

[9] S. C. Liang and J. R. Yee, "Algorithms for Interconnecting Ethernets with Multi-Port Bridges,". Submitted to IEEE Tr. on Computers, Sept. 1990.

[10] S. C. Liang and J. R. Yee, "A Gateway Allocation Algorithm for Interconnecting Existing Data Networks," in *Proc. IEEE INFOCOM*, pp. 468–473, 1989.

[11] P. C. Fetterolf and G. Anandalingam, "Optimal Design of LAN-WAN Internetworks: An Approach Using Simulated Annealing," in *ORSA Telecommunications Conference*, 1990.

[12] S. Gupta and K. W. Ross, "Performance Modelling and Optimization of Interconnected Ethernets," in *Proc. IEEE INFOCOM*, pp. 1353–1359, 1991.

[13] E. Aarts and J. Korst, *Simulated Annealing and Boltzmann Machines.* John Wiley & Sons, 1989.

[14] B. Hajek, "Cooling Schedules for Optimal Annealing," *Mathematics of Operations Research*, vol. 13, pp. 311–329, 1988.

[15] C. F. Hemrick, R. W. Klessig, and J. M. McRoberts, "Switched Multi-megabit Data Service and Early Availability Via MAN Technology," *IEEE Communications Magazine*, pp. 9–14, April 1988.

[16] N. Metropolis, A. Rosenbluth, M. Rosenbluth, A. Teller, and E. Teller, "Equation of State Calculations by Fast Computing Machines," *Journal of Chemical Physics*, pp. 1087–1092, 1953.

[17] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, "Optimization by Simulated Annealing," *Science*, pp. 671–680, May 1983.

[18] R. Gusella, "A Measurement Study of Diskless Workstation Traffic on an Ethernet," *IEEE Tr. on Communications*, pp. 1557–1568, September 1990.

[19] W. E. Leland and D. V. Wilson, "High Time-Resolution Measurement and Analysis of LAN Traffic: Implications for LAN Interconnection," in *Proc. IEEE INFOCOM*, pp. 1360–1366, 1991.

[20] D. Gross and C. M. Harris, *Fundementals of Queueing Theory, 2nd ed.* John Wiley, 1985.

[21] D. Bertsekas and R. Gallager, *Data Networks.* Prentice-Hall, 1987.