PERFORMANCE OF TRAFFIC MANAGEMENT STRATEGIES FOR INTERCONNECTED HIGH-SPEED LOCAL AND METROPOLITAN AREA NETWORKS

Izhak Rubin and Teresa Cheng

6731 Boelter Hall
Department of Electrical Engineering
University of California, Los Angeles
(213) 825-2326

Abstract

We consider interconnected high-speed local and metropolitan area networks in which call-oriented resource allocation mechanisms are used for traffic management. Management domains are defined for the network and call requests must be admitted by the appropriate domain manager. Assuming Poisson arrival rates and exponential service times, the state process can be modeled as a Markov process with the state defined as the number of active calls of each call type at time t. An equilibrium distribution can be found for this process but the state space quickly becomes intractable with increases in link capacities and in the number of subnetworks. To reduce the state space, we partition the network into subnetwork groups, assuming independence. The probability of blocking can be calculated iteratively and leads to an approximate equilibrium distribution of the system. From these values we can compute the desired performance measures: network utilization, delay, throughput, and packet loss probabilities. An example is presented which illustrates the effects of various surveillance strategies and decision policies on network performance. Finally, considerations on implementing the iterative procedure using parallel processing are discussed.

1 Introduction

We wish to evaluate the effect of surveillance strategies and decision policies on network performance by focusing on admission control policies in high speed interconnected networks. We begin by defining network management domains and providing a control structure for traffic management. Network managers have observability functions and decision functions which determine whether resource requests will be satisfied. A Markov process model of such a network is defined. In theory the equilibrium distribution can be calculated, but in practice the state space grows too rapidly. We then define an iterative algorithm which provides approximate performance measures. Its relationship to Kelly’s algorithm [1] for calculating the probability of blocking in circuit switched networks is discussed. An example demonstrating the algorithm and illustrating some network management tradeoffs is presented. Finally, we consider the possibility of using parallel processing in our iterative algorithm.

2 Hierarchical Control Structure

2.1 Network Structure

We consider a network of LANs interconnected via other LANs and MANs. For simplicity, assume that each LAN operates in broadcast mode. The broadcast feature implies that each network manager has a single pool of capacity resources to allocate. The networks may be connected in an arbitrary topology, but we assume that the control structure of the network may be represented by a single parent hierarchical tree. In a network with arbitrary topology routing becomes an issue, however, we do not address that problem here.

2.2 Control Structure

Each network has a network control unit (NCU) which is responsible for monitoring the status of the network and making appropriate network management decisions. The control structure of the network can be represented by a single parent hierarchical tree in which each node represents a network manager and its associated resources. In some cases, as in Figure 1, the mapping from the physical network to the control structure is self-evident, but in other cases, the structure may have to be negotiated at startup or upon entry to the network. Management domains are defined on the network control structure such that each domain consists of one or more networks. One domain contains the entire network. Domains may fall under the jurisdiction of another domain, but the structure must be strictly nested; a domain must fall entirely within the jurisdiction of all superdomains. The domain manager is one of the NCUs within the domain, typically the one that is the root of the subtree defined by restricting the hierarchical tree to members of that management domain.

* This work was supported by a University of California MICRO grant, by Pacific Bell grant D890724, by US WEST contract D890701, and by a UCLA Academic Senate grant.

Figure 1. (a) A physical network example; (b) The corresponding hierarchical network management control representation.

Figure 2. The network domain structure imposed upon the hierarchical control tree.

323.6.1.
Traffic Management Control Policy

Traffic management is conducted via resource allocations in response to requests along control paths. These control paths are uniquely identified by origin-destination pairs since the control structure is a tree. Associated with each type of request are the control path and required service parameters such as throughput, delay, and packet loss probabilities. For any given network domain, and with respect to the corresponding domain manager, resource requests fall within three categories:

1. requests which can be satisfied by this domain manager
2. requests which do not require resources from this domain manager
3. requests which can be partially satisfied by this domain manager.

Path requests may be satisfied by the smallest domain which contains the necessary resources, i.e., the smallest domain for which it falls into category 1, provided that satisfying such requests do not contradict upper level directives. For simplicity, we assume a pure blocking system with no partial satisfaction of requests. In addition to satisfying resource requests directly, upper level network managers may act in distributing the resources allocated to local and long distance traffic at the subdomains. For example, a domain manager may direct subdomain managers to reserve a certain number of elementary circuits for long distance traffic.

Let the state of the system be the number of each type of resource request that is currently being satisfied. Since each type of resource request has associated service parameters, the state gives complete information about the state of the network. Assume that basic system parameters such as node capacities and service requirements of various traffic types are known by the appropriate network managers. Network managers may not have information about the complete state of the network. The information they have is defined via observation functions which are functions of the system state. For example, one possible observation function is an aggregation function such that the network manager knows how many resources have been allocated, but not the precise pattern of requests that have been satisfied. Observation functions may have several components: the local component which is comprised of information on the local network which may be directly observed by the manager, and reported information which is information reported by superdomains or subdomain managers on their networks. There is a distinction between the local component of the observation function which represents the network managers which must satisfy the request. This set is indexed by \( l = 0, 1, \ldots, L - 1 \). The vector \( c = (c_0, c_1, \ldots, c_{L-1}) \) is the capacity vector in units of elementary circuits with \( c_i \), denoting the capacity of the local network managed by NCU\( i \). In a packet switched system this unit has the natural interpretation. In a packet switched system, we assume that the maximum number of elementary circuits has been defined to satisfy a maximum delay constraint. Traffic characteristics are defined by the vectors:

\[
r = (r_0, r_1, \ldots, r_{L-1}) \quad r_i \text{ gives the resource requirements for traffic type } l
\]

\[
\lambda = (\lambda_0, \lambda_1, \ldots, \lambda_{L-1}) \quad \lambda_i \text{ is the arrival rate of request type } l
\]

\[
\mu = (\mu_0, \mu_1, \ldots, \mu_{L-1}) \quad \mu_i \text{ is the average holding time of resources for request type } l.
\]

The state of the system is given by \( x = (x_0, x_1, \ldots, x_{L-1}) \), where \( x_i \) is the number of currently satisfied type \( l \) service requests. For future reference, define the following notation:

\[
x' = (x_0, x_1, \ldots, x_L, x_1 + 1, x_2, \ldots, x_{L-1})
\]

\[
x' = (x_0, x_1, \ldots, x_{L-1}, x_1 - 1, x_2, \ldots, x_{L-1})
\]

Let \( A \) be an indicator matrix such that

\[
A(i, j) = \begin{cases} 1 & \text{if traffic type } l \text{ requires resources from network } i \\ 0 & \text{otherwise} \end{cases}
\]

Define the vector operation \( \otimes \) such that \( (x \otimes \gamma)_i = x_i y_i \). The network capacity constraints are then

\[
(x \otimes r) A \leq c
\]

3.2 Decision and Observation Functions

Define the set of management domains \( M \) indexed by \( m = 0, 1, \ldots, M - 1 \) such that each domain consists of a set of nodes on the hierarchical tree. Recall that each node on the hierarchical tree represents a network domain manager and its associated resources. Each domain manager has associated observation functions \( \Phi_m(x) \) and decision function \( D_m(x, l) \) which depend implicitly upon the system and call parameters which we assume to be public knowledge. Decisions may differ in their level of centralization. For instance, let \( m_u \) denote a subdomain \( u \) of domain \( m, u = 0, 1, \ldots, U - 1 \), then

\[
D_m(x, l) = \begin{cases} 1 & \text{if request } l \text{ is satisfied} \\ 0 & \text{if request } l \text{ is refused} \end{cases}
\]

is a centralized policy while

\[
D_m(x, l) = \prod_{u=0}^{U-1} D_{m_u}(x, l)
\]

allows for input from the subdomain managers who may have more detailed information about the state of their domains. Note that for a given request type \( l \), the decision function \( D_m(x, l) \) is associated with the highest level domain manager \( m \) involved with this request type.

3.3 Solution Technique

Assume that each resource request type forms an independent Poisson arrival stream, and that the holding times are exponential; then \( X = \{X(t), t \geq 0\} \) forms a continuous time Markov process. Let \( \Omega \) be the set of allowable states determined by the combination of observation functions and decision policies in effect. Given the system parameters, the observation functions and the decision functions, we can calculate the nonzero elements of the infinitesimal generator or transition rate matrix \( Q \) for the Markov process to be given by

\[
Q(x, x') = \lambda_m D_m(x, l) \quad \text{for } l = 0, 1, \ldots, L - 1 \text{ and } x, x'
\]

\[
Q(x, x) = \mu_r x_l
\]

\[
Q(x, x) = \sum_{l=0}^{L-1} Q(x, x')
\]

for \( l = 0, 1, \ldots, L - 1 \) and \( x, x' \in \Omega \).

Our model is set up as a finite state, homogeneous, irreducible Markov process, so we are guaranteed of the existence of a unique equilibrium distribution which satisfies:

\[
\pi Q = 0 \quad \text{and } \sum_i \pi(i) = 1
\]

323.6.2.
In practice, enumerating $\Omega$ is a nontrivial process. Our approach is to begin with all the feasible states consistent with the capacity constraints, and then to calculate the transition probabilities from these states given the observation functions and decision functions in effect. Once this process is complete, the transition rate matrix can be used to determine which states are unreachable. These states are then eliminated.

4 Iterative Algorithm

Solving for the equilibrium distribution of the above system is a simple task conceptually, but the state space grows rapidly with the number of networks and with the capacity at each network. We would like to find a more efficient way to compute the distribution while allowing the flexibility for varying observation and decision functions.

Our solution is to partition the network into subnetwork groups and solve for each subnetwork separately, assuming independence. Choosing groups to coincide with management domains, at least at the lower levels of the hierarchical tree, will allow for a natural mapping of observation and decision functions from management domains to subnetwork groups. Partitioning the system into subnetwork groups imposes some restrictions on the type of decisions that are available from the network point of view. At each subnetwork group, decisions must be made based on current local information rather than current global information since we assume independence. The observation functions from management domains not in the subnetwork group must be based on equilibrium distributions. Decision policies imposed by higher level domain managers have to be built into the decision policies at subdomains.

The decision structures which are to be considered affect the choice of local traffic types at each subnetwork group. A different traffic type must be defined for each set of calls which the manager will address with a different decision rule. One possible breakdown is to differentiate between local and global traffic with respect to the pattern of decomposition into groups. In this way the network manager can account for a decision to reserve a specified number of elementary circuits for a given type of traffic.

![Diagram of subnetwork groups](image)

Figure 3. Illustration of decomposition into 3 subnetwork groups.

4.1 Algorithm Definition

$G$ is the set of groups indexed by $g = 0, 1, \ldots, G - 1$. Local parameters for subnetwork group $g$ are:

- $N_g$: the number of nodes in the subnetwork group
- $L_g$: the number of local traffic types in the group
- $\Omega_g$: the state space of the group, having $S_g$ elements
- $c_{ig}$: the capacity vector, $c_{ig} = (c_i(l), l = 0, 1, \ldots, N_g - 1)$
- $\gamma_g$: the offered traffic rate vector, $\gamma_g = (\gamma_i(l), l = 0, 1, \ldots, N_g - 1)$
- $\mu_g$: the service rate vector, $\mu_g = (\mu_i(l), l = 0, 1, \ldots, L_g - 1)$
- $b_g$: the probability of blocking vector, $b_g = (b_i(l), l = 0, 1, \ldots, L_g - 1)$
- $\pi_g$: the equilibrium distribution, $\pi_g = (\pi_i(s), s = 0, 1, \ldots, S_g - 1)$

In addition we define the following mapping matrices

$$U_g(l, s) = \begin{cases} 1 & \text{if traffic type } l \text{ requires resources from group } s \text{ for group } g \\ 0 & \text{otherwise} \end{cases}$$

which maps global traffic into subnetwork groups while

$$U_g(l, s) = \begin{cases} 1 & \text{if } l \in L_g \\ 0 & \text{otherwise} \end{cases}$$

is an indicator function for global traffic and local traffic where $l$ is the traffic type associated with group $g$. Note that several types of global traffic are often aggregated into one type of local traffic. This aggregation accounts for a large portion of the state space reduction. It will also be helpful to have the information from $U_g$ in another form:

$$T_g(l) = \begin{cases} 1 & \text{if } U_g(l, l) = 1 \\ 0 & \text{otherwise} \end{cases}$$

Define a blocking matrix

$$B_g(l, s) = \begin{cases} 1 & \text{if } s \text{ is a blocking state for traffic type } l \\ 0 & \text{otherwise} \end{cases}$$

where $s$ is a blocking state for $l$ if no new resource requests for type $l$ will be satisfied.

Assume that the state process at each subnetwork group is independent of the state processes at other groups. The offered traffic rate at each group is then equal to the original arrival rate reduced by an amount which takes into account the probability of blocking at each of the other subnetwork groups. To make this notion precise, we write

$$\gamma_g(l) = \sum_{i=0}^{L_g-1} \lambda_i(1 - b_g(T_i(l)))^{(1-b_g(T_i(l)))\gamma_i(l)}\gamma_i(l)$$

where

$$\gamma_i(l) = \sum_{i=0}^{L_g-1} \lambda_i(1 - b_g(T_i(l)))^{(1-b_g(T_i(l)))\gamma_i(l)}\gamma_i(l)$$

The infinitesimal generator matrix for each group can be defined in a manner analogous to that for the entire system, with the nonzero elements equal to:

$$Q_g(x_i, x_j) = \gamma_i(l)Q_g(x_i, l)$$

$$Q_g(x_i, x_j) = \mu_j x_j$$

$$Q_g(x_i, x_j) = -\sum_{i=0}^{L_g-1} \gamma_i(l)Q_g(x_i, l)$$

where $l = 0, 1, \ldots, L_g - 1$ and $x_i, x_j \in \Omega_g$.

To find the equilibrium distribution of each subnetwork group, we solve (for each group $g$)

$$\pi_gQ_g = 0$$

$$\sum_{s \in \Omega_g} \pi_g(s) = 1$$

323.6.3.
The local probabilities of blocking can then be then calculated by

\[ b_i \approx B_i r_i \]  

(4)

The probability that a type 1 service request is rejected (denoted as \( L_0 \)) is then approximated by

\[ L_0 = 1 - \prod_{i=1}^{n} (1 - b_i) \]

This series of relationships suggests an iterative method for solving for the probabilities of blocking and therefore the equilibrium distribution at each subnetwork group. Begin by assuming initial values for the probability of blocking vectors \( b_i \). For each group, use equation (1) to calculate the offered traffic, substitute these values into (2), solve (3), and compute the new probability of blocking using (4). At iteration \( n+1 \), if \( |b_i(n+1) - b_i(n)| \leq \varepsilon \) for all \( i \) and for some appropriately chosen \( \varepsilon \), then stop. Otherwise, use the most recently calculated blocking probabilities and repeat.

### 4.2 Algorithmic considerations

The storage space and execution time of both the iterative algorithm described above and the exact solution method (described in Section 3) depend primarily on the size of the state space. The following values illustrate the state space reduction achieved by using the iterative algorithm in solving for the performance of the network described in Figure 3 as compared to solving for the exact solution.

<table>
<thead>
<tr>
<th>Solution Technique</th>
<th>Number of states</th>
</tr>
</thead>
<tbody>
<tr>
<td>Exact Solution</td>
<td>1634</td>
</tr>
<tr>
<td>Iterative Solution</td>
<td></td>
</tr>
<tr>
<td>Subnetwork 0</td>
<td>5</td>
</tr>
<tr>
<td>Subnetwork 1</td>
<td>145</td>
</tr>
<tr>
<td>Subnetwork 2</td>
<td>66</td>
</tr>
<tr>
<td>Total</td>
<td>216</td>
</tr>
</tbody>
</table>

While we have not proved the convergence of our iterative algorithm, test cases have converged rapidly. We present one example illustrating the maximum difference between iterations.

**Figure 4.** The rate of convergence achieved in our test case. The maximum difference is the infinity norm of the vector of differences in estimates before an iteration and after.

Our iterative approach is similar to Kelly's method [1], but we use subnetwork groups rather than individual nodes as our basic entity, and solve exactly for the equilibrium distribution within each subnetwork group. Although Kelly's method is faster, the hierarchical iterative approach tends to be more accurate since it solves for exact values within subgroups. Figure 5 illustrates estimates provided by the exact solution, the hierarchical iterative algorithm, and Kelly's method for the case where decisions are made solely on the basis of capacity constraints.

### 5 Example Results

In controlling the network, the network manager must perform trade-offs between conflicting criteria such as efficiency and fairness. The network manager must prevent one traffic type or several traffic types from "hogging" the resources. One possible control strategy is to have window control functions determining the maximum resources which specific traffic types may utilize given the state of the network. If the network is heavily loaded, one traffic type may be limited to \( n_1 \) elementary circuits, while the limit is \( n_2 \) when the network is moderately loaded, and \( n_3 \) when the network is lightly loaded, with \( n_1 < n_2 < n_3 \). Determining optimal values for \( n_1, n_2, \) and \( n_3 \) as well as thresholds for heavy and medium traffic requires study.

To illustrate the effects of a threshold policy for limiting the number of calls of a specific call type, we calculate the performance of our example network with and without a threshold policy as the arrival rate of traffic type \( x(3,5) \) is increased (with all other arrival rates remaining constant). Without a threshold policy limiting \( x(3,5) \) calls, all other calls requiring resources that are also required by \( x(3,5) \) calls suffer from a sharp increase in the probability of blocking, although calls not requiring common resources benefit since some competing calls are blocked as shown in Figure 6. Under a threshold based traffic management policy, the calls requiring resources in common with \( x(3,5) \) suffer some increase in the probability of blocking, but noticeably less than in the previous case as illustrated in Figure 7. The burden of the increased load is carried by \( x(3,5) \) calls which have a significantly higher probability of blocking.

**Figure 5.** This graph illustrates the probability of blocking values provided by Kelly's algorithm, the method for calculating the exact solution as described in section 3, and the hierarchical iterative algorithm.

**Figure 6.** The probability of blocking without a threshold management policy.
6 Parallel Processing Considerations

We consider coarse-grained parallel processing in which a small number of powerful processors is loosely coupled so that each one may be performing a different type of task at any given time.

6.1 Task Allocation

One natural way to assign tasks in the iterative algorithm is to assign one processor to the computations required for each subgroup. This task assignment works best if the subnetwork groups have been chosen so that each one has approximately the same number of states.

6.2 Process Communications Requirements

The processor interconnection is the mechanism by which processors exchange information. Possibilities include shared memory, message passing, or a combination of the two. In examining message passing systems, the topology of the network containing the processors must be considered. In our case assume that there is shared memory although each processor may have its own local memory in addition. The information that must be stored in shared memory is the probability of blocking for each call type at each subnetwork group. Additional system parameters and matrices may be stored in shared memory to avoid duplication at each local memory site.

6.3 Synchronization and Convergence Issues

A synchronous algorithm is divided into phases during which the processors do not interact. All information exchange takes place at the end of phases. The hierarchical iterative algorithm appears to be a naturally synchronous algorithm. Given the probabilities of blocking at the other subnetwork groups, each processor calculates its offered rate, instantaneous rate matrix, equilibrium distribution, and finally its current probability of blocking vector. This marks the end of a phase. All processors write the newly calculated probability of blocking values into shared memory and proceed to a new phase.

Since each processor may finish at a different time, a mechanism for signaling the end of a phase must be devised.

Implementing the hierarchical iterative algorithm in this way differs in one key respect from the algorithm presented in section 4. The algorithm is defined serially so that the update formulas take the form:

\[ b_i(t + 1) = f(b_i(t), b_{i-1}(t), \ldots, b_{i-N}(t), b_{i-N}(t)) \]

In synchronous parallel algorithm the update formulas are of the form:

\[ b_i(t + 1) = f_j(b_i(t), b_j(t), \ldots, b_{j-N}(t), b_{j-N}(t)) \]

The form of the updates is analogous to the comparison between the Gauss-Seidel method and Jacobi's method for finding the fixed point of linear systems. The Gauss-Seidel method is frequently much faster than Jacobi's method. Intuitively, convergence is accelerated if the updated values of the variables are incorporated into subsequent updates of other variables as quickly as possible.[2]

Asynchronous algorithms are more flexible regarding the use of information from other processors. In the synchronous algorithm, if processor loads are poorly balanced, the parallel implementation of the algorithm will be inefficient since processors have to wait for the slowest one to complete each phase. In an asynchronous algorithm, local algorithms do not have to wait for predetermined data to become available; the processors continue to compute with whatever data is available at the time.

The parallel algorithm can be modified so that each process need not wait for updates from all the other processes: it needs only one new update. In this way updated values of variables can be incorporated into the computation faster than in the synchronous algorithm. It is possible that this implementation can realize some of the speed advantage of the Gauss-Seidel form over the Jacobi form without sacrificing parallelism.

In implementing the iterative algorithm in a parallel manner, we discovered that the size of the subgroups affects the rate of convergence. By using a few large subgroups the algorithm converged only a bit more slowly than when using the serial method, although the values oscillated a bit before settling to the correct number. When partitioning the network into smaller subgroups, a greater number of iterations was required before convergence. In the extreme case when each node was a separate subgroup, the iteration never converged at all, but oscillated between two values. With larger groups, values computed within the group are essentially incorporated instantaneously. In the case with many subgroups, the estimates oscillate between high blocking and low blocking levels because of a phase lag: at one iteration each group sees a high offered traffic rate and thus computes a high blocking probability. At the next iteration each group sees a low offered rate and computes a low blocking probability, resulting in oscillation.

6.4 Time Complexity

In comparing the serial algorithm versus the parallel algorithm, the rates of convergence must be weighted against the time per iteration. As mentioned in the previous section, parallel algorithms are likely to converge less rapidly than serial algorithms. More detailed analysis of these tradeoffs is necessary.

7 Conclusion

We have described an algorithm which can be used to analyze traffic management policies. An example was provided to illustrate the benefits of traffic management in providing fair access to network resources as illustrated for an interconnected local and metropolitan area network structure. Our approach addresses the performance evaluation of traffic management for interconnected network management systems by using a natural domain structure and providing a mechanism for analyzing the information flow between network managers which is necessary for effective control.

References
