International Congress on Computational Methods in Engineering
May 2 - 6 1993
School of Engineering
Shiraz University

بسمه تعالی

کنگره بین‌المللی روشهای محاسباتی
در مهندسی
۱۲ الی ۱۶ اردیبهشت ۷۲
دانشکده مهندسی
دانشگاه شیراز

COMPUTATION OF THE EXACT LIMITING DISTRIBUTION FOR THE BLOCKING PROBABILITY IN NETWORK MODELS

Paul A. Farrell, Kazim Khan, and Hassan Peyravi
Department of Mathematics and Computer Science
Kent State University
Kent, Ohio, 44242

e-mail: peyravi@cs.kent.edu

The end-to-end blocking probability is used as a measure of performance for circuit switched networks and delay is used for packet switched networks. In both cases, alternate routes are required to reduce the blocking probability, increase the throughput, and improve reliability of the network. In this paper, first, we describe a model to compute the end-to-end blocking probability analytically. We discuss a way in which the redirected traffic does not have to be a Poisson and/or the network be symmetric. We use state dependent networks to derive the Markovian processes. Later, we discuss a new symbolic parallel algorithm in which the set of alternate paths between two given nodes in a network can be computed. This set can be used to compute the set of all valid states the network could attain. A parallel algorithm is also developed to identify all valid states among a large number of states.

INTRODUCTION

The *end-to-end* blocking probability has been used as a performance criterion to quantify network reliability. In a network with unpredicted traffic, the shortest path algorithm may not decrease the the overall blocking probability. It has been shown that different types of routing schemes for circuit switched traffic with a uniform traffic, would lead to different performance trade-offs [1]. It is, however, not clear how the trade-offs would be affected by non-uniform traffic on the links.

Three parameters determine the performance of a particular routing algorithm, the run-time of the algorithm in which the appropriate path is computed and established, the probability that the algorithm fails to establish a connection due to the network congestion, network failure, etc., and the amount of randomness used by the algorithm. Peleg and Upfa [2] studied the trade-off between these parameters for the problem of routing on a bounded degree network.

In a real network, it is neither practical nor efficient to analyze and compute the blocking probability sequentially. Parallel algorithms are required to evaluate the routes within an acceptable period. This becomes crucial when the network configuration changes due to failure or update.

First, we briefly discuss a new analytical approach which can be used to identify the trade-offs[3]. The approach is based on a state-dependent technique in which the end-to-end blocking probability can be formulated. This requires the computation of the set of all alternate paths between a given pair of nodes in the network. Later, we present a set of parallel algorithms to compute all alternate paths of different lengths in a network as well as an approach to compute the set of valid states of the network. Since the number of states grows exponentially with the number of nodes in a network, parallel algorithms are essential to find the set of alternate paths, set of valid states, and the balance equations in a reasonable amount of time.

Since the shortest path algorithm does not guarantee a lower blocking probability, especially when the traffic is not uniform, alternate routing schemes have been widely used to improve the network reliability [1, 4]. To decide which alternate route is to be chosen, knowledge of the end-to-end blocking probability is essential. This can be obtained, when the distribution of calls (voice, data, etc.) is known. Several special cases, such as fully connected networks, symmetric networks, or networks with a specific topology have been studied analytically and/or examined by simulation models [1, 3, 4, 5, 6, 7].

Section 2 presents an overview of the finite state approach using Markov chains to compute the blocking probability sequentially. Generation of balance equations governing the operation of the birth-death process or state-dependent queueing system at equilibrium, and the parallel algorithms to compute the set alternate paths needed for the computation of the end-to-end blocking probability are also given. Computation of the set of valid states, and state transitions of a network are also discussed in this section. Section 3 presents a set of parallel algorithms to speed up this computation. Conclusions and open issues are discussed in section 4.

STATE-DEPENDENT NETWORKS: BIRTH-DEATH PROCESS

For nonhierarchical networks, it is shown [1] that there is a cut-off point for the traffic load before which alternate routing gives lower blocking probability than nonalternate routing. Krupp [4] developed a mathematical model for symmetric and uniformly loaded networks with alternate routing schemes. In this model, the blocking probability is considered as a function of the external traffic load and a bistable behavior was observed during overloads. Later, Akinpelu [5] extended the analysis to nonsymmetric networks and similar instability results were observed. Yum and Schwartz [1] compared different types of routing algorithms for fully connected and symmetric networks. Their results showed that alternate routing performs better than nonalternate routing, if the traffic load is light. For heavy traffic, direct routing performs better as far as the end-to-end blocking probability of a particular link is concerned.

Here, we discuss a technique that can be generalized for an arbitrary network with uniform or nonuniform traffic on the links. We can thus assume different mean arrival rates, λ_{ij}, on different links $1 \leq i, j \leq n$, and $i < j$. Consider the 3-node network of Figure 1. For each source-destination pair, there is a primary path and a secondary (alternate) path in the network. For an arbitrary network, we may have a nonuniform number of paths between any given pair of nodes and these may not be disjoint. In this example, there are six paths (3 direct and 3 indirect); $p_1 = \ell_{12}$, $p_2 = \ell_{13}$, $p_3 = \ell_{23}$, $p_4 = \ell_{13} + \ell_{23}$, $p_5 = \ell_{12} + \ell_{23}$, and $p_6 = \ell_{12} + \ell_{13}$.

A network consists of a set of nodes, $N = \{n_1, n_2, \cdots, n_k\}$ and a set of links $L = \{\ell_{ij};\ 1 \leq i, j \leq k\ , i < j\}$. Link ℓ_{ij} connects node n_i to node n_j and carries c_{ij} channels. Figure 2 illustrates a network of five nodes and six links and it is a 1-fault tolerant, i.e., it can handle one link or switch failure. Without loss of generality, we assume undirected graphs in this work. A route R is an alternating sequence of nodes and links, beginning and ending with nodes. In a typical network,

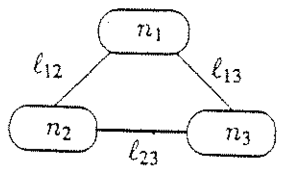

Figure 1: A 3-node fully connected network.

N and L might be measured in tens and the set R could be much larger.

A call is a basic unit of circuit traffic. Each call is originated from a source node n_i and is connected to a node n_j. The call arrival process entering at each node is assumed to be Poisson with rate λ_{ij} and the call holding times are exponentially distributed with mean $1/\mu$, for an Erlang rate of $\rho_{ij} = \lambda_{ij}/\mu$. The call arrival distributions for different nodes are not necessarily uniform. The call setup and disconnection times are very short compared to the call holding time and assumed to be zero. The first step to derive the Markov process is to compute the set of valid states. In this model each link is assumed to carry at most one call at a time. This reduces the complexity of the computation for this analysis. Similar results hold for a larger number of channels per link.

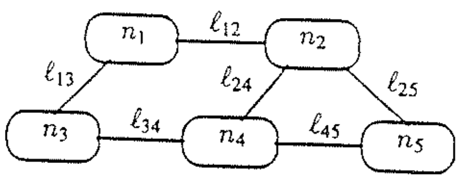

Figure 2: 'A Configuration of a five node network.

Transmittance Matrix A matrix representation can be used for parallel execution and manipulation of a network. The *adjacency* matrix A of a network G is an $n \times n$ matrix $[a_{ij}]$, with $a_{ij} = 1$ if there exists a link ℓ_{ij} between node n_i and node n_j, and $a_{ij} = 0$ otherwise; where n is the number of nodes in the network. The powers of the adjacency matrix A give information about the number of paths from one node to another [8]. The i, j entry $a_{ij}^{(m)}$ of A^m is the number of paths of length m from n_i to n_j. The adjacency matrix A and the matrix A^2 of the network of Figure 2 are:

$$A = \begin{bmatrix} 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 \end{bmatrix} \quad A^2 = \begin{bmatrix} 0 & 0 & 0 & 2 & 1 \\ 0 & 0 & 2 & 1 & 1 \\ 0 & 2 & 0 & 0 & 1 \\ 2 & 1 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 0 \end{bmatrix}$$

The *reachability* matrix R of a network G is the $n \times n$ matrix $[r_{ij}]$ with $r_{ij} = 1$, if there exists a path p_{ij} between node n_i and node n_j, and $r_{ij} = 0$ otherwise. The entry of the reachability matrix can be obtained from the powers of A such that $r_{ij} = 1$, if and only if for some m, $a_{ij}^{(m)} > 0$.

We define the *transmittance* matrix T of a network G to be an $n \times n$ matrix such that $t_{ij} = \ell_{ij}$, if there exist a link from node n_i to node n_j, and $t_{ij} = 0$ otherwise. ℓ_{ij} is the symbolic representation for the link between nodes n_i and n_j. The transmittance matrix can be manipulated using symbolic algebra software such as MACSYMA. To fit the following algorithm into a parallel cube matrix multiplication, we assume $0 \leq i,\ j \leq n-1$ The m^{th} power of the transmittance matrix T provides alternate paths of length m between two nodes n_i and n_j subject to the following conditions:

1. $t_{ij}^{(m)} = 0$ if and only if $i = j$ and $0 \leq i, j \leq n-1$,
2. $t_{ij}^{(m)} \times t_{ij}^{(m)} = t_{ij}^{(m)}, 0 \leq i, j \leq n-1$,
3. $c\, t_{ij}^{(m)} = t_{ij}^{(m)}$, where c is a constant and $0 \leq i, j \leq n-1$.

$t_{ij}^{(m)}$ is the i, j entry of T^m and gives the alternate paths from node n_i to node n_j in the form of the *sum of products* of the links. In order to implement this algorithm in parallel, we can use the matrix multiplication algorithms discussed in [9, 10] with adapted to use the above conditions. The resulting algorithm is given in the procedure ALTERNATE ROUTES which follows. This procedure takes the transmittance matrix T, and m as input and returns the alternate paths matrix, R of degree m as output. It can run on a cube-connected SIMD (Single Instruction, Multiple Data) computer with n^3 processors. The processors can be thought of as being arranged in an $n \times n \times n$ array pattern. In this array, processor P_s, $1 \leq s \leq n^3$ occupies position (i, j, k), where $s = in^2 + jn + k$ and $0 \leq i, j, k \leq n-1$. It has three registers $A(i, j, k)$, $B(i, j, k)$, and $R(i, j, k)$. Initially, the processors in positions $(0, j, k)$, $0 \leq j, k \leq n-1$, contain the transmittance matrix, that is , $T(0, j, k) = t_{jk}$. At the end of computation, these processors contain the alternate route matrix, that is, $R(0, j, k) = r_{jk}$, $0 \leq j, k \leq n-1$.

```
procedure ALTERNATE ROUTES $(T, m, R)$
Step 1: {The diagonal elements of the transmittance matrix are made equal to 0}
        for $j = 0$ to $n-1$ do in parallel
            $R(0, j, j) \leftarrow 0$
        end for.
Step 2: {The A registers are copied into the B registers}
        for $j = 0$ to $n-1$ do in parallel
            for $k = 0$ to $n-1$ do in parallel
                $B(0, j, j) \leftarrow A(0, j, k)$
            end for
        end for.
Step 3: {The alternate routes matrix $R$ of degree $m$ is obtained
        through power $m$ of the repeated symbolic multiplications}.
        $r_{ij}$ is the sum of product with $m$ product terms.
        for $i = 1$ to $m$ do
        (3.1) CUBE MATRIX MULTIPLICATION $(A, B, R)$
        (3.2) $R(0, j, j) \leftarrow 0$
        (3.3) for $j = 0$ to $n-1$ do in parallel
                  for $k = 0$ to $n-1$ do in parallel
                      i. $r_{jk} \times r_{jk} = r_{jk}$
                      ii. $r_{jk} + r_{jk} = r_{jk}$
                      iii. $A(0, j, k) = R(0, j, k)$
                      iv. $B(0, j, k) = R(0, j, k)$
                  end for
            end for
```

end for. □

The algorithm can be implemented in parallel by invoking procedure CUBE MATRIX MULTIPLICATION of [9]. The procedure runs on a cube-connected SIMD computer with $N = n^3$ processors, each with three registers, A, B, and C. Step 1, 2, 3.2, and 3.3 take constant time. In step 3.1 procedure CUBE MATRIX MULTIPLICATION is iterated m times, $1 \leq m \leq \log n$. It follows that the running time of this procedure is $O(\log^2 n)$. Since n^3 processors are used, the overall running time is $O(n^3 \log^2 n)$. The transmittance matrix and alternate routing matrices, T^2, and T^3, of Figure 2 are computed as follows:

$$T = \begin{bmatrix} 0 & \ell_{12} & \ell_{13} & 0 & 0 \\ \ell_{12} & 0 & 0 & \ell_{24} & \ell_{25} \\ \ell_{13} & 0 & 0 & \ell_{34} & 0 \\ 0 & \ell_{24} & \ell_{34} & 0 & \ell_{45} \\ 0 & \ell_{25} & 0 & \ell_{45} & 0 \end{bmatrix}$$

$$T^2 = \begin{bmatrix} 0 & 0 & 0 & \ell_{13}\,\ell_{34} + \ell_{12}\,\ell_{24} & \ell_{12}\,\ell_{25} \\ 0 & 0 & \ell_{24}\,\ell_{34} + \ell_{12}\,\ell_{13} & \ell_{25}\,\ell_{45} & \ell_{24}\,\ell_{45} \\ 0 & \ell_{24}\,\ell_{34} + \ell_{12}\,\ell_{13} & 0 & 0 & \ell_{34}\,\ell_{45} \\ \ell_{13}\,\ell_{34} + \ell_{12}\,\ell_{24} & \ell_{25}\,\ell_{45} & 0 & 0 & \ell_{24}\,\ell_{25} \\ \ell_{12}\,\ell_{25} & \ell_{24}\,\ell_{45} & \ell_{34}\,\ell_{45} & \ell_{24}\,\ell_{25} & 0 \end{bmatrix}$$

$$T^3 = \begin{bmatrix} 0 & \ell_{13}\,\ell_{24}\,\ell_{34} & \ell_{12}\,\ell_{24}\,\ell_{34} & \ell_{12}\,\ell_{25}\,\ell_{45} & \ell_{13}\,\ell_{34}\,\ell_{45} + \ell_{12}\,\ell_{24}\,\ell_{45} \\ \ell_{13}\,\ell_{24}\,\ell_{34} & 0 & \ell_{25}\,\ell_{34}\,\ell_{45} & \ell_{12}\,\ell_{13}\,\ell_{34} & 0 \\ \ell_{12}\,\ell_{24}\,\ell_{34} & \ell_{25}\,\ell_{34}\,\ell_{45} & 0 & \ell_{12}\,\ell_{13}\,\ell_{24} & \ell_{24}\,\ell_{25}\,\ell_{34} + \ell_{12}\,\ell_{13}\,\ell_{25} \\ \ell_{12}\,\ell_{25}\,\ell_{45} & \ell_{12}\,\ell_{13}\,\ell_{34} & \ell_{12}\,\ell_{13}\,\ell_{24} & 0 & 0 \\ \ell_{13}\,\ell_{34}\,\ell_{45} + \ell_{12}\,\ell_{24}\,\ell_{45} & 0 & \ell_{24}\,\ell_{25}\,\ell_{34} + \ell_{12}\,\ell_{13}\,\ell_{25} & 0 & 0 \end{bmatrix}$$

To compute the set of valid states the network could attain, a bit map can be used. We create a one-to-one correspondence between each bit and a path in the network. We assume each link carries only one channel. For higher number of channels, the bit map can be extended. Each state of the network can be uniquely defined by a vector. A binary vector B, represents each state. $B[i] = 1$ if p_i carries a call, $1 \leq i \leq p$. For higher number of channels per link, $B[i]$ could be assigned larger integers. The state transitions can be computed by call arrival (Arr.) or call departure (Dep.) events. There are 64 possible states for the network of Figure 1. Only 14 of them are valid. The network can not get to the other 50 states due to limited available channels on some links. For example, the state in which a direct call is connected via ℓ_{12} and an indirect call is connected via p_5 or p_6 is not a valid state. If we increase the number of channels per link to 2, then we would have 3^6=729 states and only 85 of them are valid. The next step is to compute the state transitions. The state transitions can be computed by applying events such as call arrival (Arr.) and departure (Dep.) to each state of the network. The algorithm finds the shortest available alternate paths when there is more than one alternate path in the network. The state vector corresponds to paths, $p_1 = \ell_{12}, p_2 = \ell_{13}, p_3 = \ell_{23}, p_4 = \ell_{13} + \ell_{23}, p_5 = \ell_{12} + \ell_{23}$, and $p_6 = \ell_{12} + \ell_{13}$. These are illustrated on the second column of Table 1. For example, state 1 corresponds to the network when all links are idle. State 8 represents the network when two calls are in progress; a direct call on p_2, and an indirect call on p_5.

Let P be an array of size p representing the number of calls on different paths and let p_k be a path which covers link(s) ℓ_{ij} for some i and j, $i < j$, $1 \leq i, j \leq n$, and $1 \leq k \leq p$. In other words, ℓ_{ij}

is embedded in path p_k, and $1 \le k \le p$. The following conditions can be implemented in parallel to determine the validity of a particular state. The bit map $B[i]$, $1 \le i \le p$ represents a valid state if:

1. for every link, ℓ_{ij}, $0 \le flow(\ell_{ij}) \le c_{ij}$, where $flow(\ell_{ij})$ is an integer representing the number of occupied channels on link ℓ_{ij}, including channels for redirected traffic.
2. $\sum_{k=1}^{p} flow(P[k]) \le Minimum\ c_{ij}$, where link ℓ_{ij} is a link on path p_k, $1 \le i, j \le n$, $i < j$.

This procedure is inherently parallel and can be easily implemented on a parallel machine. The $\sum_{k=1}^{p} flow(P[k])$ takes $\log p$ and the $Minimum\ c_{ij}$ takes $\log \ell$ steps, where ℓ is the number of links in the network.

Table 1 illustrates the states and their transitions for the network of Figure 1 with one channel per link.

Table 1: State Transitions.

State No.	Current State, $B[i]$	Events (input)	Path	Next State	State No.	Current State, $B[i]$	Events (input)	Path	Next State
1	000000	Arr. Arr. Arr.	p_1 p_2 p_3	100000 010000 001000	8	010010	Dep. Dep.	p_2 p_2	000010 010000
2	100000	Arr. Arr. Arr. Dep.	p_5 p_2 p_3 p_1	100100 110000 101000 000000	9	011000	Arr. Dep. Dep.	p_1 p_2 p_3	111000 001000 010000
3	010000	Arr. Arr. Arr. Dep.	p_1 p_4 p_3 p_2	110000 010010 011000 000000	10	001001	Dep. Dep.	p_3 p_6	000001 001000
4	001000	Arr. Arr. Arr. Dep.	p_1 p_2 p_6 p_3	101000 011000 001001 000000	11	000100	Arr. Dep.	p_1 p_5	100100 000000
5	100100	Dep. Dep.	p_1 p_5	000100 100000	12	111000	Dep. Dep. Dep.	p_1 p_2 p_3	011000 101000 110000
6	110000	Arr. Dep. Dep.	p_3 p_1 p_2	111000 001000 100000	13	000010	Arr. Dep.	p_2 p_2	010010 000000
7	101000	Arr. Dep. Dep.	p_2 p_1 p_3	111000 001000 100000	14	000001	Arr. Dep.	p_3 p_6	001001 000000

Having the states and their transitions, one could generate the corresponding balanced equations. The balanced equations for the state transitions of Table 1 are:

$$
\begin{aligned}
(\lambda_{12}+\lambda_{13}+\lambda_{23})S_1 &= \mu_{12}S_2+\mu_{13}S_3+\mu_{23}S_4+\mu_{12}S_{11}+\mu_{13}S_{13}+\mu_{23}S_{14}\\
(\lambda_{12}+\lambda_{13}+\lambda_{23}+\mu_{12})S_2 &= \lambda_{12}S_1+\mu_{12}S_5+\mu_{13}S_6+\mu_{23}S_7\\
(\lambda_{12}+\lambda_{13}+\lambda_{23}+\mu_{13})S_3 &= \lambda_{13}S_1+\mu_{12}S_6+\mu_{13}S_8+\mu_{23}S_9\\
(\lambda_{12}+\lambda_{13}+\lambda_{23}+\mu_{23})S_4 &= \lambda_{23}S_1+\mu_{12}S_7+\mu_{13}S_9+\mu_{23}S_{10}\\
(\mu_{12}+\mu_{12})S_5 &= \lambda_{12}S_2+\lambda_{12}S_{11}\\
(\lambda_{23}+\mu_{12}+\mu_{13})S_6 &= \lambda_{13}S_2+\lambda_{12}S_3+\mu_{23}S_{12}\\
(\lambda_{13}+\mu_{12}+\mu_{23})S_7 &= \lambda_{23}S_2+\lambda_{12}S_4+\mu_{13}S_{12}\\
(\mu_{13}+\mu_{13})S_8 &= \lambda_{13}S_3+\lambda_{13}S_{13}\\
(\lambda_{12}+\mu_{13}+\mu_{23})S_9 &= \lambda_{23}S_3+\lambda_{13}S_4+\mu_{12}S_{12}\\
(\mu_{23}+\mu_{23})S_{10} &= \lambda_{23}S_4+\lambda_{23}S_{14}\\
(\lambda_{12}+\mu_{12})S_{11} &= \mu_{12}S_5\\
(\mu_{12}+\mu_{13}+\mu_{23})S_{12} &= \lambda_{23}S_6+\lambda_{13}S_7+\lambda_{12}S_9\\
(\lambda_{13}+\mu_{13})S_{13} &= \mu_{13}S_8\\
(\lambda_{23}+\mu_{23})S_{14} &= \mu_{23}S_{10}
\end{aligned}
$$

Here λ_{ij} is the arrival rate between node i and node j for the Poisson process, μ_{ij} is the departure rate for the same link, S_i is the probability that the network is at state i, and $\sum_{i=1}^{14} S_i = 1$. The call holding times are assumed to be exponentially distributed with the same parameter, μ, and hence $\mu_{ij}=\mu$ for all $1 \leq i,j \leq n$ and $i < j$. By dividing both sides of the equations by μ, we have

$$
\begin{aligned}
(\rho_{12}+\rho_{13}+\rho_{23})S_1 &= S_2+S_3+S_4+S_{11}+S_{13}+S_{14}\\
(\rho_{12}+\rho_{13}+\rho_{23}+1)S_2 &= \rho_{12}S_1+S_5+S_6+S_7\\
(\rho_{12}+\rho_{13}+\rho_{23}+1)S_3 &= \rho_{13}S_1+S_6+S_8+S_9\\
(\rho_{12}+\rho_{13}+\rho_{23}+1)S_4 &= \rho_{23}S_1+S_7+S_9+S_{10}\\
2S_5 &= \rho_{12}S_2+\rho_{12}S_{11}\\
(\rho_{23}+2)S_6 &= \rho_{13}S_2+\rho_{12}S_3+S_{12}\\
(\rho_{13}+2)S_7 &= \rho_{23}S_2+\rho_{12}S_4+S_{12}\\
2S_8 &= \rho_{13}S_3+\rho_{13}S_{13}\\
(\rho_{12}+2)S_9 &= \rho_{23}S_3+\rho_{13}S_4+S_{12}\\
2S_{10} &= \rho_{23}S_4+\rho_{23}S_{14}\\
(\rho_{12}+1)S_{11} &= S_5\\
3S_{12} &= \rho_{23}S_6+\rho_{13}S_7+\rho_{12}S_9\\
(\rho_{13}+1)S_{13} &= S_8\\
(\rho_{23}+1)S_{14} &= S_{10}\\
\textstyle\sum_{i=1}^{14} S_i &= 1.
\end{aligned}
$$

Using the MACSYMA or LINPACK software packages the system can be solved for S_i's, $1 \leq i \leq 14$. The above technique can be used to compute the end-to-end blocking probability between two nodes, a set of nodes, or the entire network as a function of traffic intensity ρ_{ij}, $1 \leq i,j \leq n$ and $i < j$ [3].

CONCLUSIONS

The results presented in this paper indicate that there is a cut-off point for the traffic intensity before which alternate routing gives lower blocking probability than nonalternate routing. An analytical model is developed to compute the end-to-end blocking probability of a network. A simulation model is developed to study different alternate routing with adaptive capability. Several adaptive routing algorithms for non-hierarchical distributed circuit switched networks have been studied. Blocking probability, average call setup time, and average number of crank back calls were extracted from the simulation as performance measures. The results shows that local adaptive routing schemes, in which explicit information is shared among nodes could decrease the average blocking probability and call setup time. Sharing information about the network status (nonlocal adaptive routings) could lead to even better performance. Further study is needed to identify the trade offs of using adaptive nonlocal routing algorithms.

Acknowledgements: The Author benefited greatly from discussion with P. Farrell and K. Khan.

REFERENCES

[1] Yum, Tak-Kin G., and Schwartz, Mischa, "Comparison of Routing Procedures for Circuit-Switched Traffic in Nonhierarchical Networks," *IEEE Trans. Commun.*, COM-35, No. 5, pp. 535-544, 1987.

[2] Peleg, David and Upfa, Eli, "A Time-Randomness Trade-off for Obvious Routing," *SIAM J. of Comput.*, 19, No. 2, pp. 256-266, 1990.

[3] Peyravi, H., Khan, M. K., and Farrell, Paul A., "Limiting Distributions of Some Circuit Switched Nonhierarchical Networks," *Pak. J. of Stat.*, 7, No. 2A, pp. 117-128, 1991.

[4] Krupp, R. S., "Stabilization of Alternate Routing Networks with Dynamic Routing," *Presented at IEEE Int. Commun. Conf.*, Philadelphia, PA, paper No. 31.2, 1982.

[5] Akinpelu, J. M., "The Overload Performance of Engineered Networks with Nonhierarchical and Hierarchical Routing," *Bell Syst. Tech. J.*, 63, pp. 1261-1281, 1984

[6] Kelly, F. P., "Blocking Probability in Large Circuit-Switched Networks," *Adv. App. Prob.*, 18, pp. 473-505, 1986.

[7] Kelly, F. P., "Routing in Circuit-Switched Networks: Optimization, Shadow Prices and Decentralization," *Adv. App. Prob.*, 20, pp. 112-144, 1988

[8] Harary, F., *Graph Theory*, Addison-Wesley, 1972.

[9] Akl. S. G., *The Design and Analysis of Parallel Algorithms*, Prentice Hall, Englewood Cliffs, New Jersey, 1989.

[10] Dekel, E., Nassimi, D., and Sahni, S., "Parallel Matrix and Graph Algorithms," *SIAM Journal on Computing*, 10, No. 4, pp. 657-675, 1981.