ON VIRTUAL CIRCUIT ROUTING AND RE-ROUTING IN PACKET-SWITCHED NETWORKS

Ren-Hung Hwang and James Kurose
Department of Computer and Information Science
University of Massachusetts
Amherst, MA 01003

COINS Technical Report 90-104

On Virtual Circuit Routing and Re-routing in Packet-switched Networks [1]

Ren-Hung Hwang and James Kurose

Department of Computer and Information Science
University of Massachusetts
Amherst, Massachusetts 01003

Abstract

In this paper we examine the effects of using different network status information in the initial placement and dynamic re-routing of virtual circuits in packet switched-networks. In the case of virtual circuit re-routing, the effects of call tear down/reestablishment times and resequencing delays are considered. Both analytic and simulation results are presented. The analysis of a dynamic rerouting policy is based on first decomposing the network model and then using a Matrix-Geometric solution of a continuous time Markov chain model. The results of our study indicate that assigning virtual circuits on the basis of the number of virtual circuits on a path provides better performance than using the number of unacknowledged packets, except when virtual circuit holding times are extremely short.

[1]This research was supported in part by the Office of Naval Research under contract N00014-87-K-304 and an equipment grant from the National Science Foundation CER-DCR-8500332

1 Introduction

The routing problem is a central issue in the design and management of computer communication networks [16,14]. In the broadest sense, two approaches towards routing may be identified. In the case of virtual circuit (VC) routing, a path is first set up on an end-to-end basis through the network. Packets flowing on this VC then traverse the network following the path established and arrive at the destination node in the sequence in which they were transmitted. The VC routing approach embodies the so-called "connection-oriented" approach towards packet-switching. The second approach towards routing is a connectionless one, with packets ("datagrams") moving through the network on an individual basis, thus possibly arriving out of order at the destination node.

Dynamic routing is often used to balance the load throughout the network and improve the delay performance by routing messages away from congested areas in the network and towards lightly-loaded areas. A number of network-level dynamic routing strategies have been proposed and implemented for datagram-oriented networks [12,15,14,16]. The dynamic routing of virtual circuits, however, has been been a much less well-studied problem. In this paper, we thus investigate various policies for the initial routing and dynamic rerouting of virtual circuits in a network. From a practical standpoint, rerouting a virtual circuit creates at least two problems. First, the guarantee of delivering data in order is lost; this implies resequencing is required. Second, an additional delay is required to shutdown a VC on its original path and set it up on an alternate path. Both of these factors will be considered in this paper. The potential benefit, of course, is that the load in the network can be balanced more adaptively, and hence performance can be improved.

Our particular focus in this paper is on examining the effects of using different network status information in making VC routing decisions. Among the network status information we consider are the number of VC's currently being routed over each of the possible paths between a source/destination pair, and the the number of unacknowledged messages (queue lengths) on each of these routes. Informally, the queue length information can be thought of as providing *instantaneous* state information, while the number of VC's on each path is a longer-term characterization of the network state. In this paper, both analytic and simulation models are developed to study these issues. The analysis of a dynamic rerouting policy is based on first decomposing the network model and then using a Matrix-Geometric solution of a continuous time Markov chain model. The results of our study indicate that assigning virtual circuits on the basis of the number of virtual circuits on a path provides better performance than using the number of unacknowledged packets, except when virtual circuit holding times are extremely short.

Dynamic routing and rerouting of VC's has also been considered in several previous efforts.

In [4], an optimal dynamic routing problem is formulated for a virtual circuit model in which each link has a small number of thresholds and weights; these weights are then used to select paths for *new* VC's entering the network; rerouting of an ongoing circuit is not considered. The goal in [4] is to find the optimal thresholds and weights such that the average delay in the network is minimized. As discussed in [4], this technique cannot obtain perfect balancing of network load among network links due to the large granularity of information. In the Codex network [1], the method for initially routing virtual circuits (i.e., when a call is first a set up) is done using a technique adapted from a datagram-based routing algorithm based on gradient projection methods [3]; related approaches towards the initial routing of VC's are considered in [2,8]. The CODEX algorithm can also dynamically reroute a set of probabilistically-determined existing VC's. In their implementation of path rerouting, a secondary path is established while data keeps flowing on the primary path. Finally, we note that a number of networks ([14]) dynamically reroute VC's when a failure of the initial route occurs. We will not consider such rerouting here and instead focus on the issue of rerouting for the purpose of performance improvement alone.

The remainder of this paper is structured as follows. In section 2 we define the network models which will be used to study different routing and re-routing policies. These policies and the manner in which they use different network status information are then described in section 3. The theoretical analysis of a particular dynamic rerouting policy is then considered in detail in section 4. Simulation results for all routing policies are provided in the following section. Section 6 contains our conclusions.

2 Network Models

In studying the VC routing and rerouting problem, we will focus on a single source/destination pair and the various paths through the network between these two nodes. Given that both the number of VC's between a source destination pair as well as the number of outstanding (unacknowledged) packets along a path will be considered in various routing policies, it is clear that both the number of virtual circuits being routed over each of the N paths (indicated by nv_i in Figure 1) as well as the number of queued packets at each of the hops on each path must be considered in analyzing the performance of various VC routing and re-routing policies. The generic network model shown in Figure 1 attempts to illustrate these two aspects of the models by showing the VC-level aspects of the model in dashed lines, while the packet-level aspects of the model are shown in solid lines.

Each individual path through the network is represented as a series of tandem queues and the interfering traffic at each hop is abstracted out of the model by the standard modeling technique of appropriately modifying the service rate (see, e.g. [13]). We consider here only a homogeneous

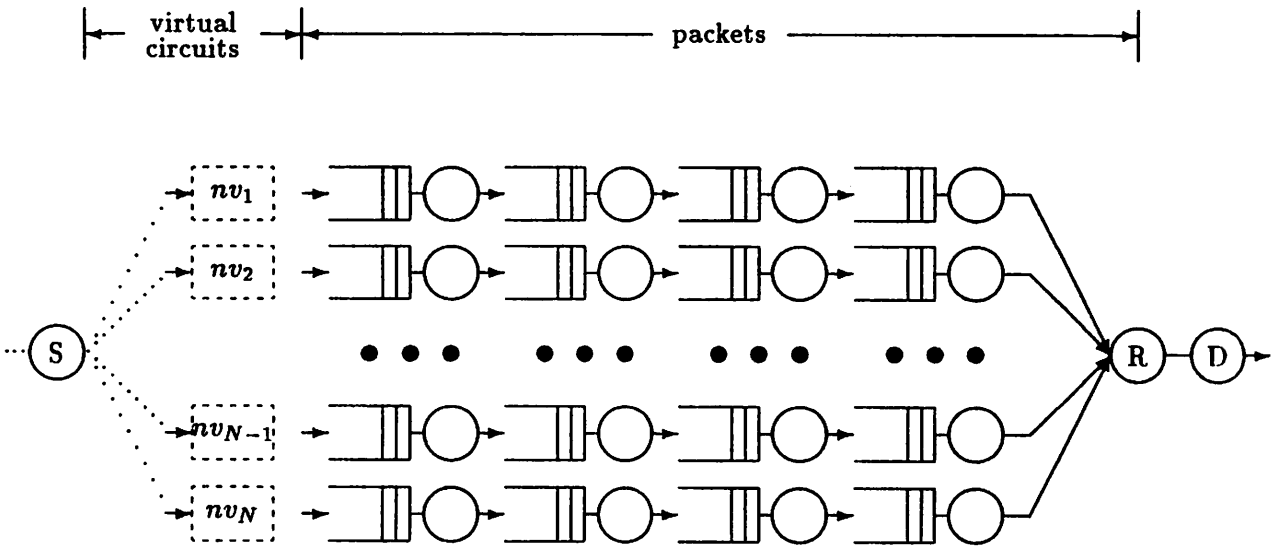

Figure 1: Virtual Circuit Model with 4 hop paths

network, in which case the service rates at the various hops may be considered identical, and the case in which all paths are routed over an independent set of K links. Although this model is idealized, our aim here is not to provide absolute measures of performance but rather to study the more fundamental issue of the effects of using different network status information in the routing and rerouting of VC's.

At the virtual circuit level, VC call setup requests arrive to a source node (labelled S in Figure 1) according to a Poisson process of rate λ_v and are initially assigned to one of the N paths between the source and destination nodes by the VC routing algorithm. Once an arriving VC has been assigned to a path, all packets generated by the VC will be constrained to follow this path until either the VC is rerouted or the VC is terminated. In the case of rerouting, all packets generated after a VC is rerouted will follow the newly selected path, while packets previously generated by the VC which have not yet reached the destination node will continue along the previous path. As discussed above, this may thus require a *reordering* buffer at the destination (labelled R in Figure 1).

Within each virtual circuit, a geometrically distributed number of packets (with mean G) are assumed to be generated during the VC's lifetime. The interarrival time between packets generated by a given VC is taken to be exponentially distributed with mean $1/\lambda_p$. The packet-level traffic being offered to each *path* thus consists of the superposition of nv_i independent Poisson processes, each with rate λ_p, where nv_i is the number of virtual circuits currently assigned to path i.

Finally, packet lengths will be modeled as being exponentially distributed with a mean of

$1/\mu_p$. Also, packet loss and corruption are not considered here and the reordering buffer at the destination node (which is used to reorder packets *within a rerouted virtual circuit* that arrive out of order) as well as the packet buffers at the intermediate nodes are assumed to be infinite.

3 Specification of Routing Policies

We will study five different routing policies and compare their relative performance under different models. In the first three policies, different information is used to select paths for new VC's when they first arrive at the source node. Once a VC is assigned a path, all traffic generated by the VC is constrained to follow that path until the VC terminates. While these policies do not permit dynamic rerouting of existing VC's, an obvious advantage is that packets are sent and received in order; hence no resequencing is required. The last two policies permit dynamic *rerouting* of a VC by possibly shifting it to a more lightly loaded path through the network.

We first consider the policies which do not reroute existing VC's; these policies (with the exception of Random) are appended with "_NS" to indicate that No Shifting of ongoing VC's is considered.

1. **RANDOM policy** : This simple policy does not require any information from the network. Under RANDOM, a newly arriving virtual circuit is assigned at random to one of the possible routes between the source and destination. We are interested in studying the RANDOM policy as it provides a baseline against which other policies may be measured.

2. **VCNUM_NS policy** : Under this policy, a newly arriving virtual circuit is assigned to the path that currently has the smallest number of on-going virtual circuits between the source-destination pair. If there are more than two routes with the same minimum number, one is chosen at random.

3. **PKQL_NS** : Under this policy, a newly arriving virtual circuit is assigned to that route which has the smallest number of unacknowledged packets. Assuming that acknowledgement delays are small with respect to message transmission delays, this policy may be thought of as assigning a newly arriving VC to that path having the smallest instantaneous sum of queue lengths along the various hops on the path.

   In the case of multiple hops, we will consider two versions of this policy. In the first version, only the queue length of the first (nearest to the source node) hop on each route is used in assigning a newly arriving virtual circuit to a path. In the second policy, we consider the sum of the queue lengths, as discussed above.

The two dynamic rerouting policies below differ (as above) in whether they use instantaneous queue length information or the number of existing VC's in rerouting an ongoing virtual circuit. In both cases, virtual circuits are considered for rerouting only when an existing VC terminates. The dynamic policies listed below are appended with "_S" to indicate that ongoing VC's may be Shifted to other paths.

1. **VCNUM_S** : Under this policy, the number of ongoing virtual circuits currently being routed over each of the paths is used by the source node to both initially assign an arriving VC to a path as well as to reroute an ongoing virtual circuit. A virtual circuit may be interrupted and rerouted to another path according to the following rules when some other ongoing VC terminates. When one virtual circuit terminates on a given path, a VC will be rerouted to the given path from another path which has at least two more on-going virtual circuits than the path on which the VC just terminated. As discussed earlier, the goal of rerouting ongoing VC's is to more evenly redistribute the network load, thus reducing delays. The price to be paid is the call teardown/setup times and possible reordering delays at the destination.

2. **PKQL_S** : Under this policy a VC can again only be rerouted from another path to a given path when a virtual circuit terminates on the given path. In this policy, however, the decision is based on the number of packets queued at the various hops along each of the paths. As in PKQL_NS, we again have two versions for this policy.

4 Approximate Analysis of the VCNUM-S policy

In this section, we present an approximate analysis of the VCNUM_S policy using a simple network model with N 1-hop paths from source to destination; extensions to the model are discussed at the end of the section. Recall that under the VCNUM_S policy, a newly arriving virtual circuit is assigned or rerouted to that path that currently has the smallest number of ongoing (existing) virtual circuits.

The following assumptions are made in our mathematical analysis; the impact of these assumptions will be discussed when our analytic results are compared with those obtained via simulation.

- When one VC terminates on a path, another VC may be *immediately* shifted from another path (according to the criteria defined in section 3). This policy thus results in an important property that will be exploited in our analysis : *the difference between the number of ongoing virtual circuits on any two paths is at most one.*
- No virtual circuit set up time or shutdown time is considered.

- Resequencing delay is not modeled.
- The maximum number of virtual circuits that a path can have is limited by M. If this number is in reality unbounded, we can in practice choose M large enough so the probability of a virtual circuit arrival causing this bound to be exceeded is arbitrarily small. This will ensure that our network model effectively accepts an infinite number of virtual circuits.

In order to obtain an exact description of the system behavior, the state description of the system should include the number of on-going virtual circuits and the number of packets on each path. The size of the system state space, however, makes this approach intractable. In our analysis, we thus *decompose* this complex model into two simpler models. In the first model, we analyze the rerouting of virtual circuits in order to obtain the distribution of the number of virtual circuits in the *system,* i.e., on all the paths between the source/destination pair. We then focus on a single path through the network and then use the results of the network-level analysis to solve for the distribution of the number of virtual circuits and the number of packets on a *single path* in the network.

4.1 System State

We first solve for the distribution of the number of virtual circuits in the system (i.e., between the source/ destination pair) as follows. Since virtual circuits are shifted dynamically to make the number of VC's on each path as balanced as possible, *the difference between the number of virtual circuits on any two paths is at most one.* Therefore, if we know there are exactly i paths from the source to the destination with X virtual circuits, then each of other $N - i$ paths can have either $X - 1$ virtual circuits or $X + 1$ virtual circuits. Furthermore, if we know X is the maximum number of virtual circuits along a path in the network, each of the other $N - i$ paths must have exactly $X - 1$ virtual circuits.

As previously stated, the VC arrival process is modeled by a Poisson process with rate λ_v and thus the interarrival times of calls are exponentially distributed. The call departure process is more complex. Recall that each call generates a geometrically distributed number of packets (with mean G), each having an exponentially distributed *packet* interarrival time with mean $1/\lambda_p$. Thus the call holding time is a geometrically distributed sum of exponentially distributed interarrival times with a mean of G/λ_p. We will model this holding time as if it were exponentially distributed. (We note that we could have alternatively specified this as our VC model in the first place, namely that calls have exponential holding times and generate packets at a Poisson rate). Based on these observations, we can define a "system-level" level state (which will model the network shown in Figure 1 at the VC level) as a two-dimensional Markov chain (X, i), $0 \leq X \leq M$ and $1 \leq i \leq N$, where M is the maximum number of virtual

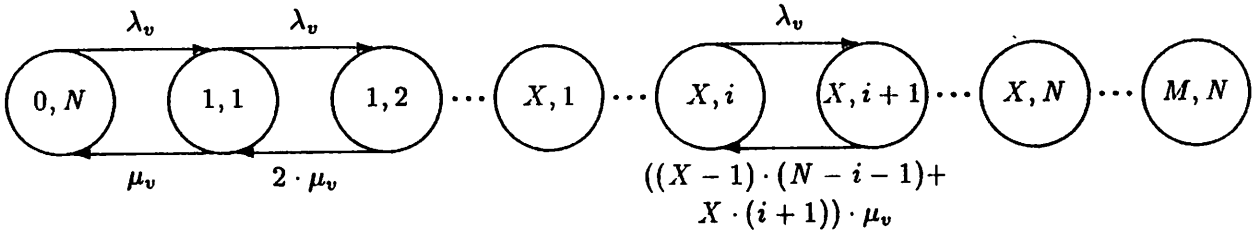

Figure 2: Markov Chain for the Virtual Circuit Population

circuits allowed between the source and destination node on any one path and N is the number of paths.

Let state (X, i) represents the case in which there are i paths from the source to the destination with X virtual circuits and $(N - i)$ paths with $X - 1$ virtual circuits. Hence the total number of virtual circuits in the system is $X \cdot i + (X-1) \cdot (N-i)$. Note that the total number of states is $M \cdot N + 1$ since when X is 0, i must be N. Note also that since the difference between the number of virtual circuits on any two paths is at most one, a VC arrival will cause a state transition from (X, i) to $(X, i+1)$, if $i < N$, or from (X, N) to $(X+1, 1)$. A more subtle point (discussed below) is that given the rerouting of ongoing virtual circuits, a departing VC on *any* path will unconditionally cause a transition from $(X, i+1)$ to (X, i), provided that $i < N$, or from $(X+1, 1)$ to (X, N), otherwise.

The above considerations result in the state transition diagram shown in Figure 2. The transition rate matrix, A, can be calculated as follows :

1. Whenever a virtual circuit arrives, it will always join one of the paths that has a minimum number of virtual circuits:

$$A((X, i) \rightarrow (X, i+1)) = \lambda_v \quad 1 \leq i \leq N-1$$

or

$$A((X, N) \rightarrow (X+1, 1)) = \lambda_v$$

where λ_v is the virtual circuit arrival rate.

2. When a virtual circuit terminates, there are three possible scenarios:

   (a) A virtual circuit from one of the paths with a larger number of virtual circuits terminates. In this case, no dynamic rerouting is performed. The path upon which the VC has terminated becomes a path with a fewer number of virtual circuits.

(b) A virtual circuit from one of the paths with a fewer number of virtual circuits terminates. In this case, a virtual circuit from one of the paths with a larger number of virtual circuits will be shifted to this path. Note that the overall effect (after VC termination and rerouting) is as if a VC had initially terminated on one of the paths with a higher occupancy. Thus, the transition rate for cases (a) and (b) combined is:

$$A((X,i) \rightarrow (X,i-1)) = (X \cdot i + (X-1) \cdot (N-i)) \cdot \mu_v \quad 2 \leq i \leq N-1$$

or

$$A((X,1) \rightarrow (X-1,N)) = (X + (X-1) \cdot (N-1)) \cdot \mu_v$$

where $\mu_v = \frac{\lambda_p}{G}$.

(c) When every path has the same number of virtual circuits, a VC termination will not cause rerouting and we have:

$$A((X,N) \rightarrow (X,N-1)) = (X \cdot N) \cdot \mu_v$$

We observe that the VC-level behavior of this system is exactly the same as that of an $M/M/(N \cdot M)/(N \cdot M)$ queue with arrival rate λ_v and service rate $\mu_v = \frac{\lambda_p}{G}$. From [5], we solve the following equations :

$$P(K) = P(K-1) \cdot \frac{\lambda_v}{K \cdot \mu_v} \quad (1)$$

$$1 = \sum_{K=0} N * MP(K) \quad (2)$$

and compute the distribution of the number of virtual circuits in the system.

4.2 Local State

After solving for the distribution of the total number of virtual circuits in the system, the next step is to solve for the joint distribution of virtual circuits and packets on a single path. Since the network is homogeneous, all paths are identical and we can approximate the behavior of the system as a whole by the behavior N coupled but identical subcomponents, with each path constituting a subcomponent. In our finer-grain analysis, *we may thus focus on a single path through the network, using the results from the network-level analysis provided above, to solve for the joint distribution of the number of packets and the number of virtual circuits in a path.*

We describe the "local" state of the path under study by (n_q, n_a) where :

n_q is the number of packets *queued* on this path. $0 \leq n_q < \infty$.

n_a is the number of *active* virtual circuits on the single path under study, $0 \leq n_a \leq M$.

4.3 Generator Matrix Calculation

Arranging our local states in increasing order of n_q, and for each value of n_q in increasing order of n_a, the infinitesimal generator matrix, Q, of this continuous time Markov chain can be calculated as follows. In the analysis below, one should carefully note the distinction between the two dimensional *system-level state* (X, i) and the two dimensional *local-level state* (n_q, n_a).

First, we define $ps(X)$ as the probability that there are X virtual circuits on the specific path under study. Using the distribution of system states calculated above, $ps(X)$ can be computed as follows:

$$ps(X) = \sum_{i=1}^{N} \frac{i}{N} \cdot P(X,i) + \sum_{i=1}^{N-1} \frac{N-i}{N} \cdot P(X+1,i) \tag{3}$$

With the boundary conditions that $P(0,1) = P(0,2) = \ldots = P(0,N-1) = P(M+1,1) = P(M+1,2) = \ldots = P(M+1,N-1) = 0$. This equation results from the fact that the only possible system states that can result in there being X virtual circuits on the path under study are $\{ (X,1),(X,2),\ldots,(X,N),(X+1,1),(X+1,2),\ldots,(X,N-1)\}$. If the system is in state (X,i), the probability that there are X virtual circuits on this path is $\frac{i}{N} \cdot P(X,i)$. If the system is in states $(X+1,i)$, then the probability of having X virtual circuits on this path is $\frac{N-i}{N} \cdot P(X+1,i)$.

We can now define the generator matrix, Q, and focus on the different transitions that are possible in this Markov Chain. In the descriptions below, we condition on the fact that $n_a = X$.

- **Packet transmission completion.**

$$Q((n_q, n_a) \rightarrow (n_q - 1, n_a)) = \mu_p \tag{4}$$

- **A newly arriving virtual circuit joins this path.** Recall that the policy for assigning a newly arriving virtual circuit is to assign it to the path with the least number of virtual circuits; if there is a tie, then one path is randomly chosen. Therefore, if the *system* is in system states $(X,1),\ldots,(X,N-1)$, there is no chance for the path under study to receive a new virtual circuit, since it already has the maximum number of virtual circuits as compared to all other existing paths. However, if the system is in state (X,N), the

path under study has the probability of $\frac{1}{N}$ of receiving a new virtual circuit (the value of $\frac{1}{N}$ results from the fact that the path under study must "compete" with other $N-1$ paths).

If the system is in one of the system states $(X+1,i), i = 1,\ldots,N-1$, the path under study has a smaller number of virtual circuits as compared to some other existing paths and can potentially receive the newly arriving virtual circuit. However, since there are also $N-i-1$ paths with the same minimum number of virtual circuits, the path under study competes with those $N-i-1$ paths to receive any newly arriving virtual circuit. We also note that in our model, whenever a virtual circuit is assigned to a path, a packet is generated immediately; i.e., whenever a virtual circuit joins a path, a packet is also generated on that path immediately. Therefore,

$$\begin{aligned} Q((n_q,n_a)\rightarrow(n_q+1,n_a+1)) &= \frac{\lambda_v\cdot(\frac{1}{N}\cdot P(X,N)+\sum_{i=1}^{N-1}\frac{1}{N-i}\cdot\frac{N-i}{N}\cdot P(X+1,i))}{ps(X)} \\ &= \frac{\lambda_v\cdot\frac{1}{N}\cdot(P(X,N)+\sum_{i=1}^{N-1}P(X+1,i))}{ps(X)} \end{aligned} \tag{5}$$

- **A virtual circuit departs or is shifted out.** According to the dynamic rerouting policies defined in section 3, a virtual circuit may be interrupted and rerouted to another path upon termination of an ongoing virtual circuit on another path. Therefore when a virtual circuit departs from the path under study, it can either decrease the number of virtual circuits on this path by one or cause a virtual circuit from another path to be shifted in (leaving the total number of VC's on this path unchanged). Given there are X ongoing VC's on the path under study, a virtual circuit can terminate on this path and cause the number of on-going VC's on this path to decrease by one only if the system is in one of the system states $(X,1),(X,2),\ldots,(X,N)$; note that a fraction i/N of these system states would actually correspond to the case in which the VC path under study had X ongoing VC's. We let t be the transition rate associated with these events. Then:

$$t = \frac{X\cdot\mu_v\cdot\sum_{i=1}^{N}\frac{i}{N}\cdot P(X,i)}{ps(X)} \tag{6}$$

A virtual circuit on the path under study may be *shifted* out if a virtual circuit terminates on another path elsewhere which has fewer virtual circuits than the path under study. However, there are $i-1$ paths with the same number of virtual circuits as the path containing the VC under study. Therefore, when a VC terminates on a path with a smaller number of VC's, with probability $\frac{1}{i}$, a virtual circuit is shifted out to that path from the path under study. We also observe that it is possible to shift a virtual circuit to another path only when the system is in one of the system states $(X,1),\ldots,(X,N-1)$.

Let t1 be such transition rate, then

$$t1 = \frac{\sum_{i=1}^{N-1} \frac{(X-1)\cdot(N-i)\cdot\mu_v}{i} \cdot \frac{i}{N} \cdot P(X,i)}{ps(X)}$$
$$= \frac{\frac{(X-1)\cdot\mu_v}{N} \cdot \sum_{i=1}^{N-1}(N-i)\cdot P(X,i)}{ps(X)} \quad (7)$$

Therefore, combining the rates at which VC's depart from the path under study, we have

$$Q((n_q,n_a)\rightarrow(n_q,n_a-1)) = \frac{X\cdot\mu_v\cdot\sum_{i=1}^{N}\frac{i}{N}\cdot P(X,i)}{ps(X)} + \frac{\frac{(X-1)\cdot\mu_v}{N}\cdot\sum_{i=1}^{N-1}(N-i)\cdot P(X,i)}{ps(X)} \quad (8)$$

- **A packet is generated or a virtual circuit is shifted in from another path.**

  Since we assume each virtual circuit is independent, the packet arrival rate is $X\cdot\frac{G-1}{G}\cdot\lambda_p$ if the virtual circuit is not shifted out immediately after a packet generated. However, there is a possibility of shifting out a virtual circuit and whenever a virtual circuit is shifted out, a packet generated by this virtual circuit will also be shifted out with this virtual circuit. Therefore, we need to discount the original rate by the shift out rate we discussed in equation (7). That is, the actual packet arrival rate is $X\cdot\frac{G-1}{G}\cdot\lambda_p - \frac{\frac{(X-1)\cdot\mu_v}{N}\cdot\sum_{i=1}^{N-1}(N-i)\cdot P(X,i)}{ps(X)}$.

  In computing the rate at which virtual circuits are shifted in, we know that a VC will be shifted in only when the system is in one of the system states $(X+1,1),\ldots,(X+1,N-1)$ and a virtual circuit terminates on the path under study. Therefore, the shift in rate is $\frac{X\cdot\mu_v\cdot\sum_{i=1}^{N-1}\frac{N-i}{N}P(X+1,i)}{ps(X)}$.

  Hence, we have the following equation :

$$Q((n_q,n_a)\rightarrow(n_q+1,n_a)) = X\cdot\frac{G-1}{G}\cdot\lambda_p - \frac{\frac{(X-1)\cdot\mu_v}{N}\cdot\sum_{i=1}^{N-1}(N-i)\cdot P(X,i)}{ps(X)} + \frac{X\cdot\mu_v\cdot\sum_{i=1}^{N-1}\frac{N-i}{N}P(X+1,i)}{ps(X)} \quad (9)$$

- The diagonal elements of the Q matrix must be chosen to make the row sums equal to zero.

Using matrix-geometric solutions presented in [11,10], Q may be block partitioned as follows, where each entry in Q is a $(M+1) \times (M+1)$ square matrix.

$$Q = \begin{bmatrix} B_0 & A_0 & 0 & 0 & 0 & 0 & 0 & \dots \\ B_1 & A_1 & A_0 & 0 & 0 & 0 & 0 & \dots \\ 0 & A_2 & A_1 & A_0 & 0 & 0 & 0 & \dots \\ 0 & 0 & A_2 & A_1 & A_0 & 0 & 0 & \dots \\ & \dots & \dots & & & & & \end{bmatrix}$$

Partitioning our state space $\prod = [(0,0),(0,1),...,(0,M),(1,0),...,(1,M),...]$ as $[\prod_0, \prod_1, \prod_2, ...]$, where $\prod_i$ is an $(M+1)$ row vector corresponding to the set of states $(i,j); 0 \leq j \leq M$, $\prod_i$ can be expressed as [11,10]:

$$\prod_i \quad = \quad \prod_0 R^i, \quad \text{i=0,1,2,...} \tag{10}$$

where R is the unique nonnegative solution with spectral radius less than 1 of the matrix quadratic equation :

$$R^2 A_2 + R A_1 + A_0 \quad = \quad 0. \tag{11}$$

R can be solved by successive substitution of the following equations [11]:

$$\begin{aligned} R_0 &= 0 \\ R_{n+1} &= -A_0 A_1^{-1} - R_n^2 A_2 A_1^{-1}, \quad \text{n=0,1,2,....} \end{aligned} \tag{12}$$

For stable systems, the sequence will converge to R. Then let $B[R] = \sum_{k=0}^{\infty} R^k B_k = RB + B$. From [10], we can solve $\prod$ by using

$$\prod_0 B[R] \quad = \quad 0 \tag{13}$$

Finally, we normalize $\prod_i$ by letting $\sum_{i=0}^{\infty} \prod_i = 1$. Since the $\prod_i$ sequence is an infinite series, from a computational standpoint we will need to cut this sequence at some appropriate point n_π.

4.4 Performance Measures

The queue length for the single hop on the path can be computed by :

$$QL \quad = \quad \sum_{i=0}^{n_\pi} \sum_{j=0}^{M} i \cdot \pi_{i,j} \tag{14}$$

and throughput of the path is :

$$S = \sum_{i=1}^{n_\pi} \sum_{j=0}^{M} \pi_{i,j} \tag{15}$$

Using Little's result in [9], the average packet delay D is given by

$$D = \frac{QL}{S} \tag{16}$$

We note that multiple hops can be easily included in the problem formulation by adding additional (infinite) dimensions to the state space and solving the resulting chain by standard numerical techniques.

5 Results

In this section we present simulation and analysic results comparing the performance (average packet delay) of the five VC routing and rerouting policies discussed in the previous sections.

We first consider a model with N one-hop paths between source and destination. The set up delay for rerouting a virtual circuit is ignored in this basic model but the resequencing algorithm is simulated at the destination node for the two dynamic rerouting policies. These initial sets of performance results are based on the following parameter settings :

- μ_p, the mean of packet transmission time, is normalized to one.
- λ_p, the arriving rate of packets in each active virtual circuit, is set to 0.2 *packets/time unit*.
- G, the mean number of packets generated by a virtual circuit, is set to 50.
- Two sets of simulations are performed by setting N, the number of paths in the network, to 4 and 20 respectively.

Each simulation was run for 1,000,000 units of service time and the method of independent replications was used to generate the 95% confidence interval [7]. The duration of the transient phase is estimated to be 100,000 units of service time. Throughput, average queue length, and average packet delay were measured for each policy for different virtual circuit arrival rates, λ_v. Note that with the same virtual circuit arrival rate, all policies share the same average throughput because the average packet arrival rates are the same. Figure 3 plots the average packet delay (including resequencing) as a function of *packet* throughput for the case of four one-hop paths; Figure 4 shows the results for the case of 20 alternate paths between the source

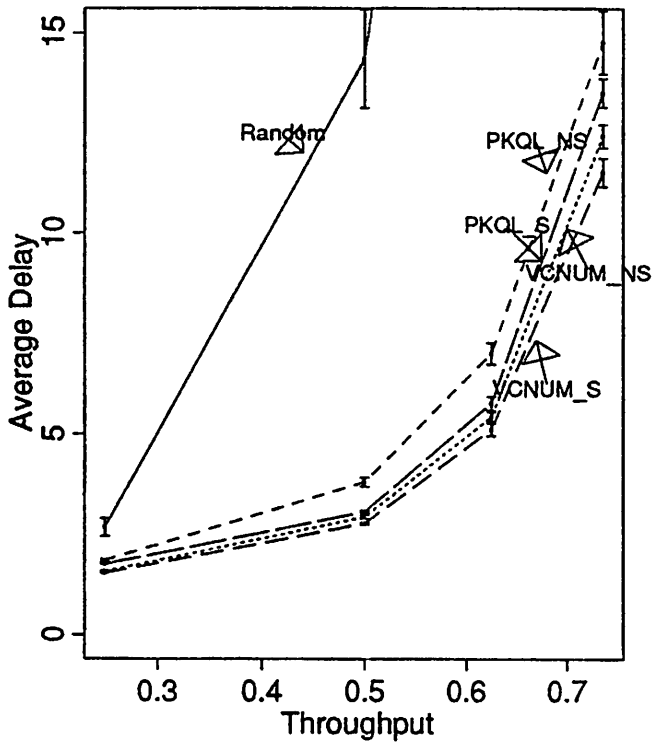

Figure 3: The single hop case: $N = 4, G = 50$.

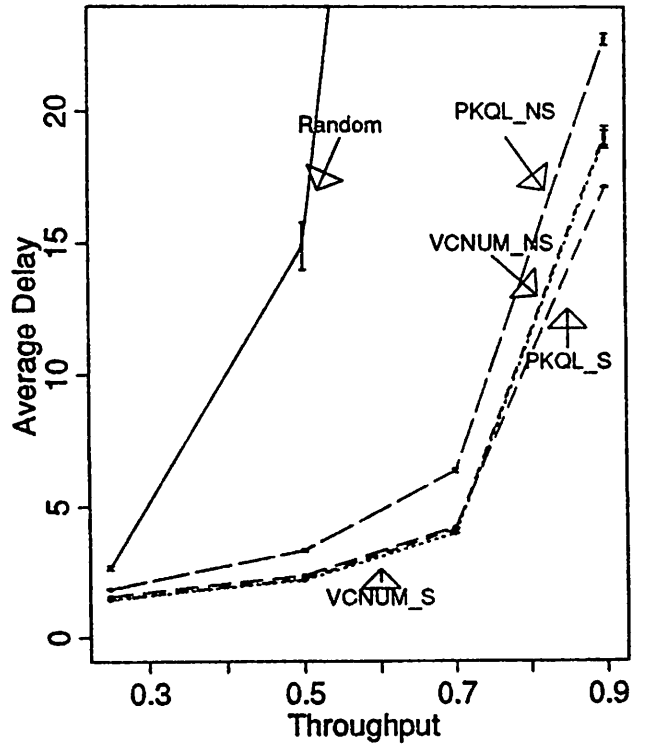

Figure 4: The single hop case: $N = 20, G = 50$.

and destination node. The vertical lines about each point indicate the 95 percent confidence interval.

Based on Figures 3 and 4, we can make the following observations:

- By comparing the RANDOM policy to four other routing policies, it is seen that all four policies that use dynamic state information in routing and rerouting VC's lead to much lower delays.
- By comparing the two non-rerouting policies to the two dynamic rerouting policies, dynamic rerouting policies lead are seen to give better performance. However, the difference is not very significant in low and middle traffic regions.
- Comparing the VCNUM_NS policy to the PKQL_NS policy, we note that the number of virtual circuits on each path is a better routing indicator than the number of unacknowledged packets on each path. In fact, we even see that the performance of the non-shifting VCNUM_NS policy is better than that of the more dynamic shifting policy, PKQL_S, for low and medium levels of network utilization.
- The VCNUM_S policy performs better than PKQL_S policy for low and middle traffic rates. However, for the case of $N = 20$, the PKQL_S policy becomes a better choice. We conjecture that the the reason for this result is that shifting VC's based on the occupancy at the intermediate queue can predict immediate delay more accurately and hence is more suitable for a high traffic situation. We conjecture that we would see the same behavior for $N = 4$, although this would not occur until higher loads.

Figure 5 compares the analytic results from section 4 with those obtained via simulation. We note that the analytic and simulation results agree quite closely, yielding results within 5% of each other. In our analytic model, we assumed that the rate at which virtual circuits were shifted in and out was Poisson; while we conjecture that this assumption is as asymptotically correct as the number of routes, N, grows large, it is not valid for small N [6]. We also conjecture that the assumption of exponential session holding times may also be contributing to the slight difference between simulation and analysis.

In our next set of results, we consider case in which each path consists of four hops (as shown in Figure 1). Set up delay for rerouting a virtual circuit is also ignored in this model but resequencing is simulated at the destination node. Recall that in the multihop case, we have two versions of the PKQL_NS and PKQL_S policies. In the first case, the queue length of the first hop on each path is used for making the decision of assigning or rerouting a virtual circuit. In the second case, the sum of the queue lengths on each path is used for making the decision. In order to distinguish these four policies, two policies in the first case, which make a

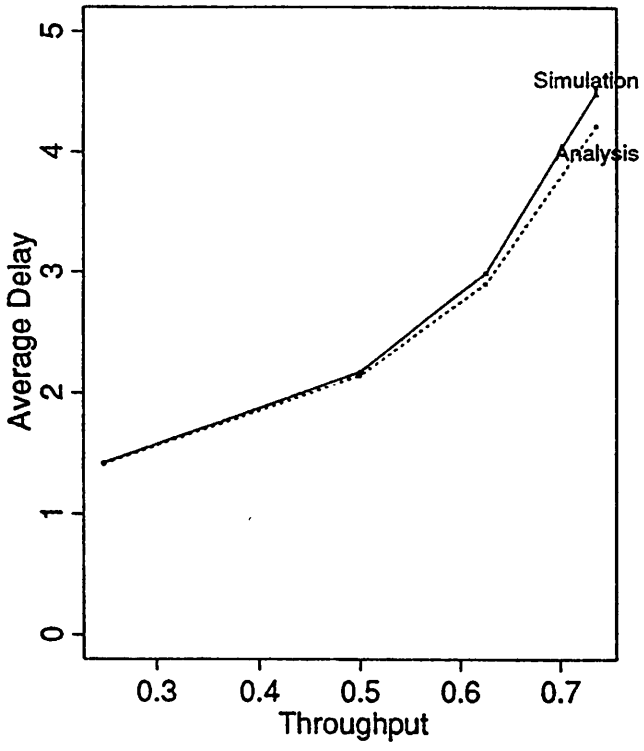

Figure 5: Simulation versus analysis: the one hop case, $N = 20$.

decision according to the queue length of the first queue, are called PKQL_NS1 and PKQL_S1 respectively. The other two policies are called PKQL_NS2 and PKQL_S2 respectively.

Three sets of simulations were performed to investigate the effect of different network configurations and different system parameters on the performance of the various policies. System parameters for these three models were set as in the previous model except:

- In the first set of simulations, a four-path, four-hop model with G equal to 50 is simulated.
- In the second set of simulations, a ten-path, four-hop model with G equal to 50 is simulated.
- In the third set of simulations, a ten-path, four-hop model with G equal to 10 is simulated.

The average packet delay versus throughput for these three models are shown in Figure 6, 7 and 8, respectively. The following observations may be drawn from these three figures.

- Dynamic routing policies still improve the performance significantly.
- Given a fixed network throughput, the average packet delay decreases as G decreases. This is due to the fact that when G decreases, the lifetime of a virtual circuit decreases and the total number of virtual circuits entering the network (in order to maintain the

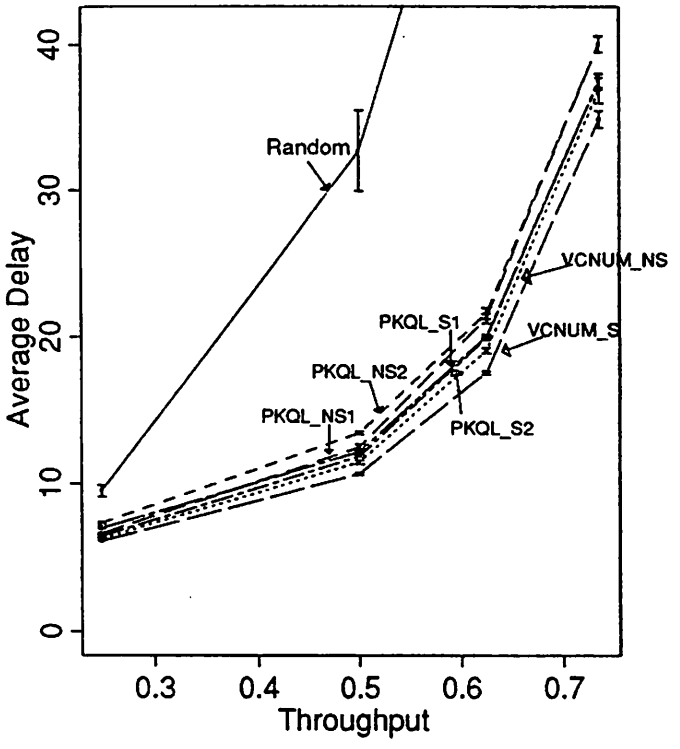

Figure 6: $N = 4, G = 50$, the four hop case

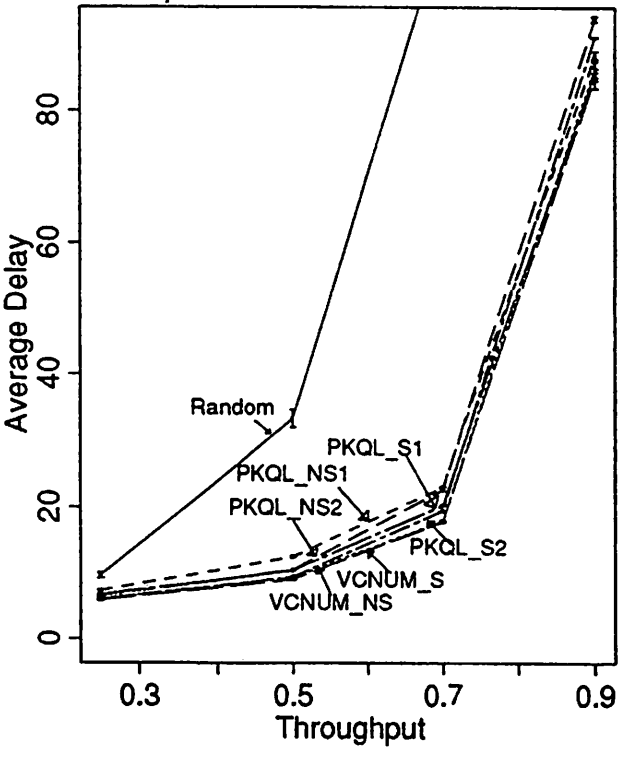

Figure 7: $N = 10, G = 50$, the four hop case

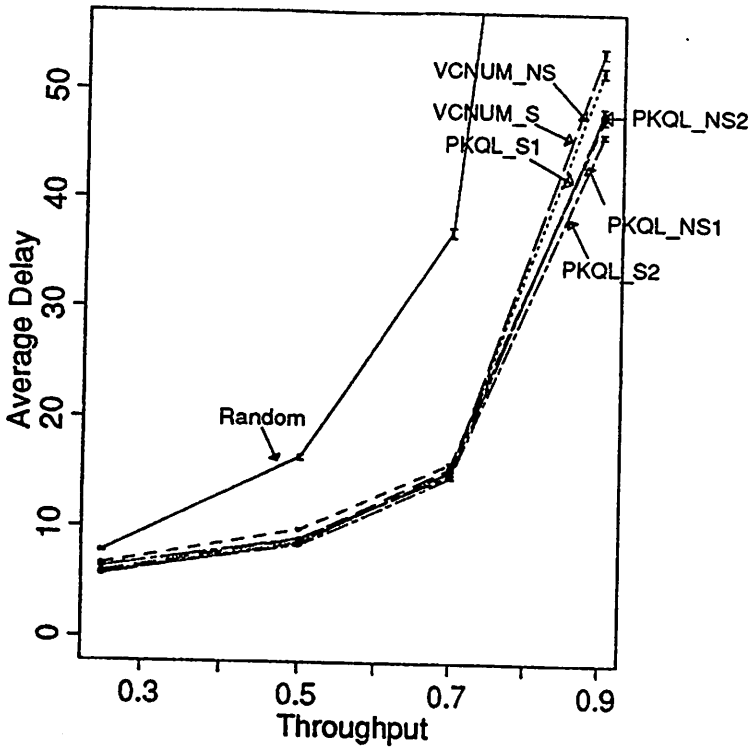

Figure 8: $N = 10, G = 10$, the four hop case

same link traffic load) increases, which implies that the system has more opportunities to balance its load; that is, VC-level load balancing is more effective as G decreases.

- When G is large, policies that are based on the number of virtual circuits on each path are superior to policies that are based on the number of unacknowledged packets. But the reverse is also true, that is, if G is small, then policies that are based on the number of unacknowledged packets are preferred. Thus, only in the case of short session holding times, is instantaneous queue length information a better predictor of future delays than VC-level state information.

- The second version of the PKQL_NS($PKQL\_S$) policy performs much better than the first version in low traffic. However, when the traffic increases, only a small difference exists between the two versions. More interestingly, when G is large, and traffic is high (i.e., queues are long), using the queue length at the first queue only is better than using the sum of the queue lengths along a path. We conjecture this results from the fact that when loads are high, the packets at the early hops along a VC are more likely to play a role in delaying packets sent into the VC than are packets towards the end of the VC.

Finally, let us consider the effects of call setup delay on the performance of the five policies. To simulate such delay, when a virtual circuit is rerouted from one path to another path, it can not transmit packets for a constant period of time, called the set-up delay time. During this

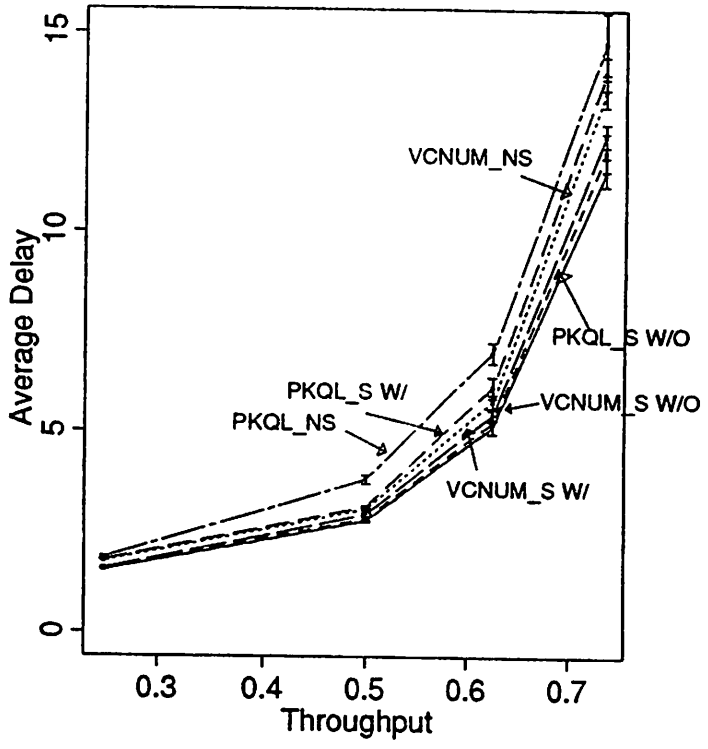

Figure 9: $N = 4, G = 50$, one hop with setup delay.

set-up delay time, all packets generated by this virtual circuit are buffered at the source node. At the end of this period, i.e., when the source node receives a rerouting acknowledgment from the destination node, the packets in the source node's buffer are transmitted in order on the *new* path. A resequencing mechanism is also simulated at the destination node. Both buffers in source node and destination node are assumed to be infinite. Only the VCNUM_S policy and the PKQL_S policy are simulated using this model, since only these two policies dynamically reroute virtual circuits. The results are compared to the results from the basic model in order to determine whether set-up delay can cause significantly larger packet delay. The system parameters are set as in the one-hop model except that there is an additional set-up delay parameter which is set to 3.6 units of service time. Figure 9 shows the performance results.

Two facts can be observed from this figure.

- Adding a set-up delay of the given magnitude does not significantly increase the average packet delay.
- After adding the set-up delay, the VCNUM_NS policy works almost as well as the VCNUM_S policy. However the PKQL_S policy still performs better than the PKQL_NS policy, even after adding the set-up delay.

6 Conclusion

In this paper, several policies for assigning virtual circuits to paths were studied within several network models; both simulation and analytic were developed to examine the performance of these policies. The problems of initial placement of VC's as well as dynamic rerouting of VC's were examined. In the latter case, the resequencing delay and set-up delay were also considered. Several conclusions can be gleaned from this study.

First, we find that when the expected lifetime of virtual circuits is relatively long, the number of virtual circuits on each path is a better indicator (for VC routing purposes) than the number of unacknowledged packets on each path; this is true even in the multi-hop model. Dynamically rerouting a virtual circuit on the basis of the number of virtual circuits on each path does decrease the packet delay; however it is not always significant. Since dynamically rerouting a virtual circuit incurs a resequencing problem and a set-up procedure, overall, it is questionable whether it is worthwhile to dynamically reroute VC's. From the results of our multi-hop model, we also found that the sum of all queue lengths on each path is not a significantly better indicator for routing decisions (and is sometimes worse) than the queue length of the first queue. Should considerably more effort be required by the source node in order to obtain the sum of all queue lengths on each path, a policy that just uses the queue length of the first queue would seem more favorable.

Finally, after considering set-up delay, dynamically rerouting a virtual circuit depending on the number of virtual circuits was found to perform almost as well as the non-rerouting policy. This confirms our conjecture that routing virtual circuits on the basis of the number of virtual circuits on each path, without dynamic rerouting, is a very simple policy and one with very good performance.

References

[1] D. Bertsekas and R. Gallager. Data networks. *Prentice Hall Publishing Company*, 1988.

[2] E. Gafni and D. Bertsekas. Path assignment for virtual circuit routing. *Proc. ACM SIGCOMM'84*, pages 21–25, 1984.

[3] R. Gallager. A minimum delay routing algorithm using distributed computation. *IEEE Transactions on Communications*, COM-25:73–85, 1977.

[4] J. M. Jaffe and A. Segall. Threshold design for dynamic routing. *IEEE Transactions on Communications*, 1986.

[5] Kleinrock. Queueing systems. *Vol 1, John Wiley & Sons*, 1975.

[6] J. F. Kurose and R. Chipalkatti. Load sharing in soft real-time distributed computer systems. *IEEE Transactions on Computers*, C-36 No. 8:993–1000, 1987.

[7] Stephen S. Lavenberg. Computer performance modeling handbook. *Academic Press*, 1983.

[8] Y. Lin and J. Yee. A distributed routing algorithm for virtual circuit data networks. *Proc. IEEE INFOCOM'89*, pages 200–207, 1989.

[9] J. Little. A proof of the queueing formula $l = \lambda \cdot w$. *Operation Research*, 9, 1961.

[10] M. F. Neuts. Matrix-geometric solutions in stochastic models – an algorithmic approach. *The Johns Hopkins University Press, Baltimore*, 1981.

[11] B. M. Rao and M. J. M. Posner. Algorithmic and approximation analyses of the split and match queue. *Commun. Statist. Stochastic Models*, 1(3):433–456, 1985.

[12] H. Rudin. On routing and delta routing : A taxonomy and performance comparison of techniques for packet-switched networks. *IEEE Transactions on Communications*, COM-24:43–57, Jan. 1976.

[13] M. Schwartz. Performance analysis of the SNA virtual route pacing control. *IEEE Transactions on Communications*, COM-30:172–184, 1982.

[14] Mischa Schwartz. Telecommunication networks. *Addison-Wesley Publishing Company*, 1987.

[15] A. Segall. The modeling of adaptive routing in data-communications networks. *IEEE Transactions on Communications*, COM-25:85–95, Jan. 1977.

[16] A. S. Tanenbaum. Computer networks. *Prentice Hall Publishing Company*, 1981.