Fast Time-Dependent Evaluation of Integrated Services Networks *

Ibrahim Matta, A. Udaya Shankar
Institute for Advanced Computer Studies and
Department of Computer Science
University of Maryland
College Park, Maryland 20742
imm@cs.umd.edu, shankar@cs.umd.edu

CS-TR-3235.1
UMIACS-TR-94-28.1
May 1995

Abstract

We present a numerical-analytical method to evaluate integrated services networks with adaptive routing, scheduling and admission controls. We apply our method to connection-oriented networks supporting different types of real-time connections. The network dynamics is described by difference equations which can be solved for both transient and steady-state performances. Results indicate that our method is computationally much cheaper than discrete-event simulation, and yields accurate performance measures. We compare the performance of different routing schemes on the NSFNET backbone topology with a weighted fair-queueing link scheduling discipline and admission control based on bandwidth reservation. We show that the routing scheme that routes connections on paths which are both under-utilized and short (in number of hops) gives higher network throughput.

A preliminary version of this paper appeared in *IEEE ICNP 94*

*This work is supported in part by ONR and DARPA under contract N00014-91-C-0195 to Honeywell and Computer Science Department at the University of Maryland, and by National Science Foundation Grants NCR 89-04590 and CCR-8811954. The views, opinions, and/or findings contained in this report are those of the author(s) and should not be interpreted as representing the official policies, either expressed or implied, of the Defense Advanced Research Projects Agency, ONR, the U.S. Government or Honeywell.

Contents

1 Introduction

Integrated services packet-switched networks, such as ATM (Asynchronous Transfer Mode) networks [45], are expected to support a wide variety of applications (e.g., multimedia, voice, mail) with heterogeneous quality-of-service (QoS) requirements. To meet these requirements, new algorithms have been proposed for controlling routing, admission, and scheduling. *Routing* provides a selection of routes, based on cost functions associated with the transmission links. *Admission* defines the criteria used to accept or reject a new incoming application, based on the service requested and the resources available. *Scheduling* defines how link resources (bandwidth, buffers, etc.) are allocated among the different services.

The overall end-to-end performance of the network hinges on the algorithms used in the routing, admission, and scheduling components. The algorithms are often *adaptive*, with parameters being varied dynamically according to service class and current or delayed system state information. Arrival and service statistics are often time-dependent. As a result, there is significant interaction among the three components.

The accurate and fast evaluation of such time-dependent systems is critical to their cost-effective design. Existing evaluation methods for these systems are inadequate. *Analytical methods are typically too coarse.* They usually assume steady-state conditions and do not account for adaptive policies and the effect of delayed feedback. Incorporating adaptive time-dependent behavior makes them analytically intractable and computationally expensive to solve numerically due to the large state space. *Simulation approaches are often too expensive.* They can handle realistic detail and dynamic situations, but they are invariably computationally prohibitive, especially for evaluating high-speed networks where the number of scheduled events (packets, connections, etc.) is usually enormous.

Our contribution

In this paper, we present a *numerical-analytical* method that yields the time-evolution of *instantaneous* performance measures. Our method takes into account the interaction and time-dependent nature of the control algorithms, and is computationally inexpensive. The numerical foundation of our method provides a modeling power close to that of simulation at a fraction of the computation expense, typically less expensive than simulation by many orders.

We apply our method to a model that permits the evaluation of a connection-oriented packet-switched network (e.g., ATM) that supports real-time communication (voice, video, etc.) by making use of various adaptive routing, scheduling and admission policies. Thus this model can be applied to achieve more comprehensive evaluation of existing strategies and to propose more effective network control schemes.

Among the main performance measures of interest are the *instantaneous end-to-end connection blocking probabilities*. To calculate them, we use the link decomposition technique [33, 19] to approximate the multi-link network as a collection of single-link networks. For each link, we approximate the relationship between the instantaneous *local* (or link-level) connection blocking probabilities and the instantaneous average numbers of established connections by the relationship at steady-state. The latter is available, usually in implicit form, from standard queueing theory [34]. We solve these instantaneous relationships iteratively [32]. After all single-link models have converged, we compute the instantaneous end-to-end connection blocking probabilities by invoking the link independence assumption.

To obtain the time behavior of the instantaneous end-to-end connection blocking probabilities, we introduce difference equations in the average numbers of established connections. These difference equations relate the instantaneous flow rates of departure and admission of connections. They can be solved iteratively in conjunction with the previous solution (in previous paragraph).

This allows the investigation of both transient and steady-state performances of various control schemes. We point out that our iterative procedure differs from iterations commonly used in steady-state analysis (e.g. [28, 30, 35, 13, 46, 9, 39, 21]), which only solve for steady-state measures. Our results indicate that our method is computationally much cheaper than discrete-event simulation, which requires the averaging of a large number of independent simulation runs to obtain reliable performance estimates. Furthermore, the performance measures it yields are very close to the exact values obtained by simulation.

In this paper, we present results for a network with NSFNET backbone topology, weighted fair-queueing link scheduling [43], admission control based on "effective bandwidth" [22], and three connection routing schemes. Two of these routing schemes adapt to delayed state information expressed in terms of link utilizations. Our results indicate that the routing scheme that selects paths which are both under-utilized and short (in number of hops) for routing new incoming connections gives higher network throughput.

In Section 2, we formulate our model. We present our solution procedure in Section 3. Sections 4 and 5 illustrate how our method can capture the effect of various control schemes. Section 4 discusses scheduling and admission, and Section 5 discusses routing. Numerical results to validate our method are presented in Sections 6 and 7. Section 6 contains results for single-link networks, and Section 7 for multi-link networks. Section 8 investigates three routing schemes on the NSFNET backbone topology. Section 9 concludes with related and future work.

2 Model

We consider networks of arbitrary topology supporting real-time communication using a connection-oriented reservation scheme. That is, before a real-time application (e.g., voice, video) can start transmitting its packets at the requested end-to-end QoS (e.g., delay), a connection has to be first established along a fixed physical route from the source node to the destination node. For this, the source node uses its routing information to choose a potential route to the destination node. (We use the terms "route" and "path" interchangeably.)

A connection setup message is then sent over this route, requesting a local QoS from each of its links such that the aggregate of these local QoS satisfies the connection's end-to-end QoS. If the request fails at any link due to lack of resources (or any other admission constraints), the connection is blocked and lost; it is assumed that it is not attempted on another (alternate) route. Otherwise, the connection is established and resources are allocated to it. At the end of transmission, this connection is torn down and resources are released.

Routing can be static or dynamic. For dynamic routing, we assume routing information is updated by periodic broadcasts by nodes of the status of their outgoing links during the last period. This periodic collection of status information is often used in routing algorithms proposed for integrated services networks (e.g., [1, 4, 11]). We assume that broadcasts of all nodes are synchronized; it will be apparent later that we can easily model unsynchronized broadcasts. We also assume that these broadcasts reach other nodes instantaneously; this is justifiable because the time to propagate routing information is small compared to the routing update period.

After each update, a node uses its new routing information to compute new routes to be used for incoming connections until the next broadcast. The routes are thus updated at discrete time instants $nT, n = 1, 2, \cdots$, where T is the routing update period. Without loss of generality, we assume $T = 1$.

We assume that a connection setup (and teardown) request on a multi-link route reaches all links of the route simultaneously; this is justifiable because connection setup (and teardown) times are small compared to the routing update interval and connection holding times.

Services

We think of the network as providing real-time services. A **service** represents connections with the same source-destination node pair and the same traffic and QoS parameters. The parameters of a service s include the following:

- Arrival rate of requests for a connection setup, $\lambda_s(t)$.
- Average lifetime of a connection from the time it is successfully established until it ends, $1/\mu_s(t)$.
- QoS requirements of a connection, for example, the end-to-end statistical delay bound $(D_s,\ \varepsilon_s)$ denoting that probability[end-to-end packet delay $> D_s$] $< \varepsilon_s$.
- Packet (or cell) generation characteristics of a connection, such as its mean transmission rate m_s and peak transmission rate M_s.

Classes

A connection of a service can potentially be established along any of the possible routes between the service's source node and the service's destination node. The **class** of a connection is defined by its service and the route it takes.

Figure 1 shows a network offering two services: service s1 from node 0 to node 4, and service s2 from node 1 to node 3. Each service has two possible routes for connection setup. Hence the network has four classes: classes c1 and c2 for s1 connections using route $\langle$ 0, 1, 2, 3, 4 $\rangle$ and $\langle$ 0, 5, 4 $\rangle$ respectively, and classes c3 and c4 for s2 connections using $\langle$ 1, 0, 5, 4, 3 $\rangle$ and $\langle$ 1, 2, 3 $\rangle$ respectively.

The instantaneous arrival rate of class-c connections of service s, denoted by $\lambda_c(t)$, is a function of $\lambda_s(t)$ and the routing algorithm. Note that with dynamic routing, class arrivals have time-varying statistics irrespective of whether the service arrivals have time-varying statistics.

Because a class is defined by the pair $\langle$ service, route $\rangle$, we can have a large number of classes, which may cause a computational bottleneck. To avoid this, we can restrict the set of possible routes, for example, to the shortest (in number of hops) and close to shortest paths.[1] This is

[1] Experiences with circuit-switched networks show that this restriction results in simple and efficient routing schemes [3, 38].

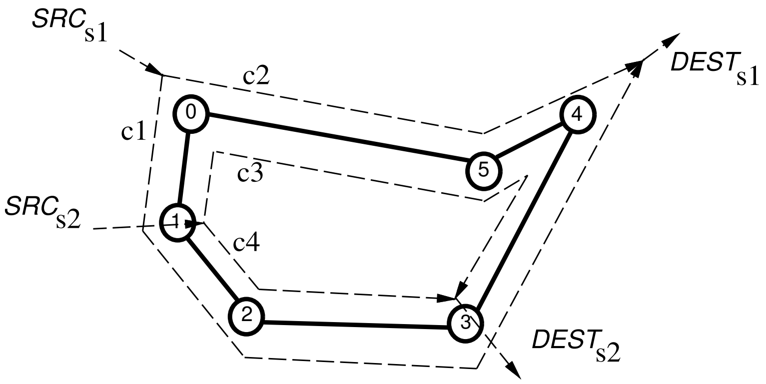

Figure 1: A network example.

acceptable because using a longer path for a connection ties up resources at more intermediate nodes, thereby decreasing network throughput. Furthermore, it also ties up more resources at each intermediate node because satisfying the end-to-end QoS requirement would require more stringent local QoS requirements. Section 5 addresses the selection of routes in more detail.

Obtaining class parameters at a link

Each link in the network is used by a subset of the classes. For example, in Figure 1, link $\langle 5, 4\rangle$ is used by two classes, namely c2 and c3. The parameters of a class at a link on its route are obtained from the parameters of its service. To do this, we make the following assumptions; some of these assumptions can be relaxed, possibly at additional computational cost:

- Connection setup requests arrive according to Poisson processes.
- The routing is probabilistic. That is, probabilities are assigned to the candidate paths and arriving connections are routed independently according to these path probabilities. With dynamic routing, the probabilities are periodically updated according to dynamic status information (e.g. measured load). Note that these probabilities could take the values 0 and 1 for single-path routing.
- For a connection setup request on a multi-link route, the requested end-to-end QoS is divided equally among the links. This is the so-called "equal allocation" policy. For example, if a connection of service s requesting an end-to-end QoS $(D_s,\ \varepsilon_s)$ is to be established on an h-link route, then we require that each link on the route guarantees a local requirement of $(\frac{D_s}{h},\ \frac{\varepsilon_s}{h})$ [41, 42].

- The packet generation characteristics of a connection established on a multi-link route do not change from link to link, i.e. remain the same as the given external characteristics.

The first assumption is often made and is reasonable in practice [20, 19, 44]. The second assumption uses a type of routing proposed in many studies (e.g., [4, 17]). The third assumption uses an end-to-end QoS allocation policy studied in [41, 23, 42].

The last assumption is valid in practice if the network admission control makes the same assumption, as for example, in the effective bandwidth approach by Guérin *et al.* [22]. It is also valid if the network uses a tightly-controlled approach that uses a non-work-conserving link scheduling discipline to reconstruct the traffic pattern at each link. An example of such approach is the Rate-Controlled-Static-Priority approach by Zhang and Ferrari [52]. Otherwise, the traffic pattern has to be characterized at each link as in [43, 12].

Given the above assumptions, it is straightforward to obtain the parameters of a class at a link. Consider, for example, the parameters of class c2 at link $\langle 0,\ 5\rangle$ or link $\langle 5,\ 4\rangle$. Connection setup requests arrive according to a Poisson process with rate $\lambda_{\mathrm{c2}}(t) = \alpha_{\mathrm{s1,c2}}(t)\ \lambda_{\mathrm{s1}}(t)$, where $\alpha_{\mathrm{s1,c2}}(t)$ is the (possibly dynamic) probability of a connection of service s1 being routed on class-c2 route. The average lifetime of a connection $\frac{1}{\mu_{\mathrm{c2}}(t)} = \frac{1}{\mu_{\mathrm{s1}}(t)}$. For an end-to-end QoS $(D_{\mathrm{s1}},\ \varepsilon_{\mathrm{s1}})$, the local QoS requirement $(D^j_{\mathrm{c2}},\ \varepsilon^j_{\mathrm{c2}}) = (\frac{D_{\mathrm{s1}}}{2},\ \frac{\varepsilon_{\mathrm{s1}}}{2})$ for $j \in \{\langle 0,5\rangle,\ \langle 5,4\rangle\}$, because the route of class c2 is two-hop long. The packet generation characteristics $(M_{\mathrm{c2}},\ m_{\mathrm{c2}},\ \cdots) = (M_{\mathrm{s1}},\ m_{\mathrm{s1}},\ \cdots)$.

Our solution procedure, by assuming link independence, makes use of the class parameters at each individual link to compute local performance measures, from which end-to-end measures are then computed. We describe this next.

3 Solution Procedure

The above model can be solved to obtain various instantaneous performance measures. We are mainly interested in calculating the end-to-end measures of each service. An intermediate step in this calculation is to compute the end-to-end measures of each of the service's classes. Among the main measures of class c are:

- $B_c(t)$, instantaneous blocking probability of class-c connections.

- $N_c(t)$, instantaneous average number of established class-c connections.

These measures depend on the performance seen by class-c connections at each link $j \in \mathcal{R}_c$, where $\mathcal{R}_c$ denotes the route of a class-c connection. In particular, we define the following:

- $B_c^j(t)$, instantaneous blocking probability of class-c connections at link $j \in \mathcal{R}_c$.
- $N_c^j(t)$, instantaneous average number of class-c connections established on link $j \in \mathcal{R}_c$. Note that $N_c^j(t) = N_c(t)$.

Then, assuming link independence, we have

$$B_c(t) = 1 - \prod_{j \in \mathcal{R}_c} [1 - B_c^j(t)] \qquad (1)$$

Let $\mathcal{C}^j$ be the set of all classes of connections using link j. The straightforward calculation of the time behavior of $\{B_c^j(t) : \; c \in \mathcal{C}^j\}$ involves solving the well-known Chapman-Kolmogorov differential equations for link j. However, these equations are extremely difficult to solve analytically [49], and computationally expensive to solve numerically [48]. Instead, we write the following difference equations for $c \in \mathcal{C}^j$, for time step $\delta \ll T$ (recall $T = 1$):

$$N_c^j(t+\delta) = [1 - \mu_c(t)\, \delta]\; N_c^j(t) \; + \; \delta\, \lambda_c(t) \prod_{j' \in \mathcal{R}_c} [1 - B_c^{j'}(t)] \qquad (2)$$

The first term in the right-hand side of equation (2) represents the average number of class-c connections established on link j which remain on link j (i.e. do not terminate). The second term represents the average number of new class-c connections established on link j during $[t, t+\delta)$.

Observe that if we could express $B_c^j(t)$ in terms of $\{N_{c'}^j(t) : \; c' \in \mathcal{C}^j\}$, then we could solve equations (2) and hence (1) inexpensively for the time behavior of the performance measures. Of course, obtaining such an expression is intractable and is probably equivalent to solving the original Chapman-Kolmogorov equations. *However it turns out that the instantaneous relationship between the $B_c^j(t)$ and the $N_c^j(t)$ is very well approximated by their relationship at steady-state*, i.e., by the relationship between the B_c^j and the N_c^j assuming that the $\lambda_c(t)$ and $\mu_c(t)$ are constants. The steady-state relationship is relatively easy to obtain. We obtain it implicitly as a fixed point of two steady-state expressions, one defining B_c^j in terms of $\{\frac{\lambda_{c'}}{\mu_{c'}} : \; c' \in \mathcal{C}^j\}$, and one defining $\frac{\lambda_c}{\mu_c}$ in terms of N_c^j and B_c^j. These two expressions are obtained next.

Steady-state blocking probability in terms of traffic intensities

Denoting such an expression by $\mathcal{S}_c^j$, we have for $c \in \mathcal{C}^j$:

$$B_c^j = \mathcal{S}_c^j(\{\frac{\lambda_{c'}}{\mu_{c'}} : \; c' \in \mathcal{C}^j\}) \qquad (3)$$

$\mathcal{S}_c^j$ can be obtained as follows. Define a *schedulable state* of link j to be a $|\mathcal{C}^j|$-dimensional vector representing the number of connections of each class $c \in \mathcal{C}^j$ that can be established simultaneously on link j, i.e. for which the local QoS is satisfied for every connection [27]. Denote the set of schedulable states by $\mathcal{F}^j$. $\mathcal{F}^j$ can be determined using a packet-level analysis knowing the parameters of each class at link j (obtained as shown in Section 2) and the link scheduling algorithm [22, 10]. Note that each link will typically have a different set of schedulable states because links have different capacities, are used by different sets of classes, etc. Section 4 illustrates the computation of $\mathcal{F}^j$ for a weighted fair-queueing scheduling [43].

The steady-state transition rates between the states of $\mathcal{F}^j$ are functions of the $\lambda_{c'}$ and $\mu_{c'}$. We obtain $\mathcal{S}_c^j$ by solving this Markov chain. In particular, denoting by $P(\sigma)$ the probability of being in a state $\sigma = (\sigma_1, \sigma_2, \cdots, \sigma_{|\mathcal{C}^j|}) \in \mathcal{F}^j$, we have

$$P(\sigma) = P(0) \prod_{c'=1}^{|\mathcal{C}^j|} \frac{(\lambda_{c'}/\mu_{c'})^{\sigma_{c'}}}{\sigma_{c'}!} \tag{4}$$

where $P(0) = [\sum_{\sigma \in \mathcal{F}^j} \prod_{c'=1}^{|\mathcal{C}^j|} \frac{(\lambda_{c'}/\mu_{c'})^{\sigma_{c'}}}{\sigma_{c'}!}]^{-1}$ is the normalization constant. This solution is valid not only for exponentially distributed connection lifetimes [28], but also for generally-distributed lifetimes [29].

Assuming a simple admission control where the arrival of a new class-c connection is blocked if its admission would lead to a nonschedulable state, we have

$$B_c^j = \sum_{\sigma \in \mathcal{F}^j} \mathbf{I}\{(\sigma_1, \cdots, \sigma_c + 1, \cdots, \sigma_{|\mathcal{C}^j|}) \notin \mathcal{F}^j\} \; P(\sigma) \tag{5}$$

where

$$\mathbf{I}\{(\sigma_1, \cdots, \sigma_c + 1, \cdots, \sigma_{|\mathcal{C}^j|}) \notin \mathcal{F}^j\} = \begin{cases} 1 & \text{if } (\sigma_1, \cdots, \sigma_c + 1, \cdots, \sigma_{|\mathcal{C}^j|}) \notin \mathcal{F}^j \\ 0 & \text{otherwise} \end{cases}$$

This is often referred to as "complete-sharing" admission control [27]. Note that $\mathbf{I}(.)$ defines the set of blocking states. Other admission control schemes can be modeled by alternative definitions of $\mathbf{I}(.)$.

Steady-state traffic intensity in terms of average number of connections and blocking probability

Equating the rates of departure and admission of class-c connections at link j, we have $\mu_c \; N_c^j = \lambda_c \, [1 - B_c^j]$. From this we have for $c \in \mathcal{C}^j$:

$$\frac{\lambda_c}{\mu_c} = \frac{N_c^j}{[1 - B_c^j]} \tag{6}$$

Instantaneous blocking probabilities in terms of average numbers of connections

From equations (3) and (6), we can express $B_c^j(t)$ approximately in terms of $\{N_{c'}^j(t) : \; c' \in \mathcal{C}^j\}$ by replacing the steady-state measures B_c^j and N_c^j by their instantaneous counterparts $B_c^j(t)$ and $N_c^j(t)$, and replacing $\frac{\lambda_c}{\mu_c}$ by an instantaneous quantity $z_c^j(t)$ that we introduce. Doing this yields the following instantaneous equations for $c \in \mathcal{C}^j$:

$$B_c^j(t) \;\; = \;\; \mathcal{S}_c^j(\{z_{c'}^j(t) : \; c' \in \mathcal{C}^j\}) \tag{7}$$

$$z_c^j(t) \;\; = \;\; \frac{N_c^j(t)}{[1 - B_c^j(t)]} \tag{8}$$

For fixed $\{N_c^j(t) : \; c \in \mathcal{C}^j\}$, we can solve equations (7) and (8) iteratively for $\{B_c^j(t) : \; c \in \mathcal{C}^j\}$. In particular, we can start with initial estimates $\{\hat{z}_c^j(t) : \; c \in \mathcal{C}^j\}$ and obtain $\{B_c^j(t) : \; c \in \mathcal{C}^j\}$ using equations (7). Then we use equations (8) to obtain new values for $\{z_c^j(t) : \; c \in \mathcal{C}^j\}$. We repeat this process until the values for $\{z_c^j(t) : \; c \in \mathcal{C}^j\}$ stabilize.

Given the $N_c^j(t)$ and $B_c^j(t)$, we can then solve for the $N_c^j(t + \delta)$ using equations (2), and we repeat the process to obtain the time evolution of the performance measures for time instants $0, \; \delta, \; 2\delta, \; \cdots$. Every T time units ($\gg \delta$), we also update the routing probabilities, which gives rise to new values for the $\lambda_c(t)$.

Figure 2 illustrates our evaluation method. Steps 7-15 represent the heart of the method. They show the computation for each time instant k consisting of two parts. In the first part (steps 8-14), for every link j, we iteratively obtain the class blocking probabilities from the average numbers of established connections. In the second part (step 15), after all link models have converged, we update the average numbers of established connections for the next time step $k + \delta$ using the difference equations. Steps 5 and 6 show the periodic updates to the $\lambda_c(t)$, resulting from the periodic routing updates based on information collected during the past time period.

Assuming that L iterations are needed for convergence of the iteration in steps 10-14, the computational complexity for each time step is $O(|\mathcal{J}| \; |\mathcal{C}^j| \; (\; (|B_c^j| + |z_c^j|)L + |N_c^j| \;))$, where $\mathcal{J}$ is the set of all links, $|B_c^j|$ is the cost of evaluating $B_c^j(.)$ via (7), $|z_c^j|$ that of evaluating $z_c^j(.)$ via (8), and $|N_c^j|$ that of evaluating $N_c^j(.)$ via (2). Our method requires storage of $O(V\ |\mathcal{J}|\ |\mathcal{C}^j|)$, where V is the number of instantaneous measures. From Figure 2, we have $V = 5$ since we have 5 instantaneous measures defined, namely, $B_c^j(.)$, $z_c^j(.)$, $N_c^j(.)$, $\lambda_c(.)$ and $\mu_c(.)$.

```
begin
  1. Given the network topology and services, determine the set of classes for every link j
     and then their local traffic and QoS parameters
  2. For every link j, determine its schedulable states F^j
     and blocking states for each class c ∈ C^j
  3. Initialize {N_c^j(0) : c ∈ C^j} for every link j   /* 0 for initially empty network */
  4. For p = 0, 1, 2, ···      /* Update routes periodically */
        begin
           5. Using {N_c^j(i) : c ∈ C^j, p − 1 ≤ i < p, p > 0} for all j and/or other information,
              compute the routing probabilities as defined by the routing algorithm
           6. Using the routing probabilities, determine for every link j
              {λ_c(i) : c ∈ C^j, p ≤ i < p + 1}
           7. For k = p, p + δ, p + 2δ, ···, p + 1 − δ
                 begin
                    8. For every link j
                               /* Obtain {B_c^j(k) : c ∈ C^j} in terms of {N_c^j(k) : c ∈ C^j} */
                       begin
                          9. Initialize {ẑ_c^j(k) : c ∈ C^j}      /* arbitrary value if k = 0 */
                                                                  /* ẑ_c^j(k − δ) if k > 0 */
                          10. repeat
                          11.    z_c^j(k) ← ẑ_c^j(k), for every c ∈ C^j
                          12.    Obtain {B_c^j(k) : c ∈ C^j} in terms of {z_c^j(k) : c ∈ C^j}
                                 using (7)
                          13.    Obtain {ẑ_c^j(k) : c ∈ C^j} in terms of {B_c^j(k), N_c^j(k) : c ∈ C^j}
                                 using (8)
                          14. until | ẑ_c^j(k) − z_c^j(k) |< ε, for every c ∈ C^j
                       end
                    15. For every link j, compute {N_c^j(k + δ) : c ∈ C^j} using (2)
                 end
        end
end
```

Figure 2: Proposed evaluation method.

Comments

The generality and time-dependency of our method allow us to evaluate various control policies through schedulable states, blocking states, and time-dependent arrival and service rates. Sections 4 and 5 illustrate how our method can capture the effects of various policies.

Our method can also be used to directly solve for steady-state, if the $\lambda_c(t)$ and $\mu_c(t)$ are constants and a solution exists. We simply set $\frac{N_c^j(t+\delta)-N_c^j(t)}{\delta} = 0$ in equations (2) and use them in conjunction with equations (7) and (8) to iteratively solve for steady-state.

Observe that it is easy to realize parallel implementations of our method by mapping the computations for different links onto different processors, and we would expect almost linear speedup.

The accuracy of our method depends on the approximation of the relationship between the $B_c^j(t)$ and the $N_c^j(t)$ by its steady-state counterpart, which is the fixed point of the iteration in steps 10-14 of Figure 2. We are analyzing the errors and convergence of this iteration. Such analysis is very hard in general because of the complex nature of the underlying nonlinear system. However it can be shown in many situations that the error in the approximation is small, and that the iteration is a contractive mapping of $[0,\ 1)$ into $[0,\ 1)$ and hence it converges to a unique fixed point [32]. Furthermore, our numerical computations indicate that the iteration converges quickly. (See Sections 6, 7 and 8.) In particular, our method provides significant computational savings over the (straightforward) discrete-event simulation approach, which requires the averaging over a large number of independent runs to obtain reliable performance estimates.

4 Scheduling and Admission

Our method accounts for scheduling at a link j through the set of schedulable states $\mathcal{F}^j$. In the following, we illustrate the computation of $\mathcal{F}^j$ for a "per-connection" link scheduling algorithm of the weighted round-robin type. An example of this type of scheduling algorithms is weighted fair-queueing [43]. Here, each class-c connection is allocated (and guaranteed) a certain amount of bandwidth on link $j \in \mathcal{R}_c$ that is enough to satisfy its local QoS requirement. This required bandwidth depends of course on the local QoS and the packet generation characteristics of the connection.

Henceforth we assume that a connection of service s requests an end-to-end statistical delay bound $(D_s,\ \varepsilon_s)$, where the delay does not include the propagation delay. This QoS requirement is also referred to as packet jitter [16, 50]. This is typically required by applications such as voice since they can tolerate some packet loss (a packet is considered lost if its delay exceeds D_s) [15, 16].

If the connection is described by a two-state model where it is either in a busy state sending packets back-to-back at peak rate or in an idle state sending no packets at all, the required bandwidth[2], denoted by R_c^j, can be obtained from the following approximation derived in [2, 22, 14]:

$$R_c^j = M_c \frac{\beta_c^j - X_c^j + \sqrt{[\beta_c^j - X_c^j]^2 + 4\ X_c^j\ \rho_c\ \beta_c^j}}{2\ \beta_c^j} \tag{9}$$

where

- M_c is the peak rate of the connection.
- m_c is the mean rate of the connection.
- b_c is the average duration of the busy period.
- $\beta_c^j = ln(\frac{1}{\varepsilon_c^j})\ b_c\ (1 - \rho_c)\ M_c$.
- $\rho_c = \frac{m_c}{M_c}$ is the probability that the connection is active (in busy state).
- $X_c^j = D_c^j \times R_c^j$ is the buffer space required by the connection.

R_c^j can be computed from equation (9) iteratively. For each class $c \in \mathcal{C}^j$, we can then determine its requirements R_c^j and X_c^j. From this, we can determine whether a state $(\sigma_1, \sigma_2, \cdots, \sigma_{|\mathcal{C}^j|})$ belongs to $\mathcal{F}^j$; it must satisfy the following two conditions:

- $\sum_{c \in \mathcal{C}^j} \sigma_c\ R_c^j$ is no greater than the total capacity of link j, denoted by Cap^j.
- $\sum_{c \in \mathcal{C}^j} \sigma_c\ X_c^j$ is no greater than the total available buffer space of the link.

For ease of presentation, we assume that there is enough link buffer space such that the second condition is always satisfied. Then for a state to be schedulable it suffices to only satisfy the first condition.

As pointed out in Section 3, $\mathcal{F}^j$ would typically be different for every link j because links have different capacities, are used by different sets of classes, etc. It is also different for different scheduling disciplines because disciplines resulting in looser performance bounds would typically have a smaller set of schedulable states.

The computation of $\mathcal{F}^j$ seems expensive as it requires determining the $|\ \mathcal{C}^j\ |$-dimension schedulable states [35]. In addition, given the admission control policy, we need to determine for each class

[2] Often referred to as effective or equivalent capacity [14, 31, 2, 22, 1].

which of the schedulable states are blocking. This computational complexity is reduced if we assume $\{R_c^j : c \in \mathcal{C}^j\}$ are integers and view the link state as belonging to the set $\{0, 1, 2, \ldots, Cap^j - 1, Cap^j\}$, where the state indicates the amount of bandwidth reserved. This *one-dimensional* link model has a simple steady-state solution in the multi-rate circuit switching literature [46].[3] In particular, let $Q^j(i)$ denote the steady-state probability of link j being in state i. Then the $Q^j(.)$ satisfy the following recurrence relation [46]:

$$i\, Q^j(i) = \sum_{c \in \mathcal{C}^j} \frac{\lambda_c}{\mu_c} R_c^j\, Q^j(i - R_c^j) \qquad i = 1, \ldots, Cap^j \tag{10}$$

where $\sum_{i=0}^{Cap^j} Q^j(i) = 1$.

With a complete-sharing admission control policy, the steady-state blocking probability for class-c connections is simply given by

$$B_c^j = \sum_{i = Cap^j - R_c^j + 1}^{Cap^j} Q^j(i) \tag{11}$$

This result is valid for Poisson arrivals and generally-distributed lifetimes. It can be used instead of the one in (5). Note that here $\mathcal{F}^j$ is implicitly defined by the constraint $0 \leq i \leq Cap^j$. This link model is usually referred to as the *stochastic Knapsack* model [8, 9].

5 Routing

Our method accounts for routing through the time-dependent class arrival rates $\lambda_c(t)$. These are affected in our model by the route selection probabilities $\alpha_{s,c}(t)$. We assume the $\alpha_{s,c}(t)$ are periodically computed based on the network topology and load averaged over the last period. The load information consists of link/path measurements, which may include quantities such as reserved link capacity and path blocking probability. Obviously these quantities should be measurable in practice; indeed a node can measure the reserved capacity for each of its outgoing links from the connection setup/teardown procedure. Also, a source node can measure the blocking probability of a path if we assume that when a setup fails at an intermediate node, this node sends a "reject" message back to the source.

[3] In a multi-rate circuit-switched network, each call may request a different number of channels. This number is however the *same* on every link along *any* route the call might take. This is not the case in the networks we are considering where the bandwidth required by a connection on a link depends on the number of links along the route taken by the connection.

These quantities should also be obtainable from our model. We can obtain the average reserved link capacity from the average number of established connections and the effective capacity of each of the link's classes, which we compute in our model. We can also obtain a path blocking probability from the classes' blocking probabilities, which we also compute in our model.

We are interested in route selection algorithms for networks of *arbitrary* topologies and offering *heterogeneous* services. We want algorithms that result in low blocking probabilities (a high successful setup rate) and hence high network throughput. Our model can capture several design choices when developing such algorithm. One design choice is related to the set of candidate paths the source node would consider for connection routing. This determines the number of classes defined for each service. We do not want the source node to consider paths that are too long since this would result in increased utilization and hence reduced throughput. So the set of candidate paths could consist of only minimum-hop paths, or it could consist of both minimum-hop paths and next-to-minimum-hop paths. (By a next-to-minimum-hop path, we mean a minimum-hop + i path for the smallest $i \in \{1, 2, \cdots\}$ such that a path exists.)

Routing schemes designed for circuit-switched networks [19] and recently proposed for ATM networks [23, 26, 24, 25] consider one-hop and two-hop paths only. Routing schemes that consider paths of arbitrary hop length are often proposed for the Internet [47, 6]. Our model can evaluate both types of schemes.

From the set of candidate paths, we should determine which path to use for routing the setup request message for a new incoming connection. A path p could be selected probabilistically at random or using path weights W_p where[4]

$$W_p \propto \frac{F_p}{H_p \times L_p} \tag{12}$$

where H_p is the number of hops of path p (this gives preference to shortest paths), L_p is a measure of the load on path p averaged over the last update period (discussed below), and F_p is either 1 or 0 depending on whether the path p is feasible or not; a path p is said to be feasible if the source "expects" a successful setup on p [1].[5] The $\alpha_{s,c}(t)$ can then be computed according to (12).

Another design issue is related to how L_p is defined. For example, L_p could be (i) the blocking probability of path p, (ii) the sum of the utilizations of the links on path p, where the utilization of a link is the fraction of the link capacity reserved, (iii) the maximum link utilization of the links

[4] W_p may depend on other factors. We use here the ones that were considered in previous works (e.g., [4, 1, 6]) when selecting routes for connections.

[5] The source would take into account the requirements of the new connection in addition to the current load on the path (assuming it is accurate) to test the feasibility of the path. This is in fact an admission control function.

on path p, or (iv) the sum of the delays of the links on path p, where the delay of a link j can be estimated as $\frac{1}{Cap^j - CapRes^j}$ where $CapRes^j$ is the average reserved link capacity (note that this delay estimation uses the $M/M/1$ delay formula [34]).

In Section 8, we use our model to compare different route selection policies assuming the "per-connection" link scheduling and complete-sharing admission described in Section 4.

6 Validation Results for Single-Link Networks

In this section, we compare the results obtained using our method with those obtained using discrete-event simulation for single-link networks. In our method, we obtain instantaneous performance measures through equations (7), (8), and (2). We take the time step δ to be 0.1.

The simulation model differs from our analytical model in that the actual events of arrival and processing of requests are simulated according to the specified probability distributions and control policies. To obtain reliable performance estimates, a number of independent replications (i.e. simulation runs) must be carried out and averaged. In particular, let $Y^{(i)}(t)$ denote a generic measure computed at time instant t in replication i, where t takes on the successive values $t_1, t_2, \cdots, t_k, \cdots$. Then, the mean value of this measure at particular time instant t_k is estimated as $\sum_{i=1}^{N} Y^{(i)}(t_k)/N$, where N is the total number of replications. The larger N is, the more accurate the simulation estimates are [36]. In our simulations, the performance measures are computed for $t = 1, 2, 3, \cdots$.

The measures considered are precisely defined as they are introduced below. In all experiments, we start with empty systems. For the cases with $N = 50$, the observed mean of the simulation measures at various time instants typically show 95% confidence interval for a $\pm$ 10% range. For the cases with higher N, 95% confidence interval is obtained for a $\pm$ 3% range.

We assume class-c connections arrive according to a Poisson process of constant rate λ_c. A class-c connection requires a fixed amount of bandwidth R_c. If admitted, a class-c connection holds the acquired bandwidth for an exponential duration of constant rate μ_c.

We consider a single link j1 used by 10 service classes whose parameters are shown in Figure 3. We examined the use of both equations (5) and (11) to compute the class blocking probabilities. Consistent with [46], both yield the same results. In general, we found using equations (11) much less time-consuming. This is mainly because using equations (5) require finding the 10-dimension schedulable states, and also the blocking states of each class.

Figures 7, 8 and 9 show the time behavior of the total number of established connections, the

Class c	R_c	λ_c	$1/\mu_c$
1	30	0.125	5
2	15	0.5	1
3	50	0.2	2
4	10	0.1	2
5	40	0.125	1
6	25	0.5	0.5
7	30	1.0	0.5
8	10	0.0625	10
9	5	1.0	0.2
10	50	0.25	2

Figure 3: Parameters of 10 classes using a link j1 with total bandwidth of 200.

link utilization, and the total throughput, respectively. The first measure denotes the total number of connections currently established on the link, which is equal to $\sum_{c' \in \mathcal{C}^{j1}} N_{c'}^{j1}(t)$ in our method. The second measure denotes the fraction of link j1's capacity currently being reserved, which is equal to $(\sum_{c' \in \mathcal{C}^{j1}} N_{c'}^{j1}(t) \times R_{c'})/Cap^{j1}$ in our method. The third measure denotes the total current admission rate, which is equal to $\sum_{c' \in \mathcal{C}^{j1}} \lambda_{c'}[1 - B_{c'}^{j1}(t)]$ in our method. For a general network, it is equal to $\sum_{c' \in \mathcal{C}} \lambda_{c'} \prod_{j' \in \mathcal{R}_{c'}}[1 - B_{c'}^{j'}(t)]$, where $\mathcal{C}$ is the set of all classes.

In our simulations, the first two measures displayed at time instant t ($t = 1, 2, 3, \cdots$) are simply the values of these measures as observed at t. The last measure, namely the total throughput, displayed at time instant t is defined to be the total number of connections admitted in the interval $[t-1,\ t)$.

Our method yields results very close to the exact values. In addition, we found our method much less time-consuming than simulation. This is especially because the latter requires the averaging of a large number of independent simulation runs. To give an idea of the computational savings, for this experiment, on a DECstation 5000/133, our method required around 6 seconds of execution time while the 50-run and 1000-run simulations required around 25 seconds and 8 minutes, respectively. The number of iterations required at each time step for convergence of the iterative procedure in steps 7-15 of Figure 2 is less than 6 iterations for $\epsilon = 10^{-5}$ and $\hat{z}_c^j(0) = \lambda_c/\mu_c$.

7 Validation Results for Multi-Link Networks

In this section, we compare the results obtained using our method with those obtained using discrete-event simulation for multi-link networks. The purpose is to validate our link independence assumption manifested in equations (1) and (2) by the product term $\prod$. The performance measures are computed as described in Section 6. Similar confidence intervals are also observed for the measures obtained by simulation.

We consider a 3-link network with 20 service classes depicted in Figure 4. Classes 1 to 10 represent multi-hop connections modeling main traffic, while other classes represent one-hop connections modeling cross-traffic. Figure 5 shows the system parameters. We assume a class-c connection requires a fixed amount of bandwidth on every link $j \in \mathcal{R}_c$, i.e. $R_c^j = R_c$ for all $j \in \mathcal{R}_c$.

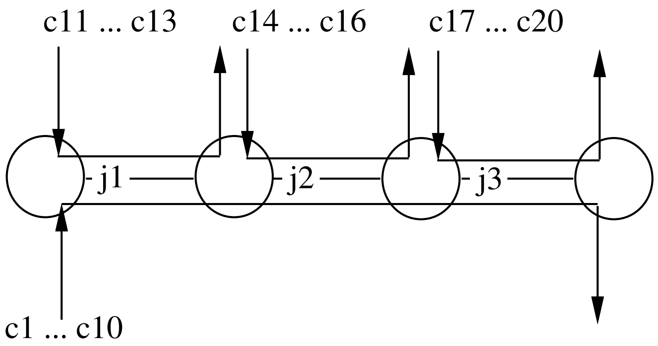

Figure 4: Multi-link network.

Figure 10 shows the instantaneous total throughput. Simulation results, denoted by **Exp**, are for Poisson arrivals and exponential lifetimes. Simulation results, denoted by **Det**, are for Poisson arrivals and deterministic lifetimes. The results show the accuracy of our method in both cases as they satisfy the assumptions required to obtain equations (11). (Our experiments with deterministic arrivals show large errors as expected.)

Next, we consider a similar multi-link network whose parameters are given in Figure 6. Here, λ_{c1} varies with time. This mimics the effect of traffic control policies such as flow control and routing. We assume λ_{c1} alternates every 20 time units between zero and 0.125, starting with zero. Figures 11 and 12 show the instantaneous total throughput and blocking probability, respectively. Our method accurately reproduces the behavior obtained by simulation. We compute the instantaneous blocking probability $B(t)$ from the throughput $\gamma(t)$ using the relation $B(t) = 1 - \gamma(t)/\lambda(t)$, where $\lambda(t)$ is the

Class c	R_c	λ_c	$1/\mu_c$
c1	30	0.125	5
c2	15	0.5	1
c3	50	0.2	2
c4	10	0.1	2
c5	40	0.125	1
c6	25	0.5	0.5
c7	30	1.0	0.5
c8	10	0.0625	10
c9	5	1.0	0.2
c10	50	0.25	2
c11	30	0.125	5
c12	15	0.5	1
c13	50	0.2	2
c14	10	0.1	2
c15	40	0.125	1
c16	25	0.5	0.5
c17	30	1.0	0.5
c18	10	0.0625	10
c19	5	1.0	0.2
c20	50	0.25	2

Figure 5: Parameters of 20 classes using 3 links r1, r2, and r3 with bandwidths of 150, 200 and 250, respectively.

instantaneous total arrival rate of requests. We do this rather than compute $B(t)$ directly from the simulations because doing that would require averaging over a very large number of replications, because $B(t)$ typically has a very low value and thus a high sample variance.

Class c	R_c	λ_c	$1/\mu_c$
c1	30	$0 \leftrightarrow 0.125$	5
c2	30	0.125	5
c3	10	0.1	2
c4	50	0.25	2

Figure 6: Parameters of 4 classes using 3 links j1, j2, and j3 with bandwidths of 50, 100 and 150, respectively.

8 Numerical Results for NSFNET

In this section, we use our model to compare three route selection algorithms. We assume the use of the "per-connection" link scheduling and complete-sharing admission described in Section 4. The required bandwidths R_c^j are computed using equation (9); if the computed value is not integer, it is rounded to the smallest integer greater than this value. We assume adequate buffer space.

We consider the performance of the routing algorithms on the topology of the NSFNET backbone shown in Figure 13. All links have capacities of 600. The time step δ equals 0.1. The routing update period T equals 5. We consider 52 services using the NSFNET backbone, with parameters as shown in Figure 14. Services with the same traffic and end-to-end QoS parameters, but with different source/destination pairs, are grouped in the same row.

We assume a source node considers only the set of minimum-hop and minimum-hop + 1 paths for connection routing. A path from the set is selected probabilistically according to path weights as explained in Section 5. The first selection algorithm, referred to as **SEL.HOP**, defines the path weight as $1/H_p$. The second selection algorithm, referred to as **SEL.UTIL**, defines the path weight as $(1 - U_p)$, where U_p is the maximum link utilization of the links on path p. The third selection algorithm, referred to as **SEL.UTIL_HOP**, defines the path weight as $(1 - U_p)/H_p$.

Figure 15 shows the instantaneous network throughput for the three routing algorithms. Figure 16 shows their instantaneous blocking probabilities. We observe that SEL.UTIL_HOP performs the best, closely followed by SEL.UTIL, and then by SEL.HOP, which is much worse. Clearly, for this network configuration, choosing paths which are both under-utilized and short for routing new incoming connections is the best strategy. We note that this is consistent with results in [4] where a route selection algorithm similar to SEL.UTIL_HOP was shown to outperform other algorithms on a 5-node connection-oriented reservationless network using discrete-event simulations. To obtain a curve here, our method required around 45 minutes of execution time rather than the tens of hours that simulation would have required.

9 Conclusions

Integrated services networks have often been analyzed under steady-state conditions (e.g. [28, 30, 35, 13, 46, 9, 37]). In this paper, we presented a numerical-analytical method to rapidly evaluate detailed and dynamic models of integrated services networks. Our results indicate that the method gives approximate, yet accurate, instantaneous performance measures and provides significant computational savings over discrete-event simulation. We have applied our method to compare different routing algorithms.

There are several areas for future work. One area is to examine routing schemes that distinguish between different types of traffic (e.g., low-throughput voice and high-throughput video), computing a different set of routes for each type. For example, for a particular traffic type with very stringent QoS requirements, we could restrict the set of candidate paths to only minimum-hop paths, while for other traffic types the set could also include next-to-minimum-hop paths. We intend to examine the capability of the routing scheme to distribute connections of each type in a way that increases the network throughput, and also the responsiveness of the routing scheme to failures and repairs.

Another area is to examine admission controls that block some connection setup requests even if their admission is feasible, possibly in order to reduce the chance of future blocking of connections of other types. In this case, blocking would occur at more schedulable states [51, 27].

Another area is to investigate policies other than the equal allocation policy for dividing the end-to-end QoS requirement among the links of a route. These policies would take into account the current link loads as measured in the last routing update period. Other QoS requirements such as packet loss can be considered.

We solved our model by an iteration that differs from iterations commonly used in steady-state analysis, which only solve for steady-state measures and ignore the effect of the routing update period (i.e. the delayed feedback between route changes and link load changes).

Our iteration uses a basic concept, that of approximating the instantaneous relationship between the blocking probabilities and average numbers of established connections by its steady-state counterpart. This concept was originally introduced in [18], where it was used to obtain steady-state blocking probability and carried load for a specific call routing and network topology.

Reference [18] considered a network of source nodes, destination nodes, and intermediate nodes, with a link from every source node to every intermediate node, and a link from every intermediate node to every destination node. Each link can carry a fixed total number of calls. The call arrival process from a source to a destination is Poisson with fixed rate. The call routing is not dynamic; a fixed fraction of the call arrivals is routed through every intermediate node. In addition, overflow traffic (due to blocking links) is routed through alternate available routes. Each call, once admitted, has an exponential holding time of fixed mean that is the same for all calls. The blocking probability of a link is given by the Erlang-B formula expressed in terms of combined traffic intensity. The system is solved for steady-state average number of calls on each link by equating the call departure rate to the call admission rate.

Our model extends this concept to general multi-class links, where, for example, each class has different resource needs, and links employ different scheduling disciplines. Also, our model can be applied to describe general dynamic routing schemes with the arrival rate of a class changing as a function of the instantaneous network state.

Our dynamic flow model is quite general, and can be used to study both transient and steady-state performances of integrated services networks. Our method has advantages over other methods that might be used to analyze transient behaviors. One such method is that of time-dependent queueing models, which involve probability distributions for all events. However, such models are extremely difficult to solve analytically [49], and computationally expensive to solve numerically [48]. A second method is that of diffusion models, which utilize averages and variances [7, 40]. Such models involve partial differential equations and are usually intractable. A third method is that of fluid models, which utilize average quantities only [5]. Such models involve ordinary differential equations and are usually tractable. However, dynamic flow models appear more accurate since they include detailed probabilistic descriptions manifested in our model in the computation of the instantaneous blocking probabilities.

References

[1] H. Ahmadi, J. Chen, and R. Guerin. Dynamic Routing and Call Control in High-Speed Integrated Networks. In Proc. *Workshop on Systems Engineering and Traffic Engineering, ITC'13*, pages 19–26, Copenhagen, Denmark, June 1991.

[2] D. Anick, D. Mitra, and M. Sondhi. Stochastic Theory of a Data Handling System with Multiple Sources. *Bell Syst. Tech. J.*, 61:1871–1894, 1982.

[3] G. Ash, J. Chen, A. Frey, and B. Huang. Real-time Network Routing in a Dynamic Class-of-Service Network. In Proc. *13th ITC*, Copenhagen, Denmark, 1991.

[4] S. Bahk and M. El Zarki. Dynamic Multi-path Routing and How it Compares with other Dynamic Routing Algorithms for High Speed Wide Area Networks. In Proc. *SIGCOMM '92*, pages 53–64, Baltimore, Maryland, August 1992.

[5] J-C. Bolot and A.U. Shankar. Analysis of a Fluid Approximation to Flow Control Dynamics. In Proc. *IEEE INFOCOM '92*, Florence, Italy, May 1992.

[6] L. Breslau, D. Estrin, and L. Zhang. A Simulation Study of Adaptive Source Routing in Integrated Services Networks. Available by anonymous ftp at catarina.usc.edu:pub/breslau, September 1993.

[7] H. Chen and A. Mandelbaum. Discrete Flow Networks: Diffusion Approximations and Bottlenecks. *Annals of Probability*, 19(4):1463–1519, October 1991.

[8] S-P. Chung, A. Kashper, and K. Ross. Computing Approximate Blocking Probabilities for Large Loss Networks with State-Dependent Routing. *IEEE/ACM Transactions on Networking*, 1(1):105–115, February 1993.

[9] S-P. Chung and K. Ross. Reduced Load Approximations for Multirate Loss Networks. *IEEE Transactions on Communications*, 41(8):1222–1231, August 1993.

[10] J. Cobb and M. Gouda. Flow Theory: Verification of Rate-Reservation Protocols. In Proc. *IEEE International Conference on Network Protocols '93*, San Francisco, California, October 1993.

[11] D. Comer and R. Yavatkar. FLOWS: Performance Guarantees in Best Effort Delivery Systems. In Proc. *IEEE INFOCOM, Ottawa, Canada*, pages 100–109, April 1989.

[12] R.L. Cruz. A Calculus for Network Delay, Part II: Network Analysis. *IEEE Transactions on Information Theory*, 37(1):132–141, January 1991.

[13] Z. Dziong and J. Roberts. Congestion Probabilities in a Circuit-Switched Integrated Services Network. *Performance Evaluation*, 7:267–284, 1987.

[14] A. Elwalid and D. Mitra. Effective Bandwidth of General Markovian Traffic Sources and Admission Control of High-Speed Networks. *IEEE/ACM Transactions on Networking*, 1(3):329–343, June 1993.

[15] D. Ferrari. Real–Time Communication in Packet–Switching Wide–Area Networks. Technical Report TR-89-022, International Computer Science Institute, Berkeley, California, May 1989.

[16] D. Ferrari. Client Requirements for Real–Time Communication Services. *IEEE Communications Magazine*, 28(11), November 1990.

[17] J. Filipiak. *Modeling and Control of Dynamic Flows in Communication Networks*. New York: Springer-Verlag, 1988.

[18] F. Le Gall and J. Bernussou. An Analytical Formulation for Grade of Service Determination in Telephone Networks. *IEEE Transactions on Communications*, COM-31(3):420–424, March 1983.

[19] A. Girard. *Routing and Dimensioning in Circuit-Switched Networks*. Addison-Wesley Publishing Company, 1990.

[20] A. Girard and M. Bell. Blocking Evaluation for Networks with Residual Capacity Adaptive Routing. *IEEE Transactions on Communications*, COM-37:1372–1380, 1989.

[21] A. Greenberg and P. Wright. Design and Analysis of Master/Slave Multiprocessors. *IEEE Transactions on Computers*, 40(8):963–976, August 1991.

[22] R. Guerin, H. Ahmadi, and M. Naghshineh. Equivalent Capacity and its Application to Bandwidth Allocation in High-Speed Networks. *IEEE J. Select. Areas Commun.*, SAC-9(7):968–981, September 1991.

[23] S. Gupta. *Performance Modeling and Management of High-Speed Networks*. PhD thesis, University of Pennsylvania, Department of Systems, 1993.

[24] S. Gupta, K. Ross, and M. ElZarki. Routing in Virtual Path Based ATM Networks. In Proc. *GLOBECOM '92*, pages 571–575, 1992.

[25] S. Gupta, K. Ross, and M. ElZarki. On Routing in ATM Networks. In Proc. *IFIP TC6 Task Group/WG6.4 Workshop on Modeling and Performance Evaluation of ATM Technology*. H. Perros, G. Pujolle, and Y. Takahashi (Editors). Elsevier Science Publishers B.V., Amsterdam, The Netherlands, 1993.

[26] R.-H. Hwang. *Routing in High-Speed Networks*. PhD thesis, University of Massachusetts, Department of Computer Science, May 1993.

[27] J. Hyman, A. Lazar, and G. Pacifici. A Separation Principle Between Scheduling and Admission Control for Broadband Switching. *IEEE J. Select. Areas Commun.*, SAC-11(4):605–616, May 1993.

[28] S. Jordan and P. Varaiya. Throughput in Multiple Service, Multiple Resource Communication Networks. *IEEE Transactions on Communications*, 39(8):1216–1222, August 1991.

[29] J. Kaufman. Blocking in a Shared Resource Environment. *IEEE Transactions on Communications*, 29(10):1474–1481, October 1981.

[30] F.P. Kelly. Blocking Probabilities in Large Circuit-Switched Networks. *Adv. Appl. Prob.*, 18:473–505, 1986.

[31] G. Kesidis, J. Walrand, and C.-S. Chang. Effective Bandwidths for Multiclass Markov Fluids and Other ATM Sources. *IEEE/ACM Transactions on Networking*, 1(4):424–428, August 1993.

[32] D. Kincaid and W. Cheney. *Numerical Analysis: Mathematics of Scientific Computing*. Brooks/Cole Publishing Company, 1991.

[33] L. Kleinrock. *Communication Nets: Stochastic Message Flow and Delay*. New York: McGraw Hill, 1964.

[34] L. Kleinrock. *Queueing Systems*, volume I and II. New York: Wiley, 1976.

[35] G. Louth, M. Mitzenmacher, and F.P. Kelly. Computational Complexity of Loss Networks. *Theoretical Computer Science*, 125:45–59, 1994.

[36] W. Lovegrove, J. Hammond, and D. Tipper. Simulation Methods for Studying Nonstationary Behavior of Computer Networks. *IEEE J. Select. Areas Commun.*, 8(9):1696–1708, December 1990.

[37] D. Mitra, R. Gibbens, and B. Huang. Analysis and Optimal Design of Aggregated-Least-Busy-Alternative Routing on Symmetric Loss Networks with Trunk Reservation. In Proc. *13th ITC*, Copenhagen, Denmark, 1991.

[38] D. Mitra and J. Seery. Comparative Evaluations of Randomized and Dynamic Routing Strategies for Circuit-Switched Networks. *IEEE Transactions on Communications*, 39(1):102–116, January 1991.

[39] D. Mitra and P. Weinberger. Probabilistic Models of Database Locking: Solutions, Computational Algorithms, and Asymptotics. *J. ACM*, 31(4):855–878, October 1984.

[40] A. Mukherjee and J.C. Strikwerda. Analysis of Dynamic Congestion Control Protocols: A Fokker-Plank Approach. In Proc. *ACM SIGCOMM '91*, Zurich, Switzerland, September 1991.

[41] R. Nagarajan, J. Kurose, and D. Towsley. Local Allocation of End-to-End Quality-of-Service in High-Speed Networks. In Proc. *IFIP TC6 Workshop on Modelling and Performance Evaluation of ATM Technology*, page 2.2, January 1993.

[42] Y. Ohba, M. Murata, and H. Miyahara. Analysis of Interdeparture Processes for Bursty Traffic in ATM Networks. *IEEE J. Select. Areas Commun.*, 9(3):468–476, April 1991.

[43] A. Parekh. A Generalized Processor Sharing Approach to Flow Control in Integrated Services Networks. Technical Report LIDS-TR-2089, Laboratory for Information and Decision Systems, Massachusetts Institute of Technology, 1992.

[44] V. Paxson and S. Floyd. Wide-Area Traffic: The Failure of Poisson Modeling. In Proc. *ACM SIGCOMM '94*, University College London, London, UK, September 1994.

[45] M. Prycker. *Asynchronous Transfer Mode - Solution for Broadband ISDN*. Ellis Horwood, 1991.

[46] J. Roberts. A Service System with Heterogeneous User Requirements - Application to Multi-Services Telecommunications Systems. In *Performance of Data Communications Systems and Their Applications*, pages 423–431. G. Pujolle (Editor). North-Holland Publishing Company, 1981.

[47] S. Sibal and A. DeSimone. Controlling Alternate Routing in General-Mesh Packet Flow Networks. In Proc. *ACM SIGCOMM '94*, pages 168–179, September 1994.

[48] D. Tipper and M.K. Sundareshan. Numerical Methods for Modeling Computer Networks Under Nonstationary Conditions. *IEEE J. Select. Areas Commun.*, 8(9):1682–1695, December 1990.

[49] S. Tripathi and A. Duda. Time-Dependent Analysis of Queueing Systems. *INFOR*, 24(3):199–219, 1986.

[50] D.C. Verma, H. Zhang, and D. Ferrari. Delay Jitter Control for Real–Time Communication in a Packet–Switching Network. In Proc. *IEEE TRICOMM*, pages 35–43, Chapel Hill, North Carolina, April 1991.

[51] J. Wieselthier, C. Barnhart, and A. Ephremides. Optimal Admission Control in Circuit-Switched Multihop Radio Networks. In Proc. *31st Conference on Decision and Control*, Tucson, Arizona, December 1992.

[52] H. Zhang and D. Ferrari. Rate-Controlled Static Priority Queueing. Technical Report TR-92-003, International Computer Science Institute, Berkeley, California, January 1992.

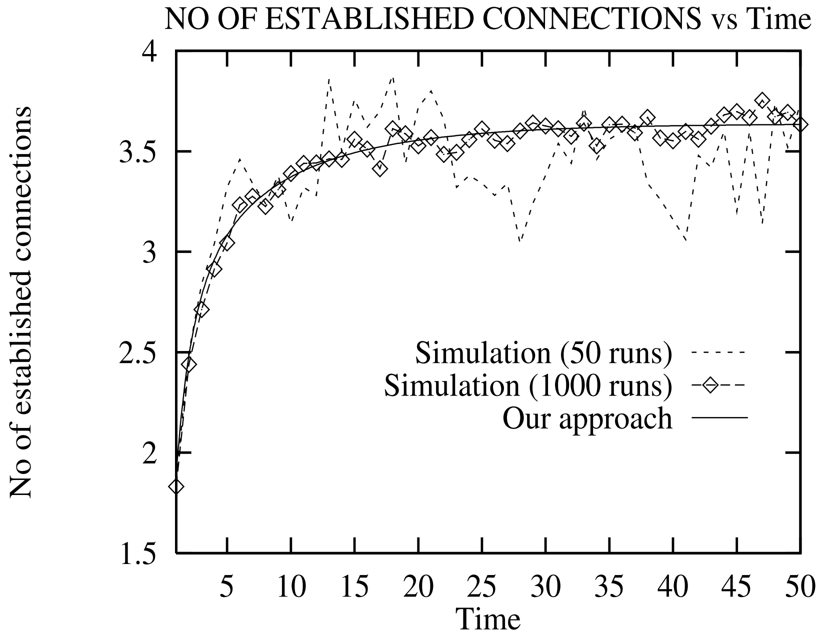

Figure 7: Total number of established connections versus time. Single-link network.

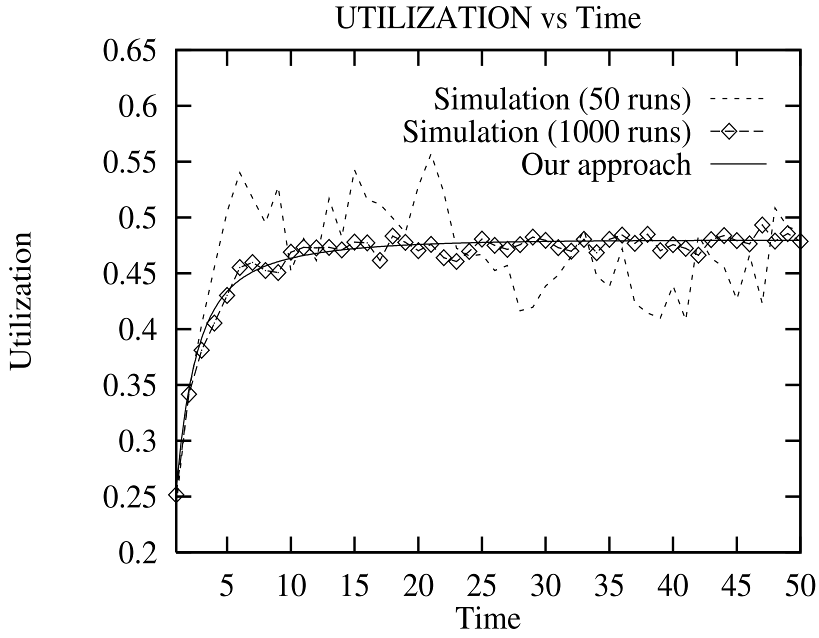

Figure 8: Link utilization versus time. Single-link network.

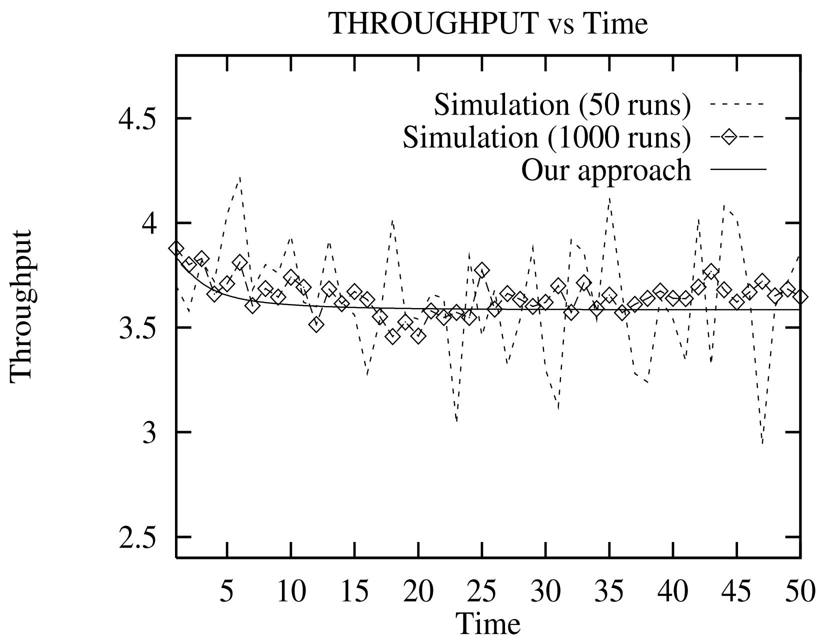

Figure 9: Total throughput versus time. Single-link network.

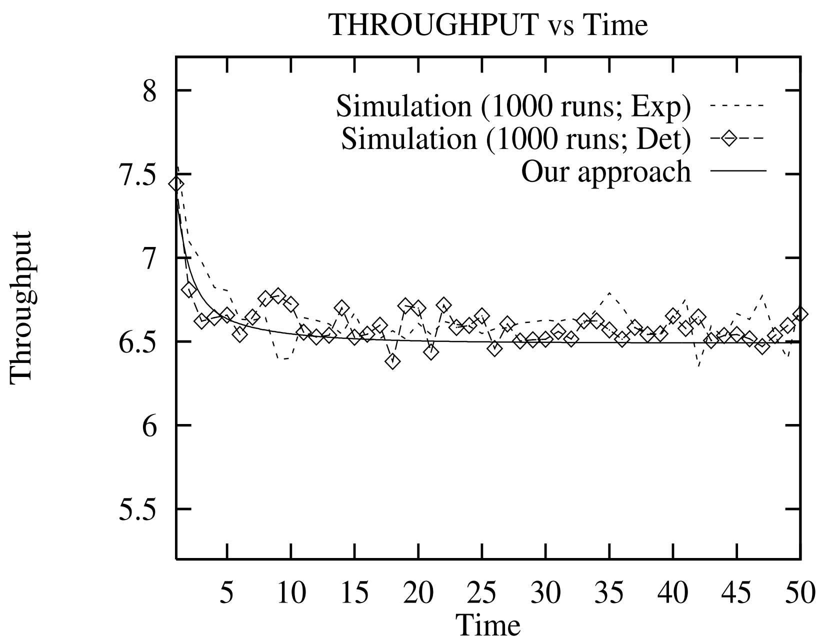

Figure 10: Total throughput versus time. Multi-link network.

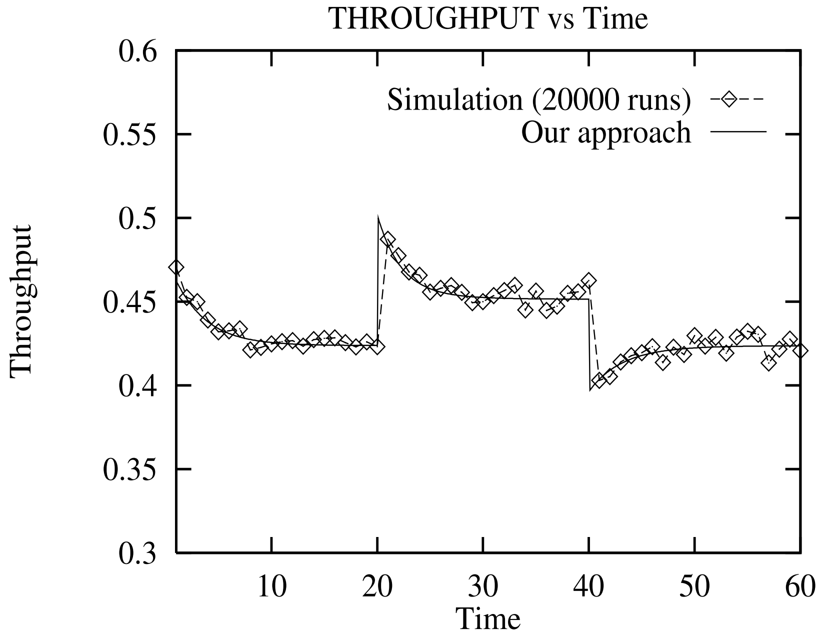

Figure 11: Total throughput versus time. Multi-link network. Time-varying arrivals.

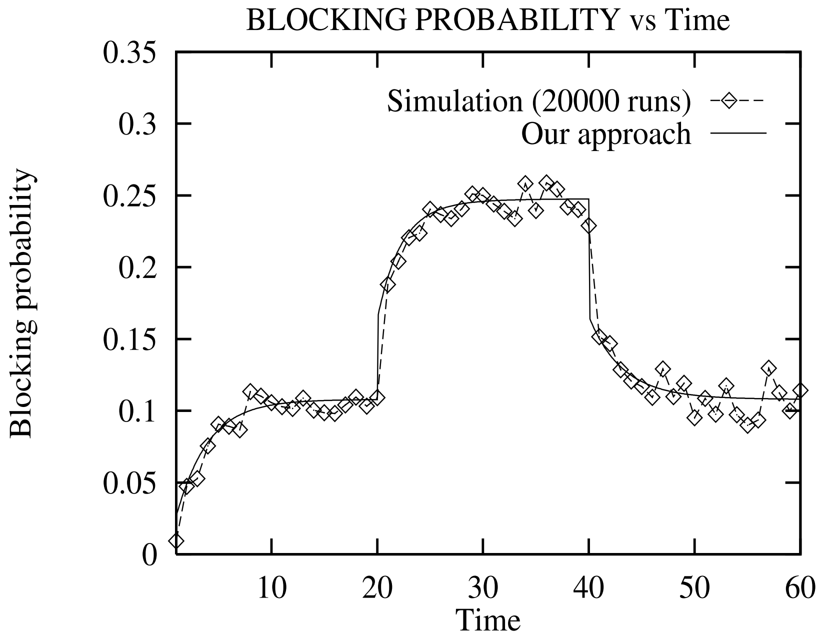

Figure 12: Blocking probability versus time. Multi-link network. Time-varying arrivals.

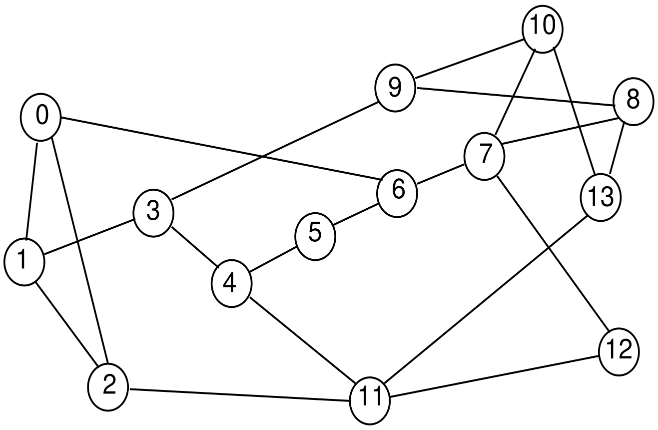

Figure 13: NSFNET backbone: 14 nodes, 21 bidirectional links, average degree 3.

$(SRC_s,\ DEST_s)$	$(M_s,\ m_s,\ b_s,\ D_s,\ \varepsilon_s)$	$(\lambda_s,\ \mu_s)$	No of services
(0, 13),(1, 13),(2, 13),(3, 13),(4, 13),(5, 13)	(30, 20, 0.1, 0.05, 10^{-4})	(2, 1)	6
(0, 13),(1, 13),(2, 13),(3, 13),(4, 13),(5, 13)	(30, 20, 0.1, 0.05, 10^{-4})	(2, 1)	6
(6, 13)	(30, 10, 0.1, 0.05, 10^{-4})	(2, 2)	1
(6, 13)	(30, 10, 0.1, 0.05, 10^{-4})	(2, 2)	1
(7, 13),(8, 13),(9, 13),(10, 13),(11, 13)	(30, 10, 0.1, 0.05, 10^{-4})	(1.8, 2)	5
(7, 13),(8, 13),(9, 13),(10, 13),(11, 13)	(30, 10, 0.1, 0.05, 10^{-4})	(1.8, 2)	5
(12, 13)	(60, 20, 0.1, 0.05, 10^{-4})	(0.3, 0.2)	1
(12, 13)	(60, 20, 0.1, 0.05, 10^{-4})	(0.3, 0.2)	1
(0, 1)	(30, 20, 0.1, 0.05, 10^{-4})	(2, 1)	1
(2, 1),(3, 1),(4, 1),(5, 1),(6, 1)	(30, 20, 0.1, 0.05, 10^{-4})	(2, 1)	5
(7, 1),(8, 1),(9, 1),(10, 1),(11, 1),(12, 1),(13, 1)	(30, 10, 0.1, 0.05, 10^{-4})	(1.8, 2)	7
(0, 1)	(30, 20, 0.1, 0.05, 10^{-4})	(2, 1)	1
(2, 1),(3, 1),(4, 1),(5, 1),(6, 1)	(30, 20, 0.1, 0.05, 10^{-4})	(2, 1)	5
(7, 1),(8, 1),(9, 1),(10, 1),(11, 1),(12, 1),(13, 1)	(30, 10, 0.1, 0.05, 10^{-4})	(1.8, 2)	7

Figure 14: Parameters of the 52 services using the NSFNET backbone.

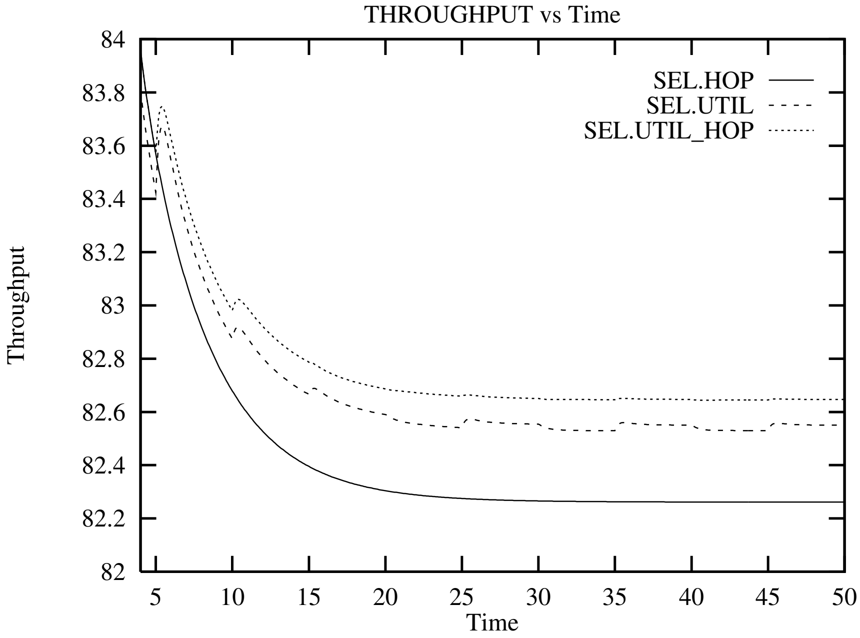

Figure 15: Total throughput versus time for the NSFNET backbone.

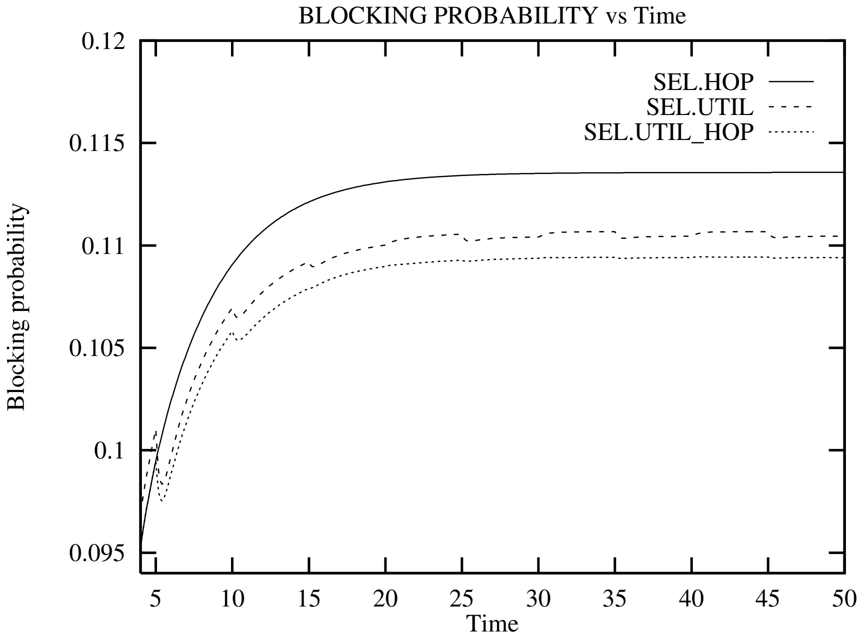

Figure 16: Blocking probability versus time for the NSFNET backbone.