Fast Time-Dependent Evaluation of Multi-Service Networks ⋆

Ibrahim Matta

College of Computer Science, Northeastern University, Boston, MA 02115, USA

A. Udaya Shankar

Department of Computer Science and Institute for Advanced Computer Studies, University of Maryland, College Park, Maryland 20742, USA

Abstract

We present a numerical-analytical method to evaluate multi-service 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.

Connection routing algorithms are usually evaluated individually in terms of steady-state performance measures. In this paper, we also use our time-dependent evaluation method to compare several connection routing schemes in terms of *instantaneous* measures. Our results show that a routing scheme which defines the cost of a path as the sum of measured link utilizations yields more stable behavior and lower connection blocking probability over a wide range of workload parameters and network configurations than other traditional schemes.

Key words: Multi-service networks, transient performance, dynamic flow models, queueing models, resource allocation algorithms, routing.

⋆ This work was supported in part by NSF CAREER grant NCR-9701988, NEU grant RSDF-377090, and by ARPA under contract DABT63-96-C-0075 administered by Army ARO to Computer Science Department, University of Maryland. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of NSF, ARPA, ARO, or the U.S. Government.

1 Introduction

Multi-service packet-switched networks, such as ATM (Asynchronous Transfer Mode) networks [49], 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.

Our method integrates techniques from several areas, namely: standard queueing theory steady-state techniques [37]; the link decomposition technique widely used for packet delay analysis in packet-switched data networks [36] as well as for call blocking probability analysis in circuit-switched telephone networks [21]; the dynamic flow technique used for approximating system dynamics and nonlinearity [18]; and the technique of repeated substitutions used in numerical analysis to solve nonlinear equations [35]. This integrated evaluation approach allows for handling the *transient behavior of general multi-class systems*. Few studies (e.g. [52]) have considered only non-blocking multi-class systems, others (e.g. [19]) have considered blocking single-class systems, and many studies have considered only steady-state behaviors. A detailed discussion of related work is in Section 9.

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 [36,21] to approximate the multi-link network as a collection of single-link networks. For each link, we approximate the relation-

ship 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 [37]. We solve these instantaneous relationships iteratively [35]. 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. [31,33,38,14,50,10,43,23]), 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 (replications) to obtain reliable performance estimates. Furthermore, the performance measures it yields are very close to the exact values obtained by simulation.

Routing schemes proposed for multi-service networks are usually evaluated individually in terms of steady-state measures. They are also sometimes evaluated using steady-state approximations that do not expose oscillatory behavior [10]. In this paper, we use our evaluation method to compare various routing schemes in terms of *transient* measures. Our results show that a routing scheme which defines the cost of a path as the sum of measured link utilizations yields more stable behavior and lower connection blocking probability over a wide range of workload parameters and network configurations than other schemes, including a traditional scheme which defines the cost of

a path as the maximum of measured link utilizations (e.g., [5,7]).

Organization of the paper

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 several connection routing schemes, including ones that adapt to *delayed* state information expressed in terms of link utilizations and delays. We present results for two networks. In the first network, routing schemes consider one-link and two-link paths when routing incoming connections. Such schemes are often proposed for ATM networks. The second network has the NSFNET-backbone topology and routing schemes consider shortest paths of arbitrary length. Such schemes are often proposed for the Internet. 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 local QoS is then requested from each of the links of this route 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; for simplicity 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 multi-service networks (e.g., [1,5,12]). 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 $T = 1$.

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:

- Time-dependent arrival rate of requests for a connection setup, $\lambda_s(t)$.
- Time-dependent 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, ε_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.

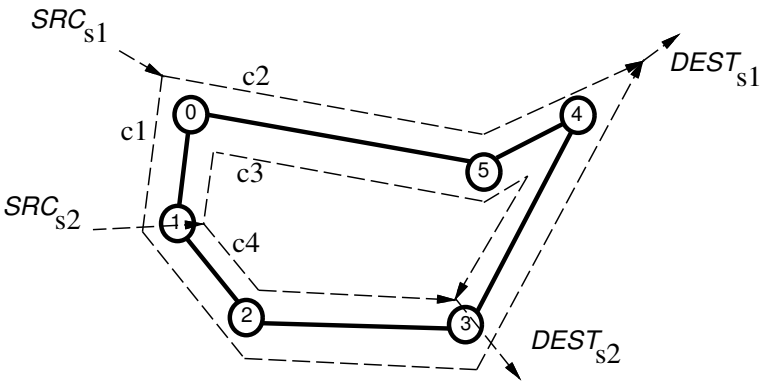

Fig. 1. A network example.

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 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 vary with time and are periodically updated according to dynamic status information (e.g. measured load). With static routing, the probabilities are constant over time. Note that deterministic or single-path routing, whether static or dynamic, is a special case of probabilistic routing where these probabilities take the values 0 and 1.
- 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 delay variation or jitter [17,54].[2]

[1] Experiences with circuit-switched networks show that this restriction results in simple and efficient routing schemes [3,42].

[2] Jitter is defined as the maximum difference in delay that can be experienced by any two packets of the same connection. The propagation delay from the source to the destination is a lower bound on the delay a packet would experience. It is

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) [16,17].

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, ε_s) is to be established on an h-link route, then we require that each link on the route guarantees a local requirement of ($\frac{D_s}{h}$, $\frac{\varepsilon_s}{h}$) [45,46].

- 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 [22,21,48]. The second assumption uses a type of routing proposed in many studies (e.g., [5,18]). The third assumption uses an end-to-end QoS equal allocation policy studied in [45,25,46].

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.* [24]. 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 [56]. Otherwise, the traffic pattern has to be characterized at each link as in [47,13].

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

straightforward to account for the propagation delays since these are fixed and we would simply subtract them from the requested end-to-end delay bound to obtain the requested bound on delay variation.

route. The average lifetime of a connection $\frac{1}{\mu_{c2}(t)} = \frac{1}{\mu_{s1}(t)}$. For an end-to-end QoS $(D_{s1},\ \varepsilon_{s1})$, the local QoS requirement $(D^j_{c2},\ \varepsilon^j_{c2}) = (\frac{D_{s1}}{2},\ \frac{\varepsilon_{s1}}{2})$ 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 $(M_{c2},\ m_{c2},\ \cdots) = (M_{s1},\ m_{s1},\ \cdots)$.

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:

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

Then, assuming link independence, we have

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

Let $\mathcal{C}^j$ be the set of all classes of connections using link j. For Markovian systems, 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 [53], and computationally expensive to solve numerically [52]. Instead, we write the following difference equations for $c \in \mathcal{C}^j$, for time step $\delta \ll T$ (recall $T = 1$):[3]

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

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 $[t, t+\delta)$. We note that taking the departure rate of class c on link j to be $\mu_c(t)\ N_c^j(t)$ is only an approximation as it is strictly valid for an $M/M/\infty$ system [18]. However, our validation results in Sections 6 and 7 indicate that equation (2) is a good approximation for blocking systems with general lifetime distributions.

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 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

[3] More sophisticated methods, for example fourth-order Runge-Kutta [35], can be used to solve the differential equations $\dot{N}_c^j(t) = -\mu_c(t)\ N_c^j(t)\ +\ \lambda_c(t)\ \prod_{j' \in \mathcal{R}_c}[1 - B_c^{j'}(t)]$.

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 $c \in \mathcal{C}^j$:

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

$\mathcal{S}_c^j$ can be obtained as follows. Define a *schedulable state* of link j to be a $\mid \mathcal{C}^j \mid$-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 [30]. 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 [24,11]. 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 [47]. [4]

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 [31], but also for generally-distributed lifetimes [32].

[4] Using packet-level analysis to determine the resource requirements of connections using link j and hence $\mathcal{F}^j$ is a form of hierarchical modeling/analysis. This packet-level analysis would in turn take into account the effect of statistical multiplexing at the links.

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 [30]. Note that $\mathbf{I}(.)$ defines the set of blocking states. Other admission control schemes can be modeled by alternative definitions of $\mathbf{I}(.)$. [5]

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

[5] Note that many accurate and efficient solutions to equation (5) have been proposed in the *steady-state* queueing literature, which our method would readily make use of as illustrated in Section 4.

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. We note that this computation can be speeded up greatly by precalculating the B_c^j and storing them for a range of z_c^j's or N_c^j's.

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
```

Fig. 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. Other policies could similarly be considered.

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. Analyzing the errors and convergence of this iteration is very hard in general because of the complex nature of the underlying nonlinear system. However it can be shown in many situations [40] 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 [35]. Furthermore, our numerical computations in Sections 6, 7 and 8 indicate that the iteration converges quickly. 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 [47]. 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 ($D_c^j,\ \varepsilon_c^j$) and the packet generation characteristics of the connection.

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 [6], denoted by R_c^j, can be obtained from the following approximation derived in [2,24,15]:

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

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

[6] Often referred to as effective or equivalent capacity [15,34,2,24,1]. Several techniques have been proposed for accurately computing this equivalent capacity for multiplexed connections of this kind. Our method would readily make use of any one of these techniques.

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 straightforward computation of $\mathcal{F}^j$ might be expensive as it requires determining the $|\ \mathcal{C}^j\ |$-dimension schedulable states [38]. In addition, given the admission control policy, we need to determine for each class 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 [50].[7] 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 [50]:

$$i \, Q^j(i) = \sum_{c \in \mathcal{C}^j} \frac{\lambda_c}{\mu_c} \, R_c^j \, Q^j(i - R_c^j)$$

[7] 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. That is, the longer the route, the more stringent the QoS requested from each link would be, and thus more bandwidth would be required on each link.

$$i = 1, \ldots, Cap^j \tag{10}$$

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 [9,10].

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. 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. 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 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 [21] and recently proposed for ATM networks [25,28,26,27] consider one-hop and two-hop paths only. Routing schemes that consider shortest paths of arbitrary hop length are often proposed for the Internet [51,7]. Our model can evaluate both types of schemes.

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

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

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]. To test the feasibility of the path, the source considers the QoS requested by the new connection in addition to the load on the path in the source's routing view. Although this view is periodically updated and might be outdated, the source assumes that the view is accurate. In other words, F_p is the result of the source node evaluating the admission control function for the remote nodes on the path. The $\alpha_{s,c}(t)$ can then be computed according to (12).

An important issue here is how L_p is defined. L_p could be defined as:

(i) 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.

(ii) The maximum link utilization of the links on path p. This is a traditional definition of L_p (e.g., [5,7]).

(iii) 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 Cap^j is the total link capacity and $CapRes^j$ is the average reserved link capacity. Similar definitions of L_p were proposed in [1,4,20], where the cost of a link is a highly nonlinear (e.g. exponential) function of its current utilization.

In Section 8, we use our model to compare the transient behavior of different route selection policies assuming the "per-connection" link scheduling and complete-sharing admission described in Section 4. The route selection schemes differ in how they assign the weights W_p.

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 [39]. 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 [50], 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 9, 10 and 11 show the time behavior of the total number of established connections, the 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 $\sum_{c' \in \mathcal{C}^{\mathrm{j1}}} N_{c'}^{\mathrm{j1}}(t)$ in our method. The second measure denotes the fraction of link j1's capacity currently being reserved, which is equal to $(\sum_{c' \in \mathcal{C}^{\mathrm{j1}}} N_{c'}^{\mathrm{j1}}(t) \times R_{c'})/Cap^{\mathrm{j1}}$ in our method. The third measure denotes the total current admission rate, which is equal to $\sum_{c' \in \mathcal{C}^{\mathrm{j1}}} \lambda_{c'}[1 - B_{c'}^{\mathrm{j1}}(t)]$ 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

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

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

to be the total number of connections admitted in the interval $[t-1,\ t)$. This instantaneous definition of throughput is sometimes referred to as "differential or marginal throughput". Note that as time increases, instantaneous throughput decreases as connections get blocked before it stabilizes.

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 $j \in \mathcal{R}_c$.

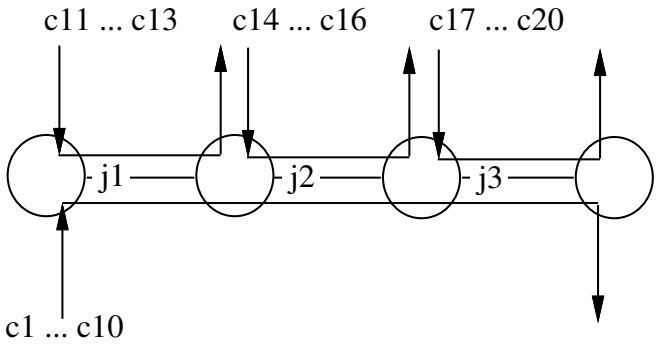

Fig. 4. Multi-link network.

Figure 12 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. We assume class c1 uses all 3 links, while each of the other classes

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

Fig. 5. Parameters of 20 classes using 3 links j1, j2, and j3 with bandwidths of 150, 200 and 250, respectively.

uses only one link—c2 uses j1, c3 uses j2, and c4 uses j3. 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 13 and 14 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 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

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

8 Numerical Results for Two Networks with Adaptive Routing

In this section, we use our model to compare several 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.

A number of routing schemes recently proposed for ATM networks [25,28,29,26,27] have been designed for networks that use the Virtual Path (VP) concept. This VP concept has been proposed to simplify network management and control. Typically, a VP is installed between two nodes over a sequence of physical links. Resources are statically allocated to the VP so that queueing occurs only at the first physical link. Thus a virtual (possibly fully-connected) network can be overlayed over the physical network. Simple routing schemes that only consider paths with one VP and two VPs are used. Thus, admission tests have to be done on at most two links resulting in small connection setup times.

Routing schemes that consider shortest paths of arbitrary hop length are often proposed for the Internet [51,7]. In this section, we consider a network with routing schemes that consider the set of one-hop and two-hop paths, and a network with NSFNET-backbone topology with routing schemes that consider the set of minimum-hop and minimum-hop + 1 paths of arbitrary length.

A path from the set is selected probabilistically according to path weights as explained in Section 5. The first selection scheme, referred to as **sumUTIL+HOP**, defines L_p as in (i) of Section 5 (*i.e.*, the sum of link utilizations on the path) and W_p as

$$W_p = \begin{cases} 100/H_p & \text{if } L_p \leq 0.01 \\ 1/(L_p \times H_p) & \text{otherwise} \end{cases}$$

The second selection scheme, referred to as **maxUTIL+HOP**, defines L_p as in (ii) of Section 5 (*i.e.*, the maximum of link utilizations on the path) and W_p as

$$W_p = (1 - L_p)/H_p$$

The third selection scheme, referred to as **sumDLY+HOP**, defines L_p as in (iii) of Section 5 (*i.e.*, the sum of link delays on the path, where delay on a

link is the inverse of available capacity) and W_p as

$$W_p = \begin{cases} 0 & \text{if } L_p = \infty \\ 1/(L_p \times H_p) & \text{otherwise} \end{cases}$$

We also considered variants of these schemes that do not give priority to shortest paths, i.e., H_p is set to 1. In this case, we omit the "+HOP" suffix from the name. We also consider **HOP**, a static scheme that does not consider load, i.e., $W_p = 1/H_p$. For a scheme that considers F_p when computing W_p, its name starts with "FEAS+". For example, **FEAS+maxUTIL** defines L_p as in (ii) of Section 5 and W_p as

$$W_p = \begin{cases} 0 & \text{if path is not feasible} \\ 1 - L_p & \text{otherwise} \end{cases}$$

We compare the time behavior of these routing schemes using our method, i.e., we obtain instantaneous performance measures through equations (2), (7) and (8). We take the discrete-time step δ to be 0.1. We first present general observations about the results, and then the details of our evaluations on two networks. Similar conclusions were obtained for other networks.

General observations

- By giving priority to shortest paths, a routing scheme always performs better. This is mainly because the use of long paths for connections is undesirable since it ties up resources at more intermediate nodes, which can be used to admit many shorter length connections.
- sumUTIL+HOP performs better or as well as all other schemes over a wide range of workload parameters and network configurations. This is because its path cost function adapts more slowly than other adaptive schemes, striking a good balance between adaptiveness and over-sensitivity. In particular, the path cost functions of maxUTIL+HOP and sumDLY+HOP are

more sensitive to load changes. The former takes the maximum, instead of the sum, over link utilizations, so whenever the link that is the bottleneck changes or its utilization changes the cost of the (possibly long) path changes. The latter uses delay, instead of utilization, to measure congestion. From the classical delay-utilization curve, around saturation, a small increase in utilization corresponds to a large increase in link delay. This dramatic change can result in the link becoming unattractive and thus completed avoided. Consequently, at the next routing update the link reports a very low cost and becomes attractive again. Such over-sensitivity leads to oscillatory behavior, which in turn degrades performance. Note that the degree of oscillations would depend on many factors such as connection lifetimes and arrival rates, the routing update interval, network topology, etc.

- By using F_p when computing W_p, a scheme performs as well. Recall that F_p is the result of the source node evaluating the admission control function for the remote nodes on the path. F_p depends on the requested QoS and the routing view maintained by the source node. This view is periodically updated and thus might be outdated. Based on this potentially inaccurate view, the source node might reject a candidate path if it does not "expect" that sending the setup message to perform admission control at nodes along the path will succeed. Also, the source node might reject the call if it believes that none of the candidate paths is feasible. However, we found that having admission control at the source is effective. It does not cause excessive unnecessary blocking and can potentially reduce the overhead of performing admission control tests inside the network that will actually fail.

Network 1

We consider the network topology shown in Figure 7. We assume all links have the same capacity C. We consider 4 services using the network, with parameters as shown in Table 1. The routing update period equals 2. Fig-

ures 15 through 18 compare routing schemes on lightly, moderately and heavily loaded configurations by varying C. Figure 15 shows that maxUTIL+HOP has around 43% lower blocking probability than maxUTIL and around 71% lower blocking probability than HOP. It illustrates that giving higher priority to shortest paths results in better and more stable performance.

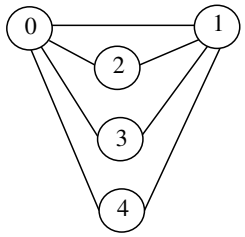

Fig. 7. Network 1: 5 nodes, 7 bidirectional links.

Table 1
Parameters of the 4 services using network 1.

$(SRC_s, DEST_s)$	$(M_s,\ m_s,\ b_s,\ D_s,\ \varepsilon_s)$	$(\lambda_s,\ \mu_s)$
(0, 1)	(30, 20, 0.1, 0.05, 10^{-4})	(20, 1)
(2, 1)	(30, 20, 0.1, 0.05, 10^{-4})	(20, 1)
(3, 1)	(30, 20, 0.1, 0.05, 10^{-4})	(20, 1)
(4, 1)	(30, 20, 0.1, 0.05, 10^{-4})	(20, 1)

Figures 15 and 16 also show that when the network is lightly to moderately loaded, all the adaptive routing schemes, sumDLY+HOP, maxUTIL+HOP and sumUTIL+HOP, have similar performance. They all outperform the static scheme HOP by around 71% lower blocking probability when $C = 900$, and by around 48% lower blocking probability when $C = 800$.

Figures 17 and 18 show that when the network is heavily loaded, sumUTIL+HOP outperforms all other schemes as it strikes a good balance between adaptiveness and over-sensitivity. maxUTIL+HOP and sumDLY+HOP exhibit oscillatory behavior that results in higher blocking probability. In Figure 17, sumUTIL+HOP outperforms maxUTIL+HOP and HOP by around

22% lower blocking probability. In Figure 18, sumUTIL+HOP outperforms maxUTIL+HOP by around 26% lower blocking probability, and outperforms sumDLY+HOP by around 20% lower blocking probability for $t \in (35 : 50]$.

Network 2

We consider the NSFNET-backbone topology shown in Figure 8. We assume all links have the same capacity C. We consider 100 services using the NSFNET-backbone, with parameters as shown in Table 2. Services with the same traffic and end-to-end QoS parameters, but with different source/destination pairs, are grouped in the same row. n $(SRC_s, DEST_s)$ means n services of this type s between nodes SRC_s and $DEST_s$ are specified. The routing update period equals 5.

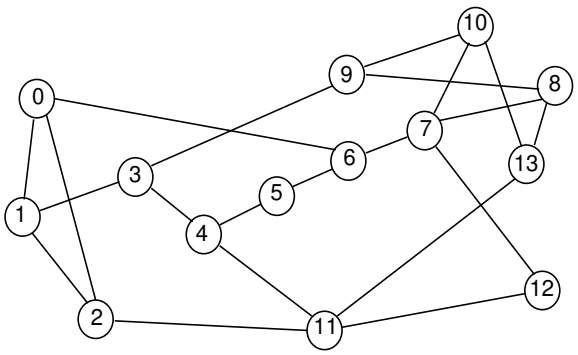

Fig. 8. Network 2: 14 nodes, 21 bidirectional links, average degree 3.

Figures 19 through 22 compare routing schemes on lightly, moderately and heavily loaded configurations by varying C. Figure 19 shows that when the network is very lightly loaded, all routing schemes have similar performance. They all give very low blocking probability of about 0.3%. maxUTIL performs slightly better than other schemes, closely followed by sumDLY+HOP, then HOP and sumUTIL+HOP. Figure 20 shows when the network is more loaded, maxUTIL+HOP exhibits some oscillations. HOP has a slightly higher blocking probability than other schemes.

Figures 21 and 22 show that when the network is moderately or heavily

Table 2

Parameters of the 100 services using Network 2.

$(SRC_s, DEST_s)$	$(M_s,\ m_s,\ b_s,\ D_s,\ \varepsilon_s)$	$(\lambda_s,\ \mu_s)$
2(0, 13),2(1, 13),2(2, 13),2(3, 13),2(4, 13),2(5, 13)	(30, 20, 0.1, 0.05, 10^{-4})	(2, 1)
1(13, 0),1(13, 1),1(13, 2),1(13, 3),1(13, 4),1(13, 5)	(30, 20, 0.1, 0.05, 10^{-4})	(2, 1)
2(6, 13),1(13, 6)	(30, 10, 0.1, 0.05, 10^{-4})	(2, 2)
2(7, 13),2(8, 13),2(9, 13),2(10, 13),2(11, 13)	(30, 10, 0.1, 0.05, 10^{-4})	(1.8, 2)
1(13, 7),1(13, 8),1(13, 9),1(13, 10),1(13, 11)	(30, 10, 0.1, 0.05, 10^{-4})	(1.8, 2)
2(12, 13),4(13, 1)	(60, 20, 0.1, 0.05, 10^{-4})	(0.3, 0.2)
2(1, 13)	(60, 20, 0.1, 0.05, 10^{-4})	(0.3, 0.2)
1(13, 12)	(60, 20, 0.1, 0.05, 10^{-4})	(0.3, 0.2)
2(0, 1)	(30, 20, 0.1, 0.05, 10^{-4})	(2, 1)
2(2, 1),2(3, 1),4(4, 1),4(5, 1),4(6, 1)	(30, 20, 0.1, 0.05, 10^{-4})	(2, 1)
4(7, 1),4(8, 1),4(9, 1),4(10, 1),4(11, 1),4(12, 1)	(30, 10, 0.1, 0.05, 10^{-4})	(1.8, 2)
1(1, 4),1(1, 5),1(1, 6)	(30, 20, 0.1, 0.05, 10^{-4})	(2, 1)
1(1, 7),1(1, 8),2(1, 9),2(1, 10),2(1, 11),2(1, 12)	(30, 10, 0.1, 0.05, 10^{-4})	(1.8, 2)

loaded, maxUTIL+HOP performs the worst. It exhibits oscillatory behavior that results in higher blocking probability. All other schemes have similar performance, and outperforms maxUTIL+HOP by around 38% lower blocking probability for $t \in (30 : 50]$ when $C = 900$, and by around 19% lower blocking probability for $t \in (30 : 50]$ when $C = 600$.

Figure 23 illustrates that with maxUTIL, admission control at the source is effective. Similar results were obtained for other schemes.

9 Conclusions

Multi-service networks have often been analyzed under steady-state conditions (e.g. [31,33,38,14,50,10,41]). In this paper, we presented a numerical-analytical method to rapidly evaluate detailed and dynamic models of multi-service 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 several dynamic routing schemes for real-time connections in terms of transient measures.

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 [55,30].

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 [19], where it was used to obtain steady-state blocking probability and carried load for a specific call routing and network topology.

Reference [19] 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 multi-service 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 ex-

tremely difficult to solve analytically [53], and computationally expensive to solve numerically [52]. A second method is that of diffusion models, which utilize averages and variances [8,44]. Such models involve partial differential equations and are usually intractable. A third method is that of fluid models, which utilize average quantities only [6]. 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] B. Awerbuch, Y. Azar, and S. Plotkin. Throughput Competitive On-Line Routing. In Proc. 34^{th} *Annual Symp. on Found. of Comp. Sci.*, Palo Alto, November 1993.

[5] 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.

[6] 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.

[7] 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.

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

[9] 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.

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

[11] 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.

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

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

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

[15] 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.

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

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

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

[19] 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.

[20] R. Gawlick, A. Kamath, S. Plotkin, and K. Ramakrishnan. Routing and Admission Control in General Topology Networks. Technical Report STAN-CS-TR-95-1548, Department of Computer Science, Stanford University, Stanford, CA 94305, 1995.

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

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

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

[24] 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.

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

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

[27] 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.

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

[29] R-H. Hwang, J. Kurose, and D. Towsley. MDP Routing in ATM Networks Using Virtual Path Concept. In Proc. *IEEE INFOCOM*, pages 1509–1517, Toronto, Ontario, Canada, June 1994.

[30] 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.

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

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

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

[34] 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.

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

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

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

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

[39] 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.

[40] I. Matta. *Fast Evaluation and Dynamic Control of Integrated Services Networks*. PhD thesis, University of Maryland at College Park, Department of Computer Science, August 1995.

[41] 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.

[42] 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.

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

[44] 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.

[45] 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.

[46] 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.

[47] 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.

[48] 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.

[49] M. Prycker. *Asynchronous Transfer Mode - Solution for Broadband ISDN*. Prentice Hall, 1995.

[50] 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.

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

[52] 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.

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

[54] 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.

[55] 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.

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

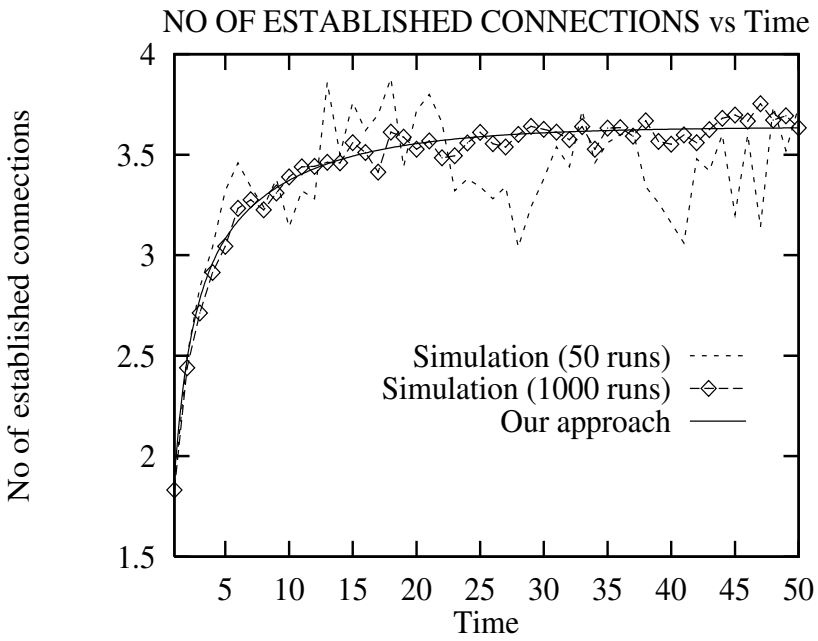

Fig. 9. Total number of established connections versus time. Single-link network.

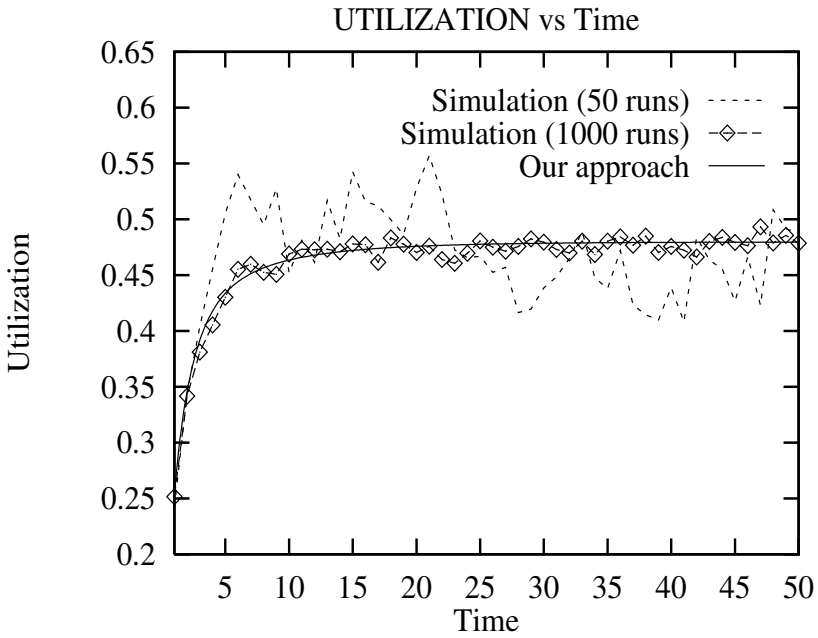

Fig. 10. Link utilization versus time. Single-link network.

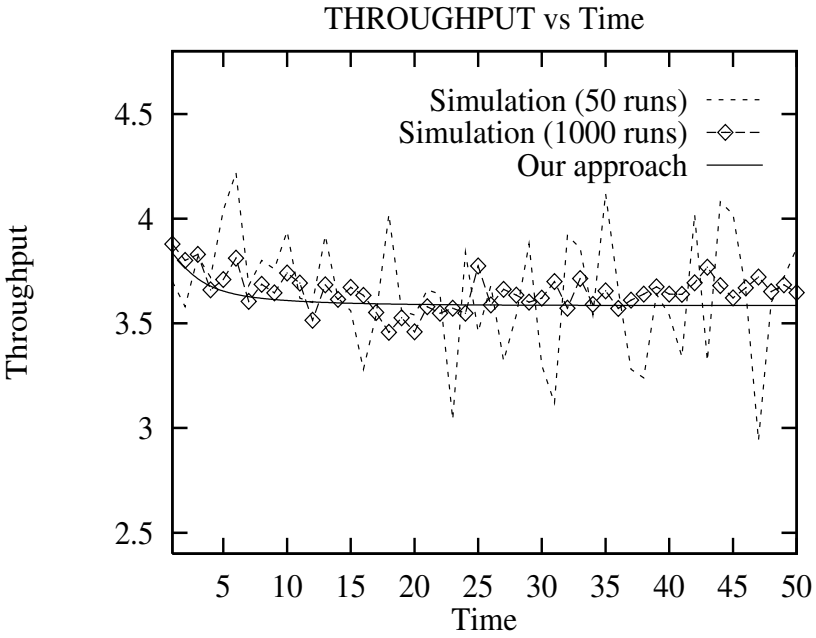

Fig. 11. Total throughput versus time. Single-link network.

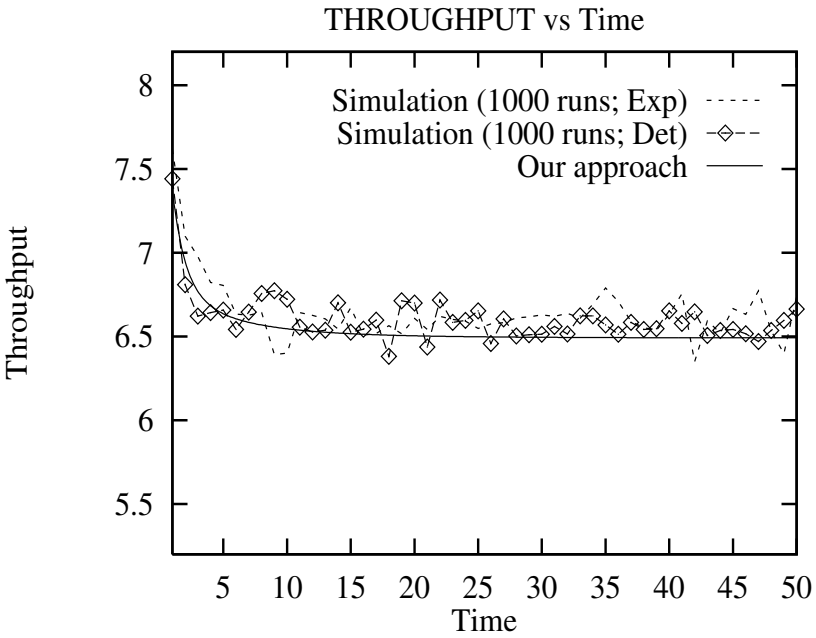

Fig. 12. Total throughput versus time. Multi-link network.

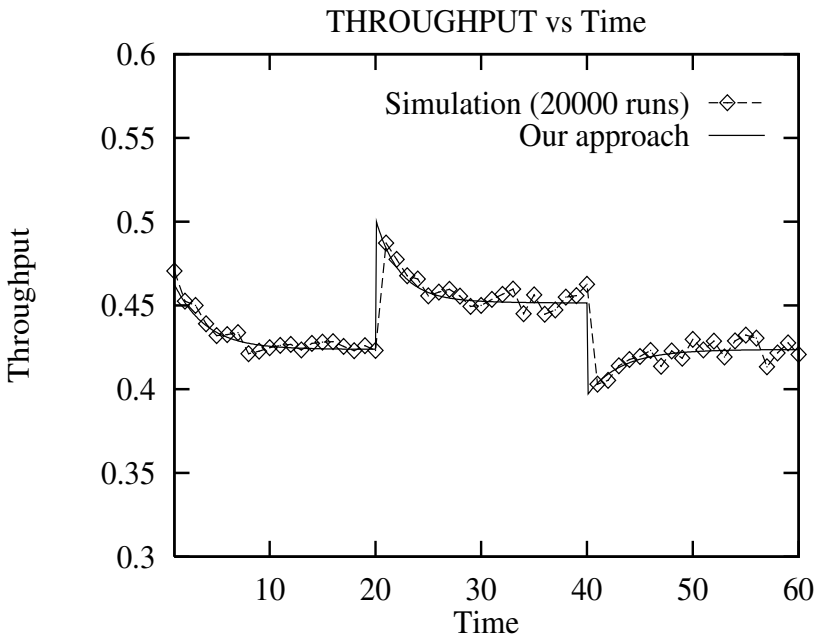

Fig. 13. Total throughput versus time. Multi-link network. Time-varying arrivals.

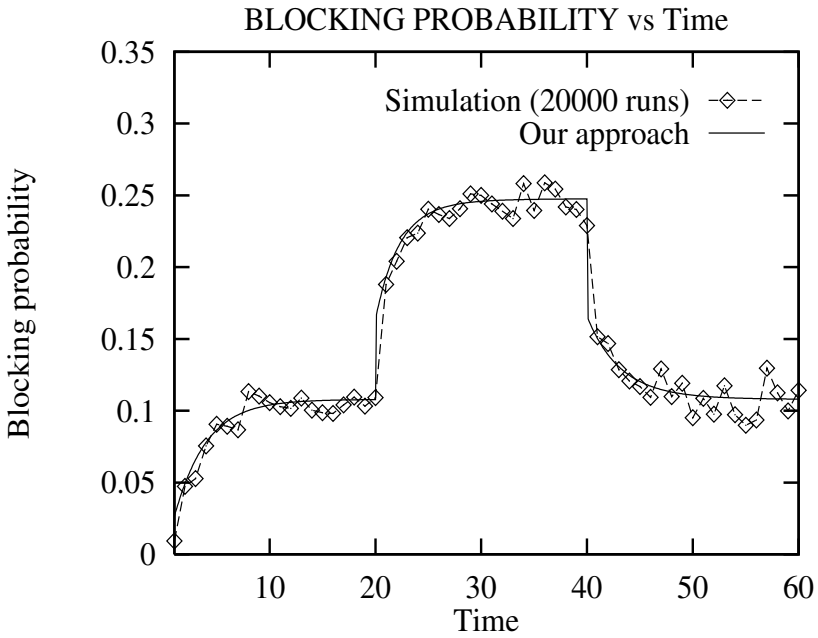

Fig. 14. Blocking probability versus time. Multi-link network. Time-varying arrivals.

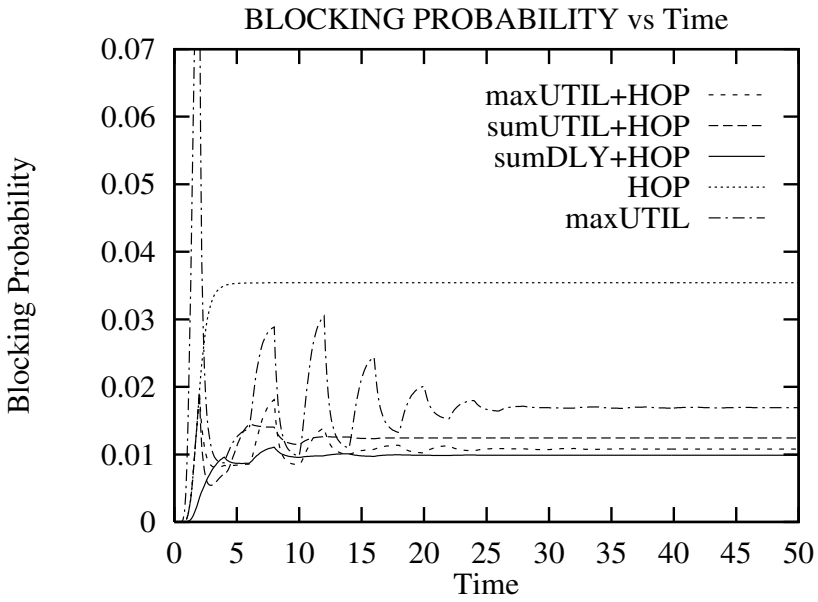

Fig. 15. Blocking probability versus time. Network 1 with $C = 900$.

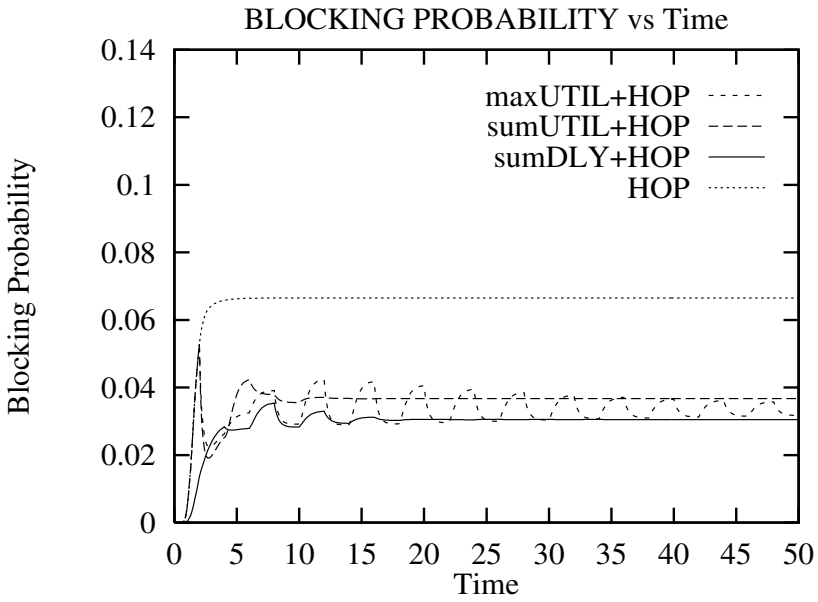

Fig. 16. Blocking probability versus time. Network 1 with $C = 800$.

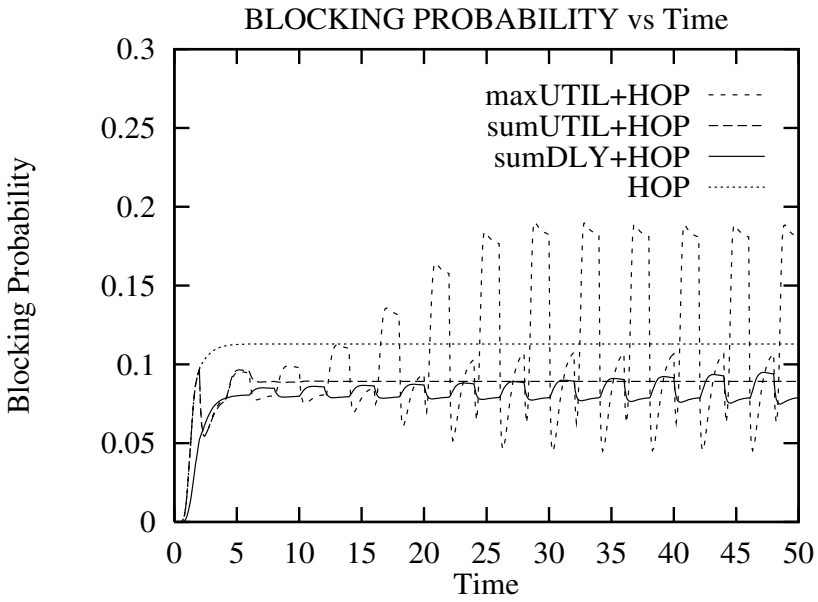

Fig. 17. Blocking probability versus time. Network 1 with $C = 700$.

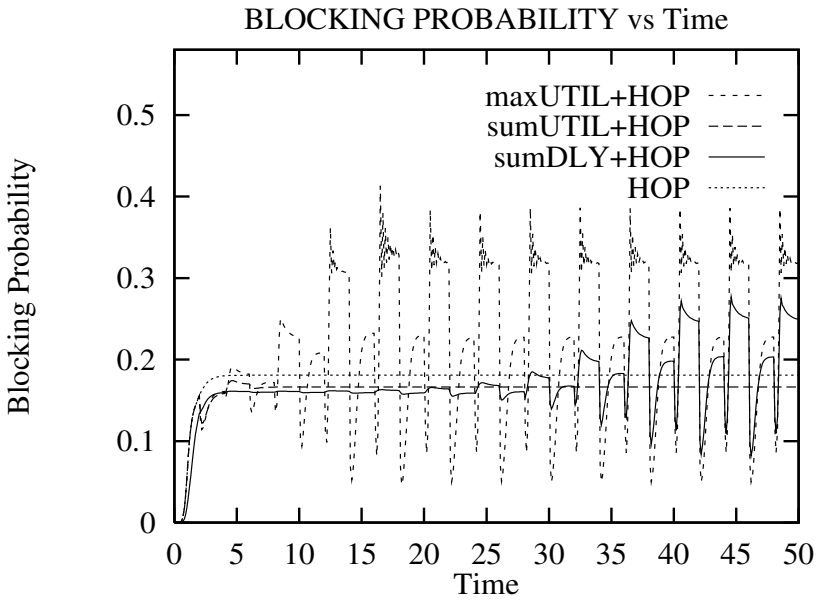

Fig. 18. Blocking probability versus time. Network 1 with $C = 600$.

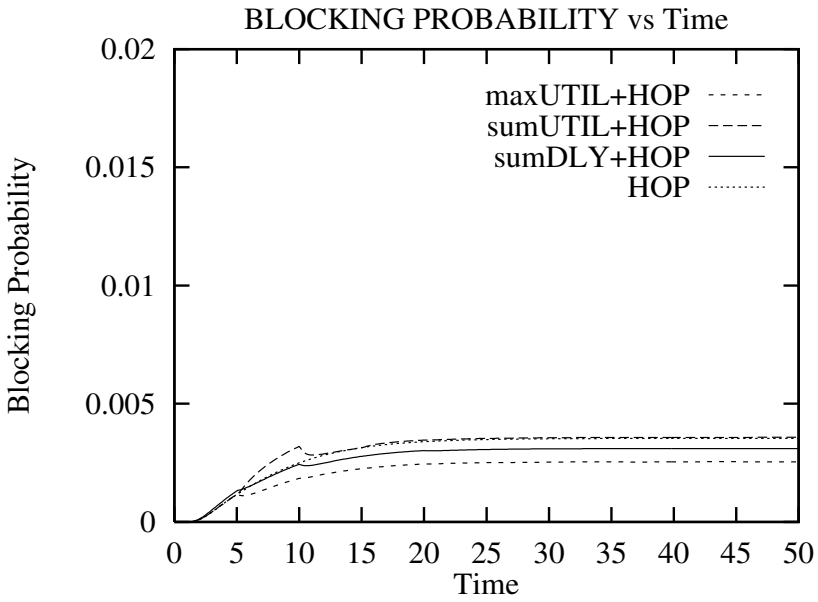

Fig. 19. Blocking probability versus time. Network 2 with $C = 1200$.

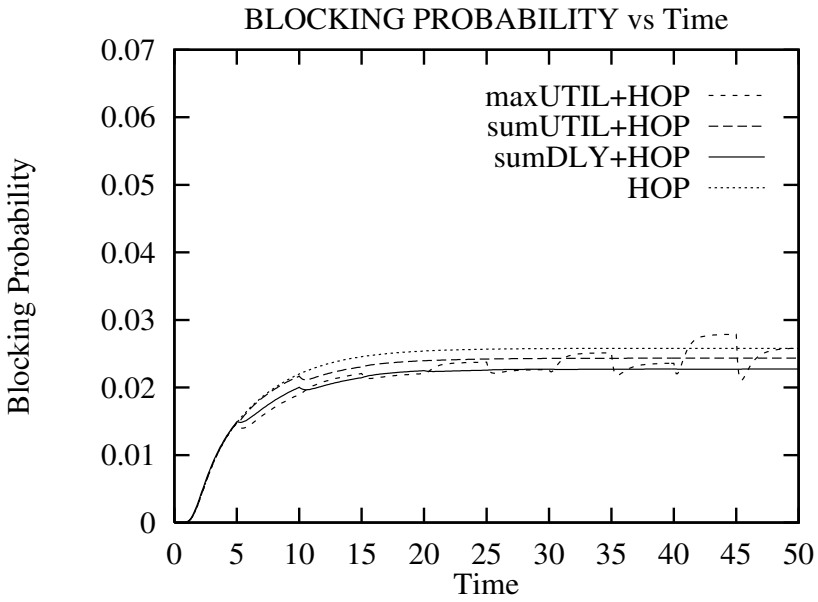

Fig. 20. Blocking probability versus time. Network 2 with $C = 1000$.

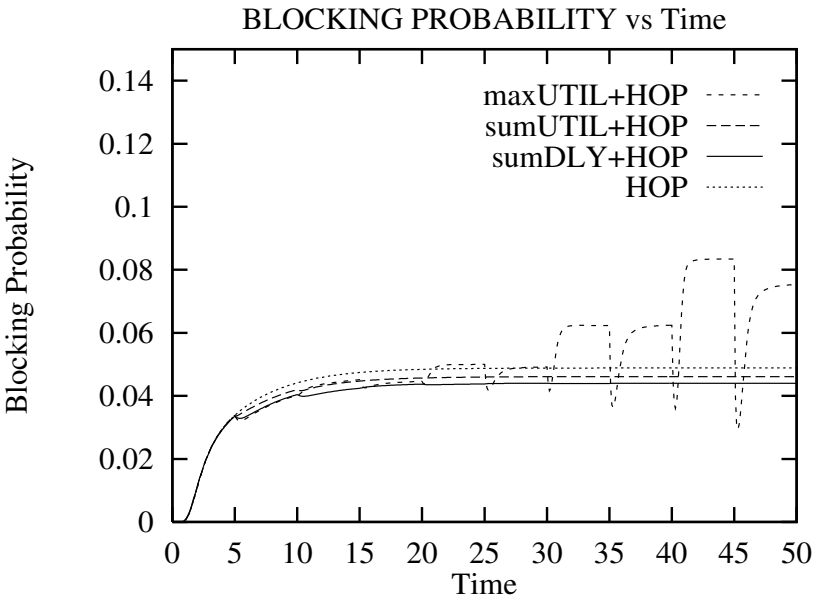

Fig. 21. Blocking probability versus time. Network 2 with $C = 900$.

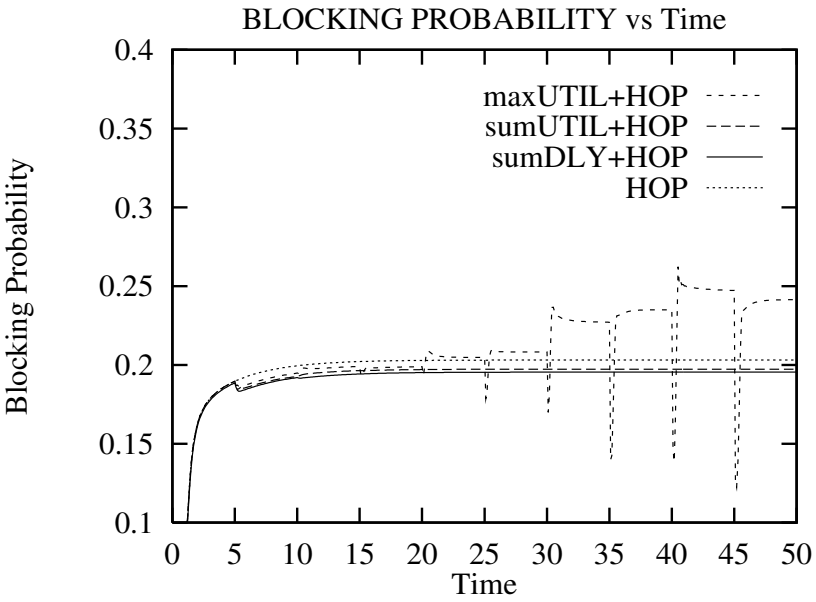

Fig. 22. Blocking probability versus time. Network 2 with $C = 600$.

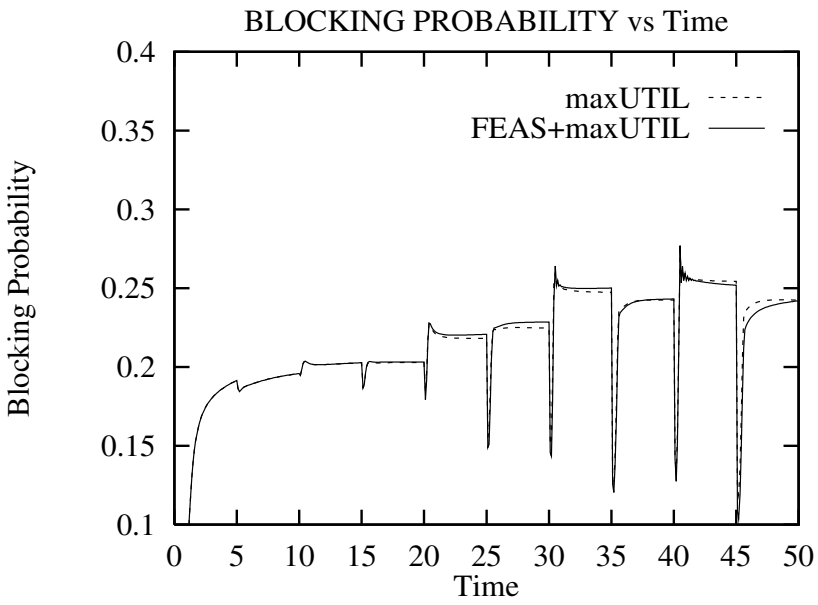

Fig. 23. Blocking probability versus time. Network 2 with $C = 600$.