On Call Processing Delay in High Speed Networks

Ren-Hung Hwang and James F. Kurose and Don Towsley

Abstract— **In future BISDN networks, significant burdens will be placed on the processing elements in the network since call routing and admission policies will be more computationally intensive than those in present day networks. Thus, the bottleneck in future networks is likely to shift from the communication links to the processing elements. The delays at these elements are influenced by their processing capacity and factors such as routing algorithms, propagation delays, admission control functions, and network topology. The goal of this paper is to characterize the behavior of these factors on the call setup time and accepted call throughput. This behavior is examined for three sequential routing schemes and two flooding routing schemes under various network parameters and different forms of admission control. The results of our study indicate that processing capacity and the admission control function can affect the call setup time and accepted call throughput significantly while propagation delay does not affect these performance measures significantly.**

1 Introduction

In future broadband ISDN (BISDN) networks, significant burdens will be placed on the processing elements in the network as call routing and admission policies become more computationally intensive than those in present day networks (due to the need to ensure that accepted calls will be provided with a guaranteed quality of service (QOS) [1, 2]). In this paper, we study the *computational* delays associated with call admission, routing and call setup in BISDN networks.

Routing in future BISDN networks will contain elements from both traditional packet-switched and circuit-switched networks. The CCITT specifications on ATM [3] specify a cell-based, packet-like transport mode within the network. However, the need to provide a guaranteed QOS has resulted in the need for a *call-level* admission control mechanism and the reservation of resources (e.g., bandwidth [4]) by a call on a link-by-link basis. In this latter case, when a call is "offered" to a route, computation is required to determine whether the selected route can indeed support the additional call while continuing to meet the QOS guarantees of existing calls. The manner in which calls are offered to the various routes (e.g., the order in which routes are attempted and the decision as to whether multiple call setup paths will be attempted in parallel) will clearly influence the call setup time as well as the maximum call arrival rate that can be supported by the network call processing elements. Network parameters such as call processing capacities, propagation delays, and path lengths will also influence these performance measures. It is the influence of these routing, call setup and network characteristics on the call setup time and call processing capacity that we seek to characterize in this paper.

The influence of routing on the call setup time and call processing capacity is examined with five routing algorithms: three well-known algorithms in the circuit-switching literature and two controlled flooding versions of these algorithms. The influence of call admission control is modeled and examined for different forms of a "call admission control functions". The NSF T3 network is used in our examples as the underlying network topology. We develop analytical models for evaluating the average call setup time and call blocking probability for these five algorithms and validate our analysis through simulations. The results of our study indicate that for a given exogenous call arrival rate, there is a trade off between call setup time and call blocking probability. The crankback routing scheme yields the lowest call blocking probability but also the highest call setup delay. On the other hand, the SOC routing scheme yields the highest call blocking probability but also the lowest call setup delay. As expected, flooding algorithms yield smaller call setup delays only when the call processing capacity is abundant. We also find that the form of the admission control function (to be defined precisely later) has a significant influence on the call setup time and the processing loads at processing elements. The influence of propagation delay is examined in [5] and shown to not affect the call setup delay significantly as long as it is relatively small as compared to the call holding time.

The focus of this paper is rather different from previous works on routing in the literature. The analytical models we have developed are based on the link-decomposition method [6], a commonly-used technique in evaluating network performance in the circuit-switching literature. Based on this method, a number of methods have been proposed for computing the end-to-end blocking probability in the circuit-switching literature, e.g. [7, 8, 9, 10, 11, 12]. Whitt [13] also presents a model for calculating the blocking probability in setting up virtual circuits with fixed-path routing in packet-switched networks. All of these works, however, have focussed on computing the blocking probability and ignore the call set up delay. Also, a common assumption made by these works is that a call is either set up in zero

R.-H. Hwang is with the Dept. of Computer Science and Information Engineering, National Chung Cheng University, Chia-Yi, Taiwan, R.O.C.. J. Kurose and D. Towsley are with the Dept. of Computer Science, University of Massachusetts.

time (if there is sufficient bandwidth) or cleared in zero time (if blocked on some link). However, these assumptions are not necessarily true in high-speed networks. For example, a connection for transferring a 1MB file on a 1GB network only lasts for 1 millisecond which is about the same order of magnitude of call setup/clearing time. Thus, in this paper, we are particularly concerned with accurately modeling the call setup time, and studying the effects of different call setup policies and non-zero call clearing time on the call setup time.

One recent work which has examined the processing aspects of design and control in future BISDN networks is [14]. In [14], the authors identify the impact of limited processor capacity on the design and control of high-speed packet-switched networks. General guidelines for processor-limited routing and congestion control algorithms for such networks are discussed. In our work, we limit ourselves to the call setup problem while considering additional factors, such as propagation delay and QOS requirements. We also provide quantitative results.

The remainder of this paper is structured as follows. In section 2, we discuss our model of the network. In section 3, we specify the routing mechanisms examined. Section 4 describes the analytical models for different routing mechanisms. The results of our investigation on the influence of processing delay, admission control function, and routing algorithms on the average call setup delay and accepted call throughput are presented in section 5. The influence of propagation delay on routing, presented in [5], is summarized in section 6. Section 6 also discusses the issue of whether it is important to model resources held by blocked calls. Finally, section 7 summarizes this chapter.

2 Network Models

We consider a connection-oriented high-speed network consisting of N nodes with each node having some number of incoming and outgoing links. We adopt the network node structure described in [15, 16], in which a node consists of two components: the switching subsystem (SS) and the network control unit (NCU). The SS is a fast hardware switch with relatively limited functionality. The NCU is a slower, but more sophisticated, processor. Packets (or cells) that need only be relayed through the node are handled by the SS directly without the involvement of the NCU. We further assume that control packets associated with routing are the only control packets processed by the NCU. Each NCU is assumed to have a sufficiently large buffer to avoid the loss of routing control packets due to buffer overflow.

External connection requests, each having an associated QOS requirement, can arrive at any node in the network. When a call request arrives, the routing algorithm (which resides in the NCU's of the network nodes) chooses a path for the call from a set of possible paths (according to the rules specified in next section) and then proceeds as follows. First, the source node invokes the admission control function to check if the new call can be accepted on the first link on the path. The call is accepted on the link if sufficient bandwidth is available on the link to meet the new call's QOS requirement, while maintaining the agreed-upon QOS for existing calls. If the call is accepted on the link, a certain amount of the bandwidth, determined by the QOS requirement, is reserved for the call. The source node then passes the call request to its downstream neighbor on the chosen path. This neighbor then passes the call request to its downstream neighbor if the QOS can also be guaranteed at the next link. This process continues until either the request is successfully passed to the destination node or the request cannot be passed further toward the destination node by an intermediate node on the path. In the first case, the call is established and the resources reserved along the path during the call set-up phase provide the guaranteed QOS required by the call. In the second case, the call cannot be established on the path, bandwidth that has been reserved for this call is released, and another path, if available, may then be tried.

3 Specification of Routing Algorithms

As described in section 2, the call routing algorithm residing in the NCU specifies which path should be tried first as well as which alternate paths should be tried next when a call request is blocked on a path. We assume that, for each source-destination pair, there is a predefined *routing tree* available at the source node [6]. A routing tree can be viewed as a predefined set of possible paths connecting the source and destination node. These paths are tried in some order defined by a routing algorithm when a new call setup arrives. Three routing algorithms, which use knowledge of the network topology but no explicit network status information (such as current link delay or traffic load) are studied; these algorithms are well-known in the circuit-switching literature. Two controlled flooding algorithms based on these three algorithms will also be investigated. We identify the original three algorithms as sequential routing algorithms and the modified algorithms as parallel algorithms. Specifications of the five routing algorithms are described as follows.

1. **Original Office Control (OOC) routing:** According to this algorithm, the choice of the next path to try is always made by the source node. When a call request is rejected on a link on the chosen path, a blocking message is returned along the path back to the source node, bandwidth reserved previously is released, and the source node sends another call request on the path to be tried next. The call is blocked if a blocked message is received on the last path.
2. **Sequential Office Control (SOC) routing:** The SOC control algorithm is also called progressive control, sequential control, or spill-forward without crankback. This algorithm tries to improve upon the OOC control algorithm by allowing the intermediate nodes to react

to link blocking. Each intermediate node is given a set of possible outgoing links to try. When a call request is blocked on one of the links, instead of immediately returning a blocking message to the source node, the intermediate node tries to set up the call on another link from the set. If none of the possible outgoing links at the intermediate node are able to provide the required QOS, the call is blocked.

3. **Crankback routing:** Crankback routing is also called spill-forward with crankback. It improves upon the SOC algorithm by allowing a blocked message to be sent back to upstream nodes in the routing tree. That is, a node is blocked if all of its possible outgoing links are blocked and a call is blocked if the source node is blocked.

Instead of trying downstream neighbors one at a time, the controlled-flooding versions of these algorithms try all downstream neighbors simultaneously. They are referred to as controlled flooding algorithms because call requests are sent only to neighbors that appear in the routing graph. For the OOC controlled flooding algorithm, only the source node sends out multiple call request messages. Controlled flooding versions of SOC and Crankback collapse into one identical algorithm in which nodes with more than one downstream neighbor send out simultaneous multiple call request messages.

4 Analytical Models

In studying the influence of network characteristics such as routing algorithms, the network performance measures in which we are interested are the average end-to-end call set up delay (which includes call processing delays and propagation delays) and the call blocking probability. The influence of network characteristics on these performance metrics will be examined through analytical models. In this section, we present the analytical model for the sequential OOC routing algorithm for general network topologies. The reader is referred to [17] for the analytical models for the other four routing algorithms. In our analytical models, we make the following assumptions and notations.

1. The network is modeled as a directed graph $G = (V, E)$.
2. Let W be the set of all O-D pairs, $W = \{(a, b) : a, b \in V\} = \{w\}$.
3. Let R_w be the routing tree of O-D pair w.
4. The network topology and routing trees are fixed.
5. External call arrivals to each O-D pair, w, are assumed to be governed by a stationary Poisson process with parameter λ_w.
6. The call holding time, denoted by τ, for each call is assumed to be arbitrarily distributed with mean $\bar{\tau}$.
7. Call request processing (service) times at each node are assumed to be exponentially distributed. The average call processing delay through node x is denoted by $\bar{T}_x$.
8. The propagation delay of each link, denoted by D_ℓ for link ℓ, is assumed to be constant.
9. We assume that call requests are transmitted on dedicated channels and do not contend for transmission media. We also assume that they are not lost in the switches (or NCU's) and their transmission delays are negligible (due to the extremely high transmission rate).
10. The release of resources, either due to call termination or call abortion, consumes zero processing time.

Resources on a link are reserved either for the entire call holding time or a short *blocked-call-clear time*, the time from when the resource is first reserved by a call until it is released, due to the call being blocked downstream. This blocked-call-clear time includes the downstream call processing delays and propagation delays. Unlike [7, 8, 9, 10, 11, 12], we do not assume that call setup time and the blocked-call-clear time are negligible. Indeed, modeling these non-negligible delays and determining their effect on call setup times is one of the goals of this research.

The design of algorithms for deciding whether to admit/reject the call on a given link is beyond the scope of this research. However, we model the effects of such algorithms in the following manner. From the standpoint of admission control, the number of existing connections together with their QOS requirements is the minimal information needed to make new connection acceptance decisions. Note that these existing connections should include not only connections already established but also connections that are in the process of being set up, having already reserved resources on a link. In this study, we assume that the total number of existing connections on a link is the only parameter that affects admission control. Define $B_\ell(x)$ as the probability that link ℓ cannot guarantee the QOS for a new call or for some existing call given the addition of the new call, where x is a measure of the current number of existing calls at link ℓ. The rationale for such a nondeterministic "call admission function" is that external calls may require different amounts of resources. Thus $B_\ell(x)$ gives the probability that the resource requirements of an arriving call exceeds the remaining available capacity of link ℓ, causing the call to be blocked[1]. Modeling the call admission control in this manner enables us to decouple specific QOS mechanisms from the routing issues which are the main focus of this research. In our numerical examples, we examine the mean call setup delays and accepted call throughputs associated with various routing algorithms and network parameters when $B_\ell(x)$ is concave, convex, or linear in x.

Our use of $B_\ell(x)$ allows us to model the interaction between the call admission and routing problems at a high level of abstraction. Specifically, we do not consider *how* (i.e., on what basis) the call admission decision is made.

[1]Note that if all calls have the same QOS requirements, $B_\ell(x)$ becomes a step function. That is, $B_\ell(x) = 0$ if $x < C_\ell$ and $B_\ell(x) = 1$ if $x = C_\ell$, where $< C_\ell$ is the link capacity. However, future high-speed networks are expected to support various types of traffic with different QOS requirements.

Rather, we simply use the fact that a decision is made (as modeled by our $B_\ell(x)$ function) in examining the effects of call processing delays on call routing algorithms. The fact that $B_\ell(x)$ is a probability models the heterogenous nature of the calls in the sense that for a given state, x, one offered call's QOS requirements and resource demands may be such that the call is rejected, while for the same state, x, another call's QOS requirements and demands may be different, causing this second call to be accepted. Also, given two calls with identical resource requirements and identical state x, one call might be accepted and one call might be rejected. This is because the number of calls alone does uniquely determine the resources required by a set of calls, in the case that call requirements are not identical.

The computation of $B_\ell(x)$ itself is beyond the scope of this paper, as it depends strongly on the definition of the resources being reserved, the QOS requirements being made, the underlying scheduling mechanism, and the philosophy of the call admission process (e.g., hard guarantees versus statistical guarantees). The computation of $B_\ell(x)$ might well require a more detailed state structure (e.g., recording the bandwidth requirements of accepted call), although we note that work on blocking approximations in multirate circuit-switched networks [18, 19] suggest that this need not be the case.

An exact analytical model for evaluating the performance of the system is too complicated to be of practical use. However, given the offered traffic to an individual link and node, we generally can have quite simple but accurate models for computing performance measures for the individual links and nodes. An approximation technique, the so-called link-decomposition method [6], from circuit-switched networks will be used to evaluate the network performance metrics in which we are interested. The original link-decomposition method decomposes the overall network problem into a set of independent link problems. After the performance of each link is computed, the overall network performance such as mean end-to-end call setup delay and call blocking probability can be recovered from the individual link-performance measures such as nodal processing delay and link blocking probability. It has been shown in the literature [7, 8, 9, 10, 11, 12] that this decomposition method yields reasonable estimates of performance metrics in circuit-switched networks. Additional assumptions required for the link-decomposition method include:

- Call set-up requests (either originating at the node or being received from "upstream" nodes) to a node are assumed to arrive according to a Poisson process.
- Calls are blocked independently at all links [7, 8, 13].

The difficulty with the link-decomposition method arises from the fact that the link-offered traffic and node-offered traffic are unknown and coupled with the individual link and node performance measures. The coupling between the link- and node-offered traffic and the link blocking probability and the mean nodal processing delay yields a fixed point problem. A numerical method used most frequently to solve the fixed point problem is the so called relaxation method. By using the relaxation method, the network-level and link/node-level performance measures are obtained by using an iterative procedure which, at each iteration, does the following:

Algorithm A:

1. Given the individual link-blocking probabilities (due to the admission control), compute the traffic offered to each link and node.
2. The resulting link- and node-offered traffic in turn yield, via simple link and node models, a new set of values for the link-blocking probabilities.

We first discuss how to calculate the nodal processing delays and link-blocking probabilities and then show how the offered traffic to each link and node are computed for both sequential and parallel OOC routing algorithms specified in the previous section. Finally, we show how the end-to-end call set up delay and call blocking probabilities are calculated for both routing policies.

4.1 Computing the Average Nodal Processing Delay

As discussed earlier, when call setup requests arrive at a node, a certain amount of computation must be performed in order to determine whether to admit/reject the new call. We refer to the delay associated with the computation (both waiting for processing by the NCU as well as NCU processing itself) as the *nodal processing delay*. In addition to assuming that exogenous call arrivals are Poisson, we further assume that the arrival process of call setup requests to each node (NCU) is a Poisson process [7, 8, 13]. Define X to be the time required by an NCU to process a call request (hereafter, we refer this as the call processing time). Recall that X is an exponential random variable. Therefore, the average call processing delay (queueing delay plus service time), $\bar{T}$, is given by [20],

$$\bar{T} = \frac{\bar{X}}{1 - \lambda \bar{X}},$$

where λ is the arrival rate and $\bar{X}$ is the mean call processing (service) time at a node. (The notation $\bar{Y}$ will be used to represent the mean of a random variable Y throughout this paper.)

Besides the average processing delay, the average end-to-end call set up delay also includes the propagation delay on links traversed by a call request. The propagation delay will be modeled as a delay center with deterministic service time.

4.2 Computing the Link-blocking Probabilities

Before we compute the link-blocking probability (i.e., the probability that an arriving call cannot obtain the link resources it needs to satisfy its QOS requirement), it is very

important to clearly make the distinction between node-offered call-requests and link-offered call requests. The link-offered call requests are the calls offered by call setups to the *single* link under consideration. The node-offered call requests are the calls offered to *any* of the outgoing links of this node. In the case of OOC and Crankback control, these call requests include the overflow call requests from previous paths.

A call request can be blocked on a link due to admission control. From the standpoint of admission control, the number of existing connections together with their QOS requirements is the minimal information needed to make new connection acceptance decisions. Note that these existing connections should include not only connections already established but also connections that are in the process of being set up, having already reserved resources on a link. In this study, we assume that the total number of existing connections on the link is the only parameter that affects admission control. Since different calls may have different QOS requirements, we model the admission control by a function, $B_\ell(x)$, as noted earlier. When a call request tries to reserve bandwidth (according to its QOS requirement), the request will be rejected with probability $B_\ell(x)$, which depends on the number of existing calls, x.

Let us consider a single link in isolation, as shown in Figure 1. (Note that the call setup requests arriving at the node shown in the figure are only the requests that are intended to be sent to the link under consideration; the call requests that come to the node to be sent out on other outgoing links of this node are not shown.) As we will see shortly, the incoming requests to a link can be divided into a number of distinct classes (e.g., in the case of OOC control, each class will be distinguished by being on the ith path attempted between a source/destination pair and being the jth link on that path). We refer to the arrival rate of class ι calls as γ_ι and define $\Gamma = \sum_{\iota=1}^{n} \gamma_\iota$. Each class of calls can be further divided into two types. The first type of calls, which will eventually be accepted if not blocked on the link under investigation, has an arrival rate of γ_ι^s. The second type of calls, which has an arrival rate of γ_ι^f, will be blocked downstream if not blocked on this link. Here, the superscripts s and f refer to whether a call request is eventually successful on this path or fails (is blocked). Let δ_ι^s and δ_ι^f denote the mean resource holding times for the two classes of calls. Note that a busy server in a multiserver queue shown in Figure 1 models a call that has successfully reserved resource at this link (these resources may be held for the duration of a call or may be released shortly if a call is blocked downstream). Since the link capacity is limited, the maximum number of such busy servers in Figure 1 is limited and is denoted by C_ℓ.

The steady state blocking probability of a link can be obtained by solving for the steady state distribution of the number of calls that are holding resources (i.e., are resident in the multiserver queues) on the link. For computing the steady state distribution of the number of calls (which will eventually either be accepted or blocked downstream) on the link, the original call blocking model shown in Figure 1 can be shown to be equivalent to the $M/G/C_\ell/C_\ell$ queue shown in Figure 2. The mean service time, $1/\mu$, for this $M/G/C_\ell/C_\ell$ queue is given by

$$\frac{1}{\mu} = \sum_{\iota=1}^{n} \frac{\gamma_\iota^s}{\Gamma} \cdot \delta_\iota^s + \sum_{\iota=1}^{n} \frac{\gamma_\iota^f}{\Gamma} \cdot \delta_\iota^f.$$

The arrival rate, $\tilde{\lambda}$, for this queue is state dependent and is given by

$$\tilde{\lambda}(\iota) = \Gamma \cdot (1 - B_\ell(\iota)) \quad \iota = 0, ..., C_\ell - 1.$$

Given the arrival rate and service rate, we can compute the steady state distribution of the number of calls currently allocated bandwidth at this link, $\Pi = (\pi_0, \pi_1, ..., \pi_C)$ where π_x is the steady state probability that there are x calls holding resources at this link and is given by [21]

$$\pi_x = \frac{\prod_{i=0}^{x-1}(\tilde{\lambda}(i)/\mu)/x!}{\sum_{j=0}^{C}\prod_{i=0}^{j-1}(\tilde{\lambda}(i)/\mu)/j!}.$$

Once Π is known, the probability that an arriving call is blocked at this link due to admission control is given by

$$\mathcal{L}_\ell = \sum_{\iota=0}^{C_\ell} \pi_\iota \cdot B(\iota) \quad \iota = 0, ..., C_\ell.$$

4.3 Offered Call Requests, Mean Call Setup Delay, and Call Blocking Probability

The computation of the link- and node-offered call arrival rate is complicated and depends heavily on the routing tree used. In this section, we show how the link- and node-offered call arrival rates are computed for the sequential OOC routing algorithm. The following notation is used in the analysis:

- As shown in Figure 3, let k_w be the number of paths in the OOC routing tree R_w. R_w is an ordered set of paths, that is,

$$\begin{aligned} R_w &= (R_w^1, R_w^2, ..., R_w^{k_w}), \\ R_w^i &= (a_{w,i,1}, ..., a_{w,i,\ell_w^i+1}), \end{aligned}$$

  where R_w^i is the ith path in routing tree R_w of length ℓ_w^i, $a_{w,i,1}$ is the source node and a_{w,i,ℓ_w^i+1} is the destination node ($\forall i$). Let $R_w^{i,j}$ denote the jth node of the path R_w^i.
- Let Ψ_x be the set of paths which contain node x,

$$\Psi_x = \{R_w^i | x \in R_w^i\}.$$

- Let $\Psi_{(x,y)}$ be the set of paths which consist of link (x, y),

$$\Psi_{(x,y)} = \{R_w^i | (x, y) \in L(R_w^i)\},$$

  where $L(r)$ is the set of links on route r. If $r = (a_1, a_2, ..., a_n)$, then

$$L(r) = \{(a_1, a_2), (a_2, a_3), ..., (a_{n-1}, a_n)\}.$$

- Let $Sub(r,i,j)$ be a sub-vector of r defined by

  $Sub((a_1,...,a_n),i,j) = (a_i,a_{i+1},...,a_j), \;\; 1 \le i \le j \le n.$

- Let $\mathcal{L}_\ell$ be the steady state blocking probability of link ℓ.
- Let P_w^{rej} be the end-to-end call blocking probability of O-D pair w.
- Let $\bar{T}_w^{setup}$ be the average end-to-end call set up delay of O-D pair w.
- Let $P_{w,i}^f$ be the probability that a call request is blocked on the ith path of R_w.
- Let $\mathcal{T}_{w,i,j}^s$ be the delay from the time a call request is sent by the jth node of path R_w^i until it is successfully received by the destination node given that the call request will be eventually successfully received.
- Let $\mathcal{T}_{w,i,j}^f$ be the delay from the time a call request is sent by the jth node of path R_w^i until it is blocked at some node downstream given that the call request will be blocked some node later on.

4.3.1 Sequential OOC Routing Algorithm

First, let us calculate certain quantities that will be needed in the computation of both the link-offered call request rate, call setup delay and call blocking probability. Under the independence assumption, the probability that a call fails on the ith path of R_w, $P_{w,i}^f$, is given by

$$P_{w,i}^f = \sum_{j=1}^{\ell_w^i} \mathcal{L}_{(R_w^{i,j},R_w^{i,j+1})}[\prod_{\ell \in L(Sub(R_w^i,1,j))}(1-\mathcal{L}_\ell)] \quad (1)$$

The mean delay from the time a call request is sent by the jth node of path R_w^i until it is successfully received by the destination node is given by

$$\bar{\mathcal{T}}_{w,i,j}^s = \sum_{a \in Sub(R_w^i,j+1,\ell_w^i)} \bar{T}_a + \sum_{\ell \in L(Sub(R_w^i,j,\ell_w^i+1))} D_\ell \quad (2)$$

where the first term in the right hand side of the equation is the total processing delay and the second term is the total propagation delay.

Similarly, the average delay from the time a call request is sent by the jth node of path R_w^i until it is blocked at some node downstream given that the call request will be blocked some node later on is given by

$$\bar{\mathcal{T}}_{w,i,j}^f = \sum_{n=j+1}^{\ell_w^i} [(\sum_{a \in Sub(R_w^i,j+1,n)} \bar{T}_a + \sum_{\ell \in L(Sub(R_w^i,j,n))} D_\ell) \times P_{w,i,j,n}^f], \quad (3)$$

where $P_{w,i,j,n}^f$ is the probability that it is blocked at the nth node given that it is blocked at some node after the jth node on path R_w^i.

$$P_{w,i,j,n}^f = \{[\prod_{\ell \in L(Sub(R_w^i,j+1,n))}(1-\mathcal{L}_\ell)]\mathcal{L}_{(R_w^{i,n},R_w^{i,n+1})}\}/$$

$$\{\sum_{m=j+1}^{\ell_w^i}[\prod_{\ell \in L(Sub(R_w^i,j+1,m))}(1-\mathcal{L}_\ell)]\mathcal{L}_{(R_w^{i,m},R_w^{i,m+1})}\}$$

Node-offered call requests

Let us now focus on a single *node*, say node x, in isolation. First, let us consider the case in which the node x is reached as the jth node of path i in route tree R_w. Assume that when a call request is blocked on the source node's jth outgoing link, the call request is immediately tried on the $(j+1)$st link. The rate of call requests offered to node x from path R_w^i, denoted by $g_{x,w,i}$, is given by:

$$g_{x,w,i} = \begin{cases} \lambda_w[\prod_{m=1}^{i-2} P_{w,m}^f](P_{w,i-1}^f - \mathcal{L}_{(R_w^{i-1,1},R_w^{i-1,2})}) \\ \quad \text{if } x \text{ is the source node of } R_w, \\ \lambda_w[\prod_{m=1}^{i-1} P_{w,m}^f][\prod_{\ell \in L(Sub(R_w^i,1,j))}(1-\mathcal{L}_\ell)] \\ \quad \text{otherwise.} \end{cases} \quad (4)$$

The aggregated call request rate offered to node x, G_x, is then given by

$$G_x = \sum_{R_w^i \in \Psi_x} g_{x,w,i}. \quad (5)$$

Link-offered call requests

Let us now focus on a single *link*, say link ℓ, in the network. Consider the case in which the link ℓ is reached as the jth link of path R_w^i. As shown in Figure 1, the incoming traffic to a link is divided into several classes; each class corresponding to the traffic offered by each path that includes ℓ. (Thus the number of classes of traffic on a link depends on the number of routing paths which consist of this link.) Let $\gamma_{\ell,w,i}$ be the offered call request rate to this link from path R_w^i. Considering the blocking probability of previous paths and nodes, $\gamma_{\ell,w,i}$ is given by

$$\gamma_{\ell,w,i} = \lambda_w[\prod_{m=1}^{i-1} P_{w,m}^f][\prod_{\ell' \in L(Sub(R_w^i,1,j))}(1-\mathcal{L}_{\ell'})]. \quad (6)$$

Recall that each class of offered call requests to link ℓ can be further divided into two types: those that are eventually successful on this path and those that fail. The arrival rate of each type of offered call requests, $\gamma_{\ell,w,i}^s$ and $\gamma_{\ell,w,i}^f$ respectively, and the mean call holding time, $\delta_{\ell,w,i}^s$ and $\delta_{\ell,w,i}^f$, are computed as follows,

$$\gamma_{\ell,w,i}^s = \lambda_w[\prod_{m=1}^{i-1} P_{w,m}^f](1-P_{w,i}^f), \quad (7)$$

$$\gamma_{\ell,w,i}^f = \gamma_{\ell,w,i} - \gamma_{\ell,w,i}^s, \quad (8)$$

$$\delta_{\ell,w,i}^s = \bar{\tau} + \bar{\mathcal{T}}_{w,i,j}^s, \quad (9)$$

$$\delta_{\ell,w,i}^f = \bar{\mathcal{T}}_{w,i,j}^f. \quad (10)$$

Recall that $\bar{\tau}$ is the mean call holding time. Relations (7)-(10) can be used to compute the blocking probability on the link during an iteration of **Algorithm A** discussed in Section 4.2.

Mean Call setup delay and call blocking probability

Since a call is lost if it is blocked on all paths, the probability that a call request of O-D pair w is rejected, P_w^{rej}, is given by

$$P_w^{rej} = \prod_{i=1}^{k_w} P_{w,i}^f. \quad (11)$$

Let $\bar{T}_{w,i}$ be the expected time for the source node of w to receive a blocking message from path i given that it is blocked at path R_w^i.

$$\bar{T}_{w,i} = \sum_{j=2}^{\ell_w^i} \frac{\mathcal{L}_{(R_w^{i,j},R_w^{i,j+1})}[\prod_{\ell \in L(sub(R_w^i,1,j))}(1-\mathcal{L}_\ell)]}{P_{w,i}^f} \times [\sum_{a \in Sub(R_w^i,1,j)} \bar{T}_a + \sum_{\ell \in L(Sub(R_w^i,1,j))} D_\ell] \quad (12)$$

Finally, the average call set up delay for O-D pair w, $\bar{T}_w^{setup}$, is given by

$$\bar{T}_w^{setup} = \sum_{i=1}^{k_w}[\sum_{j=1}^{i-1} \bar{T}_{w,j} + \sum_{a \in Sub(R_w^i,1,\ell_w^i)} \bar{T}_a + \sum_{\ell \in R_w^i} D_\ell] \times \frac{(1-P_{w,i}^f)\prod_{j=1}^{i-1} P_{w,j}^f}{1-P_w^{rej}}. \quad (13)$$

5 Numerical Study on NSFNET

In this section we report the results of a study of the previously described policies on the NSFNET T3 backbone network. The NSFNET T3 backbone network as of July, 1992 is shown in Figure 4. We assume there are two T3 links, one for each direction, between each directly connected Core Nodal Switching Subsystem (CNSS) pair [2]. Thus, the network topology we study consists of 12 nodes, 30 T3-links and 132 source/destination pairs. Each node is assumed to have the same processing capacity. We assume the exogenous call arrival rate for each source/destination pair is proportional to the traffic load at each node reported in NSFNET [22]. The reader is referred to [23] for the detailed traffic model setup.

In the following sections, we first validate our analytical models by simulation results. We then compare the performance of the five routing schemes discussed in previous sections and study the effects of call processing delay and the admission control function on the call setup delay.

5.1 Analytical Results versus Simulation Results

The performance results comparing analysis and simulation are based on the parameter values below. In subsequent sections, we will vary these values and examine their effect.

[2]There is only one T3 link between each directly connected CNSS pair and the traffic of each direction is transmitted at 22.5Mbps.

- The propagation speed is assumed to be 125 miles per millisecond, i.e., two third of the speed of light.
- The mean call holding time, $\bar{\tau}$, is set to 10 seconds.
- The mean service time for a call request is set to 2 milliseconds.
- The maximum number of connections a link can accommodate at a time, C, is set to 1406. (With T3 links, this is equivalent to assuming that a connection requires on average 32 KBits per second.)
- The admission control function is set to $B(x) = (\frac{x}{C})^2$.

Figures 5(a) and (c) compare the analytic results with the simulation results for the average call setup delay between Hartford and San Francisco. Figures 5(b) and (d) compare the analytic results with the simulation results for overall network blocking probability. Each simulation point is observed over 10 independent runs. The duration of each run is 3000 units of call holding time. For each run, the initial 10% is discarded. The vertical lines about each point indicate the 95 percent confidence interval. We note that the analytic and simulation results agree very closely under various traffic loads for all routing mechanisms.

5.2 Comparison of Different Routing Algorithms

The trade-offs between call setup delays and call blocking probabilities for different routing algorithms are given in Figure 6. Consider the three sequential routing algorithms. From Figure 6, we observe that the trade-off between the call blocking probability and the call setup delay. For a given exogenous call arrival rate, the crankback algorithm yields the lowest blocking probability but the highest average call setup delay. On the other hand, the SOC algorithm yields the highest blocking probability but lowest average call setup delay. However, for a given network throughput of accepted calls, as we will observe in Figure 7, the SOC algorithm always yields the smallest average call setup delay and highest maximum achievable throughput. (Note that the maximum achievable throughput is the asymptotic point at which the average call setup time goes to infinity. When the average call setup time approaches infinity, the connections that are in the process of being setup will hold the resources that have already being reserved for an infinite time and thus the blocking probability will approach unity.) This observation is slightly different from what we observed in an earlier study [5]. In [5], we studied a 8-node hypercube network and observed that the SOC algorithm yielded a slightly lower maximum achievable throughput than the crankback algorithm. The main reason for the difference is that when the network topology is very sparse (as in the NSFnet case), the trade-off between the call blocking probability and the call setup delay becomes significant. In order to reduce the call blocking probability, we would like to have more alternate paths in the routing trees. But because the network topology is very sparse, increasing the number of alternate paths will unavoidably include paths with longer propagation delays and more intermediate nodes which, consequently increases

the average call setup delay. The SOC algorithm thus performs better in this network because calls are offered more chances to be routed through "short" paths than under the other two routing algorithms.

In comparing the results of the parallel (flooding) algorithms to the sequential algorithms, we observe that for a given exogenous call arrival rate, the sequential OOC algorithm and the parallel OOC algorithm yield almost the same blocking probability. The parallel SOC/crankback algorithm yields a slightly higher blocking probability than the sequential crankback algorithm for a given exogenous call arrival rate because it reserves more resources during call setup. Since the parallel algorithms require much more processing, the processing elements quickly saturate and, thus, yield much lower achievable maximum throughputs. For the given parameter values, we also observe that parallel schemes yield lower average call setup delays only for small exogenous arrival rates. This is because at small exogenous arrival rates, processing delays are relative small at each node, thus the parallel algorithms can take the advantage of attempting call setups in parallel over all possible paths.

5.3 The Effects of Call Processing (Service) Time

The effect that the mean control packet processing (service) time has on the call setup delay and accepted call throughput is shown in Figure 7. By comparing Figure 7(a) and (b), we observe that increasing the processing requirements for a call (i.e., from 1 millisecond to 3 milliseconds) affects the performance of these five routing algorithms significantly. Parallel routing algorithms generate more call request messages than sequential algorithms. Therefore when processing capacity is very limited, parallel routing algorithms will saturate the processing elements very quickly. Thus, sequential algorithms can offer much higher throughputs. Certainly, at very low traffic loads, we always expect to see that parallel algorithms can yield lower mean call set up delays than sequential algorithms. This is also confirmed by the analytic results. On the other hand, when the processing capacity is abundant and the communication bandwidth is the bottleneck, we observe that parallel algorithms yield not only lower mean call setup delays but also higher throughputs. The reason why sequential OOC and crankback algorithms yield lower maximum achievable throughputs is due to overflow call requests from previously attempted paths.

5.4 The Effects of the Admission Control Function

The preceding results have assumed $B(x) = (\frac{x}{C})^2$. In Figure 7 we study the effects of different forms of admission control by varying $B(x)$. Three forms of the admission control function are studied: a convex function ($B(x) = (\frac{x}{C})^2$), a linear function ($B(x) = \frac{x}{C}$), and a concave function ($B(x) = \sqrt{\frac{x}{C}}$).

By comparing the graphs in Figure 7, we observe the following interesting behavior. First, the average call setup time is very sensitive to the processing time requirements when the admission control function is convex. In other words, the processing capacity can easily become the performance bottleneck if the admission control function is a convex function. As a consequence, the parallel algorithms, which require much more processing capacity, perform poorly in Figure 7(b) because processing elements are the bottleneck of the network. On the other hand, in Figure 7(d), the processing capacity is abundant and the parallel algorithms yield both lower call setup delay and higher achievable maximum throughput.

Second, with a convex admission control function, all routing algorithms yield higher throughputs than with a concave or linear admission control function. This is because, for a given number of connections on a link, a concave admission control function and a linear admission control function block more calls than a convex admission control function does.

Third, the performance of the crankback routing algorithm becomes significantly worse than the other two algorithms when the admission control function changes from convex to concave. Again, this is because there are more alternate paths in the crankback algorithm than in the OOC algorithm. Thus, a call is given more chances to try longer paths in the crankback algorithm. As a result, the call blocking probability under the crankback routing algorithm also becomes significantly lower than under the other two algorithms.

6 Discussion

6.1 Modeling Resources Held by Blocked Calls

One of the differences between our work and previous works in the literature is that we explicitly model the resources held by blocked calls (the reader is referred to Figure 1). Whether it is important to model resources held by blocked calls depends on the ratio between the sum of call processing (service) time and propagation delay and the call holding time. When this ratio is larger than 0.01, we observe that ignoring the resources held by blocked calls introduces noticeable error on network performance measures such as blocking probability and call setup delay.

If the ratio of the sum of the call processing time and propagation delay and the mean call holding time is very small, we can ignore the modeling of resources held by blocked calls in our analysis. In this case, the computation of $\gamma^f_{(x,y),R^i_w}$, $\delta^f_{(x,y),R^i_w}$, and the second part of $\delta^s_{(x,y),R^i_w}$, which corresponds to the downstream processing delay and propagation delay, can be ignored in our analysis. Furthermore, since the call blocking model in Figure 1 is now independent of the call processing delay, we can also ignore the computation of node-offered traffic and nodal call processing delay in the iterative algorithm (Algorithm A).

After we obtain the link blocking probabilities from Algorithm A, we then can compute the processing delay at each node, average call setup delay and call blocking probability for each O-D pair in the same manner as in section 4.

6.2 The Effect of Propagation Delay

The effect of propagation delay on call setup delay and accepted call throughput has been studied in [5]. The influence of propagation delay is twofold. First, given the processing delays at the nodes of a path, increasing the propagation delay of this path will also increase the mean call setup delay by the same amount. This has a more significant effect on the sequential OOC and crankback algorithms because more time is required to crankback blocked call requests to the source node (under the sequential OOC algorithm) or upstream nodes (under the sequential crankback algorithm). Second, an increase in the propagation delay also means that the blocked calls hold their resources longer. This can have significant effect on the call blocking probability if the amount of blocked calls and their resource holding times are not negligible as compared to the amount of successful calls and normal call holding times. This influence is especially important for flooding algorithms because they generate more call requests and, consequentially, more blocked calls. However, these two influences have little effect on the mean call processing delay or the network blocking probability as long as the propagation delays are relatively small as compared to the call holding time. On the other hand, in the analysis of flooding schemes, modeling propagation delays is very important in determining the probability that a successfully received call request follows a particular path.

7 Summary

In this paper, we examined the influence of processing capacity and factors such as routing algorithms, propagation delays, and admission control functions, on the mean call setup time and call blocking probability. The influence of routing on the mean call setup time and call processing capacity was examined under five routing algorithms: three well-known algorithms in the circuit-switching literature and two controlled flooding versions of these algorithms. The influence of call admission control was modeled and examined with three different forms of "call admission control function". The NSF T3 network was used in our examples as the underlying network topology. We developed analytical models for evaluating the average call setup time and call blocking probability for these five algorithms and validated our analysis by simulations. The results of our study indicate that for a given exogenous call arrival rate, there is a trade-off between the mean call setup time and the call blocking probability. The crankback routing algorithm yields the lowest call blocking probability but also the highest mean call setup delay. On the other hand, the SOC routing algorithm yields the highest call blocking probability but also the lowest mean call setup delay. As expected, flooding algorithms yield smaller call setup delays only when the call processing capacity is abundant. We also find that the form of the admission control function has a significant influence on the mean call setup time and the processing loads at processing elements.

The contribution of this work is to quantitatively study these important factors on call setup time and call blocking probability. In order to examine the influence of call processing delay and propagation delay, resources (includes processing resources and transmission resources) held by blocked calls during the call setup time were explicitly modeled; this has been ignored in all past research. We observe that it is important to model resources held by blocked calls only when the mean call processing (service) time or the propagation delay is not significantly small as compared to call holding time. This might be true for some applications in high-speed networks such as file transfer, or, if we do the call setup at the burst level for bursty sources.

In high-speed networks, multicast routing becomes very important because of the need to support new applications such as multimedia, multiparty conferencing. One way of setting up a multicast connection is to set up multiple point-to-point connections, each connects the source node to a destination node. Another way to set up a multicast connection is to build a tree-shaped route which connects the source node to all destination nodes. No matter which approach is adopted, multicast routing requires much more processing capacity and yields a higher call setup delay than point-to-point routing. In our future work, we plan to extend our analytical models to study the interaction of call processing delays and multicast routing in high-speed networks.

References

[1] CCITT SG. XVIII, "Draft Recommendation I.311: B-ISDN General Network Aspects." Geneva, June 1990.

[2] CCITT SG. XVIII, "Draft Recommendation I.350."

[3] CCITT SG. XVIII, "I-Series Recommendations."

[4] H. Ahmadi, J. Chen, and R. Guérin, "Dynamic Routing and Call Control in High-Speed Integrated Networks," in *13th International Teletraffic Congress*, June 1991.

[5] R.-H. Hwang, J. F. Kurose, and D. Towsley, "The Effect of Processing Delay and QOS Requirements in High Speed Networks," *INFOCOM'92*, pp. 160–169, May 1992.

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

[7] M. Butto, G. Colombo, and A. Tonietti, "On Point to Point Losses in Communication Networks," in *Eighth International Teletraffic Congress*, 1976.

[8] W. S. Chan, "Recursive Algorithms for Computing End-to-End Blocking in a Network with Arbitrary Routing Plan," *IEEE Transactions on Communications*, vol. COM-28, pp. 153–164, February 1980.

[9] M. D. Gaudreau, "Recursive Formulas for the Calcu-

lation of Point-to-Point Congestion," *IEEE Transactions on Communications*, vol. COM-28, pp. 313–316, March 1980.

[10] A. Girard and Y. Ouimet, "End-to-End Blocking for Circuit-Switched Networks: Polynomial Algorithms for Some Special Cases," *IEEE Transactions on Communications*, vol. COM-31, pp. 1269–1273, December 1983.

[11] P. M. Lin, B. J. Leon, and C. R. Stewart, "Analysis of Circuit-Switched Networks Employing Originating-Office Control with Spill-Forward," *IEEE Transactions on Communications*, vol. COM-26, pp. 754–765, June 1978.

[12] J. R. Yee, *Distributed Routing and Flow Control Algorithms for Communications Networks*. PhD thesis, Department of Electrical Engineering and Computer Sicence, MIT, 1985.

[13] W. Whitt, "Blocking When Service Is Required From Several Facilities Simultaneously," *AT&T Technical Journal*, vol. vol. 64, pp. 1807–1856, October, 1985.

[14] M. Gerla and L. Fratta, "Design and Control in Processor Limited Packet Networks," in *Twelfth International Teletraffic Congress*, 1989.

[15] I. Cidon and I. Gopal, "PARIS: An Approach to Integrated High-speed Private Networks," *International Journal of Digital & Analog Cabled Systems*, pp. 77–86, April-June 1988.

[16] I. Cidon, I. Gopal, and A. Segall, "Fast Connection Establishment in High Speed Networks," *ACM SIGCOMM*, pp. 287–296, September 1990.

[17] R.-H. Hwang, J. F. Kurose, and D. Towsley, "The Effect of Processing Delay and QOS requirements in High Speed Networks," Technical Report 91-52, Dept. of Computer Science, University of Massachusetts at Amherst, 1991.

[18] R.-H. Hwang, J. F. Kurose, and D. Towsley, "State Dependent Routing for Multirate Loss Networks," *GLOBECOM'92*, pp. 565–570, December 1992.

[19] S.-P. Chung and K. W. Ross, "Reduced Load Approximations for Multirate Loss Networks," *IEEE Transactions on Communications*, vol. 8, pp. 1222–1231, Vol Com-41.

[20] L. Kleinrock, *Queueing Systems*. New York: Wiley-Interscience, 1975.

[21] D. Gross and C. M. Harris, *Fundamentals of Queueing Theory*, ch. 5.2.2, p. 295. John Wiley & Sons, Inc., 1985.

[22] Merit Network Information Center Services, *Statistical Reports Pertaining to the NSFNET Backbone Networks*. accessible via anonynmous FTP NIS.NSF.NET.

[23] R.-H. Hwang, "Routing in High-Speed Networks," Technical Report 93-43, Dept. of Computer Science, University of Massachusetts at Amherst, 1993.