The Effect of Processing Delay and QOS Requirements in High Speed Networks [1]

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

Department of Computer Science
University of Massachusetts
Amherst, Massachusetts 01003

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 processing delays at these elements are influenced by network parameters such as routing algorithms, propagation delays, admission control functions (due to QOS requirements), path lengths (or network topology), and processing capacities at these elements. The goal of this paper is to characterize the influence of these network characteristics on the call setup time and accepted call throughput. This influence is examined with three sequential routing schemes, two flooding routing schemes and one fast connection establishment protocol proposed in PARIS network under various network parameters and different forms of admission control. Analytic models for different routing algorithms are developed and are validated by simulation results. The results of our study indicate that call processing delay associated with admission control function affect the network performance significantly while propagation delay does not affect the performance significantly.

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

1 Introduction

Advances in fiber optic communication links over the last decade have shifted the network performance bottleneck from the communication links to the processing elements. In the case of future broadband ISDN (BISDN) networks, further 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 (due to the need to insure that accepted calls will be provided with a guaranteed quality of service (QOS) [1, 2]). In this paper, we thus focus on the *computational* delays associated with call admission, routing and call setup in BISDN networks.

Routing in future BISDN networks will have elements of both traditional packet-switched and circuit-switched networks. The CCITT specifications on ATM [3] specify a cell-based, packet-like transport mode for information 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 meeting the QOS guarantees already made to existing calls. The manner in which calls are offered to the various routes (e.g., the order in which routes are attempted, 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 the network processing elements can support. 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.

We investigate six routing algorithms in this study : three well-known algorithms in the circuit-switching literature, two controlled-flooding versions of these algorithms and one fast connection establishment protocol proposed in [5]. Analytical models are developed to study these six algorithms. The analysis is based on the link-decomposition method [6], a commonly-used technique in evaluating network performance in the circuit-switching literature. The analytical models are validated by simulations. The results of our study indicate that SOC and Crankback routing schemes always perform better than OOC routing scheme. Two controlled-flooding routing schemes perform better than sequential schemes only when the processing delay is relatively small as compared to the call holding time. The fast connection establishment protocol performs best when the link-level blocking probabilities are low and the processing delays are not very small. Moreover, we find that propagation delay is not an important performance factor as long as the round trip propagation delay is relatively small as compared to the call holding time. The processing delay and the call admission control function, however, play an important role on the network performance.

In future BISDN networks, call blocking becomes an important network performance issue because a call request may be rejected as a result of admission control. Call admission itself is a very complicated problem and beyond the scope of this paper. In this paper, we simply assume that admission control is some given function of the number of existing calls. In circuit-switched networks, call blocking probability is the most important and common network performance measure. 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, therefore, ignore the call set up delay. A common assumption made by these works is that calls are either setup in zero time (if there are sufficient bandwidth) or cleared in zero time (if blocked on some link). The focus of this paper is very different from these works since 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.

Another related work on the performance of a flooding-based routing scheme is [15]. [15] presents an analysis for the performance of the MKS circuit-switching communication system designed by PKI. The call setup procedure in this system makes use of a flooding scheme to find a free path between any two subscribers (nodes). A critical quantity that must be computed in that work is the probability that the *first* call setup request message received at the destination node has followed a given path (equation 14 in that paper). The same computation is also needed in our analysis of the two controlled-flooding routing algorithms. [15] discusses how different assumptions can be made to make the computation tractable. To keep the computation tractable, we assume that the total waiting times along different paths are independent (assumption 1a in [15]) and that the total waiting time along a path is the sum of threshold exponentially distributed random variables (assumption 2c in [15]).

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 gives the analytical models for different routing mechanisms. Analytical results and simulation results are presented in section 5. Finally, section 6 summarizes this paper.

2 Network Models

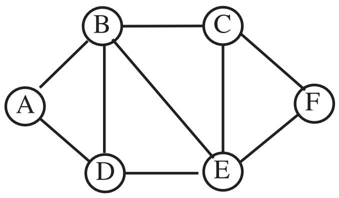
We consider a connection-oriented high-speed network consisting of N nodes, $N \geq 2$, with each node having some number of incoming and outgoing links. We adopt the network node structure described in [16, 5], in which a node consists of two components: the switching subsystem (SS) and the network 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.

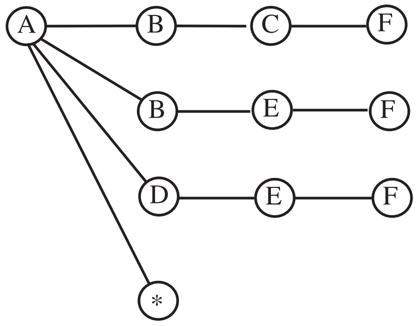
3 Specification of Routing Algorithms

As described in section 2, the call routing algorithm residing in the NCU defines 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 will be tried in some order defined by the routing rule to route a new call set-up. Three routing rules, which use knowledge of the network topology but no explicit network status information such as current link delay, traffic load, ..., etc, are studied; these rules are well-known in the circuit-switching literature. Two controlled flooding versions based on these three rules and a fast connection establishment protocol proposed in [5] based on the OOC control rule will also be investigated. We identify the original three rules as sequential routing rules, the controlled flooding rules as parallel-sequential rules (because all paths are tried in parallel but intermediate nodes on a chosen path are checked sequentially) and the protocol introduced in [5] as the sequential-parallel rule (because paths are tried sequentially but intermediate nodes are verified in parallel).

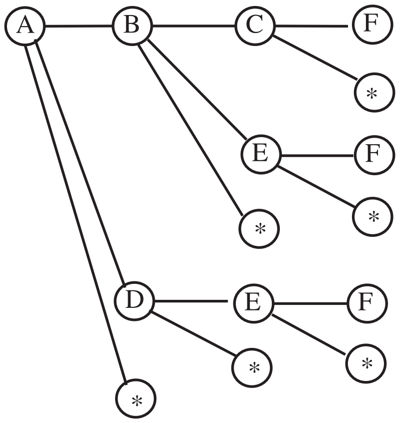
We first consider the three routing rules originating from circuit-switched networks. We describe these rules by the help of *augmented route tree* [11] as shown in Figure 1. A call is blocked if a so-called *loss node*, a node labeled by an asterisk, is reached. Paths are tried sequentially, from top to bottom, left to right, with the following control rules.



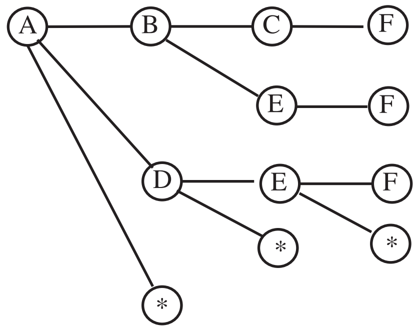
(a) An example network.



(b) The augmented route tree for OOC control.



(c) The augmented route tree for SOC control.



(d) The augmented route tree for crankback control.

Figure 1: Augmented Route Tree for OOC, SOC and Crankback Control Rules

1. **OOC control rule:** According to this rule, the choice of the next path to try is always made by source node. When a call request is rejected on a link on the chosen path, a

blocking message is returned along the path to the source node. Bandwidth reserved previously is released and the source node sends another call request on the path to be tried next. For example, as shown in Figure 1(b), path $A - B - C - F$ is tried first. If it is blocked, path $A - B - E - F$ is then tried. If it is also blocked, path $A - D - E - F$ then is tried. The call is blocked if a blocked message is received on the last path, i.e. $A - D - E - F$. Note that this rule does not use any information about *which* link resulted in the call being blocked on a path.

2. **SOC control rule:** The SOC control rule is also called progressive control, sequential control, or spill-forward without crankback. This rule tries to improve the OOC control rule 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 returning this 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 is able to provide the required QOS, the call is blocked. For example, as shown in Figure 1(c), if the call request is accepted on link $A - B$ then node B is responsible for selecting the next link. A block on link $B - C$ results in another trial on link $B - E$. The call request is blocked at node B if link $B - E$ rejects the request.

3. **Crankback control rule:** Crankback control is also called spill-forward with crankback. It improves the SOC rule by allowing a blocked message to be sent back to upstream nodes in the routing tree. Recall the situation described in SOC rule; in the case of Crankback, when a call request is blocked on link $B - E$, a blocking message is sent back to node A, the upstream node of node B. The call request is then tried on link $A - D$. The call is blocked only when none of the paths defined in the routing tree is available. In the example shown in Figure 1(d), the call is blocked if link $A - D$, $D - E$, or $E - F$ is blocked.

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

In [5], the authors propose a protocol for fast connection establishment for high speed packet-switched networks. The idea is to speed up the call set up delay by allowing the processing at intermediate nodes be done in parallel. In this paper, we study an extreme case of the protocol which yields the maximum speedup. The protocol works very similar to the OOC rule. The only difference is that when the SS of an intermediate node receives a call setup message, it relays the message to the next node on the path immediately without waiting for the result of the admission control from the NCU. The NCU of each immediate node then sends its result to the destination node directly. In this way, the NUC's of all intermediate nodes will be able to do the admission control process simultaneously. As in [5], we assume each intermediate node can send message directly to any remote nodes. We further assume that the result sent from NCU's of intermediate nodes to the destination node is error free.

4 Analytical Models

In studying the influence of network characteristics such as routing algorithms, the network performance measures in which we are interested are 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 models for homogeneous networks. With the homogeneity assumption, the network topology is symmetric, every node is treated independently, and behaves in a statistically identical manner. The same is true for the link as well. Appendix B discusses how this methodology can be extended to heterogeneous networks. In our analytical models, we make the following assumptions and notations.

1. External call arrivals at each node are assumed to be governed by a stationary Poisson process with parameter λ.
2. Call set-up requests (either originating at the node or being received from "upstream" nodes) to a node are also assumed to arrive according to a Poisson process.
3. The call holding time, denoted by τ, for each call is assumed to be arbitrarily distributed with mean $\bar{\tau}$.
4. Call request processing (service) times, denoted by T, are assumed to be exponentially distributed with parameter $1/\bar{T}$.
5. Propagation delays, denoted by D_p, are assumed to be constants.
6. 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).
7. Calls are blocked independently at all links [7, 8, 13].
8. The release of resources, either due to call termination or call abortion, consumes zero processing time.

9. The network topology and routing trees are fixed.

10. Each node has M statistically identical incoming, as well as outgoing, links.

11. Let $\mathcal{L}$ be the steady state blocking probability on each link. Note that $\mathcal{L}$ is identical for each link due to the homogeneity assumption.

12. Let P_{rej} be the end-to-end call blocking probability.

13. Let $\bar{T}_{setup}$ be the average end-to-end call set up delay.

Resources on the 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 the 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. 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. Throughout the remainder of this chapter, we will assume that all links are alike and drop the dependence on the link identity. In our numerical examples, we examine the call setup delays and accepted call throughput associated with various routing algorithms and network parameters when $B(x)$ is concave, convex, or linear.

Our analytic models are based on the link-decomposition method. Similar to the original link-decomposition method, we first decompose the overall network problem into a set of independent link and node problems. After the performance of each link and node is computed, the overall network performance such as end-to-end call setup delay and call blocking probability can be recovered from the individual link- and node-performance measures such as nodal processing delay and link blocking probability.

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. That is, to compute the link blocking probability and the mean nodal processing delay we need to solve a set of nonlinear equations that have the following forms:

$$\begin{aligned} \mathcal{L} &= F_1(\mathcal{L}, \bar{T}) \\ \bar{T} &= F_2(\mathcal{L}, \bar{T}) \end{aligned}$$

Like most sets of nonlinear equations, solutions to this set of nonlinear equations can be obtained only by numerical methods. The numerical method used most frequently 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 the 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 different routing policies.

4.1 Computing Nodal Processing Delay

As discussed earlier, when a call setup request arrives 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 this computation (both waiting for processing by the NCU as well as NCU processing itself) as the *nodal processing delay*, denoted by T. 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 (queuing delay plus service time), $\bar{T}$, is given by [17],

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

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 chapter.)

Besides the processing delay, the 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 a deterministic service time.

4.2 Computing Link Blocking Probability

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 the 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.

Let us consider a single link in isolation, as shown in Figure 2. (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 i calls as γ_i and define $\Gamma = \sum_{i=1}^{n} \gamma_i$. 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_i^s. The second type of calls, which has an arrival rate of γ_i^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) downstream. Let δ_i^s and δ_i^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 2 models a call that has successfully reserved resources at this link (these resources may be held for the duration of a call or may be released shortly if a

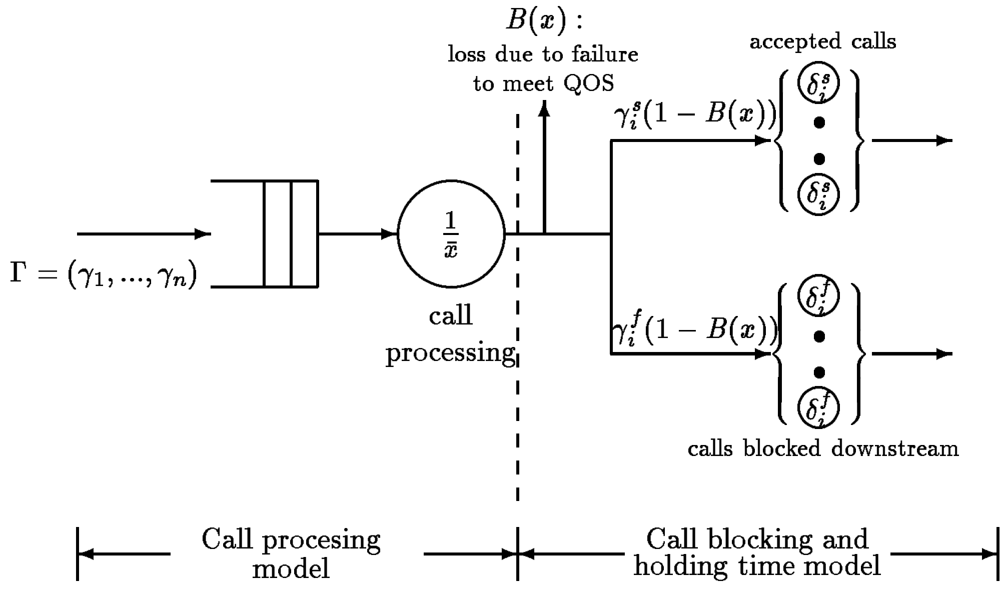


Figure 2: Queueing model for processing delay and call holding times

call is blocked downstream). Since the link capacity is limited, the maximum number of such busy servers in Figure 2 is limited and is denoted by C.

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 2 can be shown to be equivalent to the $M/G/C/C$ queue shown in Figure 3. The mean service time, $1/\mu$, for this $M/G/C/C$ queue is given by

$$\frac{1}{\mu} = \sum_{i=1}^{n} \frac{\gamma_i^s}{\Gamma} \cdot \delta_i^s + \sum_{i=1}^{n} \frac{\gamma_i^f}{\Gamma} \cdot \delta_i^f .$$

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

$$\tilde{\lambda}(i) = \Gamma \cdot (1 - B(i)) \quad i = 0, ..., C-1.$$

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 [18]

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

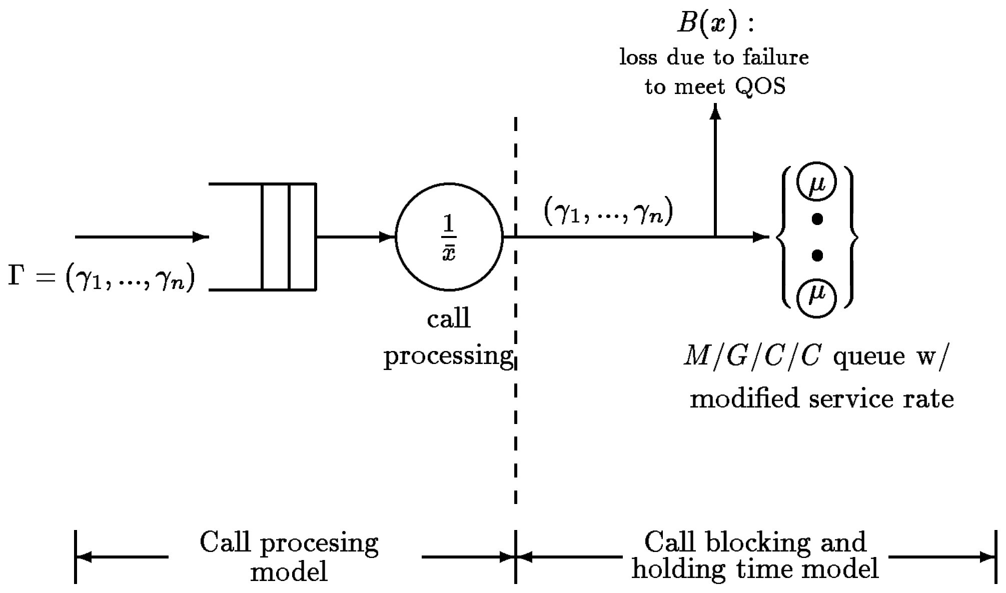


Figure 3: Equivalent queueing model for processing delay and call holding times

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

$$\mathcal{L} = \sum_{i=0}^{C} \pi_i \cdot B(i) \quad i = 0, ..., C.$$

4.3 Offered Call Requests, Call Setup Delay, and Call Blocking Probability in Homogeneous Networks

The computation of the link- and node-offered call arrival rate is complicated and depends heavily on the routing tree used. For ease of explanation, we show how the link- and node-offered call arrival rates are computed in a homogeneous network.

4.3.1 Sequential Routing Rules

In the following three sections (Section 4.3.1.1, 4.3.1.2, and 4.3.1.3) we show how to compute the node- and link-offered call arrival rate, call blocking probability, and average call setup delay for the three sequential routing rules.

4.3.1.1 OOC Routing Rule

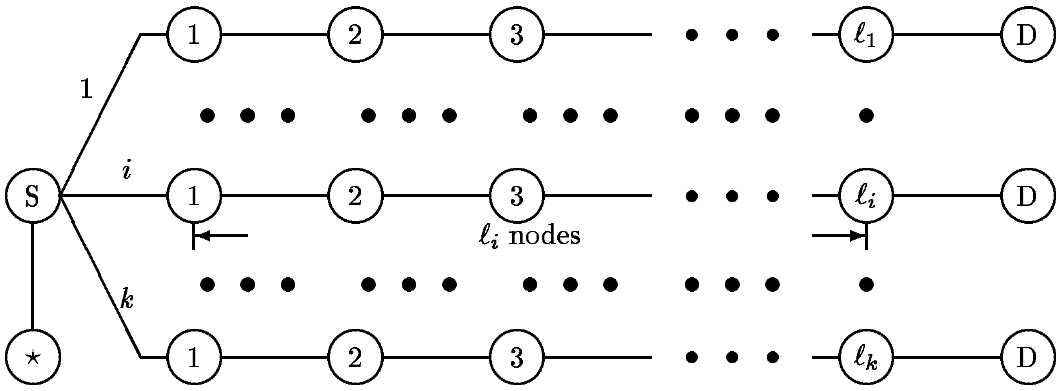


Figure 4: Augmented Route Tree for OOC Control Rules (homogeneous case).

As shown in Figure 4, we assume that there are k possible paths between each source-destination pair, each with $\ell_i, i = 1, ..., k$, intermediate nodes. Let us begin by calculating certain quantities that will be required in the computation of the link- and node-offered call request rate, end-to-end call blocking probability, and mean end-to-end call setup delay.

First we define P^i_{fail} as the probability that a call fails on path i given that an attempt is made to set the call up on that path. Under the independence assumption, we have

$$\begin{aligned} P^i_{fail} &= \sum_{j=0}^{\ell_i} \mathcal{L}(1-\mathcal{L})^j, \\ &= 1-(1-\mathcal{L})^{\ell_i+1}. \end{aligned} \tag{1}$$

Define $\bar{\mathcal{N}}_i$ as the expected number of nodes visited along path i by a call setup message given that it fails on the selected path. $\bar{\mathcal{N}}_i$ is given by

$$\begin{aligned} \bar{\mathcal{N}}_i &= \sum_{j=1}^{\ell_i} (j+1)\frac{(1-\mathcal{L})^j \mathcal{L}}{P^i_{fail}}, \\ &= \frac{1-[(\ell_i+1)\mathcal{L}+1](1-\mathcal{L})^{\ell_i+1}}{\mathcal{L}P^i_{fail}} - \frac{\mathcal{L}}{P^i_{fail}}. \end{aligned} \tag{2}$$

Let $\bar{N}_{i,j}$ denote the expected number of nodes a call set up message visits after the jth node on path i given that it has successfully reserved resources at the jth node and fails at some node farther down path i. $\bar{N}_{i,j}$ is given by

$$\bar{N}_{i,j} = \sum_{n=1}^{\ell_i-j} n\frac{(1-\mathcal{L})^{n-1}\mathcal{L}}{\sum_{m=1}^{\ell_i-j}(1-\mathcal{L})^{m-1}\mathcal{L}},$$

$$= \frac{1 - [(\ell_i - j)\mathcal{L} + 1](1-\mathcal{L})^{\ell_i - j}}{\mathcal{L}[1-(1-\mathcal{L})^{\ell_i - j}]}. \tag{3}$$

Link-offered call requests

Let us now focus on a single *link* in the network. As shown in Figure 2, the incoming traffic to a link is divided into several classes. Under OOC control, the incoming traffic to a link is divided into $\sum_{i=1}^{k}(\ell_i + 1)$ classes which will be referred to as (i,j), $i = 1, ..., k$, $j = 0, ..., \ell_i$, representing the jth link on the ith path between all source/destination pairs. The arrival rate of the (i,j)th class of traffic, which is referred to as $\gamma_{i,j}$, is the total call setup traffic offered to a link by call requests which reach this link as the jth link of path i for all source/destination pairs in the network. $\gamma_{i,0}$ can be viewed as the rate of exogenous call requests entering the network at a node incident to this link, and being offered to this link as the first link on the ith path from this source node. Because of the homogeneity assumption and the fact that each node has M statistically identical outgoing links, we have the following equations for all of the links.

$$\gamma_{1,0} = \frac{\lambda}{M}, \tag{4}$$

$$\gamma_{i,0} = \frac{\lambda}{M}\prod_{j=1}^{i-1} P_{fail}^{j}, \quad i = 2, ..., k, \tag{5}$$

$$\gamma_{i,j} = \gamma_{i,j-1}(1-\mathcal{L}), \quad j = 1, ..., \ell_i, \;\; i = 1, ..., k. \tag{6}$$

Recall that each class of offered call requests to link (i,j) 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, $\gamma_{i,j}^{s}$ and $\gamma_{i,j}^{f}$ respectively, and the mean call holding time, $\delta_{i,j}^{s}$ and $\delta_{i,j}^{f}$, are computed as follows,

$$\gamma_{i,j}^{s} = \gamma_{i,j}(1-\mathcal{L})^{\ell_i - j}, \quad j = 0, ..., \ell_i, \;\; i = 1, ..., k, \tag{7}$$

$$\begin{aligned} \gamma_{i,j}^{f} &= \gamma_{i,j} - \gamma_{i,j}^{s}, \\ &= \gamma_{i,j}(1-(1-\mathcal{L})^{\ell_i - j}), \quad j = 0, ..., \ell_i, \;\; i = 1, ..., k, \end{aligned} \tag{8}$$

$$\delta_{i,j}^{s} = (\ell_i - j) * (\bar{T} + D_p) + D_p + \bar{\tau}, \tag{9}$$

$$\delta_{i,j}^{f} = \bar{N}_{i,j}(\bar{T} + D_p). \tag{10}$$

Recall that $\bar{T}$ is the mean processing delay (queueing plus service) at each node (which is still unknown at this point, since the overall call arrival rates are unknown), D_p is the propagation delay and $\bar{\tau}$ 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.

Node-offered call requests

Let us now focus on a single *node* in isolation. First, we 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 which reach this node as the source node is referred to as G_0, and is given by:

$$G_0 \quad = \quad \lambda + M \sum_{i=1}^{k-1} \gamma_{i,0}(P^i_{fail} - \mathcal{L}). \tag{11}$$

Note that G_0 includes *"first time"* exogenous call requests as well as call setup attempts which have been previously blocked. The node-offered call request rate, G, is then given by

$$G \quad = \quad G_0 + M \sum_{i=1}^{k} \sum_{j=1}^{\ell_i} \gamma_{i,j}. \tag{12}$$

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 is rejected is given by

$$P_{rej} \quad = \quad \prod_{i=1}^{k} P^i_{fail}. \tag{13}$$

The average call set-up delay, $\bar{T}_{setup}$, is then computed as follows:

$$\bar{T}_{setup} \quad = \quad \sum_{i=1}^{k} \left[\sum_{j=1}^{i-1} \bar{\mathcal{N}}_j + \ell_i + 1 \right] [\bar{T} + D_p] \frac{(1 - P^i_{fail}) \prod_{j=1}^{i-1} P^j_{fail}}{1 - P_{rej}}. \tag{14}$$

The term $\sum[\bar{\mathcal{N}}_j + \ell_i + 1]$ in equation (14) is the expected number of nodes a call setup message visits given that the call is successfully set up on the ith path. $\bar{T} + D_p$ is the expected processing and propagation delay. The final (fractional) term is the probability that the call is successfully set up on path i given that the call is not blocked.

4.3.1.2 SOC Routing Rule

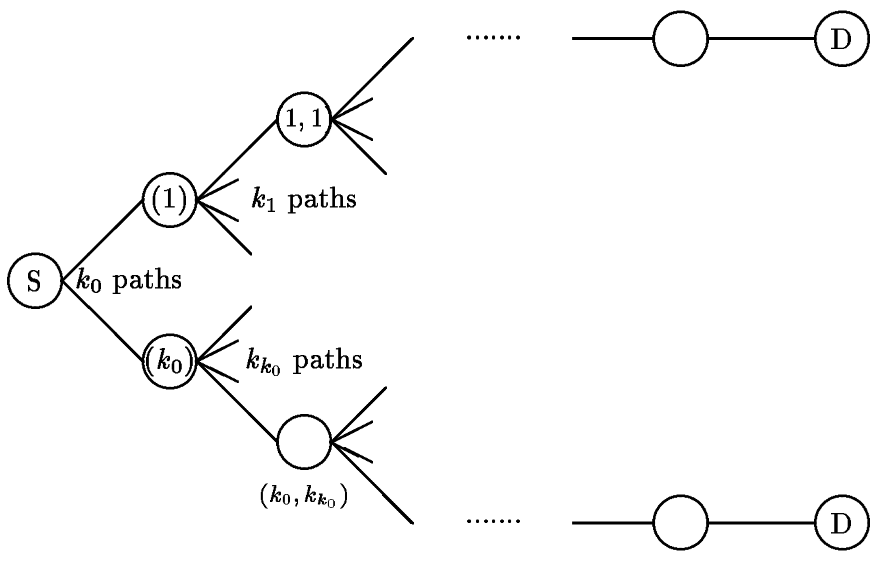


Figure 5: Routing tree for SOC and Crankback control rules (homogeneous case)

The routing tree shown in Figure 5 is shared by the SOC routing rule and the crankback routing rule. The *loss nodes* in the tree are omitted. Each node in the routing tree is given a unique label according to the following rules. The node with label $(i_1, ..., i_{\ell-1}, i_\ell)$ is the i_ℓ-th child of node $(i_1, ..., i_{\ell-1})$ and has $k_{i_1,...,i_\ell}$ children labeled $(i_1, ..., i_\ell, 1), ..., (i_1, ..., i_\ell, k_{i_1,...,i_\ell})$. The source node is labeled (0) while its children are labeled $(1), (2), ..., (k_0)$ respectively. (Note that due to the homogeneity assumption, all O-D pairs have routing trees of the same form and each node will be node $(i_1, ..., i_\ell)$ in the routing tree of some O-D pair, $\forall(i_1, ..., i_\ell)$.) The set of all nodes that are directly connected to the destination node by a link is denoted by $\mathcal{D}$. A node $n \in \mathcal{D}$ always tries the direct link first.

The following notation, which is a straightforward generalization of the notation used for OOC, is used in the analyses of both SOC and Crankback.

- $\bar{P}_{i_1,...,i_\ell}$: the probability that a call request is blocked at node $(i_1, ..., i_\ell)$ or later given that it has reached node $(i_1, ..., i_\ell)$.
- $\bar{N}_{i_1,...,i_\ell}$: the expected number of nodes visited at and after node $(i_1, ..., i_\ell)$ given that the call request is blocked at node $(i_1, ..., i_\ell)$ or later.
- $\bar{\mathcal{N}}_{i_1,...,i_\ell}$: the expected number of nodes visited at and after node $(i_1, ..., i_\ell)$ given that the call request is successfully set up through node $(i_1, ..., i_\ell)$ and subsequent nodes.
- $G_{i_1,...,i_\ell}$: the offered call request rate to the *node* under consideration by call requests which reach this node as the $(i_1, ..., i_\ell)$ node in the routing tree.

- $\gamma_{(i_1,...,i_\ell),j}$: the offered call request rate to the *link* under consideration by call requests which reach this link as the jth outgoing link of node $(i_1,...,i_\ell)$ in the routing tree.
- $\gamma^s_{(i_1,...,i_\ell),j}$: the part of $\gamma_{(i_1,...,i_\ell),j}$ which will eventually be successfully set up. Recall that the mean resource holding time for successfully setup calls is $\delta^s_{(i_1,...,i_\ell),j}$.
- $\gamma^f_{(i_1,...,i_\ell),j}$: the part of $\gamma_{(i_1,...,i_\ell),j}$ which will be later blocked at some node. As before, the mean resource holding time for calls blocked downstream is $\delta^f_{(i_1,...,i_\ell),j}$.

Let us now 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. The quantities $\bar{P}_{i_1,...,i_\ell}$, $\bar{N}_{i_1,...,i_\ell}$ and $\bar{\mathcal{N}}_{i_1,...,i_\ell}$ satisfy the following recursions. Note that the computation can be done by scanning the routing tree only once. We first compute the probability that a call request is blocked at node $(i_1,...,i_\ell)$ or later.

$$\bar{P}_{i_1,...,i_\ell} \quad = \quad [\sum_{j=1}^{k_{i_1,...,i_\ell}} \mathcal{L}^{j-1}(1-\mathcal{L})\bar{P}_{i_1,...,i_\ell,j}] + \mathcal{L}^{k_{i_1,...,i_\ell}} \qquad (15)$$

with $\bar{P}_{i_1,...,i_\ell,1} = 0, \ \forall(i_1,...,i_\ell) \in \mathcal{D}$. (Recall that $(i_1,...,i_\ell,1)$ is the destination node.) The term $\mathcal{L}^{j-1}$ in equation (15) is the probability that a call setup message is blocked at this node on the first $j-1$ outgoing links. The term $(1-\mathcal{L})\bar{P}_{i_1,...,i_\ell,j}$ is the probability that a call setup message successfully reserves bandwidth on the jth outgoing link but is blocked downstream. The final term $\mathcal{L}^{k_{i_1,...,i_\ell}}$ is the probability that a call setup message is blocked at this node on all outgoing links (i.e., that none of the outgoing links to the destination can provide the requested QOS).

For $\bar{N}_{i_1,...,i_\ell}$, we have

$$\bar{N}_{i_1,...,i_\ell} \quad = \quad \left(\sum_{j=1}^{k_{i_1,...,i_\ell}} \frac{\mathcal{L}^{j-1}(1-\mathcal{L})\bar{P}_{i_1,...,i_\ell,j}}{\bar{P}_{i_1,...,i_\ell}} \bar{N}_{i_1,...,i_\ell,j} \right) + 1 \qquad (16)$$

with $\bar{N}_{i_1,...,i_\ell,1} = 0, \ \forall(i_1,...,i_\ell) \in \mathcal{D}$. The first (fractional) term in equation (16) is the probability that a call setup message successfully reserves bandwidth on the jth outgoing link but fails downstream given that it is eventually blocked. The term "1" accounts for the visit to the current node. Note that, as we assumed in the OOC rule, a call request blocked on the jth outgoing link is immediately tried on the $(j+1)$st link.

Similarly, the expected number of nodes visited at and after node $(i_1, ..., i_\ell)$ given that the call request is ***successfully*** set up through node $(i_1, ..., i_\ell)$ and subsequent nodes is given by

$$\bar{\mathcal{N}}_{i_1,...,i_\ell} = \left(\sum_{j=1}^{k_{i_1,...,i_\ell}} \frac{\mathcal{L}^{j-1}(1-\mathcal{L})(1-\bar{P}_{i_1,...,i_\ell,j})}{1-\bar{P}_{i_1,...,i_\ell}} \bar{\mathcal{N}}_{i_1,...,i_\ell,j} \right) + 1 \qquad (17)$$

with $\bar{\mathcal{N}}_{i_1,...,i_\ell,1} = 0, \ \forall (i_1, ..., i_\ell) \in \mathcal{D}$. The first (fractional) term in equation (17) is the probability that a call setup message is successfully set up through the jth child of node $(i_1, ..., i_\ell)$ given that it is successfully set up through node $(i_1, ..., i_\ell)$.

Link-offered call requests

The quantities $\bar{P}_{i_1,...,i_\ell}$, $\bar{N}_{i_1,...,i_\ell}$ and $\bar{\mathcal{N}}_{i_1,...,i_\ell}$ are nodal performance measures. Now let us consider link-level measures and focus on a single link in the network and compute the offered call request rate at this link. The following equations are used to compute these rates and mean resources holding times of each class of calls,

$$\begin{aligned}
\gamma_{(0),j} &= \frac{\lambda}{M}\mathcal{L}^{j-1}, \quad j = 1, ..., k_0, & (18)\\
\gamma_{(i),j} &= \gamma_{(0),i}(1-\mathcal{L}), \mathcal{L}^{j-1} \quad i = 1, ..., k_0, \ j = 1, ..., k_i, & (19)\\
\gamma_{(i_1,...,i_\ell),j} &= \gamma_{(i_1,...,i_{\ell-1}),i_\ell}(1-\mathcal{L})\mathcal{L}^{j-1}, \quad j = 1, ..., k_{i_1,...,i_\ell}, & (20)\\
\gamma^{s}_{(i_1,...,i_\ell),j} &= \gamma_{(i_1,...,i_\ell),j}(1-\bar{P}_{i_1,...,i_\ell,j}), & (21)\\
\gamma^{f}_{(i_1,...,i_\ell),j} &= \gamma_{(i_1,...,i_\ell),j}\bar{P}_{i_1,...,i_\ell,j}, & (22)\\
\delta^{s}_{(i_1,...,i_\ell),j} &= \bar{\mathcal{N}}_{i_1,...,i_\ell,j}(\bar{T}+D_p) + D_p + \bar{\tau}, & (23)\\
\delta^{f}_{(i_1,...,i_\ell),j} &= \bar{N}_{i_1,...,i_\ell,j}(\bar{T}+D_p). & (24)
\end{aligned}$$

The term $\frac{\lambda}{M}$ in equation (18) is the portion of exogenous call requests offered to the link by call requests which reach this link as the first outgoing link of the source node. (The division by M results from the homogeneity assumption and the fact that each node has M statistically identical outgoing links.) The final term, $\mathcal{L}^{j-1}$, in equation (18) is the probability that a call request is blocked on the first $j-1$ links. (Recall that $\gamma_{(0),j}$ is the traffic offered to the link by call requests which reach this link as the jth outgoing link of the source node.) Similarly, the term $\gamma_{(0),i}(1-\mathcal{L})$ in equation (19) is the offered call requests which reach this link as the first outgoing link of node (i). Therefore, the right side of equation (19) gives the rate at which call requests reach this link as the jth outgoing link of node (i). The term $\gamma_{(i_1,...,i_\ell),j}(1-\bar{P}_{i_1,...,i_\ell,j})$ in equation (21) is the portion of the calls offered to the link by call requests which reach this link as the jth link of node $(i_1, ..., i_\ell)$ and which are eventually ***successfully*** set up if not blocked on this link. (The reader is referred to Figure 2.)

Node-offered call requests

Let us now focus on a single node in isolation. In order to calculate the node-offered call request rate, let us focus on the M incoming link flows. Recall that the rate at which call requests reach this node as the $(i_0, ..., i_\ell)$ node for all source/destination pairs in the network is referred to as $G_{i_0,...,i_\ell}$. Because of the homogeneity assumption, $G_{i_0,...,i_\ell}$ can be computed as follows,

$$\begin{aligned} G_0 &= \lambda, & (25)\\ G_j &= M\ \gamma_{(0),j}(1-\mathcal{L}), \quad j = 1, ..., k_0, & (26)\\ G_{i_1,...,i_\ell} &= M\ \gamma_{(i_1,...,i_{\ell-1}),i_\ell}(1-\mathcal{L}). & (27)\end{aligned}$$

The term $\gamma_{(i_1,...,i_{\ell-1}),i_\ell}(1-\mathcal{L})$ in equation (27) is the call request rate arriving at the node by call requests which reach this node as the $(i_0, ..., i_\ell)$ node in routing tree from *one* incoming link, and is multiplied by M because each node has M statistically identical incoming links.

The node-offered call request rate, G, is then given by

$$G = \sum G_{i_1,...,i_\ell}.$$

The summation is for all nodes in the routing trees of all source/destination pairs in the network except the destination node.

Call setup delay and call blocking probability

Recalling that the source node has the label "0", the end-to-end call blocking probability is given by

$$P_{rej} = \bar{P}_0 \qquad (28)$$

and the average call set-up delay is given by

$$\bar{T}_{setup} = \bar{\mathcal{N}}_0(\bar{T} + D_p) \qquad (29)$$

4.3.1.3 Crankback Routing Rule

The notation used in the analysis of the crankback routing rule is the same as that used in the analysis of the SOC routing rule. Again, we begin by presenting expressions for the quantities $\bar{P}_{i_1,\ldots,i_\ell}$, $\bar{N}_{i_1,\ldots,i_\ell}$ and $\bar{\mathcal{N}}_{i_1,\ldots,i_\ell}$. The analysis is similar to that of SOC and hence we present the equations with little discussion,

$$\bar{P}_{i_1,\ldots,i_\ell} = \prod_{j=1}^{k_{i_1,\ldots,i_\ell}} [\mathcal{L} + (1-\mathcal{L})\bar{P}_{i_1,\ldots,i_\ell,j}] \tag{30}$$

with $\bar{P}_{i_1,\ldots,i_\ell,1} = 0,\ \forall(i_1,\ldots,i_\ell) \in \mathcal{D}$,

$$\bar{N}_{i_1,\ldots,i_\ell} = \left(\sum_{j=1}^{k_{i_1,\ldots,i_\ell}} \frac{(1-\mathcal{L})\bar{P}_{i_1,\ldots,i_\ell,j}}{\mathcal{L} + (1-\mathcal{L})\bar{P}_{i_1,\ldots,i_\ell,j}} (\bar{N}_{i_1,\ldots,i_\ell,j} + 1) \right) + \frac{\mathcal{L}}{\mathcal{L} + (1-\mathcal{L})\bar{P}_{i_1,\ldots,i_\ell,k_{i_1,\ldots,i_\ell}}} \tag{31}$$

with $\bar{N}_{i_1,\ldots,i_\ell,1} = 0,\ \forall(i_1,\ldots,i_\ell) \in \mathcal{D}$. The first term corresponds to the fact that if the call is blocked at node $(i_1,\ldots,i_\ell)$, it must be blocked on each outgoing link or at some "downstream" node of this link. Also, recall that if the call blocks on the jth outgoing link, it is immediately tried on the $(j+1)$st link,

$$\begin{aligned} \bar{\mathcal{N}}_{i_1,\ldots,i_\ell} = & \sum_{j=1}^{k_{i_1,\ldots,i_\ell}} \frac{[\prod_{m=1}^{j-1}(\mathcal{L} + (1-\mathcal{L})\bar{P}_{i_1,\ldots,i_\ell,m})](1-\mathcal{L})(1-\bar{P}_{i_1,\ldots,i_\ell,j})}{1-\bar{P}_{i_1,\ldots,i_\ell}} \\ & [(\sum_{m=1}^{j-1} \frac{(1-\mathcal{L})\bar{P}_{i_1,\ldots,i_\ell,m}}{\mathcal{L} + (1-\mathcal{L})\bar{P}_{i_1,\ldots,i_\ell,m}} (\bar{N}_{i_1,\ldots,i_\ell,m} + 1)) + \bar{\mathcal{N}}_{i_1,\ldots,i_\ell,j} + 1] \end{aligned} \tag{32}$$

with $\bar{\mathcal{N}}_{i_1,\ldots,i_\ell,1} = 0,\ \forall(i_1,\ldots,i_\ell) \in \mathcal{D}$.

<u>Link-offered call requests</u>

The link-offered call request rate is computed as follows:

$$\begin{aligned} \gamma_{(0),1} &= \frac{\lambda}{M}, \\ \gamma_{(0),j} &= \frac{\lambda}{M} \prod_{m=1}^{j-1} [\mathcal{L} + (1-\mathcal{L})\bar{P}_m], \quad j = 2,\ldots,k_0, \\ \gamma_{(i_1,\ldots,i_\ell),1} &= \gamma_{(i_1,\ldots,i_{\ell-1}),i_\ell}(1-\mathcal{L}), \end{aligned}$$

$$\begin{aligned}
\gamma_{(i_1,\ldots,i_\ell),j} &= \gamma_{(i_1,\ldots,i_\ell),1}\prod_{j=1}^{i-1}[\mathcal{L}+(1-\mathcal{L})\bar{P}_{i_1,\ldots,i_\ell,j}], \quad j=1,\ldots,k_{i_1,\ldots,i_\ell}, \\
\gamma^s_{(i_1,\ldots,i_\ell),j} &= \gamma_{(i_1,\ldots,i_\ell),j}(1-\bar{P}_{i_1,\ldots,i_\ell,j}), \\
\gamma^f_{(i_1,\ldots,i_\ell),j} &= \gamma_{(i_1,\ldots,i_\ell),j}\bar{P}_{i_1,\ldots,i_\ell,j}, \\
\delta^s_{(i_1,\ldots,i_\ell),j} &= \bar{N}_{i_1,\ldots,i_\ell,i}(\bar{T}+D_p)+D_p+\bar{\tau}, \\
\delta^f_{(i_1,\ldots,i_\ell),j} &= \bar{N}_{i_1,\ldots,i_\ell,i}(\bar{T}+D_p).
\end{aligned}$$

Node-offered call requests

The node-offered call request rate is computed as $G = \sum G_{i_1,\ldots,i_\ell}$. where the summation is for all nodes in the routing trees of all source/destination pairs in the network except the destination node and

$$G_{i_1,\ldots,i_\ell} = M\left(\gamma_{(i_1,\ldots,i_\ell),1} + \sum_{j=1}^{k_{i_1,\ldots,i_\ell}-1}\gamma_{(i_1,\ldots,i_\ell),j}(1-\mathcal{L})\bar{P}_{i_1,\ldots,i_\ell,j}\right).$$

Call setup delay and call blocking probability

Equations (28) and (29), used in SOC, also applies here.

4.3.2 Sequential-Parallel Routing Rules

In this section, we show how the link- and node-offered call arrival rate, call blocking probability, and average call setup delay can be computed for the routing scheme which uses OOC control for route selection and the protocol proposed by [5] for call establishment.

As in Section 4.3.1.1, we assume that the routing tree shown in Figure 4 is available at each source node. Let us begin by calculating certain quantities that will be needed in later computations. We first compute the probability that a call fails at path i given that a call setup is attempted on path i. The computation is the same as equation (1), i.e.,

$$P^i_{fail} = 1-(1-\mathcal{L})^{\ell_i+1} \tag{33}$$

Let $P(n)$ be the probability that the call request processed by the node under study arrives at the destination node earlier than the first blocking message from other n nodes on the same path. Since the processing times are i.i.d. exponential variables, $P(n)$ is given by,

$$P(n) = \frac{1}{n+1} \tag{34}$$

Now let $\bar{T}_{i,j}$ be the expected time for the jth node on path i to find out that a call request currently being established is rejected at some other node on the path given that this call request successfully reserves bandwidth at this node. Let us consider the situation that among the remaining $(\ell_i - 1)$ nodes, the call request fails at n of them. Let $T^{i,j}_{min(n)}$ be the time for the destination node to receive the first blocking message (from those blocked nodes) after it has received the request from node j given that the destination node received the request from node j given that the destination node received the request from node j first. Here $T^{i,j}_{min(n)}$ is the minimum of n i.i.d. exponential random variables plus a constant propagation delay. Therefore, the expection of $T^{i,j}_{min(n)}$ is given by

$$\bar{T}^{i,j}_{min(n)} = \frac{\bar{T}}{n} + (\ell_i - j + 1)D_p \tag{35}$$

where $\bar{T}$ is the mean processing delay at each node and D_p is the propagation delay on each link. Now we can compute $\bar{T}_{i,j}$ as follows,

$$\begin{aligned} \bar{T}_{i,j} &= \sum_{n=1}^{\ell_i-1} \frac{\binom{\ell_i - 1}{n}(1-\mathcal{L})^{\ell_i-1-n}\mathcal{L}^n}{1-(1-\mathcal{L})^{\ell_i-1}} \left(P(n)\bar{T}^{i,j}_{min(n)} + (1-P(n))(\ell_i - j + 1)D_p\right) \\ &= \left(\sum_{n=1}^{\ell_i-1} \frac{\binom{\ell_i - 1}{n}(1-\mathcal{L})^{\ell_i-1-n}\mathcal{L}^n}{1-(1-\mathcal{L})^{\ell_i-1}} \frac{\bar{T}}{n(n+1)}\right) + (\ell_i - j + 1)D_p \end{aligned} \tag{36}$$

The term $(1 - P(n))(\ell_i - j + 1)D_p$ accounts for the situation that the destionation node has received a blocked message from one of thos blocked nodes when it receives the request from node j.

<u>Link-offered traffic</u>

Let us now focus on a single link and compute the traffic rate offered to this link. Recall that the incoming traffic to a link is divided into $\sum_{i=0}^{k}(\ell_i + 1)$ classes which are referred to as $\gamma_{i,j}, \ i = 1, ..., k, \ j = 0, ..., \ell_i$.

$$\gamma_{1,0} = \frac{\lambda}{M} \tag{37}$$

$$\gamma_{i,0} = \frac{\lambda}{M}\prod_{j=1}^{i-1} P^j_{fail} \quad i = 2, ..., k \tag{38}$$

$$\gamma_{i,j} = \gamma_{i,0}(1-\mathcal{L}) \quad i = 1, ..., k \ j = 1, ..., \ell_i \tag{39}$$

We can see that the only difference betwen the equations used here and equations in Section 4.3.1.1 is that the quantities of $\gamma_{i,j}, j = 2, ..., \ell_i$ are increased because of the parallel set up. We also notice that the traffic offered to intermediate nodes on the same path is the same under this rule (again, because of parallel set up). As shown in Figure 2, each class of traffic is further divided into two types. Recall that $\gamma^s_{i,j}$ is the arrival rate for the type of traffic which will be successfully set up and the mean holding time for this kind of traffic is $\delta^s_{i,j}$. Also recall that $\gamma^f_{i,j}$ is the type of traffic that will be blocked downstream and the mean holding time for this kind of traffic is denoted by $\delta^f_{i,j}$. $\gamma^s_{i,j}$, $\gamma^f_{i,j}$, $\delta^s_{i,j}$, and $\delta^f_{i,j}$ are computed as follows,

$$\begin{aligned}
\gamma^s_{i,0} &= \gamma_{i,0}(1-\mathcal{L})^{\ell_i}, \quad i = 1, ..., k, && (40)\\
\gamma^f_{i,0} &= \gamma_{i,0}(1-(1-\mathcal{L})^{\ell_i}), \quad i = 1, ..., k, && (41)\\
\gamma^s_{i,0} &= \gamma_{i,j}(1-\mathcal{L})^{\ell_i - 1}, \quad j = 1, ..., \ell_i, \;\; i = 1, ..., k, && (42)\\
\gamma^f_{i,j} &= \gamma_{i,j}(1-(1-\mathcal{L})^{\ell_i - 1}), \quad j = 1, ..., \ell_i, \;\; i = 1, ..., k, && (43)\\
\delta^s_{i,0} &= \left(\sum_{n=1}^{\ell_i} \frac{1}{n}\right) \bar{T} + (\ell_i + 1)D_p + \bar{\tau}, \quad i = 1, ..., k, && (44)\\
\delta^f_{i,0} &= \left(\sum_{n=1}^{\ell_i} \frac{\binom{\ell_i}{n}(1-\mathcal{L})^{\ell_i - n}\mathcal{L}^n}{1-(1-\mathcal{L})^{\ell_i}} \cdot \frac{\bar{T}}{n}\right) + (\ell_i + 1)D_p, \quad i = 1, ..., k, && (45)\\
\delta^s_{i,j} &= \left(\sum_{n=2}^{\ell_i} \frac{1}{n}\right) \bar{T} + (\ell_i - j + 1)D_p + \bar{\tau}, \quad j = 1, ..., \ell_i, \;\; i = 1, ..., k, && (46)\\
\delta^f_{i,j} &= \bar{T}_{i,j}, \quad j = 1, ..., \ell_i, \;\; i = 1, ..., k. && (47)\\
& && (48)
\end{aligned}$$

The equations for $\gamma_{i,0}$ are different from equations for $\gamma_{i,j}$, $j = 1, ..., \ell_i$ because a call request must be processed by the source node before sending it to intermediate nodes in parallel.

Node-offered traffic

Let us now focus on a single node and compute the traffic rate offered to this node. The total traffic offered to this node by call requests which reach this node as the source node is referred to as G_0 and is given by:

$$G_0 = \lambda + M \sum_{i=1}^{k-1} \gamma_{i,0}(P^i_{fail} - \mathcal{L}) \qquad (49)$$

This is the same equation we used for sequential OOC control rule (equation 11). Recall that G_0 includes *"first time"* exogeneous traffic as well as call setup attempts which have been previously blocked. The node-offered traffic, G, is then given by

$$G = G_0 + M \sum_{i=1}^{k} \sum_{j=1}^{\ell_i} \gamma_{i,j} \tag{50}$$

Network performance

Since a call is lost if it is blocked on all paths, the probability that a call request is rejected is given by

$$P_{rej} = \prod_{i=1}^{k} P_{fail}^{i} \tag{51}$$

To compute the average call setup delay, we need more notation. Let $\bar{T}_{fail}^{i}$ be the expected time to know the call request fails at path i given that the call request fails at this path and $\bar{T}_{succ}^{i}$ be the expected time to know the call request has been successfully set up on path i. To compute $\bar{T}_{fail}^{i}$, we need to condition on the number of nodes that the call request fails again. Assume there are n out of ℓ_i nodes that are blocked. Then the time for the destination node to receive the first blocking message is the sum of the processing delay at source node, the propagation delay on this path, and the minimum processing delay of these n nodes which is $\bar{T}/n$. Thus we have

$$\bar{T}_{fail}^{i} = (1-\mathcal{L}) \left(\bar{T} + \sum_{n=1}^{\ell_i} \frac{\binom{\ell_i}{n} (1-\mathcal{L})^{\ell_i - n} \mathcal{L}^n}{P_{fail}^{i}} \frac{\bar{T}}{n} + (\ell_i + 1) D_p \right) \tag{52}$$

We have the first term, $(1-\mathcal{L})$, because blocking at the outgoing link of the source node does not cause any delay (according to our assumption).

$\bar{T}_{succ}^{i}$ is the expected maximum of ℓ_i i.i.d. exponential random variable plus the expeced processing delay at the source node. That is,

$$\bar{T}_{succ}^{i} = (1 + \sum_{j=1}^{\ell_i} \frac{1}{j}) \bar{T} + (\ell_i + 1) D_p \tag{53}$$

The average call setup delay is then computed as follows.

$$\bar{T}_{setup} = \sum_{i=1}^{k} (\sum_{j=1}^{i-1} \bar{T}_{fail}^{j} + \bar{T}_{succ}^{i}) \frac{(1 - P_{fail}^{i}) \prod_{j=1}^{i-1} P_{fail}^{j}}{1 - P_{rej}} \tag{54}$$

4.3.3 Parallel-Sequential Routing Rules

The analytical models for the two controlled-flooding routing rules are much more complicated and are presented in Appendix A.

5 Numerical Results

In this section, we first validate our analytical models by simulation results. Simulation and analytic results are presented for the different routing mechanisms discussed in the previous sections for both homogeneous network and heterogeneous network case. We first consider a homhgeneous network and study the OOC and SOC routing schemes. Then we present a heterogeneous network and study all six routing schemes discussed in previous sections. We then study the effects of call processing delay, propagation delay and admission control function on the call setup delay by using different sets of parameters in the heterogeneous network case.

5.1 Analytical Results versus Simulation Results

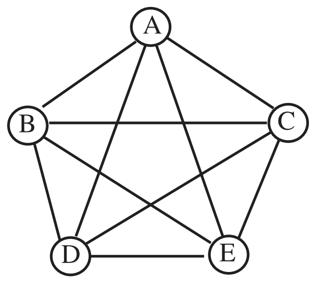
5.1.1 Homogeneous Case

A 5-node fully connected network, as shown in Figure 6(a), is studied in which each node and link are statistically identical. We consider a traffic environment with 20 source-destination pairs. The structure of the routing tree for each O-D pair is identical. Figure 6(b) shows the OOC control routing tree for the O-D pair (A, B). The routing tree for SOC control and crankback control is shown in Figure 6(c). Because of the simple topology, these two mechanisms result in one mechanism.

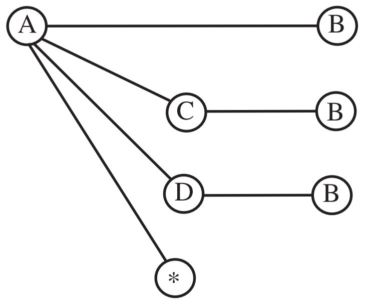
The performance results obtained are based on the following settings:

- The round trip propagation delay, D_p, is set to 0.2 of the per-node average call processing time.
- The mean call holding time,$1/\mu_c$, is set to 1000 (measured in units of average call processing time).
- The maximum number of connections a link can accommodate at a time, C, is set to 200.
- The admission control function is set to $B(x) = (\frac{x}{C})^2$.

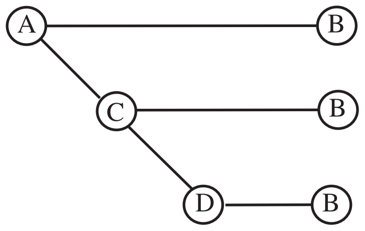
Figure 7(a) and Figure 7(b) compare the analytic results with the simulation results. The vertical lines about each point indicate the 90 percent confidence interval. We note that the analytic and simulation results agree very closely under various traffic loads.



(a) A 5-node fully connected network.



(b) OOC routing tree.



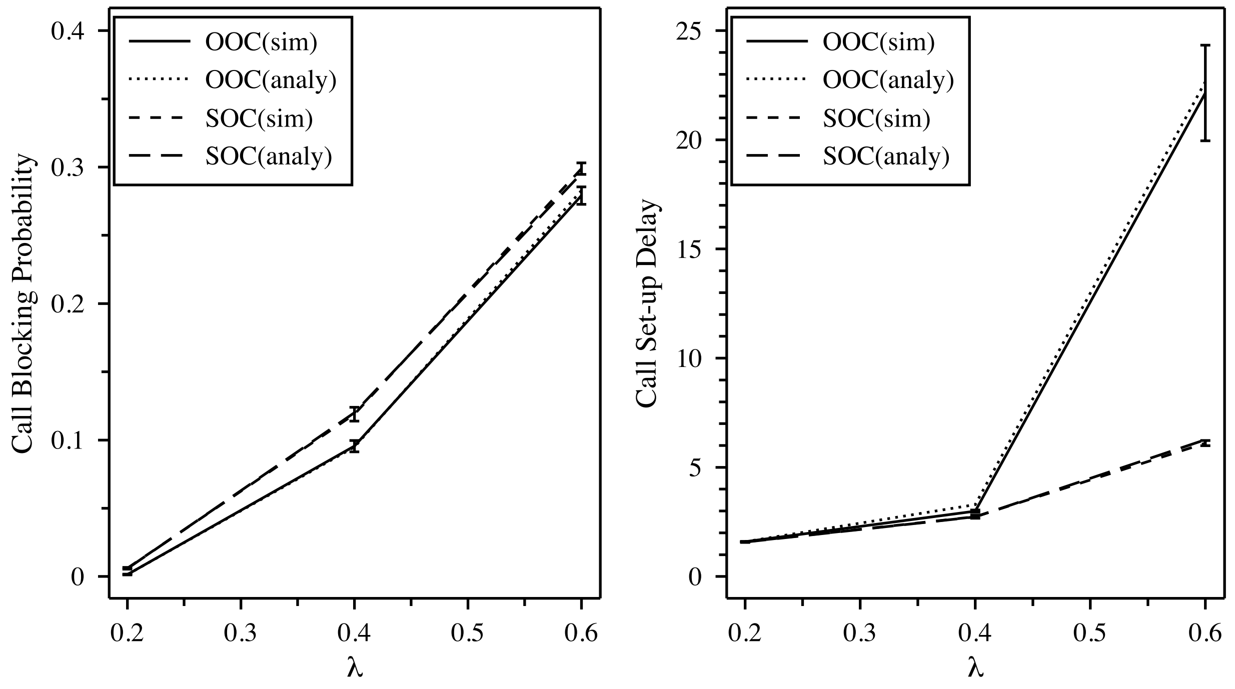
(c) SOC/Crankback routing tree.

Figure 6: Homogeneous example: a 5-node fully connected network

5.1.2 Heterogeneous Case

A 8-node hypercub network, as shown in Figure 8(a), is studied where nodes are still homogeneous but links are heterogeneous. Here we consider a traffic environment with 8 source-destination pairs where the structure of routing tree for each O-D pair is identical. The nodes are still homogeneous because each node is equally likely to be at any position in the routing tree; thus the traffic offered to each node is statistically identical. OOC routing tree for O-D pair (A, G) is shown in Figure 8(b). Figure 8(c) shows the routing tree for both SOC and crankback for O-D pair (A, G). The performance results obtained are based on the same parameter values as in the homogeneous case except for the sequential-parallel OOC rule in which the propagation delay is set to zero.

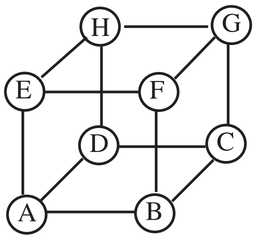
Figure 9 and Figure 10 compare the analytic results with the simulation results. The vertical lines about each point indicate the 90 percent confidence interval. As before, the analytic and simulation results agree very closely for all routing mechanisms.



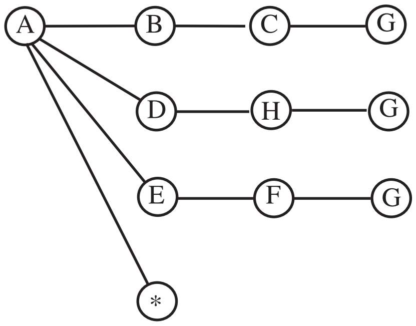
(a) Call Blocking Probability
Simulation vs Analysis
C=200, μ_c=0.001

(b) Call Set-up Delay
Simulation vs Analysis
C=200, μ_c=0.001

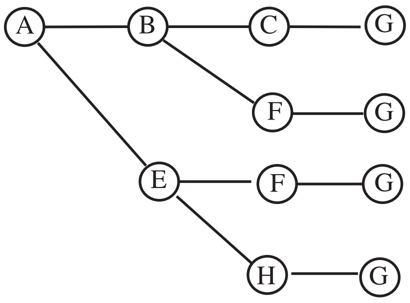
Figure 7: Homogeneous example: simulation vs analysis



(a) An 8-node hypercube network.

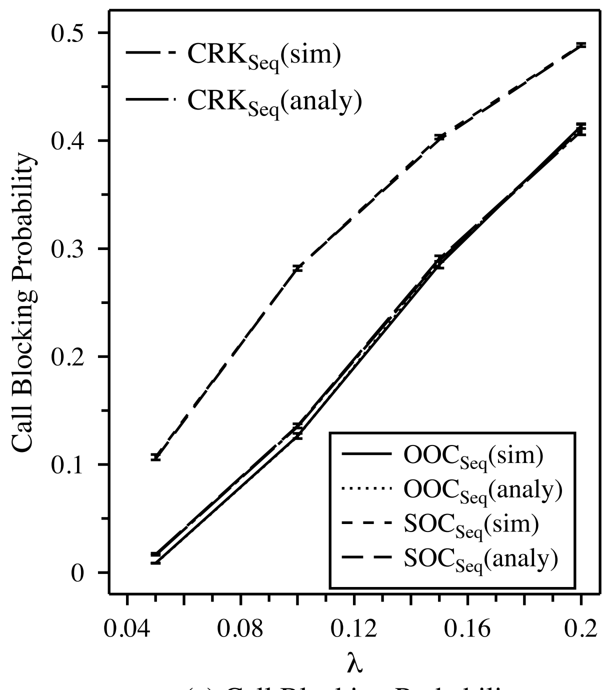


(b) OOC routing tree.

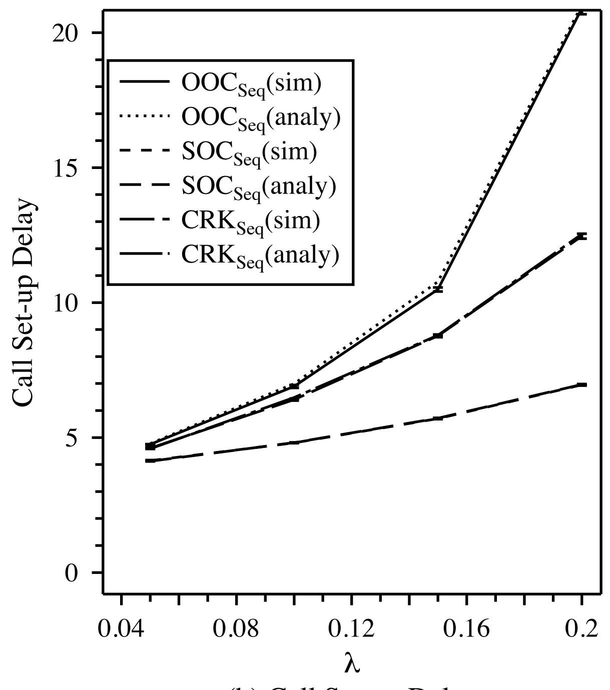


(c) SOC/Crankback routing tree.

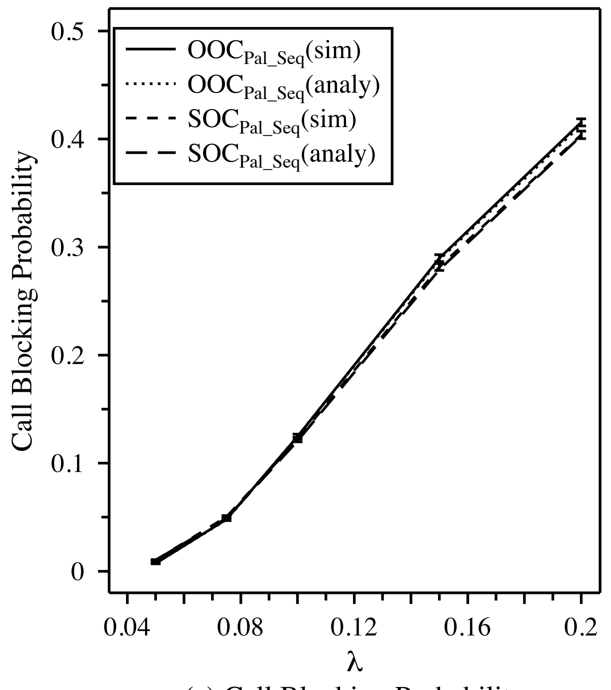
Figure 8: Heterogeneous example: a 8-node hypercube network



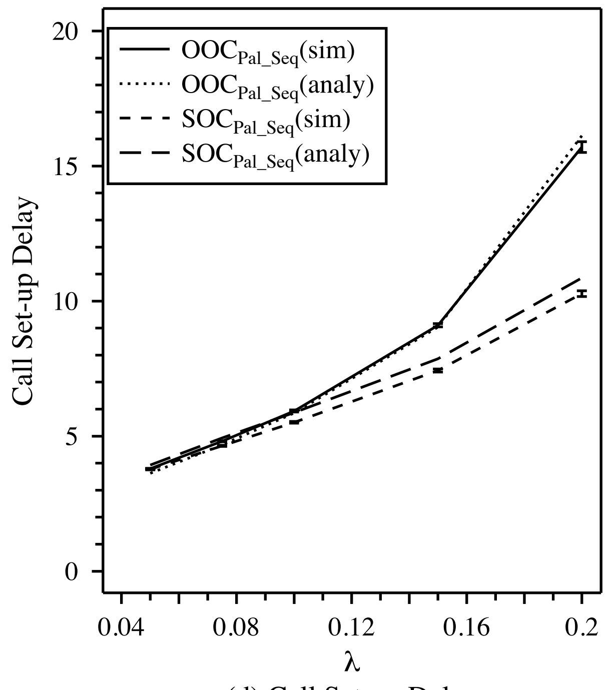
(a) Call Blocking Probability
Simulation vs Analysis
Sequential Rules
C=200, μ_c=0.001



(b) Call Set-up Delay
Simulation vs Analysis
Sequential Rules
C=200, μ_c=0.001

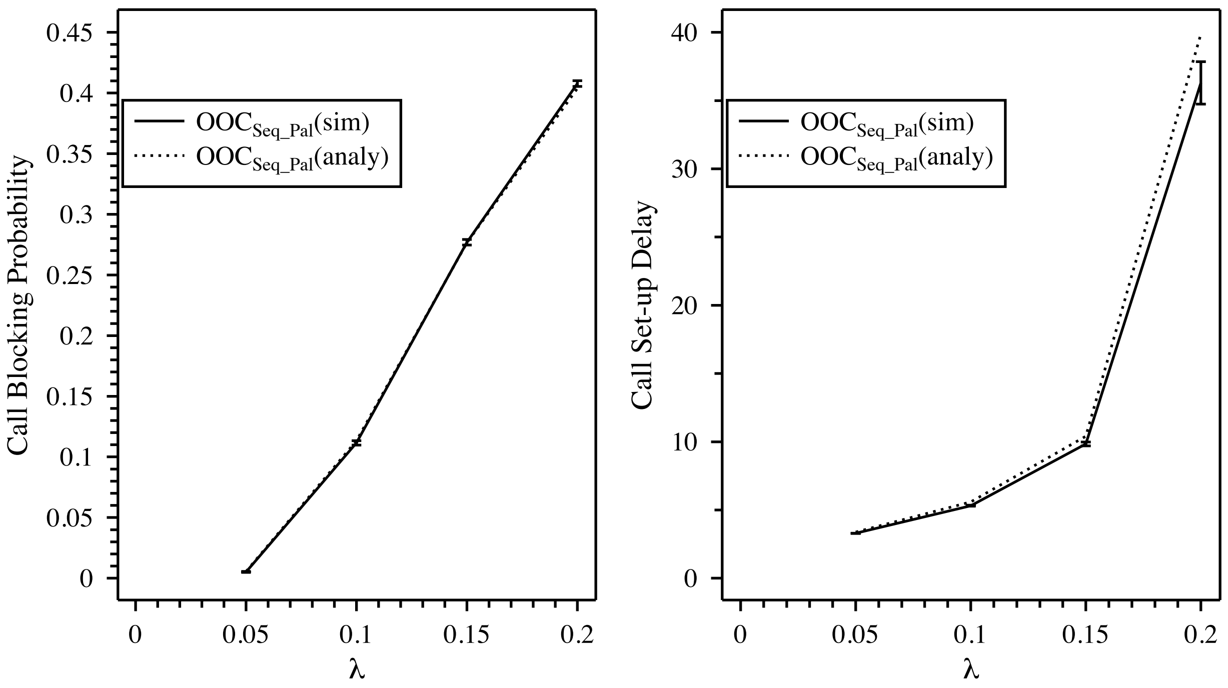


(c) Call Blocking Probability
Simulation vs Analysis
Parallel Rules
C=200, μ_c=0.001



(d) Call Set-up Delay
Simulation vs Analysis
Parallel Rules
C=200, μ_c=0.001

Figure 9: Heterogeneous example: simulation vs analysis



(a) Call Blocking Probability
Simulation vs Analysis
C=200, μ_c=0.001, D_p=0

(b) Call Set-up Delay
Simulation vs Analysis
C=200, μ_c=0.001, D_p=0

Figure 10: Heterogeneous example: simulation vs analysis (for sequential-parallel rule)

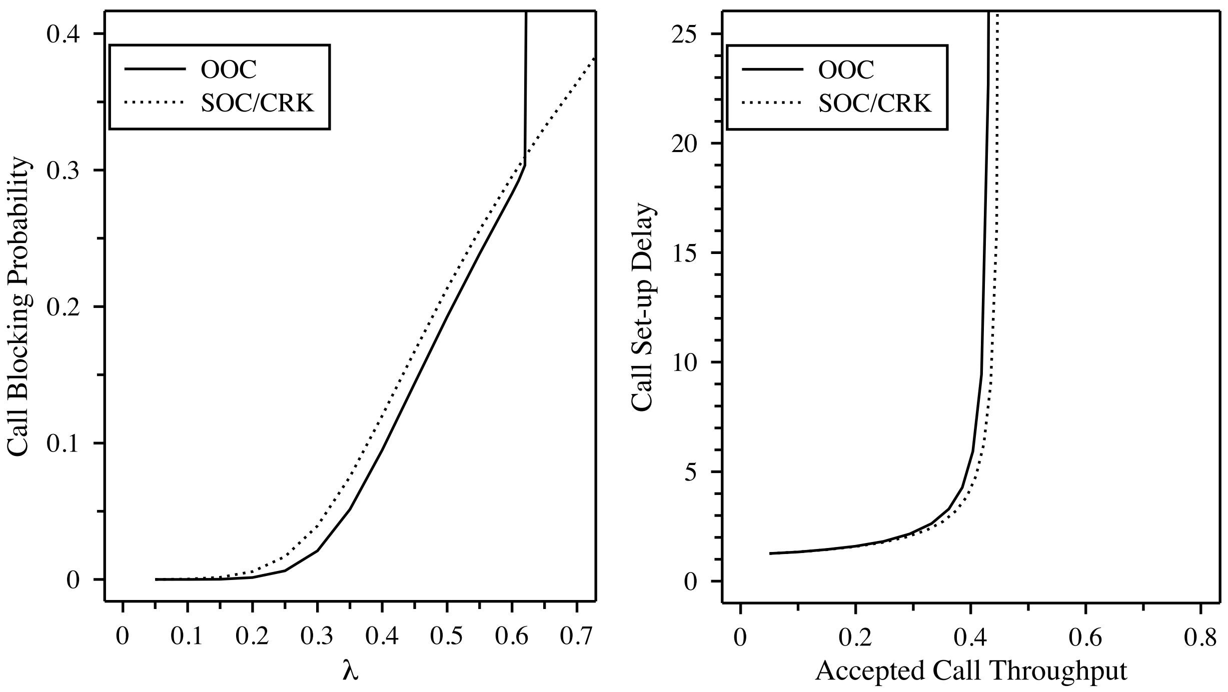
5.2 Comparison of Different Policies

The performance of different routing schemes are compared in Figure 11 and Figure 12. From Figure 11, we can see the trade-off between the call blocking probability and the call setup delay. For a given exogenous call arrival rate, the OOC control scheme yields a lower blocking probability but higher average call setup delay. However, for a given network throughput of accepted calls, as we can observe in Figure 11(b) and Figure 12, the SOC and Crankback control schemes always yield smaller average call setup delays than the OOC scheme. We also observe that the SOC and Crankback schemes yield a higher maximum achievable throughput than the OOC scheme. As compared to the Crankback scheme, the SOC scheme shows a slightly smaller average call setup delay but has a lower 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; thus the blocking probability will approach unity.)

The intuitive explanations for this result are the following. As we compare the OOC scheme to the SOC scheme, we know that in the OOC scheme a call is blocked only when it is blocked on all possible paths while in the SOC scheme a call is blocked if an intermediate node is blocked. Therefore, we would expect that for a given arrival rate, the OOC scheme should yield a lower blocking probability because a call is given more opportunities to set up. But on the other hand, the OOC scheme will generate more node- and link-offered traffic (because of more retrials), this results in longer mean processing delays at the nodes and, consequently, longer mean call set up delays. Similarly, the SOC scheme yields a lower call setup delay than the crankback scheme because it generates less traffic than the crankback scheme. It is not so obvious why the crankback scheme yields a higher maximum achievable throughput.

5.3 The effects of propagation delay and call processing delay

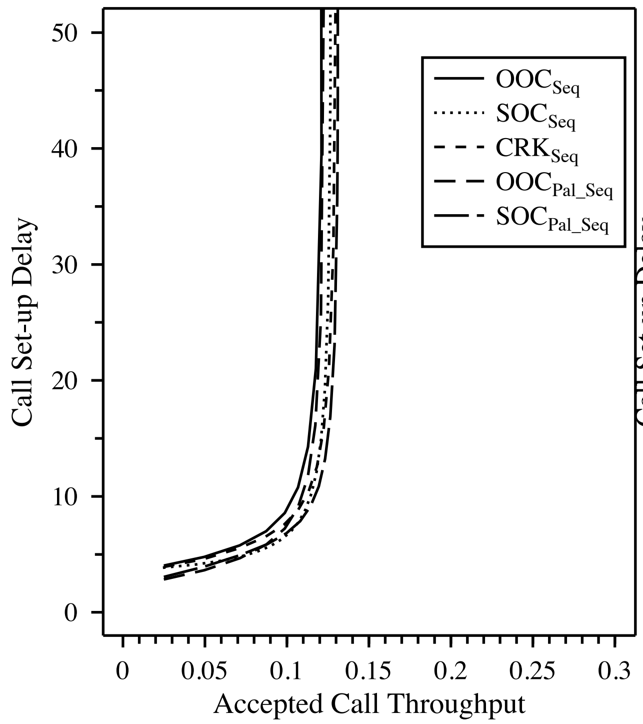
Figure 12 also shows the effects of increasing the propagation delay and the mean control packet processing time. First, when the propagation delay is increased from 0.2 units of mean processing time to 2 units of mean processing time, we find that increasing the propagation delay does not affect the performance of the three sequential routing schemes significantly. (Note that although propagation delay in high speed networks far out shadows packet transmission and queueing delay, it is not likely that it will also dominate call setup time. For example, [5] argues that the packet processing time is several orders of magnitude larger than the packet switching time. We believe our results hold for the case where the propagation delay is not significantly larger (e.g., a factor of 10) than the mean packet processing time.) However, we do notice that the mean call setup delay of the parallel version of the OOC control scheme is



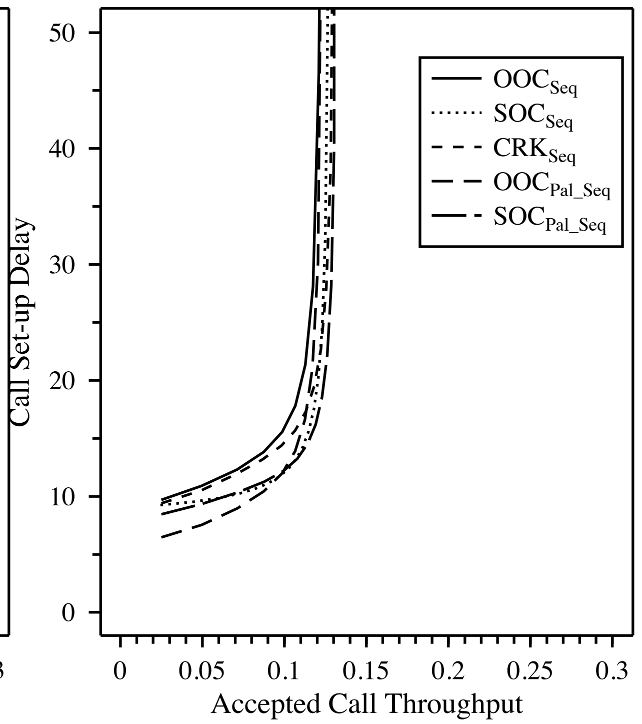
(a) Call Blocking Probability
OOC vs SOC/Crankback
C=200, μ_c=0.001, D_p=0.2

(b) Call Set-up Delay
OOC vs SOC/Crankback
C=200, μ_c=0.001, D_p=0.2

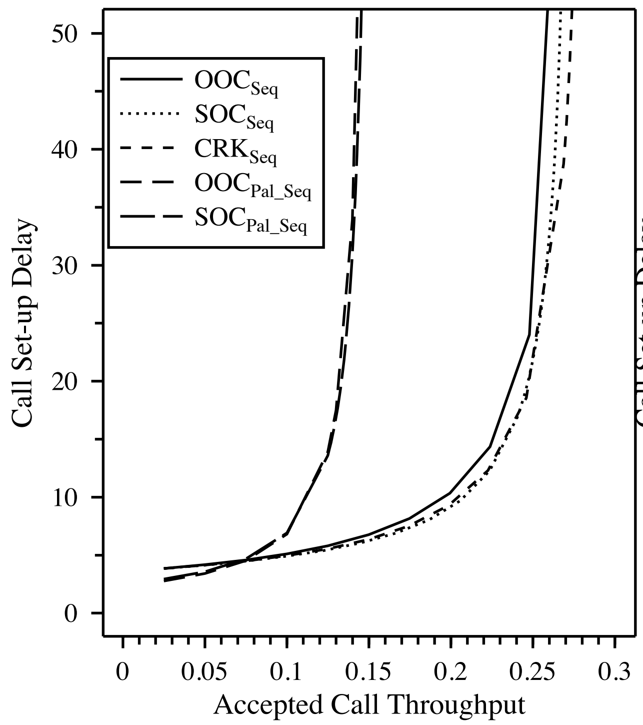
Figure 11: Homogeneous example: Comparison of different routing schemes



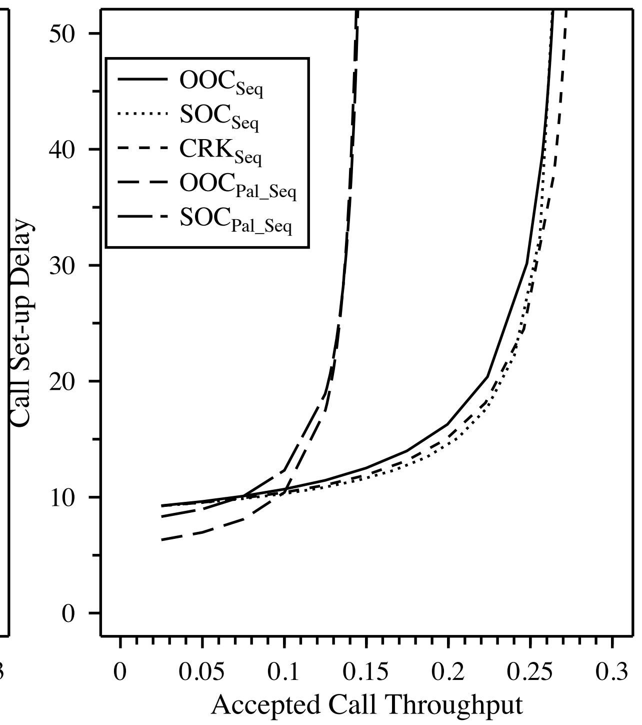
(a) Call Set-up Delay
Comparison of 5 rules
C=200, μ_c=0.001, D_p=0.2



(b) Call Set-up Delay
Comparison of 5 rules
C=200, μ_c=0.001, D_p=2



(c) Call Set-up Delay
Comparison of 5 rules
C=200, μ_c=0.005, D_p=0.2



(d) Call Set-up Delay
Comparison of 5 rules
C=200, μ_c=0.005, D_p=2

Figure 12: Heterogeneous example: effect of propagation delay

significantly lower than the mean call setup delay of other schemes at low network load when the propagation delay is high. This is due to its maximal parallelism of sending call request messages. After all, increasing the propagation delay does not change the maximum achievable throughput for all routing schemes.

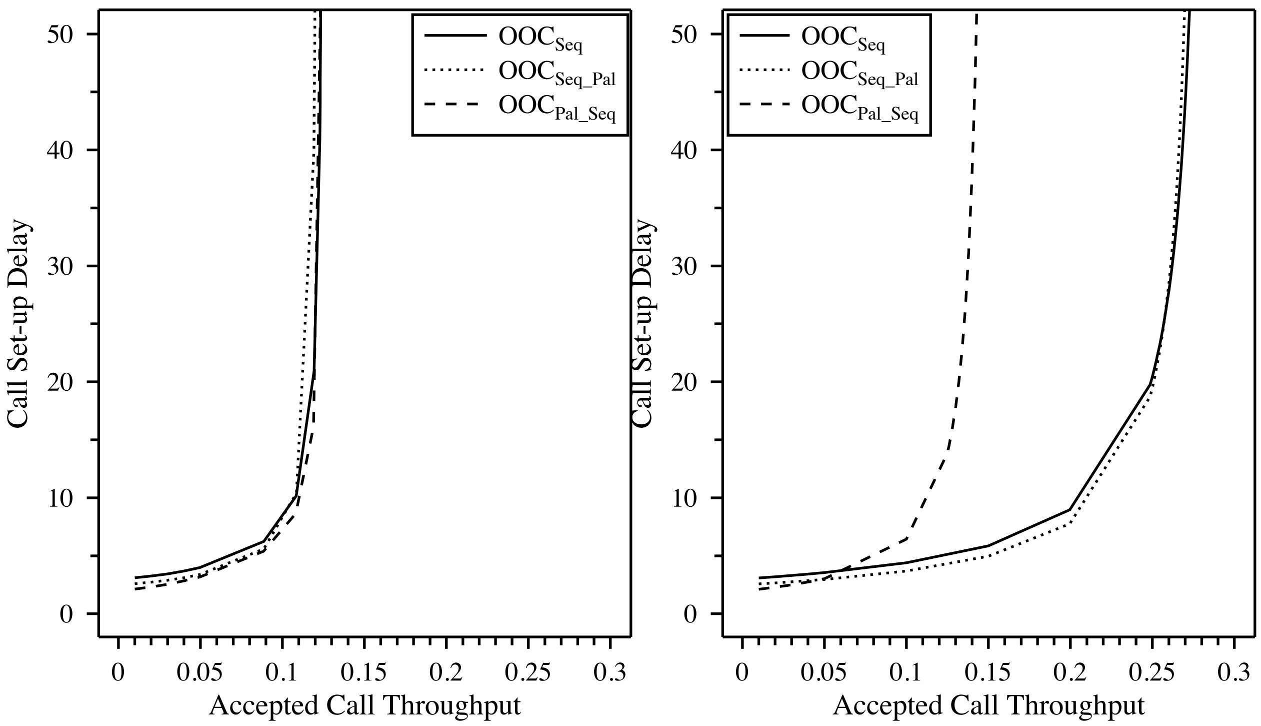
Second, by comparing Figure 12(a) and Figure 12(c) (or Figure 12(b) and Figure 12(d)), we can see that increasing the processing requirements for a call (i.e., increasing μ_c from 0.001 to 0.005) does affect the performance of these five routing schemes tremendously. We know that parallel routing schemes generate more call request messages than sequential rules. Therefore when the processing capacity is very limited, parallel routing schemes will saturate the processing elements very quickly. Thus, sequential schemes can offer much higher throughput. Certainly, at very low traffic loads, we still expect to see that parallel rules can yield lower mean call set up delays than sequential rules. This is also confirmed in the analytic results. On the other hand, when the processing capacity is abundant and the bottleneck is the communication bandwidth, we can see that parallel rules yield not only lower mean call setup delays but also higher throughputs.

Similar effects are also shown in Figure 13 where the sequential OOC, the sequential-parallel OOC and the parallel-sequential OOC routing schemes are compared under different parameter settings. Figure 13 also shows that the sequential-parallel OOC scheme performs better than sequential OOC scheme when the link-level blocking probability is low and performs better than the parallel-sequential OOC when the processing delay is not relatively small as compared to the call holding time. Thus we can conclude that the sequential-parallel OOC scheme performs best when the processing capacity is very limited but the bandwidth is abundant (so that the link-level blocking probability is low).

5.4 The effects of admission control function

As previously indicated, the proceeding results have used $B(x) = x^2$ to model the link-level admission control function. In Figure 14 we study the effects of different forms of admission control. Three forms of admission control function are studied. The first one is a convex function ($B(x) = (\frac{x}{C})^2$), the second one is a linear function ($B(x) = \frac{x}{C}$), and the last one is a concave function ($B(x) = \sqrt{\frac{x}{C}}$). As we can see that, no matter what admission control function is used, SOC and crankback schemes always perform better than the OOC scheme.

By comparing the graphs in Figure 14, we observe the following interesting behavior. First, the network performance 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 bottleneck of the network performance if the admission control function is a convex function. As a consequence, the parallel schemes perform poorly in Figure 14(a) but perform very well in



(a) Call Set-up Delay
$B(x)=(x/C)^2$
$C=200, \mu_c=0.001, D_p=0$

(b) Call Set-up Delay
$B(x)=(x/C)^2$
$C=200, \mu_c=0.005, D_p=0$

Figure 13: Heterogeneous example: simulation vs analysis (for sequential-parallel rule)

Figure 14(c). (As we can see in Figure 14(d), parallel schemes perform poorly when the admission control function is a concave function only when the mean processing delay is extremely high.)

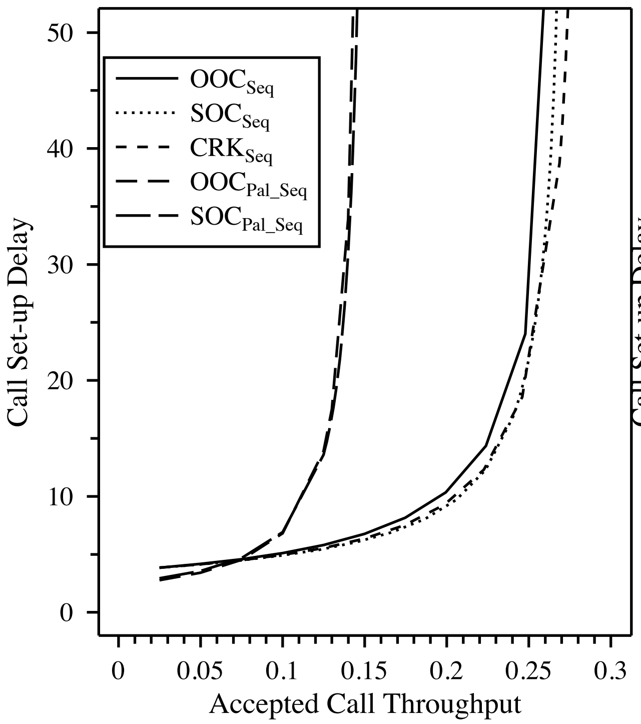
Second, with a convex admission control function, all routing schemes yield higher throughput than with a concave or linear admission control function. This is because with the same number of connections on a link, a concave admission control function and a linear admission control function block more calls than a convex admission control function does. However, we do see an exception. As we compare Figure 14(a) and Figure 14(b), we see that when the throughput of the network is limited by the processing capacity instead of communication bandwidth, parallel routing schemes with a linear admission control function can yield a higher throughput than with a convex control function. The intuitive explanation for this exception is that, since the communication resources are abundant, most of the flooding messages will be successfully received by the destination node and will be discarded because of duplication. Therefore, parallel routing schemes with convex admission control function (which implies lower link-level blocking probability) generate more call request messages that will be discarded and thus saturate the processing elements more quickly.

6 Summary

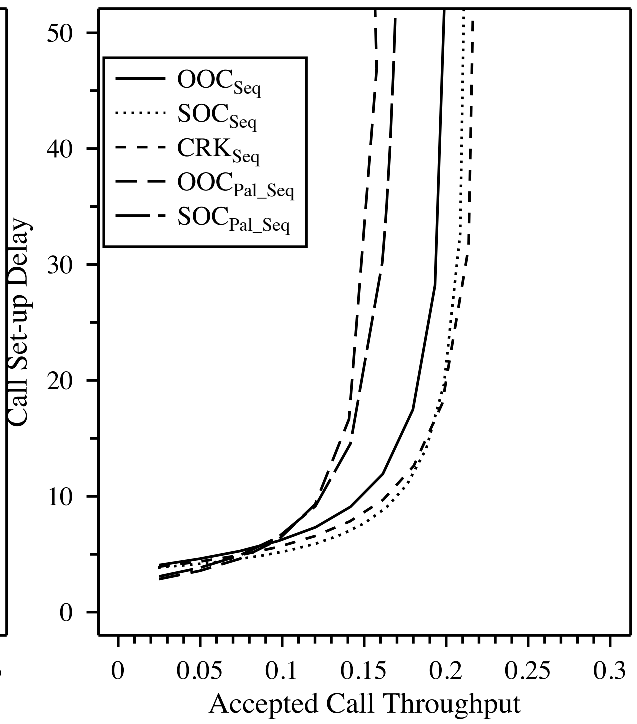
In this paper, six routing mechanisms for high speed network with QOS requirements were studied for both homogeneous and heterogeneous networks; simulation and analytic models were developed to examine the performance of these mechanisms. The effects of call processing delays, propagation delays, admission control function (due to QOS requirements) and routing algorithms on the call setup delay was our particular focus.

First, we find that the propagation delay does not affect the call setup delay significantly as long as it is relatively small as compared to the call holding time. In other words, the bandwidth held for a short amount of blocked-call-clear time due to call requests being blocked downstream does not cause a significant degradation in network performance.

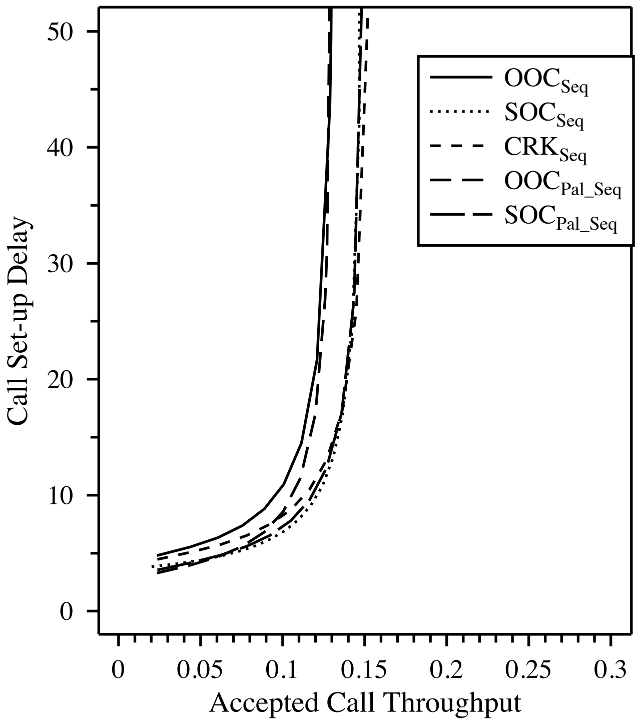
Second, we find that the call processing delay incorporated with the admission control function affects the call setup delay significantly. In general, if call processing capacity is not the bottleneck of the network performance, routing algorithms that make use of flooding schemes provide not only shorter call setup delay, but also higher network throughput. However, the shifting of the bottleneck from the link capacity to call processing capacity not only depends on the call processing delay but also on the admission control function. If the admission control function is convex, the call procesing capacity can easily become the bottleneck. On the other hand, an opposite result is seen when the admission control function is a concave function. The routing scheme that tries path sequentially but checks the bandwidth availabliity



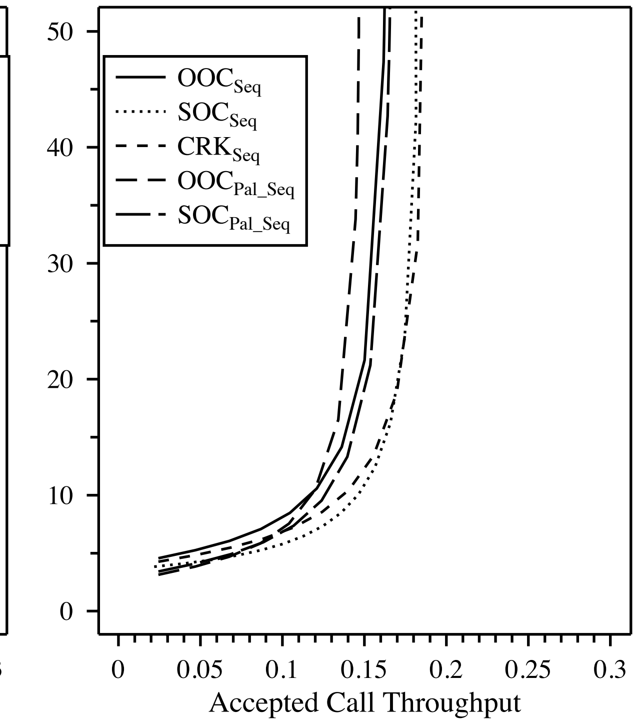
(a) Call Set-up Delay
$B(x)=(x/C)^2$
$C=200$, $\mu_c=0.005$, $D_p=0.2$



(b) Call Set-up Delay
$B(x)=x/C$
$C=200$, $\mu_c=0.005$, $D_p=0.2$

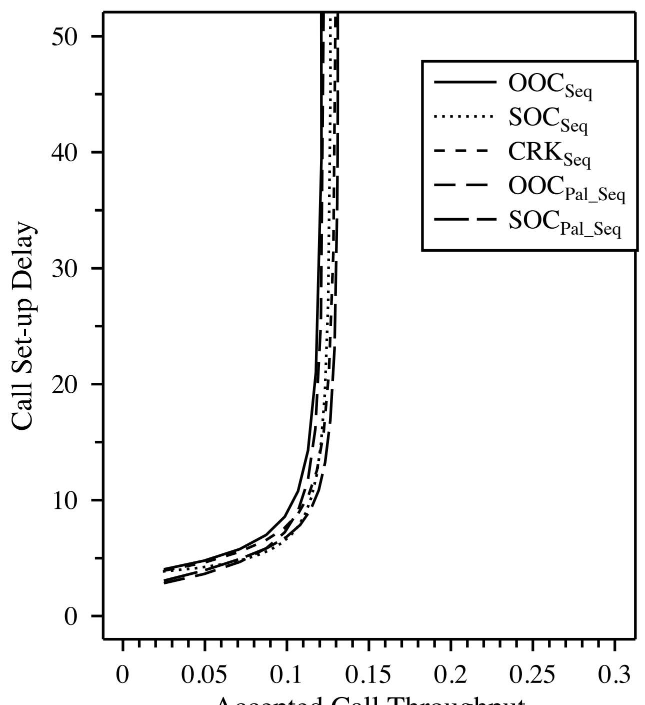


(c) Call Set-up Delay
$B(x)=(x/C)^{1/2}$
$C=200$, $\mu_c=0.005$, $D_p=0.2$

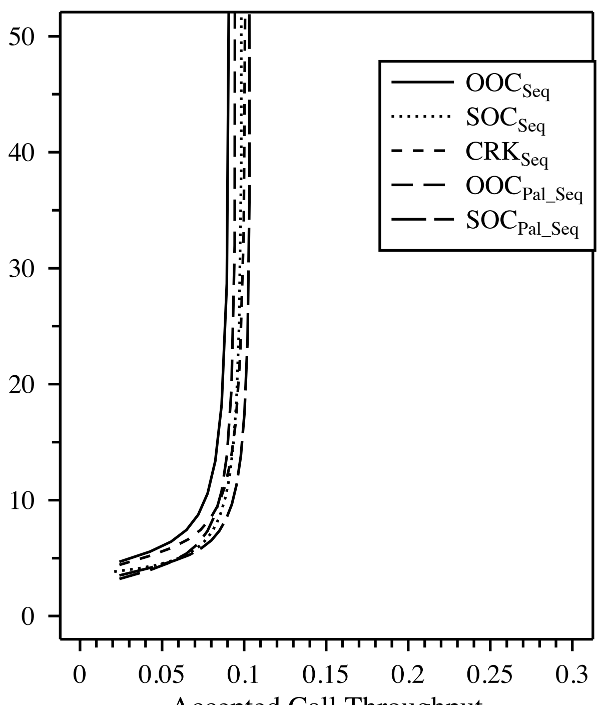


(d) Call Set-up Delay
$B(x)=(x/C)^{1/2}$
$C=200$, $\mu_c=0.05$, $D_p=0.2$

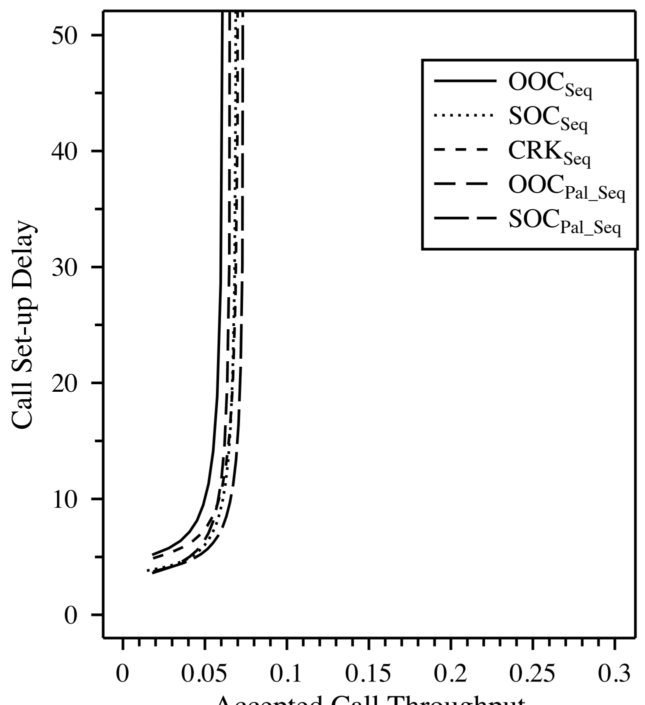
Figure 14: Heterogeneous example: effect of admission control function



(e) Call Set-up Delay
$B(x)=(x/C)^2$
$C=200$, $\mu_c=0.001$, $D_p=0.2$



(f) Call Set-up Delay
$B(x)=x/C$
$C=200$, $\mu_c=0.001$, $D_p=0.2$



(g) Call Set-up Delay
$B(x)=(x/C)^{1/2}$
$C=200$, $\mu_c=0.001$, $D_p=0.2$

Figure 14: Heterogeneous example: effect of admission control function (*Cont*)

at intermediate nodes in parallel seems very promising for networks with very limited processing capacity but abundant bandwidth.

Finally, in comparing different policies, we find that SOC and Crankback control schemes show better performance than the OOC control scheme. Moreover, the SOC control scheme yields smaller call setup delay but lower maximal achieveable throughput than the Crankback control scheme; however, the difference is not significant. The parallel routing schemes perform better than the sequential schemes only when call processing delay is not the bottleneck. In most cases, the parallel version of SOC/Crankback scheme performs slightly better than the parallel version of OOC scheme.

References

[1] CCITT SG. XVIII, "Draft Recommendation I.311."

[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 *Thirteenth International Teletraffic Congress*, June 1991.

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

[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 Calculation 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] P. Semal, "Performance Evaluation of the MKS System: Feasibility Analysis and General Principles," Technical Note N 119, Philips Research Laboratory, December 1989.

[16] 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.

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

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

[19] K. S. Trivedi, *Probability & Statistics with Reliability, Queuing, and Computer Science Applications.* Prentice-Hall, 1982.

Appendix

A Analytical Models for Controlled-Flooding Algorithms in Homogeneous Networks

In this appendix, we present the computation for the link- and node-offered call arrival rate, average call setup delay, and call blocking probability for controlled-flooding versions of OOC routing and SOC/Crankback routing in homogeneous networks.

A.1 OOC Routing Rule

One recent work which also studies the performance of a flooding algorithm is [15]. In [15], the author presents an analysis for the performance of the MKS circuit-switching communication system designed by PKI. The call setup procedure in this system makes use of a flooding scheme to find a free path between any two subscribers (nodes). The quantity of interest in [15] is the probability that the *first* call setup request message received at the destination node has followed a given path. As discussed in [15], an exact way of evaluating this probability is very hard. However, by assuming the waiting times along different paths are statistically independent, we are able to compute the quantity we need. Our analysis differs from [15] by the fact that we have a predefined routing tree for each O-D pair. This simplifies our analysis and enables us to obtain close form expressions.

In the analyses of flooding algorithms, a critical quantity that must be computed is the probability that a call request received by the destination node is a duplicated request. In the following analysis, we thus focus on the computation of this probability. Note that the notation and assumptions used in Chapter 3.3 are also adopted here. To be consistent with the sequential routing algorithms, we also assume that the destination node discards the duplicated request in zero time.

Let $P_{dup}(i)$ be the probability that a call request received through path i is duplicated. To compute $P_{dup}(i)$, we need to introduce some more notation. Let T_i be the total waiting time (processing delay + round trip propagation delay) along path i given that the call request is successfully received through path i. We assume that processing delay (queueing + service) is exponentially distributed with mean $1/\mu$. The sum of processing delay through ℓ_i nodes is then given as an Erlang-ℓ_i random variable. The sum of propagation delays is a deterministic random variable with value $\ell_i D_p$. The CDF and pdf of T_i can be expressed as:

$$F_{T_i}(t) = 1 - e^{-\mu(t-\ell_i D_p)}\left(\sum_{i=0}^{\ell_i-1} \frac{(\mu(t-\ell_i D_p))^i}{i!}\right), \quad t \geq \ell_i D_p,$$

$$f_{T_i}(t) = \frac{\mu(\mu(t-\ell_i D_p))^{\ell_i-1} e^{-\mu(t-\ell_i D_p)}}{(\ell_i - 1)!}, \quad t \geq \ell_i D_p.$$

To compute $P_{dup}(i)$, let us consider the situation where call requests are successfully set-up on paths $i_1, ..., i_m$ and i, the one under consideration. Let S be a set of paths such that $S = \{i_1, ..., i_m\}$. Let the random variable T_j be the total waiting time on path j and the random variable T^S_{min} be the time for the destination node to receive the first call request from one of the paths in S. If $\{T_j\}_{j \in S}$ is a sequence of independent random variables, T^S_{min} is given by the following equation.

$$F_{T^S_{min}}(t) = 1 - \prod_{j \in S} [1 - F_{T_j}(t)].$$

$P_{dup}(i)$ is then given by

$$P_{dup}(i) \quad = \quad \sum_{\substack{i \notin S \\ S \subseteq \mathcal{S}}} P'_S P(T^S_{min} < T_i)$$

where $\mathcal{S}$ is the set of all possible sets of paths between the same O-D pair of path i and P'_S is the probability that the call request are successfully set-up only on paths in S given that the call request is successfully set-up on path i.

$$P_S' = \prod_{j\in S}(1 - P_{fail}(j)) \prod_{j \notin S \bigcup \{i\}} P_{fail}(j).$$

Link-offered call requests

The offered call request rate of each class of calls at the link under consideration is computed as follows. The reader is referred to Section 4.3 for the definition of notations used.

$$\begin{aligned}
\gamma_{i,0} &= \frac{\lambda}{M}, \quad i = 1, ..., k, \\
\gamma_{i,j} &= \gamma_{i,j-1}(1 - \mathcal{L}), \\
&= \frac{\lambda}{M}(1 - \mathcal{L})^j, \quad j = 1, ..., \ell_i.
\end{aligned}$$

Each class of calls can then be divided into three types. The arrival rate of the first type of call, which will eventually be accepted if not blocked on the link under investigation, is referred to as $\gamma_{i,j}^s$; the arrival rate of the second type of calls, which will be blocked downstream if not blocked on this link, is referred to as $\gamma_{i,j}^f$; the third type of calls, which has arrival rate $\gamma_{i,j}^{dup}$, will eventually be rejected not because it is blocked on some downstream link but because it is a duplicated request for the same call. The arrival rates and call holding times for these types of traffic are given by:

$$\begin{aligned}
\gamma_{i,j}^s &= \gamma_{i,j}(1 - \mathcal{L})^{\ell_i - j}(1 - P_{dup}(i)), \quad i = 1, ..., k \;\; j = 0, 1, ..., \ell_i, \\
\gamma_{i,j}^f &= \gamma_{i,j}(1 - (1 - \mathcal{L})^{\ell_i - j}), \quad i = 1, ..., k \;\; j = 0, 1, ..., \ell_i, \\
\gamma_{i,j}^{dup} &= \gamma_{i,j}(1 - \mathcal{L})^{\ell_i - j} P_{dup}(i), \quad i = 1, ..., k \;\; j = 0, 1, ..., \ell_i, \\
\delta_{i,j}^s &= (\ell_i - j) * (\bar{T} + D_p) + D_p + \bar{\tau}, \\
\delta_{i,j}^f &= \bar{N}_{i,j}(\bar{T} + D_p), \\
\delta_{i,j}^{dup} &= (\ell_i - j) * (\bar{T} + D_p) + D_p.
\end{aligned}$$

Recall that $\bar{N}_{i,j}$ is the expected number of nodes a call setup message visits after the jth node on path i given that it has successfully reserved resources at the jth node and fails at some node further down path i. $\bar{N}_{i,j}$ is given by equation (3).

Node-offered call requests

The aggregated rate of offered calls to a node is computed as follows,

$$G = \lambda + M \sum_{i=1}^{k} \sum_{j=1}^{\ell_i} \gamma_{i,j}.$$

Call setup delay and call blocking probability

Recall that

$$P_{fail}(i) = 1 - (1 - \mathcal{L})^{\ell_i + 1}$$

and the end-to-end call blocking probability is

$$P_{rej} = \prod_{i=1}^{k} P_{fail}(i).$$

To compute the average call setup time, let T_{min}^{S} be the call setup time given that call requests are successfully set up on paths in $S \subseteq \mathcal{S}$. $\bar{T}_{min}^{S}$ can be obtained by computing the Laplace transform of $f_{T_{min}^{S}}(t)$. The average call set-up delay can then be computed as follows.

$$T_{setup} = \sum_{S \subseteq \mathcal{S}} P_S \bar{T}_{min}^{S}$$

where P_S is the probability that the call request are successfully set up only on paths in S given that at least one call request is successfully set up,

$$P_S = \frac{\prod_{j \in S}(1 - P_{fail}(j)) \prod_{j \notin S} P_{fail}(j)}{1 - P_{rej}}.$$

A.2 SOC/Crankback Routing Rule

The fact that a call request received by the destination node can be a duplicate request makes the analysis much more complicated. We first introduce some additional notations:

- $\mathcal{T}_{i_1,\ldots,i_\ell}$: Let $\mathcal{T}_{i_1,\ldots,i_\ell}$ be the delay of the first call request sent by node $(i_1, \ldots, i_\ell)$ to the destination given that at least one call request originated from node $(i_1, \ldots, i_\ell)$ is received successfully by the destination node. $\mathcal{T}_{i_1,\ldots,i_\ell}$ is approximated by following:

$$\mathcal{T}_{i_1,\ldots,i_\ell} = \hat{\mathcal{T}}_{i_1,\ldots,i_\ell} + \mathcal{D}_p + T \qquad (55)$$

  where random variable T is the processing delay required at node $(i_1, \ldots, i_\ell)$ and random variable $\mathcal{D}_p$ is the propagation delay. $\hat{\mathcal{T}}_{i_1,\ldots,i_\ell}$ is the time from a call request being successfully forwarded to node $(i_1, \ldots, i_\ell)$'s children in the routing tree until the call request is received by the destination node. Thus $\hat{T}_{i_1,...,i_\ell} = 0$ if $(i_1, ..., i_\ell)$ is label for the destination node. Otherwise, let m be a subset of the children of node $(i_1, ..., i_\ell)$. The CDF of $\hat{T}_{i_1,...,i_\ell}$ is then given by:

$$F_{\hat{T}_{i_1,...,i_\ell}}(t) = \sum_{m \subseteq \mathcal{M}_{(i_1,...,i_\ell)}} P_m F_{T^m_{min}}(t). \tag{56}$$

where $\mathcal{M}_{(i_1,...,i_\ell)}$ is the set of all possible subsets of the children of node $(i_1, ..., i_\ell)$. The term P_m in equation (56) is the probability that the call requests sent to the nodes in m are eventually received by the destination node.

$$P_m = \frac{\prod_{(i_1,...,i_\ell,j)\in m}(1-\mathcal{L})(1-\bar{P}_{i_1,...,i_\ell,j})\prod_{(i_1,...,i_\ell,j')\notin m}(\mathcal{L}+(1-\mathcal{L})\bar{P}_{i_1,...,i_\ell,j'})}{1-\bar{P}_{i_1,...,i_\ell}}. \tag{57}$$

The term $\bar{P}_{i_1,...,i_\ell}$ in equation (57) is the probability that a call request is blocked at node $(i_1, ..., i_\ell)$ or later. The computation of this probability is the same as in the sequential crankback routing rule.

The random variable T^m_{min} in equation (56) is the time from the call request is successfully forwarded by node $(i_1, ..., i_\ell)$ until the call request first received by the destination node given that only call requests forwarded by nodes in m are successfully received. Thus T^m_{min} is the minimum of $\mathcal{T}_{i_1,...,i_\ell,j}$'s, for all $(i_1, ..., i_\ell, j) \in m$; i.e.,

$$T^m_{min} = \min_{(i_1,...,i_\ell,j)\in m} \mathcal{T}_{i_1,...,i_\ell,j}. \tag{58}$$

The distribution of $\min(\mathcal{T}_{i_1,...,i_\ell,j})$ can be computed as in the case of parallel OOC routing rule.

- $BT_{i_1,...,i_\ell}$**:** the delay until node $(i_1, ..., i_\ell)$ determines that all of the call requests that it forwarded are rejected given that they are rejected. It is defined as,

$$BT_{i_1,...,i_\ell} = \max_{1\le j\le k_{i_1,...,i_\ell}} (BT_{i_1,...,i_\ell,j}) + T + \mathcal{D}_p. \tag{59}$$

  $BT_{i_1,...,i_\ell} = 0$ if $(i_1, ..., i_\ell)$ is label for the destination node.

- $P_{dup}((i_1, ..., i_\ell), j)$**:** Let $P_{dup}((i_1, ..., i_\ell), j)$ be the probability that a call request sent by node $(i_1, ..., i_\ell)$ on its jth outgoing link is a duplicated request given that the call request is eventually successfully received by the destination node. Let us denote the jth outgoing link of node $(i_1, ..., i_\ell)$ by ℓ. For ease of explanation, let us introduce some additional notations.

- Let $\mathcal{S}$ be the set of all possible paths between the O-D pair under consideration. Note that different paths may share common nodes and links.
- Let S_ℓ be the set of paths which contain link ℓ. That is, $\forall p \in S_\ell, \ell \in p$.
- Denote the complement of set S by $\bar{S}$. That is, $S \bigcup \bar{S} = \mathcal{S}$.
- Let $A(\ell)$ be the event that at least one call setup message is successfully received through a path in S_ℓ (i.e., at least one successfully received message has successfully reserved some resource on link ℓ.)
- Let $A^\ell_{S_1,S_2}$ be the event that call setup messages through paths in $S_1 \bigcup S_2$ are successfully received and messages through paths in $\bar{S}_1 \bigcap \bar{S}_2$ are blocked, where $S_1 \subseteq S_\ell$ and $S_2 \subseteq \bar{S}_\ell$.
- Let $Z(\ell)$ be the set of nodes between and including the source node and node $(i_1, ..., i_\ell)$. That is, $Z(\ell) = \{(0), (i_1), (i_1, i_2), ..., (i_1, ..., i_\ell)\}$.
- Let $\psi(S, a)$ be a subset of S such that $\forall p \in \psi(S, a), a \in p$.
- Let $\chi(a, S)$, where $\forall p \in S, a \in p$, be the time between when node a sends messages and the destination node receives the first message given that messages sent through paths in S are all successfully received.

For a given event $A^\ell_{S_1,S_2}$, a successfully received message through link ℓ is a duplicated request if at least one message sent through a path in S_2 is received first by the destination node. Thus, $P_{dup}(\ell|A^\ell_{S_1,S_2})$ is given by

$$P_{dup}(\ell|A^\ell_{S_1,S_2}) = 1 - P(\bigwedge_{a \in Z(\ell)} \chi(a, S_1) < \chi(a, \psi(S_2, a))). \tag{60}$$

Therefore, $P_{dup}(\ell)$ is given by

$$P_{dup}(\ell) = \sum_{\substack{S_1 \subseteq S_\ell \\ S_1 \neq \emptyset}} \sum_{S_2 \subseteq \bar{S}_\ell} P(A^\ell_{S_1,S_2}|A(\ell)) P_{dup}(\ell|A^\ell_{S_1,S_2}) \tag{61}$$

where $P(A^\ell_{S_1,S_2}|A(\ell))$ is the conditional probability for event $A^\ell_{S_1,S_2}$ given event $A(\ell)$. Clearly, the computation is combinatorial and is very cumbersome. As noted in [15], approximation techniques are needed for general large networks. However, for the sparse network such as NSFNET we study in our numerical example, the computation is still tractable.

- $\bar{P}_{i_1,...,i_\ell}$: the probability that a call request is blocked at node $(i_1, ..., i_\ell)$ or later given that it has reached node $(i_1, ..., i_\ell)$. This probability is computed as we did in analysis of the crankback routing rule.

Link-offered call requests

The equations used to compute the rate of link-offered calls are presented without further discussion.

$$\begin{aligned}
\gamma_{(0),j} &= \frac{\lambda}{M}, \quad j = 1, ..., k_0, \\
\gamma_{(i),j} &= \gamma_{(0),i}(1 - \mathcal{L}), \quad i = 1, ..., k_0, \;\; j = 1, ..., k_i, \\
\gamma_{(i_1,...,i_\ell),j} &= \gamma_{(i_1,...,i_{\ell-1}),i_\ell}(1 - \mathcal{L}), \quad j = 1, ..., k_{i_1,...,i_\ell}, \\
\gamma^s_{(i_1,...,i_\ell),j} &= \gamma_{(i_1,...,i_\ell),j}(1 - \bar{P}_{i_1,...,i_\ell,j})(1 - P_{dup}((i_1, ..., i_\ell), j)), \\
\gamma^f_{(i_1,...,i_\ell),j} &= \gamma_{(i_1,...,i_\ell),j}\bar{P}_{i_1,...,i_\ell,j}, \\
\gamma^{dup}_{(i_1,...,i_\ell),j} &= \gamma_{(i_1,...,i_\ell),j}(1 - \bar{P}_{i_1,...,i_\ell,j})P_{dup}((i_1, ..., i_\ell), j), \\
\delta^s_{(i_1,...,i_\ell),j} &= \bar{T}_{i_1,...,i_\ell,j} + D_p + \bar{\tau}, \\
\delta^f_{(i_1,...,i_\ell),j} &= \bar{BT}_{i_1,...,i_\ell,j} + D_p, \\
\delta^{dup}_{(i_1,...,i_\ell),j} &= \bar{T}_{i_1,...,i_\ell,j} + D_p.
\end{aligned} \tag{62}$$

Node-offered call requests

Recall that $G_{i_1,...,i_\ell}$ is the rate at which call requests reach this node as the $(i_1, ..., i_\ell)$ node for all source/destination pairs in the network. $G_{i_1,...,i_\ell}$'s can be easily computed by

$$\begin{aligned}
G_0 &= \lambda, \\
G_i &= \lambda(1 - \mathcal{L}), \quad i = 1, ..., k_0, \\
G_{i_1,...,i_\ell} &= G_{i_1,...,i_{\ell-1}}(1 - \mathcal{L}), \\
&= \lambda(1 - \mathcal{L})^\ell.
\end{aligned}$$

The node-offered call request rate, G, is then given by

$$G = \sum_{\forall(i_1,...,i_\ell)} G_{i_1,...,i_\ell}.$$

Call setup delay and call blocking probability

Recall that the source node has the label "0". Thus the end-to-end call blocking probability is given by

$$P_{rej} = \bar{P}_0,$$

and the average call set-up delay is given by

$$\bar{T}_{setup} = \bar{T}_0.$$

B Analytical Models for Heterogeneous Networks

With the same analytical technique, the analytical model can be extended to heterogeneous networks. In this Appendix, we show how the homogeneous assumption can be relaxed.

Without the homogeneity assumption, each O-D pair has its own routing tree, each node may have a different processing delay, each link may have a different blocking probability and propagation delay. Thus we need to introduce some new notation:

- W: the set of all O-D pairs, $W = \{w\}$.
- R_w: the routing tree of O-D pair w.
- λ_w: the external call arrival rate for O-D pair w.
- $\bar{T}_x$: the average processing delay through node x.
- $\mathcal{L}_{(i,j)}$: the steady state blocking probability of link (i, j).
- $D_{(i,j)}$: the propagation of link (i, j).

The network performance is obtained by examining every O-D pair. In the following analysis, we focus on an arbitrary O-D pair w.

B.1 Sequential Rules

B.1.1 OOC Routing Rule

Assume that there are k_w paths in the 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 with $\ell_w^i - 1$ intermediate nodes, $a_{w,i,1}$ is the source node and a_{w,i,ℓ_w^i+1} is the destination node $(\forall i)$.

The following notation is used for the analysis of OOC routing rule:

- Ψ_x: the set of paths which consists of node x,
$$\Psi_x = \{R_w^i | x \in R_w^i\}.$$
- $\Psi_{(x,y)}$: the set of paths which consists of link (x, y),
$$\Psi_{(x,y)} = \{R_w^i | (x, y) \in L(R_w^i)\}.$$
- $L(r)$: a function yields the set of links on route r. For example, Let $r = (a_1, a_2, ..., a_\ell)$. $L(r)$, the set of links in route r, is
$$L(r) = \{(a_1, a_2), (a_2, a_3), ..., (a_{\ell-1}, a_\ell)\}.$$
- $Sub(r, i, j)$: a sub-vector of r defined by
$$Sub(r, i, j) = (a_i, a_{i+1}, ..., a_j), \;\; 1 \le i \le j \le \ell.$$
- $P_{fail}^{w,i}$: the probability that a call request is blocked on the ith path of R_w.
- $\mathcal{T}_{R_w^{i,j}}$: 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.
- $BT_{R_w^{i,j}}$: 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.

The computation is very similar to homogeneous case, we, thus, present the equations without further discussion.

$$P_{fail}^{w,i} = \sum_{j=1}^{\ell_w^i} \mathcal{L}_{L(Sub(R_w^i,j,j+1))}[\prod_{(a,b)\in L(Sub(R_w^i,1,j))} (1 - \mathcal{L}_{(a,b)})]$$
$$\bar{\mathcal{T}}_{R_w^{i,j}} = \sum_{a\in Sub(R_w^i,j+1,\ell_w^i)} \bar{\mathcal{T}}_a + \sum_{(a,b)\in L(Sub(R_w^i,j,\ell_w^i+1))} D_{(a,b)}$$
$$\bar{BT}_{R_w^{i,j}} = \sum_{n=j+1}^{\ell_w^i} [(\sum_{a\in Sub(R_w^i,j+1,n)} \bar{\mathcal{T}}_a + \sum_{(a,b)\in L(Sub(R_w^i,j,n))} D_{(a,b)}) P_{R_w^{i,j,n}}]$$

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

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

Node-offered call requests

Consider the node x which is reached as the jth node of path i in route tree R_w. Let g_{x,R_w^i} be the call requests offered to node x from path R_w^i. g_{x,R_w^i} is given by:

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

The aggregated traffic offered to node x, G_x, is then given by

$$G_x = \sum_{R_w^i \in \Psi_x} g_{x,R_w^i}.$$

Link-offered call requests

Now consider the link (x, y) which is reached as the jth link of path R_w^i. Let $\gamma_{(x,y),R_w^i}$ be the traffic offered to this link from path R_w^i.

$$\gamma_{(x,y),R_w^i} = \lambda_w[\prod_{m=1}^{i-1} P_{fail}^{w,m}][\prod_{(a,b)\in L(Sub(R_w^i,1,j))} (1-\mathcal{L}_{(a,b)})]$$

As in homogeneous, the link-offered traffic is further classified into two traffic streams:

- $\gamma^s_{(x,y),R_w^i}$: the traffic which will be successfully set up,

$$\gamma^s_{(x,y),R_w^i} = \lambda_w[\prod_{m=1}^{i-1} P_{fail}^{w,m}](1-P_{fail}^{w,i}).$$

  The mean call holding time, $\delta^s_{(x,y),R_w^i}$, is given by

$$\delta^s_{(x,y),R_w^i} = \bar{\tau} + \bar{T}_{R_w^{i,j}}.$$

- $\gamma^f_{(x,y),R_w^i}$: the traffic which will be blocked at some hop down the path,

$$\gamma^f_{(x,y),R_w^i} = \gamma_{(x,y),R_w^i} - \gamma^s_{(x,y),R_w^i}.$$

  The mean call holding time, δ^f_{xy,R_w^i}, is given by

$$\delta^f_{(x,y),R_w^i} = \bar{BT}_{R_w^{i,j}}.$$

Call setup delay and call blocking probability

The end-to-end call blocking probability, P_{rej} is given by

$$P_{rej}^{w} = \prod_{i=1}^{k_w} P_{fail}^{w,i}$$

Let $\bar{T}_{R_w^i}$ 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 i.

$$\begin{aligned}\bar{T}_{R_w^i} &= \sum_{j=2}^{\ell_w^i} \frac{\mathcal{L}_{L(Sub(R_w^i,j,j+1))}[\prod_{(a,b)\in L(sub(R_w^i,1,j))}(1-\mathcal{L}_{(a,b)})]}{P_{fail}^{w,i}} \\ &\quad [\sum_{a\in Sub(R_w^i,1,j)} \bar{T}_a + \sum_{(a,b)\in L(Sub(R_w^i,1,j))} D_{(a,b)}]\end{aligned}$$

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

$$\bar{T}_{setup}^{w} = \sum_{i=1}^{k_w}[\sum_{j=1}^{i-1} \bar{T}_{R_w^j} + \sum_{a\in RD_w^i} \bar{T}_a + \sum_{(a,b)\in RD_w^i} D_{(a,b)}]\frac{(1-P_{fail}^{w,i})\prod_{j=1}^{i-1} P_{fail}^{w,j}}{1-P_{rej}^{w}}$$

B.1.2 SOC Routing Rule

In the following analysis, the routing tree of each O-D pair is labeled as in the homogeneous case. The set of all nodes that are directly connected to the destination node of O-D pair w by a link is now denoted by $\mathcal{D}_w$. We also denote the set of O-D pairs which has node x in its routing tree by Ψ_x:

$$\Psi_x = \{w | x \in R_w\}.$$

And the set of O-D pairs which has link (x, y) in its routing tree is denoted by $\Psi_{(x,y)}$:

$$\Psi_{(x,y)} = \{w | (x, y) \in L(R_w)\}.$$

where $L(R_w)$ is the set of links in routing tree R_w.

Now we define some notation that will be used in the analysis of SOC as well as Crankback routing rules.

- $\bar{P}_{i_1,\ldots,i_\ell}^{w}$: the probability that a call request is blocked at node $(i_1, \ldots, i_\ell)$ or later in tree R_w.

- $\bar{BT}^{w}_{i_1,...,i_\ell}$: the expected time for node $(i_1, ..., i_\ell)$ to know that a call request is blocked given that the call request is blocked at node $(i_1, ..., i_\ell)$ or later.
- $\bar{T}^{w}_{i_1,...,i_\ell}$: the expected time for node $(i_1, ..., i_\ell)$ to know that a call request is successfully set up given the call request is successfully set up through node $(i_1, ..., i_\ell)$.
- $g_{(i_1,...,i_\ell),w}$: traffic offered to node $(i_1, ..., i_\ell)$ in routing tree R_w.
- $g^{j}_{(i_1,...,i_\ell),w}$: the call request traffic offered from node $(i_1, ..., i_\ell)$ to its jth child in routing tree R_w.
- G_x: the aggregated traffic offered to node x.
- $\gamma_{(x,y),w}$: the call request offered traffic on link (x, y) corresponding to O-D pair w.
- $\gamma^{s}_{(x,y),w}$: the portion of $\gamma_{(x,y),w}$ that will be successfully set up.
- $\gamma^{f}_{(x,y),w}$: the portion of $\gamma_{(x,y),w}$ that will be blocked.

We also define Ω^{w}_{x} as the set of outgoing links of node x in routing tree R_w. Following recursive equations are used to compute $\bar{P}^{w}_{i_1,...,i_\ell}$, $\bar{BT}^{w}_{i_1,...,i_\ell}$ and $\bar{T}^{w}_{i_1,...,i_\ell}$.

$$\bar{P}^{w}_{i_1,...,i_\ell} = \left(\sum_{j=1}^{k_{i_1,...,i_\ell}} \Big(\prod_{(a,b)\in Sub(\Omega^{w}_{(i_1,...,i_\ell)},1,j-1)} \mathcal{L}_{(a,b)}\Big)(1 - \mathcal{L}_{((i_1,...,i_\ell),(i_1,...,i_\ell,j))})\bar{P}^{w}_{i_1,...,i_\ell,j}\Big] \right) + \prod_{(a,b)\in\Omega^{w}_{i_1,...,i_\ell}} \mathcal{L}_{(a,b)}$$

with $\bar{P}^{w}_{i_1,...,i_\ell,1} = 0, \ \forall (i_1, ..., i_\ell) \in \mathcal{D}_w$.

$$\bar{BT}^w_{i_1,\dots,i_\ell} = \sum_{j=1}^{k_{i_1,\dots,i_\ell}} \frac{\prod_{(a,b)\in Sub(\Omega^w_{(i_1,\dots,i_\ell)},1,j-1)} \mathcal{L}_{(a,b)}(1-\mathcal{L}_{((i_1,\dots,i_\ell),(i_1,\dots,i_\ell,j))})\bar{P}^w_{i_1,\dots,i_\ell,j}}{\bar{P}^w_{i_1,\dots,i_\ell}} (\bar{BT}^w_{i_1,\dots,i_\ell,j} + D_{((i_1,\dots,i_\ell),(i_1,\dots,i_\ell,j))}) + \bar{T}_{(i_1,\dots,i_\ell)}$$

with $\bar{BT}^w_{i_1,\dots,i_\ell,1} = 0, \ \forall (i_1,\dots,i_\ell) \in \mathcal{D}_w$.

$$\bar{\mathcal{T}}^w_{i_1,\dots,i_\ell} = \sum_{j=1}^{k_{i_1,\dots,i_\ell}} \frac{\prod_{(a,b)\in Sub(\Omega^w_{(i_1,\dots,i_\ell)},1,j-1)} \mathcal{L}_{(a,b)}(1-\mathcal{L}_{((i_1,\dots,i_\ell),(i_1,\dots,i_\ell,j))})(1-\bar{P}^w_{i_1,\dots,i_\ell,j})}{1-\bar{P}^w_{i_1,\dots,i_\ell}} (\bar{\mathcal{T}}^w_{i_1,\dots,i_\ell,j} + D_{((i_1,\dots,i_\ell),(i_1,\dots,i_\ell,j))}) + \bar{T}_{(i_1,\dots,i_\ell)}$$

with $\bar{\mathcal{T}}^w_{i_1,\dots,i_\ell,1} = 0, \ \forall (i_1,\dots,i_\ell) \in \mathcal{D}_w$.

Node-offered call requests

The call request offered to each node and link in routing tree R_w is given by

$$\begin{aligned}
g_{0,w} &= \lambda_w, \\
g^j_{0,w} &= g_{0,w} \prod_{(a,b)\in Sub(\Omega^w_{(0)},1,j-1)} \mathcal{L}_{(a,b)} \quad j=1,\dots,k_0, \\
g_{(i_1,\dots,i_\ell),w} &= g^{i_\ell}_{(i_1,\dots,i_{\ell-1}),w}(1-\mathcal{L}_{((i_1,\dots,i_{\ell-1}),(i_1,\dots,i_\ell))}), \\
g^j_{(i_1,\dots,i_\ell),w} &= g_{(i_1,\dots,i_\ell),w} \prod_{(a,b)\in Sub(\Omega^w_{(i_1,\dots,i_\ell)},1,j-1)} \mathcal{L}_{(a,b)}.
\end{aligned}$$

The aggregated traffic offered to node x is given by

$$G_x = \sum_{w\in\Psi_x} \sum_{(i_1,\dots,i_\ell)\equiv x} g_{(i_1,\dots,i_\ell),w},$$

where "$(i_1,\dots,i_\ell) \equiv x$" denotes node x has id $(i_1,\dots,i_\ell)$ in tree R_w.

Link-offered call requests

The call requests offered to link $((i_1,\dots,i_\ell),(i_1,\dots,i_\ell,j))$ is given by

$$\gamma_{((i_1,\dots,i_\ell),(i_1,\dots,i_\ell,j)),w} = g^j_{(i_1,\dots,i_\ell),w}$$

As before, this traffic is further classified into:

- $\gamma^{s}_{((i_1,...,i_\ell),(i_1,...,i_\ell,j)),w}$: the type of traffic that will be successfully set up,

$$\gamma^{s}_{((i_1,...,i_\ell),(i_1,...,i_\ell,j)),w} = \gamma_{((i_1,...,i_\ell),(i_1,...,i_\ell,j)),w}(1 - \bar{P}^{w}_{i_1,...,i_\ell,j}).$$

  The mean call holding time for this type of traffic is given by

$$\delta^{s}_{((i_1,...,i_\ell),(i_1,...,i_\ell,j)),w} = \bar{T}_{i_1,...,i_\ell,j} + D_{((i_1,...,i_\ell),(i_1,...,i_\ell,j))}.$$

- $\gamma^{f}_{((i_1,...,i_\ell),(i_1,...,i_\ell,j)),w}$: the type of traffic that will be blocked,

$$\gamma^{f}_{((i_1,...,i_\ell),(i_1,...,i_\ell,j)),w} = \gamma_{((i_1,...,i_\ell),(i_1,...,i_\ell,j)),w}\bar{P}^{w}_{i_1,...,i_\ell,j}.$$

  The mean call holding time for this type of traffic is given by

$$\delta_{((i_1,...,i_\ell),(i_1,...,i_\ell,j)),w} = \bar{BT}^{w}_{i_1,...,i_\ell,j} + D_{((i_1,...,i_\ell),(i_1,...,i_\ell,j))}.$$

<u>**Call setup delay and call blocking probability**</u>

The end-to-end call blocking probability for O-D pair w, P^{w}_{rej}, is

$$P^{w}_{rej} = \bar{P}^{w}_{0}.$$

and the average call set up delay for O-D pair w, $\bar{T}^{w}_{setup}$, is

$$\bar{T}^{w}_{setup} = \bar{T}^{w}_{0}.$$

B.1.3 Crankback Routing Rule

Similar to SOC routing rule, we compute $\bar{P}^{w}_{i_1,...,i_\ell}$, $\bar{BT}^{w}_{i_1,...,i_\ell}$, and $\bar{T}^{w}_{i_1,...,i_\ell}$ by following recursive equations. Recall that $(i_1, ..., i_\ell, 1)$ is the destination node if $(i_1, ..., i_\ell) \in \mathcal{D}_w$.

$$\bar{P}^{w}_{i_1,...,i_\ell} = \prod_{j=1}^{k_{i_1,...,i_\ell}} [\mathcal{L}_{((i_1,...,i_\ell),(i_1,...,i_\ell,j))} + (1 - \mathcal{L}_{((i_1,...,i_\ell),(i_1,...,i_\ell,j))})\bar{P}^{w}_{i_1,...,i_\ell,j}]$$

with $\bar{P}^{w}_{i_1,...,i_\ell,1} = 0, \ \forall (i_1, ..., i_\ell) \in \mathcal{D}_w$.

$$\bar{BT}^w_{i_1,\ldots,i_\ell} = \sum_{j=1}^{k_{i_1,\ldots,i_\ell}} \frac{(1-\mathcal{L}_{((i_1,\ldots,i_\ell),(i_1,\ldots,i_\ell,j))})\bar{P}^w_{i_1,\ldots,i_\ell,j}}{\mathcal{L}_{((i_1,\ldots,i_\ell),(i_1,\ldots,i_\ell,j))} + (1-\mathcal{L}_{((i_1,\ldots,i_\ell),(i_1,\ldots,i_\ell,j))})\bar{P}^w_{i_1,\ldots,i_\ell,j}}$$
$$(\bar{BT}^w_{i_1,\ldots,i_\ell,j} + \bar{T}_{i_1,\ldots,i_\ell} + D_{((i_1,\ldots,i_\ell),(i_1,\ldots,i_\ell,j))}) +$$
$$\frac{\mathcal{L}_{((i_1,\ldots,i_\ell),(i_1,\ldots,i_\ell,k_{i_1,\ldots,i_\ell}))}}{\mathcal{L}_{((i_1,\ldots,i_\ell),(i_1,\ldots,i_\ell,k_{i_1,\ldots,i_\ell}))} + (1-\mathcal{L}_{((i_1,\ldots,i_\ell),(i_1,\ldots,i_\ell,k_{i_1,\ldots,i_\ell}))})\bar{P}^w_{i_1,\ldots,i_\ell,k_{i_1,\ldots,i_\ell}}}\bar{T}_{i_1,\ldots,i_\ell}$$

with $\bar{BT}^w_{i_1,\ldots,i_\ell,1} = 0, \ \forall (i_1,\ldots,i_\ell) \in \mathcal{D}_w$.

$$\bar{\mathcal{T}}^w_{i_1,\ldots,i_\ell} = \sum_{j=1}^{k_{i_1,\ldots,i_\ell}} \frac{\prod_{m=1}^{j-1}[\mathcal{L}_{((i_1,\ldots,i_\ell),(i_1,\ldots,i_\ell,m))} + (1-\mathcal{L}_{((i_1,\ldots,i_\ell),(i_1,\ldots,i_\ell,m))})\bar{P}^w_{i_1,\ldots,i_\ell,m}]}{1-\bar{P}^w_{i_1,\ldots,i_\ell}}$$
$$(1-\mathcal{L}_{((i_1,\ldots,i_\ell),(i_1,\ldots,i_\ell,j))})(1-\bar{P}^w_{i_1,\ldots,i_\ell,j})$$
$$[(\sum_{m=1}^{j-1} \frac{(1-\mathcal{L}_{((i_1,\ldots,i_\ell),(i_1,\ldots,i_\ell,m))})\bar{P}^w_{i_1,\ldots,i_\ell,m}}{\mathcal{L}_{((i_1,\ldots,i_\ell),(i_1,\ldots,i_\ell,m))} + (1-\mathcal{L}_{((i_1,\ldots,i_\ell),(i_1,\ldots,i_\ell,m))})\bar{P}^w_{i_1,\ldots,i_\ell,m}}(\bar{BT}^w_{i_1,\ldots,i_\ell,m} +$$
$$\bar{T}_{i_1,\ldots,i_\ell})) + \bar{\mathcal{T}}^w_{i_1,\ldots,i_\ell,j} + \bar{T}_{i_1,\ldots,i_\ell}]$$

with $\bar{\mathcal{T}}^w_{i_1,\ldots,i_\ell,1} = 0, \ \forall (i_1,\ldots,i_\ell) \in \mathcal{D}_w$.

Node-offered call requests

The traffic offered to each link in routing tree R_w is given by

$$\begin{aligned}
g^1_{0,w} &= \lambda_w, \\
g^i_{0,w} &= \lambda_w \prod_{j=1}^{i-1}[\mathcal{L}_{((0),(j))} + (1-\mathcal{L}_{((0),(j))})\bar{P}^w_j] \quad i = 2,\ldots,k_0, \\
g^1_{(i_1,\ldots,i_\ell),w} &= g^{i_\ell}_{(i_1,\ldots,i_{\ell-1}),w}[1-\mathcal{L}_{((i_1,\ldots,i_{\ell-1}),(i_1,\ldots,i_\ell))}], \\
g^i_{(i_1,\ldots,i_\ell),w} &= g^1_{(i_1,\ldots,i_\ell),w}\prod_{j=1}^{i-1}[\mathcal{L}_{((i_1,\ldots,i_\ell),(i_1,\ldots,i_\ell,j))} + (1-\mathcal{L}_{((i_1,\ldots,i_\ell),(i_1,\ldots,i_\ell,j))})\bar{P}^w_{i_1,\ldots,i_\ell,j}] \\
& \qquad i = 1,\ldots,k_{i_1,\ldots,i_\ell}.
\end{aligned}$$

The traffic offered to node $(i_1,\ldots,i_\ell)$ in routing tree R_w is then given by

$$g_{(i_1,\ldots,i_\ell),w} = g^1_{(i_1,\ldots,i_\ell),w} \sum_{j=1}^{k_{i_1,\ldots,i_\ell}-1} g^j_{(i_1,\ldots,i_\ell),w}(1-\mathcal{L}_{((i_1,\ldots,i_\ell),(i_1,\ldots,i_\ell,j))})\bar{P}^w_{i_1,\ldots,i_\ell,j}$$

The aggregated offered traffic to node x can then be computed by

$$G_x = \sum_{w \in \Psi_x} \sum_{(i_1,\ldots,i_\ell) \equiv x} g_{(i_1,\ldots,i_\ell),w}.$$

Link-offered call requests

The call requests offered to link $(i_1, \ldots, i_\ell), (i_1, \ldots, i_\ell, j)$ is

$$\gamma_{(i_1,\ldots,i_\ell)(i_1,\ldots,i_\ell,j),w} = g^j_{(i_1,\ldots,i_\ell),w}.$$

This traffic is further classified into:

- $\gamma^s_{((i_1,\ldots,i_\ell),(i_1,\ldots,i_\ell,j)),w}$: the type of traffic that will be successfully set up,

  $$\gamma^s_{((i_1,\ldots,i_\ell),(i_1,\ldots,i_\ell,j)),w} = \gamma_{((i_1,\ldots,i_\ell),(i_1,\ldots,i_\ell,j)),w}(1 - \bar{P}^w_{i_1,\ldots,i_\ell,j}).$$

  The mean call holding time for this type of traffic is

  $$\delta^s_{((i_1,\ldots,i_\ell),(i_1,\ldots,i_\ell,j)),w} = \bar{T}^w_{i_1,\ldots,i_\ell,j} + D_{((i_1,\ldots,i_\ell),(i_1,\ldots,i_\ell,j))}.$$

- $\gamma^f_{((i_1,\ldots,i_\ell),(i_1,\ldots,i_\ell,j)),w}$: the type of traffic that will be blocked downstream,

  $$\gamma^f_{((i_1,\ldots,i_\ell),(i_1,\ldots,i_\ell,j)),w} = \gamma_{((i_1,\ldots,i_\ell),(i_1,\ldots,i_\ell,j)),w}\bar{P}^w_{i_1,\ldots,i_\ell,j}$$

  The mean call holding time for this type of traffic is

  $$\delta^f_{((i_1,\ldots,i_\ell),(i_1,\ldots,i_\ell,j)),w} = \bar{BT}^w_{i_1,\ldots,i_\ell,j} + D_{((i_1,\ldots,i_\ell),(i_1,\ldots,i_\ell,j))}.$$

Call setup delay and call blocking probability

Equations used in SOC rule also applies here.

B.2 Controlled-Flooding Rules

Extending the analytical methodology to controlled-flooding algorithms for heterogeneous networks is quite straightforward. However, the computation itself is rather complicated. As stated in Appendix A, the most difficult computation in the analysis of controlled-flooding algorithms is the the computation of the probability that a call request received through a particular path is a duplicated request and the average call set up time when more than one call requests have been received. In this section, we derive close form expressions for computing these two quantities for OOC routing rule with the limitation that each routing tree is limited to have at most three paths. The same technique can be applied to SOC/crankback rule. However, the derivation of close form expressions depends heavily on the structure of the routing tree. We are, thus, not able to show the derivation systematically. Therefore, we choose to omit the derivation of these two quantities for SOC/crankback routing rule in this report.

B.2.1 OOC Routing Rule

In the following, we show how to compute the probability that a call request received through a particular path is a duplicated request and the average call set up time when more than one call requests have been received for controlled-flooding OOC algorithm in heterogeneous networks. The notation used in Appendix A and Appendix B.1 is also adopted here.

B.2.2 Computation of $P_{dup}(k)$

Let us first consider the computation of the probability that a call request received through path k is duplicated, $P_{dup}(k)$. Let T_k be the total waiting time (processing delay + round trip propagation delay) along path k given that the call request is successfully received through path k. We assume that there are ℓ_k intermediate nodes on path k and the processing delay (queueing delay + service time) at each node is exponentially distributed with mean $1/\mu_i$, $1 \leq i \leq \ell_k$. Assume that $\mu_i \neq \mu_j$, $\forall i, j$ on path k. Then the CDF and PDF of T_k is given by [19]

$$\begin{aligned} F_{T_k}(t) &= 1 - \sum_{i=1}^{\ell_k} \frac{\prod_{j \neq i} \mu_j}{\prod_{j \neq i}(\mu_j - \mu_i)} e^{-\mu_i(t - Dp_k)}, \\ f_{T_k}(t) &= \sum_{i=1}^{\ell_k} \frac{\prod_{j=1}^{n} \mu_j}{\prod_{j \neq i}(\mu_j - \mu_i)} e^{-\mu_i(t - Dp_k)}, \end{aligned}$$

where Dp_k is the round trip propagation delay for the source node to receive the acknowledgment of the call request through path k.

To compute $P_{dup}(k)$, we need to know how many call requests have been successfully received. It is very difficult to derive a general expression for any number of call requests that are successfully received. Here, we derive the cases of two and three successfully received call requests. Let us consider the case where two call requests have been successfully received through path a and b respectively. For easy explanation, we introduce a different notation for average processing delay at each node. Let $1/\mu_{a_i}, 1 \leq a_i \leq \ell_a$, be the average processing delays for those nodes on path a. Similarly, Let $1/\mu_{b_j}, 1 \leq b_j \leq \ell_b$, be the average processing delays for those nodes on path b. Let Dp_a and Dp_b be the propagation delay of path a and b respectively. Assume $Dp_a \leq Dp_b$. Then the probability that the call request received through path b is a duplicated request (i.e., it is received by the destination node later than the call request through path a) given that the call requests sent through both paths are all successfully received is given by:

$$P_{dup}(b) \quad = \quad 1 - \sum_{i=1}^{\ell_a} \sum_{j=1}^{\ell_b} \frac{\mu_{b_j} A_i B_j}{\mu_{a_i} + \mu_{b_j}} e^{\mu_{a_i}(Dp_a - Dp_b)},$$

where

$$A_i \quad = \quad \frac{\prod_{j \neq i} \mu_{a_j}}{\prod_{j \neq i} (\mu_{a_j} - \mu_{a_i})} \quad i = 1, ..., \ell_a,$$

$$B_i \quad = \quad \frac{\prod_{j \neq i} \mu_{b_j}}{\prod_{j \neq i} (\mu_{b_j} - \mu_{b_i})} \quad i = 1, ..., \ell_b.$$

The probability that the call request through path a is duplicated can easily derived by:

$$P_{dup}(a) \quad = \quad 1 - P_{dup}(b).$$

Now consider the case where three call requests have been successfully received through path a, b and c. Adapt similar notation as above and assume that $Dp_a \leq Dp_b \leq Dp_c$. Then $P_{dup}(a), P_{dup}(b)$ and $P_{dup}(c)$ can be computed as following:

$$\begin{aligned} P_{dup}(a) \quad = \quad & \sum_{i=1}^{\ell_a} A_i e^{\mu_{a_i}(Dp_a - Dp_b)} - \sum_{i=1}^{\ell_a} \sum_{j=1}^{\ell_b} \frac{\mu_{a_i} A_i B_j}{\mu_{a_i} + \mu_{b_j}} e^{\mu_{a_i}(Dp_a - Dp_b)} \\ & + \sum_{i=1}^{\ell_a} \sum_{j=1}^{\ell_b} \frac{\mu_{a_i} A_i B_j}{\mu_{a_i} + \mu_{b_j}} e^{\mu_{a_i}(Dp_a - Dp_c) + \mu_{b_j}(Dp_b - Dp_c)} \\ & - \sum_{i=1}^{\ell_a} \sum_{j=1}^{\ell_b} \sum_{k=1}^{\ell_c} \frac{\mu_{a_i} A_i B_j C_k}{\mu_{a_i} + \mu_{b_j} + \mu_{c_k}} e^{\mu_{a_i}(Dp_a - Dp_c) + \mu_{b_j}(Dp_b - Dp_c)}, \end{aligned} \tag{63}$$

$$
\begin{aligned}
P_{dup}(b) &= 1-\sum_{i=1}^{\ell_a}\sum_{j=1}^{\ell_b}\frac{\mu_{b_j}A_iB_j}{\mu_{a_i}+\mu_{b_j}}e^{\mu_{a_i}(Dp_a-Dp_b)} \\
&\quad+\sum_{i=1}^{\ell_a}\sum_{j=1}^{\ell_b}\frac{\mu_{b_j}A_iB_j}{\mu_{a_i}+\mu_{b_j}}e^{\mu_{a_i}(Dp_a-Dp_c)+\mu_{b_j}(Dp_b-Dp_c)} \\
&\quad-\sum_{i=1}^{\ell_a}\sum_{j=1}^{\ell_b}\sum_{k=1}^{\ell_c}\frac{\mu_{b_j}A_iB_jC_k}{\mu_{a_i}+\mu_{b_j}+\mu_{c_k}}e^{\mu_{a_i}(Dp_a-Dp_c)+\mu_{b_j}(Dp_b-Dp_c)}, \\
P_{dup}(c) &= 1-\sum_{i=1}^{\ell_a}\sum_{j=1}^{\ell_b}\sum_{k=1}^{\ell_c}\frac{\mu_{c_k}A_iB_jC_k}{\mu_{a_i}+\mu_{b_j}+\mu_{c_k}}e^{\mu_{a_i}(Dp_a-Dp_c)+\mu_{b_j}(Dp_b-Dp_c)}.
\end{aligned}
$$

B.2.3 Computation of $\bar{T}_{setup}$

When more than one call request are successfully received, we need to compute the time for the first call request received. Again, let us only consider the cases where there are two or three call requests that have been successfully received. Let us first consider the case where two call requests through path a and b have been successfully received. Let T_a and T_b be the total waiting time along path a and b. Let the random variable $T_{min}^{a,b}$ be the time for the source node to receive the acknowledgment of the first call request. Then $T_{min}^{a,b}$ is the minimum of T_a and T_b. The CDF and PDF of $T_{min}^{a,b}$ is given by:

$$
F_{T_{min}^{a,b}}(t) = \begin{cases} 1-\sum_{i=1}^{\ell_a}A_ie^{-\mu_{a_i}(t-Dp_a)} & Dp_b \geq t \geq Dp_a, \\ 1-\sum_{i=1}^{\ell_a}\sum_{j=1}^{\ell_b}A_iB_je^{\mu_{a_i}Dp_a+\mu_{b_j}Dp_b}e^{-(\mu_{a_i}+\mu_{b_j})t} & t \geq Dp_b, \end{cases}
$$

$$
f_{T_{min}^{a,b}}(t) = \begin{cases} \sum_{i=1}^{\ell_a}\mu_{a_i}A_ie^{-\mu_{a_i}(t-Dp_a)} & Dp_b \geq t \geq Dp_a, \\ \sum_{i=1}^{\ell_a}\sum_{j=1}^{\ell_b}(\mu_{a_i}+\mu_{b_j})A_iB_je^{\mu_{a_i}Dp_a+\mu_{b_j}Dp_b}e^{-(\mu_{a_i}+\mu_{b_j})t} & t \geq Dp_b. \end{cases}
$$

The expectation of $T_{min}^{a,b}$ is then given by:

$$
\bar{T}_{min}^{a,b} = \sum_{i=1}^{\ell_a}A_i(Dp_a+\frac{1-e^{\mu_{a_i}(Dp_a-Dp_b)}}{\mu_{a_i}})+\sum_{i=1}^{\ell_a}\sum_{j=1}^{\ell_b}\frac{A_iB_j}{\mu_{a_i}+\mu_{b_j}}e^{\mu_{a_i}(Dp_a-Dp_b)}.
$$

Similarly, for the case where three call requests have been successfully received, we can derive the $\bar{T}^{a,b,c}_{min}$ as follows:

$$\begin{aligned}
\bar{T}^{a,b,c}_{min} &= \sum_{i=1}^{\ell_a} A_i(Dp_a + \frac{1 - e^{\mu_{a_i}(Dp_a - Dp_b)}}{\mu_{a_i}}) + \\
&\sum_{i=1}^{\ell_a}\sum_{j=1}^{\ell_b} \frac{A_i B_j}{\mu_{a_i} + \mu_{b_j}}(e^{\mu_{a_i}(Dp_a - Dp_b)} - e^{\mu_{a_i}(Dp_a - Dp_c) + \mu_{b_j}(Dp_b - Dp_c)}) + \\
&\sum_{i=1}^{\ell_a}\sum_{j=1}^{\ell_b}\sum_{k=1}^{\ell_c} \frac{A_i B_j C_k}{\mu_{a_i} + \mu_{b_j} + \mu_{c_k}} e^{\mu_{a_i}(Dp_a - Dp_c) + \mu_{b_j}(Dp_b - Dp_c)}.
\end{aligned}$$