[22] Guérineau, J.P., and Labetoulle, J., "End to end blocking in telephone networks: a new algorithm," *International Teletraffic Congress*, vol. 11, pp. 2.4.A.1:1–1.4.A.1:7, 1985.

[23] Delbrouck, L.E.N., "A unified approximate evaluation of congestion functions for smooth and peaky traffics," *IEEE Transactions on Communications*, vol. COM–29, pp. 85–91, 1981.

[24] Holtzman, J.M., "The accuracy of the equivalent random method with renewal inputs," *Bell System Technical Journal*, vol. 52, pp. 1673–1679, 1973.

[25] Wallström, B., and Reneby, L., "On individual losses in overflow systems," *International Teletraffic Congress*, vol. 9, pp. wallstrom-1–wallstrom-8, 1979.

[26] Kuczura, A., "Loss systems with mixed renewal and Poisson inputs," *Operations Research*, vol. 21, pp. 787–795, 1972.

[27] Riordan, J., *Stochastic Service Systems*, Wiley, New York, 1962.

[28] Manfield, D.R., and Downs, T., "Decomposition of traffic in loss systems with renewal input," *IEEE Transactions on Communications*, vol. COM–25, pp. 44–58, 1977.

[29] Butto, M., Colombo, G., and Tonietti, A., "On point to point losses in communication networks," *International Teletraffic Congress*, vol. 8, 1976.

[30] Deschamps, P.J., "Analytic approximation of blocking probabilities in circuit switched communication networks," *IEEE Transactions on Communications*, vol. COM–27, pp. 603–606, 1979.

[31] Delbrouck, L.E.N., "Use of Kosten's system in the provisioning of alternate trunk groups carrying heterogeneous traffic," *IEEE Transactions on Communications*, vol. COM-31, pp. 741–749, 1983.

[32] Akimaru, H., and Takahashi, H., "An approximate formula for individual call losses in overflow systems," *IEEE Transactions on Communications*, vol. COM-31, pp. 808–811, 1983.

[33] Girard, A., "Blocking probability of noninteger groups with trunk reservation," *IEEE Transactions on Communications*, vol. COM–33, pp. 113–120, 1985.

CHAPTER 4

Link-Decomposition Method

Performance measures for circuit-switched networks can take many forms, depending on the time scales considered and the ultimate objective of the planner. The most common — and probably most important — measure of the quality of a network is its ability to carry calls under normal circumstances. This qualitative measure is captured by the notion of total or average (over all origin-destination pairs) loss probability of the network. This global performance index, used to compare networks or routing algorithms, is also the main constraint in dimensioning.

Because it averages all the customers, the average blocking index gives no indication of the individual grade of services actually seen by customers. These grades of service are represented by $L^{i,j}$, the probability that a first-offered call presenting itself at node i, with node j as its destination, cannot be carried by the network. The grade of service can be expressed by the $L^{i,j}$s themselves or by the average

$$\overline{L} = \frac{\sum L^{i,j} A^{i,j}}{\sum A^{i,j}}. \tag{4.1}$$

These objectives are adequate for long-term planning, where decisions are made on an annual basis or longer, and where we can assume that the network parameters, such as the offered traffic, are stationary. In the shorter term, however, networks are not stationary. Traffic demands exhibit seasonal, daily, even hourly variations. This kind of consideration leads to other measures of the grade of service, such as the robustness of the network under traffic fluctuations — in particular, under generalized and focused overload. Similarly, network components are subject to failure. In this context, another measure of quality is the robustness of the network under failures, measured as the fraction of the nominal traffic that can be carried under specified failures. These questions are not specific to circuit-switched networks, but arise for all kinds of communication networks and indeed in other contexts. Because these problems are generally dealt with in the circuit-routing stage of network planning, they will not be discussed here any further.

We limit ourselves to network performance viewed in the context of dimensioning and routing optimization. In this case, the following assumptions generally hold:

1. External calls are represented by independent stationary Poisson processes.
2. Call-holding times are exponentially distributed.
3. Blocked calls have zero holding times.
4. Calls that cannot be routed in the network are cleared and do not return.
5. There is no blocking in switches.
6. The network topology is stable, and does not change during the calculation.

There are a number of known parameters from which to compute the loss probability. The matrix of the rates of the Poisson processes that describe the arrival of new calls to the network, called the *traffic matrix*, is denoted $A^{i,j}$. We also know the vector N_s, the number of circuits on all the links of the network. Finally, the routing policy used in the network is given, denoted by the symbolic value α whenever it is necessary to express some dependence on this parameter.

The full state-space model is far too complex to allow performance evaluation except in the case of very small networks. The difficulty lies in the fact that decisions taken at one point affect components located far away in the network, and that these interactions must somehow be accounted for. On the other hand, we generally have quite accurate models for the individual links, provided that the link parameters, especially the link-offered traffic, are known. The link-decomposition method is an approximation technique whereby the overall network problem is decomposed into a set of independent link-evaluation problems. The performance of each link is easily computed; under suitable assumptions, the overall network performance can be recovered from the individual link-performance measures. The only difficulty with the method arises from the fact that the exact values of the coupling coefficients — the link-offered traffic — are generally unknown. In practice, these values are approximated by a relaxation method, leading in most cases to an iterative procedure for the blocking evaluation.

Conceptually, the method contains two distinct stages:

1. Given the link-blocking probabilities, a computation of the end-to-end blocking for all traffic flows. The emphasis is on the combinatorial aspect of the network operation, relying heavily on the route-tree representation

for the routing. Computational methods are similar to those used to compute the reliability of a network whose links are subject to random failures.

2. An evaluation of the correct link-offered traffic, which in turn yields, via the link-blocking model, the correct values for the link-blocking probability. These values are the coupling coefficients of the system just mentioned. Here, the emphasis is on representing the traffic in terms of flows in the influence graph.

We first discuss how to calculate the end-to-end blocking for load sharing. This routing, though simple, exhibits some of the difficulties of all the other routing methods. We introduce the notion of the Erlang map and of its fixed points, proving some simple results of these equations. Then we extend the discussion to the case of fixed alternate routing. We discuss the combinatorial aspect of the blocking calculation in a route tree, showing how the Erlang fixed point also appears in the context of nonhierarchical routing. After a short discussion of the special case of hierarchical routing, we show that the Erlang map does not necessarily have a unique solution. We give a plausible explanation for this surprising result and indicate how to avoid this difficulty using state protection.

4.1 Load Sharing

After the trivial case of routing only on a direct link, load sharing is the simplest routing method. Load sharing makes a good starting point in discussing performance evaluation; we use it to introduce some basic concepts that will reappear in the remainder of this chapter and others, also providing some interesting theoretical results. Most of the discussion here is taken from [1], but is somewhat simplified; the notation has been made consistent with our conventions.

Load sharing normally operates by separating a given traffic stream corresponding to an origin-destination pair (o, d) into a number of substreams, each offered to a single path between o and d. To simplify notation, we assume that this separation has already been made and that the input parameters are the substreams obtained after the separation process instead of the original traffic matrix $A^{i,j}$. In this way, we can consider a set of streams A^l, each offered to a single path l. The calculation is simplified in the case of load sharing since the event that corresponds to a call loss is identical to the event that the path to which this call is offered is blocked. As we shall see when considering complex methods with alternate routing, this feature of load sharing greatly simplifies the calculation.

We denote the arc-path incidence matrix of the network by $\mathcal{I}_{s,l}$, where a matrix element $a_{s,l}$ is 1 or 0, depending whether link s is in path l or not. Also, let $\mathcal{P}_l$ denote the probability that path l is blocked. The problem is then to compute the value of $\mathcal{P}_l(A, N)$ for all values of l; from these values, the total loss probability is easily obtained by

$$\overline{L} = \frac{\sum_l \mathcal{P}_l A^l}{\sum_l A^l}. \tag{4.2}$$

Because we have good probabilistic models for the individual links, we would like to relate the calculation of the path blocking to the value of the link-blocking probabilities. There are two difficulties with this decomposition. The first is due to the fact that a path may have many links in tandem; the second, to the fact that different paths may intersect and therefore cannot be considered independent.

Since the loss probability cannot be calculated with the Markov model, we must consider what available models might be used to evaluate the path blocking. One model that comes to mind is the Erlang B function since it is so easily computed. Note, however, that the Erlang B function applies to a link in isolation: How can it be related to the calculation of $\mathcal{P}_l$? Now, the Erlang B function has two parameters: (1) the traffic offered to the link and (2) the number of circuits. The second parameter, which has a direct correspondence in the network, is simply the number of circuits in the group. The first parameter, however, appears nowhere in the list of parameters used to define the problem. It must therefore be computed from these values — a computation that constitutes the major difficulty encountered in evaluating $\mathcal{P}_l$.

Assuming that we can find a reasonable definition for a_s, the traffic offered to link s, we could compute a link parameter B_s, called the *link-blocking probability*, from the expression $B_s = E(a_s, N_s)$. The question is then how to relate this link-blocking probability to the path-blocking probability. This is where we start to make simplifying assumptions. We assume that, for a given path l, the events "link $s \in l$ is blocked" are all independent. Under this assumption, we then have the simple expression

$$\mathcal{P}_l = 1 - \prod_{s \in l}(1 - B_s). \tag{4.3}$$

The interpretation of this formula is quite simple. A call can be connected through the path if *all* links on the path are simultaneously free. The probability that a given link s is free is simply $(1 - B_s)$; because of the independence assumption, the probability of the combined event is the product of the individual probabilities. The path-blocking probability is then simply the complement of the probability of connecting the call.

The Erlang Fixed-Point Equation

In the preceding discussion, we reduced the problem of evaluating $\mathcal{P}_l$ to the definition and computation of the traffic offered to each link in the network. The correct definition of the offered traffic can be obtained from the results of Chapter 3. For each call request on the link, we set up a fictitious call in an infinite group. The offered traffic is the process that represents the number of busy servers in the fictitious group. Although theoretically correct, this definition is not very useful since it requires the arrival process at each link to be calculated. We would prefer to use some flow model to represent traffic, even though some ambiguity can arise in defining the flow representation.

For the sake of this discussion, let us consider a path l, numbering the links on the path sequentially in the direction from the origin to the destination. Thus link 1 is the first link encountered by the call, and so on until the last link, say z. One way to represent the flow is to assume that A^l is a flow that becomes thinned by a factor $(1 - B_s)$ as it proceeds on the path. Using this model, the traffic offered to the i^{th} link on path l is given by

$$a_i^l = A^l \prod_{s=1}^{i-1} (1 - B_s), \tag{4.4}$$

and the traffic that finally comes out of the path is given by

$$\overline{A}^l = A^l \prod_{s=1}^{z} (1 - B_s).$$

Although quite intuitive, this model suffers from a serious defect. Because of the thinning process at each link, the traffic of type l carried on each link is given by $\overline{a}_s^l = a_s^l(1 - B_s)$; it is *different* for each link. This, however, is inconsistent with the operation of circuit switching, where the number of calls of a given type carried on all the links of a path must be equal at all times — not the case for our flow model.

We can obtain a different definition of the traffic offered to a link by starting from the condition that the traffic of a given type carried on a path should be carried on all links of that path. Under the independence assumption, the path blocking is given by Eq. (4.3). We now impose the conditions that, first, $\overline{A}^l = (1 - \mathcal{P}_l)A^l$, and that, second, for each link i in the path, we must have $\overline{a}_i^l = \overline{A}^l$. Using the relation between offered and carried traffic on a link $\overline{a}_i^l = (1 - B_i)a_i^l$, we get the so-called reduced-load model:

$$a_s^l = A^l \prod_{i=1}^{z} (1 - B_i)/(1 - B_s). \tag{4.5}$$

Note the crucial difference between the definitions (4.4) and (4.5). In Eq. (4.4), the product runs from 1 to $i - 1$. In other words, the traffic offered to link i

depends on the blocking probability of the links that *precede* i in the path. This is no longer the case with definition (4.5), where the product runs from 1 to z: The traffic offered to link i depends on the blocking probability of links that precede and *follow* it on the path.

This result is often viewed as an indication that the definition of the offered traffic given by Eq. (4.5) should be rejected, since it does not coincide with our intuition of how a flow model should behave. In fact, Eq. (4.5) represents the real behavior of the path better than does Eq. (4.4). To see why the traffic offered to a link should depend on the blocking on links that appear after it on a path, consider once again the Erlang B function. The derivation of this function rests on two assumptions: (1) that the arrival process is Poisson and (2) that the holding times are exponentially distributed. There is, however, another hidden assumption in the model: A call that arrives and finds a free server always enters service immediately. This assumption is no longer true when we examine a link in a path. A call request that arrives at a link and finds a free server does not immediately produce a new call in service. There is a nonzero probability that this call request will not begin service even though there is a free server on the link; this probability is precisely the probability that the rest of the path after link i is blocked. In other words, the correct model to analyze a link in a path is not the $M/M/N$ queue, but the $M/M/N$ queue with balking. Now, if we assume that the balking probability β is independent of the state of link i, it is not hard to show that β is just the product of the blocking probabilities on the remaining path after link i, and that the correct expression for blocking on link i is given by the Erlang B function, but with the value of traffic given by Eq. (4.5), not by Eq. (4.4). This in turn explains why Eq. (4.5) is generally more accurate than Eq. (4.4).

Given Eq. (4.5) for the calculation of the traffic of type l offered to link s, we can compute the total traffic offered to this link simply by adding up the carried traffic of all paths that go through link s. Using the arc-path incidence matrix $\mathcal{I}$, this is simply expressed by

$$\begin{aligned}\overline{a}_s &= \sum_l \mathcal{I}_{s,l}\overline{A}^l \\ &= \sum_l \mathcal{I}_{s,l} A^l \prod_t (1-B_t)^{\mathcal{I}_{t,l}},\end{aligned}$$

and the offered traffic by

$$a_s = \sum_l \mathcal{I}_{s,l} A^l \prod_t (1-B_t)^{\mathcal{I}_{t,l}} / (1-B_s). \tag{4.6}$$

We now begin to see how the traffic flows are tightly coupled by the routing. The traffic offered to a link in the network depends directly on the blocking probability of all links on all paths that use this link. This means that there

can be some long-range effects such that a change in one part of the network may affect components located far away.

This dependence is quite pronounced. Now that we have an expression of the traffic offered to link s in terms of the link-blocking probabilities, we must recall that the blocking probabilities are themselves given as functions of the traffic offered to the link, in the present case by the Erlang relation $B = E(a, N)$. Because a depends on the Bs of other links, as seen by Eq. (4.6), we are led to the unavoidable conclusion that the correct values for the link-blocking probabilities can be given only implicitly, as the solution of the set of coupled nonlinear equations:

$$B_s = E\left(\frac{\sum_l \mathcal{I}_{s,l} A^l \prod_l (1 - B_l)^{\mathcal{I}_{t,l}}}{(1 - B_s)}, N_s\right). \tag{4.7}$$

This set of equations is the first manifestation of a phenomenon that will reappear throughout our discussion of performance analysis of circuit-switched networks. In fact, this type of nonlinear behavior is probably the rule, and only under very exceptional conditions can the situation be avoided.

The set of equations in (4.7) is called the Erlang map and its solution the Erlang fixed point; its study is fundamentally important in analyzing circuit switching. Two aspects particularly merit attention, the first of which concerns the number of solutions of the system. While a linear system can only have zero, one, or an infinite number of solutions, a nonlinear system can have any number of solutions, depending on the parameters of the equations. We will show that, in the case of load sharing, the solution is fortunately unique. The second aspect, of course, is the choice of a numerical method to solve the system.

Unicity of the Erlang Fixed-Point Solution

An interesting theoretical result concerning load sharing is the proof that the Erlang fixed-point equation has a unique solution [1]. This proof is carried out by showing that a solution of the fixed-point equations also satisfies the Kuhn-Tucker conditions of a convex minimization problem, and consequently must be unique. First, we introduce the transformation $B_s = 1 - e^{-y_s}$ for link s. Note that this mapping is one-to-one and maps the interval $[0, 1]$ of probabilities onto the interval $[0, \infty]$. In other words, we can equally well refer to B_s or y_s since they can be converted uniquely into each other. We also define for a link the function $\bar{a}_s(y_s, N_s)$, which is the traffic carried on the link for the value y_s. Note that, for a fixed N_s, this function is monotone increasing in y; it can be computed as follows. Given y, compute $B = 1 - e^{-y}$. From this, solve the equation $B = E(a, N)$ to obtain the offered traffic a. From the value of a, compute $\bar{a} = a(1 - B)$.

Consider now the optimization problem:

$$\min_{\substack{y \\ \mathbf{y} \geq 0}} \sum_l A^l e^{-\sum_m y_m \mathcal{I}_{m,l}} + \sum_s \int_0^{y_s} \overline{a}(z, N_s) dz \tag{4.8}$$

The first term is a sum of convex functions. The integrand of the second function is monotone increasing in z, which means that the function defined by the integral has an increasing derivative: It is thus also convex. As a consequence, if the problem has a minimum, this minimum is unique. We can write the Kuhn-Tucker necessary conditions — and in this case they are also sufficient — for optimality; we get

$$\sum_l \mathcal{I}_{s,l} A^l e^{-\sum_t y_t \mathcal{I}_{t,l}} = \overline{a}_s(y_s, N_s). \tag{4.9}$$

We now show that a solution to the fixed-point equation (4.7) is also a solution to Eq. (4.9). Rewriting Eq.(4.7) in terms of $\mathbf{y}$, we get

$$1 - e^{-y_s} = E\left(e^{y_s} \sum_l A^l \mathcal{I}_{s,l} e^{-\sum_t y_t \mathcal{I}_{t,l}}, N_s\right).$$

We know that the traffic offered to link s is given by

$$a_s = e^{y_s} \sum_l A^l \mathcal{I}_{s,l} e^{-\sum -t y_t \mathcal{I}_{t,l}}.$$

Replacing, we obtain

$$1 - E(a_s, N_s) = e^{-y_s}.$$

Multiplying both sides by a_s, we get

$$a_s \left[1 - E(a_s, N_s)\right] = a_s e^{-y_s}$$

or

$$\overline{a}_s(y_s, N_s) = \sum_l A^l \mathcal{I}_{s,l} e^{\sum_t y_t \mathcal{I}_{t,l}},$$

which shows that the solution of the fixed-point equation is also a global optimum of the minimization problem defined by Eq. (4.8). Because this optimum is unique, there can be no other solution to the fixed-point equation.

Numerical Solutions

Like most sets of nonlinear equations, Eq. (4.7) cannot be solved analytically, which means that numerical calculations are the only way to obtain solutions. Also, because these solutions will be used as part of other algorithms, such as

routing optimization and dimensioning, the efficiency of the selected technique is of paramount importance.

Solving nonlinear equations numerically is generally quite difficult. Rather than going into depth on this vast subject, we briefly review some of the more popular methods [2] for solving systems of nonlinear equations $F_i(\mathbf{x}) = 0$, indicating how they apply to the problem of finding a solution of the Erlang fixed-point equations.

Newton's Method. The first method that comes to mind to compute a solution of nonlinear equations is that of Newton [2]. This technique uses the first-order development of the vector $\mathbf{F}(\mathbf{x})$ near the current point $\mathbf{x}_0$ to compute a new estimate of the solution. Developing each component of the vector into a power series, we get

$$F_i(\mathbf{x}) = F_i(\mathbf{x}_0) + <(\mathbf{x} - \mathbf{x}_0), \nabla F_i(\mathbf{x}_0)>,$$

which can be written in matrix form as

$$\mathbf{F}(\mathbf{x}) = \mathbf{F}(\mathbf{x}_0) + (\mathbf{x} - \mathbf{x}_0)\mathbf{J}, \tag{4.10}$$

where $\mathbf{J}$, called the *Jacobian matrix* of the system, is given by $J_{i,j} = \partial F_i / \partial x_j$. Since we are looking for the solution of $F_i(\mathbf{x}) = 0$, we choose $\mathbf{x}$ as the next approximation for Eq. (4.10) when $F_i(\mathbf{x}) = 0$. We obtain

$$0 = \mathbf{F}(\mathbf{x}_0) + (\mathbf{x} - \mathbf{x}_0)\mathbf{J}$$

and

$$\mathbf{x} = -\mathbf{F}(\mathbf{x}_0)\mathbf{J}^{-1} + \mathbf{x}_0, \tag{4.11}$$

which becomes the new approximation of $\mathbf{x}_0$. The procedure is repeated; this time $\mathbf{x}$ becomes the new value $\mathbf{x}_0$. The main advantage of Newton's method is that it is known that, if $\mathbf{x}_0$ is close enough to the solution, the method has a quadratic convergence rate. This fast convergence has a price, however, since it requires the Jacobian matrix to be computed and inverted at each iteration, which can impose a heavy computational load.

In the present case, the system to be solved is

$$E_s(a_s(\mathbf{B}), N_s) = B_s.$$

The F functions are

$$F_s = E_s(a_s(\mathbf{B}), N_s) - B_s,$$

and the Jacobian matrix is given by

$$J_{s,t} = \frac{\partial E_s}{\partial a_s} \frac{\partial a_s}{\partial B_t} - \delta_{s,t}.$$

The first partial derivative, computed at fixed B_s, is a diagonal matrix. The coupling between the links comes from the second term, generally a full matrix,

which can be evaluated as follows. Rewrite Eq. (4.6) as

$$a_s = \sum_{l|s\in l} A^l \prod_{\substack{t\in l \\ t\neq s}} (1 - B_t)$$

and, taking the derivative with respect to B_r,

$$\frac{\partial a_s}{\partial B_r} = -\sum_{l|s\in l} A^l \prod_{\substack{t\in l \\ t\neq s \\ t\neq r}} (1 - B_t). \tag{4.12}$$

We have pointed out two elements of Newton's method that can lead to excessive computation. One is the requirement to invert the Jacobian matrix at each iteration. In fact, there exist algorithms in which this inversion is never done and that operate with successively more accurate representations of the inverse of the Jacobian matrix itself, rather than the matrix J. These methods, called *quasi-Newton*, are described in any standard textbook on numerical analysis or optimization, such as [3].

Even with these methods, derivatives must be calculated. For some idea of the amount of calculation required, let us assume that we are calculating the full Jacobian matrix. We see from Eq. (4.12) that, for a given component, it is necessary to go through all the paths, computing partial products of probabilities along these paths. Although we could probably organize the calculation to avoid duplication, this is still a heavy computational burden, tending to limit the use of Newton's method to small networks.

Also, we must point out that Newton's method, at least in its original form, suffers from another defect that prevents its use for the solution of the Erlang fixed point: We must have $0 \leq B_s \leq 1$ at all times during the calculation. We say that a solution that meets these requirements is *feasible*. Newton's method, however, has no built-in mechanism to allow such restrictions to be placed on the solution set. In other words, Newton's method is unconstrained, while the fixed-point equation requires a constrained solution. We could guarantee that the algorithm remain in the domain by a change of variable of the form $\sin^2 x_{s,t} = B_{s,t}$ and solve for x with an unconstrained method. In this case, we must be careful to avoid the artificial solutions introduced by the transformation, which generally requires looking at second-order information at the solution point. We could also arbitrarily prevent the solution from going out of the domain by setting any variable at the nearest bound if it goes outside the range $[0, 1]$. This, however, may have a detrimental effect on the convergence of the method, and is probably not a good way to proceed.

Minimization. A more natural way to obtain a feasible solution is by means of nonlinear minimization techniques. If we compute $\min z = \sum_i F_i(\mathbf{x})^2$, then the solution, if one exists, is found when $F_i(\mathbf{x}) = 0$. The advantage of this method arises from the existence of very efficient minimization techniques [4]

when the only constraints are bounds on the variables, which computationally are no more complex than unconstrained problems. Note that the computational requirements of these methods are largely determined by the calculation of the gradient of the objective function

$$\frac{\partial z}{\partial B_r} = \sum_i \frac{\partial F_i}{\partial B_r},$$

which is precisely the computation required for the Jacobian matrix in Newton's method. Thus, although the question of feasibility is handled more naturally in the context of minimization, the numerical efficiency is still too low to permit large-scale applications.

Relaxation. The method used most frequently, and the one with the smallest computational requirement, is the *relaxation method*, also known as *repeated substitution*. Using the fact that the system equations have the form $\mathbf{x} = \mathbf{F}(\mathbf{x})$, we replace the current value of $\mathbf{x}$ in F to obtain the new value. The advantage is that the gradients of F need not be computed, which is the reason this method is so efficient numerically. The relaxation method also ensures that the variables remain in the domain $0 \leq \mathbf{x} \leq 1$, which is not the case for the other unconstrained methods. On the negative side, however, is the fact that convergence is not guaranteed unless the eigenvalues of J are all less than 1 at the solution or unless we can show that F is a contraction mapping. In fact, it is not difficult to construct simple examples where the solution is known analytically but where the substitution method fails to find it (see Problem 4.12). These conditions cannot be verified unless we happen to know the solution, and the relaxation method could fail to converge even though there is a unique solution to the system. It would then be necessary to resort to another method, with a corresponding increase in computation time.

4.2 Computation of L^k as a Function of B_s for Alternate Routing

In the case of load sharing, the relationship between end-to-end loss probability and link-blocking probability is given by $L^k = \mathcal{P}^k$, since a blocking event on a path corresponds to a call loss. This is not the case in alternate routing, the first truly practical method we analyze. In alternate routing, the event that a path is blocked at the time a call attempts to use it *does not* imply a call loss. This relation is much more complex, depending on the structure of the route tree that describes the routing. In the first part of our work on alternate routing, therefore, we explore this relationship, as well as efficient numerical methods to calculate it.

The notion of calculating end-to-end probability from the given values of the link-blocking probabilities is fairly recent. The method is adapted from the techniques used in network reliability — in particular, from the work of Lee [5] on linear probability graphs. These techniques were originally developed to compute of the internal blocking of switches; its application to switched communication networks originated with Butto, Colombo, and Tonietti [6] in 1976. The problem of computing blocking for a given set of Bs is purely combinatorial since it depends only on the route trees and is independent of $A^{i,j}$ and $N_{s,t}$.

The algorithms proposed in the literature fall into three categories. The first method [6,7] consists of writing the exact formulas for the blocking, given the particular structure of the route trees considered. This, of course, severely limits the flexibility of the method since a new set of formulas must be derived whenever a new routing is considered. An important breakthrough occurred when Chan [8] pointed out that the calculation is recursive for a wide class of routings. This insight has led to a variety of algorithms, both for the general case and for important special cases. Independently of the work of Chan, another technique was proposed [9] based on tree-enumeration techniques. These last two methods are closely related; it is known that any recursive procedure can be given a tree-enumeration form. We now present examples of the three types of algorithms for a variety of special routing methods and for the most general case.

Two-Link Paths

We first study the simplest case, where overflow paths have a single tandem. Although there can be many overflow paths for any given origin-destination pair, the structure of the route trees permits a simple evaluation algorithm. This is because all the paths in the route of a given origin-destination pair are disjoint (see Fig. 4.1).

Because of the simple form of the trees, we can write analytic formulas giving the value of the loss probabilities $L^{i,j}$ as a function of the blocking on the links. In fact, the loss probability is simply given by $L = \prod_k \mathcal{P}^k$, where $\mathcal{P}^k$ is the blocking probability of the k^{th} path in the route tree. This can also be presented as a recursive procedure, which is easier to implement for numerical calculations. To simplify our presentation, we assume that the direct link (i, j) is always found in the first position in each route tree, although the more general case of an arbitrary first path poses no difficulty. We define P_k as the probability of an overflow to tandem k. P_k is also the probability that a call cannot be routed on any of the paths preceding the path through k.

For an OOC command, $L^{i,j}$ can be computed using the recursion formula and assuming that the link-blocking probabilities are independent. We define

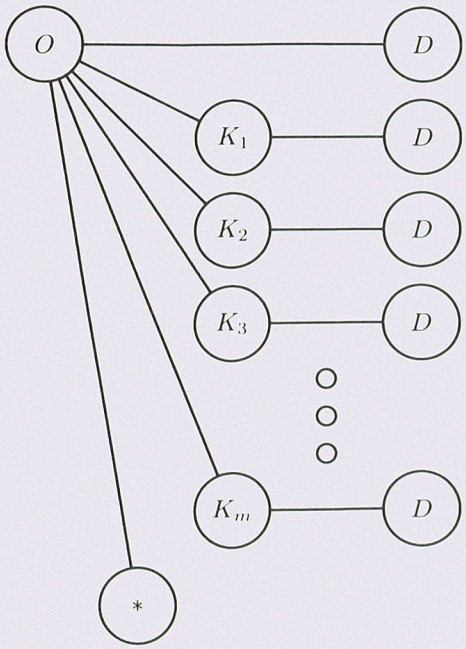

Figure 4.1 Two-Link Alternate Routing with OOC

P_1 as equal to B_{ij}, the probability of having to overflow on the first alternate path. For general k, we have

$$P_k = P_{k-1}\left[1 - (1 - B_{i,k-1})(1 - B_{k-1,j})\right] \tag{4.13}$$

In the case of m overflow paths, we find

$$L^{i,j} = P_{m+1}. \tag{4.14}$$

Similar formulas can be easily found for SOC control (see Problem 4.1).

Trees without Multiple Arcs

The two-link case is probably the most important practical case of nonhierarchical alternate routing; it is used almost exclusively in all currently proposed techniques. To gain some understanding of the inherent complexity of the blocking calculation when more than two links are permitted for some alternate paths, we now turn to more complex route trees. We consider three approaches to the calculation. First we express the procedure in terms of scanning

the nodes of the route tree. Then we exhibit, in a more general context, the enumerative aspect of the calculation, demonstrating its intrinsic exponential complexity. Finally we reformulate the procedure in recursive terms.

The first case is for route trees in which paths are no longer disjoint but can share some links. We can take advantage of a simple structure when the trees are such that a link appears no more than once in a given tree. This condition simplifies the calculation for the following reason. Suppose we are considering the probability that a call will be routed on a particular path. This involves the probabilities that the path is free and that the call overflows to the path. If some links on the path appear on some other path before the one considered, these two probabilities are not independent, even assuming that the individual link-blocking probabilities are independent. Thus conditional probabilities must be computed, with a worst-case complexity that grows very fast with the network size. One way to ensure that this does not occur is to assume that no arc appears more than once in a given route tree, in which case unconditional probabilities can be used.

Another reason to describe this model in detail is that it constitutes a good example of the use of tree-enumeration techniques in the blocking calculation. First we give a set of recursive equations that implicitly define the value of L^k. Although these equations could be solved directly by a recursive program, they are expressed in such a way that a straightforward tree-enumeration method can be used to compute a solution.

We distinguish two types of nodes in the tree: (1) *overflow control nodes* or, more simply, *control nodes*, of degree higher than 2, and (2) the others, called *ordinary nodes*. Control nodes are important because they represent the choices between overflow or loss at different points in the tree. The sequence of possible choices, given that previous choices have led to a control node, is contained in the subtree originating at the control node in question.

Define o and d as the origin and destination nodes, respectively, for the tree under consideration. Control nodes are labeled according to their depth in the tree. The depth of control node k is defined as the number of control nodes between k and o plus one. The origin has depth 0. If node i is a control node, and l the l^{th} subtree originating at i, we define

$D_{i,l}$ = The probability of overflow on l, given that one has reached i.

$D_{i,0} = 1$, $i = o, \ldots$ We arbitrarily define the probability of overflow on the first path on the subtree to be 1.

$C_{i,l}$ = The probability that the next control node will be reached on the l^{th} alternate path, given that node i has been reached.

Suppose we now choose a leaf in the tree, for example, node m; define P_m as the probability of reaching node m. This node m can be either the

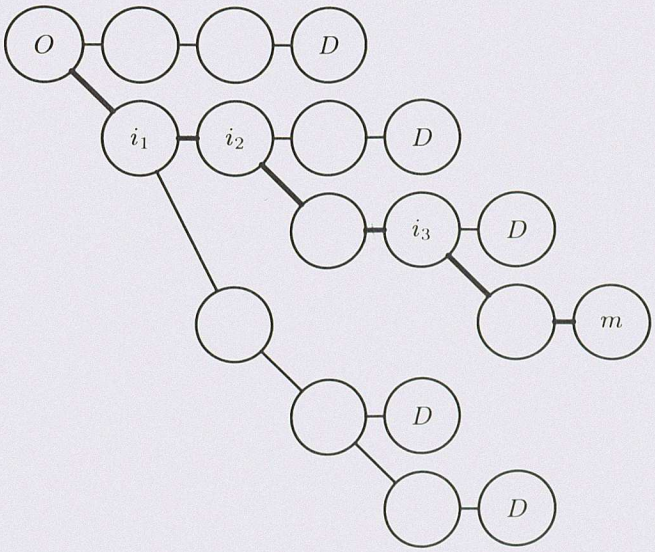

Figure 4.2 Route Tree with No Repeated Arc

destination node D or a loss node. We are interested in P_m because it represents the occurrence of a completed call or a lost call, depending on the nature of m; Either quantity is of interest. If the event of using path m represents a call completion, we can use its probability to compute link-offered traffic, as described in Section 4.3. If the event is a lost call, its probability is used in evaluating the end-to-end loss probability for the traffic stream corresponding to the tree being considered.

The selection of m defines a set of control nodes i_1, i_2,... from o to m and, at each control node, a corresponding overflow on levels l_1, l_2, ... (see Fig. 4.2). Since the probabilities of blocking on the network links are independent, we can write

$$P_m = D_{o,l_0} C_{o,l_0} D_{i_1,l_1} C_{i_1,l_1} \cdots C_{i_m,l_m}. \tag{4.15}$$

Because of the independence assumption, calculating the Cs poses no difficulty. Since the paths are disjoint, no link appears more than once in the tree at level k, and we can write

$$C_{i,l} = \prod_s (1 - B_s), \tag{4.16}$$

where s denotes all the links between the two consecutive control nodes i and $i+1$ on path l. Because arcs are not repeated, sequences of links between control

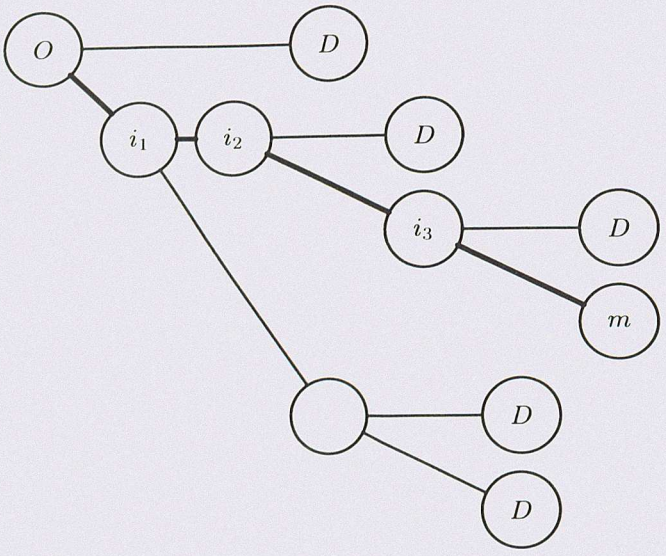

Figure 4.3 Equivalent Route Tree

nodes can be replaced by a single equivalent link, with blocking probability $C_{i,l}$. The equivalent route tree then takes the form shown in Fig. 4.3. From now on, we assume for simplicity that such a reduction has been made whenever possible.

The difficult part of the calculation is in the Ds. For this, we require additional variables:

$Q_{i,l}^{(k)}$ = The probability that we will reach any leaf by following path l out of node i, assuming that we have reached node i at level k.

$Q_i^{(k)}$ = The probability that we will reach any leaf, assuming that we have reached node i at level k.

$Q_i^{(k)}$, the connection probability from node i, summarizes all the events that may lead to a connection after having reached the node. It is obvious that

$$Q_i^{(k)} = \sum_{j=0}^{r} D_{i,j} Q_{i,j}^{(k)}, \tag{4.17}$$

where r is the number of subtrees at node i. A recursion formula can also be

given for the Qs:

$$Q_{i,l}^{(k)} = C_{i,l} Q_l^{(k+1)}, \tag{4.18}$$

where the node corresponding to $Q_l^{(k+1)}$ is the first control node encountered when we follow path l out of i and at level $k+1$.

Finally, we can express the Ds as functions of Qs by

$$D_{i,l} = \left(1 - Q_{i,l-1}^{(k)}\right) D_{i,l-1}. \tag{4.19}$$

The set of formulas (4.16–4.19) defines exactly the Cs and Ds, and thus the probabilities of reaching a leaf. Note that the recursion formula (4.19) uses only $Q_{i,l-1}^{(k)}$ and $D_{i,l-1}$ for the computation of $D_{i,l}$. It follows the sequence of overflows and can be computed in that order if the Qs from the lower level are known.

The recursions (4.18) and (4.17) proceed in the opposite direction from the sequence of overflows. Note, however, that only the $D_{i,l}$ from the preceding subtree are required to calculate Eq. (4.17); the Q_is from the immediate lower level are required for Eq. (4.18). Thus the complete computation can be done by enumerating the tree nodes once in each direction. This algorithm is efficient because one need go through each of the $N(N-1)$ route trees *only once*; all the required quantities can be computed in this single passage. This property is derived from the fact that any given link never appears more than once in a tree. As we shall see, the more general route trees require several iterations in each tree — or essentially the same thing, the storage of a quantity of information that increases exponentially with the size of the problem.

General Route Trees with OOC

The next most complex case is for originating office control, with the possibility that an arc may appear more than once in a given route tree. A complete analysis of the problem is given by Butto, Colombo, and Tonietti [6] in the form of equations clearly showing the exponential nature of the calculation. For this discussion, assume that we are considering a particular route tree k. To simplify the notation, the tree index is not used in the exposition. Under the independence assumption, the probability $\mathcal{P}_j$ that a path j in the tree is blocked is given by

$$\mathcal{P}_j = 1 - \prod_{s \in j} (1 - B_s).$$

Assume there are m paths altogether in the tree. For a tree with an arbitrary structure, but with OOC control, an event corresponding to the blocking on one path is not independent of an event corresponding to the blocking of another path. The solution of [6] is to consider all possible combination of states

(blocked or not blocked) on all the links that are shared between paths. These links are called *repeated* links since they would show up more than once in an enumeration of all the paths represented by the tree. The blocking probabilities on the paths can be expressed as conditional probabilities as follows.

Let $l = \{l_1, l_2, \ldots l_t\}$ denote the list of repeated links. To each repeated link i associate a state variable δ_i, which is 0 or 1 according to whether the link is free or busy. Let $\boldsymbol{\Delta} = \delta_i, i = 1 \ldots t$ be the state vector for the repeated links. Consider now the 2^t functions $L(\delta_1, \delta_2, \ldots \delta_t)$, defined as the conditional probability that the call cannot be routed through the tree, given the state vector $\boldsymbol{\Delta}$. The loss probability for the tree k can be written as

$$\begin{aligned} L^k &= (1 - B_{l_1}) \ldots (1 - B_{l_t}) L(0, 0, \ldots 0) \\ &+ B_{l_1}(1 - B_{l_2}) \ldots (1 - B_{l_t}) L(1, 0, \ldots 0) \\ &+ \ldots \\ &+ B_{l_1} B_{l_2} \ldots B_{l_t} L(1, 1, \ldots 1). \end{aligned} \tag{4.20}$$

As is clear from this expression, the number of terms to be considered is of the order 2^t, which has exponential worst-case growth. Having removed the repeated links from the tree, the conditional blocking probabilities are independent and are the product of individual path-blocking probabilities, with the repeated arcs removed. We get

$$L(\delta_1, \delta_2, \ldots \delta_t) = \prod_{j=1}^{m} \mathcal{P}_j(\boldsymbol{\Delta}), \tag{4.21}$$

where the blocking probability on path j is given by

$$\mathcal{P}_j(\boldsymbol{\Delta}) = [1 - \prod_{s \in j} (1 - B'_s)] \tag{4.22}$$

and

$$B'_s = \begin{cases} B_s & \text{if } s \notin l_i \\ 0 & \text{if } s \in l_i \text{ and } \delta_i = 0 \\ 1 & \text{if } s \in l_i \text{ and } \delta_i = 1 \end{cases}$$

and $s \in j$ indicates that link s is in path j. We also define for further reference the probability that all paths between and including paths i and j are blocked for a given tree and a given state vector $\boldsymbol{\Delta}$ as

$$\begin{aligned} C(i, j, \boldsymbol{\Delta}) &= \prod_{k=i}^{j} \mathcal{P}_k(\boldsymbol{\Delta}) \quad j \geq i, \\ &= 1 \quad i > j. \end{aligned} \tag{4.23}$$

We get

$$\begin{aligned} L(\boldsymbol{\Delta}) &= C(1, m, \boldsymbol{\Delta}) \\ L^k &= \sum_{\boldsymbol{\Delta}} L(\boldsymbol{\Delta}) \end{aligned} \tag{4.24}$$

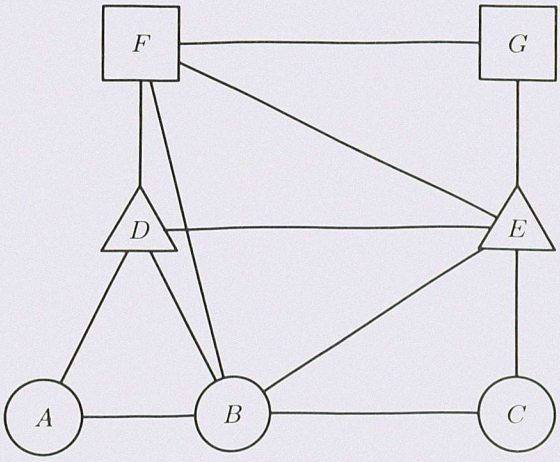

Figure 4.4 Seven-Node Hierarchical Network

where the last sum is to be understood in the sense of Eq. (4.20) over all state vectors $\boldsymbol{\Delta}$. This formulation does not take into account the recursive nature of the calculation, but clearly shows its exponential growth, since a complete enumeration of all the state vectors $\boldsymbol{\Delta}$ is required.

Manual calculations are reasonably easy to use as long as the route trees are not too complex, as can be seen from the seven-node network of Fig. 4.4. Considering the route tree for stream (A, C) shown on Fig. 4.5, we see that three links — 1, 3, and 4 — are shared among multiple paths. The state vector $\boldsymbol{\Delta}$ has three components, and there are eight terms to consider in the calculation (see Table 4.1).

Summing the eight terms, we obtain

$$L = B_2B_5\,[1 - (1 - B_6)(1 - B_7)]$$
$$(1 - B_1)(1 - B_3)(1 - B_4) + B_2(1 - B_1)(1 - B_3)B_4 + (1 - B_1)B_3 + B_1$$

General Route Trees with SOC

Another simplification that occurs for networks with SOC control was first noted by Butto, Colombo, and Tonietti [6]. They also perceived the recursive nature of the calculation, using a somewhat more complex notation than given here. In this case, although there may be arcs in common on many paths, it is unnecessary to compute conditional probabilities when calculating the

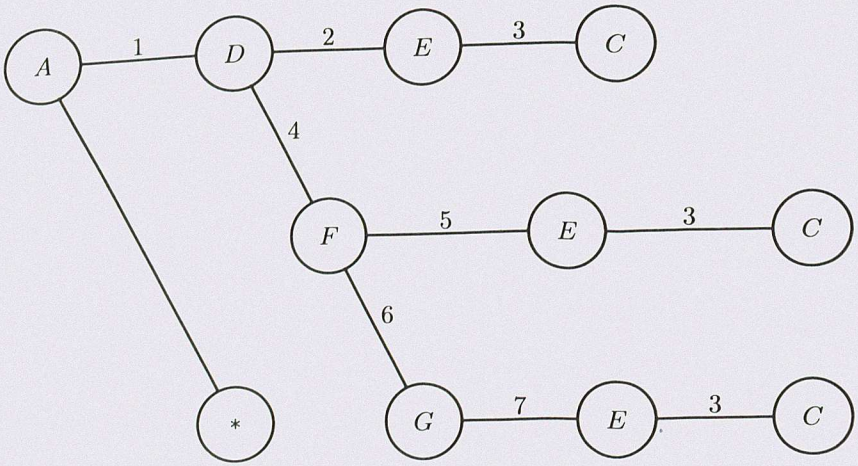

Figure 4.5 Route Tree for Stream (A, C) for Seven-Node Network

Link Number			P	Factor
1	3	4		
0	0	0	$B_2B_5\,[1-(1-B_6)(1-B_7)]$	$(1-B_1)(1-B_3)(1-B_4)$
0	0	1	B_2	$(1-B_1)(1-B_3)B_4$
0	1	0	1	$(1-B_1)B_3(1-B_4)$
0	1	1	1	$(1-B_1)B_3B_4$
1	0	0	1	$B_1(1-B_3)(1-B_4)$
1	0	1	1	$B_1(1-B_3)B_4$
1	1	0	1	$B_1B_3(1-B_4)$
1	1	1	1	$B_1B_3B_4$

Table 4.1 Calculation of End-to-End Blocking for the Seven-Node Network Using the Method of Butto, Colombo, and Tonietti [6]

probability of using path i: The probability of using a path is independent of the state of the arcs on this path that appear elsewhere in the route tree. To

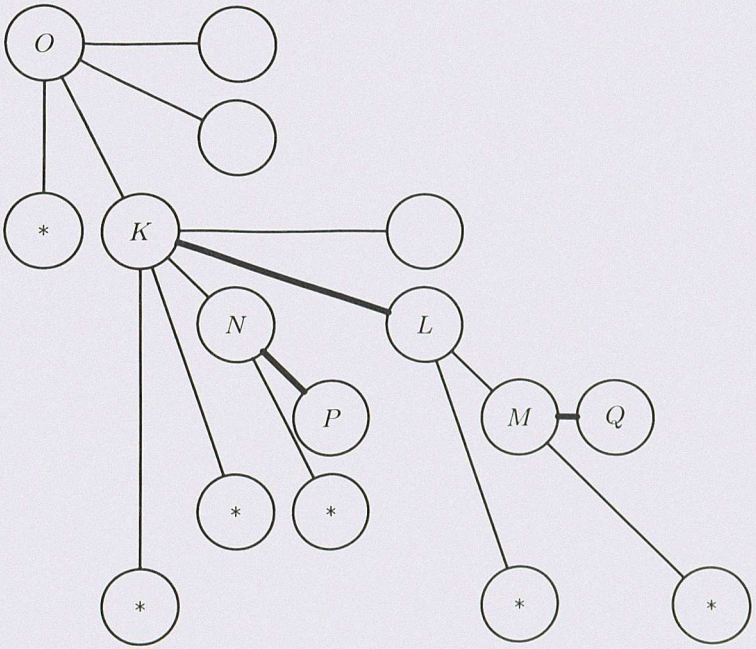

Figure 4.6 Route Tree for SOC Control. Link s Indicated by Thicker Line.

see this, consider the fragment of a route tree indicated in Fig. 4.6. Let path i be (O, K, N, P). Suppose that link s, indicated by a thicker line, appears elsewhere in the tree before path i. Note that the two ends of link s, which in reality correspond to the same nodes of the network, are labeled differently in the route tree in order to clarify the discussion.

Let (O, K) be the arc leading to path i from O, and (K, N) the arc leading from node K to path i. If s occurs somewhere in the tree before path i, this can happen in one of two ways. Either s is adjacent to one of the nodes on path i, such as the arc (K, L), or it is not adjacent to one of these path nodes, such as the arc (M, Q). In the first case, there is zero probability that we will use path i. In the second case, the probability of using path i is independent of the state of s before path i; this is so because if a call reaches node M, it will never be carried on path i. It either will be carried on the remainder of the path from M or will be lost because of the presence of the loss node adjacent to node M.

As an aside, route trees such as the one in Fig. 4.6, although perfectly correct from a theoretical point of view, are almost never found in practice (see Section 2.5). The reason is that the routing of a call that reaches s depends on the previous history of that call, that is, whether it came as an overflow from a link originating at i or whether it was partly carried on the route tree. Normally this kind of information for making routing decisions is not available to switches, and route trees of this kind are only of theoretical interest. This simplifying feature of full SOC networks is exploited in the algorithms proposed by Butto, Colombo, and Tonietti [6] in the form of equations, and, in a recursive form, by Gaudreau [10], who emphasizes its use for hierarchical networks.

The recursive nature of the calculation can be seen from the following argument. First note that, in a SOC network, all nodes are control nodes. Thus the level of a node is simply its distance from the origin, measured by the number of links encountered to reach the node. A calculation similar to that of Section 4.2 can be used to compute the end-to-end blocking probability. Equations (4.16–4.19) become

$$D_{i,l} = \prod_{j=1}^{l-1} B_{i,j} \tag{4.25}$$

$$C_{i,l} = (1 - B_{i,l}) \tag{4.26}$$

$$Q_i^{(k)} = \sum_{l=1}^{m} (1 - B_l) Q_l^{(k+1)} \prod_{s=1}^{l-1} B_s \tag{4.27}$$

which can be solved by a single passage over the route tree, by decreasing level.

Arbitrary Route Trees

Let us now consider the case of a completely general route tree for traffic k. Following the notation of [8], we introduce the notion of a *link set*. For a given route tree, link sets are subsets of links that are present in the tree. The actual composition of a link set is determined by the structure of the tree — in particular, by the relative position of links that appear more than once in the tree. The precise composition of link sets is given recursively in the algorithm; for now, let us think of them as fragments of paths in the tree.

Let U_i be defined as the i^{th} link set in some list. The link sets may be either complete paths or portions of paths. The link sets are ordered in the list according to the order of paths in the route tree. We want to compute $Pr(U_i)$, the probability that U_i is used. We consider two distinct cases, according to whether U_i is the first link set in the tree or not. If the link set is the first one in the list, we have

$$Pr(U_1) = \prod_{s \in U_i} (1 - B_s).$$

If the link set is one to which there is some overflow, we have

$$Pr\{U_i \text{ is used}\} = Pr\{U_i \text{ is available}\} \times \qquad (4.28)$$
$$Pr\{\text{None of } U_1U_2 \ldots U_{i-1} \text{ has been used } \mid U_i \text{ is available}\}$$

where the condition "U_i is available" means that all the links on link set i are free. The complexity of the problem stems from the fact that the "U_i is available" condition *must* be taken into account in evaluating Eq. (4.28), since the fact that link set U_i is free may affect the probability of overflow to i. If the links of U_i do not appear in $U_1U_2 \ldots U_{i-1}$, then the condition is no longer required, and we find the equations described in Section 4.2.

Define the link set

$$U_{k(i)} = U_k - U_i,$$

where U_k belongs to a path preceding U_i. $Pr\{U_{k(i)}\}$ is then the probability that all the links in the link set $U_{k(i)}$ are free. Eq. (4.28) is thus written as

$$\begin{aligned} Pr\{U_i \text{ is used}\} &= Pr\{U_i \text{ free}\} Pr\{U_{1(i)}U_{2(i)} \ldots U_{i-1(i)} \text{ all busy}\} \\ &= Pr\{U_i \text{ free}\} \left[1 - Pr\{\text{at least one of } U_{1(i)}U_{2(i)} \ldots U_{i-1(i)} \text{ free}\}\right] \end{aligned}$$

Define, for arbitrary link sets $U_1, U_2, \ldots U_i$,

$$Q(U_1 \ldots U_i) \triangleq Pr\{U_i \text{ is used}\}. \qquad (4.29)$$

This probability depends on the link sets preceding U_i. We can then define recursively

$$Q(U_1) = \prod_{s \in U_1} (1 - B_s) \qquad (4.30)$$

$$Q(U_1 \ldots U_i) = \prod_{s \in U_i} (1 - B_s) \times \left[1 - \sum_{k=1}^{i-1} Q(U_{1(i)}U_{2(i)} \ldots U_{k(i)})\right] \qquad (4.31)$$

We thus have a recurrent formation for the Qs that can be evaluated as the tree is enumerated. This calculation proceeds as follows for the route tree of Fig. 4.5:

$$Q(U_1) = (1 - B_1)(1 - B_2)(1 - B_3)$$

$$Q(U_1U_2) = (1 - B_1)(1 - B_4)(1 - B_5)(1 - B_3) \left[1 - \sum_{k=1}^{1} Q(U_{1(2)})\right]$$

Since $U_{1(2)}$ is simply link 2, the corresponding $Q(U_{1(2)} = (1 - B_2)$; replacing, we have

$$Q(U_1U_2) = (1 - B_1)(1 - B_4)(1 - B_5)(1 - B_3)B_2.$$

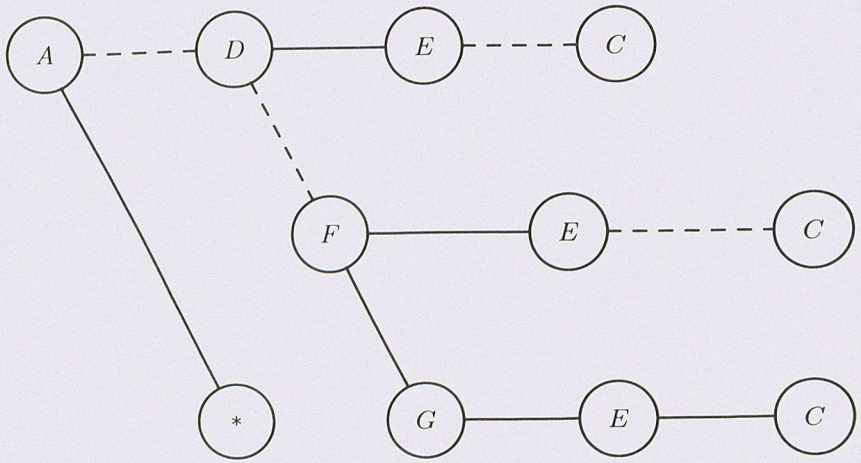

Figure 4.7 Partial Route Tree for Calculating the Probability of Connection on Link Set

The probability of connection on the third path is given by

$$Q(U_1U_2U_3) = (1-B_1)(1-B_4)(1-B_6)(1-B_7)(1-B_3) \times \left[1 - Q(U_{1(3)}) - Q(U_{1(3)}U_{2(3)})\right].$$

Here, too, the partial link sets take a simple form since $U_{1(3)}$ is link 2 and $U_{2(3)}$ is link 5. Under these conditions, we get

$$Q(U_{1(3)}) = 1 - B_2.$$

The second probability of connection on the link set can be calculated from the partial route tree shown in Fig. 4.7. In this tree, the links that do not contribute to the probability are shown with a dotted line; the others are solid. The probability of using link set $U_{1(3)}U_{2(3)}$ can be computed from this tree as usual, simply by dropping the dotted arcs from the calculation. We then get

$$Q(U_{1(3)}U_{2(3)}) = B_2(1-B_5),$$

and replacing, we get

$$Q(U_1U_2U_3) = (1-B_1)(1-B_4)(1-B_6)(1-B_7)(1-B_3) \times [1 - (1-B_2) - B_2(1-B_5)],$$

which is the remaining term in the calculation. Because we are considering OOC control, the only loss node is in the last position in the route tree; the

loss probability is given by

$$L = 1 - \sum_{k=1}^{3} Q(U_1 U_2 \ldots U_k).$$

This recurrence can be used to compute the probability of loss as well as the probability of connection, since loss nodes are treated like ordinary destination nodes in the route tree. The complexity cannot be polynomially bounded, however, because the number of conditional clauses on the probabilities can grow arbitrarily large for an arbitrary network and routing plan. Finally, note that this formulation covers all the static routing cases with alternate routing and therefore can be used to calculate the probability of end-to-end blocking for hierarchical routes such as those found in most telephone networks [10].

4.3 The Erlang Fixed-Point Equation for Alternate Routing

Section 4.2 describes how the $L^{i,j}$ can be computed, given the $B_{s,t}$. This purely combinatorial problem does not depend on the traffic matrix or the link capacities, but only on the structure of the route trees. Here we present the second part of the evaluation method: computation of the correct link-blocking probabilities. A general discussion of the problem, using a simple one-moment model, leads us back to the Erlang fixed-point equation; we investigate its properties. We show that the solution need not be unique, indicating how such undesirable behavior can be prevented through the use of state protection.

Ordered Networks

Before presenting the general case of alternate routing, let us discuss an important special case that shows that every alternate routing policy does not necessarily lead to a fixed-point system. First we introduce the notion of an *ordered network*, basing our discussion on the concept of an ordered routing (see Section 2.5). For such a routing, the sequence of overflow and carried calls occurs so that it is possible to find a partial ordering $\mu(i)$ between all the network links that is identical for all traffic streams. This notion is useful for calculating traffic in cases where this partial ordering is also valid for the link-offered traffic.

More specifically, we say that a network is ordered if the links can be ranked in such a way that the traffic offered to link s of rank $\mu(s)$ depends only on the traffic offered to links $t \in \mathcal{A}(s)$. This notion is similar to the notion of order for routing but is now applied to the traffic. A network is ordered only if certain conditions are met on the overflow of blocked calls, as well as

on trunks in tandem. First, the overflow must proceed according to the partial order determined by the routing, which is itself ordered. This condition is not sufficient, however, because the traffic offered to the first link of a path composed of many links in tandem depends on the blocking on the downstream links. In addition to the condition on overflow, the network is ordered if either the downstream links precede the link in the routing — which is a contradiction — or the blocking on these downstream links is small enough that it can safely be neglected when calculating the traffic offered to the link. This distinction is important because the order assumption is commonly made for hierarchical networks, but without reference to links in tandem. Fortunately, for hierarchical networks operating at their design grade of service, the blocking probability is always small on the hierarchical route leading from any given node to the destination. Hence it is valid to assume that the balking effect can be neglected; we can safely assume that these networks are ordered. This assumption, however, is *not* valid if the network is operated at high blocking on the finals. It is also not valid for nonhierarchical networks, where there is not necessarily a low-blocking path from a given node to the destination. In these cases, although it is possible to have an ordered routing, as in the case of the hierarchical network, the network itself is not ordered.

Calculation of Offered Traffic for Ordered Networks

In cases where the network *is* ordered, the solution procedure is not iterative. To see this, consider the influence graph corresponding to a given ordered network. Link blocking is calculated in a single iteration over all nodes of the influence graph, in increasing node order. Let k be the current value of the order:

1. For node i of order k, the value a_i^m of all parcels offered to it is known by the order assumption. Using the appropriate blocking model (e.g., the Erlang B function if the Poisson model is used for internal traffic), compute

$$\begin{aligned} a_i &= \sum_m a_i^m \\ B_i &= E(a_i, N_i) \end{aligned} \tag{4.32}$$

   and from this, $\hat{a}$, $\bar{a}$, and, if necessary, the other moments.

2. Update the traffic offered to the two nodes out of i:

$$\begin{aligned} a_{\rho(i,m)} &\leftarrow a_{\rho(i,m)} + \bar{a}_i^m \\ a_{\sigma(i,m)} &\leftarrow a_{\sigma(i,m)} + \hat{a}_i^m \end{aligned}$$

   A call is lost when $\sigma(i, m) = 0$.

3. Repeat for all parcels of node i.

4. Repeat for all nodes at level k.

This method is widely used for dimensioning networks in almost all telephone administrations. Let us mention some of its more important features.

First note that the method consists of finding a multicommodity flow in the influence graph, using some peculiar routing rules. Consider for simplicity the case where all traffics are Poisson. Each node has a parameter N_s that determines how the flow entering the node will be split between the overflow and the carried arcs. We emphasize again that, because of the nature of alternate routing, this separation cannot be assigned arbitrarily, but is a function of the entering traffic and N_s. In this simple model, no other information is used to effect this separation, although we will see other cases where this is not so.

Second, consider Eq. (4.32). It is very similar to Eq. (4.7) and also to Eq. (4.39). Eq. (4.32), however, has a simplifying feature that is not present in the other two equations: The total offered traffic a_i depends only on the traffic offered to the links $j \mid \mu(j) < \mu(i)$. The consequences of this fact are explored in more depth in Problem 4.13.

Finally, the end-to-end blocking and the link-blocking probabilities are computed "on the fly." No use is made of route trees, and the lost traffic is simply the sum of traffic parcels that have nowhere to go at a given stage of the calculation. This technique is often sufficient for hierarchical networks, particularly in the context of dimensioning, where only the link-blocking probabilities are of interest (see Chapter 8 for a discussion of dimensioning hierarchical networks).

This method of progressively moving through the influence graph in increasing order of level is closely related to the SOC mode of overflow control used in telephone networks. Other overflow control rules cannot easily be modeled using the influence graph, usually requiring the use of the route tree.

Calculation of Offered Traffic for Two-Link Routing

First note that given the route trees and the B_ss, the link-offered traffics a_s can be computed — an operation called *link loading*. In general, these traffics are functions of *all* the B_ss and can be written $a_s(\mathbf{B})$ to emphasize this dependence. The calculation of the link-offered traffic depends on the structure of the route trees for each traffic stream. We show how this can be done in the case of two-link alternate paths with OOC control, following the technique of Lin, Leon, and Stewart [7] in presenting the traffic calculation. We define

A_l^k = Traffic offered to the l^{th} path in the route tree for traffic k. This is the amount of new traffic that has been blocked on all paths preceding l in the tree.

$\overline{A}_l^k$ = Traffic carried on the l^{th} path in the route tree for traffic k.

$\mathcal{P}_l^k$ = Blocking probability of the l^{th} alternate path in the route tree for traffic k.

P_l^k = The probability that a call arriving at the origin node of stream k will overflow to level l.

The index l will also be called the *level* of the path in a particular tree.

The route trees are scanned in some arbitrary order. For route tree k, scan the tree in increasing order of l, the index of overflow paths. At each level l in the tree, assume that the two links are, respectively, s and t:

1. Using the independence assumption, compute the path blocking:

$$\mathcal{P}_l^k = 1 - (1 - B_s)(1 - B_t). \tag{4.33}$$

   Note that the link-loading procedure described here would also be valid if we used a different path model.

2. Compute the traffic carried on the path. The underlying assumption behind the path model is that this traffic is also carried on the two links in the path. Thus we have

$$\begin{aligned} \overline{A}_l^k &= A_l^k(1 - \mathcal{P}_l^k) \\ \overline{a}_x^k &= \overline{A}_l^k, \quad x = s, t \end{aligned} \tag{4.34}$$

3. The contribution of this route tree to the traffics offered to links s and t is given by

$$a_x^k = \frac{\overline{a}_x^k}{(1 - B_x)}, \quad x = s, t \tag{4.35}$$

   Calculating the offered traffic in this way takes into account the possibility that the call may be blocked downstream.

4. Compute the cumulative probability of overflow and the traffic offered to the next alternate path:

$$P_{l+1}^k = \mathcal{P}_l^k P_l^k \tag{4.36}$$

$$A_{l+1}^k = A_l^k P_l^k \tag{4.37}$$

This last step yields the current estimate for L^k when the last path of the route tree is processed. When all the route trees have been processed, we know

all the traffic parcels being offered to all the links; the total traffic offered to a link is given by

$$a_s = \sum_k a_s^k. \tag{4.38}$$

To summarize, we see that the traffic offered to a given link s is a function of the link-blocking probabilities B_t through Eqs. (4.33–4.35). To be consistent with the assumptions already made about the traffic — that is, that it is described by a Poisson process — we must also impose the condition that

$$B_s = E(a_s(\mathbf{B}), N_s), \tag{4.39}$$

where we emphasize the dependence of a_s on all the blocking probabilities by writing it as $a_s(\mathbf{B})$. The correct values of a_s (and, by the same token, of B_s) are given by the solution of the system (4.39). This, of course, is another manifestation of the Erlang fixed point; we must examine the two standard questions about this system: (1) the number of solutions and (2) how to solve the system efficiently. Note that the relation $a_s(\mathbf{B})$ is different from the equivalent relation obtained in the case of load sharing and thus that the answers obtained for that case will not necessarily apply here.

Solution Methods

The solution techniques are identical to those described in Section 4.1, with one addition: the calculation of the Jacobian matrix of the system for use with either Newton's method or one of the optimization techniques.

In this case, the system to be solved is

$$E_s(a_s(\mathbf{B}), N_s) = B_s.$$

Consequently, the functions are

$$F_s = E_s(a_s(\mathbf{B}), N_s) - B_s$$

and the Jacobian matrix is given by

$$J_{s,t} = \frac{\partial E_s}{\partial a_s}\frac{\partial a_s}{\partial B_t} - \delta_{s,t}.$$

The first partial derivative $\partial E/\partial a_s$, computed at fixed B_s, is a diagonal matrix. The coupling between the links comes from the second term, which generally is a full matrix. It can be evaluated as follows. Consider a particular route tree corresponding to a commodity k, and let m be the level of link s in the tree, j the level of some link t, and l a running index referring to levels in this tree. We adopt the convention that the other link in addition to s, in a path is denoted by s', and that all quantities indexed by s' take the expected value if s happens to be the direct link. For instance, if s is the direct link, the quantity

$(1 - B_{s'})$ is the probability of completing the call on the second link, which in this case does not exist. The convention means that in this case $(1 - B_{s'}) = 1$. With this notation, we get

$$\frac{\partial a_s}{\partial B_t} = \sum_{k|s\in k} A^k \frac{\partial}{\partial B_t} \prod_{l=1}^{m-1} \mathcal{P}_l^k(\mathbf{B})(1 - B_{s'}).$$

This last derivative is given by

$$\frac{\partial}{\partial B_t} \prod_{l=1}^{m-1} \mathcal{P}_l^k(\mathbf{B})(1 - B_{s'}) = \begin{cases} \prod_{l=1}^{m-1} \mathcal{P}_l^k(\mathbf{B}) & \text{if } t = s' \\ \prod_{\substack{l=1 \\ l\neq j}}^{m-1} \mathcal{P}_l^k(1 - B_{l'})(1 - B_{s'}) & \text{if } j < \text{level of } t \\ 0 & \text{otherwise} \end{cases} \tag{4.40}$$

Although not iterative, the calculation can be made by scanning the route trees that contain the link with respect to which the derivative is being computed. Nevertheless, there can be a large amount of computation, as was the case for load sharing, and for precisely the same reasons.

In addition to evaluating the performance of a given network, these gradients play a central role in many areas of routing optimization and dimensioning of circuit-switched networks. Algorithms are needed that permit the efficient calculation of these gradients for large networks. This subject has not received the attention it deserves and requires more research.

The relaxation method can also be used to solve the equations. In practice, this method is favored because it has the lowest computational requirements of the three techniques. The usual word of caution is in order here: There is no guarantee that the relaxation method will find a solution, even if one exists. Because the theoretical conditions for convergence cannot be tested until a solution is reached, the algorithm may fail to converge (although this possibility seems rare for the Poisson model used here).

4.4 Stability and State Protection

We have seen that computing the link-blocking probabilities involves the solution of a system of nonlinear equations (4.39). The existence of a solution to the nonlinear system (4.39) is generally not in doubt. Until recently, the question of uniqueness had not received a great deal of attention. All practical computation methods implicitly assume that there is only one solution to the model, not examining the possibility that others may exist. Let us now turn to the question of uniqueness, leading into the study of stability and state protection.

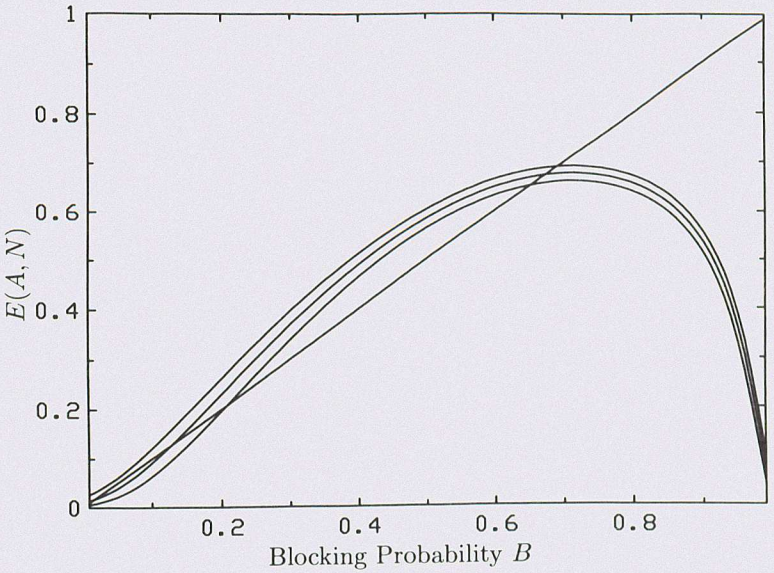

Figure 4.8 Multiple Solutions of the Erlang Fixed-Point Equation for Symmetric Networks; Traffic = 40, 42 and 44 Erlangs, 100 Trunks, 9 Alternate Routes

Nonlinear systems need not have a unique solution; as we will see, the flow equations for two-link nonhierarchical alternate routing do indeed have multiple solutions. These solutions, a consequence of mutual overflow and multiple alternate routing, can be viewed as a form of classical hysteresis. We will describe a state-protection technique that seems to prevent this behavior.

Multiple Solutions

First, let us exhibit networks in which the Erlang fixed-point equation has more than one solution. To do this, we simplify the general system of equations by introducing the notion of a symmetric network, defined by selecting $A_{i,j} = A$ and $N_{i,j} = N$. In addition, the route trees are such that the offered traffic is identical for all links. (Incidentally, the production of a truly symmetric alternate routing is not trivial, and requires some thought; here we say only that this can be done in general [11].)

Because of the symmetry, we can say that $B_s = B$, and similarly that $a_s = a$. Under these conditions, the system (4.39) reduces to a single nonlinear equation in B, which is much easier to study, and which can be plotted. We give

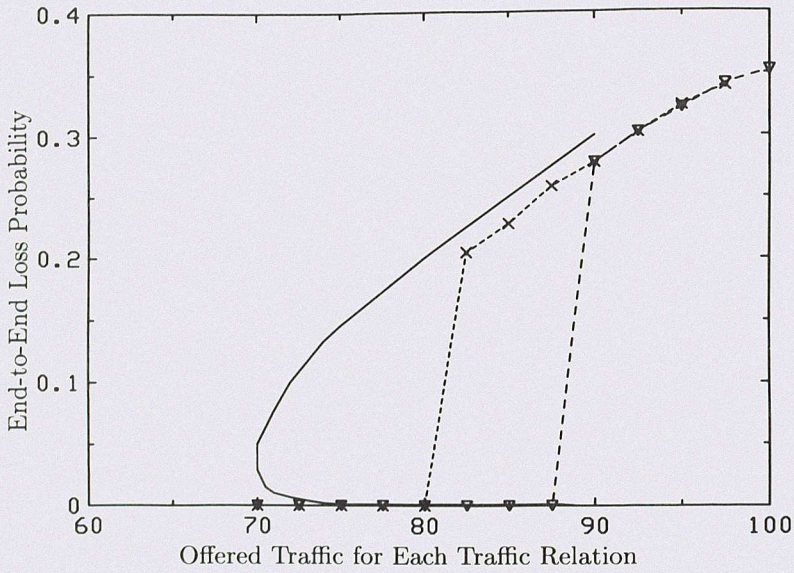

Figure 4.9 Multiple Values for End-to-End Blocking

an example in Fig. 4.8, where each side of Eq. (4.39) is traced as a function of a single variable B, the link-blocking probability, for a number of values of the offered traffic A. Solutions are given by the intersection of one of the curves with the diagonal; as the traffic changes, multiple solutions appear to the system, then disappearing. This phenomenon has also been reported in [12,13]. The net effect is that, for some values of the parameters, there can be three values of the end-to-end blocking probability, as shown on the solid curve of Fig. 4.9 (only two branches are visible on the plot because of the vertical scale).

This behavior is somewhat surprising since it contradicts the conclusion obtained from our analysis of the network as a Markov chain, which guarantees that there exists a unique set of stationary state probabilities and hence a unique end-to-end loss probability. The question is then how to reconcile this apparent contradiction.

The first possible explanation is that the apparent contradiction is a by-product of the rather severe assumptions that were made in the model — either the assumption of Poisson traffic used for calculating link-blocking probabilities or the independence assumption used for calculating the path-blocking probabilities. Unfortunately, as recent work [14] indicates, these multiple solutions arise in connection with real instabilities in networks operating with nonhierar-

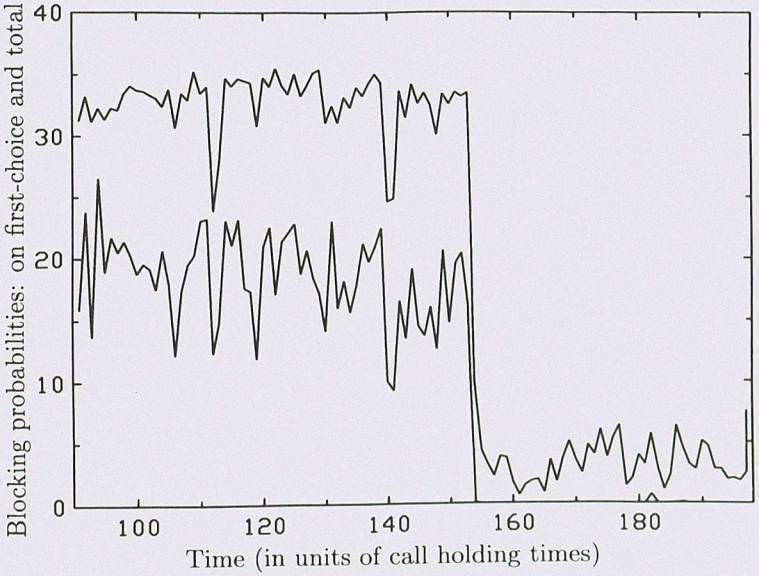

Figure 4.10 Simulation of Metastable Network

chical routing. The results show that a network that has reached a stationary regime at a relatively high blocking value can stay in this state for a very long time, up to 14 holding times, after the input traffic has been reduced to a value where the blocking should be quite low.

This phenomenon is also present in the network represented in Fig. 4.10. In this network, the total input traffic is reduced by 3% at time $t = 152$. The total end-to-end loss probability (the lower curve in the figure) and the probability of overflow (the upper curve) both undergo dramatic drops, from about 20% to virtually zero in the first case. The point is that this abrupt change in blocking probability is completely out of proportion to the change in the input value — an indication that something is not quite right in the network.

An even more troublesome phenomenon occurs in Fig. 4.11, where the traffic is dropped by 2% at time $t = 65$. Once again, we note a very large drop in loss probability; the network enters a low blocking state for approximately 10 holding times. Most disturbing, however, is that the network then undergoes a spontaneous transition to the high blocking state again, where it remains for a very long period. Note also the spontaneous transition to the low blocking state that occurs at $t = 55$ but that does not last very long.

Another way to view this phenomenon is as a classical hysteresis. This is shown in Fig. 4.9, where the network blocking is measured for different values

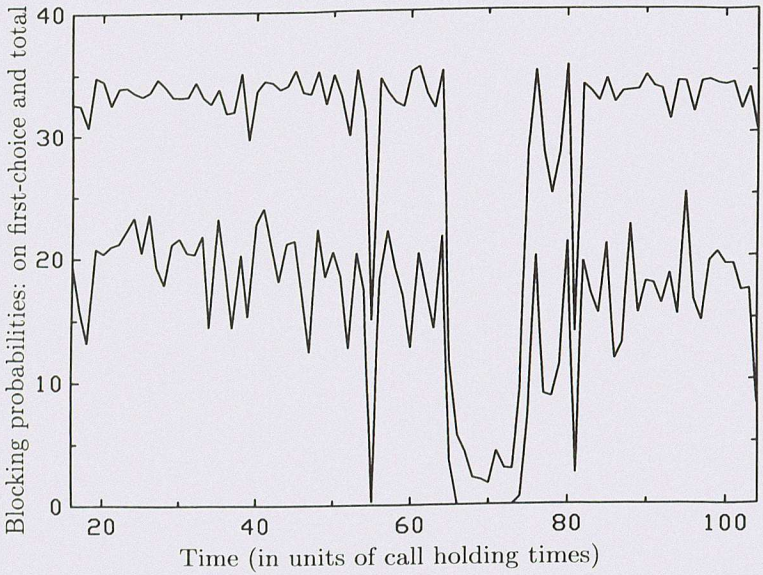

Figure 4.11 Spontaneous Transitions in Unstable Network

of the total offered traffic. The upper portion of the curve was produced by gradually decreasing traffic, while the lower portion corresponds to measurements made while increasing traffic. Each point was obtained after the network had reached a quasi-stable mode of operation after 200 holding times. As we see, the upper branch drops suddenly to a low value following the trajectory at the left, while the lower branch jumps to a high blocking value following the trajectory at the right, at a point higher than for the trajectory followed by the upper branch. This is the classical behavior of systems with hysteresis.

These results clearly show that the phenomenon is real, not a by-product of the simplifications used to compute a solution. Two questions remain: (1) What is the cause of this phenomenon? (2) How can it arise in an ergodic system in a stationary state?

We can obtain some indication concerning the first question by noting that load sharing does not exhibit such a behavior since the Erlang fixed point is unique. Also, we know that the fixed-point system for hierarchical routing (and for all ordered routings) is triangular, and can be solved in a single iteration over the links. It therefore seems that this hysteresis phenomenon is due not only to alternate routing, but also to the presence of mutual overflow in a network. Unfortunately, this is about as far as we can go in this direction; there is no definite proof that these statements are indeed true in general.

As for the second perplexing question, the answer, of course, is that the simulation results are not the true stationary results. What really happens is that the state space is divided into two groups of states, which we could call *macrostates*. In one macrostate, most calls are alternate routed with a correspondingly high loss while, in the other, most calls are direct routed with a much lower loss probability. It is plausible that once the network has entered the high-blocking regime, it will remain there for a long time: Once this regime is established, the probability that a new call can be routed directly is small, and most new traffic will continue to be alternate routed. The situation will endure until enough calls terminate in such a way that the probability of direct routing once again becomes large, at which time the network will drop into the low-blocking regime. This happens because of the statistical nature of the input processes; eventually there will be a time period during which few new calls arrive, permitting the transition. Depending on the load, the probability that the system will leave a macrostate can be small; when considered over time periods that are short with respect to the mean interval between transitions between the macrostates, the network appears to be in a nonstationary regime.

Note that stationarity is defined as a limit as $t \to \infty$. Even though the system may oscillate periodically between two macrostates with very different blocking, taking measurements for long periods gives a stationary measure of loss probability, which will lie somewhere between the two values. The point is that this value is not a very meaningful description of a network's performance. In other words, although the network has a precise mean performance, we would not want to operate it under conditions where there are large deviations from this mean for long time periods — precisely what happens in the cases we have seen. Such undesirable behavior must be prevented.

Theoretical Results

In certain simple cases, sufficient conditions can be imposed that guarantee a unique solution of the Erlang fixed point for alternate routing — for symmetric networks, for example, if it is assumed that the network is of infinite size. A straightforward calculation of this kind can be found in [15]. Consider once again the model with state protection given by Eqs. (3.29–3.31). With the assumption that, for each stream, there is an infinite number of overflow paths of S links each, we can compute the link-offered traffic as

$$a = A\left[1 + \frac{2B}{1 - B'}\right], \tag{4.41}$$

where A is the total first-offered traffic for each link. Replacing in the traffic model, we obtain the following equation for the link-offered traffic:

$$\sum_{i=0}^{m-1} \frac{a^i}{i!}\left(\frac{a}{A} - 1\right) = Sa^m \frac{A^{N-m}}{N!}. \tag{4.42}$$

The problem is to determine under what conditions — and in particular, for what value of m, the state protection level — this equation has a unique solution. The problem can be solved if we consider Eq. (4.42) as a polynomial equation of degree m in a with coefficients b_k, $k = 0, \ldots m$. Expanding in powers of a, we get

$$0 = \left(\frac{1}{(m-1)!A} - S\frac{A^{N-m}}{N!}\right)a^m + \left(\frac{1}{(m-2)!A} - \frac{1}{(m-1)!}\right)a^{m-1} + \ldots + \left(\frac{1}{A} - 1\right)a - 1.$$

We can use Descarte's rule to determine the maximum number of solutions to this equation. We consider two cases, $k < m$, and $k = m$. In the first case, we see that $b_k < 0$ whenever $A > k$. In particular, we can ensure that all these coefficients have the same sign (all negative) if $A > m - 1$. For the second case, we have $b_m < 0$ if

$$A > \left(\frac{N!}{S(m-1)!}\right)^{\frac{1}{N-m+1}} \triangleq A^*.$$

We can express the condition on b_m as

$$b_m \begin{cases} < 0 & \text{if } A > A^* \\ > 0 & \text{if } A < A^* \end{cases}$$

Suppose now that we select a protection level m^* such that

$$m^* - 1 = A^* = \left\lfloor \left(\frac{N!}{S(m^*-1)!}\right)^{\frac{1}{N-m^*+1}} \right\rfloor.$$

Whenever $A < m^* - 1$, there is a single change in sign in the coefficients, and hence there is a unique solution. When A exceeds this threshold, all the coefficients of the polynomial have the same sign, and there is no solution. If the protection threshold is different from m^*, then there can be more than a single sign change in the coefficients for $A^*(m) < m^* - 1$, indicating that there can be more than one solution to the polynomial equation. Also, we can see that, unless state protection is used, all asymptotic networks are unstable if $N > 2$.

A similar analysis is due to Marbukh [16], who considered the state equations for the symmetric network in the asymptotic case. Given an (o, d) call, define the set of available paths $R(o, d, s, m)$ as all paths between o and d with exactly s intermediate nodes and no more than m circuits busy on each link of the path. Two classes of routing were considered. Type 1 routings randomly select a path in $R(o, d, s, m)$, while type 2 routings randomly select a path

from the subset of $R(o,d,s,m)$ made of the least-loaded paths. Let $p_i(t)$ be the probability of having i busy circuits at time t on a given trunk. We can write the Kolmogorov equations for these probabilities as follows:

$$\frac{dp_i}{dt} = -p_i\,(i + A[1-\delta_{i,N}]) - Aq_i(\mathbf{p}) + p_{i-1}A\,(1-\delta_{i,0}) + Aq_{i-1}(\mathbf{p}) + p_{i+1}\,(i+1)\,(1-\delta_{i,N}). \tag{4.43}$$

The arrival rate at the link in question is composed of first-offered Poisson traffic A and some unknown amount of overflow traffic. The fraction of the first-offered traffic that arrives at the link, represented by q_i, is given by

$$q_i = \begin{cases} sp_ip_N \Big/ \sum_{j=0}^{m-1} p_j & \text{if } i < m \\ 0 & \text{otherwise} \end{cases} \qquad \text{when } r = 1$$

$$= \begin{cases} sp_N & \text{if } i < m,\ j < i,\ p_j = 0 \\ 0 & \text{otherwise} \end{cases} \qquad \text{when } r = 2$$

The system has two asymptotically stable solutions depending on the value of A. The asymptotically stable solutions for routing $r = 1, 2$ are

$$p_i = \begin{cases} (p_1^*, p_2^*, \ldots p_n^*) & \text{if } A < A_r^*(s,n) \\ (0,0,\ldots 1) & \text{if } A > N/s \end{cases}$$

For $r = 1$, A_1^* is the smallest value of A such that

$$E(A(x), N) = x \tag{4.44}$$

has a solution $x \in [0,1]$, and where

$$A(x) = A\frac{1 + (s-1)x}{1-x}$$

and the p_i^*s are given by

$$p_i^* = x^* \frac{N!}{i!}\left[\frac{1-x^*}{A(1+(s-1)x^*}\right]^{N-i},$$

where x^* is the smallest solution of Eq. (4.44). Note that $A(0) = A$ and $A(1) = \infty$. Thus the system always has a solution at $x = 1$, and the left-hand side is $E(A,N)$ at $x = 0$. Given the shape of the Erlang B function, the system either has a single solution or has three solutions. Two of these are stable; the implication (although there is no formal proof) is that the third one is unstable.

For $r = 2$,

$$A_2^* = \max_{i=0,1,\ldots N-1}\left(\frac{1}{s}\frac{N!}{i!}\right)^{\frac{1}{N-i}}.$$

The stationary probabilities are given by

$$p_i^* = \begin{cases} 0 & \text{if } i \leq l-1 \\ p_N^* \dfrac{(N! - sl!A^{N-i})}{(A-l)l!A^{N-l-1}} & \text{if } i = l \\ p_N^* \dfrac{N!}{i!A^{N-i}} & \text{otherwise} \end{cases}$$

where l is determined by the inequality

$$\left(\frac{lN!}{sl!}\right)^{\frac{1}{N-l+1}} < A < \left(\frac{N!}{sl!}\right)^{\frac{1}{N-l}}$$

and p_n^* is given by the normalizing condition on the P_is.

This phenomenon is intimately related to alternate routing with mutual overflow. Ordered networks have no mutual overflow, and the nonlinear system of equations (4.39) is triangular. Given that the Erlang B function is uniquely invertible, the system has a unique solution. In any event, the reality of the phenomenon is no longer in doubt. Furthermore, it is conceivable that a network operating on the lower portion of the curve may suddenly jump on the upper part because of a temporary surge in traffic. The implication is that the network would then stay in that high blocking state for very long periods, long after the original perturbation disappeared. This behavior is clearly undesirable, and ways of preventing it must be found.

State Protection

As is clear from the previous discussion, networks with mutual overflow are inherently unstable, at least to the extent that they can operate in the hysteresis mode that we just described. As is also clear, a mode of operation such as the one described by Fig. 4.11 is absolutely unacceptable in a real network, and some means must be provided to prevent it. One such technique, originally suggested by Grandjean in a different context [17], was proposed by Akinpelu [14] under the name *trunk reservation*. The technique has also been proposed for adaptive routing methods based on residual capacity under the name *state protection* [18]. Because of the potential for confusion with physical trunk reservation, we shall use the second term here, with the understanding that it has precisely the same meaning as in [14].

The current practice in telephone networks is to connect a call whenever there is a free trunk on a group. A call is blocked only when all trunks are busy. State protection operates by distinguishing among various types of calls offered to a group, blocking some of them *before* all trunks are busy. Specifically, first-offered calls are blocked only in the all-trunk-busy condition, where calls overflowing from some other group are blocked whenever $m < N$ trunks are busy.

The term *trunk reservation* naturally arises because part of the group's capacity $N - m$ is reserved for first-offered traffic. This, however, is very

different from *physical* trunk reservation, where some specific trunks are marked inaccessible for some calls. Here, any call can occupy any trunk, and the restriction is only on the *number*, not the *identity*, of the trunks that cannot be occupied by overflow calls. For this reason, we use the term *state protection*, which does not have the connotation of physical reservation.

Given that such a state-protection scheme is to be used, one must be able to compute the link-blocking probabilities in order to compute the network performance. Thus the traffic model must be modified to take into account the state protection. The link-loading phase is the computation of a and $\hat{a}$ given B and B'. In this case, stream 1 is identified with first-offered traffic, and stream 2 with overflow traffic. The actual computation of the link-offered traffic, a straightforward modification of the equivalent formulas for alternate routing without protection, is left as an exercise (Problem 4.21). The link-blocking stage is given by Akinpelu's model (see Section 3.6).

For this discussion, we assume that there are two functions of the trunk sizes, traffic parameters and reservation level, such that, for each link,

$$p = f(a, \hat{a}, m, N) \tag{4.45}$$

$$\hat{p} = g(a, \hat{a}, m, N) \tag{4.46}$$

and where, of course, a and $\hat{a}$ are functions of p and $\hat{p}$. As usual with the two-phase method, the correct values of p and $\hat{p}$ are given by the solution of this system of equations.

Effect of State Protection

We can estimate the effect of state protection both on the theoretical solution of the flow equations and on the behavior of the network. In the first case, we must express the analog of Fig. 4.8 for the case of state protection. The single-moment model had a single unknown variable for the symmetric network case, and the model with state protection now has two, that is, p and $\hat{p}$, whose value determines the network blocking probability. In order to have a graphic representation, it is necessary to rewrite the system (4.45) and (4.46) as

$$F(a, \hat{a}, m, N) \triangleq p - f(a, \hat{a}, m, N) = 0 \tag{4.47}$$

$$G(a, \hat{a}, m, N) \triangleq \hat{p} - g(a, \hat{a}, m, N) = 0 \tag{4.48}$$

The system can be represented graphically by plotting the isocontours of F and G at the value 0 in the p and $\hat{p}$ plane. The intersection(s) of these contours defines the solution(s) of the system. An example of multiple solutions is shown in Fig. 4.12, and the effect of increasing the protection level is shown in Figs. 4.13 and 4.12.

It is obvious that no level of protection ensures that multiple solutions do not occur. Mason, DeSerres, and Meubus [19] have given sufficient conditions

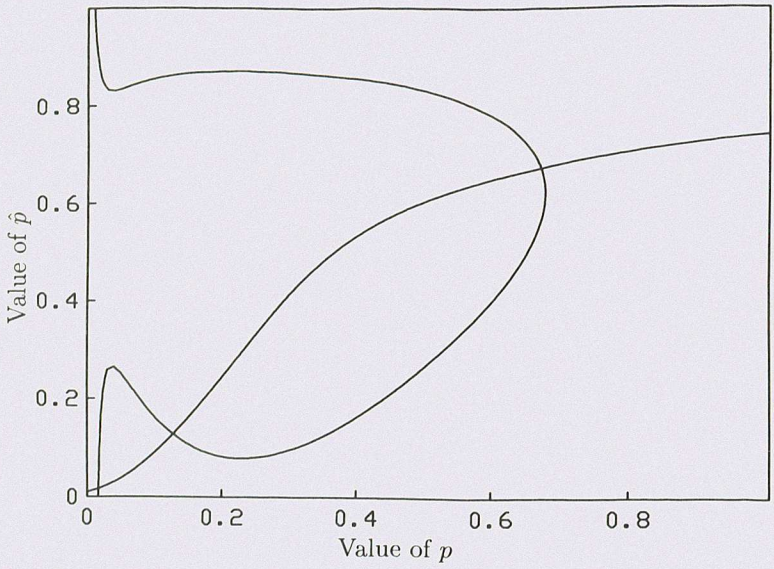

Figure 4.12 Contours of F and G Functions for No State Protection; Traffic = 42 Erlangs, 100 Trunks, 9 Alternate Routes

under which this cannot occur, again for symmetric and asymptotic networks. The actual effect of the reservation on the behavior of realistic asymmetric finite networks remains to be determined.

Finally, as shown in Fig. 4.15, state protection, in this case at a level of one, indeed eliminates the sudden changes of states, and the network operates in a much more satisfactory mode.

4.5 Summary

What have we learned so far? The most important fact to come out of this discussion is the presence of a lurking fixed-point system of equation in most cases of alternate routing and load sharing. This system determines to a large extent the difficulty of evaluating the performance of a network and, to an even larger extent, the difficulties in other areas of network optimization. We have also encountered one consequence of this system, the possibility of hysteresis, and indicated how it can be prevented by state protection.

Because of the simplifying assumptions made in the discussion, these results are mostly of theoretical significance. Our assumptions include the use

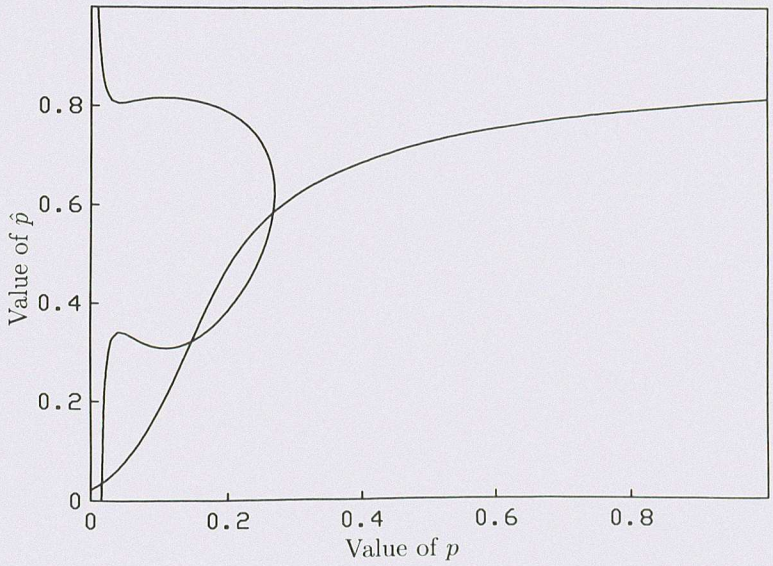

Figure 4.13 Contours of F and G; $N - m = 1$, Traffic = 42 Erlangs, 100 Trunks, 9 Alternate Routes

of the Poisson model for the internal traffic and the reduced-load path model of Eq. (4.5). In practice, analysis methods do not always use these assumptions, and one objective of the next chapter is to see how the two-phase method described here can be extended to cover other cases.

We will also describe other generalizations of the link-decomposition method — in particular, the cluster- and path-decomposition techniques. Although these techniques have not yet been used, they may turn out to be useful for analyzing circuit-switched networks with more complex types of traffic than voice.

Problems

4.1. Compute the end-to-end blocking probability as a function of link blocking probability for SOC control in the case of two-link nonhierarchical routing.

4.2. Consider the double-sector tandem network of Fig. 4.16. Assuming that the link-blocking probabilities are known for all the links, compute