Dynamic Routing of Circuits and Cells

David Bookstaber[1]

Abstract

This project investigates methods of routing data through a nonhierarchical network based on fixed "policies." The motivation is the standard telephone network used in the United States, which depends on setting up a circuit between two nodes in order to accommodate a call between them. The goal of any routing methodology is to accommodate as many calls as possible, since rejecting any call represents a loss of revenue to the network. A dynamic routing policy consists of an ordered list of routes between each node pair on the network. When a call arrives, the network accepts it on the first unblocked route it finds, and rejects it if all routes in its list are full.

This project concentrates on the work of Denardo & Park, who invent a routing method that assigns a cost to each link in the network, prices routes as the sum of the costs of their links, and chooses routes with the lowest cost. They present an efficient algorithm for generating such optimized routing policies, and prove that its results are optimal. However, their work does not address newer digital networks that send data in cells (fixed-length packets) instead of demanding an uninterrupted circuit to carry a call. In practice, these Asynchronous Transfer Mode (ATM) networks can carry more calls than circuit networks, while providing virtually the same quality of service. This project compares the properties of circuit and ATM networks, and shows that the routing methodology of Denardo & Park is applicable to the latter.

Introduction to Circuit Networks

A telecommunications network consists of *nodes* connected by undirected *links*. Each link has a fixed capacity, T, which is the maximum number of calls it can carry. *Calls* arrive at random between pairs of nodes. The network must either accept each call by opening a circuit along some *route* of links between the nodes, or else reject the call. Rejection of a call, or *loss*,

[1] Computer Science & Mathematics Senior Project report, Spring 1999. Advised by Eric Denardo.

typically occurs either because all of the routes have some link that is already at capacity (*blocked*), or because accepting the call would probably cause the rejection of more than one other call. Our goal is to minimize the expected number of calls lost across the entire network. For a phone company, this means carrying as many calls as possible on the installed hardware, since each call generates revenue. In that sense, a rejected call is lost revenue, and an underutilized network is a waste of money.

Traffic routing on U.S. telecommunications networks is done according to preplanned routing tables, which may be changed every hour and recalculated every week.[2] Under such *dynamic routing* paradigms, a *routing policy* is a series of routes defined for each node-pair that are tried in a specific order when a call arrives on that node-pair. We deal with *nonhierarchical* networks, which consist of nodes that are logically fully connected and functionally equal, so that there exists a direct route between all node-pairs. This direct route is always chosen first if possible. Only if it is blocked are alternative routes involving more than one link tried.[3]

Routing policies are optimized for a specific *network*, which consists of a set of nodes, links, and Poisson demands for service (calls) between node-pairs. Following convention, only one- and two-link routes are considered between node-pairs. It has been shown that allowing more complex routes does not significantly decrease loss rates. Obviously, the routing problem is only interesting for busy networks, and if the network is very busy it would be unwise to use many links to satisfy a single call when each link could potentially serve its own call. Ash confirms that, in practice, multi-link networks route about 98% of their traffic on one- and two-link paths. He attributes this also to costs inherent in switching. Note also that considering routes of greater than two links would vastly increase the complexity of the problem, since routes for the same node-pair could share links.

The dynamic routing methods treated here are not sensitive to immediate loads on the network, but see only whether a route is blocked or not. This corresponds to the DNHR (Dynamic Nonhierarchical Routing) hardware AT&T used on its networks in the 1980s. More recently, AT&T has adopted a Real-Time Network Routing (RTNR) system that is more sensitive to the state of the network, but which requires more intelligent switching hardware.

[2] Ash, p.23

[3] This nondependence on a single fixed route is critical to the ability of networks to respond to local failures or traffic surges.

Optimized Circuit Routing

Denardo & Park solve the problem of finding an optimal routing policy by a process of "decentralization," wherein the cost of routes is determined from the sum of the cost of their links. Instead of examining each route at each possible network load, they calculate the expected marginal decrease in revenue from accepting a call on a single link. Thus, the cost of accepting a call is the sum of the costs of assigning another call to each link, where the cost of adding a call to a link is the expected number of calls that will be rejected as a result. Obviously, calls are assigned to the cheapest route available, and offered no route that would in probability cause the rejection of more than one other call.

We make the following three assumptions about the circuit network:

- Call attempts on each node-pair α are independent and Poisson with a mean call rate of λ_α.
- Call durations are i.i.d. with mean $1/\mu$.
- Link behavior is independent of other links. (Obviously, this is an approximation since traffic on routes consisting of more than one link correlates their behavior. However, Denardo and others verify by simulation that this assumption underestimates the network reject rate by only a few percent.)

Routing policies vary with the network load. In order to compute an optimal routing policy, we will first examine an individual link to determine the marginal effect on the rejection rate of accepting a call. Then we will be equipped to define and analyze routing policies. Our goal will be to find fixed points of policies, which describe their steady-state effect on the network. In order to do this, we will first see how a policy transforms arrival rates on a network, then how to determine whether a fixed point exists, and finally how to construct an optimal policy and find its fixed point. In the process, we will learn about the behavior of networks.

How much does the expected rejection rate of a link increase if another call is accepted?

For a link i, the probability $N_i = T_i$ (*usage* = *capacity*) that the link is full is the Erlang probability ρ. That is, blocking probability on a specific link with capacity T and call rate[4] A is given by

$$\rho = \frac{\left(A/\mu\right)^T}{T!\sum_{n=0}^{T}\frac{\left(A/\mu\right)^n}{n!}}.$$[5]

Thus the expected increase in rejection rate if another call is accepted on the link now is given by

$$E(T,A) = \rho\left[\frac{T}{1-\rho} - \frac{A}{\mu}\right],$$

which is known as the *Erlang improvement formula*. In other words, at any moment accepting another call is expected to cause the rejection of E future calls on that link. (A few notes regarding this equation: This value is useful in designing network capacities—"Moe's principle" dictates that links should be sized so that their E values are equal. $E \times (A/\mu)$ = decrease in reject rate when link's capacity is increased from $T - 1$ to T. Denardo & Park provide a proof using Markov chains that this incremental cost of assigning a call to a link, E, depends only on the mean of call service times, $1/\mu$.[6] They also prove that E accurately describes the effect on the rejection rate if the arrival rate is perturbed by ε.)

Defining a Routing Policy

As already described, a routing policy consists of an ordered sequence of routes to be tried for each node-pair in the network. Given a policy, we want to find the actual arrival rates A it produces in the steady state of the network—i.e., what the policy actually does. We will describe policies by an operator f that maps arrival vectors $A \rightarrow A'$. Therefore, we want to find the fixed point of f—the steady-state arrival rates.

[4] In what follows, we will not always be interested in the raw call rates between node pairs, λ. Because our routing policy will displace these calls along different links, we will look instead at actual arrival rates A on each link under the policy. We will see later how to calculate A.
[5] This formula is also called the Erlang loss formula, and was proven by A. K. Erlang in 1917.
[6] Both Denardo and Ross prove that the steady-state probabilities in a network depend only on the *mean* of the call durations.

How do we evaluate f(A)?

A policy δ is an ordered list of routes $\{ r_i \}$ for each node-pair α. The policy puts a call on the first unblocked route in the list. Given a vector *A* of arrival rates for each link, the probability of a particular route being used by a call on α is given by the "Offer" probability,

$$O^\delta (r_k) = O^\delta (r_{k-1})[1 - P(r_{k-1} \mid A)] , \qquad r \in \delta_\alpha ,$$

which is "the probability of an offer being made on the previous route times the probability it was rejected." Here, *P*(*r* | *A*) is "the probability route *r* is not blocked given arrival rates *A*." Referring back to the Erlang probability, it is clear that this is the product of the probability of each link in the route being unblocked—i.e.,

$$P(r \mid A) = \prod\nolimits_{j \in r} (1 - \rho_j) .$$

Again, the value of ρ depends on the arrival rates given by *A*, and the ordering of the routes *r* in the probability *O* depends on the ordering given by the policy δ. Given these probabilities and the raw call rates λ we can calculate *f*(*A*) link by link, since for each link *i* we have

$$[f^\delta (A)]_i = \sum\nolimits_{r \in R_i} \lambda_{\alpha(r)} O^\delta (r) \frac{P(r \mid A)}{1 - \rho_i} .$$

This sums the *offer rate* on a route times the probability of link *i* blocking the route, over all routes *r* that include link *i*, giving us a new vector of arrival rates.

How do we know if a fixed point exists?

This question forces us to consider the effects of rerouting induced by adding a call to the network. In order to do so, we need to define a few variables that describe what happens to a call when it is rerouted. Our aim with these definitions is the construction of a *reroute matrix* $\boldsymbol{\Gamma}$ that describes how incremental traffic is routed around the network. By analyzing the properties of this matrix, we can determine whether the network has a fixed point.

First, calculate the rate at which call attempts are blocked on route *r* because of link *i*,

$$\Lambda_i (r) = \lambda_{\alpha(r)} O^\delta (r) \frac{P(r \mid \lambda)}{1 - \rho_i} .$$

$\Lambda_i (r)$, then, is "the call rate on the node-pair served by *r*, times the probability of the call being offered on route *r*, times the probability that all links in route *r* are unblocked except link *i*."

We will also need to know, given that link *i* has blocked a call, that the probability the call was attempted on route $r \in R_i$ is

$$Q_i(r) = \frac{\Lambda_i(r)}{\sum_{s \in R_i} \Lambda_i(s)}.$$

Now we can define the probability of a blocked call on a route *r* being passed to another route *s*. We say that $s \overset{\delta}{>} r$ if policy δ ranks route *s* after route *r*. Thus, the probability that a call blocked on route *r* due to link *i* is actually assigned to route *s* is

$$\gamma_i(r,s) = Q_i(r)P(s)\prod_{t:s \overset{\delta}{>} t \overset{\delta}{>} r}[1 - P(t)] \qquad \text{for } s \overset{\delta}{>} r.$$

In other words, $\gamma_i(r,s)$ is "the probability a call is blocked on route *r* due to link *i*, times the probability route *s* is unblocked, times the probability every route ranked after *r* but before *s* is blocked."

Finally, we need to consider the blocking probability of calls blocked on *i*. This is

$$\beta_i = \sum_{r \in R_i} Q_i(r) \prod_{s \overset{\delta}{>} r}[1 - P(s)],$$

the probability of losing a call, given that it was blocked on link *i*, because all other routes listed by the policy are also blocked. Thus the immediate effect of assigning a call to link *i* is to increase the expected number of future calls that will not only be blocked on link *i*, but also completely lost, by

$$\tau_i = E_i \beta_i.$$

To answer our original question, we need to determine the cumulative effect on the network of adding traffic to a link. Obviously, rerouting from a blocked route to another one may in turn cause other calls to be rerouted that wouldn't otherwise. In order to analyze this effect, we will construct a *reroute matrix* **Γ**, whose ijth entry is "the expected change in the number of future calls that will be assigned to routes that include link *j* due directly to rerouting the calls that will be blocked on link *i*, because one call was assigned to link *i* while it was unblocked." The matrix is produced by the following

Algorithm to compute the reroute matrix Γ:

1. Set $\Gamma_{ij} = 0$.
2. For each link *i*, each route $r \in R_i$, each route $s \overset{\delta}{>} r$, and each $j \in s$,

   $$\Gamma_{ij} \leftarrow \Gamma_{ij} + E_i \gamma_i(r,s)$$

3. For each link *i*, each route $r \in R_i$, and each link $j \in r \setminus i$,

   $$\Gamma_{ij} \leftarrow \Gamma_{ij} - E_i Q_i(r).$$

The algorithm considers the following: Assigning a call to link *i* while that link is unblocked increases the expected number of future calls that link *i* will block by E_i. Each of these future calls had been assigned to route $r \in R_i$ with probability $Q_i(r)$ and will be reassigned to route $s \overset{\delta}{>} r$ with probability $\gamma_i(r,s)$. Thus, for each $s \overset{\delta}{>} r$, the traffic on each link $j \in s$ is expected to increase by $E_i \gamma_i(r,s)$. At the same time, the traffic on all the other links in route *r* is expected to decrease by $E_i Q_i(r)$.

This construction of **Γ** facilitates an analysis of rerouting across the entire network. Since the ij^th^ entry of **Γ** describes the increase of traffic to link *j* due to an increase on link *i*, the ij^th^ entry of **Γ²** describes the second-degree effects on *j* of an increase on link *i*—i.e., the traffic rerouted over *j* from various routes that had traffic increased due to the increase on *i*. The call losses at each stage are given by the vectors τ, $\mathbf{\Gamma}\tau$, $\mathbf{\Gamma}^2\tau$, etc. Therefore, the cumulative loss of calls across the network is described by $(\mathbf{I} + \mathbf{\Gamma} + \mathbf{\Gamma}^2 + \ldots)\tau$.

A square matrix **Γ** is *transient* if $\Gamma^k \to 0$ as $k \to \infty$. Under these conditions, the matrix $(\mathbf{I} - \mathbf{\Gamma})$ has an inverse, and by calculus we know that

$$\mathbf{D} = (\mathbf{I} - \mathbf{\Gamma})^{-1} = \mathbf{I} + \mathbf{\Gamma} + \mathbf{\Gamma}^2 + \ldots.$$

So, the matrix **D** describes the expected number of calls that will be assigned to link *j* due to the assignment of one call to link *i* while *i* is unblocked. Furthermore, $|\mathbf{D}\tau|$ is the expected number of calls that will be lost from the network due to this assignment.

The properties of these matrices are important to understanding a routing policy. For example, if **Γ** weren't transient, assigning a single additional call to an unblocked link could cause an avalanche of rerouted traffic. I.e., rerouting that call would cause another two calls to be rerouted, which would block even more calls, and so forth. Under such conditions, a policy has no fixed point. However, if $\mathbf{\Gamma}$ is transient a fixed point exists. Furthermore, when calculating a fixed point we are concerned with the *rate of convergence* of rerouted traffic. This is determined by the largest eigenvalue (in absolute value) of $\mathbf{\Gamma}$—its *spectral radius*. Therefore, we might expect numerical difficulties in computing a fixed point if the spectral radius of $\mathbf{\Gamma}$ is not significantly less than one.

How do we construct an optimal policy?

We want to find the policy that minimizes the system-wide reject rate. For a policy δ with fixed point A^δ this is given by $\sum_\alpha \lambda_\alpha [\prod_{r\in\delta_\alpha}(1-P(r\mid A^\delta)]$, where δ_α is the set of ranked routes listed by the policy for node-pair α. Denardo & Park minimize the reject rate through a decentralized measure of cost—they assign a cost to each link, and let the cost of a route be the sum of the costs of its links. Based on our derivations in the previous section, it is clear that the cost of a link i is

$$C_i = \sum_{j=1}^{L} \mathbf{D}_{ij}\tau_j \; .$$

C_i, then, is "the expected number of calls that will be lost due to the assignment of one call to a route that includes link i." Clearly, no call should be assigned to a route whose cost exceeds one. Denardo proves that ranking routes according to cost in this manner achieves a minimal network-wide loss of calls.

How do we find a fixed point of f?

Denardo & Park execute an algorithm that starts with a simple problem and slowly increases its complexity to accommodate the complete network. This process of successive approximation begins with a policy that uses only single-link routes, for which it is easy to compute blocking probabilities (since they are just the Erlang probabilities for each link). Then a small fraction of the traffic is allowed to use multi-link routes, and the policy is re-optimized. The process continues, allowing more and more traffic to use multi-link routes. Each step begins with the computation of the fixed point for the policy determined in the previous step. Then the policy's reroute matrix and cost vectors are computed. Finally, for each node-pair a policy is selected that ranks routes in increasing order of cost, and that lists no route with a cost greater than one.

This process naturally produces cyclic routing policies, which is what dynamic routing networks often use in practice.[7] It is worth noting that on a six-node, totally-connected model network used by AT&T, Denardo's implementation computes an optimal policy in just a few seconds.[8] This algorithm is $O\left(\sum T_i + R \cdot L \cdot M\right)$, where *R* is the number of routes considered, *L* is the number of links, and 1/*M* is the fraction of traffic opened to multi-link routes each step (resulting in a cyclic policy of period *M*). The first term accounts for the calculation of Erlang probabilities for each link. The second term accounts for the calculation of offer probabilities and the transformation of arrival rates by the new policy, and also covers the computation of the reroute matrix. These are the most expensive operations involved in the process.

Introduction to ATM

AT&T recently announced that by the end of this year, it would stop buying traditional telephone switches. Instead it, like all other major telecommunications network providers, is switching to networks that rely on Asynchronous Transfer Mode (ATM) technology. Frank Ianna, president of AT&T's network unit, said of ATM, "If analog was the first generation and digital switching was the second and digital transmission was the third, this is the fourth."[9]

Instead of a circuit, a call on an ATM network establishes a *Virtual Circuit* (VC; also *Virtual Channel*) along which it can send data cells of 53 bytes.[10] Links in an ATM network consist of multiplexers that combine the numerous VCs on the link into a single stream of data no larger than the link's capacity, or bandwidth. The advantages of using cells instead of circuits include the ability to use channels of varying capacity, and to pack data more tightly. In practice, digital services that use these networks—voice, video, data—have variable data rates, especially after compression. Therefore, while calls have a peak rate at which they may carry data,[11] they

[7] Ash, p.27
[8] A more brute-force, applied algorithm for calculating fixed points is provided by Ash, who couples linear programming with flow models of the network to make candidate paths converge on optimal routing schemes. Ross also gives several approximation algorithms. However, these algorithms can take orders of magnitude longer than Denardo & Park's to reasonably converge. (Ash, Chapters 4 & 6; Ross, Chapter 7)
[9] "AT&T's Embrace of New Technology Signals Next Era," New York Times, 8 March 1999, C1.
[10] Typically, 5 bytes of a cell are used for routing, error correction, and other "non-payload" data.
[11] In the case of standard voice service, this maximum rate is 64kbps.

do not transmit at this peak rate for their entire duration. For example, voice calls have frequent periods of silence during which no data needs to be sent. ATM networks do not carry any data during these periods, which frees up bandwidth for other calls.[12]

The qualities of VCs vary by *service type*. Thus, where circuit network calls consist of a single service that uses a fixed unit of bandwidth, ATM calls can have widely varying characteristics, including different requirements for peak data rates and different distributions of cell transmission rates over the course of a call. Another property of ATM calls that can vary by service type is the Quality of Service (QoS) requirements, which consist of the maximum allowable cell delay and cell loss. For example, while a voice conversation would be sensitive to cell delays of more than a few milliseconds, a few cells could be lost without a listener noticing. In contrast, a wide area computer network allows for delays on the order of seconds, but cannot afford to lose any cells of data.

Optimized ATM Routing

The most simple ATM routing method—*peak-rate admission*—reduces readily to the circuit method of Denardo & Park already described. In this case, a route is considered blocked if adding a particular VC would cause the sum of the peak rates of its VCs to exceed its capacity. This ensures that a link can carry all of its calls without delaying or losing any cells. However, it does not provide any improvement in service over the simple circuit network. In fact, in this case we can theoretically transform ATM to circuits by quantizing call sizes and circuits: The VC with the smallest peak rate would behave like a standard circuit call, and VCs with peak rates c times greater would behave like c simultaneous calls.[13] Something like this is used in a technique called *multirate circuit switching*, and it has the advantage over peak-rate ATM of not consuming 10% of its bandwidth with non-payload data. Still, ATM is more easily adapted to accommodate new services and bandwidth requirements, and ATM technology is superceding that of multirate circuit switching.

[12] One can observe this by listening carefully to a call made with current digital cellular telephones, for example.

[13] This does not truly match the circuit model because the call arrival rates would not be strictly independent—a high-bandwidth service would be simulated by the simultaneous arrival of several calls. While the consequences are probably not significant in practice, this is not considered further here because the real benefits of ATM are presented in what follows.

The real benefits of ATM are reaped by considering a more relaxed demand characteristic of calls, referred to as *effective bandwidth*. Here, some measure of the average data transmission rate of calls is calculated. A method that uses this value will admit calls as long as their cumulative effective bandwidth doesn't exceed the link's capacity. Since effective bandwidth is generally less than the peak bandwidth consumption of a call, this method will allow more calls to be carried than a standard circuit network. However, such a strategy jeopardizes QoS properties of the calls since, in the worst case, it is possible for all calls on a link to simultaneously demand their peak bandwidth, causing the link to delay or lose cells. Thus, we face a tradeoff between accepting more calls and ensuring QoS. The calculation of effective bandwidth is undertaken for each service to satisfy performance preferences in this tradeoff.

Since different types of calls have different demands for bandwidth and QoS, ATM networks can employ a strategy for *service separation* wherein all calls with the same properties (i.e., providing the same service) are logically combined in a single buffer. In this way, a link can divide its bandwidth over many service types, but the analysis of blocking probabilities becomes more tractable because each service is examined independently. Consider the following illustration: A link with capacity 100 carries 30 units of voice calls and 65 units of data. Given 30 units of voice bandwidth, this network is carrying 35 voice calls while ensuring minimal QoS (where a circuit network could carry only 30). Data, which can sustain much higher delays than the voice, and which has much more erratic demand probabilities, is packed into the 65-unit virtual data buffer in such a way that 130 voice-sized data sessions are occurring simultaneously.[14] In this state, the link has 5 units free, and so if necessary it could either expand its voice service bandwidth to accept several more voice calls, or accommodate up to 10 more data sessions.

Single Service

It is straightforward to apply Denardo & Park's routing method to single-service ATM networks. In addition to average duration and arrival rate, each VC is specified by QoS requirements and effective bandwidth that determine the maximum number of calls that can be

[14] In reality, commercial providers of internet Digital Subscriber Lines will oversubscribe many times the capacity of a link. Unless a customer pays the hefty premium for a guaranteed data rate, they may pack as many as a dozen 300kbps subscribers, for example, onto a single 300kbs link. Due to usage patterns, in practice the service does not substantially degrade in such an oversubscribed condition, and users usually get the full 300kbps they expect.

accommodated with a given capacity. Thus, we simply substitute the effective capacity of each link for its nominal capacity in the Erlang probability, and then the Denardo & Park method yields an optimal routing method for a VC of our choosing.[15]

Multiple Services

In the case of a link carrying multiple services, Ross derives a steady-state distribution from which we can derive the blocking probabilities. Ross refers to the problem of accommodating multiservice calls to a link as the *stochastic knapsack.* If *k* services can place calls on the link, we need to consider the set of service levels the link can accommodate. Let b_k be the effective bandwidth of service *k*, and n_k the number of service-*k* calls in progress on the link. We are interested in the set $S = \{\bar{n} : \bar{b} \cdot \bar{n} \le T\}$. In this case, Ross gives the steady-state probability of $\bar{n} \in S$ by

$$\pi(\bar{n}) = \frac{1}{G}\prod_{k=1}^{K}\frac{\left(\lambda_k/\mu_k\right)^{n_k}}{n_k!}, \text{ where the normalizing factor is } G = \sum_{\bar{n}\in S}\prod_{k=1}^{K}\frac{\left(\lambda_k/\mu_k\right)^{n_k}}{n_k!}.$$[16]

The blocking probability for service *k* on a link of capacity *T* becomes

$$\rho_k = \sum \pi(\bar{n}),\ \bar{b}\cdot\bar{n} > T - b_k .$$

Because these expressions can be prohibitively large for even moderate networks, Ross[17] provides a recursive algorithm for calculating blocking probabilities of this form in time linear with the capacity *T* and number of services *K*. Ross also gives an even simpler approximation for blocking probabilities that converges to the true values as *T* increases, as well as other efficient algorithmic methods.[18]

A few considerations can rapidly complicate this picture. For example, we have assumed that the market for network services is efficient—i.e., that each call generates revenue equal to

[15] The situation is complicated slightly when considering a multi-service network carrying services with very different QoS properties. The mechanics of multiplexing such calls are outside the theoretical interest of this paper, but are analyzed in Ross, Chapter 5. Also, note that QoS on two-link routes may be only half that on one-link routes. I.e., if each link can delay a cell for one microsecond, a cell traversing a two-link route might be delayed for two microseconds. Although, strictly speaking, this has an impact on the cost of using two-link routes since it takes more bandwidth on each link to ensure the exact same QoS as on a one-link route, this probably does not need to be included in our model since a degradation in QoS of this caliber is not catastrophic in practice.

[16] Note that this expression reduces to the Erlang probability if there is only one service. Note, also, that a closed-form expression can be derived for services of continuously variable size (see Ross, Chapter 2.8). The normalizing constant, *G*, is especially amenable to multidimensional Monte Carlo summation.

[17] See Ross p.25, *Algorithm 2.1*.

the revenue forgone by accepting it. However, in a multi-service network it is possible that the revenue rates per unit bandwidth consumed for different services will not always satisfy this property, and thus an arbitrage opportunity exists for the routing policy designer (albeit, a computationally expensive one[19]) who may want to block cheap calls more often than they would otherwise be lost. Note, for example, that the probability a large call will be blocked is greater than the blocking probability of a small call. Therefore, it may be preferable to reject a small call that would otherwise be accepted in order to keep reserve bandwidth open for a more efficient large call. Fortunately, using Denardo's model, it is not difficult to account for revenue disparities if we multiply the cost measures, C_i, by the expected revenue of each call, *w*. In that case, we reject a call *k* on route *r* only if its expected revenue, ${}^{w_k}\!/_{\mu_k}$, is less than $w \cdot C_r$.

In general, there are economies of scale to each service that need to be considered. I.e., as more and more service-*k* calls are packed into a link, the rate of increase in bandwidth required to maintain QoS decreases. While these gains can be enjoyed automatically in a single-service ATM network, there are no simple methods for accommodating this fact into a multi-service network with the dynamic routing policies considered here, since the routing policy cannot determine how many service-*k* calls are in progress on a link at any given time.

The routing method of Denardo & Park not only generates optimal policies for circuit networks, but is also applicable to other networks. This paper has shown how it is readily adapted to ATM networks, including the problem of routing multiple services. This suggests that it is widely applicable to other networks that have reasonable behavior and tractable expressions of blocking probability on a link.

[18] Ross, p.49; also Chapter 3.
[19] See Ross, Section 4.3 for a dynamic programming solution; Section 5.7 for approximation methods.

References

Gerald R. Ash, 1998. *Dynamic Routing in Telecommunications Networks*. McGraw-Hill, New York.

Eric V. Denardo & Haechurl Park, 1998. "Efficient circuit switching: routes ranked by cost." Unpublished manuscript.

Keith W. Ross, 1995. *Multiservice Loss Models for Broadband Telecommunications Networks*. Springer-Verlag, London.

Variables

A	Actual arrival rate of calls
α	Node-pair
b_k	Effective bandwidth of service *k*
β_i	Probability of losing a call given that it was blocked on link *i*
C_i	Cost of link *i*
$\mathbf{D}$	Cumulative effects of rerouting
δ	Routing policy
E_i	Expected increase in blocking rate if another call is accepted on link *i* now
$f(A)$	Transformation of arrival rates by a policy
$\mathbf{\Gamma}$	Reroute matrix—traffic increases across the network of adding a call
$\gamma_i(r,s)$	Probability a call blocked on route *r* due to link *i* is completed on route *s*
i	A specific link
K	Number of services
k	A specific service
L	Number of links
$\Lambda_i(r)$	Rate at which call attempts are blocked on route *r* because of link *i*
λ	Raw arrival rate of calls
M	Period of cycle in cyclic routing policy
$1/\mu$	Mean call duration
N	Number of calls in progress on a link
$O^{\delta}(r_k)$	Probability of a particular route being used under a given policy
$P(r \mid A)$	Probability route *r* is blocked given arrival rates *A*
$Q_i(r)$	Probability a call blocked on link *i* was made through route *r*
R_i	Routes containing link *I*
r	A route
ρ	Erlang probability – the blocking probability on a link with *T* and *A*
T	Capacity of a link
τ_i	Expected increase in loss rate if another call is accepted on link *i* now