

Control and Optimization Methods in Communication Network Problems

ANTHONY EPHREMIDES, FELLOW, IEEE, AND SERGIO VERDÚ, SENIOR MEMBER, IEEE

Abstract—In this paper we focus on two areas of communication network design in which methods of control and optimization theory have proven useful. These are the area of multiple access communication (for networks with shared links such as radio networks and local area networks) and the area of network routing (for networks with point-to-point interconnections). We review a few selected problems in each area to show the role of the control concepts involved and we then proceed to identify other areas of communication network design in which the same control theoretic and optimization methodology may be applicable and useful. We do not survey the work done in this area, nor do we review work in control areas whose methods are applicable in other communication network problems. Instead, we attempt to bring to the attention of the control systems community the numerous instances of problems arising in the pure communication network design process that can benefit from the attention and the capabilities of this community.

I. INTRODUCTION

COMMUNICATION networks are designed and built in order to share resources. If interconnecting systems and bandwidths were available at no cost, then the solution to the problem of communication would be to assign dedicated communication links (channels) of sufficient capacity to every pair of conceivable users to meet their needs. This not being the case, it is necessary to multiplex the sources of communication traffic in order to optimize various cost criteria. Frequently, this optimization is dynamic and done on the basis of feedback that monitors the evolution of the degree of utilization of the network resources. Thus, we should expect a number of problems arising in communication network design to fit naturally in the framework of control systems design. In this paper we wish to demonstrate that indeed this is the case and to show how various control and optimization methodologies have been used in the study of communication networks.

In the beginning there was a single communication network, the telephone network. It represented a multibillion dollar investment and seemed to serve reasonably adequately the voice communication needs. The explosive growth in data communication needs during the last 30 years built up the pressure for additional and alternative networking options. As a result, the notion of store-and-forward switching (known also as message switching) was introduced in the early 1960's. This notion represented a breakthrough since it constituted a radical reversal of thinking with respect to the circuit-switching process; namely, instead of securing an open, dedicated "pipe" for the transmission of

messages by means of hardware switches, it allowed a step-by-step (node-by-node) forwarding of messages, thereby permitting each node to switch messages by deciding when and where to transmit the messages in its buffer. In the last 20 years we have seen an avalanche of technologies (fast switching, time division multiplexing, local area networks, fiber optical networks, integrated services digital networks, etc.) and a proliferation of operational public and private networks that put these technologies to test and challenged communication engineers. In addition, they should challenge control engineers as well.

Without attempting a survey of this vast application area we wish to promulgate the viewpoint that many (if not most) specific sub-problems in the network design process are natural control problems. In support of this thesis, we choose, first, to demonstrate how two major areas in communication networks (routing and multiple access) have benefitted from the use of techniques borrowed from what is traditionally perceived as control systems methodology and, second, to mention additional areas that are likely to benefit from the control systems community. As illustrated in this paper, the techniques that have proved useful in communication networks include: dynamic programming (e.g., [2], [6], [8]–[10], [22], [29], [38], [39], [47], [49], [54]); linear programming (e.g., [50], [51]); constrained and iterative optimization (e.g., [5], [14], [16], [42]); Markov decision theory tools (e.g., [2], [26], [29], [38]); control of Markov chains (e.g., [11], [17], [18], [20], [40], [45]); stability analysis of stochastic systems via Lyapunov methods (e.g., [31], [43]); sample path dominance (e.g., [2], [52]); and convergence of distributed and asynchronous algorithms (e.g., [6], [16], [42]).

The problem of routing is encountered in all and every network that does not permit the source to reach the destination in a single transmission hop, but instead it must traverse a path of intermediate links. By contrast, the problem of multiple access is encountered primarily in those networks that permit the nodes to reach their destination directly in one hop by having to share the same link with other transmitting nodes. In addition, the two problems are fundamentally different in nature and, jointly, cover considerable ground in the networking area. Finally, together they facilitate the identification of additional design issues and the extension of the applicability of suitable control methods. Thus, they represent "cornerstone" areas of network design.

Routing can be studied either macroscopically or microscopically. The macroscopic viewpoint considers basically a flow model and determines the splitting of the flow in order to reach the destination in minimum time with efficient use of the network resources. It is traditionally referred to as *static routing*. The microscopic viewpoint dissects the flow process down to the atomic level of the individual transmission unit, the message (a string of bits commonly referred to as packet), and determines the path each message must follow at each of its hops through the network. It is traditionally referred to as *dynamic routing*. Both viewpoints are explored in Section II.

Multiple access is a collective term that refers to numerous problems that deal with the dynamic allocation of a single resource among users who can coordinate their use of that resource only by making use of that resource. These problems arise primarily in the context of radio channels but also in the

Manuscript received September 13, 1988; revised December 27, 1988 and April 24, 1989. Paper recommended by Associate Editor, T. L. Johnson. This work was supported in part by the National Science Foundation under Grants NSF-85-00108 and NSF ECS-88-57689, by the Office of Naval Research under Contracts N00014-84-K-0207 and N00014-87-K-0054, and by the Army Research Office under Contract DAAL-03-87-K-0062.

A. Ephremides is with the Department of Electrical Engineering, University of Maryland, College Park, MD 20742.

S. Verdú is with the Department of Electrical Engineering, Princeton University, Princeton, NJ 08544.

IEEE Log Number 8929606.

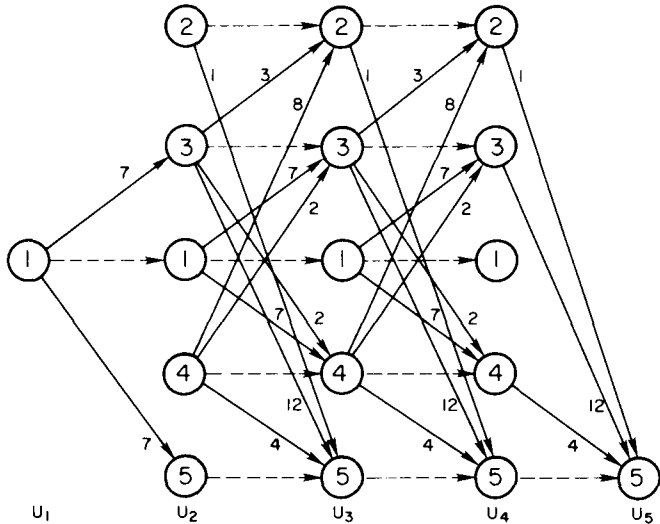


Fig. 1. Layered network showing link lengths. Source is node 1 in U_1 and destination is node 5 in U_5 .

context of shared cable resources in local area networks. In Section III, we explore the main multiple access problems where control methods have been successfully applied.

Both in the case of routing as well as in the case of multiple access we place the emphasis on the control techniques that have been used. We then show how these techniques, sometimes with slight modification, can be naturally transported to other problem areas such as voice-data integration, flow control, and the scheduling of messages and links. This is done in Section IV.

II. NETWORK ROUTING

The problem of routing in communication networks is one that has received early attention and has experienced significant breakthroughs in the brief history of the field of communication networks. It is one of the first problems that gained prominence as a result of the emergence of store-and-forward switching. It is also one in which analytical tools and available theories applied nicely from the beginning.

A. Static Routing

Given a network (a set of nodes connected by directed links) a path connecting the source node to the destination node has to be selected from the set of all possible such paths.¹ In the simplest formulation, the problem is one of finding the shortest path, i.e., a length is assigned to each link and the optimization criterion is the total path length. This problem is one of the archetypical combinatorial optimization problems (the solution can be found by exhaustive enumeration of a finite set of possibilities—all possible paths from source to destination). Among the many existing shortest path algorithms (see, e.g., [41]), the Bellman-Ford algorithm (1956) is of particular interest to our exposition, both because it is based on dynamic programming and because, as we will see below, it easily lends itself to distributed asynchronous implementation. A natural choice to find the shortest path from source to destination in a layered network (i.e., one in which the nodes can be grouped in subsets U_1, \dots, U_M such that the source and destination nodes belong to U_1 and U_M , respectively, and there are links only between nodes in adjacent layers U_{k-1} and U_k) such as the one in Fig. 1, is the dynamic programming algorithm, where the shortest paths and distances (costs-to-go) of the nodes in layer U_k to the destination are computed based on the shortest paths and distances of the nodes in layer U_{k+1} . If the

¹ All the algorithms and results discussed in this section can be extended to the case where there are several source-destination pairs in the network.

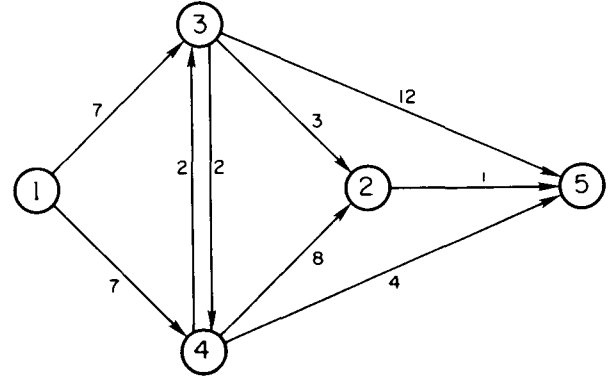


Fig. 2. Arbitrary network showing link lengths. Source is node 1 and destination is node 5.

network is not layered (such as that in Fig. 2), its shortest path can be obtained by finding the shortest path in a layered network derived from the original one as specified in the Bellman-Ford algorithm: the number of layers is equal to the number of nodes in the original network, say N , each layer contains a copy of each of the N nodes, and there is a link connecting two nodes in consecutive layers if such a link exists in the original network; in addition, copies of the same node in consecutive stages are connected by a zero-length link. (Fig. 1 was actually derived from Fig. 2 using this rule.) It is easy to see by induction that $D_k(i)$, the cost-to-go of node i in layer $N - k$, is the minimum length of any path from i to the destination that uses at most k links (in the original network). Since no shortest path uses more than $N - 1$ links (link lengths are assumed nonnegative and, therefore, no path containing loops need be considered), the cost-to-go of node i at layer 1, $D_{N-1}(i)$ will indeed be the length of the shortest path from node i to the destination. Thus, the Bellman-Ford algorithm can be formulated as the iteration

$$D_k(i) = \min_{j \in N(i)} [D_{k-1}(j) + d_{ij}] \quad \text{for } k = 1, \dots, N-1 \quad (2.1)$$

where d_{ij} is the length of the link from i to j , $N(i)$ is the set of nodes for which such a link exists and it is assumed that $D_0(i) = \infty$ if i is not the destination node, which corresponds to the removal of all the nodes but the destination in the final layer (Fig. 1).

Contrary to what may appear at first glance there is a lot more to network routing than finding shortest paths. After all, the shortest path may not be the best path. The reason is that the real goal is to minimize the *delay* experienced in going from source to destination, and the delay encountered in each link is usually a function of the amount of traffic carried by the link (as the link becomes congested, it takes longer to go through it), which is referred to as the link flow and is quantified in packets (or messages) per second. Then, assuming a given desired flow level from source to destination, the problem is how to distribute it among all the possible paths so as to minimize the total delay. In contrast to the previous more elementary formulation of the routing problem which led to the shortest path combinatorial optimization problem and which corresponds to the special case in which the link delays are independent of the flows, we now face a continuous optimization problem which can be written as

$$\begin{aligned} \text{minimize } F(x) &= \sum_{(i,j)} D_{ij} \left(\sum_{n \in P(i,j)} x(n) \right) \\ \text{subject to } x &\in X = \left\{ (x(1), \dots, x(J)) \in R^J, \right. \\ &\left. \sum_{n=1}^J x(n) = \lambda, x(n) \geq 0 \right\} \quad (2.2) \end{aligned}$$

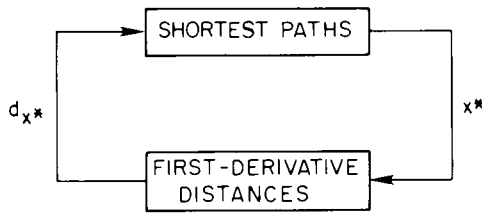


Fig. 3. Characterization of the solution to the minimum-delay routing problem.

where the set of all paths from source to destination is labeled $\{1, \dots, J\}$; $\mathbf{x} = (x(1), \dots, x(J))$ is the vector of unknown nonnegative path flows which sum up to λ , the desired flow from source to destination; $P(i, j) \subset \{1, \dots, J\}$ is the subset of paths that traverse link (i, j) ; and $D_{ij}(x)$ is the portion of the overall delay contributed by the link from node i to node j when the flow it carries is equal to x . In order to characterize a global solution to the optimization over a convex set in (2.2), it is natural to restrict attention to convex penalty functions. In practice, it is common that the incremental delay in a link grows with the amount of traffic it carries and, therefore, it can be assumed that the functions D_{ij} are convex without affecting significantly the practical applicability of the results.

Now, the characterization of the solution to (2.2), \mathbf{x}^* , is straightforward. Since the feasible set X and the penalty function F are convex, it is necessary and sufficient that the directional derivative of the penalty function be nonnegative when evaluated at \mathbf{x}^* in the direction of any of the elements of X (e.g., [37])

$$0 \leq \lim_{\alpha \downarrow 0} \frac{1}{\alpha} [F((1-\alpha)\mathbf{x}^* + \alpha\mathbf{x}) - F(\mathbf{x}^*)] \quad \forall \mathbf{x} \in X \quad (2.3)$$

which translates into

$$\begin{aligned} 0 &\leq \sum_{(i,j)} D'_{ij} \left(\sum_{m \in P(i,j)} x^*(m) \right) \sum_{n \in P(i,j)} [x(n) - x^*(n)] \\ &= \sum_{n=1}^J [x(n) - x^*(n)] d_{x^*}(n) \quad \text{for all } \mathbf{x} \in X \end{aligned} \quad (2.4)$$

where $d_x(n) = \sum_{(i,j) \in L(n)} D'_{ij}(\sum_{m \in P(i,j)} x^*(m))$ is the length of path n when the length of each link is equal to the derivative of its delay evaluated at the set of flows \mathbf{x} , and $L(n)$ is the set of links used by path n . The solution to (2.4), \mathbf{x}^* , is the vector in X that minimizes its inner product with the vector of distances \mathbf{d}_{x^*} . Thus, \mathbf{x}^* puts all its weight on the smallest component(s) of \mathbf{d}_{x^*} . The conclusion is that the optimum flow uses only shortest paths computed according to the *derivative* of the link delays.

This solution to the minimum-delay routing problem allows us to check whether a given set of flows is optimum. Unfortunately, it does not tell us how to find the optimum flows. Indeed, we face the chicken-and-egg situation depicted in Fig. 3. The optimum flows are obtained by solving a shortest path problem; but in order to compute the link lengths it is necessary to know the optimum flows. Nevertheless, the foregoing characterization of the optimal solution does suggest a possible iterative procedure to find the optimum set of flows. Starting with a given set of flows \mathbf{x} one can compute the minimum derivative shortest paths for that flow, and hence, a new flow, $\mathbf{x}^*(\mathbf{x})$ that is positive only along those shortest paths. The process can then be repeated, until there is no appreciable cost decrease. The region of convergence of such a procedure can be improved by letting the new flow be a convex combination of \mathbf{x} and $\mathbf{x}^*(\mathbf{x})$, i.e.,

$$\mathbf{x}_{k+1} = (1 - \alpha_k)\mathbf{x}_k + \alpha_k \mathbf{x}^*(\mathbf{x}_k).$$

This is the so-called *flow deviation method* of Fratta, Gerla, and Kleinrock [14], where $0 \leq \alpha_k \leq 1$ is chosen to minimize

$$F((1 - \alpha_k)\mathbf{x}_k + \alpha_k \mathbf{x}^*(\mathbf{x}_k))$$

which is a special case of the feasible-direction nonlinear programming algorithm due to Frank and Wolfe [13]. The convergence of the flow deviation method to the optimum routing is rather slow because unfavorable paths tend to carry considerable flow during many iterations unless the initial routing guess is particularly fortuitous. Such a behavior can be improved by reducing the flow along each nonminimum derivative path in accordance to the delay experienced in that path. This is the idea of iterative routing algorithms based on *gradient projection* nonlinear optimization methods (e.g., [4]) in which the flow decrease along a nonminimum derivative path is proportional to the difference between its length and that of the shortest path (according to the first derivative of the delay function). If such a decrease would result in a negative flow, then the flow along that path is set to zero (hence, the *projection* to the set of feasible flows).

We have seen that the problem of static network routing can be formulated as a conceptually straightforward optimization problem that admits well-known solutions in nonlinear programming. What sets optimum routing in communication networks apart from other multicommodity flow problems arising in operations research is the fact that the optimization is carried out in real time, and often, in distributed fashion, where each node makes its own routing decisions based on local information. The review of centralized routing has revealed that the shortest path problem plays a central role in solving for the optimum routing regardless of whether the link congestion measures depend on the link flow or not. Hence, we will start the exposition of distributed routing algorithms by discussing the distributed version of the Bellman-Ford shortest path algorithm.

The Bellman-Ford updating equation in (2.1) suggests that the algorithm is suited for decentralized operation because each node can update its own estimate of distance to the destination (cost-to-go) provided it receives from its neighbors their own estimates [appearing on the right-hand side of (2.1)]. The feature that makes the study of the distributed Bellman-Ford algorithm interesting is that it can run completely *asynchronously*, in the sense that the updating and communication times need not be coordinated and convergence can be guaranteed by simply assuming that updating and communication between nodes never cease, without any requirements whatsoever on the rate of communication. The proof of convergence is a nice illustration of the analysis of decentralized algorithms where the processors are allowed to perform their computations and to communicate the corresponding results completely independently of one another [5], [6]. The idea is to show that the estimates computed in the distributed asynchronous algorithm are always sandwiched by the estimates computed by the centralized version of the algorithm when started at two different initial conditions, and that both centralized estimates converge to the true distances to the destination node.

Those centralized estimates are denoted by $\bar{\mathbf{D}}_k = (\bar{D}_k(1), \dots, \bar{D}_k(N))$ and $\underline{\mathbf{D}}_k = (\underline{D}_k(1), \dots, \underline{D}_k(N))$, and are the result of the centralized Bellman-Ford iteration (2.1) when it is started with initial conditions $\bar{\mathbf{D}}_0 = (\infty, \dots, \infty, 0)$ and $\underline{\mathbf{D}}_0 = (0, \dots, 0)$, respectively. (The destination node is assumed to be the N th node.) Define the operator [see (2.1)]

$$\begin{aligned} B_i[\mathbf{D}_k] &= \min_{j \in N(i)} [D_k(j) + d_{ij}] \\ &= D_{k+1}(i) \end{aligned} \quad (2.6)$$

if $1 \leq i < N$, and $B_N[\mathbf{D}_k] = D_k(N)$. This operator is monotone in the sense that if $\mathbf{D} \leq \mathbf{D}^*$ (i.e., if $D(i) \leq D^*(i)$, $i = 1, \dots, N$), then

$$B_i[\mathbf{D}] \leq B_i[\mathbf{D}^*]. \quad (2.7)$$

The monotonicity of B_i implies that

$$\underline{\mathbf{D}}_k \leq \underline{\mathbf{D}}_{k+1} \leq \bar{\mathbf{D}}_{k+1} \leq \bar{\mathbf{D}}_k \quad (2.8)$$

and, moreover, it is easy to show that for sufficiently large k

$$\underline{D}_k = \bar{D}_{N-1} = \bar{D}_k \quad (2.9)$$

which is the vector of distances from each node to destination as we saw in the discussion of the centralized algorithm.

In the asynchronous distributed version of the algorithm, it is assumed that each node i keeps at time $t \geq 0$ an estimate of its distance to destination $A_t(i)$, and an estimate of the distance from each of its neighbors $j \in N(i)$ to destination $A_t^j(j)$, which is simply the latest estimate received from node j . In view of (2.8) and (2.9), convergence of the algorithm will follow if we show that for every index k , there exists a time $t_k > 0$ such that for all $t \geq t_k$

$$\underline{D}_k \leq A_t \leq \bar{D}_k \quad (2.10)$$

and for $i = 1, \dots, N - 1$

$$\underline{D}_k(j) \leq A_t^j(j) \leq \bar{D}_k(j) \quad j \in N(i). \quad (2.11)$$

This is shown by induction. If $k = 0$, then (2.10) and (2.11) hold as long as the initial estimates of the decentralized algorithm are nonnegative. Assuming that the induction hypothesis is true for k , the monotonicity of B_i implies that if $t \geq t_k$, then

$$\underline{D}_{k+1}(i) = B_i[\underline{D}_k] \leq B_i[A_t^j] \leq B_i[\bar{D}_k] = \bar{D}_{k+1}(i). \quad (2.12)$$

But $A_t(i)$ is a piecewise constant function of time which only jumps at the updating times of node i , at which times it takes the value

$$A_t(i) = B_i[A_t^j].$$

Therefore, we can write

$$\underline{D}_{k+1}(i) \leq A_t(i) \leq \bar{D}_{k+1}(i) \quad \text{for } t \geq t_k(i) \quad (2.13)$$

where $t_k(i)$ is the smallest updating time of node i which is greater than t_k . Moreover, if we wait long enough after $\max_i t_k(i)$, not only all the nodes will have carried out their first updates after t_k but the result of those computations will have been communicated to their neighbors because of the assumption that updating and communication occur infinitely often. Hence, there exists $t_{k+1} \geq \max_i t_k(i)$ such that for all $t \geq t_{k+1}$ and for all i and j

$$A_t^j(j) = A_s(j)$$

for some $s \geq t_k(j)$ (which depends on t , i , and j). Thus, it follows from (2.13) that

$$\underline{D}_{k+1}(j) \leq A_t^j(j) \leq \bar{D}_{k+1}(j) \quad j \in N(i) \quad i = 1, \dots, N-1$$

completing the induction proof and, therefore, the proof of convergence of the distributed asynchronous Bellman-Ford algorithm.

When the link delays depend on the traffic flows, it is also possible to obtain the optimal routing that solves (2.2) in a distributed asynchronous fashion. Gradient projection algorithms are better suited for this task than the flow deviation method because in the latter method a higher degree of synchronization is required in order for the nodes to use the same step size at each iteration. In the distributed asynchronous implementation of gradient projection optimum routing algorithms, each node broadcasts from time to time the values of its outgoing flows to its upstream neighbors, who in turn pass that information on to their upstream neighbors. In this way, the source keeps estimates at all times of the link flows and can carry out the gradient projection iteration autonomously based on those estimates. The first algorithm based on this idea was due to Gallager [16], who posed an alternative formulation to (2.2), where the unknowns are the fractions of flow routed to each outgoing link at each node, rather

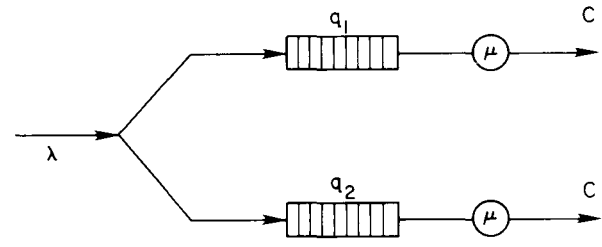


Fig. 4. Queuing model of a node with one incoming link and two outgoing links.

than the path flows. Tsitsiklis and Bertsekas [42] showed the convergence of the distributed asynchronous implementation of gradient projection optimal routing algorithms provided the time between consecutive broadcasts is small enough relative to the speed at which the flows generated by the algorithm change. The approach for showing the stability of this algorithm is very different from the proof of convergence of the distributed Bellman-Ford algorithm where the monotonicity of the dynamic programming mapping implied that the estimates are closer and closer to the solution regardless of the actual sequence of communication and computation times. The idea here is that if the step size of the algorithm is small enough, then the flows change so slowly with respect to the periods between communication times that their evolution is very close to that of the centralized algorithm which uses the unique, true value of each link flow.

B. Dynamic Routing

As mentioned earlier, there are two fundamentally different philosophies to network routing: either viewing it as a "flow" problem in which the traffic of messages is modeled as a "macro"-commodity entering the network as a single entity (static or quasi-static routing), or as an individualized-message path-finding problem in which the traffic is broken down to its constituent elementary units (dynamic routing)—a dichotomy akin to that of statistical/quantum mechanics in physics. Whereas the first approach leads to optimization problems where time plays no role, the essential ingredient of the second approach is the randomness of the time-evolution of the buffers in the network, thus placing dynamic routing within the sphere of stochastic control.

The most elementary instance of dynamic routing is the simple queuing system shown in Fig. 4 which models a node with one incoming link and two outgoing links. It simplifies considerably the dynamics of the message arrival process and of the service time characteristics and ignores processing delay. Thus, the arrival instants of messages over the incoming link are assumed to constitute a Poisson process of constant rate λ . Upon arrival each message is put in the buffer of one of the two outgoing links. This action represents the "control." The buffers are assumed to have unlimited (infinite) capacity and the message lengths are assumed to be random with exponential distribution (an obvious additional simplification) with parameter μ . The two outgoing links have equal capacity of C bits/s. Thus, each link is modeled as a queuing system with exponential service time distribution with parameter μC . It is desired to characterize the optimal control policy that minimizes the average total delay per message based on the observations of the "state" of the system, namely the number of messages q_1 and q_2 in the two buffers. The model, of course, assumes that the head-of-the-line message is dropped from the buffer as soon as the transmission of its last bit is completed.

This model, despite its simplicity, proved to be rather difficult to analyze. For details, see [10]; it is not important to repeat them here. It should suffice to state that the main result, which simply requires that upon arrival a message should join the shortest queue (with arbitrary decision in case the two queues have equal numbers of messages), was hardly surprising. Yet an intricate

argument on the dynamic programming equation (DPE) was needed and there were some counter-intuitive side-results including the relaxation of the Poisson assumption on the arrivals, and the fact that in the incomplete state information case, the *certainty-equivalent control* (i.e., send the message to the *expected* shortest queue) need not be optimum unless both queues have the same number of customers initially.

The optimality of the send-to-shortest-queue (SS) policy in the complete state information case can be proved in a rather strong sense. At all times, the sum ($q_1 + q_2$) and maximum ($\max\{q_1, q_2\}$) of the number of messages in both buffers are stochastically minimized by the SS policy in the sense of the partial order between random variables according to which the random variable X is *stochastically smaller* than Y if $P[X \leq a] \geq P[Y \leq a]$ for all a . The proof of optimality can be obtained by the method of *forward induction* [53], whereby the desired stochastic ordering between the queue sizes under the optimum and an arbitrary policy is shown to be preserved at each transition.

The problem formulation of [10] is one of many related ones (see [8], [9], [22], [24], [33], [38], [54], [55]) which are slightly more complicated but share some fundamental characteristics which, in fact, extend beyond the confines of the routing problem into the areas of priority assignment, resource allocation, and flow control. They are all Markovian decision process (MDP) problems. In the sequel we will describe a fairly general MDP that includes the dynamic routing problem as a special case. In fact, it includes almost all of the queueing control problems that have been studied in connection with communication network issues. We will then outline the solution methodologies that have been used. These include basically: 1) the derivation of optimality conditions from the DPE associated with the corresponding MDP; 2) the use of sample path stochastic dominance arguments, and finally; 3) the reformulation of the MDP as a linear program. We should emphasize, lest the reader be unduly encouraged, that the problems in this area are sufficiently complex, so that only modest results can be generally obtained despite involved arguments and nontrivial machinery. Typically, these results characterize some structural properties of the optimal policy. However, knowledge of such structure is often sufficient to permit close approximation of the actual optimal policy by well-founded heuristics.

Let us recall briefly what an MDP is (for details, see [30]). We need a state description of the process to be controlled. Let S be its state space. When in state $s \in S$, a set A_s of admissible control actions is specified. When action $a \in A_s$ is applied, there is a transition from state s to s' that is governed by the probability distribution $p(s'|s, a)$, and which occurs after a random time τ which is exponentially distributed with distribution denoted by $t(\tau|s, a, s')$. Clearly, p and t together describe the stochastic dynamics of the process to be controlled. Finally, each transition is accompanied by a cost penalty that we denote by $c(\tau, s, a, s')$.

The dynamic routing problem we considered before fits in this formulation easily. In that case, the state space is $S = \{0, 1, 2, 3, \dots\}^2$. An element $s = (q_1, q_2) \in S$ is simply the pair of values of the respective queue sizes. The set of actions A_s is the same for any state and consists of a_1 and a_2 where a_i is the action that assigns an arriving message to the buffer of link i . The distribution p is of trivial form, in that the transitions are deterministic. Assignment of an arrival to queue i augments q_i by one. Note, now, that in addition to the arrival instants, the departure (or service completion) instants are important because they induce state transitions as well. A departure from queue i reduces q_i by one. When a departure occurs there is no meaningful control action that can be applied in this particular problem. The exponential distribution t corresponds to times between arrivals and/or departures.² Finally, the cost rate c must

reflect the delay. By Little's result in queueing theory, we know that the average delay is proportional to the average number of customers in the queue. Thus, $c(\tau, a, s, s')$ can be taken to be simply equal to $(q_1 + q_2)$. This MDP formulation can be extended to encompass more complicated queueing control problems.

Let us return now to the general MDP. We need to specify the notion of a control policy and the optimization criterion. Let us denote by ξ_1, ξ_2, \dots , the state transitions that occur at instants t_1, t_2, \dots . A policy π is a sequence of decision rules π_1, π_2, \dots , where π_n determines the choice of action at the transition time t_n . It can be viewed as a conditional distribution on the set of actions parametrized by the past history of the process.

The optimization criterion that corresponds to the practical case of expected total delay is the long-run average expected cost; namely, if we denote by $V(\pi, i, t)$ the expected cost incurred under policy π , with initial state i , until time t we consider as the optimization criterion the value function

$$V(\pi, i) \triangleq \liminf_{t \rightarrow \infty} \frac{V(\pi, i, t)}{t}.$$

For technical reasons, however, that are well known to optimization specialists, it is easier to establish optimality conditions if we consider, instead, the so-called α -discounted cost, i.e.,

$$V^\alpha(\pi, i) = \int_{t=0}^{\infty} e^{-\alpha t} dV(\pi, i, t).$$

The latter converges to the former as $\alpha \rightarrow 1$ under a variety of stationarity conditions. For technical reasons that will become apparent in the sequel, we will also consider the finite-horizon costs. These are defined in a similar fashion except that we let time extend only to t_n , the instant of the n th transition. If we denote by $V^\alpha(i)$ and $V(i)$ (and also $V_n^\alpha(i)$, $V_n(i)$ for the finite horizon cases) the values of these cost functions when π is chosen optimally, we are led to the following DPE:

$$V^\alpha(i) = \inf_{a \in A_i} \sum_{i'} [c(i, a, i') + \beta(i, a, i') V^\alpha(i')] p(i' | a, i)$$

or

$$V_{n+1}^\alpha(i) = \inf_{a \in A_i} \sum_{i'} [c(i, a, i') + \beta(i, a, i') V_n^\alpha(i')] p(i' | a, i)$$

where

$$\beta(s, a, s') \triangleq \int_0^{\infty} e^{-\alpha t} dt (\tau | s, a, s')$$

and

$$c(s, a, s') \triangleq \int_0^{\infty} c(\tau, s, a, s') dt (\tau | s, a, s')$$

are the discount factor and cost values per transition, respectively.

The DPE is of fundamental importance in the study of MDP's because the value function V^α has the usually convenient properties of convexity, supermodularity, and other forms of monotonicity that lead readily to sufficient conditions for optimality. The difficulty with the analysis of the DPE is that the optimality conditions are heavily problem-dependent and often lead to explosively large numbers of cases to be verified separately. This is especially true for MDP's that arise from queueing models. For this reason, and because of additional difficulties that arise when the state is on the boundaries (see [22]), it became evident that alternative methods of solution were needed.

² A slight modification of the model of transitions, called uniformization, is useful in that it introduces dummy transitions from a state into itself; thus, some situations which introduce nonessential complications can be handled without departure from this discrete transition time formulation.

One alternative method that has received attention recently and which produced successful results in problems of queueing control (akin to the routing problem) is a probabilistic method called sample-path or stochastic dominance. This method bypasses completely dealing with the value function. Instead, it focuses directly on seeking the optimal policy. Let G be the class of admissible policies. If we suspect that the optimal policy π has a property p , then we can proceed as follows in order to prove that it actually does have that property. Let S be a subset of G , to which we know the optimal policy belongs. We consider a subset of policies $S_p \subset S$, all elements of which have the property p . For every $\pi \notin S_p$, we attempt to construct a policy $\tilde{\pi}$ which outperforms π . If we succeed, we must conclude that the optimal policy belongs to S_p . In constructing $\tilde{\pi}$ we often need to engage in a careful reorganization of the underlying probability space in order to align the sample paths properly, so that the comparison of the two policies can be made for every sample path. This procedure is full of risks and extreme care is required to avoid faulty arguments. Note, also, that to apply this method usefully, we must have “guessed” the properties of the optimal policy correctly. Thus, at best, it is a method to verify the validity of our conclusions, rather than a method that leads us to the right conclusions.

Successful use of the stochastic dominance approach was made in [52] and [50] where a problem that is dual to the problem of dynamic routing was studied. Specifically, in a two-server queueing system in which the two servers have unequal service rates, we wish to determine whether and when the slower server needs to be activated if we are interested in minimizing the usual total expected delay function. That the optimal policy has a threshold form (namely that the slower server must be activated when the queue size exceeds a crucial value) was proven in [29] via the DPE method. However, the alternative proof via the arguments of stochastic dominance was much simpler and led to a generalization of the result to cases of nonexponential arrivals and/or service, that could not have been easily accomplished by means of the DPE method.

Another successful use of the stochastic dominance method has been noted in [2]. In this case the problem of optimally choosing which customer to serve next in a single queueing system was considered under the constraint that each customer must begin (or terminate) service by an individually assigned random deadline or else it is dropped from the system. The cost criterion is then to minimize the expected number of lost customers. It was proven that scheduling the customer with shortest time to extinction minimizes this cost.

Although these problems differ from routing, the model structures are quite similar, and it has been observed that, usually, queueing control problems with such structural similarities can be studied equally successfully.

The third method, which was first used in [38] in the study of a specific queueing control problem, and which has been broadly extended recently in [51], is the linear programming approach. Almost any queueing control problem that can be formulated as a MDP (therefore the problem of dynamic routing, as well) can be converted to an equivalent linear program (LP). The advantages of this conversion are that it is problem-independent and it leads occasionally to successful study of semi-Markov decision problems as well. Furthermore, it facilitates considerably the characterization of optimal solution properties. Here is how this equivalence can be demonstrated.

Let us concentrate on an MDP under a finite-horizon, discounted cost formulation.³ We shall consider a queueing model with state dynamics given by

$$x_{k+1} = x_k + \xi_{k+1} z_{k+1}.$$

³ The reason that we cannot work directly with infinite horizons is the possibility of so-called duality gaps in linear programming theory with infinite-dimensional variables.

Here, x_k denotes the state at t_k (the instant of the k th transition), ξ_k represents that transition, and z_k represents the control action at that transition. The transition ξ_k can represent an arrival or a departure as an increment of the state. The control z_k is conveniently defined to enable ($z_k = 1$) or disable ($z_k = 0$) a transition. For example, in the routing model discussed at the beginning of the section, the state is equal to a two-dimensional vector of queue sizes, and the transition corresponding to sending an arriving message to the first queue would be represented by $\xi_k = [1 \ 0]^T$. Indeed, a variety of queueing control problems (in fact, the vast majority of those that have been considered in connection with communication network problems) can be so represented.

Note that the crucial aspect of this state equation is the *linear* dependence on the controls. Note also that usually the cost function is linear in the state (since the usual cost criterion is the expected delay which is coupled to the queue sizes, and hence the state, by Little’s result). Consequently, the cost is linear in the controls. The minimization of the cost over the set of control trajectories is constrained since the state equation must be satisfied and the state must always belong to an admissible set (typically, a set of vectors with integer-valued coordinates belonging to given ranges). Thus, the constraints are also linear in the controls, and the problem is easily formulated as an LP. There are, however, two points that require attention. First, the controls are integer-valued, i.e., $z_k \in \{0, 1\}$. Second, in the MDP the vectors ξ_k are random and depend on past history.

The first problem is taken care of in one of two ways: by construction or by use of a property of the constraint matrix of the linear program, called unimodularity. The construction method involves using a noninteger optimum control whose quantized version satisfies the MDP optimality conditions (see [38], [51] for details). The use of unimodularity involves a well-known result in the theory of integer linear programming (e.g., [34]): if the constraint matrix of an LP is integer-valued and totally unimodular (i.e., each of its sub-determinants is $+1$, -1 , or 0), then all the vertices of the feasible polytope are integer-valued. Therefore, no further restrictions are needed to guarantee that the solution of a conventional LP will result in the integer-valued optimal control. Fortunately, in many queueing problems of interest (including the dynamic routing problem), the constraint matrix is indeed totally unimodular.

The second problem is easily taken care of by thinking of the z_k ’s as functions from the sample space Ω to the action space. Thus, the cost criterion can be written as a functional on the underlying probability space.

Let $z_k(\omega_k)$ represent the control action at the k th transition, where ω_k denotes the random “history” until the k th transition. We have

$$x_{k+1}(\omega_{k+1}) = x_k(\omega_k) + z_{k+1}(\omega_{k+1}) \xi_{k+1}(\omega_{k+1}).$$

Let S and Z be the set of admissible states and controls, respectively. The β -discounted, n -step, expected cost under policy z and initial condition x is given by

$$J_n^\beta(x, z) = E_z \sum_{k=0}^{n-1} \beta^k L(z_k)$$

where

$$L(z_k) = c^T x_k + d^T z_k$$

(c and d denote constant column vectors). This is a cost function that is adequately general. For example, in a pure resource allocation problem without blocking or rejection of messages we have $d = 0$, while in pure blocking problems we take $c = 0$. The state equation, after repeated iterations, yields

$$x_k(\omega_k) = x + \sum_{j=1}^k z_j(\omega_j) \xi_j(\omega_j), \quad k > 0.$$

Therefore,

$$J_n^\beta(x, z) = E_x \sum_{k=0}^{n-1} \beta^k \left\{ c^T x + c^T \sum_{j=1}^k z_j \xi_j + d^T z_k \right\} \\ = \frac{1-\beta^n}{1-\beta} c^T x + E_x \sum_{k=1}^n \beta^k \left\{ \sum_{j=1}^k c^T z_j \xi_j + d^T z_k \right\}.$$

But

$$E_x(z_k) = \sum_{\omega_k} z_k(\omega_k) Pr(\omega_k).$$

Hence

$$J_n^\beta(x, z) = \frac{1-\beta^n}{1-\beta} c^T x + \sum_{k=1}^n \sum_{\omega_k} \gamma_k(\omega_k) z_k(\omega_k)$$

where $\gamma_k(\omega_k)$ is a known function that depends on $Pr(\omega_k)$, c , ξ_k , and β^k . Consequently, the MDP is equivalent to

$$\min_{z_k} \sum_{k=1}^n \sum_{\omega_k} \gamma_k(\omega_k) z_k(\omega_k)$$

subject to

$$\left(x + \sum_{j=1}^k z_j(\omega_j) \xi_j(\omega_j) \right) \in S$$

which is a conventional LP where the initial condition plays the role of a parameter, the sensitivity with respect to which can be studied by the well-developed theory of sensitivity analysis of linear programming [15].

In conclusion, we see that the MDP is converted to an equivalent LP under very mild conditions that are usually satisfied by dynamic routing and other queueing control problems. Thus, a third alternative methodology becomes generally available for the study of these problems. Whether to choose from the arsenal the DPE approach, or the LP method, or stochastic dominance tools, depends on the problem and on the, as yet undeveloped, intuition that the investigator should possess.

III. MULTIPLE-ACCESS COMMUNICATIONS

The communication networks considered in the discussion of routing problems in Section II consist typically of a set of nodes connected by point-to-point communication links. Each of these links viewed in isolation can be modeled as a classical communication channel with one sender and one receiver. In this section, we consider multipoint-to-point communication links where several transmitters share a common channel. Multiple-access channels are the basic building blocks of radio networks, satellite communication, and local area networks, and during the last 15 years have attracted the attention of many communication, information, and control theorists.

There is a wide variety of strategies to divide the "resources" of a communication channel among several geographically dispersed transmitters. The simplest methods are those that assign a permanent independent sub-channel to each transmitter (e.g., in frequency division multiple access and time division multiple access); these strategies are easy to analyze and are widely used in practice in situations where the users need to transmit at fairly steady rates. If the transmitters are *bursty* (i.e., the ratio of peak-to-average rate at which the need to transmit is high) those static methods are inefficient since most of the time the channel is underutilized while demand (and induced delay) accumulates at

busy terminal locations. Dynamic channel sharing strategies overcome this problem by allocating channel resources on an on-demand basis. Consistent with the overall spirit of this paper, our goal here is not to review this vast topic, but rather to demonstrate how control theory can play a useful role in its study. Here we wish to single out two multiple access strategies: random access and simultaneous transmission, which are broadly representative of dynamic channel sharing systems and in which control theoretic concepts have played a pivotal role.

In random access communication, the conceptual allocation model is addressed without an effort to exploit the signaling degrees of freedom and the micro-structure of the transmitted messages. For this purpose, a crude channel model is considered, that achieves this separation of the "macro" from the "micro" problem. In simultaneous transmission systems, however, a more refined viewpoint is adopted, by taking the realities of the medium into account, modeling them, and exploiting them.

A. Random-Access

The object of interest here is the so-called *collision channel* model, in which messages (called packets) require one time unit (called slot) for transmission and are sent by a population of users who are synchronized so that their slots coincide at the receiver, but are otherwise uncoordinated and unaware of which and how many users have packets to transmit. If two or more packets are simultaneously transmitted, it is assumed that the receiver is unable to recover any of the messages, and they have to be retransmitted in a future slot. In the ALOHA algorithm, which was developed in the early 1970's [1] at the University of Hawaii and marked the beginning of the area of random-access communication, each packet that has been unsuccessfully transmitted before is transmitted with probability p in the next slot. New packets which have not attempted transmission before are transmitted with probability either 1 or p depending on which version of the ALOHA algorithm is used. In our discussion, we will assume the latter choice.

Under these conditions, and assuming that the number of newly generated packets in each slot is a random variable (with mean λ) independent from slot to slot, the number of packets awaiting transmission (called backlog) is a Markov chain taking values in $\{0, 1, 2, \dots\}$. The central problem is to investigate under what conditions the backlog Markov chain is *ergodic*, i.e., it is stable in the sense that it reaches a steady state in which the periods between the times when there are no packets to transmit are not too infrequent (they have finite expected value). The transition probabilities of the Markov chain are parametrized by the rate of arrival of new packets λ and the retransmission probability p . Whereas λ is fixed and given, p is chosen by the transmitters. Hence, we are dealing with a fairly simple *controlled Markov chain* whose control space is the interval $(0, 1]$. In the original ALOHA algorithm, the control p remained constant and common to all transmitters regardless of the information acquired by listening to the channel, thereby resulting in the open-loop control of the Markov chain. Despite several "proofs" of the stability of ALOHA published during the 1970's, neither the actual system built in Hawaii nor the ideal Markov chain model were stable. The reason why the open-loop system is unstable can be easily understood by considering the backlog drift, $d(n)$, which is defined as the expected increase in the backlog over the next slot when the current value of the backlog is equal to n . It is easy to see that the backlog drift is given simply by the expected number of new packets per slot minus the expected number of successfully transmitted packets in the next slot, i.e.,

$$d(n) = \lambda - [np(1-p)^{n-1}]. \quad (3.1)$$

The drift quantifies the expected evolution of the Markov chain from each state, and therefore it is a valuable tool in analyzing the stability of the chain. For any $p \in (0, 1]$ the term in brackets in

(3.1) goes to 0 as $n \rightarrow \infty$, and hence, the drift is positive and close to λ for sufficiently large backlogs. This implies that when the backlog is large it tends to grow, thereby eliminating any hope for stability. Using standard results, this reasoning can be formalized straightforwardly to prove not only the instability of the open-loop system [11] for all values of λ and p , but the fact that the backlog goes to infinity with probability one [25], [35], [40].

Fortunately, the system can be stabilized by closed-loop control. Let us examine first the case of complete-state information, i.e., each station is informed at the end of each slot of the current value of the backlog and chooses the retransmission probability on the basis of that information. As far as stability is concerned, the best choice of the retransmission probability p is the value that minimizes the drift because that results in the maximum possible arrival rate that guarantees stability (called the throughput). It follows from (3.1) that the optimum value of p is

$$p^*(n) = \frac{1}{n}, \quad n = 1, 2, \dots \quad (3.2)$$

and the resulting drift is

$$d^*(n) = \lambda - \left[1 - \frac{1}{n}\right]^{n-1} \quad (3.3)$$

which is negative for $n > 1$ when $\lambda < e^{-1}$, and is positive for large backlogs when $\lambda > e^{-1}$. Therefore, the throughput of the closed-loop system with complete state information is $e^{-1} = 0.368$. However, the relevance of complete state information feedback is rather limited in practice. This is because the instantaneous value of the backlog is available to each station only if there exists so large a degree of communication among the transmitters that much more efficient algorithms than ALOHA can be used.

The case of partial state information is the problem of interest in practice, since the only feedback available to each station is the outcome (collision, success, empty) of the transmission in each slot. The analysis of the controlled system with partial state information was pioneered by Hajek and Van Loon [20] who proposed a recursive updating law of the retransmission probabilities as a function of the channel outcomes. This feedback policy was shown in [21] to attain the throughput achievable with complete-state information, namely e^{-1} . Those papers and subsequent works have referred to the problem as *decentralized control* of ALOHA, motivated by the fact that each station chooses the retransmission probability autonomously based on the channel feedback. However, it is useful to recognize that the problem boils down to (centralized) stochastic control with one decision maker and incomplete state information because all stations are constrained to use the same retransmission probabilities.

We will review here the proof of stability of the following *certainty-equivalence* closed-loop control:

$$p(\hat{n}) = \frac{1}{\hat{n}} \quad (3.4)$$

where \hat{n} is an estimate of the backlog updated according to

$$\hat{n}_{k+1} = \begin{cases} \max\{1, \hat{n}_k + \alpha\} & \text{kth slot is idle} \\ \hat{n}_k + \beta & \text{kth slot is success or collision.} \end{cases} \quad (3.5)$$

The throughput attainable with this feedback law depends on the constants $\alpha < 0$ and $\beta > 0$. As we will see, there exists a set of choices for those constants that results in throughput equal to e^{-1} .

Unlike the case of complete-state information, the proof of stability is not straightforward because now it is the two-dimensional process formed by the backlog and its estimate $\{(n_k,$

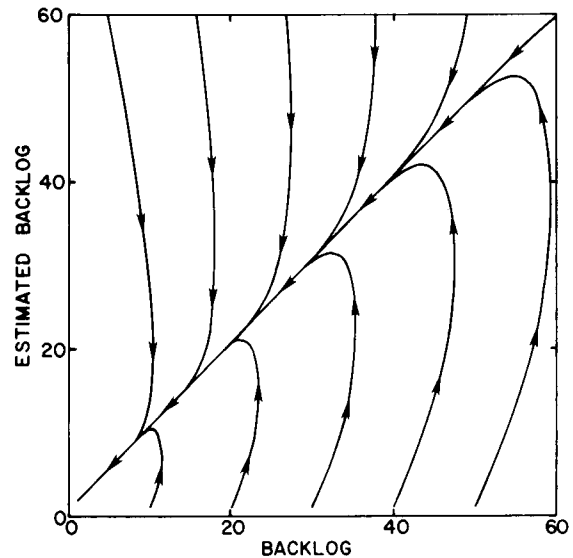


Fig. 5. Drift of (backlog, backlog estimate) Markov process for decentralized control with $\alpha = -1.48$, $\beta = 0.8$, and $\lambda = 0.33$.

$\hat{n}_k)\}_k$ (rather than the backlog itself) which is a Markov process. According to (3.4) and (3.5) the drift of this Markov process is given by

$$\begin{aligned} E[(n_{k+1}, \hat{n}_{k+1}) - (n_k, \hat{n}_k) | (n_k, \hat{n}_k) = (n, s)] \\ = \left(\lambda - \frac{n}{s} \left[1 - \frac{1}{s}\right]^{n-1}, \beta + (\max\{\alpha, 1-s\} - \beta) \left[1 - \frac{1}{s}\right]^n \right) \\ \triangleq (d(n, s), c(n, s)). \end{aligned} \quad (3.6)$$

Contrary to what we saw in the case when the state is known, it is not true that the backlog drift is negative for sufficiently large backlogs. As we can see in Fig. 5, if the estimate is far from the true value, then the backlog may actually tend to increase.

However, at every point in the state space the tendency of the process is to approach the diagonal where the estimate is equal to the true value of the backlog. Furthermore, as Fig. 5 or the analysis of the perfect-state information case shows, the drift along the diagonal is negative. Such a behavior is a strong indication of the stability of the controlled Markov process.

This can be proved using a powerful sufficient condition found by Mikhailov [31] for the stability of a Markov process taking values in $\mathbb{R}^+ \times \mathbb{R}^+$. In essence, Mikhailov's condition states that it is enough to restrict attention to those points of the state space where either the backlog or its estimate are large and at which the drift is *radial*, i.e.,

$$\frac{d(n, s)}{c(n, s)} = \frac{n}{s};$$

then, it is sufficient for stability that the drift point towards the origin at those states. To see that this condition is indeed satisfied for our system, we compute first the asymptotic drifts along the radius $\{(n, s): n/s = \psi\}$ for $\psi \in [0, \infty)$

$$d(\psi) = \lim_{s \rightarrow \infty} d(\psi s, s) = \lambda - \psi e^{-\psi} \quad (3.7a)$$

$$c(\psi) = \lim_{s \rightarrow \infty} c(\psi s, s) = \beta + (\alpha - \beta) e^{-\psi}. \quad (3.7b)$$

It can be checked using (3.7) that if the constants α and β in (3.5) are chosen such that $\beta > 0.23\lambda$ and $\lambda - e^{-1} = \beta + (\alpha - \beta)e^{-1}$, then the drift is radial only at $\psi = 1$ (cf. Fig. 5), where it points towards the origin as long as $d(1) = \lambda - e^{-1} < 0$.

Mikhailov's sufficient condition can be justified constructing a

stochastic Lyapunov function to prove the stability of a Markov process $\{x_k\}_k$ with state space $\mathbb{R}^+ \times \mathbb{R}^+$. To that end, it is advantageous to switch to polar coordinates (r, ϕ) and to define the radial drift $\delta(r, \phi)$ as the projection of the drift along the direction of the point (r, ϕ) and the tangential drift $\mu(r, \phi)$ as the projection of the drift along the direction perpendicular to (r, ϕ) . Denote the asymptotic drifts $\delta(\phi) = \lim_{r \rightarrow \infty} \delta(r, \phi)$ and $\mu(\phi) = \lim_{r \rightarrow \infty} \mu(r, \phi)$ and define the function

$$V(r, \phi) = r\Phi(\phi)$$

where

$$\Phi(\phi) = \exp\left(-C \int_0^\phi \mu(v) dv\right) \quad \phi \in \left[0, \frac{\pi}{2}\right].$$

Note that $V(r, \phi)$ is a candidate Lyapunov function because it is positive outside the origin and $V(r, \phi) \rightarrow \infty$ as $r \rightarrow \infty$. Furthermore, it can be shown [31] that the asymptotic drift of the candidate Lyapunov function is equal to

$$\lim_{r \rightarrow \infty} E[V(x_{k+1}) - V(x_k) | x_k = (r, \phi)] = \Phi(\phi)[\delta(\phi) - C\mu^2(\phi)]. \quad (3.8)$$

Now, under Mikhailov's condition, the asymptotic drifts are assumed continuous on $[0, \pi/2]$ and $\delta(\phi) < \epsilon$ for any phase such that $\mu(\phi) = 0$ (i.e., whenever the drift is radial it points towards the origin), therefore, the constant C can be chosen large enough so that the left side of (3.8) is upper bounded by a negative constant. This implies that $V(r, \phi)$ is indeed a stochastic Lyapunov function and therefore standard results on the stability of stochastic systems [27], [45] can be applied to show the stability of the system.⁴

In some multiaccess environments, the receiver can indeed demodulate reliably one or more packets even in the presence of other interfering packets and the collision channel model no longer applies to those cases. The results reviewed in this section can be generalized to a general channel with *multipacket reception* capability, to show that: 1) the throughput of open-loop ALOHA is equal to the limit of the expected number of successfully received packets per slot as the backlog goes to infinity [17]; and 2) the throughput of closed-loop ALOHA (with either complete or partial state information) is equal to the maximum over v of the expected number of successfully received packets per slot when the number of attempted transmissions is a Poisson random variable with mean v [18].

Returning to the case of the collision channel, the next natural step is to drop the main restriction in the ALOHA algorithm, namely, that all stations use the same retransmission probability. This is done in a class of random-access algorithms referred to as collision resolution algorithms which are characterized by the fact that not only are all blocked packets eventually retransmitted successfully, but all users eventually become aware that these packets have been successfully retransmitted. Contrary to the ALOHA algorithm, the decision whether or not to transmit a packet takes into account the previous history of attempted retransmissions of that particular packet. The introduction of this new dimension into the problem renders Markov chain tools considerably less useful than in the foregoing analysis and converts it into a very difficult decentralized stochastic control problem, for which the optimum throughput remains unknown⁵ despite many efforts.

⁴ Another choice of stochastic Lyapunov function for the specific case of decentralized control of ALOHA can be found in [43].

⁵ The best known algorithm has been shown to achieve a throughput of 0.488 using Howard's policy iteration for sequential infinite-horizon problems [32] or by reduction to a simple optimization problem [48]. On the other hand, it is known that the optimum throughput is upper bounded by 0.568 [44].

B. Simultaneous Transmission

In contrast to random-access communication systems, in simultaneous transmission multiple-access systems, the transmitters send their messages simultaneously, independently, and without monitoring the channel in any way. The most common type of simultaneous transmission system is code-division multiplexing, where each user modulates a preassigned signature waveform known by the receiver.

Specifically, we will assume that in order to send the message $\{b_k(i) \in \mathcal{A}\}_{i=0}^{M-1}$ (i.e., a string of M symbols drawn from a finite set \mathcal{A}), the k th user transmits

$$\sum_{i=0}^{M-1} b_k(i)s_k(t-iT)$$

where $\{s_k(t), 0 \leq t \leq T\}$ is the waveform assigned to the k th user, and T is the symbol period. Then the demodulator receives the sum of the signals transmitted by the K active users embedded in noise

$$r(t) = \sum_{k=1}^K \sum_{i=0}^{M-1} b_k(i)s_k(t-iT-\tau_k) + n(t) \quad (3.8)$$

where the offsets $\tau_{k-1} \leq \tau_k \in [0, T)$ model the fact that the users do not synchronize their transmissions. Then the task of the receiver is to recover the transmitted information strings $\{b_k(i)\}_{i=0}^{M-1}, k=1, \dots, K$. Following [47] we will show how to obtain an optimum multiuser demodulator via dynamic programming. First, denote the MK -vector

$$d = \{d_{k+iK} = b_k(i), k=1, \dots, K, i=0, \dots, M-1\}$$

and the multiuser signal in (3.8)

$$S(t, d) = \sum_{k=1}^K \sum_{i=0}^{M-1} b_k(i)s_k(t-iT-\tau_k) = \sum_{i=1}^{MK} d_i z_i(t) \quad (3.9)$$

where $z_{k+iK}(t) = s_k(t-iT-\tau_k)$.

A reasonable criterion for demodulating the information carried in $S(t, d)$ upon observation of $r(t)$ is to select the MK -vector d that best explains the received waveform in the sense of minimizing the energy of the corresponding noise realization, i.e.,

$$\min_{d \in \mathcal{A}^{MK}} \|S(t, d) - r(t)\|^2. \quad (3.10)$$

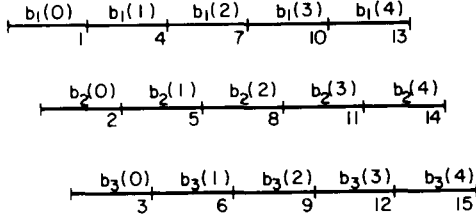
If the noise $n(t)$ is white and Gaussian, then this criterion results in maximum likelihood decisions. Equivalently, the objective is to find the vector that solves

$$\max_{d \in \mathcal{A}^{MK}} \Omega(d) \quad (3.11)$$

where

$$\Omega(d) = 2 \int_{-\infty}^{\infty} S(t, d)r(t) dt - \int_{-\infty}^{\infty} S^2(t, d) dt. \quad (3.12)$$

Since the maximization in (3.11) is over a finite set, we could solve it by the brute-force method of evaluating $\Omega(d)$ for each possible argument. However, it is possible to decompose $\Omega(d)$ in a sequential fashion that lends itself to efficient optimization. From

Fig. 6. Symbol epochs for $K = 3$ and $M = 5$.

(3.9) it is immediate to write the first integral in (3.12) sequentially

$$\int_{-\infty}^{\infty} S(t, \mathbf{d}) r(t) dt = \sum_{j=1}^{MK} d_j y_j \quad (3.13)$$

where

$$y_j = \int_{-\infty}^{\infty} z_j(t) r(t) dt. \quad (3.14)$$

This implies that the objective function (3.12) depends on $r(t)$ only through the quantities $\{y_j\}_{j=1}^{MK}$, which are obtained by correlating $r(t)$ with each of the signature waveforms during each symbol epoch. In order to find an explicit expression for the second integral on the right-hand side of (3.12), which is the energy of the multiuser signal, we will denote

$$R(j, l) = \int_{-\infty}^{\infty} z_j(t) z_l(t) dt. \quad (3.15)$$

It follows immediately from the definition that these coefficients satisfy the following properties.

- 1) $R(k + iK, k + iK) = \int_0^T S_k^2(t) dt \triangleq w_k$.
- 2) $R(k + iK, n + iK) = R(k, n)$ for all i .
- 3) $R(j, l) = 0$ unless $|j - l| < K$.

The first property indicates that each of the diagonal elements of $R(i, j)$ is equal to the energy of one of the K assigned waveforms. The second and third properties can be illustrated by referring to Fig. 6 which represents the symbol epochs of three asynchronous users sending strings of $M = 5$ symbols. Each symbol period in Fig. 6 is labeled with the index of the corresponding component of the vector \mathbf{d} . The second property indicates that the cross-correlations between two signals depend only on their relative location (e.g., $R(4, 6) = R(13, 15)$ in Fig. 6) and the third property states that each symbol only interferes with $2K - 2$ symbols of the other users [e.g., in Fig. 6, $d_9 = b_3(2)$ only overlaps with $d_7 = b_1(2)$, $d_8 = b_2(2)$, $d_{10} = b_1(3)$, and $d_{11} = b_2(3)$]. It follows from these properties that the coefficients in (3.15) can be obtained from the $K \times K$ matrix $\{R(k, n)\}_{k+1, n=1}^K$ whose diagonal elements correspond to the energy per symbol of each user and whose off-diagonal elements correspond to the cross-correlations between the signature waveforms of each pair of users. Using (3.15), the foregoing properties, and letting $\kappa(j) \in \{1, \dots, K\}$ be the modulo- K remainder of j (i.e., for some i , $j = \kappa(j) + iK$), we can write

$$\begin{aligned} \int_{-\infty}^{\infty} S^2(t, \mathbf{d}) dt &= \sum_{j=1}^{MK} \sum_{l=1}^{MK} d_j d_l R(j, l) \\ &= \sum_{j=1}^{MK} d_j \left[w_{\kappa(j)} + 2 \sum_{l=j-K+1}^{j-1} d_l R(j, l) \right] \\ &= \sum_{j=1}^{MK} d_j \left[w_{\kappa(j)} + 2 \sum_{n=1}^{K-1} d_{j-n} g_{\kappa(j)}(K-n) \right] \end{aligned} \quad (3.16)$$

where $g_k(m) = R(k + K, k + m)$. Putting together (3.12), (3.13), and (3.16) we see that we can express $\Omega(\mathbf{d})$ as a sum of MK terms, each of which depends on K components of \mathbf{d} and such that consecutive terms depend on the same components but one. Specifically, we can write

$$\Omega(\mathbf{d}) = \sum_{j=1}^{MK} \lambda_j(x_j, d_j) \quad (3.17)$$

where

$$\lambda_j(x, u) = u[2y_j + u w_{\kappa(j)} - 2x^T g_{\kappa(j)}] \quad (3.18)$$

and x_j is the state of a shift-register $K - 1$ dimensional system

$$\begin{aligned} x_{j+1}^T &= [x_{j+1}(1), \dots, x_{j+1}(K-1)] = [x_j(2), \dots, x_j(K-1), d_j]; \\ x_0 &= 0. \end{aligned} \quad (3.19)$$

It is now apparent that the solution to (3.11) entails solving a *finite-horizon deterministic optimal control problem* with additive costs per stage for the linear system in (3.19), and with a finite admissible control set \mathcal{A} . Therefore, optimum multiuser demodulation is equivalent to a shortest path problem in an M -stage layered directed graph, where at each stage there are A^{K-1} states. This optimization problem can be solved by dynamic programming (e.g., [7]) in backward or forward fashion. In practice, it is necessary to demodulate the transmitted symbols in real-time, and since M is usually a very large integer, it is not feasible to wait until all the observables $\{y_j\}_{j=1}^{MK}$ have been obtained before starting to make decisions. Therefore, a suboptimum version of the forward dynamic programming algorithm is adopted in practice whereby each decision is based on the paths corresponding to the cost-to-arrive function computed a fixed number of steps ahead. This real-time version of forward dynamic programming is known in communication theory as the Viterbi algorithm [12], and was originally devised (without resorting to the dynamic programming framework) for decoding convolutional codes. The maximum-likelihood criterion used in (3.10) is not the only possible optimality criterion. For example, if the objective is to minimize the probability of error for each user, then the multiuser demodulator uses a *backward-forward* dynamic programming algorithm [49] whereby optimum decisions are based on the independent computation of a cost-to-go and a cost-to-arrive function.

IV. OTHER PROBLEM AREAS

Routing and multiple access are not the only problem areas in the field of communication networks which control theory can help formulate, study, and solve. We have deliberately chosen to confine our attention to these two areas in order to get across in a concise manner our belief that the field of communication networks offers a rich selection of applications for control theory. We would feel remiss, however, if we did not even make an attempt to provide a taste of some of the numerous other design and operation issues that, again, bring forth control systems concepts and techniques. For this purpose, and with a conscious effect not to expand in depth but only to describe, we will mention two areas from point-to-point networks and one from radio networks. The first two concern flow control and integrated switching, respectively, while the third concerns the problem of scheduling transmission in multihop networks. Unlike the cases of routing and multiple access, these areas have not yet fully benefitted from the use of control theoretic approaches although such approaches would be very well suited to them indeed.

A. Flow Control

A stark reality in the design of networks is that despite the reduction of the cost of memory, storage at each node is going to be finite. Coupled with another reality, namely that data transmissions on the whole continue to be bursty, it implies that buffer overflow may occur and, along with it, congestion and deadlocks. Flow control is the name we use to describe the collection of measures taken to avoid buffer overflow and highly congested nodes in the network. Congestion and saturation are often the consequences of diverging, unstable behavior. Thus, it is of interest not only to optimize over possible flow control strategies, but to determine their robustness against disturbances or modeling inaccuracies that may lead to unstable behavior.

The control variables in flow control problems are admission (or blocking) probabilities for messages or sessions at the source node. In practice these are often implemented in terms of a bang-bang control strategy known as *window flow control* whereby input ports are allowed to continuously inject messages into the network at the full desired input rate until the number of *unacknowledged*⁶ messages exceeds the value of the "window size" w . A simple, yet unanswered question is, what should the value of w be?

Previous efforts to use control theory tools to analyze optimal flow control problems include [28] and [46] where the optimality of window flow control is proved within the domain of a simplified model, and [39] where dynamic programming value iteration techniques are used to characterize optimal flow control performance. An alternative approach to the flow control problem is to subsume it into the static routing problem considered in Section II-A [19]: suppose that for every source-destination pair a fictitious direct link is added between them. We can then interpret the blocking action of a flow control procedure as a diversion of the blocked portion of the traffic through this fictitious link to the destination. Thus, we can consider that no traffic is blocked. Of course, in order to discourage the use of this fictitious link we must augment the overall delay cost function with a term that penalizes appropriately the use of this link.

B. Integrated Switching

A revolutionary development in the field of networks whose implementation is currently under way is the combination of the capabilities of what have been separately developed in the past and called voice networks and data networks. Voice is a commodity that must meet different requirements than data. For example, speech signals have inherent redundancy that make them quite robust with respect to occasional errors or deliberate compression. At the same time, except in applications of voice messaging, speech signals occur in the context of real-time conversations and, as such, must encounter short and, more importantly, constant delay. On the other hand, data must preserve their integrity and cannot tolerate errors; however, long and variable delays can be often tolerated.

How does one design a single network that can handle such dissimilar commodities with automated procedures? The natural course of events in the last decade or two was to attempt to force data on primarily voice networks or to let voice ride on what were mainly data networks. The literature is full of ideas for baseline integration that are mostly heuristic and difficult to analyze. An attempt to formulate the problem of integrated switching as an optimization problem was presented in [50]. In its simplest form the model is as follows: consider a single node in the network with a single outgoing link on which incoming voice calls and data packets must be multiplexed. Let W be the bandwidth of the outgoing link. Let V be the bandwidth required for the continuous, uninterrupted accommodation of a single voice call. Let,

⁶ Note the implicit assumption of delayed feedback information from the destination to the source node.

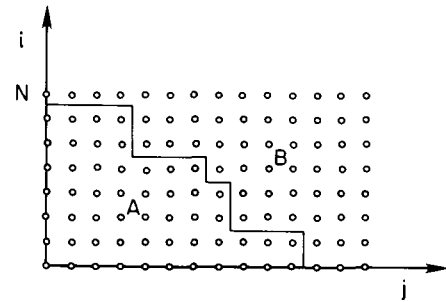


Fig. 7. Switching-type optimum policy for integrated switching.

therefore, $N = W/V$ be the maximum number of calls that can be assigned dedicated circuits simultaneously if no data packets are transmitted. A voice call can either be accepted (and assigned the necessary bandwidth V) or blocked. Data packets can be stored in a buffer facility. If, at a given time, there are i calls in the system, the data packets can be served at the full rate corresponding to the remaining bandwidth $W - iV$. Such a switching architecture represents what has been called the *movable boundary* idea in integration. A natural MDP can be simply formulated as follows: choose the control action of blocking or accepting a call upon arrival in order to minimize the weighted sum of the average data packet delay and the call-blocking probability. If we assume that both arrival streams (voice calls and data) are independent Poisson processes, that the call holding time is exponentially distributed, and that the message lengths are likewise exponential, we can apply the technique described in Section II of converting the MDP to an LP and show that the optimal policy has the useful *switching-type* form. Namely, if i is the number of ongoing calls and j the total number of data messages at the node, the optimal control action should be to block the call in region B of the state space as shown in Fig. 7 and to accept it in region A .

C. Link Scheduling

Let us now turn our attention back to the radio network environment. In Section III the multiple access channel was considered and a number of difficult but interesting control problems were identified. Throughout that discussion, it was assumed that all terminals are within a single transmission hop from the destination. In many radio networks, however, this is not the case. Messages need to be relayed via intermediate nodes to their final destinations. Thus, the familiar problem of routing arises again, except that this time there is a new twist to it. In point-to-point networks, transmissions between different node pairs can take place simultaneously because there are dedicated, "hard-wired" links between the corresponding nodes. In a radio (or, more generally, in a multiaccess/broadcast) environment, if the nodes are densely connected, not all transmissions can take place simultaneously (unless separate dedicated channels or simultaneous transmission signaling techniques (Section III-B) are used). They must be scheduled in time to avoid the interference that would occur otherwise.

It becomes evident that the mere fact that the transmission among a group of nodes must take place one at a time raises the question whether the intended transmissions are routing-wise optimal any more. Several versions of this problem have been studied [3], [23], [36]. In every case and even if the routing problem is sidestepped, we are led to hard combinatorial optimization problems where questions of computational complexity and distributed implementation are of primary importance.

V. CONCLUSION

It should be clear by now that the theory of linear and nonlinear optimization, dynamic programming, stochastic control, stability

analysis, and distributed control have found interesting applications arising in the analysis and design of communication networks. Unlike other complex systems that have been successfully studied by control system theorists in the past (such as chemical plants, flexible aircraft, robot systems, etc.), communication networks stand out in that the commodity to be controlled is information (including its transmission, storage, processing, etc.). This feature, perhaps, misleads and intimidates those who do not feel sufficiently inter-disciplinarian to tackle these problems. We hope that by having selected to present a few examples in which concrete, purely control-theoretic problems can be formulated and have been (or can be) studied successfully, we may encourage attention by the control community to this application area that is especially rich in new challenges.

As stated from the outset, we did not attempt to survey or completely cover the multiple control facets of communication networks. The collection in this paper merely represents an effort to illuminate a few selected problem areas and to show how control techniques apply to them.

REFERENCES

- [1] N. Abramson, "Development of the ALOHANET," *IEEE Trans. Inform. Theory*, vol. IT-31, pp. 119-123, Mar. 1985.
- [2] P. O. Bhattacharya et al., "A (not very) simple dynamic routing problem," in *Proc. 25th Allerton Conf. Contr. Commun. Comput.*, Urbana, IL, 1987, pp. 998-1006.
- [3] D. J. Baker, J. Wieselthier, and A. Ephremides, "A distributed algorithm for scheduling the activation of links in a self-organizing mobile radio network," in *Proc. IEEE Int. Conf. Commun.*, Philadelphia, PA, June 1982, pp. 2F.6.1-5.
- [4] M. Bazarraa and C. Shetty, *Nonlinear Programming: Theory and Algorithms*. New York: Wiley, 1979.
- [5] D. Bertsekas and R. Gallager, *Data Networks*. Englewood Cliffs, NJ: Prentice-Hall, 1987.
- [6] D. P. Bertsekas, "Distributed dynamic programming," *IEEE Trans. Automat. Contr.*, vol. AC-27, pp. 610-616, June 1982.
- [7] D. P. Bertsekas, *Dynamic Programming: Deterministic and Stochastic Models*. Englewood Cliffs, NJ: Prentice-Hall, 1987.
- [8] C. Buyukkoc, P. Varaiya, and J. Walrand, "The μc rule revisited," *Advances Appl. Probability*, vol. 17, pp. 237-238, 1985.
- [9] C. A. Courcoubetis, "Optimal control of a queueing system with simultaneous service requirements," *IEEE Trans. Automat. Contr.*, vol. AC-32, pp. 717-727, 1987.
- [10] A. Ephremides, P. Varaiya, and J. Walrand, "A simple dynamic routing problem," *IEEE Trans. Automat. Contr.*, vol. AC-25, pp. 690-693, Aug. 1980.
- [11] G. Fayolle, E. Gelenbe, and L. Labetoulle, "Stability and optimal control of the packet switching broadcast channel," *JACM*, vol. 24, pp. 375-386, 1977.
- [12] G. D. Forney, "The Viterbi algorithm," *Proc. IEEE*, vol. 61, pp. 268-278, Mar. 1973.
- [13] M. Frank and P. Wolfe, "An algorithm for quadratic programming," *Naval Res. Logist. Quart.*, vol. 3, pp. 149-154, 1956.
- [14] L. Fratta, M. Gerla, and L. Kleinrock, "The flow deviation method: An approach to store-and-forward communication network design," *Networks*, vol. 3, pp. 97-133, 1973.
- [15] T. Gal, *Postoptimal Analyses, Parametric Programming, and Related Topics*. Englewood Cliffs, NJ: McGraw-Hill, 1979.
- [16] R. Gallager, "A minimum delay routing algorithm using distributed computation," *IEEE Trans. Commun.*, vol. COM-23, pp. 73-85, 1977.
- [17] S. Ghez, S. Verdu, and S. Schwartz, "Stability properties of slotted Aloha with multipacket reception capability," *IEEE Trans. Automat. Contr.*, vol. 33, pp. 640-649, July 1988.
- [18] S. Ghez, S. Verdu, and S. Schwartz, "Optimal decentralized control in the multipacket channel," *IEEE Trans. Automat. Contr.*, to be published.
- [19] S. J. Golestaani, "A unified theory of flow control and routing on data communication networks," Ph.D. dissertation, Mass. Inst. Technol., Cambridge, 1980.
- [20] B. Hajek and T. van Loon, "Decentralized dynamic control of a multiaccess broadcast channel," *IEEE Trans. Automat. Contr.*, vol. AC-27, pp. 559-569, June 1982.
- [21] B. Hajek, "Hitting-time and occupation-time bounds implied by drift analysis with applications," *Adv. Appl. Prob.*, vol. 14, pp. 502-525, 1982.
- [22] —, "Optimal control of two interacting service stations," *IEEE Trans. Automat. Contr.*, vol. AC-29, pp. 491-499, June 1984.
- [23] B. Hajek and G. Sasaki, "Link scheduling in polynomial time," *IEEE Trans. Inform. Theory*, vol. 34, pp. 910-917, Sept. 1988.
- [24] J. M. Harrison, "Dynamic scheduling of a multiclass queue: Discount optimality," *Oper. Res.*, vol. 23, pp. 270-282, 1975.
- [25] F. P. Kelly, "Stochastic models of computer communications systems," *J. R. Statist. Soc. B*, vol. 47, pp. 379-395, 1985.
- [26] G. P. Klimov, "Time sharing service systems," *Theory Probability Appl.*, vol. 19, pp. 532-551, Sept. 1974; see also vol. 23, pp. 314-321, June 1978.
- [27] H. Kushner, *Introduction to Stochastic Control*. New York: Holt, Rinehart, and Winston, 1971.
- [28] A. Lazar, "Optimum flow control of a class of queueing networks in equilibrium," *IEEE Trans. Automat. Contr.*, vol. AC-28, pp. 1001-1007, Nov. 1983.
- [29] W. Lin and P. R. Kumar, "Optimal control of a queueing system with two heterogeneous servers," *IEEE Trans. Automat. Contr.*, vol. AC-29, pp. 696-703, Aug. 1984.
- [30] S. A. Lippman, "Semi-Markov decision processes with unbounded rewards," *Management Sci.*, vol. 19, pp. 717-731, 1973.
- [31] V. A. Mikhailov, "Geometrical analysis of the stability of Markov chains in R^4 and its application to throughput evaluation of the adaptive random multiple access algorithm," *Problems of Inform. Transmission*, vol. 24, pp. 47-56, Jan.-Mar. 1988.
- [32] J. Moseley and P. Humblet, "A class of efficient contention resolution algorithms for multiple access channels," *IEEE Trans. Commun.*, vol. COM-33, pp. 145-151, Feb. 1985.
- [33] P. Nain and K. W. Ross, "Optimal priority assignment with hard constraint," *IEEE Trans. Automat. Contr.*, vol. AC-31, pp. 883-888, 1986.
- [34] C. H. Papadimitrou and K. Steiglitz, *Combinatorial Optimization: Algorithms and Complexity*. Englewood Cliffs, NJ: Prentice-Hall, 1982.
- [35] S. Parekh, F. Schoute, and J. Walrand, "Instability and geometric transience of the Aloha protocol," in *Proc. 26th Conf. Decision Contr.*, Los Angeles, CA, Dec. 1987, pp. 1073-1077.
- [36] M. J. Post, P. E. Sarachik, and A. S. Kershenbaum, "A biased greedy algorithm for scheduling multi-hop radio networks," in *Proc. Conf. Inform. Sci. Syst.*, Johns Hopkins Univ., Baltimore, MD, 1985, pp. 564-672.
- [37] R. T. Rockafellar, *Convex Analysis*. Princeton, NJ: Princeton University Press, 1970.
- [38] Z. Rosberg, P. Varaiya, and J. Walrand, "Optimal control of service in tandem queues," *IEEE Trans. Automat. Contr.*, vol. AC-27, pp. 600-610, June 1982.
- [39] Z. Rosberg and I. Gopal, "Optimal hop-by-hop flow control in computer networks," *IEEE Trans. Automat. Contr.*, vol. AC-31, pp. 813-822, Sept. 1986.
- [40] W. Rosberg and D. Towsley, "On the instability of slotted-ALOHA multiaccess algorithm," *IEEE Trans. Automat. Contr.*, vol. AC-28, pp. 994-996, Oct. 1983.
- [41] R. E. Tarjan, *Data Structures and Network Algorithms* (CBMS-NSF Reg. Conf. Series in Appl. Math. no. 7). Philadelphia, PA: SIAM, 1983.
- [42] J. N. Tsitsiklis and D. P. Bertsekas, "Distributed asynchronous optimal routing in data networks," *IEEE Trans. Automat. Contr.*, vol. AC-31, pp. 325-332, Apr. 1986.
- [43] J. N. Tsitsiklis, "Analysis of a multiaccess control scheme," *IEEE Trans. Automat. Contr.*, vol. AC-32, pp. 1017-1020, Nov. 1987.
- [44] B. S. Tsybakov and N. B. Likhanov, "An improved upper bound on capacity of the random multiple-access channel," *Problemi Pederachi Informatsii*, vol. 23, pp. 64-78, 1987.
- [45] R. Tweedie, "Sufficient conditions for ergodicity and recurrence of Markov chains on a general state space," *Stoch. Proc. Appl.*, vol. 3, pp. 385-403, 1975.
- [46] F. Vakil and A. Lazar, "Flow control protocols for integrated networks with partially observed voice traffic," *IEEE Trans. Automat. Contr.*, vol. AC-32, pp. 2-14, Jan. 1987.
- [47] S. Verdu, "Minimum probability of error for asynchronous Gaussian multiple-access channels," *IEEE Trans. Inform. Theory*, vol. IT-32, pp. 85-96, Jan. 1986.
- [48] —, "Computation of the efficiency of the Mosley-Humblet contention resolution algorithm: A simple method," *Proc. IEEE*, vol. 74, pp. 613-614, Apr. 1986.
- [49] S. Verdu and H. V. Poor, "Abstract dynamic programming models under commutativity conditions," *SIAM J. Contr. Optimiz.*, vol. 4, July 1987.
- [50] I. Viniotis and A. Ephremides, "Optimal switching of voice and data at a network node," *Proc. 26th CDC*, Los Angeles, CA, Dec. 1987, pp. 1504-1507.
- [51] J. Viniotis, Ph.D. dissertation, Univ. Maryland, College Park, MD, 1988.
- [52] J. Walrand, "A note on optimal control of a queueing system with two heterogeneous servers," *Syst. Contr. Lett.*, vol. 4, pp. 131-134, 1984.
- [53] —, *An Introduction to Queueing Networks*. Englewood Cliffs, NJ: Prentice-Hall, 1988.
- [54] R. Weber, "On the optimal assignment of customers to parallel servers," *J. Appl. Prob.*, vol. 15, pp. 406-413, 1978.
- [55] Z. Wu, P. B. Luh, S. Chang, and D. A. Castanon, "Optimal control of

a queuing system with two interacting service stations and three classes of impatient tasks," *IEEE Trans. Automat. Contr.*, vol. 33, pp. 42-49, 1988.



Anthony Ephremides (S'68-M'71-SM'77-F'84) was born in Athens, Greece, in 1943. He received the Ph.D. degree in electrical engineering from Princeton University, Princeton, NJ, in 1971.

He has been with the University of Maryland, College Park, since 1971. He has also spent semesters on leave at M.I.T., the University of California, Berkeley, and ETH, Zurich. He is active in professional consulting as President of Pontos, Inc. Currently, his research interests lie in the areas of communication systems: performance analysis;

modeling; optimization; simulation; and design.

Dr. Ephremides is Director of Division X of the IEEE and has served as

President of the Information Theory Society and on the Board of Governors of the Control Systems Society. He has been Associate Editor of the IEEE TRANSACTIONS ON AUTOMATIC CONTROL and General Chairman of major IEEE Conferences.



Sergio Verdú (S'80-M'84-SM'88) was born in Barcelona, Catalonia, Spain, in 1958. He received the Telecommunication Eng. degree from the Polytechnic University of Barcelona in 1980 and the Ph.D. degree in electrical engineering from the University of Illinois at Urbana-Champaign in 1984.

Upon completion of his doctorate he joined the faculty of Princeton University, Princeton, NJ, where he is currently an Associate Professor of Electrical Engineering. His current research

interests are in the areas of multiuser communication and information theory.

Dr. Verdú is a recipient of the National University Prize of Spain, the Rheinlein Outstanding Junior Faculty Award of the School of Engineering and Applied Science at Princeton University, and the NSF Presidential Young Investigator Award. He is currently serving as Associate Editor of the IEEE TRANSACTIONS ON AUTOMATIC CONTROL, and as a member of the Board of Governors of the IEEE Information Theory Society.