Computer Science Department Technical Report
University of California
Los Angeles, CA 90024-1596

PERFORMANCE EVALUATION OF DISTRIBUTED COMPUTER-COMMUNICATION SYSTEMS

Leonard Kleinrock

October 1988
CSD-880084

Performance Evaluation of Distributed Computer-Communication Systems

Leonard Kleinrock[1]
Computer Science Department, University of California
Los Angeles, California 90024-1596, USA

The computer-communication field is rich with extremely challenging problems for the queueing theorist. In this paper, we describe a number of computer-communication applications in wide-area networks, packet radio networks and local area networks. All of these involve sophisticated queueing theoretic models which have led to advanced applications of existing theory and, in some cases, to new methods in queueing theory. A significant component in many of these applications is that of multi-access to a common resource (typically, a communication channel); we devote a significant portion of this paper to multi-access systems. In addition to identifying problems of interest to the queueing theorist, this paper summarizes the latest results for the mean response time of many current computer-communication applications.

1. Introduction

The advent of the digital computer brought about a major rebirth in applied queueing theory. Indeed, since the early 1960's we have seen a sequence of applications in the information processing field which have challenged the capabilities of that theory and which have forced the theory to be extended in new directions. There is no reason to believe that that challenge will subside for some time to come.

The response to these challenges has produced some important extensions in queueing theory. For example, there has been a major (and largely successful) assault on queueing networks of various kinds. Some of the results are quite beautiful, and exhibit robustness in various directions (see [81]). We have also seen the development of some very important approximation techniques. Multi-access and broadcast communication problems have forced us to develop further analytical techniques. Many of the applications lead to coupled queues which involve two-dimensional queueing problems in the simplest cases, and higher dimensions in the usual cases: here too, new approaches in queueing theory have had to be developed (e.g., the solution of Riemann-Hilbert problems). Due to the difficulty of many of these problems, we find

1. This work was supported by the Defense Applied Research Projects Agency under Contract No. MDA 903-82-C-0064.

that the theory has had to tread a very fine line between approximation and attention to detail.

We organize this paper as follows. In Section 2, we discuss the problems inherent in sharing resources in a distributed environment. In Section 3, we discuss response time and 'power' as measures of system performance. Computer network performance analysis, design and buffer sharing are discussed in Section 4. A large number of multi-access schemes are described and evaluated in Section 5 (the largest section of this paper). Finally, in Section 6, a few ideas from the distributed processing field are presented.

2. Distributed resource sharing as a problem in queueing theory

The earliest problems from information processing which we addressed using queueing theory were simple isolated problems (e.g., memory access). Shortly thereafter, the first major class of problems arose in an attempt to analyze and design computer networks [55]. It was recognized immediately that the exact analysis of networks was hopelessly intractable and so some key (and sweeping) assumptions had to be introduced to allow for a solution (see Section 4.1). But this analysis was, in some sense, premature; in the mid-60's, the computer industry was not yet aware of the need for networks, and so very little work continued in this direction at that time. Instead, the advent of time sharing caught the fancy of the data processing industry. It was found that these systems presented another major opportunity for the use of queueing theory to provide the tools for a proper analytic treatment (see Lavenberg [81] for an excellent summary and history of this area). It was only in the late 60's that networking became fashionable and began to attract large numbers of researchers. Then, in the mid-70's, wireless communication systems (e.g., broadcast satellite communications, packet radio, etc.) presented another set of challenges, and once again models, analyses and approximations were developed. The study of local area networks (LANs) followed in the early 80's and continues to occupy our attention up to the present time.

A major driving force in all these developments was micro-electronics, i.e., integrated chip technology. Of course, the major impact of VLSI was to radically drop the cost of computing and this led to advances such as supercomputers and personal computers; these, in turn, have led to parallel and distributed information processing systems. Indeed, distributed systems have emerged during this period in a variety of forms: distributed communications (e.g., packet switching); distributed processing (e.g., multiprocessing); distributed data base; and distributed control (e.g., multi-access communications).

There is a common theme shared by all of these systems, namely, the fact that they are all distributed. It is this that gives rise to some fascinating new problems for the queueing analyst. A distributed system represents a collection of resources (servers) which are provided to serve a set of users (customers). What makes these problems new is that users may not have convenient access to (nor knowledge of) all of these resources.

Let us elaborate on this issue. In a classic single (or multiple) server system, all users have access to all the resources (i.e., any server can serve any

customer) and, in addition, all users can 'see' the queue (i.e., the users themselves can observe which position they currently occupy in the queue). This is an ideal situation in that the server acts as a perfect dynamically shared resource. In a distributed system, we do not ordinarily have the luxury of such a perfectly organized queue. Let us list the ways in which a user may not have proper access to a resource:

1. *The resource may be* **forbidden** *(completely inaccessible) to a user*
   For example, if we had two separate single server systems, the second server is inaccessible to the first server's customer stream. A simple study to quantify the cost of this inaccessibility was published in [82]. The model was that of M identical and independent M/M/1 queues, each operating at a utilization factor, ρ. The quantity $Q = P$[at least one customer is waiting in some queue and at least one server is idle] was found to be equal to

$$Q = 1 - \rho^M + (1 - \rho)^M[\rho^M - (1+\rho)^M]. \tag{2.1}$$

   Fig. 1 shows this function and clearly demonstrates that this distributed system invokes a large cost (as measured by Q) when $M \gg 1$ for all values of ρ inside the unit interval.

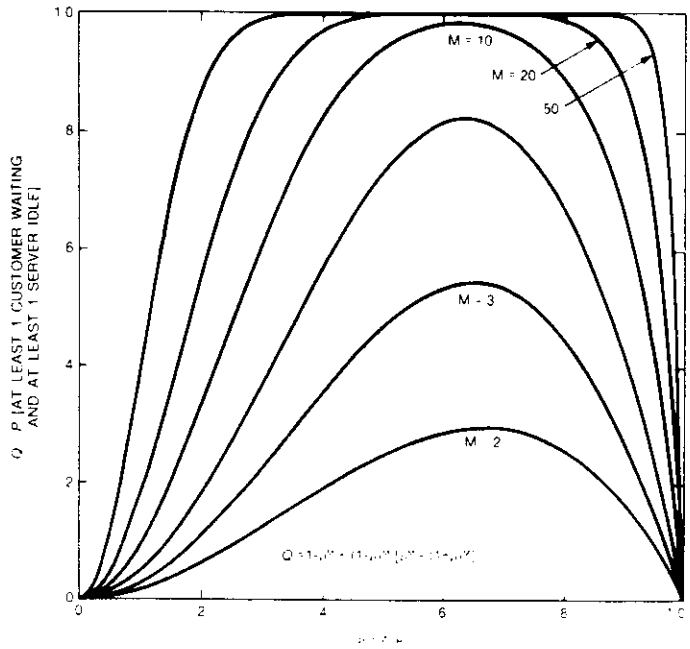


FIGURE 1. *Performance of M M/M/1 queueing systems*

M=5 has been added by the desk editor

A second (simple) way to quantify this cost of inaccessibility is to compare the mean response time, T_M, for these M systems to the mean response time, T_1, of a single M/M/1 system serving the total input from the M streams and with a large server who is M times faster than each of those above. That is, if each of the M separate systems has an input rate λ and a mean service time $\bar{x}$, then the single large server system has an input rate $M\lambda$ and a mean service time $\bar{x}/M$. In both cases, the server utilization is $\rho = \lambda\bar{x}$. From M/M/1 theory [59], we find this ratio to be

$$\frac{T_M}{T_1} = M \tag{2.2}$$

for all $0 \leqslant \rho < 1$. This is another measure of the cost of forbidden resources, and is referred to as the 'economy of scale' principle in [64].

2. *The user may lack* **precise** *information about the state (i.e., the congestion) of a resource*
For example, in a computer network, a path must be chosen for a message to travel. The current congestion situation for channels which are many hops away from the current location of the message may only be known approximately (due, say, to the method by which this information is collected and dispersed).

3. *The user may not have* **current** *information about the state of a resource*
The same example as for (2) applies here. State information may be precise, but may be delayed before it is available to the message.

4. *Immediate and precise information provided to a user about a resource may become* **outdated** *by the time the user has an opportunity to use the resource*
The same example as for (2) applies here as well. The difficulty here is that it takes some time for the message to make its way along the journey to distant channels.

In order to provide users improved access to resources in distributed systems, a number of procedures have been developed. These procedures are referred to as 'multi-access' procedures and fall into three canonical classes:

1. Static assignment algorithms;
2. Dynamic assignment algorithms;
3. Random access algorithms.

(See for example, [64].) Each access method is an attempt to overcome the inaccessibility problem created by the distributed nature of the system. Each is trying, in some sense, to make all users aware of the distributed queue in which they reside so as to allow the users to coordinate their access to the resources. Each method incurs a price for this coordination attempt as follows:

ALGORITHM	EXAMPLE	PRICE
Static	FDMA	Idle resources
Dynamic	Token Passing	Control overhead
Random	Ethernet	Collisions

TABLE 1. *The price for distributed access*

By idle, we mean there is a resource available which some customer could have used, but was disallowed since it was assigned to another user (or group of users) who did not, at that time, need the resource; for example, a collection of separate M/M/1 queues. Control overhead refers to the cost of the dynamic allocation of resources; for example, the time it takes to move the token around in token passing schemes. By collisions, we mean that two or more users attempt to use the same resource simultaneously, in which case, neither gets served; for example, this happens with the Ethernet access scheme when two or more users transmit on the common bus.

With multi-access communications, another key quantity must be taken into account. This quantity involves three important system variables, namely,

C = capacity of the communication channel (bits/sec);
b = the (average) number of bits in the packet to be transmitted;
L = 'length' of the channel (measured in seconds and representing the time it takes from when the first bit is pumped into the channel until that bit appears at the output of the channel, i.e., L is the channel propagation delay).

The key quantity referred to above is denoted by a, and is defined as

$$a = LC/b \tag{2.3}$$

which is readily seen to be the (dimensionless) ratio of the channel propagation delay (L) to the time (b/C) it takes to pump the packet into the channel. It is also equal to the number of packets which can be 'pipelined' in the channel. The value of a turns out to be extremely important in determining system performance and therefore in system design. For example, let us calculate the value of a for four important computer communication cases: a local area network; a long-haul packet-switched network; a broadcast satellite system; and a long-haul fiber optic link. The assumptions and values for a corresponding to these cases are shown in Table 2:

SYSTEM	C (Mbps)	b (bits)	L (microsec)	a
Local Net	10	1000	5	.05
Packet Net	.05	1000	20,000	1
Satellite	.05	1000	250,000	12.5
Fiber Link	1000	1000	20,000	20,000

TABLE 2. *Some typical values for* $a = LC/b$

Thus we see that the value of a varies by a factor of 400,000 over this set of interesting problems. As we said, this value is a key determinant in system design. As an example, whereas the Ethernet local area network access scheme (carrier sense multiple access with collision detection - CSMA/CD - see Section 5.5.4) works well for $a \ll 1$, it would be a disaster in all the other systems.

Considerations such as those listed in this section have motivated a number of new queueing theoretic problems and solutions. We discuss some of these below. In the discussion of these applications, however, we focus on those aspects of distributed resource sharing for which queueing theory has a role to play; we will not address a number of other topics which may be of significance to an application area but which do not have a queueing theoretic component.

3. PERFORMANCE MEASURES

3.1. Response time

We define the response time (sojourn time) of a system to be the interval from when a user generates a request for service until that service is complete. The mean response time is $T = E[\text{response time}]$ where $E[X]$ denotes the expected value of the random variable X. This is the major performance measure used in most queueing systems studies, particularly in computer communications. T is usually given as a function of the system load which measures the efficiency of the server. The load in queueing theory is usually denoted by ρ. In computer communications, we use the same symbol (ρ), but sometimes we use S, particularly when we discuss multi-access systems of certain types. Indeed, when the mean service time, ($\bar{x}$), is not a system design parameter, then we often consider T to be a function only of λ, the system throughput (measured in units of customers per second).

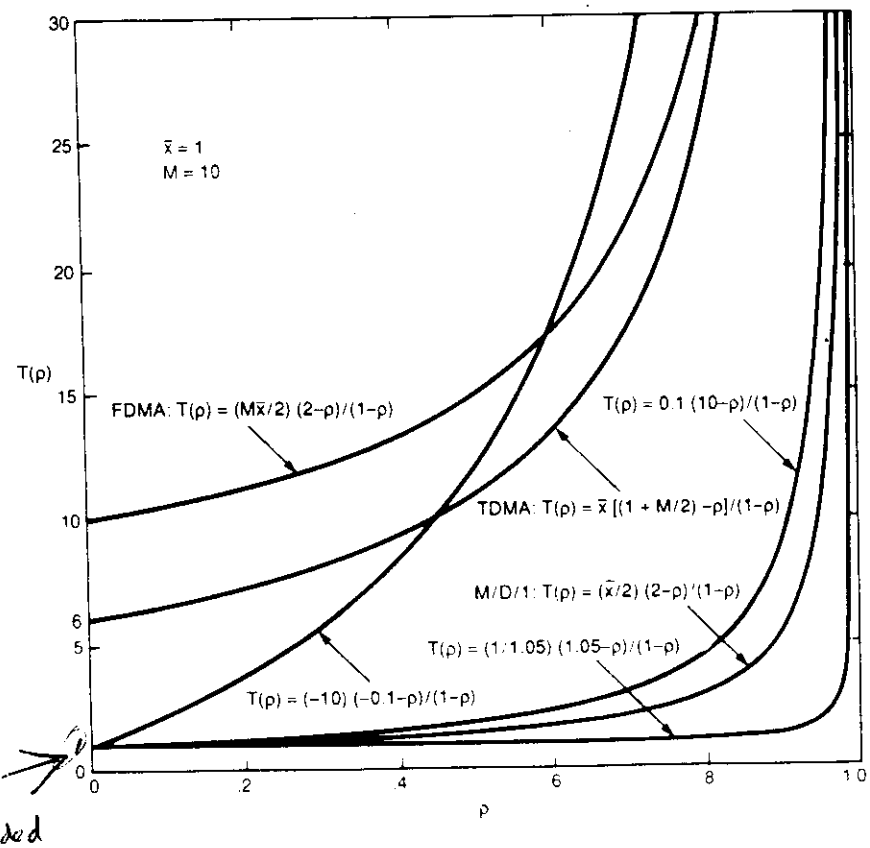
In many of the systems we discuss below, the evaluation of $T = T(\rho)$ is very difficult, and an explicit expression for this function is not available. In [65], a simple approximation for this function is offered which provides three degrees of freedom which allow the analyst to adjust the shape of $T(\rho)$ to match the problem at hand. This approximation is

$$T(\rho) = A\frac{Z - \rho}{P - \rho}, \tag{3.1}$$

and is referred to as the 'ZAP' approximation. The three parameters are used as follows:

1. The value of P (the pole) is equal to the maximum load the system can support (in classical queueing theory, this is $P=1$; in the ALOHA system discussed below, it is $P=1/e$).
2. The value of Z (the zero) is selected to adjust the relative shape of the growing delay function. If $Z = P + \epsilon$ (for small, positive ϵ), then we will have a mean response time function which has a very flat behavior over $0 \leqslant \rho < P$ and a very sharp rise to infinity as $\rho \to P$. On the other hand, if $Z \gg P$ or if $Z \ll 0$, then the mean response time will vary more smoothly over its range.
3. A is selected to match the value of $T(0)=AZ/P$. We note that A and Z must both have the same sign.

See Fig. 2 for typical cases. Often, this expression can be made to exactly match $T(\rho)$ as, for example: M/D/1 ($Z=2$, $A=\bar{x}/2$, $P=1$); FDMA with M users ($Z=2$, $A=M\bar{x}/2$, $P=1$); TDMA with M users ($Z=1+M/2$, $A=\bar{x}$, $P=1$); and others.



I has been added

FIGURE 2. *The ZAP approximation*

3.2. Power

Ideally, one would love to operate the system such that it would exhibit minimum delay and maximum throughput at the same time. In fact, this is possible for D/D/1. However, it is far from the case for most systems of interest. Therefore, one wonders where the 'appropriate' operating point should be. If you hate delay much more than you value efficiency, then you would be happy to operate in the vicinity of ρ near zero (giving minimum delay, but poor efficiency). On the other hand, if you consider efficiency much more important than delay, you would work much closer to the maximum allowable ρ (and suffer a large delay). Where, indeed should you operate?

A quantity known as *power*, and denoted by P (not to be confused with the pole, P, from Section 3.1 nor with the number of processors, P, from Section 6.2), was introduced in [36] and [140] which combined both mean response time and throughput into a single measure. Their definition is $P=\lambda/T=$ throughput/mean response time. Clearly, if this measures your relative preferences, then you would like to maximize P. In [66], this optimization problem was addressed and it was shown that maximum power occurs whenever

$$dT(\lambda)/d\lambda = T(\lambda)/\lambda. \tag{3.2}$$

It is trivial to see that in the $T-\lambda$ plane, this maximum power point occurs at that value of λ where a ray out of the origin is tangent to the response time curve (and where this ray makes the smallest angle with the horizontal axis). See Fig. 3 for an example. In fact, this can be at a point where the mean response time has no derivative.

It turns out for all M/G/1 systems, that maximum power occurs at that λ which produces $\bar{N}=E[\text{number of customers in system}]=1$ exactly! In [68], this observation is discussed in terms of deterministic reasoning. It is found that $\bar{N}=1$ is a common invariant in these problems; this fact is exploited in a detailed study of power in [33] and in [47].

4. Computer networks

An enormous literature on computer networks has been created since the first comprehensive work was published by Kleinrock in 1964 [55]. In that work, he solved many of the basic network analysis and design problems.

4.1. Analysis

The two most significant quantities to solve for in a computer network are γ, the network throughput (the average number of messages per second delivered to destinations) and T, the mean time a message spends in the network (the mean sojourn time in the network). Messages originate at their source node, follow some path through the network, and arrive at their destination nodes. When they reach an intermediate node along their path, they join a queue while awaiting transmission (over, say, channel i) to the next node in their journey. The mean time they spend waiting for, plus transmitting over, channel i is denoted by T_i. Let λ_i denote the average number of messages per second passing over channel i, and let M denote the total number of communication

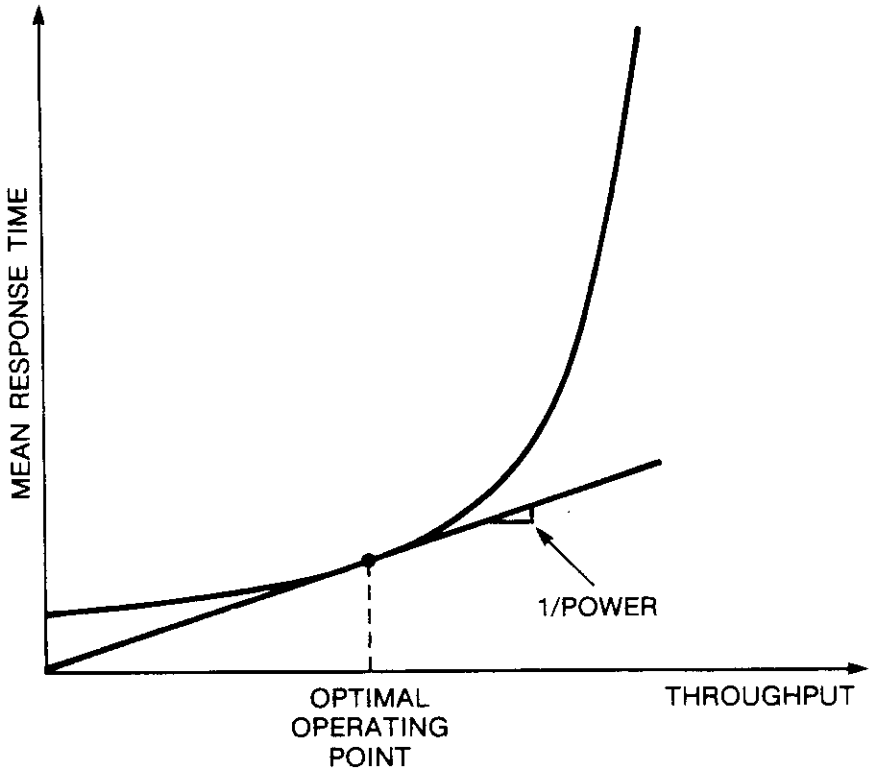


FIGURE 3. *The key system profile and power*

channels in the network. The following expression gives the exact relationship among these quantities [55]:

$$T = \sum_{i=1}^{M} \frac{\lambda_i}{\gamma} T_i. \tag{4.1}$$

This is easily proven with two applications of Little's result [59] as follows. Let $\bar{N} = E$[number of messages in the network]. The first application of Little's result at the network level gives

$$\gamma T = \bar{N}. \tag{4.2}$$

Let $\bar{N}_i = E$[number of messages waiting for or transmitting over channel i]. By assumption, if a message is in the network, it is either waiting for or transmitting over some network channel. Thus

$$\bar{N} = \sum_{i=1}^{M} \bar{N}_i. \tag{4.3}$$

The second application of Little's result at each network channel gives

$$\bar{N} = \sum_{i=1}^{M} \lambda_i T_i. \tag{4.4}$$

From equations (4.2) and (4.4) we arrive at the result of equation (4.1)! Equation (4.1) is the key equation for the *exact* analytical behavior of extremely general computer networks.

In (4.1), the network throughput γ can be found by summing all elements in the (given) traffic matrix; that is, we assume we are given the entries, γ_{jk}, in the traffic matrix (average number of messages entering the network per second with origin j and destination k). The channel traffic λ_i can be calculated from the traffic matrix and the routing procedure. The difficult part of the evaluation is to calculate the mean response time, T_i, for each channel queueing system. The source of the difficulty has to do with the correlation of a message's (i.e., a customer's) service time in successive network nodes.

For example, consider the simple two-node series network shown in Fig. 4. Here we assume that messages arrive to Node 1 from an external Poisson source at a rate of γ messages per second, each with an exponentially chosen length (with an average of $1/\mu$ bits). The messages queue (FCFS) while awaiting transmission (service) from Node 1 to Node 2 over a noiseless communication channel whose capacity is C bits/sec. Thus, the average transmission time on the channel is $1/\mu C$ seconds. When a message is finished being transmitted from Node 1, it then joins a queue while awaiting transmission out of Node 2; we assume that the channel supporting this transmission also has a capacity of C bits per sec. After this second transmission, the message leaves the system. Of interest is the response time (sojourn time) of the message in the network. Clearly, the first node acts as a simple M/M/1 queue. However, the second node presents some difficult problems which come about since the service time of a message on the second channel is *exactly the same* as its service time on the first channel. This not only increases the dimensionality of the state space, but it also introduces a dependence between the sequence of interarrival times and service times in the second node. This two-node problem was posed in [55], and was not solved until 1979 when BOXMA [11] published a solution as the content of his doctorate work under J.W. Cohen. This solution applied only to the case where both channels had the same capacity! His solution was so involved that it provided further evidence that the general network (with arbitrary capacities, paths and topology) would likely never yield to an exact solution.

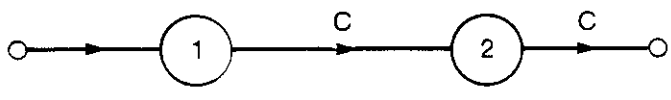


FIGURE 4. *Two-node tandem net*

Recognizing this essential difficulty, Kleinrock introduced a bold 'Independence Assumption' which cracked the problem wide open and admitted a

straightforward solution. This (admittedly false) assumption states that a message's length is *independently* selected from the same exponential distribution each time it enters a node. Using this assumption, Kleinrock could then model the network as an open queueing network (a Jackson network [49]) whose solution for the joint distribution of customer occupancy in each node has a product form. See [81] for a discussion of queueing networks. Considerable experimental evidence supports the use of this model insofar as it has been used to calculate the mean sojourn time of messages in computer networks. Indeed, using this assumption, each channel may be modeled as an independent M/M/1 queueing system whose mean response time is simply

$$T_i = \frac{1}{\mu C_i - \lambda_i}, \quad i = 1,2,\ldots,M. \tag{4.5}$$

Equations (4.1) and (4.5) form the simplest model for the mean response time in computer networks. The Jackson network solution also gives the joint distribution for customer occupancy, but does not give the distribution of sojourn time. WONG [137] found the sojourn distribution for a limited class of networks (networks with no 'overtaking'); see also [54,89,136].

Equation (4.5) may be refined in a number of ways to account for a number of practical factors which we have neglected. These include, for example, the nodal processing time, the propagation delay, the transmission time from the exit node to the attached Host computer, the interference due to control traffic, etc. These refinements may be found in [64].

RUBIN [108] also studied the tandem queue (see Fig. 4). However, he allowed arbitrary channel capacities, an arbitrary number of nodes in the tandem, and a *constant* length for each arriving message. He was able to show that this tandem system could be modeled by an M/G/1 queue and so the Pollaczek-Khinchin results for M/G/1 solve this system completely. Unfortunately, his results do not extend to more complex topologies nor to randomly chosen message lengths.

4.2. Design

There are many aspects to the design of a computer network. Among these are: the choice of node locations; hardware structure of the node; software and protocol design; layered communication structure; topological structure; routing procedure; and capacity of communication circuits. In addition, one must make a number of model assumptions such as: channel cost function (tariff); traffic matrix; message arrival process; and message length distribution. Depending on the formulation of the design problem, the objective function and design constraints can exchange roles. In any case, one must consider: T, the mean system response time; D, the total dollar budget available to spend on the design; and some measure of network reliability. For a complete discussion of these issues see, for example, [64].

We may select two (equivalent) versions of the (optimal) design problem. The first is:

PROBLEM 1

Minimize: $$T = \sum_{i=1}^{M} \frac{\lambda_i}{\gamma} T_i = \sum_{i=1}^{M} \frac{\lambda_i/\gamma}{\mu C_i - \lambda_i}$$

With respect to: Topological design
Routing procedure
Channel capacity assignment

Subject to: $$D = \sum_{i=1}^{M} d_i(C_i)$$
and deliver the traffic specified in the traffic matrix

Where: C_i = Capacity of i-th channel
$d_i(C_i)$ = cost (dollars) to supply C_i units of capacity to the i-th channel
M = Number of channels in net

An alternate (dual) to this problem statement is:

PROBLEM 2

Minimize: $$D = \sum_{i=1}^{M} d_i(C_i)$$

With respect to: Topological design
Routing procedure
Channel capacity assignment

Subject to: $T \leqslant T_{MAX}$
and deliver the traffic specified in the traffic matrix

Where: T_{MAX} is a given allowable maximum mean response time

Clearly, the first problem asks us to minimize network delay at a fixed maximum cost while the dual version asks us to minimize network cost while meeting a maximum network delay criterion. The reliability constraint can be stated in a variety of ways; we do not discuss reliability in this paper.

For design purposes, we usually make the simplest possible model assumptions. Typically, these are:

Exponentially distributed message lengths (mean $1/\mu$ bits)
Independence assumption
Poisson message arrivals (at a rate of γ_{jk} messages/sec originating at node j and destined for node k)

Nodal switch locations preselected
Nodal switches supplied at no cost
Nodal processing times are negligible (infinite nodal processing speed)
Propagation delay is zero (speed of light is infinite)
No control traffic

These assumptions lead us to a Jackson open network model as discussed above. Nevertheless, the optimal design problems stated above are still far too difficult to permit an exact solution. Therefore, the problem is usually decomposed into simpler sub-problems. The 'cleanest' and most commonly discussed sub-problem is the 'Capacity Assignment Problem' which is Problem 1 above, in which the topology and the routing procedure (i.e., the λ_i's) are assumed to be given, and the only issue is to optimally spend D dollars to supply each of the M channels in the topology with a capacity so that the overall mean response time is minimized. The solution depends upon the form of the cost function $d_i(C_i)$. The simplest case considered is the case of a linear cost function, i.e.,

$$d(C_i) = d_i C_i. \tag{4.6}$$

For this optimal assignment, the (minimum) message delay is

$$C_i = \frac{\lambda_i}{\mu} + \left[\frac{D_e}{d_i}\right] \frac{\sqrt{\lambda_i d_i}}{\sum_{j=1}^{M} \sqrt{\lambda_j d_j}}, \quad i = 1,2, \ldots, M. \tag{4.7}$$

D_e is defined below. For this case, the known solution [55] for the minimum T is:

$$T = \frac{\bar{n}}{\mu D_e} \left[\sum_{i=1}^{M} \sqrt{\lambda_i d_i/\lambda}\right]^2, \tag{4.8}$$

with $\bar{n} = \sum_{i=1}^{M} \lambda_i/\gamma$ (is average path length). MEISTER, MULLER, and RUDIN [88] generalized this linear cost-capacity case by changing the objective function from that given in PROBLEM 1 to the following

$$T^{(k)} = \left[\sum_{i=1}^{M} \frac{\lambda_i}{\gamma} (T_i)^k\right]^{1/k} \tag{4.9}$$

where T_i is once again given in (4.5). The form for the optimum capacity and delay may be found in [64,88].

We can generalize the capacity-cost function beyond that of linear to obtain other assignments and other performance equations. However, in all cases, we observe two invariants. First, all optimal capacity assignments require that the i-th channel be provided a minimum capacity of λ_i/μ bits per sec. (otherwise the utilization factor for this channel would exceed unity). Second, a strict lower bound on the number of dollars necessary to provide a feasible solution to this problem for any cost function $d_i(C_i)$ is

$$D_{\min} = \sum_{i=1}^{M} d_i\left(\frac{\lambda_i}{\mu}\right). \tag{4.10}$$

In (4.7) and (4.8), D_e is defined as $D_e = D - D_{\min}$. Without this minimum dollar expenditure, some channel will be overloaded, and then, from (4.1), we see that the mean response time will blow up. Thus, for all capacity-cost functions, the system feasibility condition is $D > D_{\min}$.

A second subproblem assumes that the topology as well as the capacity assignment are given and one wishes to optimally route the traffic through the network to minimize delay; this is the 'Flow Assignment Problem'. That is, we wish to find the optimal λ_i's subject to the constraint that we satisfy the requirements of the traffic matrix. This problem turns out to be that of minimizing a convex function over a convex set; consequently, almost any valley-finding or steepest descent type of algorithm will work. A particularly effective method, known as the Flow Deviation Method, may be found in [31]; this method is based on queueing-theoretic principles.

In order to 'solve' the more general problem, e.g., PROBLEM 2 above, one must resort to sub-optimal methods. We will not discuss these here, except to point out that a number of different heuristic methods have been developed, e.g., [30,38,39,40,64] and these all tend to give the same profile in the throughput versus network-cost plane.

4.3. Buffer sharing

An issue of importance in computer network implementation is that of buffer sharing. As messages (or packets) arrive at a node, they must be buffered while awaiting transmission out of that node. These nodes have a finite number of message buffers and so one would like to allocate these buffers in some appropriate fashion.

The comprehensive model developed in [52] is as follows. Consider a single node with a storage capacity of B buffers. This node is subject to M independent Poisson input streams, the m-th of which has a rate λ_m messages per second, and which must be transmitted out on the m-th channel (of capacity C_m) leaving this node ($m = 1,2, \ldots, M$). Message lengths are exponentially distributed with a mean of $1/\mu$ bits per message. The average message transmission (service) time is, therefore, simply $1/\mu C_m$ seconds on the m-th channel.

Thus, we have M M/M/1 queueing systems which are coupled only through the fact that they share a common finite storage capacity of B buffers. If no space is available for an incoming message, this message is lost; all accepted messages are served in a first-come-first-serve fashion on their respective outgoing channels. Five buffer sharing schemes are considered in [52]. The first is Complete Partitioning (CP) in which the buffer space is permanently partitioned into M separate regions; no sharing at all is allowed in CP. At the other extreme is the second scheme, Complete Sharing (CS) in which all of the storage space is available to any incoming message, independent of which M/M/1 system it belongs to. Neither of these is satisfactory for all loading

conditions. In order to provide some, but not excessive, sharing, a third scheme is introduced known as Sharing with Maximum Queue Length (SMXQ). In this scheme, a maximum is placed on how many buffers may be utilized by each M/M/1 system. SMXQ may not provide for full storage utilization under heavy traffic, and so a fourth scheme, known as Sharing with Minimum Allocation (SMA) is also included. This last provides a guaranteed minimum number of buffers for each of the traffic streams in addition to a common shared pool of buffers. However, the shared pool in SMA may be monopolized by a single stream unfairly, and so the fifth scheme considered is a combination of the last two, namely, Sharing with a Maximum Queue and Minimum Allocation (SMQMA).

The sharing of the finite storage capacity introduces a dependency among the M queueing systems. However, the entire system is a birth-death queueing process [59] whose state vector is $\mathbf{n} = (n_1, n_2, \ldots, n_M)$ where n_m is an integer random variable equal to the number of customers in the node from the m-th queueing system. The equilibrium probability associated with this vector is given by the well-known product form solution for closed queueing networks [4] as follows:

$$P(\mathbf{n}) = P(n_1, n_2, \ldots, n_M) = \begin{cases} C_x \rho_1^{n_1} \rho_2^{n_2} \ldots \rho_M^{n_M}, & \text{for } \mathbf{n} \in F_x \\ 0, & \text{otherwise} \end{cases} \tag{4.11}$$

where $\rho_m = \lambda_m / \mu C_m$. The subscript x refers to one of the five particular buffer allocation schemes described above, and F_x represents the set of allowed states for scheme x. The quantity C_x is simply the probability of an empty system for scheme x and is given by

$$C_x^{-1} = \sum_{\mathbf{n} \in F_x} \rho_1^{n_1} \rho_2^{n_2} \ldots \rho_M^{n_M}. \tag{4.12}$$

Having obtained the joint probability distribution for each scheme, one can then obtain the probability of blocking, the throughput and the average delay for the messages which successfully pass through this shared node.

In [52], the five schemes are compared relative to blocking, throughput and mean response time. It is shown that for most loading situations, some restriction on the shared space is desirable.

LAM and WONG [80] used closed queueing networks to model finite buffer behavior as well. They also used these tools to model window-controlled virtual channels in computer networks.

5. MULTI-ACCESS COMMUNICATIONS

Whenever we have a resource (a communications channel, a computer, a storage facility, a server of any sort) which may be requested by more than one user at a time, then we refer to this as a multi-access situation; clearly, one then requires a method for resolving this conflict, as discussed in Section 2.

In Section 4, we discussed computer communication systems in which the connectivity was provided by point-to-point links. However, there are a number of communication systems which provide a broadcast link in which all

users hear every transmission. Examples of these systems are satellite broadcast systems, packet radio systems, and local area networks. Some authors reserve the term *multi-access* for such broadcast links, and we adopt that distinction in this section. The rules used for resolving conflicts in this environment are referred to as 'multi-access protocols'.

The algorithms in this section (Section 5) represent practical implementations in this environment and each must pay its price to nature (as described in Section 2). (See also [65]).

5.1. Definitions and model assumptions

Here we summarize the usual definitions and assumptions for models of multi-access systems. A number of these are relaxed or altered in specialized papers.

1. The time to transmit a fixed length packet (of say, b bits) is referred to as a 'slot'. This duration is equal to b/C seconds.
2. A 'minislot' is the time required for all users in the broadcast environment to hear any user's transmission; that is, it is the system propagation delay, L seconds.
3. The key parameter 'a' is the ratio of a minislot duration to a slot duration. See (2.3).
4. New arrivals to the total system come from a Poisson stream in the continuous time case. In discrete time we allow a more general arrival process. In both cases, the arrival rate is λ messages per second, or, in its normalized form, S packets per slot.
5. Packets which are not successful must be retransmitted. The retransmissions plus the new transmissions form a Poisson process at a rate G packets per slot. This is known as the 'bold' Poisson assumption and is a gross approximation to the actual state of affairs.
6. A user (terminal) with a packet which must be retransmitted is said to be 'backlogged'.
7. The channel is a broadcast channel, i.e., all users are in range and line-of-sight of all other users (this is said to be a 'single-hop' environment).
8. If more than one user transmits at the same time (in a single-hop environment), a collision occurs. In this case, no transmissions are successful, and the identity of the transmitting users is not available to any users that hear the collision. If exactly one user transmits, it is assumed to be perfectly received by all.
9. When the user population is of infinite size we assume that the users are unbuffered. In this case, at most one packet may be held for transmission by a user.

The notation we use is as follows.

M = number of users
C = channel capacity (bits/sec)
b = (fixed) packet length (bits)

b/C = slot duration = packet transmission time (sec)
$\bar{x}$ = E[message transmission time using the full capacity C] (sec)
β = E[message transmission time] (slots) = $\bar{x}C/b$ (slots)
$\overline{x^2}$ = second moment of message transmission time (sec^2)
λ = arrival rate to all M users (λ/M to each user) (messages/sec)
$\rho = \lambda\bar{x}$ = utilization factor
$S = \rho$ = throughput (packets/slot)
σ^2 = variance of number of messages (bulk) arriving in a slot
L = propagation delay (sec)
$a = LC/b$ (slots)
T = E[message response time] (sec)
$W = E$[message waiting time] $= T - \bar{x}$ (sec)

5.2. Taxonomy

One useful way to distinguish the many multi-access protocols is to divide them into the three groups described in Section 2, namely, static, dynamic, and random. A static assignment algorithm assigns portions of the resource to the users in a predetermined (static) fashion. A dynamic algorithm makes the allocation 'on the fly' according to the current demand of the users. A random algorithm allows users to simultaneously access the resource (e.g., the channel) to a greater or lesser extent (depending on the algorithm); whenever more than one user attempts to transmit in the same bandwidth at the same time in the same range, then a 'collision' occurs, and none of these attempts are successful. Below, we discuss some examples from each of these methods, focusing on the queueing aspects of the problems.

Static assignment schemes perform very poorly at light load (e.g., at $S = 0$, FDMA has a mean response time equal to $M\bar{x}$) but do perform well at high load (indeed, FDMA can achieve a throughput approaching $S = 1$, the maximum throughput for our channel). On the other hand, random access schemes (such as ALOHA) perform beautifully at light load, but very rapidly degrade as the load increases (e.g., ALOHA only supports a throughput of $S <$ $1/e$).

We now examine a number of these schemes broken down into the three groups discussed. Our major objective is to define the algorithm, provide some references, and (when possible) give the mean waiting time W.

5.3. Static assignment algorithms

In these schemes, the input traffic is partitioned into substreams and each substream is assigned a portion of the communication resource on a permanent basis. This assignment may be done in a number of ways, the two most well-known being

FDMA - Frequency Division Multiple Access
TDMA - Time Division Multiple Access

In FDMA, the communication bandwidth is split into, say, M subchannels, each of which is assigned to one of M substreams into which the input traffic has been partitioned. The m-th substream (say with an arrival rate of λ_m messages per second, each with message lengths drawn from some general distribution with a mean message length of $1/\mu$ bits) can only be served by the m-th subchannel (say of capacity C_m bits per second). The load on the m-th queue is equal to

$$\rho_m = \frac{\lambda_m}{\mu C_m} = \lambda_m \bar{x}_m. \tag{5.1}$$

For example, if each input traffic stream is Poisson at the rate $\lambda_m = \lambda/M$, if the messages are all of a fixed size ($1/\mu$ bits), and $C_m = C/M$, then each subsystem is an M/D/1 system where the average time to transmit (service) a message is $M/\mu C$ seconds. Note from the definition of W in Section 5.1 that it equals $T - \bar{x}$ where $\bar{x} = 1/\mu C$, the time to service the message when the full capacity, C, of the channel is used. Thus, using [59], we have:

$$W = T - 1/\mu C = \frac{(2-\rho)M/\mu C}{2(1-\rho)} - 1/\mu C,$$

and so

$$W = \left[\frac{(M-2)(1-\rho)+M}{2(1-\rho)}\right]\bar{x}, \tag{5.2}$$

where $\rho = \lambda/\mu C$ and $\bar{x} = 1/\mu C$.

In TDMA, the time axis is partitioned into an infinite sequence of adjacent intervals. M subsequences are defined, with the m-th such subsequence preassigned for the exclusive use of the m-th substream. Whenever an interval is assigned to the m-th substream, that implies that the entire channel bandwidth is to be used by that substream during that interval. Usually, the intervals are each of a fixed length (i.e., a slot), and the subsequences are derived in a 'round-robin' fashion; that is, every m-th slot is assigned to the m-th substream. This is usually referred to as 'synchronous' TDMA. If the resultant capacity assigned to the m-th substream is, again, C_m, then (5.1) applies to this substream. Again, if the input traffic is Poisson ($\lambda_m = \lambda/M$), the message transmission time is constant (and equal to a slot duration), and $C_m = C/M$, then we see that an arrival must wait, on average, M/2 (slots) until his substream's first assigned slot becomes available. If he finds, on average, $\bar{N}$ messages from his substream upon his arrival, then he must wait an additional $\bar{N}M$ slots, on average, before he finally enters service; this is because each round-robin cycle takes exactly M slots. Thus his mean wait W is

$$W = \left(\frac{M}{2} + \bar{N}M\right)\bar{x},$$

where $\bar{x}$ = slot duration. But, by Little's result, $\bar{N} = (\lambda/M)W$. Thus we find:

$$W = \frac{M\bar{x}}{2(1-\rho)}. \tag{5.3}$$

where $\rho = \lambda\bar{x}$. See (5.12) for a generalization of this result.

In both cases, more general inputs produce a G/G/1 queue for the subsystem. As stated in Section 2, the penalty for static assignment is that some of the system capacity may lie idle when the m-th subsystem goes empty while another subsystem may have a queue of waiting messages.

5.4. *Dynamic assignment algorithms*

5.4.1. Asynchronous TDMA. This is a TDMA scheme in which the time axis is not slotted and is not preassigned, but rather the entire input stream forms a single queue which is served by the channel in some order (typically first-come-first-serve). Nowadays this is referred to as a statistical multiplexer. It is nothing more than a single-server queueing system and represents a perfect resource sharing device.

Of course, we have assumed that all the users can form a single queue with no help from the system, which is an unrealistic assumption in most multi-access environments. In that sense, then, it represents an ideal model for multi-access and gives the minimum wait. In the case of M/D/1, we have the following [59]

$$W = \frac{\rho\,\bar{x}}{2\,(1-\rho)}. \tag{5.4}$$

5.4.2. Polling, Mini-Slotted Alternating Priority (MSAP) and token passing. Polling is a well-known technique in telephony and many of these polling methods have been adopted in data communications. The classical method, *roll-call polling*, is such that a central station coordinates traffic from many users by broadcasting unique user addresses one at a time; when a user hears his address, he may transmit any data he has waiting [44,112,119]. A second technique, *hub-go-ahead polling*, reduces the polling overhead by allowing control to transfer b tween users directly [64,119]. This last technique was the predecessor to the token ring. Indeed, the major application of the mathematical analysis of polling systems has been to local area network performance.

An extensive literature has been developed on polling systems in the last few years. An excellent survey of many of the results from these studies is given in [119] and we refer the reader to this work for a complete list of references. Some of the early basic work on applying these techniques to analyzing computer-communication systems was done by KAYE [53] and also by KONHEIM and MEISTER [73] in which they exploited the theory of random walks.

Let us now present some of the known results for polling systems, after which we apply these results to certain computer-communication systems.

The typical polling model is that of a 'patrolling repairman' in which a server serves a finite number, M, of users by 'walking' from user to user looking for work to be done (messages to transmit). Usually, it is assumed that he patrols the users in a cyclic fashion (this is the assumption we use for most of

this paper). The users may be modeled as unbuffered terminals or buffered terminals. The walking time between users may be fixed or random, with possibly different parameters for user pairs; the walking time in the model corresponds to the polling overhead in computer-communication systems. The amount of work that the server accepts upon finding a busy user may be: *exhaustive* (the server serves a given user until that user's work is emptied); *gated* (the server serves all that work he finds upon arriving to the user, but accepts none of the work that arrives while he is serving that user - this work will be done on his next visit); or *limited* (the server will serve up to a fixed maximum amount of work (or number of customer arrivals at the user) each time he visits a user). New arrivals to a user arrive at a rate λ/M messages per second, the rate summed over all M users being λ messages per second.

Of interest to the analysis is the *polling cycle* (the time from when a given user is polled until that user is polled again) and the *message response time* (the interval from when a message arrives until it is delivered).

Below, we list some of the known results for users that have infinite buffers and which are statistically identical (i.e., all users have the same distributions for the walking time between users, for the arrival process and for the message service time). The additional notation is as follows:

r = E[walking time between two terminals] (sec)
$R = Mr$ = E[walking (polling) time in a polling cycle] (sec)
D = E[polling cycle] (sec)
δ^2 = variance of polling time between two terminals (sec^2)
T_m = E[time spent serving user m in a polling cycle] (sec)

We begin by deriving an expression for the mean polling cycle duration of infinite buffer polling systems with an exhaustive service policy. For more generality in this derivation, we temporarily let the walking time between terminals m and $(m + 1)$ modulo M be equal to r_m and we let λ_m be the arrival rate to terminal m. Since the cycle time is the sum of the polling (walking) time between users and the time spent transmitting messages at each stop in the cycle, we have

$$D = \sum_{m=1}^{M} r_m + \sum_{m=1}^{M} T_m.$$

But T_m is simply the time needed to transmit all messages that arrived in the previous polling cycle (whose mean duration is D). In D seconds, we will have, on average, $\lambda_m D$ arrivals, each of which will take an average of $\bar{x}$ seconds to transmit. Thus,

$$T_m = \lambda_m D \bar{x}.$$

Defining the utilization factor for the m-th user as $\rho_m = \lambda_m \bar{x}$, we have

$$D = \frac{\sum_{m=1}^{M} r_m}{1 - \sum_{m=1}^{M} \rho_m}.$$

In the identical user case, $r_m = r$, $R = rM$, $\lambda_m = \lambda/M$ and $\rho_m = \rho/M = \lambda\bar{x}/M$ and so

$$D = \frac{R}{1 - \rho}. \tag{5.5}$$

This result also applies to the gated service policy [74].

A useful conservation law was published in [56] which expressed a linear constraint on the set of mean waiting times (W_m) for M (work-conserving) priority groups in an M/G/1 environment ($m = 1,2, \ldots, M$). That law states that

$$\sum_{m=1}^{M} \frac{\rho_m}{\rho} W_m = \frac{\sum_{m=1}^{M} \lambda_m \overline{x^2}/2}{1 - \rho} \tag{5.6}$$

where the subscripts refer to the individual priority groups in the obvious way. For the three service policies we consider in this section (and more generally for mixtures of them), there have been developed pseudo-conservation laws as reported in [12,13]; these laws resemble that in (5.6).

Next, we consider a discrete time system (with the time unit equal to one slot) using *exhaustive* polling in an environment with an arbitrary arrival process (with mean $\lambda b/C$, and variance σ^2, for the number of messages (i.e., the bulk size) arriving in a slot), an arbitrary discrete message service time distribution (with a mean $\bar{x}$ seconds and second moment $\overline{x^2}$), an arbitrary discrete walking time distribution (with mean r seconds and variance δ^2) and users with an infinite buffering capacity. The mean waiting time is given by [119]

$$W = \frac{\delta^2}{2r} + \frac{\lambda\overline{x^2} + R(1 - \frac{\rho}{M}) + (\frac{M\sigma^2\beta}{\lambda} - 1)\bar{x}}{2(1 - \rho)} - \frac{b}{2C}. \tag{5.7}$$

We comment that the study of the gambler's ruin problem is the key to analyzing this system [73].

For the continuous time version of the above discrete time problem, but with (non-bulk) Poisson message arrivals at rate λ/M per second to each user, we have [119]

$$W = \frac{\delta^2}{2r} + \frac{\lambda\overline{x^2} + R(1 - \frac{\rho}{M})}{2(1 - \rho)}. \tag{5.8}$$

We now consider the same discrete time system discussed above, but here we adopt the *gated* service policy (as opposed to the exhaustive policy). The mean message waiting time is given by [119]

$$W = \frac{\delta^2}{2r} + \frac{\lambda\overline{x^2} + R(1 + \frac{\rho}{M}) + (\frac{M\sigma^2\beta}{\lambda} - 1)\bar{x}}{2(1 - \rho)} - \frac{b}{2C}. \tag{5.9}$$

Note that the mean wait for the gated system exceeds that of the exhaustive

system by the amount $r\rho/(1-\rho)$.

For the continuous time version of this gated policy, we use the same assumptions as in the continuous time exhaustive problem discussed above and obtain [119]

$$W = \frac{\delta^2}{2r} + \frac{\lambda\overline{x^2} + R(1+\frac{\rho}{M})}{2(1-\rho)}. \tag{5.10}$$

As in the discrete time case, we note that the mean wait for the gated system exceeds that of the exhaustive system by the amount $r\rho/(1-\rho)$.

Let us now consider a discrete time system as above, but with a *limited* service policy. In particular, each time the m-th user is polled, at most one (the limit) message will be transmitted. We find the mean wait to be [119]

$$W = \frac{\delta^2}{2r} + \frac{\lambda\overline{x^2} + R(1+\frac{\rho}{M}) + (\frac{b}{C}+R)(\frac{M\sigma^2\beta}{\lambda}-1) + \lambda\delta^2}{2(1-\rho-r\lambda)} - \frac{b}{2C}. \tag{5.11}$$

The continuous time version of this limited service policy, with (non-bulk) Poisson arrivals, gives the following mean wait [119]:

$$W = \frac{\delta^2}{2r} + \frac{\lambda\overline{x^2} + R(1+\frac{\rho}{M}) + \lambda\delta^2}{2(1-\rho-r\lambda)}. \tag{5.12}$$

Let us now compare the mean wait for the three discrete time systems considered above. The exhaustive policy is given in (5.7), the gated policy is given in (5.9), and the limited policy is given in (5.11). We observe that if $\sigma^2 \geq \lambda/M\beta$ (e.g. the case $\sigma^2 = \lambda/M\beta$ corresponds to Poisson arrivals), then

$$W_{\text{exhaustive}} \leq W_{\text{gated}} \leq W_{\text{limited}}. \tag{5.13}$$

Moreover, if we compare the mean wait for the three continuous time policies in the non-bulk cases, we compare (5.8), (5.10) and (5.12) and find that the inequalities given in (5.13) hold here as well.

For the three continuous cases, we observe that in the limit as the walking time shrinks to zero ($r\to 0$) and $M\to\infty$, such that R and ρ remain constant, we have

$$W = \frac{\lambda\overline{x^2} + R}{2(1-\rho)}. \tag{5.14}$$

Note that this is simply the classic M/G/1 mean wait plus half the mean polling cycle duration.

The polling methods considered above all adhered to a *cyclic* polling order. In [72], a set of *random* polling systems is considered. The assumptions are the same as above (namely, M terminals with infinite buffers, discrete time, constant service time per packet equal to the slot size, general bulk arrival process, arbitrarily distributed walking time, and three service policies - exhaustive, gated and limited). However, the messages are all assumed to be exactly one

packet long (and therefore $\bar{x} = b/C$ seconds, $\overline{x^2} = \bar{x}^2 = (b/C)^2$ and $\beta = 1$ slot). The essentially new component is the polling order. We now introduce $P_m = P$[terminal m is polled next]. This is clearly a memoryless random polling policy. In the symmetrical case, $P_m = p$ and the following results can be obtained [72]. First, we find that the mean duration of the polling cycle, D, for the exhaustive and gated service policies once again is given as in (5.5). Next, we give expressions for the mean wait for the three service policies; we write these equations to allow an easy term-by-term comparison with the cyclic case. For the exhaustive case, we have

$$W = \frac{\delta^2}{2r} + \frac{\lambda(\bar{x})^2 + R(1-\frac{\rho}{M}) + (\frac{M\sigma^2}{\lambda} - 1)\bar{x} + (R-r)}{2(1-\rho)} - \frac{b}{2C}. \tag{5.15}$$

For the gated case, we have

$$W = \frac{\delta^2}{2r} + \frac{\lambda(\bar{x})^2 + R(1+\frac{\rho}{M}) + (\frac{M\sigma^2}{\lambda} - 1)\bar{x} + (R-r)}{2(1-\rho)} - \frac{b}{2C}. \tag{5.16}$$

Lastly, for the limited case, we have

$$W = \frac{\delta^2}{2r} + \frac{\lambda(\bar{x})^2 + R(1+\frac{\rho}{M}) + (\frac{b}{C} + R)(\frac{M\sigma^2}{\lambda} - 1) + \lambda\delta^2 + (R-r)}{2(1-\rho-r\lambda)} - \frac{b}{2C}. \tag{5.17}$$

Note that the result comparing these three expressions with those for the cyclic case (see (5.13)), also holds for this random case. Moreover, the differences among the three mean waits in the cyclic case and in the random case are the same.

As mentioned earlier, these polling system results have been applied to a number of computer-communication problems. In the remainder of this section, we discuss a few such applications.

In [125] roll-call polling is used to model discrete-time message flow from a set of M packet radio terminals to a central receiving station. The central station sends a fixed length poll to each terminal inquiring about traffic. The (one-way) propagation delay is L seconds. The constant walking (polling) time is $r = (2L + T_p)$ where T_p is the time to transmit the polling request; clearly, also, $\delta^2 = 0$. The message packets have a constant length of b bits (and take $\bar{x} = b/C$ seconds to transmit over the channel of C bits/sec) and arrive according to a Poisson process at a rate of λ/M messages per second at each terminal. As in (2.3), $a = LC/b$. The system throughput is simply $S = \lambda\bar{x} = \rho$. For this application, the mean wait is given by (5.7) which yields

$$W = \frac{S\bar{x} + (1-\frac{S}{M})R}{2(1-S)} + \frac{L}{2}. \tag{5.18}$$

If we convert this application to continuous time, then (5.8) gives

$$W = \frac{S\overline{x} + (1 - \frac{S}{M})R}{2(1-S)}. \tag{5.19}$$

Hub-go-ahead polling has been adapted in [69] to create the Mini Slotted Alternating Priority access scheme (MSAP) which was the predecessor of the token ring access method. (The Broadcast Recognizing Access Method (BRAM) [20] is similar to MSAP.) It is implemented by preassigning a cyclic polling sequence to the terminals and synchronizing their clocks. When the m-th terminal finishes transmitting its messages, all other terminals will hear a silent period of L seconds (a minislot). At that point, terminal $(m + 1)$ modulo M will recognize that it is his turn to transmit. If he has data, he transmits it and the process of 'token-passing' continues. If he has no data to send, then he remains silent and after another minislot of silence, terminal $(m + 2)$ modulo M receives his turn, etc. In this case, $r = L$ and again, $\delta^2 = 0$. We then find from (5.7) that the normalized mean wait is

$$W = \frac{S\overline{x} + (1 - \frac{S}{M})ML + (1-S)L}{2(1-S)}. \tag{5.20}$$

The token ring has been treated by Bux [14] by applying (5.8). In the token ring, as in MSAP, a control token is circulated around a ring. Upon receiving a free token, a user has the right to transmit his message (using the exhaustive service policy); if he has any messages, he marks the token as busy and transmits his data on the ring local area network, addressing this data to his intended receiver. When the data returns to him after circulating around the ring, this user removes the data he transmitted and then marks the token as free. Applying our result, we have

$$W = \frac{\lambda\overline{x^2} + (1 - \frac{S}{M})L}{2(1 - S)}. \tag{5.21}$$

In the case of a gated service policy for the token ring, we may use (5.10) to obtain

$$W = \frac{\lambda\overline{x^2} + (1 + \frac{S}{M})L}{2(1 - S)}. \tag{5.22}$$

In the case of a limited service policy, we have from (5.12)

$$W = \frac{\lambda\overline{x^2} + (1 + \frac{S}{M})L}{2(1 - S - r\lambda)}. \tag{5.23}$$

These last two results were also given in [119] and [26]. In [26], De Moraes and Rubin also give the mean wait for the token bus, in which a token passes

along a bidirectional local area network bus among terminals which may not necessarily be adjacent on that bus. These expressions differ from those above only by the propagation delay which now increases to the entire length of the bus rather than the inter-terminal propagation delay used in the ring.

We close this section by listing a few of the many other applications of these polling results to computer-communication problems. Polling with finite storage capacity is treated in [133]. Multiple servers are treated in [96] and [104]. The Fasnet system is evaluated in [45] and also in [130] which also evaluates Expressnet. Automatic repeat request (ARQ) retransmission systems may be found in [75]. Newhall loops (one of the first loops proposed) are discussed in [18] and [19]. The register-insertion ring is analyzed in [46]. Nonsymmetric token rings are treated in [29].

5.4.3. Reservation protocols. Polling schemes (and Tree Access Schemes - see Section 5.5.3) are basically passive schemes in which users wait to be asked if they have any data to send. Reservation schemes allow the user to actively request access to the channel. The idea is that a portion of the channel time or bandwidth is set aside as a reservation subchannel. Users place reservations on the reservation subchannel for space on the data subchannel. Once the reservation is accepted, then the user is guaranteed space on the data subchannel, and everyone in the system knows that he has that space reserved.

There are a large number of different reservation schemes described in the literature. Some of them appear as hybrid schemes which incorporate certain features of static and random assignment algorithms. In this section, we briefly describe a few of these.

One of the first reservation schemes described was that of ROBERTS [107]. Here, the channel is slotted into data slots, with the occasional introduction of a reservation slot. A reservation slot is subdivided into a number of smaller slots (which are used for reserving data slots), and these slots are accessed by the users via slotted ALOHA (see Section 5.5.1). When a reservation makes it past the ALOHA contention, all users hear the request and schedule that user for the number of data slots he requested using the first available unassigned data slots; thus a 'queue in the sky' is maintained and does not require a central controller to do the scheduling.

The Split channel Reservation Multiple Access scheme (SRMA) divides the channel by frequency division into the data and reservation channels [125]. It operates in one of two modes, depending on whether the reservation channel is used as a single channel (which carries both requests for space by the user and replies to these requests which give the user permission to begin transmitting) or as two control channels (one for requests and one for replies which schedule the request in the future). Control signals may compete for space on the request channel by any access scheme (e.g., some random access scheme).

Global Scheduling Multiple Access (GSMA) creates a reservation channel (for communicating to a central controller) and a data channel, both with no collisions using TDMA [83]. This is accomplished by using two subframes in each major frame. Each subframe consists of a number of slots. The control

subframe has one slot assigned to each user for making his reservations. The central controller assigns slots in the data subframe according to the requests.

In a satellite environment, Reservation-ALOHA [24] uses a frame consisting of a fixed number of equal sized slots; the frame is longer than the propagation delay. A user who successfully transmitted in a given slot in the previous frame has the exclusive use of that slot in the next frame. If a slot was either idle or contained a collision in the previous frame, then it is 'up-for-grabs' by any user; such a slot is accessed using ALOHA.

A similar scheme, the Round-Robin Reservation Scheme [7], uses the same frame structure, but guarantees that the number of slots in a frame exceeds the number of users. One slot in each frame is permanently assigned to each user. In addition to sending his data packets, a user appends to each packet a count of how many more packets he has queued. When a slot contains a count of zero, then it is assigned by some collision-free algorithm (in this case, by round robin) to another user. When the original 'owner' of a slot wishes to transmit and another user has been assigned his slot, then the original owner transmits, causing a collision, thereby announcing that he is taking his slot back.

In [10] the Split Reservation Upon Collision (SRUC) access scheme is described. It uses fixed length slots, each of which is partitioned into a data subslot and a control subslot. Ordinarily, the data subslots are in the contention mode, in which case, any user may access them. Whenever a collision occurs, however, all data subslots switch to the reserved mode which means that they will all be used to resolve the conflict only among those users who created the collision; no new users may enter the fray until the conflict is fully resolved. The control subchannel is used by a subset of the conflicting users to assist in the conflict resolution. The Tree access schemes of Section 5.5.3 operate in a similar fashion.

The Priority-Oriented Demand Assignment access method (PODA) uses implicit as well as explicit reservations, priorities, and centralized as well as distributed control [50]. The channel is divided into two subframes - a data subframe and a control subframe. The data subframe contains data and piggybacked control information. The control subframe is used to hasten the delivery of urgent control information. Control frame access can be any of a number of suitable types. The fraction of control versus data subframes can be dynamically adapted to loading conditions.

Group Random Access (GRA) uses a TDMA access method, where groups of users (rather than individual users) share fixed TDMA slots using (possibly) different access schemes for each group [109,110].

5.5. Random assignment algorithms

In these schemes, more than one user may attempt to use the channel at the same time, resulting in a collision; in this case no successful transmission occurs. On the other hand, one of the users may be lucky and find no one else using the channel, in which case he will be successful. Such schemes are discussed in this section.

5.5.1. ALOHA. ALOHA access schemes are, in some sense, the most random. Essentially no constraints are placed on the users; they are allowed to transmit any time they wish. Packets are assumed to be of fixed length (exactly one slot long). The ALOHA access algorithm was introduced by ABRAMSON [1] in an unslotted environment (i.e., users were completely unsynchronized). The relationship between the input S and the channel traffic G (recall we are using the bold Poisson assumption) is

$$S = Ge^{-2G}. \tag{5.24}$$

This leads to a maximum throughput of $S = 1/2e = 0.184...$ (packets per slot) which occurs at $G = 0.5$.

An improvement was recommended by ROBERTS [106] by going to a slotted system in which a minimal constraint was placed on users whereby they could begin transmission only at a slot boundary. That is, time was discretized into intervals exactly one slot in duration. With this modification, the S, G relationship becomes

$$S = Ge^{-G}. \tag{5.25}$$

The maximum throughput is now equal to $S = 1/e = 0.368...$ and this occurs when $G = 1$. At $G=1$, the expected number of transmissions is exactly one (satisfying one's intuition regarding optimality). We saw this optimality point occur in our discussion of power in Section 3.2; it occurs in many other situations as well (see, for example, [138]). We note further that the expected number of transmissions required by a packet is simply G/S; the expected number of unsuccessful transmissions is $(G/S) - 1$. In 1973, ABRAMSON [2] analyzed the throughput for slotted ALOHA systems with a *finite* number of users. He found, among other things, that the maximum throughput profile occurred when $G = 1$, exactly as with the infinite population model.

In 1973, KLEINROCK and LAM [57,76] established the equations for the mean response time of the slotted ALOHA system. (It was no surprise to find that the behavior at light load is excellent, but that it badly degrades as the load increases toward $S = 1/e$).

This analysis for an infinite population slotted ALOHA model used the bold Poisson assumption and assumed that retransmissions were uniformly selected over the next K slots once a user learned that his packet was unsuccessful; among the results obtained there was the following for the mean wait:

$$W = \frac{1-q}{q_t}\left[L + \frac{b}{C}\left(\frac{K+1}{2}\right)\right] + L, \tag{5.26}$$

where

$$q = \left[e^{\frac{-G}{K}} + \frac{G}{K}e^{-G}\right]^K e^{-S}, \tag{5.27}$$

and

$$q_t = \left[\frac{e^{\frac{-G}{K}} - e^{-G}}{1 - e^{-G}}\right]\left[e^{\frac{-G}{K}} + \frac{G}{K}e^{-G}\right]^{K-1} e^{-S}, \tag{5.28}$$

and

$$S = G\,\frac{q_t}{1 - q + q_t}. \tag{5.29}$$

The detailed behavior of slotted ALOHA with a finite population of M unbuffered terminals may be described by a discrete-time Markov chain given by the number of backlogged terminals at the slot boundaries [60,76]. Only when a terminal's buffer is empty may a new packet be generated (by the terminal's external source) and this will occur with probability σ in a slot. The combined input rate into all terminals at time (slot) t is denoted by $S(t)$. For this Markov chain analysis, we define p to be the probability that a backlogged packet retransmits in the next slot (of duration $\bar{x}$ seconds) and often is taken as a given parameter of the problem. Thus, the retransmission delay is geometrically distributed with a mean of $\bar{x}/p$ seconds. We let $N(t)$ be a random variable representing the total number of backlogged terminals at slot t (referred to as the *channel backlog* at t). The vector $[N(t), S(t)]$ is the channel load at time t. For $N(t) = n$ we have

$$[\,N(t),\, S(t)\,] = [\,n,\, (M - n)\,\sigma\,], \tag{5.30}$$

giving rise to what we call the *linear feedback model*. Let

$$p_{ij} = P[N(t+1) = j \mid N(t) = i].$$

Then, these one-step transition probabilities are simply [60,76]

$$p_{ij} = \begin{cases} 0, & j \leqslant i-2, \\ ip(1-p)^{i-1}(1-\sigma)^{M-i}, & j = i-1, \\ [1-ip(1-p)^{i-1}](1-\sigma)^{M-i} + & \\ (M-i)\sigma(1-\sigma)^{M-i-1}(1-p)^{i}, & j = i, \\ (M-i)\sigma(1-\sigma)^{M-i-1}[1-(1-p)^{i}], & j = i+1, \\ \binom{M-i}{j-i}\sigma^{j-i}(1-\sigma)^{M-j}, & j \geqslant i+2. \end{cases} \tag{5.31}$$

One then does the usual Markov chain analysis to solve for the equilibrium distribution for $N(t)$, namely,

$$p_n = \lim_{t \to \infty} P[N(t) = n],$$

by solving the system of equations,

$$p_n = \sum_{i=0}^{M} p_i p_{in}.$$

and, of course, requiring the p_n's to sum to unity.

Given $N(t) = n$, there is no guarantee that the channel input rate, $S(t)$ and the channel output rate, which we denote by $S_{\text{out}}(t)$, will be the same. In fact, given $N(t) = n$, we may solve for $S_{\text{out}}(t)$ (which we denote by $S_{out}(n)$ when $N(t) = n$) to obtain

$$S_{\text{out}}(n) = (1-p)^n(M-n)\sigma(1-\sigma)^{M-n-1} + np(1-p)^{n-1}(1-\sigma)^{M-n}. \tag{5.32}$$

In the limit when $M \to \infty$ and $\sigma \to 0$ (such that $M\sigma = S$), we have the infinite population model in which new packets are generated for transmission over the channel at the constant Poisson rate S (packets per slot); in this case, (5.32) gives

$$S_{\text{out}}(n) = (1-p)^n Se^{-S} + np(1-p)^{n-1}e^{-S}. \tag{5.33}$$

Given that the input and output rates may differ, it was natural [58] to investigate the difference $S(t) - S_{out}(t)$ which has since come to be known as the *drift*; the drift is simply the expected growth of the channel backlog in a slot. If we assume a steady state (more on this later), then, following [128] the expected output rate, $\bar{S}$, is given by

$$\bar{S} = \sum_{n=0}^{M} S_{out}(n)p_n, \tag{5.34}$$

and $\bar{N}_q$ the average number of backlogged terminals may be found from

$$\bar{N}_q = \sum_{n=0}^{M} np_n. \tag{5.35}$$

If we use Little's result, we find the mean wait in this slotted ALOHA system to be

$$W = \frac{\bar{N}_q\bar{x}}{\bar{S}}. \tag{5.36}$$

Since the output and input rate, on average, must be equal in equilibrium, we have

$$\bar{S} = (M - \bar{N}_q)\sigma \tag{5.37}$$

and so

$$W = [(M/\bar{S}) - (1/\sigma)]\,\bar{x}. \tag{5.38}$$

Let us examine the dynamic behavior somewhat further. If we assume the channel backlog is $N(t) = n$, then we may calculate $S_{out}(n)$ as a function of n and $S(n)$. If we further identify the *equilibrium contour* to be the projection of the curve $S_{\text{out}}(n) = S(n)$ onto the $(n,S(n))$ plane (where we denote $S(t)$ by $S(n)$ when $N(t) = n$), then we obtain the curve as shown in Fig. 5.

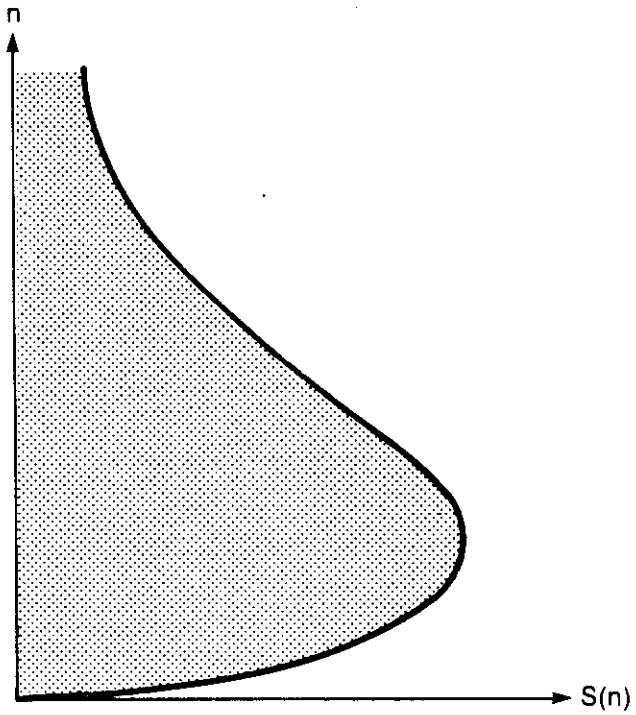


FIGURE 5. *The equilibrium contour*

In this figure, the shaded area corresponds to the region where the drift is negative (i.e., $S_{out}(n) > S(n)$) and $N(t)$ tends to reduce; outside the shaded area, the drift is positive ($S_{out}(n) < S(n)$) and $N(t)$ tends to increase. In Fig. 6, we show four such curves along with the channel *load line*, $n = M - S(n)/\sigma$ (see (5.30)); these curves are obtained by selecting different values for the retransmission probability, p. The system is constrained to lie on the load line. If the system state is in the shaded region, then the backlog decreases down the load line; if not, it rises up the load line. Since we expect the system to lie in the vicinity of the equilibrium contour, and since we require it to lie on the channel load line, then any intersection of the load line with the equilibrium contour is a point to be studied. In Fig. 6a, we observe that there is only one such intersection which we refer to as the *channel operating point*. We also observe that the drift will drive us back to this point if the system takes any excursion along the load line away from this point. Such a point (where the drift pulls us back) is a stable equilibrium point. If the system has only one such stable point, the channel is said to be *stable*. See Fig. 7a for the distribution of backlog for a stable channel. On the other hand, in Fig. 6b, we see a case with three intersections, two of which are stable. The third (middle) point is such that the drift will drive us away from the intersection if we take a small excursion along the load line; such a point is called an

unstable equilibrium point. A channel with two stable points and one unstable point is referred to as a *bistable* or an *unstable channel;* such a channel will spend its time largely in the vicinity of the two stable points and will occasionally drift between the two (as can be seen from the backlog distribution shown in Fig. 7b). Of the two stable points in Fig. 6b, the one with large backlog and small throughput is clearly undesirable and is referred to as the *channel saturation point;* the one with large throughput and small backlog is the (channel operating) point at which we would like to operate. In Fig. 6c, we show the case of an infinite population channel; clearly all infinite population channels are unstable unless some dynamic control policy is used (see below). An *overloaded* or *saturated channel* is one with a single stable operating point which occurs for large backlog and small throughput, as shown in Fig. 6d; its backlog distribution may be seen in Fig. 7c.

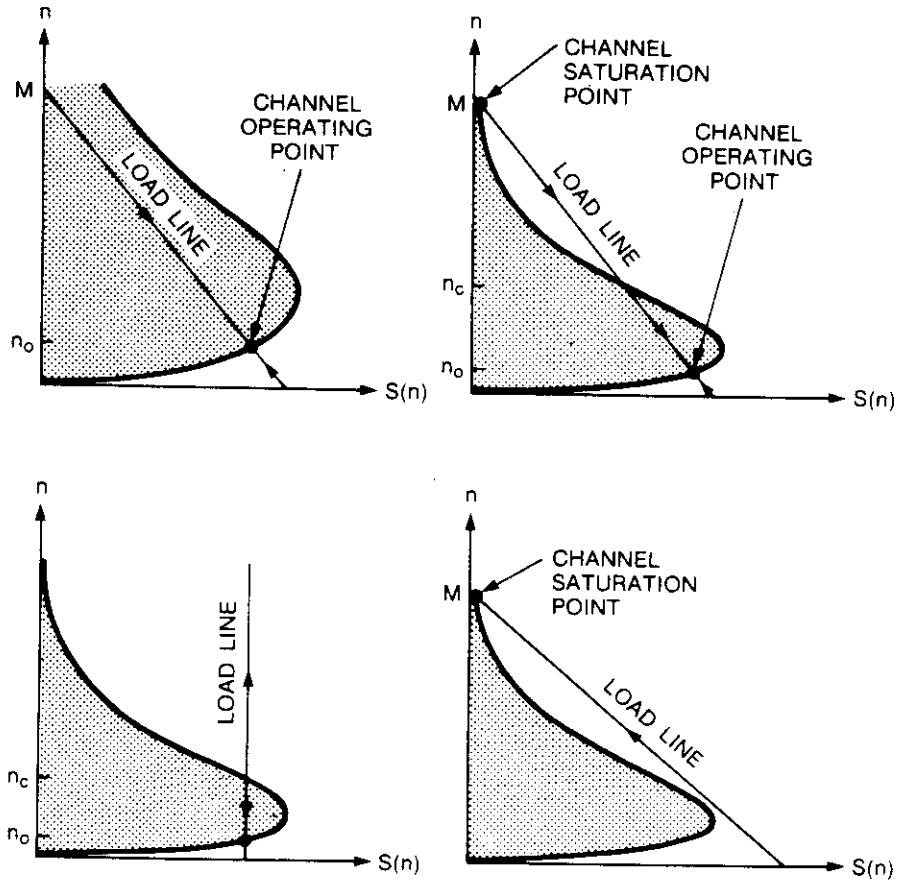


FIGURE 6. *Stable and unstable channels*

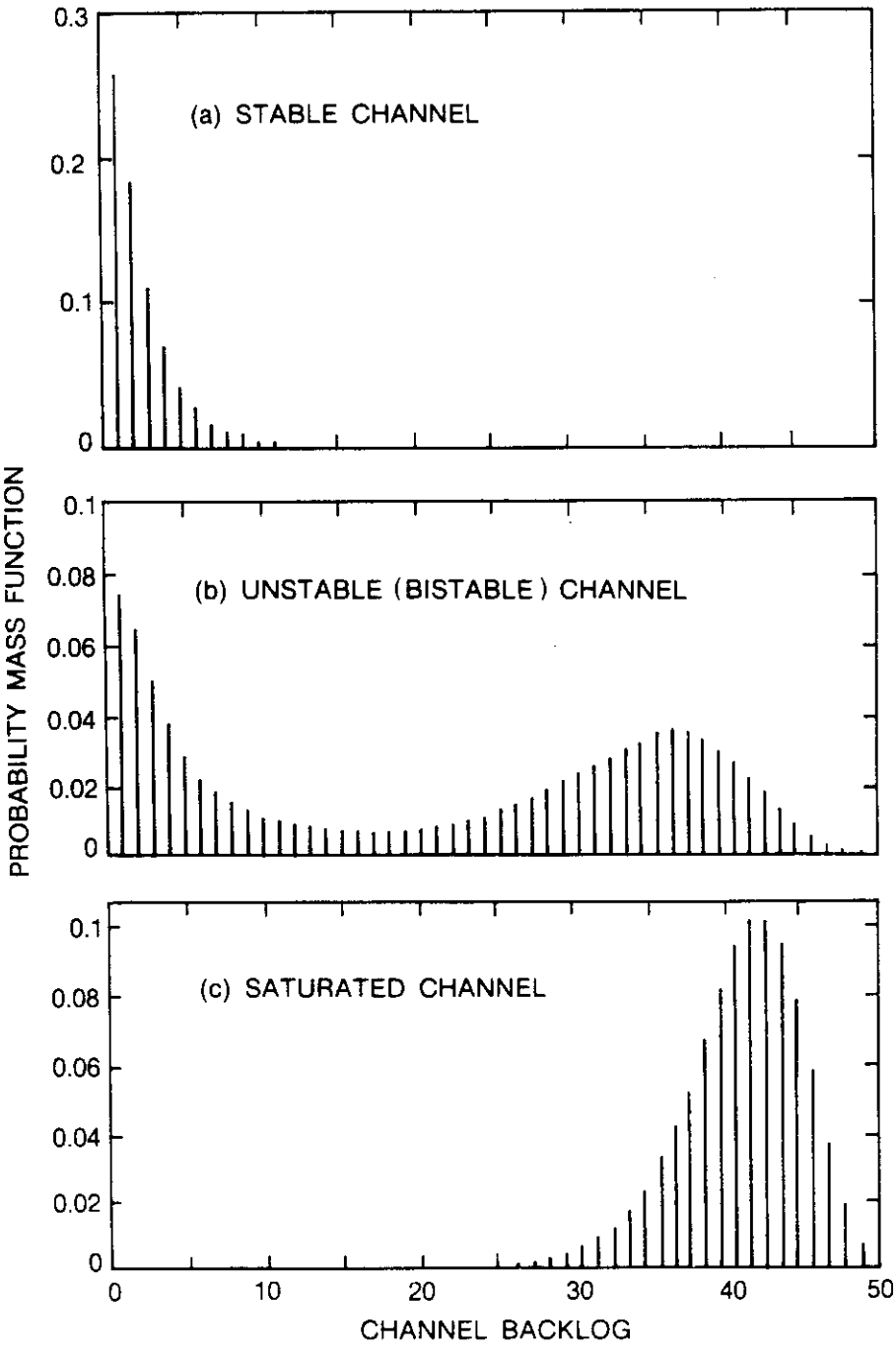


FIGURE 7. *Distribution of backlog*

The ideas regarding stability and equilibrium points described in [60,76] led FUKUDA [32] to develop a method known as *Equilibrium Point Analysis* (EPA): this method was extended by Tasaka who wrote a comprehensive monograph on the subject [122]. The formal idea behind EPA is to assume that the system is always at an equilibrium point (i.e. at a point of zero drift). Furthermore, in the EPA approach, the expectation of any random variable taken with respect to the channel backlog, $N(t)$, is approximated by the value of that random variable at the equilibrium point. In [122], this method is applied to evaluate the throughput and mean delay for a number of multi-access schemes (Slotted ALOHA, Reservation ALOHA, TDMA, SRUC, buffered CSMA/CD - see below, and BRAM).

For an analysis of slotted ALOHA with a finite number of *buffered* terminals, see [111,122].

The fundamental instability of ALOHA systems we have identified above was studied in [27,60,76]. In order to stabilize ALOHA, a number of dynamic control algorithms have been introduced and studied (e.g., [7,41,42,60,64,76,77,78,79,90,105,111]). These stabilization methods typically control either the input rate to the system or the retransmission parameters as a function of the backlog (or an estimate of the backlog). As an example of retransmission control, let us observe the behavior of the otherwise unstable channel shown in Fig. 6b. We see that as we move along the load line (due to statistical fluctuations), the drift is negative as long as $n < n_c$. Suppose we track (perhaps only an estimate of) the backlog and find that $n = n_c$; we are now in danger of entering the (unstable) region of positive drift for $n > n_c$. However, if we now reduce p, the retransmission probability, then the equilibrium contour will change to that shown in Fig. 6a and we will once again find ourselves in a (stable) negative drift situation! Of course, this reduced value of p implies a longer delay between retransmissions which is not a desirable situation. In order to remedy this, we switch back to the original value for p when n falls below n_c giving us negative drift and smaller retransmission delay once again. Thus, by tracking the backlog, n, we dynamically control the channel in a fashion which guarantees stability.

Since ALOHA is fundamentally unstable, the performance of stabilized ALOHA is of interest. The distribution of delay is found numerically in [21] for a stabilized slotted ALOHA system. In [35], an approximate expression for the mean wait of a stabilized slotted ALOHA is given as

$$W = \frac{h}{C}\left[\frac{e - \frac{1}{2}}{1 - eS} - \frac{(e^S - 1)(e - 1)}{S\left[1 - (e^S - 1)(e - 1)\right]}\right] + L. \quad (5.39)$$

5.5.2. The urn model. The URN scheme [67] was devised to behave like ALOHA at light load and TDMA at heavy load; thus, in some sense, it is an adaptive access scheme. Its principle of operation is to divide the population of conflicting users into subgroups each of which is likely to contain only one busy user and then to select one of the subgroups to transmit. In this scheme, we assume that we have a slotted channel supporting M user terminals. At the beginning of each slot, we assume that every user knows N, the exact *number* of users that have packets ready to send; they simply do not know the *identity* of these busy users. Based on this knowledge, what is the optimum way for this information to be used?

If we require all users to behave in a symmetric fashion, i.e., each busy user will transmit into the next slot with an equal probability, say P, then the optimum policy is $P = 1/N$. However, symmetric policies are clearly not optimum as can be seen in Fig. 8. This figure shows the throughput for $M = 2$ and $N = 2$ as a function of the probability of transmission for each of the two users. Indeed, the optimum (asymmetric) policy for this case is for one of the busy users to surely transmit ($P_i = 1$) and for the other to keep quiet ($P_j = 0$) where $i,j = 1,2$, and $i \neq j$. However in a distributed environment it is difficult to select a single busy user when one only knows how many in the total population are busy.

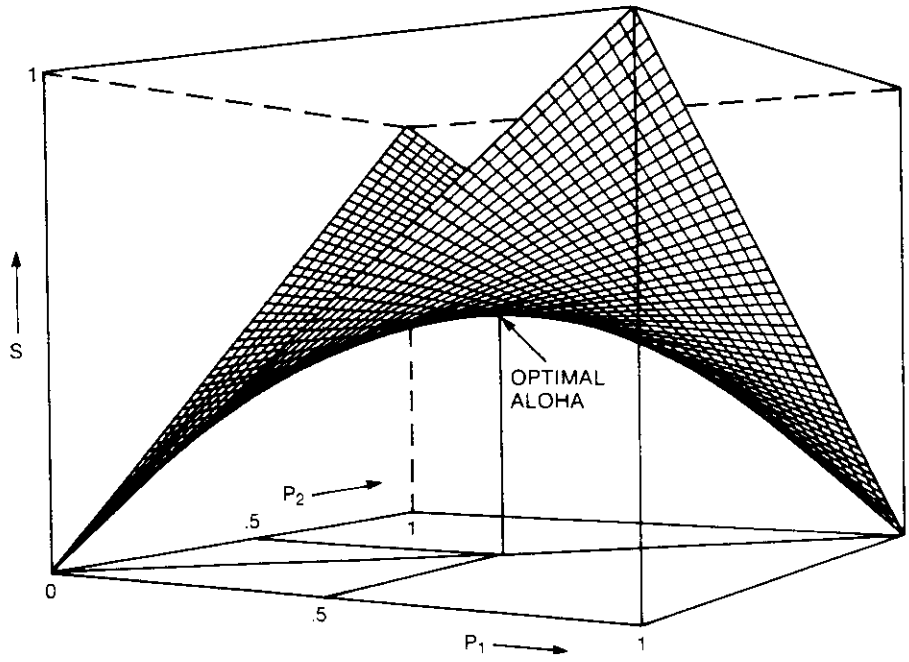


FIGURE 8. *Throughput surface for two busy terminals*

As shown in [67], the optimum policy, then, is to select a critical number of users, k, and to give each of these permission to transmit; the only users who will transmit are those that are both busy and selected. The value of k is chosen to maximize the throughput. This is equivalent to maximizing the probability of selecting exactly one busy user; in this case, it will also be true that the expected number of busy users selected will be equal to unity. The value of k is found from the following expression

$$k = \left\lfloor \frac{M}{N} \right\rfloor, \tag{5.40}$$

where $\lfloor X \rfloor$ denotes the integer part of X. In order to guarantee that all users select the same set of k users, one can require that each user use the same pseudorandom number generator to make the selection; in fact, it is only required that each user decide whether he, himself, is to be selected (again, using the same pseudorandom generator). The throughput is given by

$$S = \left\lfloor \frac{M}{N} \right\rfloor \frac{\text{bin}(M - \left\lfloor \frac{M}{N} \right\rfloor, N - 1)}{\text{bin}(M,N)}, \tag{5.41}$$

where we have denoted the binomial coefficient as $\text{bin}(a,b) = \dfrac{a!}{b!(a-b)!}$. The algorithm is quite robust, not even requiring that all users have exactly the same estimate of the number of backlogged users.

5.5.3. Tree access schemes. Tree access algorithms (also known as splitting algorithms) represent a variation of the key idea in the URN model. Once again, we are interested in dividing up the potentially conflicting users into smaller groups, thereby reducing the difficulty of resolving the conflict. (Of course, polling and token passing represent a mechanism for reducing the number of competing users down to a single user.) The idea here is to find some property of the competing users which allows them to form subgroups; each subgroup will be resolved separately in some sequential fashion. The process may be applied recursively to further divide each subgroup into subsubgroups, etc.

In 1978, a few months after the URN model was published, Hayes introduced a method known as 'probing' [43]. Independently, and at about the same time, CAPETANAKIS [15,16] and TSYBAKOV and MIKHAILOV [134] introduced the Tree Algorithm which has since had many refinements and improvements. Sometime later, Molle and Kleinrock published the Virtual Time CSMA protocol [93,94]. All of these fall into the class of Tree Access Schemes.

In the probing algorithm, subgroups of users are polled rather than polling individual users. The idea is to attempt to eliminate many idle users simultaneously. Each user is assumed to have a unique identifier (usually a binary address) which is used to form the subgroups. For example, suppose we had a population of 16 users. By means of a centralized poller (or some equivalent

distributed poller), the entire population is simultaneously asked if anyone has a message to send. Any users with packets to transmit would respond in the affirmative by transmitting energy on the common channel. If a response is heard on the common channel, then the poller splits the population into two equal sized (sub)groups (say users 1 to 8 and users 9 to 16). The first group is then polled. If no answer returns (indicating that users 1 to 8 are idle), then the second group (9 to 16) is polled. Suppose, in fact, only the second group responds; it is then split further (into 9 to 12 and into 13 to 16). The process continues until a responding subgroup contains only one user, in which case the (single) user transmits his message. Then, the most recent responding subgroup is polled until it, too, is reduced to one responding user, etc. With such a procedure, large groups of idle users are eliminated very easily. It is also possible to adjust the size of the groups polled according to the traffic load on the system. Indeed, the optimum size group to poll is such that the probability of finding that group idle is 0.5 based on information theoretic arguments. Hayes provides an analytic treatment of probing in his text [44]; there he solves for the cycle time behavior and for the mean message delay.

The original Tree algorithm [16,134] operates as follows. When a collision occurs, say in the k-th slot, then all users not involved in that collision suspend their activity until those involved in the collision fully resolve their conflict. The conflicting users randomly split into two subgroups (say using the same pseudorandom coin flips or based on the user's address or based on the arrival time of the user's packet). The first of these subgroups transmits in slot $k + 1$. If this results in an idle or a success, then the second subgroup transmits in slot $k + 2$; otherwise, the second subgroup suspends (until the first subgroup resolves their conflict) and the first subgroup splits again into two further subsubgroups, etc. When all packets from this first subgroup are successfully transmitted, the second subgroup proceeds. When the second subgroup finishes, then all (or possibly, a subset whose size is selected optimally as a function of the traffic) of the originally suspended busy users, plus any new ones which turned busy during the previous conflict resolution period, will transmit and the process repeats. As shown in [16], the maximum throughput (for the case where number of users that come out of suspension is chosen optimally) is 0.43... packets per slot. A Tree access algorithm using *ternary* feedback (terminals can distinguish idle, success and collision slots), immediate transmission by newly generated packets and m-ary, as opposed to binary, splitting using minislots is presented in [99]; assuming $\bar{x} = b/C$ and $L = 0$, they obtain

$$W = \frac{b}{C}\left[\frac{\alpha e^{2\alpha}\left(1 - \frac{1+\alpha}{2} e^{-\alpha}\right)}{(1 - \alpha e^{\alpha})(1 - \alpha)}\right] + \frac{b}{C}\left[\frac{e^{\alpha} - \alpha}{1 - \alpha} - \frac{1}{2}\right], \tag{5.42}$$

where $a = \lambda b/C$ (packets per slot) and where we see the maximum throughput is 0.567... packets per slot.

Improvements and variations to the basic Tree algorithm followed quickly after its introduction. For example, it can be seen that if a collision is followed by an idle slot from the first of two subgroups, then the second subgroup will surely generate a collision on the next slot (all the colliding users from the first collision were clearly assigned to the second subgroup); in such a case, the second subgroup should be split in two immediately [84]. This increases the maximum system throughput to 0.46... packets per slot. Another improvement is to recognize that if one collision is followed by a second collision, then the subgroup that did not transmit in the second collision can properly be considered as untested users and placed in the suspended group as if they had never transmitted. This leads to a first-come-first-serve Tree algorithm whose maximum throughput is 0.4871... ; see [34,35,92,102,135] for analyses of this first-come-first-serve algorithm and improvements of it which show that the maximum stable throughput achievable by any tree algorithm lies between 0.4878 and 0.587. An approach to bounding the mean delay for a class of tree algorithms is given in [37]. In [103] a Tree algorithm using a packet admission algorithm based on windows in time is analyzed; there, they solve for the mean response time for a finite non-homogeneous Poisson user population. A study of adaptive polling which allows the polled terminals to respond using a richer alphabet (than binary) is given in [132].

We observe that the Tree algorithms described above take some advantage of the fact that users may be distinguished by their unique time of arrival (i.e., the time when they generate a new packet for transmission). This observation leads us to the Virtual Time CSMA (VTCSMA) protocol [93,94] which takes exquisite advantage of this distinction. In this protocol, which is actually a variation of the CSMA protocols described below in Section 5.5.4, each user keeps two clocks running. The first clock keeps track of real time. The second keeps track of virtual time (which, ideally, should be the same for all users). The virtual time clock stops running whenever the channel is sensed busy; whenever the channel is idle it runs either at a rate $\eta > 1$ sec/sec (if virtual time is less than real time) or at a rate equal to unity (if virtual time and real time are the same). Thus the virtual time clock is never ahead of real time (and is often behind it). Whenever the arrival time of user's packet is passed by the virtual time clock, then that user will transmit his packet. This protocol greatly increases the probability that a transmission will be successful on its first attempt. It is also quite robust, even allowing the individual virtual time clocks to drift relative to each other. The details as to what should be done when a collision occurs can be resolved by any of a number of existing protocols; VTCSMA is designed to assist the first transmission. MOLLE [95] presents a delay model for VTCSMA (and other 'Moving Server' random access protocols); he finds that the mean wait until the *first* transmission is given by

$$W_F = \frac{\lambda \overline{x^2} \left[\frac{\eta}{\eta - 1} \right]^2}{1 - \left[\frac{\eta}{\eta - 1} \right] S}. \tag{5.43}$$

To approximate the overall mean wait for VTCSMA, one may use the expression

$$W \approx W_F + (\frac{G}{S} - 1)\, T_R + L\,, \tag{5.44}$$

where we recall that $(G/S - 1)$ is the average number of retransmissions and T_R is the mean delay caused by the retransmission protocol. In [86], a performance analysis of VTCSMA is given which studies the system stability and throughput; in [87], this analysis is extended and also an explicit expression for the mean wait is given.

5.5.4. Carrier Sense Multiple Access - CSMA. When the key parameter a from Section 2 is small ($a \ll 1$), then we may add some further intelligence to the simple ALOHA protocol to greatly improve its performance. The addition is to require that each user listen to ('sense' the carrier of) the channel before attempting any transmission. If the channel is sensed busy (either due to an ongoing successful transmission or a collision), then the user surely will not transmit, but will follow one of several protocols described below. If the channel is sensed idle, then the user will follow a transmission policy which also depends upon the particular protocol. In all cases, we succeed in preventing a sure collision (as would have occurred in ALOHA) when a user begins transmitting while another user can be sensed to already be in the process of transmitting. These systems were first studied (in the environment of packet radio systems) by KLEINROCK and TOBAGI [59,123].

In all of the CSMA protocols, it is possible for two (or more) users to collide if a user senses the channel idle when, in fact, another user has just begun a transmission which has not yet 'reached' the listening user (i.e., the listening user senses the channel at a time that is less than a propagation delay from when the transmitting user began his transmission). Whenever a user detects that his transmission has been ruined by a collision, that user schedules the retransmission (i.e., he will once again sense the channel) at some later time according to a retransmission delay distribution.

The first CSMA protocol is known as *non-persistent CSMA*. In this case, if the channel is sensed idle, the user will transmit his packet immediately. If the channel is sensed busy, the user acts as though there was a collision (see above) and at the rescheduled time, repeats the algorithm.

The next CSMA protocol is known as *1-persistent CSMA*. Here, again, the user will transmit if he senses the channel idle. If he senses the channel busy, then he continues to sense the channel until it goes idle and then, with probability one, he will transmit. We observe that if more than one user senses the channel busy, then, when the channel goes idle, there will be a guaranteed collision. One way to help prevent the latter from occurring is given in the following protocol.

The third protocol is referred to as *p-persistent CSMA*. It is identical to 1-persistent CSMA except that when the channel goes from busy to idle, a waiting user will transmit with probability p. With probability $1-p$, he will

reschedule his transmission by L seconds (see Section 2), that is, by one propagation delay (a minislot). If the channel is idle at this next attempt (i.e., after one minislot), then he will transmit with probability p, etc. If it is sensed busy after this minislot, the user will act as if his packet met with a collision and will reschedule his transmission according to the retransmission delay distribution.

The $S:G$ relationships for these three protocols (and variations) are as follows (see [61,62,123]):

Non-persistent CSMA

$$S = \frac{Ge^{-aG}}{G(1 + 2a) + e^{-aG}}; \tag{5.45}$$

Slotted non-persistent CSMA

$$S = \frac{aGe^{-aG}}{1 - e^{-aG} + a}; \tag{5.46}$$

1-persistent CSMA

$$S = \frac{G[1 + G + aG(1 + G + aG/2)]e^{-G(1 + 2a)}}{G(1 + 2a) - (1 - e^{-aG}) + (1 + aG)e^{-G(1 + a)}}; \tag{5.47}$$

Slotted 1-persistent CSMA

$$S = \frac{Ge^{-G(1 + a)}[1 + a - e^{-aG}]}{(1 + a)(1 - e^{-aG}) + ae^{-G(1 + a)}}; \tag{5.48}$$

p-persistent CSMA

$$S(G, p, a) = \frac{(1 - e^{-aG})[P_s'\pi_0 + P_s(1 - \pi_0)]}{(1 - e^{-aG})[\bar{a}\bar{t}'\pi_0 + a\bar{t}(1 - \pi_0) + 1 + a] + a\pi_0}; \tag{5.49}$$

where P_s', P_s, $\bar{t}'$, $\bar{t}$ and π_0 are defined in [62,123].

A delay analysis for CSMA is given in [127,128]. Stability considerations for CSMA protocols may be found in [25,35,85,126]. The analysis in [127,128] for non-persistent CSMA leads to a system of equations which must be evaluated numerically. However, as was the case with ALOHA, we find that CSMA is fundamentally unstable, and some form of control procedure must be used to render the system stable. In [35] a stabilization procedure is discussed which adjusts the probability of retransmitting a backlogged packet as a function of the estimated number of backlogged packets in the system; this procedure is based on the procedure given in [77]. Upon optimizing this retransmission probability, the mean wait is approximated by

$$W \approx \frac{(S + 2\sqrt{2a})\,\bar{x}}{2\,[1 - S(1 + \sqrt{2a})]}. \tag{5.50}$$

An approximate analysis for p-persistent slotted CSMA with a finite number of buffered terminals is given in [115] which requires the numerical evaluation

of a set of equations.

The CSMA protocols were first introduced for application in packet radio networks [51]. In such radio networks, it is relatively easy to sense the channel before you transmit over it; however, once a user begins transmission, it is imperative that his receiver be turned off. Thus, the user cannot monitor the channel while he is transmitting. However, in a wire-based local area network, it is perfectly feasible to 'listen while transmitting' as well as to 'listen before transmitting'. If one does sense the channel during transmission, then a user can detect if his transmission is destroyed by a collision that hits him after he starts. This is referred to as 'collision detection' (CD). This is often used with CSMA to produce yet another protocol, Carrier Sense Multiple Access with Collision Detection - CSMA/CD. In fact, this is the media access protocol used with Ethernet [91]. The form of carrier sense used with Ethernet is, in fact, 1-persistent; the guaranteed collisions which can occur with this CSMA protocol are quickly aborted using the CD capability. The throughput analysis for CSMA/CD is nicely treated in a unified fashion in [117,120]; an earlier treatment was given in [127]. For example, the $S:G$ relationship for unslotted 1-persistent CSMA/CD with an infinite population of users, is

$$\begin{aligned} S = {} & G(1+bG)e^{-(a+b)G} / \\ & \Big[G(a+b)\Big[1-(1+G)e^{-(1+a)G}\Big]-(1+bG)\Big[1-G(1-a-b)\Big]e^{-(a+b)G} \\ & +e^{-bG}\left[2+\frac{G}{4}(a+5b)+\frac{1}{2}b(a+b)G^2\right] \\ & +\frac{G}{4}(2bG+3+2aG)(a+b)e^{-(2a+b)G} \\ & -G(1-b)e^{-(1+a+b)G}-\frac{1}{4}e^{-2bG}\left[1-(1+2aG)e^{-2aG}\right]\Big], \end{aligned} \tag{5.51}$$

where b = collision detection time (normalized to an average packet transmission time).

The first delay analyses for CSMA/CD were carried out under slightly different assumptions and approximations in [79,127]. The analysis in [127] for a finite population slotted non-persistent CSMA/CD system requires the numerical evaluation of a series of equations (as for CSMA). The analysis in [79] for an infinite population slotted p-persistent CSMA/CD system leads to the following explicit equation for the mean wait:

$$W = 2L(1+\frac{1}{p}) - \left[\frac{1-e^{-2\lambda L}}{pC^*(\frac{S}{\bar{x}})-(1-e^{-2\lambda L})}\right]\left[\frac{1}{p}+L(p-3)\right] \tag{5.52}$$

$$+\frac{S\frac{\overline{x^2}}{2\bar{x}}+SL+\lambda\frac{L^2}{2}+2(\bar{x}+L)\frac{2L}{p}+4L^2(1+2\frac{1-p}{p^2})}{1-S-\lambda L(1+\frac{2}{p})}.$$

Here, $C^*(s) = B^*(s)e^{-sL}$ and $B^*(s)$ is the Laplace transform of the service time probability density function. Analysis of a finite population of buffered terminals using CSMA/CD is provided in [23,115,116,121,122]. We point out that CSMA/CD is also unstable and requires a control procedure to stabilize it. In [35], the mean wait for an infinite population stabilized and optimized slotted p-persistent CSMA/CD system is given as

$$W \approx \frac{\dfrac{S\overline{x^2}}{2\bar{x}} + L(2.31+S)}{1 - S(1+3.31a)}. \tag{5.53}$$

An approach for the analysis of a number of random access schemes with a finite number of buffered terminals is given in [98].

5.5.5. Output processes. Let X be the time between successful packet transmissions. If we can find its mean, $\bar{X}$, then the inverse of this will be the system throughput; that is, $S = 1/\bar{X}$. This approach was taken by Ferguson for ALOHA [28] and by Tobagi for ALOHA and CSMA [128]. An extensive treatment using this 'output process' approach is given by TAKAGI and KLEINROCK [117] for a number of random access protocols. In that treatment, memoryless protocols are defined to be those for which a user acts independently of the previous history of the system whenever he senses the channel idle. For these protocols, we view the channel as alternating between 'transmission periods' (defined as a continuous interval during which at least one user is transmitting, or any transmission is being sensed) and 'idle periods' (defined as a continuous interval during which no transmissions are taking place or being sensed). An unsuccessful transmission period has a duration F, and a successful transmission period has duration T; an idle period has duration I. These random variables have means $\bar{F}$, $\bar{T}$ and $\bar{I}$ and pdf's whose Laplace transforms are $F^*(s)$, $T^*(s)$ and $I^*(s)$, respectively. Further, let $X^*(s)$ be the Laplace transform of the pdf for X, the interdeparture time of successful packets. Lastly, let γ be the probability of a successful transmission once it has been started by breaking an idle period (γ depends upon the protocol and other system parameters). As shown in [117], $X^*(s)$ is given by

$$X^*(s) = \frac{\gamma\, T^*(s)}{\dfrac{1}{I^*(s)} - (1-\gamma)\, F^*(s)}. \tag{5.54}$$

and the system throughput is given by

$$S = \left[\frac{1-\gamma}{\gamma}(\bar{I} + \bar{F}) + \bar{I} + \bar{T} \right]^{-1}. \tag{5.55}$$

These last two fundamental results may be applied to a number of random access protocols as shown in [117].

5.5.6. Multi-hop systems. Essentially all of the material so far in this section has been devoted to the multi-access broadcast channel. An underlying assumption in these models is that all users can hear all other users. In a number of environments, particularly, in packet radio systems, this broadcast assumption is false. For example, users may not be in line-of-sight of each other or, they may not be in range of each other. In this case the connectivity is less than complete and the users are connected via a graph which describes who can hear whom. Thus, in order to send a packet from one user to another, it may be necessary to relay the packet through intermediate users (acting as repeaters for such packets). Whereas this routing problem creates many analytic (and operational) problems, it also permits more than one successful transmission to take place simultaneously as long as these transmissions do not overlap at any of the receivers; this is referred to as 'spatial reuse' of the channel. This environment is known as a 'multi-hop system'.

The analytical problems associated with multi-hop systems are ferocious! A number of studies have been published which solve some very specialized aspects of certain models of the system, but to date, no satisfactory complete solution is available.

One of the earliest studies of this problem was given in [63]. Another early study [124] investigated the 'hidden terminal problem' in which CSMA was shown to degrade badly if even a small fraction of the users could not be heard; in that same study, an access protocol known as 'Busy Tone Multiple Access' [124] was devised to solve this problem. Another study of BTMA is given in [113].

One must also study the effect of different transmission ranges on the system performance. Among the published works here, we include [3,4,8,97,114,118]. A general rule for optimizing the system throughput was given in [138]; this rule states that a user should transmit at a rate and range such that the additional throughput he achieves is exactly equal to the rate at which he destroys successful transmissions by other users.

A nice approach to solving for the throughput and blocking probability of multi-hop packet radio systems was first presented in 1980 [8] and refined in [9]. The key idea in this approach is to identify a set of busy (i.e., transmitting) terminals; under the appropriate assumptions regarding the channel traffic and the access method (e.g., CSMA), the authors show that each such set of busy nodes represents a state in a Markov process. They show that the stationary probabilities for sets of busy terminals have a simple product form solution which depends on the message transmission times only through their mean (and not on the distribution); this is similar to the usual queueing network situation [4]. In [129], the underlying reversible Markov process was studied and, using the conditions necessary for an access protocol to have a product-form solution, they analyzed the throughput for a number of multi-hop channel access schemes.

5.5.7. A macroscopic approach. Coupled queueing systems usually present enormous analytic complexity. Multi-hop packet radio networks represent one such example. (One notable exception is the class of product-form queueing networks.) It does seem hopeless to attempt to track the exact behavior of each terminal in a multi-hop environment as the number of terminals grows. Yet queueing theory has traditionally required a fine-grained analysis of many interacting elements in order to evaluate coarse-grained equilibrium performance measures. However, a recent and novel approach to dealing with large distributed queueing systems is given in [100,101,139].

A sketch of the approach via an example follows. Consider a multi-hop packet radio system serving M terminals which use CSMA as their access method. Further assume that each terminal schedules new or retransmitted messages for transmission (i.e., listens to the channel for transmission of a message) according to a Poisson process at a rate λ. Message lengths are assumed to be exponentially distributed with a mean of b bits and the channel speed is assumed to be C bits per second; thus the average message transmission time (if successful) is $\bar{x} = b/C$ seconds. Further, assume $L = 0$ (i.e., zero propagation delay). The equilibrium probability, P_k of having k simultaneous transmissions in the system, is given by

$$P_k = \frac{\rho^k}{Z_M}, \tag{5.56}$$

where $\rho = \lambda\bar{x}$ and Z_M is the *partition function* of the system defined as

$$Z_M = \sum_{k=0}^{M} \alpha_M^{(k)} \rho^k. \tag{5.57}$$

Here, $\alpha_M^{(k)}$ is the number of ways in which k terminals out of M can be transmitting at the same time. We recognize that the partition function is the generating function for the number of simultaneous transmissions in the system. Once the partition function is computed for a system, then one can obtain the system throughput, among other measures. Note the similarity of this approach to that of [8].

In [139], an analogy between this approach and that of statistical mechanics is made. Drawing upon an observation of Beneš [6], it is observed that statistical mechanics need not track the dynamics of the system to obtain the equilibrium behavior. In a series of papers ([100,101,139]), we are treated by the application of these ideas to a number of computer and communication examples and to a correspondence between quantities from statistical mechanics and multi-access systems: microstate energy $<=>$ size of set of independent transmissions; global energy $<=>$ system throughput; volume occupied by a terminal $<=>$ E[number of transmissions blocked by that terminal when it transmits]; pressure of a multi-access system $<=>$ average rate of blocked transmissions; critical phase transition $<=>$ discontinuous throughput behavior. One wonders if this intriguing approach will lead to a kinetic theory of packet motion in networks of many types.

6. Distributed processing

No discussion of performance evaluation would be complete without discussing distributed processing systems; these play an important role in today's distributed environment and create a greater need for suitable computer-communication networks. Moreover, they present a number of fascinating queueing theoretic problems of great interest.

6.1. A simple model for distributed processing

In [70], the question is posed, 'Is distributed processing any good?'. Surprisingly, the analysis there shows that it never improves performance. Let us review that analysis.

The model is a rather simple one. We assume that a Poisson stream of arrivals (at rate λ jobs per second) requires work from a processing system whose total processing power is C operations per second. Each arriving job requires an exponentially distributed number of operations, with an average of $1/\mu$ operations. However, rather than assuming a single processor of capacity C, we allow ourselves to consider a general series-parallel configuration as shown in Fig. 9.

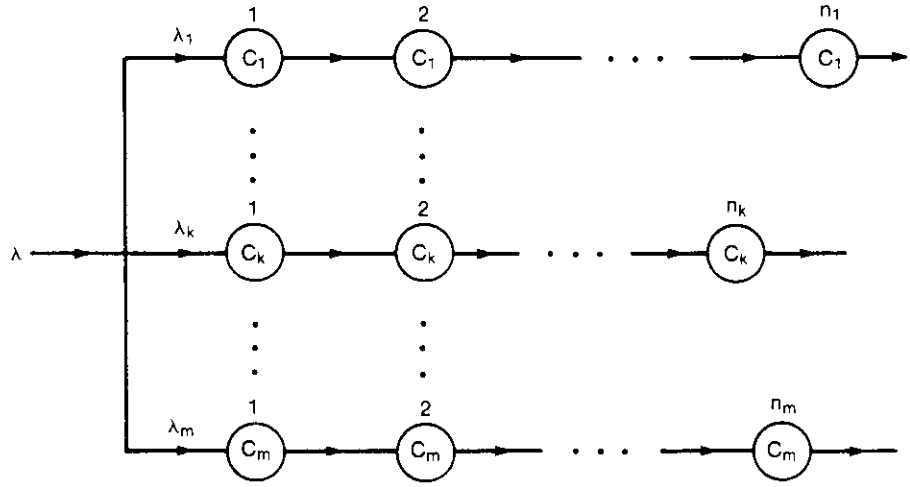


Figure 9. *The general series-parallel topology*

There are m parallel chains, the k-th of which contains n_k processors in series. We have allocated the total system capacity among smaller processors such that each processor in the k-th chain has the same capacity, C_k. Clearly, the *total* system capacity (in operations/sec) is

$$C = \sum_{k=1}^{m} n_k C_k. \tag{6.1}$$

We assume that an entering job will randomly select the k-th chain in which to be processed; the result of this selection produces a net Poisson flow of λ_k jobs

per second on the k-th chain. We further assume that each of the n_k processors in this chain will do $1/n_k$ of the work necessary for this job and that each such processor takes an exponentially distributed time to complete the work (with mean $1/\mu n_k C_k$ seconds). This assumption of individual exponentiality contradicts the earlier assumption of an exponentially distributed total job length, but we use it here for simplicity; an analysis which accounts for the Erlangian nature of these series chains may be found in [131]. Each processor in the k-th chain has a utilization factor given by

$$\rho_k = \lambda_k / \mu n_k C_k. \tag{6.2}$$

Define

$T(\rho) =$ E[response time for a job in the series-parallel network],

$T_0(\rho) =$ E[response time for a job served in a central processor of capacity C].

We wish to calculate $T(\rho)/T_0(\rho)$. Clearly, if distributed processing is to reduce the mean response time for jobs, then this ratio should be less than unity. Unfortunately, this is never true! As shown in [70],

$$\frac{T(\rho)}{T_0(\rho)} = \sum_{k=1}^{m} n_k \frac{\rho_k/(1-\rho_k)}{\rho/(1-\rho)}, \tag{6.3}$$

where $\rho = \lambda/\mu C$.

Some examples will illustrate the performance. For the case $m = 1$ and $n_1 = n$ (the pure series chain), we have

$$T(\rho)/T_0(\rho) = n. \tag{6.4}$$

For the case $n_k = 1$, $C_k = C/m$ and $\lambda_k = \lambda/m$ for $k = 1,2, \ldots, m$ (the pure parallel case), we have

$$T(\rho)/T_0(\rho) = m. \tag{6.5}$$

For the case $n_k = n$, $C_k = C/mn$ and $\lambda_k = \lambda/m$ for $k = 1, 2, \ldots, m$ (the symmetrical series-parallel case), we have

$$T(\rho)/T_0(\rho) = mn. \tag{6.6}$$

Things are just getting worse! It appears as though the mean response time is directly proportional to the number of processors over which we distribute the work. Indeed, for the general series-parallel case where all processors have the same utilization factor $\rho = \lambda/\mu C$, we have

$$T(\rho)/T_0(\rho) = \sum_{k=1}^{m} n_k. \tag{6.7}$$

As discussed in [70], the practical justification for the great current interest in distributed processing (in spite of the pessimistic results just described) is due to the enormously reduced cost of microprocessors as opposed to mainframes.

6.2. A new class of queueing problems

A computer job is often modelled as a collection of tasks which depend upon each other in some sequential fashion. These dependencies are often drawn as a partially ordered tree known as a computation graph for jobs. The level of resolution for tasks varies, as does the architecture for processing the job (see [71]). An example of a computation graph is given in Fig. 10.

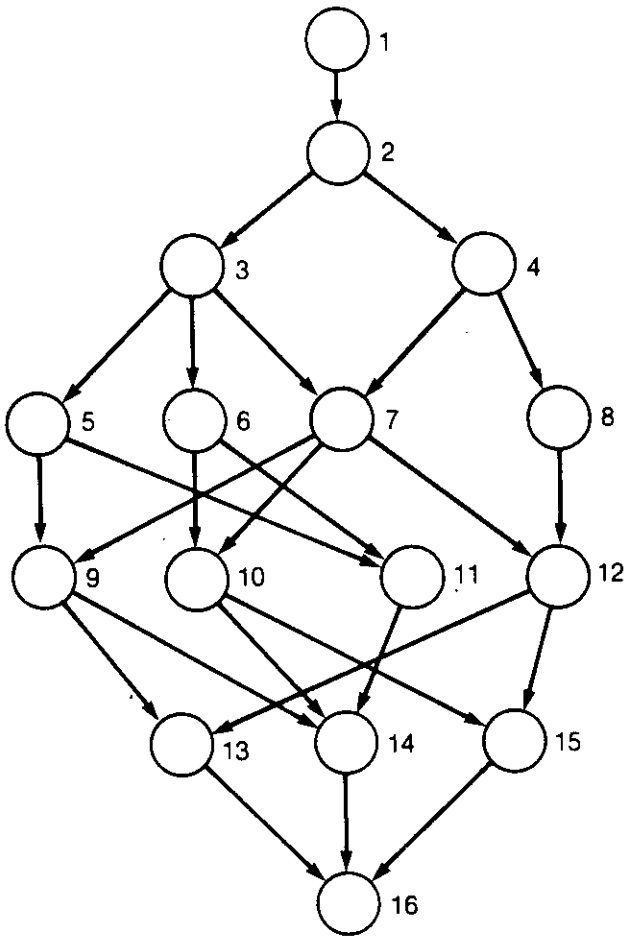
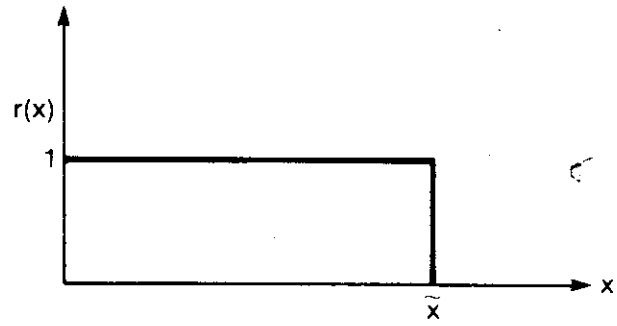


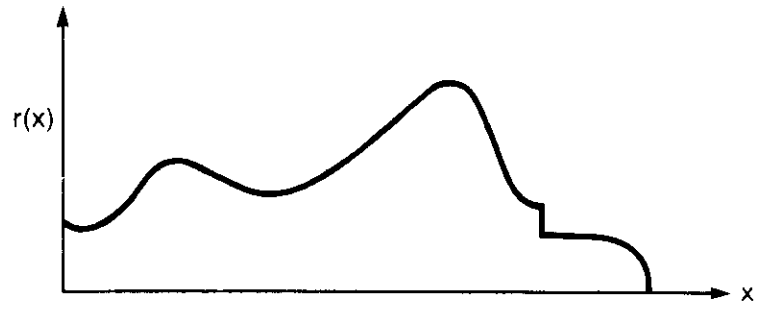
FIGURE 10. *The graph model of computation*

The nodes represent tasks and the arrows represent dependencies in the sense that a task at the head end of an arrow cannot begin execution until the task at the tail end of the arrow is completed. Let us assume that each task requires a random amount of time to be processed (for simplicity, assume that each is exponentially distributed with rate μ). Assume we have P processors which are available to work on the job and that not more than one processor may work on a given task. When node (task) 1 is being processed, the job

proceeds at a rate μ. When it completes, node 2 begins and again the job proceeds at a rate μ. When it finishes, nodes 3 and 4 both become activated and, assuming at least two processors are available, the job then proceeds at a rate 2μ, etc. What we see is that the rate at which a job can absorb work depends upon where it is in its processing cycle (i.e., which tasks are currently being processed) and also depends upon how many processors are available to work on the job (other jobs may be competing for a finite number of processors). Of course, we would like to handle the case of an arbitrary distribution of task times. The classic queueing model, shown in Fig. 11a assumes that the rate at which a job can absorb work is constant, and that the total amount of work is a random variable, $\tilde{x}$. Specifically, we plot $r(x)$, the desired rate of service (seconds per second) as a function of the elapsed time in service (assuming no competition for service). In Fig. 11b, we show a general picture for our new class of problems where the rate varies with the amount of service received.



a. *Classical queueing*



b. *New queueing class*

FIGURE 11. *Desired rate of service as a function of the elapsed time in service*

To date, the class of queueing systems shown in Fig. 11b has not been solved. We point out that the work of COHEN [22] offers an approach to this

problem. Further, it has been shown in [5] for any computation graph (possibly randomly selected among the jobs), any service time distribution, any arrival process and any work-conserving processor scheduling algorithm, that $\bar{B}$, the average number of busy processors, is given by

$$\bar{B} = \begin{cases} \lambda \bar{X} & \text{for } \lambda \bar{X} < P, \\ P & \text{for } \lambda \bar{X} \geqslant P, \end{cases} \tag{6.8}$$

where P is the number of processors available, λ is the rate at which jobs arrive, and $\bar{X}$ is the average number of seconds of processing required by a job.

7. Conclusions

An enormous literature is devoted to the modeling, analysis and design of computer-communication systems. The thrust for this activity comes from the practical application areas of data communications and data processing. This same practical environment has forced the theoretician/engineer to find working approximate (if not exact) solutions to some very difficult problems.

Nevertheless, from a theoretical point of view, many of the interesting problems are still unsolved. This has caused the queueing theorist to develop new techniques for many of these problems. The next few years will only increase the pressure and challenge to the theorist to keep up with the breakneck pace at which new problems arise.

References

1. N. Abramson (1970). The ALOHA system - another alternative for computer communications. *1970 Fall Joint Computer Conference, AFIPS Conference Proceedings, Vol. 37*, Montvale, NJ:AFIPS Press, 281-285.
2. N. Abramson (1973). Packet switching with satellites. *1973 National Computer Conference, AFIPS Conference Proceedings, Vol. 42*, 695-702.
3. G. Akavia, L. Kleinrock (1984). On a self-adjusting capability of ALOHA networks. *IEEE Trans. Commun. COM-32*, 40-47.
4. F. Baskett, K.M. Chandy, R.R. Muntz, F. Palacios-Gomez (1975). Open, closed, and mixed networks of queues with different classes of customers. *J. Assoc. Comput. Mach. 22*, 248-260.
5. A. Belghith, L. Kleinrock (1984). *Analysis of the Number of Occupied Processors in a Multi-Processing System*, Technical Report No. 850027, Computer Science Dept., School of Engineering and Applied Science, University of California, Los Angeles.
6. V.E. Beneš (1963). A 'thermodynamic' theory of traffic in connecting networks. *The Bell System Tech. J. 42*, 567-607.
7. R. Binder (1975). A dynamic packet switching system for satellite broadcast channels. *International Conference on Communications '75, Vol. 3*, San Francisco, California, 41-1 - 41-5.
8. R. Boorstyn, A. Kershenbaum (1980). Throughput analysis of

multihop packet radio. *International Conference on Communications '80. Vol. 1,* Seattle, WA, 13.6.1 - 13.6.6.

9. R. BOORSTYN, A. KERSHENBAUM, B. MAGLARIS, V. SAHIN (1987). Throughput analysis in multihop CSMA packet radio networks. *IEEE Trans. Commun. COM-35,* 267-274.
10. F. BORGONOVO, L. FRATTA (1978). SRUC: A technique for packet transmission on multiple access channels. *Proceedings of the Fourth International Conference on Computer Communications,* Kyoto, 601-607.
11. O.J. BOXMA (1979). On a tandem queueing model with identical service times at both counters, I, II. *Adv. in Appl. Probab. 11,* 616-643 and 644-659.
12. O.J. BOXMA, W.P. GROENENDIJK (1987). Pseudo-conservation laws in cyclic-service systems. *J. Appl. Probab. 24,* 949-964.
13. O.J. BOXMA, W.P. GROENENDIJK (1988). Waiting times in discrete-time cyclic-service systems. *IEEE Trans. Commun. COM-36,* 164-170.
14. W. BUX (1981). Local-area subnetworks: a performance comparison. *IEEE Trans. Commun. COM-29,* 1465-1473.
15. J.I. CAPETANAKIS (1977). *The Multiple Access Broadcast Channel: Protocol and Capacity Considerations,* Ph.D. Dissertation, Dept. of Electrical Engineering and Computer Science, MIT.
16. J.I. CAPETANAKIS (1979). Generalized TDMA: the multi-accessing tree protocol. *IEEE Trans. Commun. COM-27,* 1476-1484.
17. A.B. CARLEIAL, M.E. HELLMAN (1975). Bistable behavior of slotted ALOHA-type systems. *IEEE Trans. Commun. COM-23,* 401-410.
18. R.T. CARSTEN, E.E. NEWHALL, M.J.M. POSNER (1977). A simplified analysis of scan times in an asymmetrical Newhall loop with exhaustive service. *IEEE Trans. Commun. COM-25,* 951-957.
19. R.T. CARSTEN, M.J.M. POSNER (1978). Simplified statistical models of single and multiple Newhall loops. *Conference Record, 1978 National Telecommunications Conference, IEEE78CH1354-0,* Birmingham, Alabama, 44.5.1 - 44.5.7.
20. I. CHLAMTAC, W.R. FRANTA, K.D. LEVIN (1979). BRAM: the broadcast recognizing access method. *IEEE Trans. Commun. COM-27,* 1183-1190.
21. L.P. CLARE (1986). Delay analysis of stable slotted ALOHA systems. *Proceedings, IEEE INFOCOM '86,* Miami, Florida. 10-19.
22. J.W. COHEN (1979). The multiple phase service network with generalized processor sharing. *Acta Informatica 12,* 245-284.
23. E.J. COYLE, B. LIU (1982). Calculation of the stability characteristics and buffer requirements of asynchronous CSMA/CD networks. *International Conference on Communications '82,* Philadelphia, Pennsylvania, 7F.1.1 - 7F.1.5.
24. W.R. CROWTHER, R. RETTBERG, D. WALDEN, S. ORNSTEIN, F. HEART (1973). A system for broadcast communication: reservation-ALOHA. *Proceedings, 6th Hawaii International Conference on System Science.*
25. G. CUNNINGHAM, J. MEDITCH (1987). Distributed retransmission

controls for slotted nonpersistent and virtual time CSMA. *Proceedings, IEEE INFOCOM '87,* San Francisco, California, 754-762.

26. L.F.M. DE MORAES, I. RUBIN (1984). Analysis and comparison of message queuing delays in token-rings and token-buses local area networks. *International Conference on Communications, IEEE 84CH2028-9,* 130-135.
27. G. FAYOLLE, E. GELENBE, J. LABETOULLE (1977). Stability and optimal control of the packet switching broadcast channel. *J. Assoc. Comput. Mach. 24,* 375-386.
28. M.J. FERGUSON (1977). A bound and approximation of delay distribution for fixed-length packets in an unslotted ALOHA channel and a comparison with Time Division Multiplexing (TDM). *IEEE Trans. Commun. COM-25,* 136-139.
29. M.J. FERGUSON, Y.J. AMINETZAH (1985). Exact results for nonsymmetric token ring systems. *IEEE Trans. Commun. COM-33,* 223-231.
30. H. FRANK, H.W. CHOU (1972). Topological optimization of computer networks. *Proceedings IEEE 60,* 1385-1397.
31. L. FRATTA, M. GERLA, L. KLEINROCK (1973). The flow deviation method: An approach to store-and-forward communication network design. *Networks 3,* 97-133.
32. A. FUKUDA, K. MUKUMOTO, T. HASEGAWA (1978). Adaptive retransmission randomization schemes for a packet switched random-access broadcast channel. *Proceedings of the Fourth International Conference on Computer Communications,* Kyoto, 543-548.
33. H.R. GAIL (1983). *On the Optimization of Computer Network Power,* Ph.D. Dissertation, UCLA-CSD-830922, Computer Science Dept., School of Engineering and Applied Science, University of California, Los Angeles.
34. R. GALLAGER (1978). Conflict resolution in random access broadcast networks. *Proceedings AFOSR Workshop on Communication Theory Applications,* Provincetown, MA, 74-76.
35. R. GALLAGER, D. BERTSEKAS (1987). *Data Networks,* Prentice Hall, Inc., Englewood Cliffs, NJ.
36. A. GIESSLER, J. HANLE, A. KONIG, E. PADE (1978). Free buffer allocation - an investigation by simulation. *Computer Networks 1,* 191-204.
37. L. GEORGIADES, L.F. MERAKOS, P. PAPANTONI-KAZAKOS (1987). A method for the delay analysis of random multiple-access algorithms whose delay process is regenerative. *IEEE Journal on Selected Areas in Communications SAC-5,* 1051-1062.
38. M. GERLA (1973). *The Design of Store and Forward (S/F) Networks for Computer Communications,* Technical Report UCLA-ENG-7319, Computer Science Dept., School of Engineering and Applied Science, University of California, Los Angeles.
39. M. GERLA, H. FRANK, W. CHOU, J. ECKL (1974). A cut saturation algorithm for topological design of packet-switched communication networks. *Conference Record, 1974 National Telecommunications*

Conference, 1074-1085.

40. M. GERLA, L. KLEINROCK (1977). On the topological design of distributed computer networks. *IEEE Trans. Commun. COM-25,* 48-60.
41. M. GERLA, L. KLEINROCK (1977). Closed loop stability controls for S-ALOHA satellite communications. *Proceedings of the Fifth Data Communications Symposium,* Snowbird, Utah, 2-10 - 2-19.
42. B. HAJEK, T. VAN LOON (1982). Decentralized dynamic control of a multi-access broadcast channel. *IEEE Trans. Aut. Control AC-27,* 559-569.
43. J.F. HAYES (1978). An adaptive technique for local distribution. *IEEE Trans. Commun. COM-26,* 1178-1186.
44. J.F. HAYES (1984). *Modelling and Analysis of Computer Communications Networks,* Plenum Press, New York.
45. D.P. HEYMAN (1983). Data-transport performance analysis of Fasnet. *The Bell System Tech. J. 62,* 2547-2560.
46. W. HILAL, J.F. CHIU (1987). A study of register-insertion rings. *Proceedings, IEEE INFOCOM '87,* San Francisco, California, 479-491.
47. J.-H. HUANG (1988). *On the Behavior of Algorithms in a Multi-Processing Environment,* Ph.D. Dissertation, Computer Science Dept., School of Engineering and Applied Science, University of California, Los Angeles.
48. T.C. HOU, V.O.K. LI (1986). Transmission range control in multihop packet radio networks. *IEEE Trans. Commun. COM-34,* 38-44.
49. J.R. JACKSON (1957). Networks of waiting lines. *Oper. Res. 5,* 518-521.
50. I.M. JACOBS, R. BINDER, E.V. HOVERSTEN (1978). General purpose packet satellite networks. *Proceedings IEEE 66,* 1448-1467.
51. R.E. KAHN, S.A. GRONEMEYER, J.BURCHFIEL, R.C. KUNGELMAN (1978). Advances in packet radio technology. *Proceedings IEEE 66,* 1468-1496.
52. F. KAMOUN, L. KLEINROCK (1980). Analysis of shared finite storage in a computer network node environment under general traffic conditions. *IEEE Trans. Commun. COM-28,* 992-1003.
53. A.R. KAYE (1972). Analysis of a distributed control loop for data transmission. *Proceedings of the Symposium on Computer-Communications Networks and Teletraffic,* Polytechnic Institute of Brooklyn, 47-58.
54. F.P. KELLY, P.K. POLLETT (1983). Sojourn times in closed queueing networks. *Adv. in Appl. Probab. 15,* 638-658.
55. L. KLEINROCK (1972). *Communication Nets; Stochastic Message Flow and Delay,* McGraw-Hill Book Company, New York, 1964 (out of print), reprinted by Dover Publications, New York.
56. L. KLEINROCK (1965). A conservation law for a wide class of queuing disciplines. *Naval Research Logistics Quarterly 12,* 181-192.
57. L. KLEINROCK, S. LAM (1973). Packet switching in a slotted satellite channel. *1973 AFIPS Conference Proceedings, National Computer Conference and Exposition, Vol. 42,* New York, NJ:AFIPS Press, 703-710.

58. L. KLEINROCK, S. LAM (1974). On stability of packet switching in a random multi-access broadcast channel. *Computer Nets, a Supplement to the Proceedings of the Seventh Hawaii International Conference on System Sciences,* Honolulu, Hawaii, 74-77.
59. L. KLEINROCK (1975). *Queueing Systems, Volume I: Theory,* Wiley Interscience, New York.
60. L. KLEINROCK, S. LAM (1975). Packet switching in multi-access broadcast channel: Performance evaluation. *IEEE Trans. Commun. COM-23,* 410-423.
61. L. KLEINROCK, F.A. TOBAGI (1975). Packet switching in radio channels: part I - carrier sense multiple-access modes and their throughput-delay characteristics. *IEEE Trans. Commun. COM-23,* 1400-1416.
62. L. KLEINROCK, F.A. TOBAGI (1975). Random access techniques for data transmission over packet switched radio channels. *1975 National Computer Conference, AFIPS Conference Proceedings, Vol. 44,* Anaheim, NJ:AFIPS Press, 187-201.
63. L. KLEINROCK (1975). On giant stepping in packet radio networks. *UCLA, Packet Radio Temporary Note #5, PRT 136.*
64. L. KLEINROCK (1976). *Queueing Systems, Volume II: Computer Applications,* Wiley Interscience, New York.
65. L. KLEINROCK (1977). Performance of distributed multi-access computer-communication systems. *Information Processing '77, Proceedings of IFIP Congress '77,* North-Holland, Amsterdam, 547-552.
66. L. KLEINROCK (1978). On the flow control in computer networks. *International Conference on Communications, Vol. II,* Toronto, Ontario, 27.2.1 - 27.2.5.
67. L. KLEINROCK, Y. YEMINI (1978). An optimal adaptive scheme for multiple access broadcast communication. *International Conference on Communications, Vol. I,* Toronto, Ontario, 7.2.1 - 7.2.5.
68. L. KLEINROCK (1979). Power and deterministic rules of thumb for probabilistic problems in computer communications. *International Conference on Communications,* Boston, Massachusetts, 43.1.1 - 43.1.10.
69. L. KLEINROCK, M. SCHOLL (1980). Packet switching in radio channels: New conflict-free multiple access schemes. *IEEE Trans. Commun. COM-28,* 1015-1029.
70. L. KLEINROCK (1984). On the theory of distributed processing. *Proceedings of the Twenty-Second Annual Allerton Conference on Communication, Control and Computing,* Urbana-Champaign, 60-70.
71. L. KLEINROCK (1985). Distributed systems. *ACM/IEEE-CS Joint Special Issue, Commun. ACM 28,* 1200-1213, and *Computer 18,* 90-103.
72. L. KLEINROCK, H. LEVY (1988). The analysis of random polling systems. To be published in *Operations Research.*
73. A.G. KONHEIM, B. MEISTER (1974). Waiting lines and times in a system with polling. *J. Assoc. Comput. Mach. 21,* 470-490.
74. P.J. KUEHN (1979). Multiqueue systems with non-exhaustive cyclic service. *The Bell System Tech. J. 58,* 671-698.

75. P.J. KUEHN (1981). Performance of ARQ-protocols for HDX transmission in hierarchical polling systems. *Performance Evaluation 1*, 19-30.
76. S.S. LAM (1974). *Packet Switching in a Multi-Access Broadcast Channel with Applications to Satellite Communication in a Computer Network*, Computer Science Department, School of Engineering and Applied Science, Engineering Report UCLA-ENG-7429, University of California, Los Angeles.
77. S.S. LAM, L. KLEINROCK (1975). Dynamic control schemes for a packet switched multi-access broadcast channel. *1975 National Computer Conference, AFIPS Conference Proceedings, Vol. 44*, Anaheim, NJ:AFIPS Press, 143-153.
78. S.S. LAM (1979). On protocols for satellite packet switching. *International Conference on Communications*, Boston, 58.6.1 - 58.6.6.
79. S.S. LAM (1980). A carrier sense multiple access protocol for local networks. *Computer Networks 4*, 21-32.
80. S.S. LAM, J.W. WONG (1982). Queuing network models of packet-switching networks - part 2: networks with population size constraints. *Performance Evaluation 2*, 161-180.
81. S.S. LAVENBERG (1988). A perspective on queueing models of computer performance. O.J. BOXMA, R. SYSKI (eds.). *Queueing Theory and its Applications - Liber Amicorum for J.W. Cohen*, North-Holland, Amsterdam.
82. M. LIVNY, M. MELMAN (1982). Load balancing in homogeneous broadcast distributed systems. *Proceedings ACM Computer Network Performance Symposium*, 47-55.
83. J.W. MARK (1978). Global scheduling approach to conflict-free multiaccess via a data bus. *IEEE Trans. Commun. COM-26*, 1342-1352.
84. J.L. MASSEY (1980). *Collision-Resolution Algorithms and Random Access Communications*, Technical Report No. UCLA-ENG-8016, School of Engineering and Applied Science, University of California, Los Angeles.
85. J.S. MEDITCH, C.T.A. LEA (1983). Stability and optimization of the CSMA and CSMA/CD channels. *IEEE Trans. Commun. COM-31*, 763-774.
86. J.S. MEDITCH, C.T.A. LEA (1985). Stability and throughput in virtual time CSMA. *Computer Networks and ISDN Systems 10*, 19-26.
87. J.S. MEDITCH, D.H.L. YIN (1986). Performance analysis of virtual time CSMA. *Proceedings, IEEE INFOCOM '86*, Miami, Florida, 242-251.
88. B. MEISTER, H. MULLER, H. RUDIN (1971). New optimization criteria for message-switching networks. *IEEE Trans. Commun. COM-19*, 256-260.
89. B. MELAMED (1982). Sojourn times in queueing networks. *Math. of Oper. Res. 7*, 223-244.
90. R.M. METCALFE (1973). Steady-state analysis of a slotted and controlled ALOHA system with blocking. *Proceedings 6th Hawaii International Conference on Systems Science*, Honolulu, 375-378.
91. R.M. METCALFE, D.R. BOGGS (1976). ETHERNET: Distributed packet

switching for local computer networks. *Commun. ACM 19,* 395-403.

92. V.A. MIKHAILOV, B.S. TSYBAKOV (1981). Upper bound for the capacity of a random multiple access system. *Prob. Peredachi Infor. (USSR) 17,* 90-95.
93. M.L. MOLLE (1981). *Unifications and Extensions of the Multiple Access Communications Problem,* Ph.D. Dissertation, Report No. 810730, (UCLA-ENG-8118), Computer Science Dept., School of Engineering and Applied Science, University of California, Los Angeles.
94. M.L. MOLLE, L. KLEINROCK (1985). Virtual time CSMA: Why two clocks are better than one. *IEEE Trans. Commun. COM-33,* 919-933.
95. M.L. MOLLE (1987). Modelling the delay in 'moving server' random access protocols. Y. YEMINI (ed.). *Current Advances in Distributed Computing and Communications,* Computer Science Press, Inc., 17-32.
96. R.J.T. MORRIS, Y.T. WANG (1984). Some results for multi-queue systems with multiple cyclic servers. H. RUDIN, W. BUX (eds.). *Performance of Computer Communication Systems,* North-Holland, Amsterdam, 245-258.
97. R. NELSON, L. KLEINROCK (1985). Spatial TDMA: A collision free multihop channel access protocol. *IEEE Trans. Commun. COM-33,* 934-944.
98. M. NIKTASH, J.S. RIORDON, S.A. MAHMOUD (1987). A model for analysis of channel access protocols. *Proceedings, IEEE INFOCOM '87,* San Francisco, California, 381-389.
99. Y. OIE, T. SUDA, H. MIYAHARA, T. HASEGAWA (1987). Throughput and delay analysis of free access tree algorithms with mini-slots. *Proceedings, IEEE INFOCOM '87,* San Francisco, California, 258-267.
100. E. PINSKY (1985). *Canonical Approximation in the Performance Analysis of Distributed Systems,* Ph.D. Dissertation, Computer Science Department, Colurubia University.
101. E. PINSKY, Y. YEMINI, M. SIDI (1987). The canonical approximation in the performance analysis of packet radio networks. Y. YEMINI (ed.). *Current Advances in Distributed Computing and Communications,* Computer Science Press, Inc., 140-160.
102. N. PIPPENGER (1981). Bounds on the performance of protocols for a multiple access broadcast channel. *IEEE Trans. Information Theory IT-27,* 145-151.
103. G. POLYZOS, M. MOLLE, A. VENETSANOPOULOS (1985). Delay analysis of three conflict resolution algorithms: The non-homogeneous case. *Proceedings of IEEE Global Telecommunications Conference '85,* New Orleans, Louisiana, 1504-1509.
104. T. RAITH (1985). Performance analysis of multibus interconnection networks in distributed systems. M. AKIYAMA (ed.). *Eleventh International Teletraffic Congress,* North-Holland, Amsterdam, 4.2A-5-1 - 4.2A-5-7 .
105. R.L. RIVEST (1985). *Network Control by Bayessian Broadcast.* Report MIT/LCS/TM-285, Cambridge, MA: MIT. Laboratory for Computer Science.

106. L.G. ROBERTS (1972). ALOHA packet system with and without slots and capture. *ARPANET Satellite System Note 8 (NIC 11290);* reprinted in *Computer Communication Review 5 (1975),* 28-42.
107. L.G. ROBERTS (1973). Dynamic allocation of satellite capacity through packet reservation. *1973 National Computer Conference and Exposition, AFIPS Conference Proceedings, Vol. 42,* New York, NJ:AFIPS Press, 711-716.
108. I. RUBIN (1974). Communication networks: Message path delays. *IEEE Trans. Information Theory IT-20,* 738-745.
109. I. RUBIN (1977). A group random-access procedure for multi-access communication channels. *Conference Record, 1977 National Telecommunication Conference,* Los Angeles, 12:5-1 - 12:5-7.
110. I. RUBIN (1977). *Integrated Random-Access Reservation Schemes for Multi-Access Communication Channels,* Technical Report UCLA-ENG-7752, School of Engineering and Applied Science, University of California, Los Angeles.
111. T.N. SAADAWI, A. EPHREMIDES (1981). Analysis, stability, and optimization of slotted ALOHA with a finite number of buffer users. *IEEE Trans. Aut. Control AC-26,* 680-689.
112. M. SCHWARTZ (1977). *Computer-Communication Network Design and Analysis,* Prentice-Hall, Englewood Cliffs, NJ.
113. M. SIDI, A. SEGALL (1981). A busy-tone-multiple-access-type scheme for packet-radio networks. G. PUJOLLE (ed.). *Performance of Data Communication Systems and Their Applications,* North-Holland, Amsterdam, 1-10.
114. J. SILVESTER, L. KLEINROCK (1978). Optimum transmission radii for packet radio networks or why six is a magic number. *Conference Record, 1978 National Telecommunications Conference,* Birmingham, Alabama, 4.3.2 - 4.3.5.
115. J. SILVESTER, I. LEE (1982). Performance modeling of buffer CSMA - an iterative approach. *Globecom '82, IEEE Global Telecommunications Conference, Vol. 3,* Miami, Florida, F1.6.1 - F1.6.5.
116. H. TAKAGI, L. KLEINROCK (1984). Diffusion process approximation for the queuing delay in contention packet broadcasting system. H. RUDIN, W. BUX (eds.). *Performance of Computer Communication Systems,* North-Holland, Amsterdam, 111-124.
117. H. TAKAGI, L. KLEINROCK (1985). Output processes in contention packet broadcasting systems. *IEEE Trans. Commun. COM-33,* 1191-1199.
118. H. TAKAGI, L. KLEINROCK (1985). Throughput-delay characteristics of some slotted-ALOHA multihop packet radio networks. *IEEE Trans. Commun. COM-33,* 1200-1207.
119. H. TAKAGI (1986). *Analysis of Polling Systems,* The MIT Press, Cambridge, Massachusetts. Also H. TAKAGI and L. KLEINROCK (1985). *Analysis of Random Polling Systems,* JSI Research Report, TR87-0002, IBM Japan Science Institute.

120. H. TAKAGI, L. KLEINROCK (1987). Throughput analysis for persistent CSMA systems. *IEEE Trans. Commun. COM-35,* 243-245.
121. T. TAKINE, Y. TAKAHASHI, T. HASEGAWA (1987). Performance analysis of a buffered CSMA/CD. L.F.M. DE MORAES, E. DE SOUZA E SILVA, L.F.G. SOARES (eds.). *Proceedings of the Third International Conference on Data Communication Systems and Their Performance,* Editora Campus, Ltda. (Rio de Janeiro), 241-255.
122. S. TASAKA (1986). *Performance Analysis of Multiple Access Protocols,* The MIT Press, Cambridge, Massachusetts.
123. F.A. TOBAGI, L. KLEINROCK (1974). Carrier sense multiple-access for packet switched radio channels. *International Conference on Communications,* Minneapolis, Minnesota, 191-215.
124. F.A. TOBAGI, L. KLEINROCK (1975). Packet switching in radio channels: part II - the hidden terminal problem in carrier sense multiple-access and the busy-tone solution. *IEEE Trans. Commun. COM-23,* 1417-1433.
125. F.A. TOBAGI, L. KLEINROCK (1976). Packet switching in radio channels: part III - polling and dynamic split-channel reservation multiple channel access. *IEEE Trans. Commun. COM-24,* 832-845.
126. F.A. TOBAGI, L. KLEINROCK (1977). Packet switching in radio channels: part IV - stability considerations and dynamic control in carrier sense multiple access. *IEEE Trans. Commun. COM-25,* 1103-1119.
127. F.A. TOBAGI, V.B. HUNT (1980). Performance analysis of carrier sense multiple access with collision detection. *Computer Networks 4,* 245-259.
128. F.A. TOBAGI (1982). Distribution of packet delay and interdeparture time in slotted ALOHA and carrier sense multiple access. *J. Assoc. Comput. Mach. 29,* 907-927.
129. F.A. TOBAGI, J.M. BRAZIO (1983). Throughput analysis of multihop packet radio networks under various channel access schemes. *Proceedings of the IEEE INFOCOM '83,* San Diego, California, 381-389.
130. F.A. TOBAGI, M. FINE (1983). Performance of undirectional broadcast local area networks: Expressnet and Fasnet. *IEEE Journal on Selected Areas in Communications SAC-1,* 913-926.
131. F.A. TOBAGI, H. KANAKIA (1986). A comparison of distributed and centralized processing systems. T. HASEGAWA, H. TAKAGI, Y. TAKAHASHI (eds.). *Computer Networking and Performance Evaluation,* North-Holland, Amsterdam, 483-493.
132. D. TOWSLEY, J.K. WOLF (1984). On adaptive tree polling algorithms. *IEEE Trans. Commun. COM-32,* 1294-1298.
133. P. TRAN-GIA, T. RAITH (1986). Multi queue systems with finite capacity nonexhaustive cyclic service. T. HASEGAWA, H. TAKAGI, Y. TAKAHASHI (eds.). *Computer Networking and Performance Evaluation,* North-Holland, Amsterdam, 213-225.
134. B.S. TSYBAKOV, V.A. MIKHAILOV (1978). Free synchronous packet access in a broadcast channel with feedback. *Prob. Peredachi Infor. (USSR) 14, 32-59.*

135. B.S. TSYBAKOV, V.A. MIKHAILOV (1980). Random multiple access of packets: part and try algorithm. *Prob. Peredachi Infor. (USSR) 16, 65-79.*

136. J. WALRAND, P. VARAIYA (1980). Sojourn times and the overtaking condition in Jacksonian networks. *Adv. in Appl. Probab. 12,* 1000-1018.

137. J.W. WONG (1978). Distribution of end-to-end delay in message-switched networks. *Computer Networks 2,* 44-49.

138. Y. YEMINI, L. KLEINROCK (1979). On a general rule for access control or, silence is golden... . *Proceedings of the International Symposium on Flow Control in Computer Networks,* Versailles, France, 335-347.

139. Y. YEMINI (1983). A statistical mechanics of distributed resource sharing mechanisms. *Proceedings of the IEEE INFOCOM '83,* San Diego, California, 531-540.

140. Y. YOSHIOKA, T. NAKAMURA, R. SATO (1977). An optimum solution of the queuing system. *Electronics and Communications in Japan 60-B,* 590-591.