#208

ANALYTICAL TECHNIQUES FOR COMPUTER COMMUNICATION NETWORKS

L. Kleinrock

Rapporteurs: Mr. J.R. Arnott
Dr. I. Mitrani
Mr. L.B. Wilson

Abstract Queueing and Network-flow theoretic tools are briefly described and their use in the solution of problems in computer communication networks is examined.

Introduction

After a review of resource-sharing concepts the basic principles of queueing theory and network flow theory are introduced. This leads to a discussion of the application of mathematical models for flow in time-sharing systems. A brief description of the Arpanet follows and further analysis and design methods are given for computer networks. Finally there is a discussion of some measurements of the performance of the Arpanet. This material is discussed at length in the author's books (Kleinrock 1974).

1 Resource Sharing Concepts

The central theme of resource sharing is that to the individual user the costs of shared usage are generally less than that of individual ownership. With organized sharing no individual user need go unserved and the various levels of sharing try to equate utilization of the resource with cost. At one extreme a private system has only one user "sharing" a resource. For the majority of users who arrive in bursts and who only use the resource say 5% of the time such a system is wasteful. A more general system is one where a large number of users share a large single resource; here we take advantage of the central limit theorem which states that an individual in a large population behaves like the average. The benefits of this system lie in the quantity discount effects and in the averaging (smoothing) effect. However the system must operate under the constraint of the input demand being less than the resource capacity.

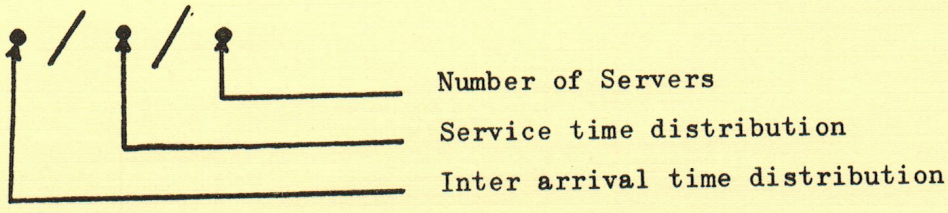
In the study of computer communication networks it is interesting to compare the two component industries. The communications industry is conservative static, regulated and supported with a well defined theory. This is just the opposite of the computer industry which is dynamic, rapidly changing and has a wide distributed plant with little theoretical grounding. Although those two view points seem at first sight to be incompatible it is noted that a similar comparison in the components of a terminal system (user terminals and the computer facility) also reveal incompatibilities. The computer facility is large, centrally located, synchronous with standard alpha-numeric codes with a high, smooth utilization. In comparison there are a large number of terminals, many types, geographically distributed, asynchronous with incompatible alpha numeric codes. A more serious problem with terminals is that they have a low duty cycle and operate in a burst mode.

The various levels of sharing can be illustrated in the design of a remote terminal network. The star net solution with a modem at the computer facility and a low speed asynchronous line for each terminal provides a fast but expensive system. A cheap but slow solution is given with the multidrop (polled) net using a low speed asynchronous line as a minimal spanning tree. When the remote terminals can be geographically grouped a medium speed synchronous line can connect the computer, via modems, to a multiplexor, concentrator or communications orientrated processor located at the centre of the group.

2.0 Introduction to Queueing Systems (Kleinrock 1974)

When a resource is shared there is the likelihood of contention for the use of that resource. In order to compare the different methods of sharing the resource, a set of tools is required to analyze the system. One set of tools is provided by queueing theory.

A queue is denoted by a three-part descriptor as in Figure 2.1.



For example the queue M/G/1 describes a queue with Markovian (M) inter-arrival time, general (G) service time and a single server. In addition the queueing discipline must also be specified, for instance first come first served.

The simplest queueing system is the M/M/1 where the inter-arrival times and service times are exponentially distributed with means λ^{-1} and μ^{-1} respectively. The fraction of time that the server is busy is λ/μ, denoted by ρ and it can be shown for this system that p_n, the probability that there are n customers in the queue is equal to $(1-\rho)\rho^n$ and that the average number of customers, $\bar{n} = \sum_{0}^{\infty} i p_i = \frac{\rho}{1-\rho}$ provided $\rho < 1$. The $(1-\rho)$ factor, or its equivalent, appears in all queueing distributions. In this particular system if the server has a 50% utilization ($\rho = .5$) then the average queue length is one but if the utilization of the server is raised to 90% then the average queue length is increased to nine, etc.

Another important result of the queuing system is the average time (T) spent by a customer in the system. The simplest way to obtain T is to use Little's theorem which states that $\lambda T = \bar{n}$. So for the M/M/1 system $T = 1/\mu(1-\rho)$.

Generalizations can be made for multiple servers, non-Markovian statistics and time dependant behaviour. However the more general the queueing system the more difficult becomes the analysis. For instance with the G/G/1 system we can state the upper bound $T \leq \bar{x} + \frac{\lambda(\sigma_a^2 + \sigma_b^2)}{2(1-\rho)}$ but the exact expression for T is not known. In this expression $\bar{x}$, $1/\lambda$, σ_a^2, σ_b^2 respectively denote the mean service time, mean inter-arrival time, the variance of the inter-arrival times and the variance of the service times. Only when σ_a^2 and σ_b^2 are zero (that is the system is deterministic (D/D/1)) will the average time in the system remain bounded by the mean service time, subject to ρ being less than unity.

Now consider large shared systems M/M/m. When the ratio of the average time spent in the system (T) to the average service time $(1/\mu)$ is plotted against the load (ρ) a set of curves, Figure 2.2., is produced for different values of m. As m increases and the load on the system remains constant the wait time decreases. In the limit as m tends to infinity the system behaves as D/D/1 (that is, no queues are formed provided $\rho < 1$).

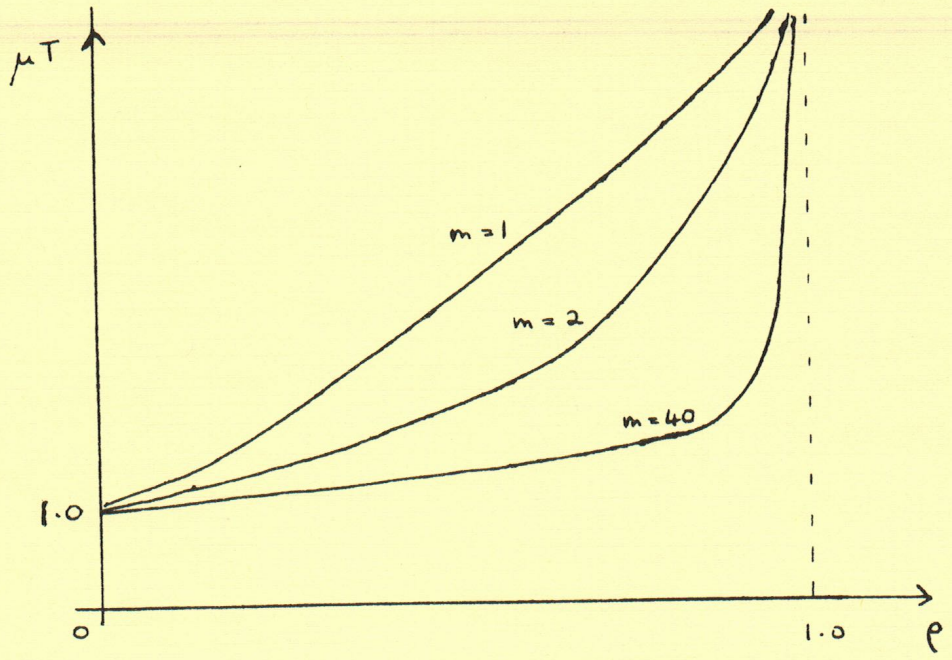


Figure 2.2 M/M/m curves

Most practical queueing systems are of a discrete nature but since little analysis can be done with integers, the transient phenonema may be approximated with continuous averages. Another form of approximation is to include a variance term, that is, a diffusion approximation, which has given fairly good results not only in the overloaded case but also in the stable and underloaded case. The latter is applicable to the study of computer systems.

Queueing theory can also analyze priority queueing disciplines, in particular pre-emptive queueing disciplines which are often encountered in time-sharing systems.

3.0 Network Flow Theory (Frank 1971)

The other set of tools which is useful in analyzing computer networks belongs to network flow theory. Most people think of networks in terms of graphs with the implication that the solution is easily seen but some problems in topological design of networks are so difficult that they cannot, as yet, be solved.

Some problems are easily solved. One instance is the design of a network in which the nodes are connected to a central resource at minimal cost, where cost is proportional to the length. Its solution is the Minimal Spanning tree algorithm which can be determined by iteratively inserting shortest branches subject to no loops being formed.

Another important quantity is the maximum flow between two nodes (the source and terminal) in a network of nodes where the lines of communication between any two nodes have a finite capacity in a given direction. This solution

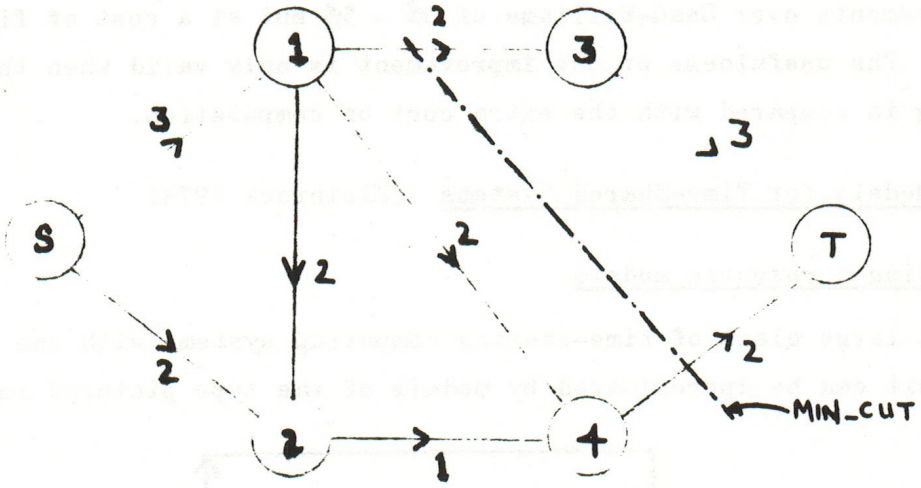


Figure 3.1 Min-cut algorithm

is obtained with the MAX FLOW MIN-CUT theorem; for example, in the network in Figure 3.1 the min-cut illustrated would stop all flow from source to terminal and this minimum cut has a maximum flow capacity of four. The labelling algorithm permits one to find the maximum flow. A modification of this algorithm can also be used to solve the shortest path (between any two nodes) problem. The shortest path can be measured in length, cost, time, or some other unit of "badness" that needs to be minimized.

A fourth kind of problem encountered in communication networks design is to provide a network of maximum flow at minimum cost. The approach taken to solve this problem is to make a sequence of shortest path problems where the length, or cost of a channel changes as units of flow are allocated to that channel. To prevent further loading of a fully utilized channel infinite costs are attributed to that channel.

The last problem to be considered is to connect terminals to a central facility subject to two constraints. The first is that the terminals are supplying data at some fixed rate and secondly that all the lines have the same finite capacity. For large networks, effective algorithms to find the optimum solution have not yet been found, however relatively simple methods exist to reach a solution within 5% to 10% of the optimum. One method, the Esau-Williams, works on the principle that if there are two nodes, A and B, connected to the computer then savings can be made by joining A to B if B is nearer to the computer than A and that the line from B to the computer has the capacity to accept both A's and B's traffic. A recent experiment showed improvements over Esau-Williams of 3% - 5% but at a cost of fifty times the work. The usefulness of the improvement is only valid when the value of the saving is compared with the extra cost of computation.

4 Models for Time-Shared Systems (Kleinrock 1974)

4.1 Single resource models

A large class of time-sharing computing systems with one central processing unit can be approximated by models of the type pictured in figure 4.1.

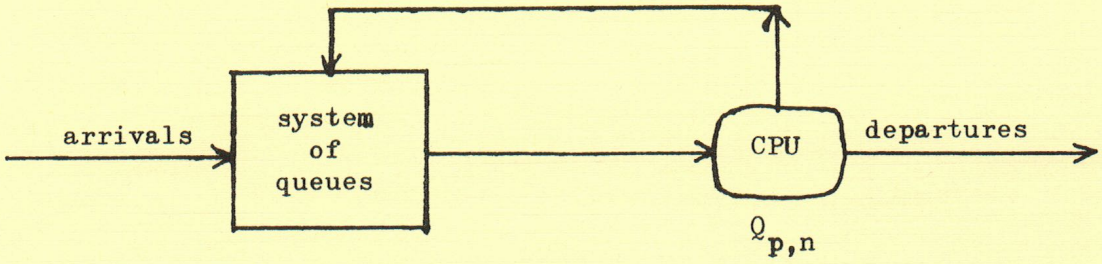


Figure 4.1

Jobs arrive in the system, join one of perhaps several queues, are selected for service according to some scheduling algorithm, receive a quantum of service, then either depart from the system or rejoin a queue to wait for more service.

To use a model of this kind for a particular system, one must make specific assumptions regarding the pattern of arrivals, the scheduling algorithm, the service requirements of jobs and the quantum size. Analysis of the model can then provide expressions for various quantities of interest, such as the average time T that a job spends in the system, the average time T(t) that a job requiring service t spends in the system, etc.

The arrival process is usually assumed to be Poisson (rate λ), i.e. with interarrival times having exponential distribution $A(t) = 1 - e^{-\lambda t}$. The service requirements are sometimes specified in terms of the number of quanta and sometimes in terms of the distribution $B(t)$ of service time required by a job. The quantum size $Q_{p,n}$ may depend on the external priority of the job p and on the number n of visits the job has made to the CPU.

The Batch Processing model is a simple special case within the above framework. Here there is a single queue served in order of arrival (FCFS) and when a job is admitted for service, all his demand is satisfied (no cycling). The classic result for the M/G/1 system gives in this case

$$T(t) = t + \frac{\lambda \bar{t}^2}{2(1-\rho)}$$

where $\bar{t}^2$ is the second moment of the service time and ρ is the traffic intensity. The average time that a job requiring service t spends waiting $W(t)$, is equal to

$$W(t) = T(t) - t = \frac{\lambda \bar{t}^2}{2(1-\rho)} = \frac{W_0 \rho}{1-\rho}$$

where W_0 is the random modification of the service time (the average remaining service time of the job in service, as seen by a random observer). It can be seen that the average waiting time of a job does not depend on its demand for service.

The Round-Robin model is an approximation of real time-sharing systems. Here too, there is a single queue served in order of arrival. Service is given in quanta of fixed size; if after receiving a quantum of service a job requires more it returns to the end of the queue.

The analysis of the round-robin model is quite complicated and, more important, the resultant expressions are cumbersome and difficult to deal with. This is due to the finite quantum size: the service of a job has to be related to the number of quanta it needs. A convenient way of getting round this difficulty is to consider the round-robin with infinitesimal quanta, the Processor Sharing Model.

When the quantum size is (almost) zero, there is no waiting as such; all jobs in the system receive service in parallel, at a rate inversely proportional to the number of jobs in the system. The average time-in-the-system for a job requiring service t is given by

$$T(t) = \frac{t}{1-\rho}$$

and the average 'wasted time' for such a job is

$$W(t) = \frac{\varrho t}{1-\varrho}$$

Now W(t) is a linearly increasing function of t. Short jobs waste less time in the system than long ones, but the 'penalty rate', the ratio W(t)/t, is constant.

There are a number of scheduling stategies which do not select jobs for service in order of arrival. The simplest of these is the Last-Come-First-Served strategy. According to it, a new arrival is admitted for service immediately, preempting the job being served. Its time in the system consists of its service plus all interruptions (each being, in effect, a busy period) generated by new arrivals during its service. Hence

$$T(t) = t + \lambda t \frac{1}{\mu(1-\varrho)} = \frac{t}{1-\varrho}$$

- the same value as in the Processor Sharing system.

A variation on this theme is the FB system. In it, the CPU serves the job which has received least amount of service (a new job enters service immediately) and is shared if there is more than one such job.

Other strategies are based on priority assignements. The next job to be admitted for service is the one with the highest priority among those waiting. The priorities can be preemptive or non-preemptive, static or changing with time. For example, jobs may increase their priority at rate α while waiting, and at rate β while being served ($\alpha > \beta$).

An interesting question in this area is the following: Given a function W(t), what are the necessary and sufficient conditions that it represents the average waiting time for jobs of length t, for some time-sharing system? This question has not been answered yet, although some necessary conditions are known. W(t) must lie within certain upper and lower bounds (for large values of t these are given by the FB and FCFS systems, respectively) and must satisfy certain known conservation laws.

4.2 Multiple resource models

A more realistic view of computing systems should take into account peripheral devices as well as the central processing unit. Consider, for example, the model pictured in Figure 4.2.

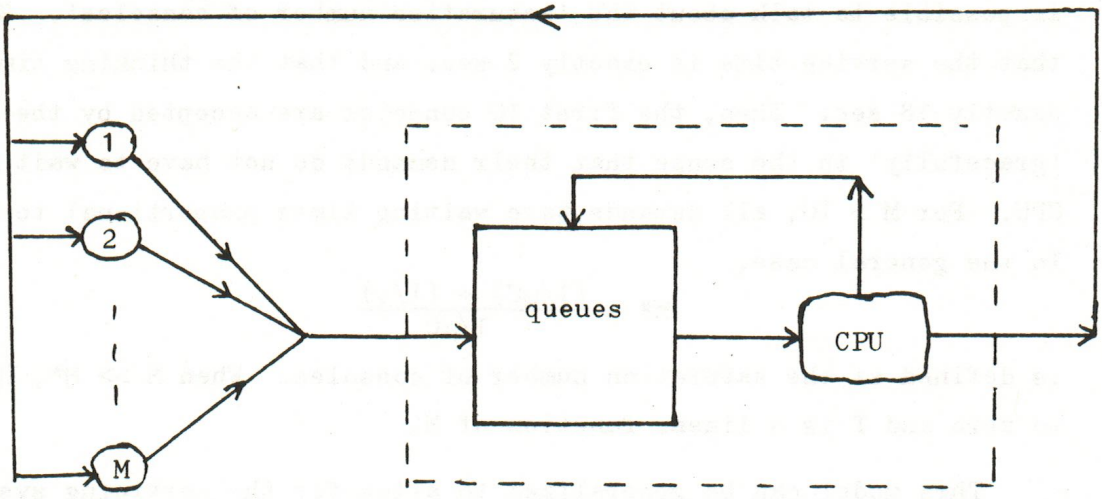


Figure 4.2

The system consists of a time-shared CPU which is fed with demands from a finite number (M) of consoles. When a demand is satisfied it 'returns' to the console which issued it; the console 'thinks' for a period of time and then issues another demand. The quantity of interest here is the 'response time' (T), the average time between issuing a demand and its return. Let $\frac{1}{\mu}$ be the average number of instructions in a demand and C be the speed of the CPU in instructions per second; the average service time for a demand is then $\frac{1}{\mu C}$ sec. Let $\frac{1}{\gamma}$ be the average thinking time. Application of Little's theorem to the subsystem of consoles and to that of the CPU gives

$$T = \frac{M}{\mu C(1-\Pi_o)} - \frac{1}{\gamma} \qquad \text{or} \qquad \mu CT = \frac{M}{1-\Pi_o} - \frac{\mu C}{\gamma}$$

where Π_o is the probability that the CPU is idle. This result is independent of the distributions of thinking and service times. If both these distributions are exponential then

$$\Pi_o = \left[\sum_{i=0}^{M} \frac{M!}{(M-i)!} \left(\frac{\gamma}{\mu C}\right)^i \right]^{-1}$$

The values of T obtained from the above formulae for different values of M are in close agreement with corresponding values observed in a real system.

Even though there is a finite number of demands at the CPU at all times and therefore the system cannot saturate in the normal sense, it is possible to talk about the 'saturation number of consoles'. Suppose that the service time is exactly 2 sec. and that the thinking time is exactly 18 sec. Then, the first 10 consoles are accepted by the system 'gracefully' in the sense that their demands do not have to wait at the CPU. For $M > 10$, all demands have waiting times proportional to $M - 10$. In the general case,

$$M^* = \frac{(1/\mu C) + (1/\gamma)}{1/\mu C}$$

is defined as the saturation number of consoles. When $M \gg M^*$, Π_0 is close to zero and T is a linear function of M.

This model can be generalized to allow for the servicing system to consist of several devices, rather than one. Thus it is possible to specify which device will saturate first, which will saturate next, etc.

4.3 Queueing network

The nodes in a queueing network are queueing systems consisting of one or more servers, with associated queues. The arcs indicate where customers go when they leave a node. The simplest type of network is that in which customers never visit the same node more than once (feedforward networks); an example of such a network is shown in Figure 4.3.

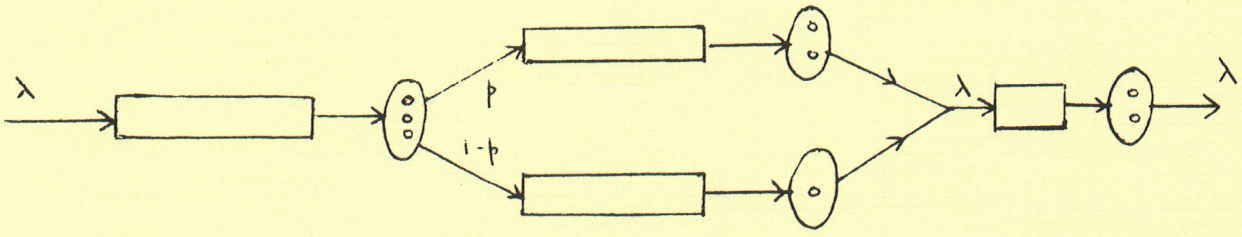


Figure 4.3

A powerful tool for the analysis of feed-forward networks is provided by Burke's Theorem, (Burke 1956) which states that the departure process from an M/M/m queueing system is Poisson. Since Poisson processes remain Poisson after splitting and merging, Burke's theorem implies that every node in a feedforward network behaves like an independent M/M/m system, provided of course that the input to the network is Poisson and all service times are distributed exponentially.

Jackson (Jackson 1957, 1963) studied general queueing networks with N nodes. Customers arrive into a node from outside (in a Poisson stream) and from other

nodes (not necessarily in a Poisson stream). After receiving service (exponential) they either leave the network or go to other nodes. The following parameters characterize the system:

γ_i = external input rate at node i

$r_{i,j}$ = probability of going to node j after leaving node i

λ_i = total input rate at node i

μ_i = service rate of each server in node i

m_i = number of servers in node i

These parameters are not independent; since λ_i is also the rate at which customers leave node i, the following balance equations must hold:

$$\lambda_i = \gamma_i + \sum_{j=1}^{N} \lambda_j \; r_{j,i}$$

Jackson's result states that the joing probability distribution of the number of customers at the N nodes is given by

$$p(n_1, n_2, \ldots, n_N) = p_1(n_1)\, p_2(n_2) \ldots p_N(n_N)$$

where $p_i(n_i)$ is the probability of there being n_i customers in an $M/M/m_i$ system with input rate λ_i and service rate μ_i. This result is remarkable, bearing in mind that the total input to each node is not necessarily Poisson.

A special case of Jackson's model is the 'Closed Network' model analyzed by Gordon and Newell, (Gordon 1967): customers do not enter and do not leave the network; there are K customers 'trapped' in it.

The balance equations

$$\lambda_i = \sum_{j=1}^{N} \lambda_j \, r_{j,i}$$

are homogeneous and cannot be solved uniquely for the λ_i's. The joint distribution of the number of customers in the network still has the product form, but includes a normalizing constant the determination of which is fairly complicated.

5 Arpanet Description (Kleinrock 1974)

The Arpanet (Advanced Research Projects Agency Network) has changed considerably since its inception in 1967, and the evolution of the Network is shown in Figure 5.1. The network started on the West Coast and has

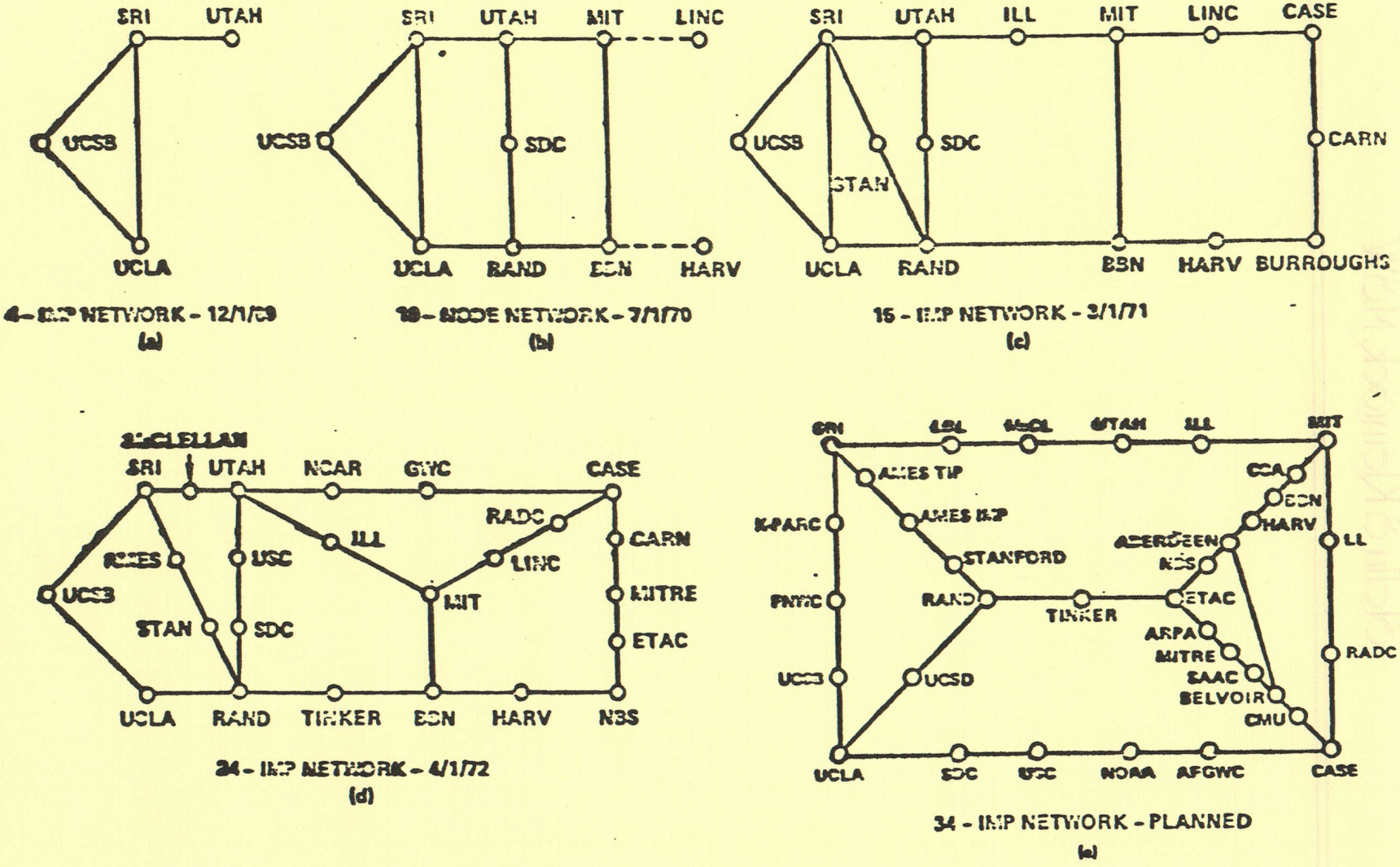


Figure 5.1 Evolution of the ARPA Network

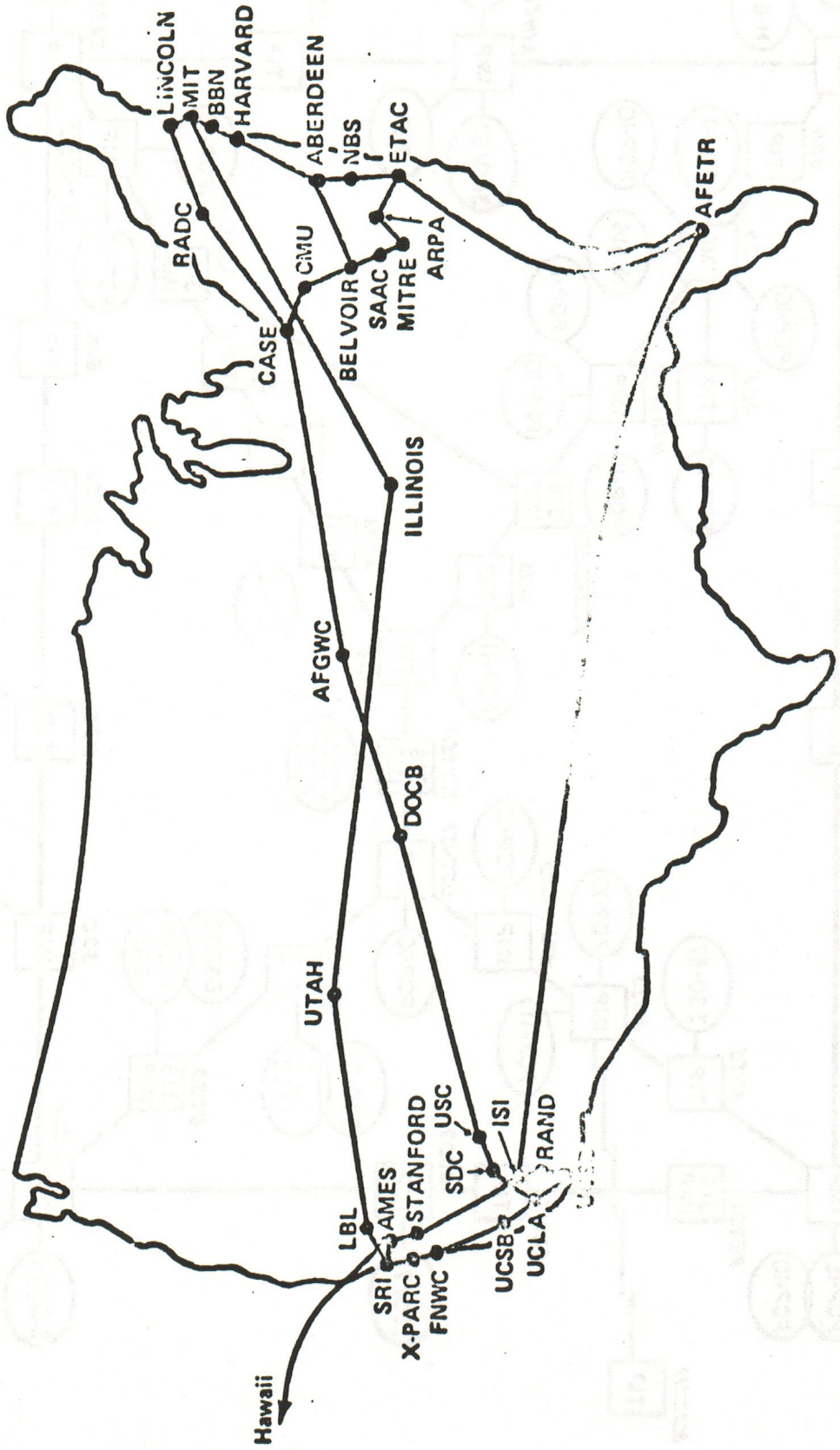


Figure 5.2 The ARPA Network February 1973

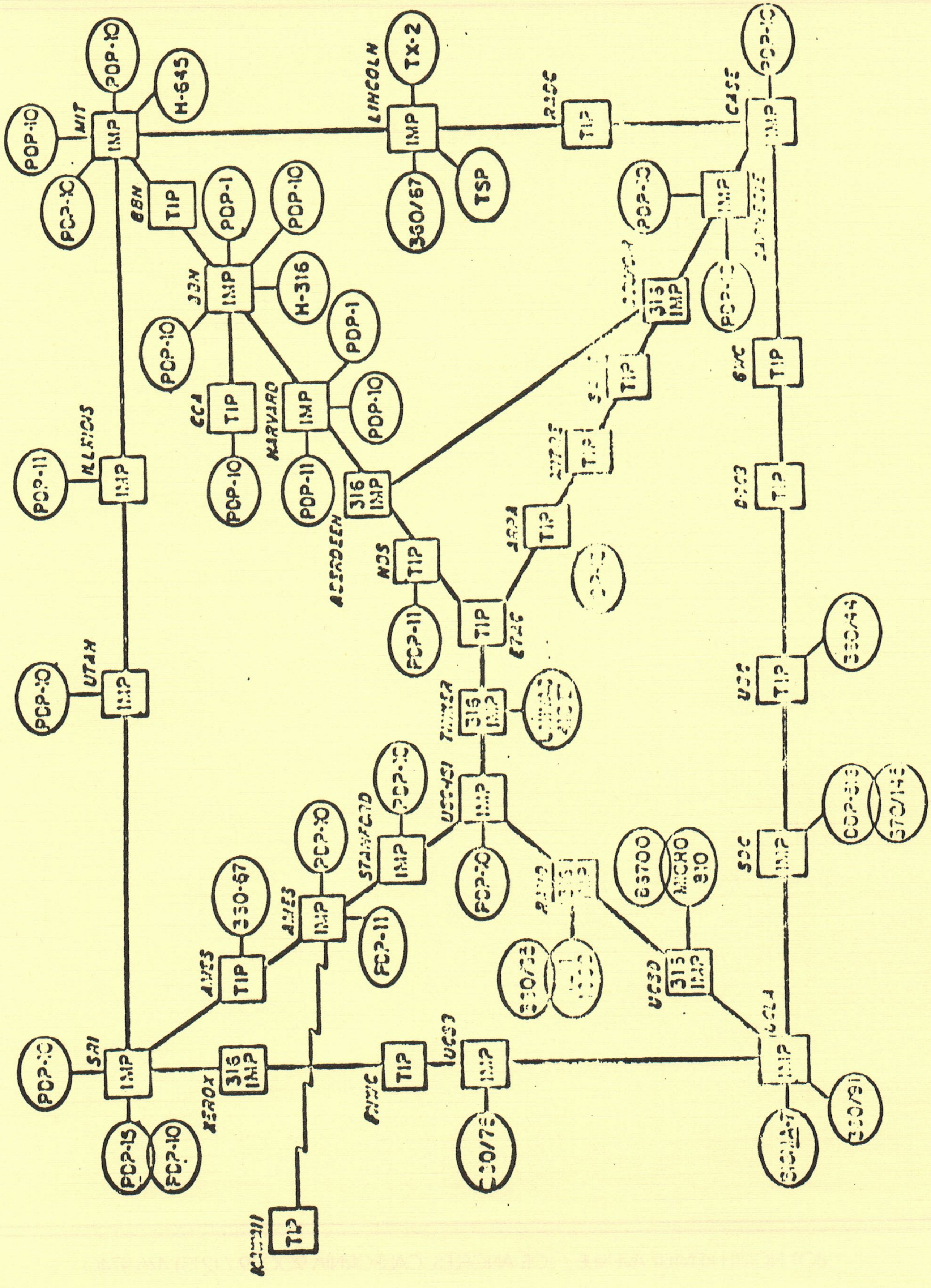


Figure 5.3 ARPA Network, Logical Map, January 1973

gradually spread across country and overseas. Some lines in early versions of the network have since been deleted due to optimization. The nationwide geographical map of the network in February 1973 is shown in Figure 5.2 and the logical map of the network at about the same time is shown in Figure 5.3 The logical map shows the computing facilities at the nodes of the network and the IMP's (Interface Message Processor) and TIP's (Terminal IMP). Since the network operating procedures were designed not to interfere with existing facilities the IMP's and TIP's are there to carry out the message handling tasks such as relay, acknowledgement, routing etc. In designing such a network there are three factors, firstly the physical layout that is the IMPS, lines and hosts, secondly the traffic in the network and finally the operating rules which include such things as protocols, flow control procedures etc. These three factors must be considered separately and then designed to fit together.

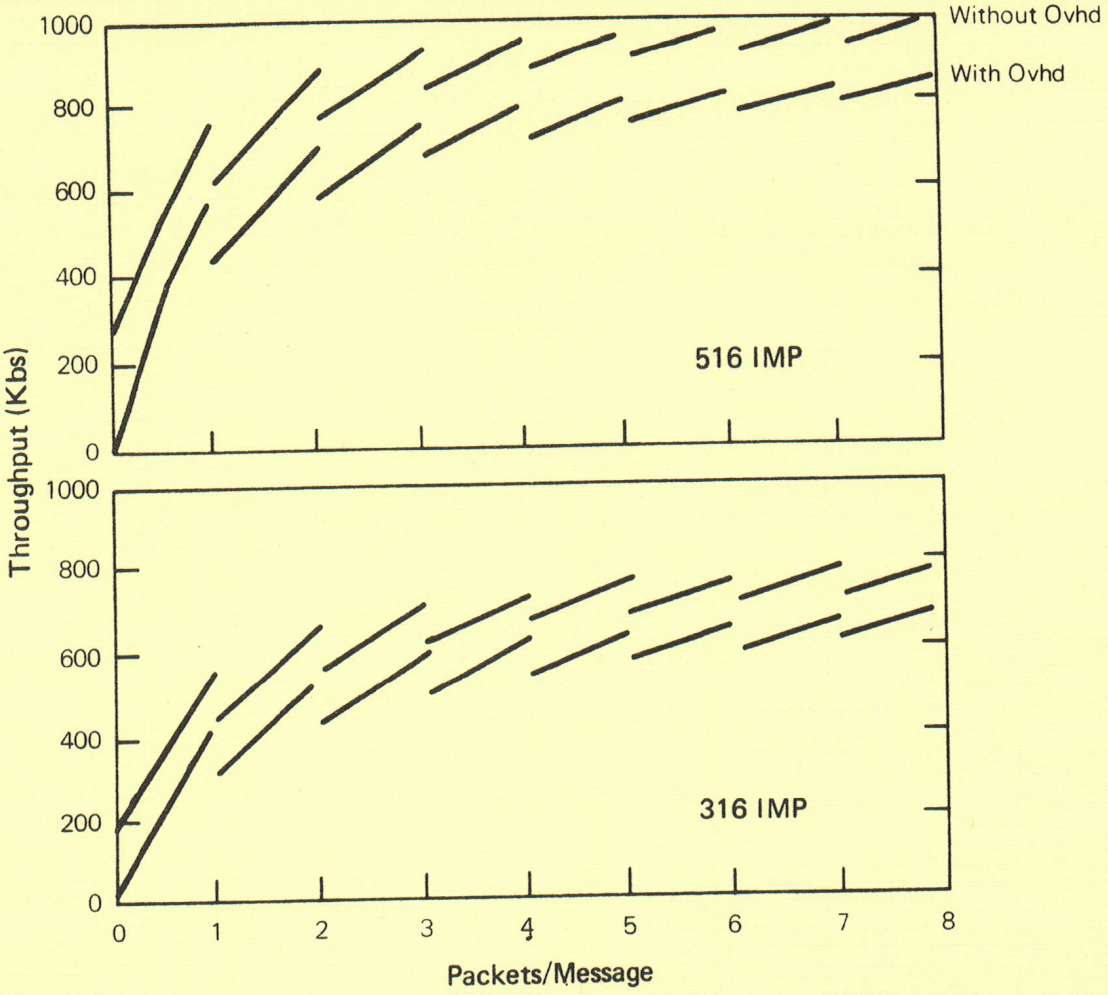
Two types of IMP are currently used and these are compared below.

	Honeywell DDP - 516	Honeywell DDP -316
Maximum Throughput (approximate)	850 KB/Sec.	700 KB/Sec.
Word length	16 bits	16 bits
Store size	12K*	12K*
Cycle time	0.96 μsec.	1.6 μsec.
Cost (approximate)	$100,000	$50,000

As regards overhead the IMPS use 168 bits of transmission overhead/pkt and as the data maximum is 1008 bits/pkt the maximum packet size is 1176 bits. The core within the IMP (for the 12K case) is divided into 24 pages, 512 words/page. Most of this is used for program and data and approximately 40 buffers are left for packets (97 buffers in the 16K version).

The throughout picture of the IMP is shown in Figure 5.4. Also in the network there are TIP's which can combine an IMP and a host and a typical TIP configuration in the network is shown in Figure 5.5. Without a local host a TIP is little more than a relay station in the network.

* upgraded to 16K



IMP PROCESSOR THROUGHPUT VS. MESSAGE LENGTH
The higher curves plot Line Capacity, the lower curves plot Net Throughput.

Figure 5.4

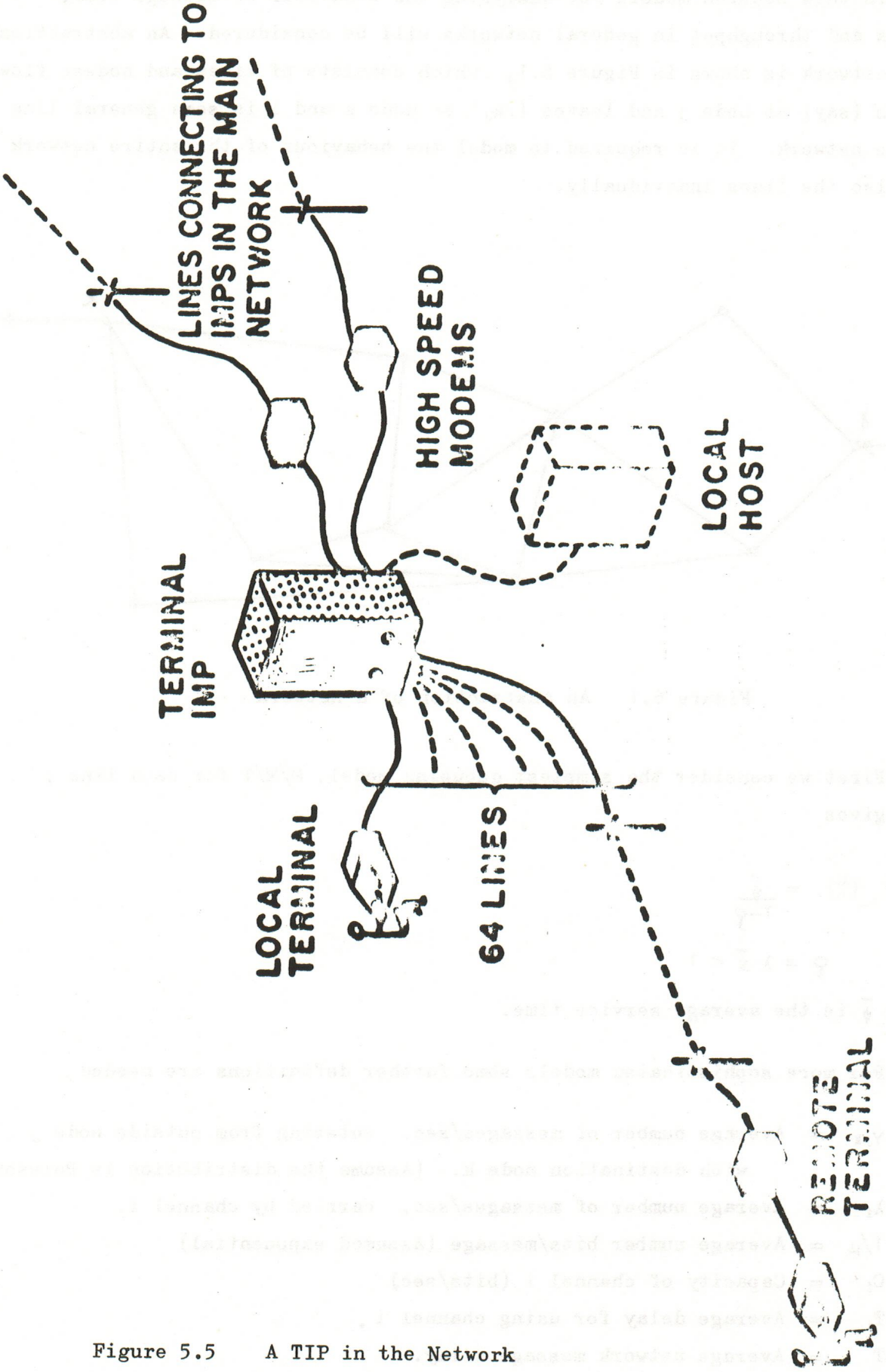


Figure 5.5 A TIP in the Network

In this section models for analyzing the behaviour of message flow, delays and throughput in general networks will be considered. An abstraction of a network is shown in Figure 6.1, which consists of lines and nodes; flow enters (say) at node j and leaves (say) at node k and i is some general line in the network. It is required to model the behaviour of the entire network and also the lines individually.

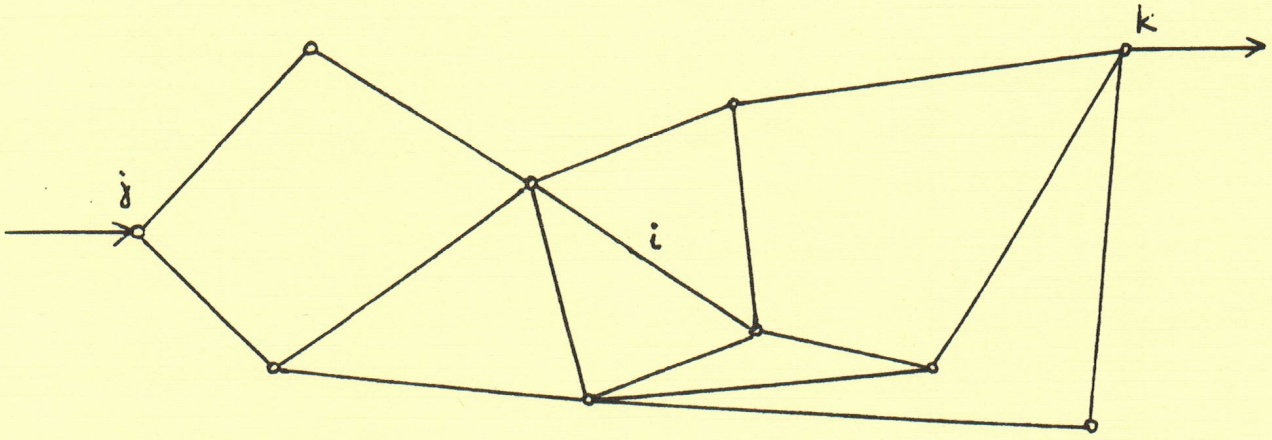


Figure 6.1 An abstraction of a network

First we consider the simplest queueing model, M/M/1 for each line ; this gives

$$\text{Delay} \quad (T) = \frac{\bar{\psi}}{1-\rho}$$

$$\rho = \lambda \bar{\psi} < 1$$

where $\bar{\psi}$ is the average service time.

For more sophisticated models some further definitions are needed .

Let γ_{jk} = Average number of messages/sec. entering from outside node j with destination node k. (Assume the distribution is Poisson)

λ_i = Average number of messages/sec. carried by channel i.

$1/\mu$ = Average number bits/message (Assumed exponential)

C_i = Capacity of channel i (bits/sec)

T_i = Average delay for using channel i .

T = Average network message delay.

Consider the model of a node in the network (see Figure 6.2).

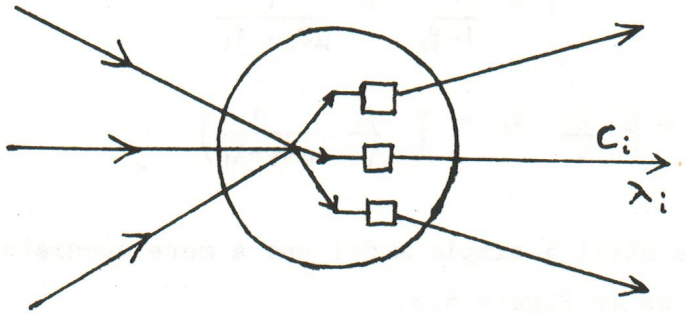


Figure 6.2 Model of a node in the network

$$\bar{\psi}_i = \frac{1/\mu}{C_i}$$

$$\rho_i = \lambda_i \bar{\psi}_i = \frac{\lambda_i}{\mu C_i}$$

Total number of messages entering network per sec. on average

$$\gamma = \sum_j \sum_k \gamma_{jk}$$

Total number of messages in network $\lambda = \sum_i \lambda_i$

The ratio of λ to γ is important and is in effect the average path length of the network.

Let T (the average message delay) be the network performance measure. We decompose T into simple components; consider the j-k traffic and let Z_{jk} be the average delay from node j to node k.

$$T = \sum_j \sum_k \frac{\gamma_{jk}}{\gamma} Z_{jk}$$

It can be shown that further decomposition of this formula into single channel delays T_i gives

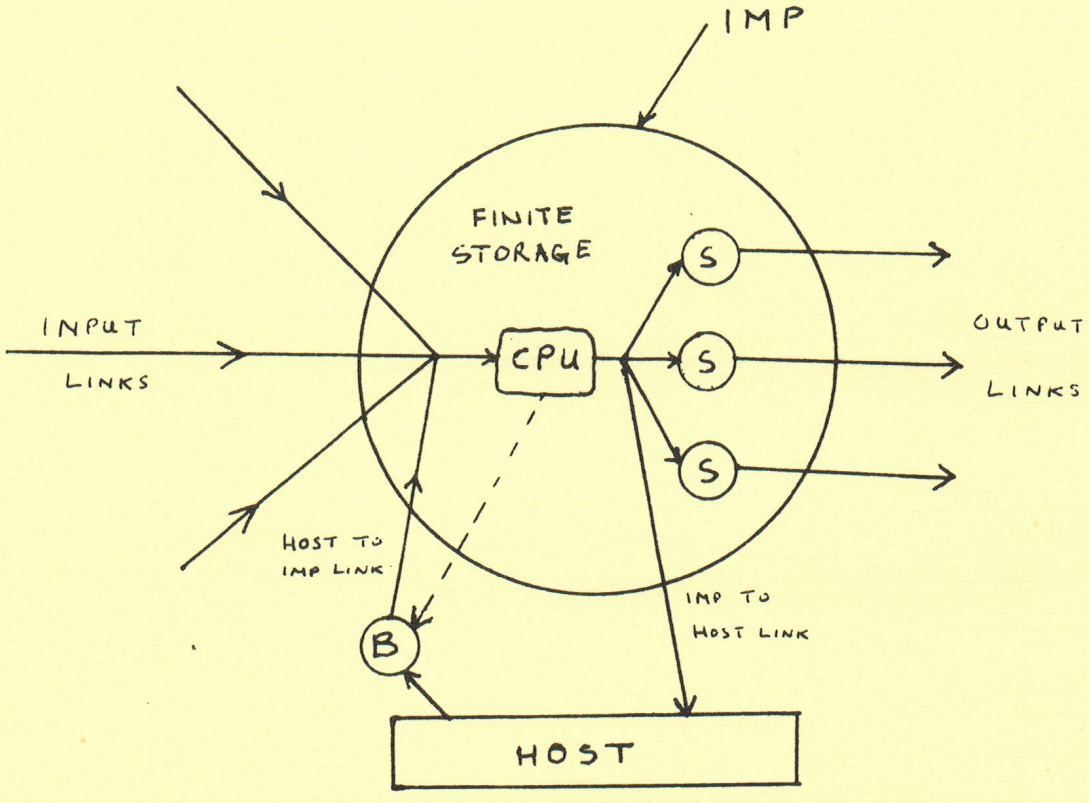
$$T = \sum_i \frac{\lambda_i}{\gamma} T_i$$

Thus the analysis reduces to finding λ_i and T_i; λ_i is obtained from the routing procedure and it will be assumed to be a <u>fixed</u> routing procedure. To find T_i we assume each node to behave like an M/M/1 system; this is not strictly true but turns out to be a good approximation.

$$T = \frac{\psi_i}{1-\rho_i} = \frac{1}{\mu C_i - \lambda_i}$$

$$\text{Thus } T = \sum_i \frac{\lambda_i}{\gamma} T_i = \sum_i \frac{\lambda_i}{\gamma} \left(\frac{1}{\mu C_i - \lambda_i}\right)$$

This is still a simple model and a more accurate one for T_i would be constructed as in Figure 6.3.



B : Switch which can block traffic flow to the IMP

S : Server

Figure 6.3 A more complex model of a node in a network

Let

K = Nodal processing time (constant)

P_i = Propagation time ($\sim$ speed of light) for C_i

$1/\mu'$ = Average length of data message

$\frac{1}{\mu}$ = " " " mixed (data and control) traffic

Then $$T = K + \sum_i \frac{\lambda_i}{\gamma} \underbrace{\left[\frac{1}{\mu' C_i} + \frac{\lambda_i / \mu C_i}{\mu C_i - \lambda_i} \right]}_{T_i} + P_i + K$$

Simulation of a 19-node network under a fixed routing policy shows that the analysis works very well; if the analysis is extended from single-packet messages to multipacket messages the situation is more complicated but still analysable.

For a typical queueing system the delay is plotted as a function of throughput in Figure 6.4. The curves A and B are interesting in that they suggest a very simple two-parameter model as shown in Figure 6.5, which is essentially a D/D/1 model. The two quantities T_o and γ^* are found as follows

$$T = \sum_i \frac{\lambda_i}{\gamma} \left(\frac{1}{\mu C_i - \lambda_i} \right)$$

T_o = "no-load" delay

$$= \sum_i \frac{\lambda_i}{\gamma \mu C_i}$$

To find γ^* :

1. Place any load γ on the network
2. Calculate $\rho_i = \frac{\lambda_i}{\mu C_i}$ for each channel
3. Let i_o be index of largest ρ_i
4. Scale γ until $\rho_{i_o} = \frac{\lambda_{i_o}}{\mu C_{i_o}} = 1$
5. This γ is then γ^* .

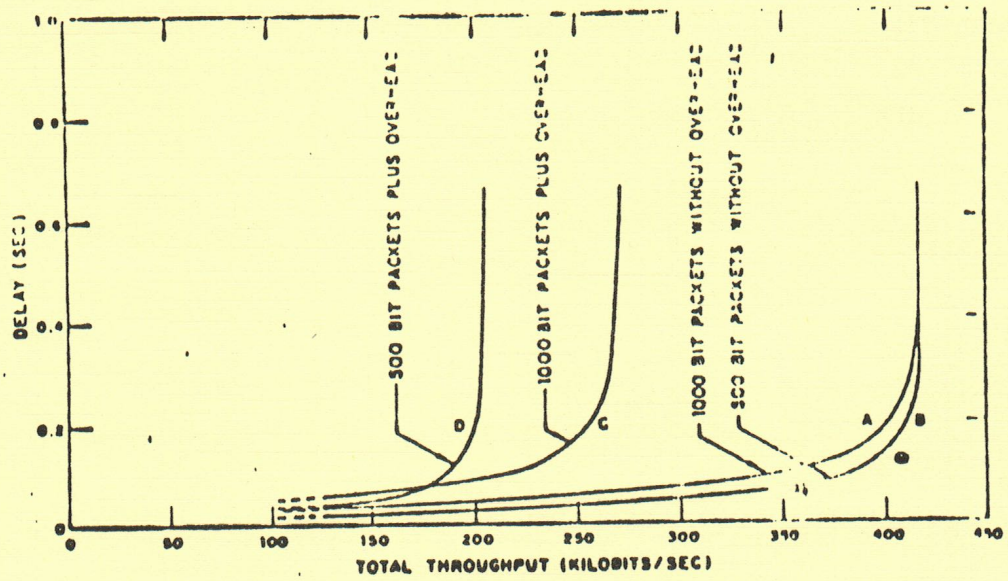


Figure 6.4 Delay as a function of throughput (network load).

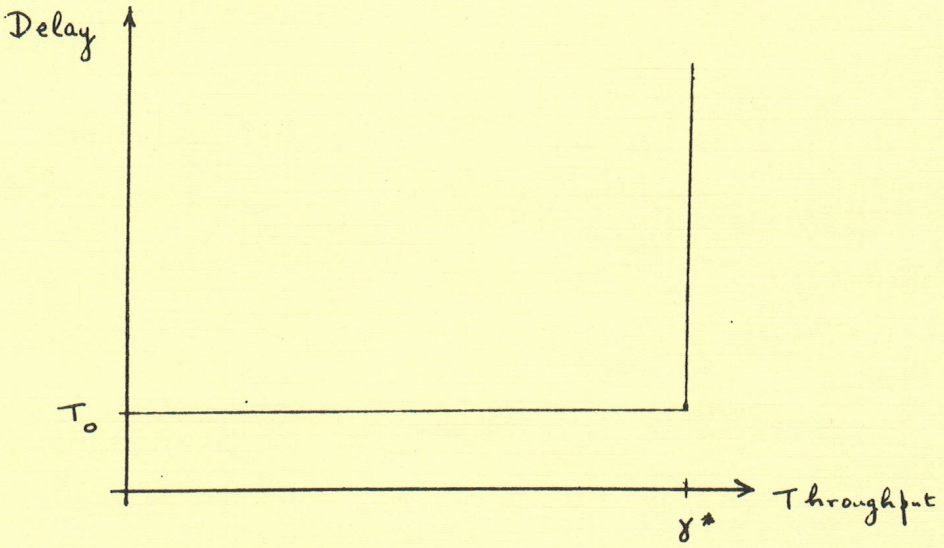


Figure 6.5 Simple 2-parameter model

7 Design Methods for Computer Networks (Kleinrock 1964, 1974)

This is a much harder problem than that tackled in the previous section. The problem is to minimize the delay $T = \sum_i \frac{\lambda_i}{\gamma} T_i$ by varying the three parameters (Topological design, Routing procedure, and capacity assignment for each channel) subject to a fixed maximum cost D and also to given external traffic requirements.

There is a dual problem - design the cheapest network at an acceptable delay. The level of difficulty of these problems depends mostly on the form of D.

Three separate problems will be considered.

1. Capacity Assignment only (CA)

Minimize T (varying capacity assignment) subject to a fixed D and given the topology and routing (λ_i).

2. Flow Assignment only (FA)

Minimize T (varying the routing) given the topology and capacities. This problem is no longer subject to D.

3. Capacity and Flow Assignment (CFA)

Minimize T (varying both capacities and routing) subject to a fixed D given a fixed topology.

In no case are we varying topologies; this is very difficult but the analysis of the third problem does allow us to look at different topologies.

Before going on to these problems the routing procedures and flow control procedures need consideration. Routing procedures involve flow allocation strategies and adaptive routing strategies. The strategies can be shortest path, minimum time path, shortest path with excess capacity, or a linear program algorithm. The properties required in routing procedures are fast adaption to status changes, low cost and reliable implementation and efficiency.

The Flow Control procedures need to prevent congestion and lock up, the latter of which can occur both in reassembly and store and forward. It is also required to deliver short messages rapidly and give a high bandwidth transfer of long files. Unfortunately these latter two requirements usually conflict.

7.1 C.A. Problem

(a) Linear Costs (continuous) $D = \sum_i d_i C_i$

Solution: $C_i = \frac{\lambda_i}{\mu} + \frac{D_e}{d_i} \frac{\sqrt{\lambda_i d_i}}{\sum_j \sqrt{\lambda_j d_j}}$ square root assignment

D_e is "excess" cost $= D - \sum_i \frac{\lambda_i d_i}{\mu} > 0$

$$T = \frac{\bar{n}}{\mu D_e} \left(\frac{\sum_i \sqrt{\lambda_i d_i}}{\lambda} \right)^2$$

where $\bar{n} = \frac{\lambda}{\gamma}$ is average path length.

(b) Logarithmic costs. $D = \sum_i \log \alpha C_i$

$C_i = K_\alpha \lambda_i$ proportional capacity assignment.

(c) Power law costs $D = \sum_i d_i C_i^\alpha, 0 \leq \alpha \leq 1$

C_i satisfies a polynomial equation which can be solved numerically.

(d) Discrete Options.

These are difficult since it is essentially an integer programming problem, however continuous approximations give results which are surprisingly good.

7.2 F.A. Model

Before looking at the analysis for Flow assignment it is interesting to consider the differences between the models and real life situations. The basic trouble is that whilst the models minimize delays and find the optimal channel flows, λ_i, they give no indication of how to achieve this in the network, i.e. inside the IMP.

The routing procedure algorithms for these models can vary in sophistication. They are usually dynamic and thus vary with time. The simplest is periodic update (PU) and works as follows: each node informs its immediate neighbours of the minimum delay time from it to every other node in the network. Neighbouring nodes interchange this information synchronously every so often. Therefore at any instant in time each IMP by adding the appropriate delay time to the immediate neighbouring nodes can find the minimum delay path from itself to any other node in the network. The minimum path will vary dynamically. The actual queue lengths are used to estimate the delay times used in the algorithm.

Another algorithm is AUA (asyncronous update algorithm). In this variation nodes inform their neighbours only when the delay time information changes and

then they inform them immediately. In the S.Q. + B + P.U. (shortest queue plus bias plus periodic update) algorithm a node looks only at its queues and sends the message on the shortest, however a certain amount of bias is introduced in that information is known as to what the actual shortest path to the destination node is. The periodic update is there for safety so that the algorithm can respond quickly to such things as line breaks. These algorithms have not yet been implemented but simulation shows them to be very promising.

Basically the Flow Assignment problem is not difficult. We are required to find channel flows, λ_i, and to minimize T given the net topology. channel capacities (C_i) and flow requirements (γ_{jk}).

T is a convex function of $\frac{\lambda_i}{\mu}$ and the flows themselves form a convex set; thus T has a unique minimum.

7.3 CFA Problem

It is a difficult problem to try to do both flow and channel assignment simultaneously.

Consider $D = \sum_i d_i C_i$

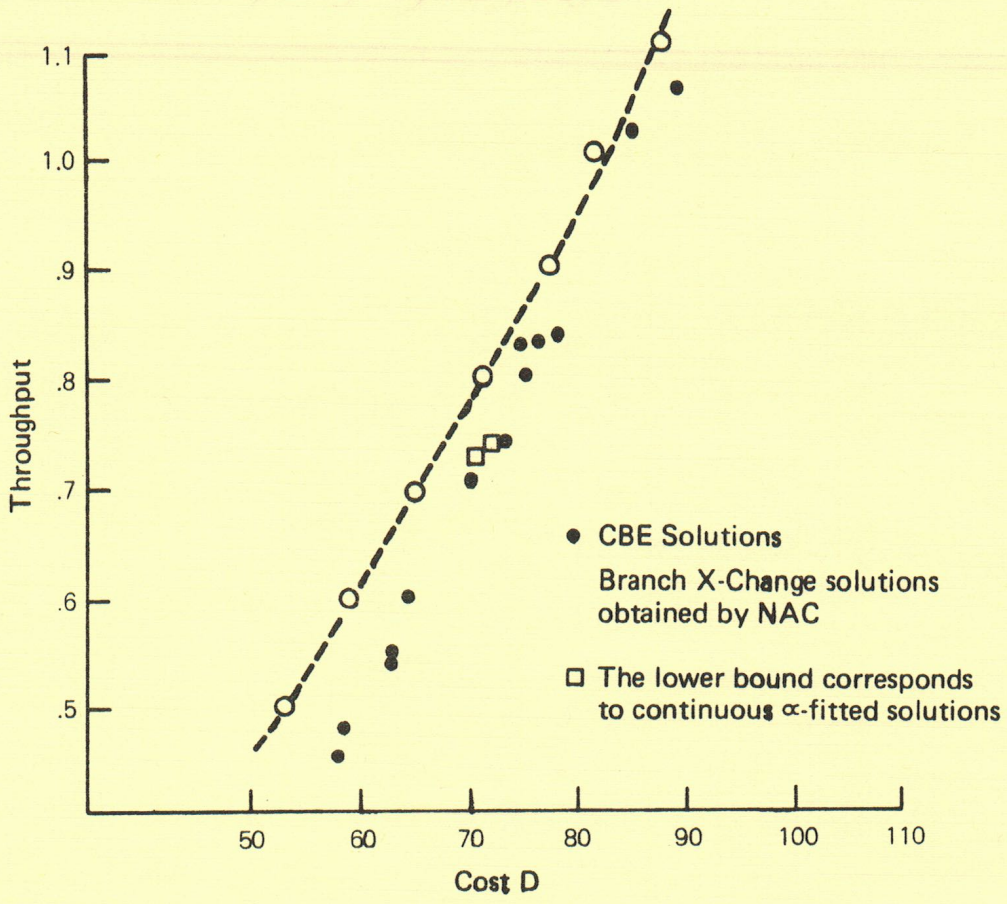
The solution to minimize D for $T \leq T_{max}$ given the capacity is

$$D = \sum \frac{\lambda_i d_i}{\mu} + \frac{1}{\gamma T_{max}} \left(\sum \sqrt{\frac{\lambda_i d_i}{\mu}} \right)^2$$

which is concave with respect to $\{\lambda_i\}$.
The basic algorithm is the Concave Branch Elimination Method (CBE):

1. Initialize the topology.
2. Assign lengths at random.
3. Assign $\{\lambda_i\}$ by F.A. problem
4. Assign $\{C_i\}$ by linearized CA problem and if no "significant" improvement store the result and then go to step 2.
5. Compute new $l_i = \frac{\partial T}{\partial(\lambda_i/\mu)}$ and go to step 3.

The method has the interesting property of eliminating lines by letting their length dwindle down to zero and thus in a sense designs the topology. The design solutions for such a problem are compared in Figure 7.1 in which throughput is plotted against cost.



COMPARISON OF SOLUTIONS FOR THE DESIGN OF NETWORKS

Figure 7.1

If

$$D = \sum d_i \cdot C_i^{\alpha}$$

and we let α decrease then we find the following properties in the design process:

a) Stronger economies of scale are present

b) Few large channels are best

c) For $\alpha \leq 0.6$ only trees are found

d) Variation in the number of channels decreases

e) Variation in cost increases

f) Number of local minima increases

g) Cost changes quickly.

If $0.8 \leq \alpha \leq 1$ a highly connected initial topology is more likely to contain the optimum. The number of local minima is small as is the cost variation. Thus the CBE method is effective.

If $0.5 \leq \alpha \leq 0.8$, the CBE method works only if the initial topology is carefully chosen. For $\alpha \leq .5$ the CBE method is not good and branch exchange is probably better. This is because a highly connected topology will settle on a poor local minima and we would expect economies of scale to favour low-connected topologies.

8 Arpanet Measurement and Performance (Kleinrock 1974)

UCLA has the job of measuring the performance of the ARPA network and for this purpose there are elaborate measurement software packages in each IMP which can be turned on by any host. This software can measure averages taken over long intervals, or take snap-shots or trace the passage of messages through the network.

For example the ARPA network map at any instant in time can be drawn. In August this year 7 day statistics were accumulated for the hosts and channels. These showed some interesting figures, for example most messages fitted into one packet and multi-packets were seldom used. Also buffer size is much larger than the average packet size.

The average shortest path between sites was measured along with the traffic. This showed a lot of people talking to near neighbours or even themselves using the IMP as a switch. Another series of tests showed that the present Arpanet will saturate in 1974 when it reaches 10 million packets per day. Finally a particular demonstration of the Arpanet allowed us to get a bench-mark for the cost of communications as a proportion of the computing cost. This indicated that 365 packets was equivalent to one dollar of computing. From this we were able to conclude that the network

was cost effective and that as networks get larger the relative proportion of the communications cost will go down.

References

Burke 1956 — Burke, P.J. "The Output of a Queueing System", Oper Res., Vol. 4, No. 6 (December 1966) pp. 699-704.

Frank 1971 — Frank, H., and I.T. Frisch, Communication, Transmission and Transportation Networks, Addison-Wesley Publishing Co., Reading, Mass., 1971.

Gordon 1967 — Gordon, W.J. and Newell, G.F., "Closed Queueing Systems with Exponential Servers", Operations Res., 15 (1967) pp. 254-265.

Jackson 1957 — Jackson, J.R., "Networks of Waiting Lines", Operations Res. (1957) 5:518-521.

Jackson 1963 — Jackson, J.R., "Jobshop-Like Queueing Systems", Man. Sci., 10, 1 (October 1963) pp. 131-142.

Kleinrock, 1964 — Kleinrock, L., Communication Nets; Stochastic Message Flow and Delay, McGraw-Hill, New York, 1964. Out of Print. Re-printed by Dover Publications, New York, 1972.

Kleinrock, 1974 — Kleinrock, L., Queueing Systems, Volume I: Theory, Volume II: Computer Applications, Wiley-Interscience (New York) 1974.