PRIORITIES AND PAY FUNCTION ROUTING FOR A PACKET SWITCHING NETWORK

A. Butrimenko

U. Sichra

September 1975 WP-75-117

Priorities and Pay Function Routing for a Packet Switching Network

A. Butrimenko
U. Sichra

In every network there arises the problem of choosing a path for a message so that it reaches its destination. If there are many users requesting the same channel it could be occupied and another would have to be chosen, provided that it would also be a good path and free, otherwise the message would have to queue up.

The choice of an objective function for the selection procedure of a path is part of the decision process when constructing the routing algorithm. The selection is carried out with respect to an objective function which is maximized (or minimized). There exist some useful objective functions commonly used for routing in a computer network: minimization of distance, minimization of delay or cumulative costs, maximization of throughput or reliability, etc.

The cumulative delay along a path is very often taken as an objective function for the routing algorithm, its value being used as a performance criterion to be compared with that of another network or routing strategy. But it is clear that the average delay or average delivery time is not detailed enough to be used when comparing different performances, especially as other features of the system are disregarded.

A multidimensional optimization would probably give better comparative parameters, but as the solution is of real-time nature such a procedure would be too long and complicated. Here again the problem would arise of which out of several parameters should be taken as a comparative value.

One of the difficulties is that in every network there are messages (or jobs) of varying importance. Usually this problem is solved by assigning priorities to each class of messages (jobs), mostly only a few priorities.

We will now describe a particular algorithm for handling messages in a store and foreward network. This can be described as a decentralized, adaptive, markovian algorithm [1].

Decentralized means that there is not a single node or a set of nodes which are responsible for the routing in a whole network, but that each node constructs its own routing matrix on the basis of information gathered from neighbouring nodes, and at the same time no node has information on all the possible routes in the network.

The information at every node is regularly updated, leading to a general adaptivity of the network. The messages in each node are handled independently of the source, and this is thus a markovian-type routing. We will work with this method, which is almost an extension of the methods described in some earlier publications [1,2] and is now in use in the ARPA network. We then extend this method of routing messages for cases with a few priorities. It will be seen that the simple method of handling messages of various priorities has some awkward features and that the philosophy of fixed priorities can be wrong in many cases. Even more so, it does not reflect the proper interests of the users.

We will describe another method based on so called pay functions [3]. It will be shown that the first method with normal fixed priorities is an extreme case of the pay function method. And what is more important, the accuracy of conventional methods to actual requirements of the users can be evaluated by comparing the results with those originated by the introduction of pay functions.

Later on we will also discuss some results of the network simulation with various types of pay functions.

We will limit our study to a store and forward network, where all the messages have the same length and each one propagates through the network as a whole.*

*These constraints can be taken off, but are introduced here for the sake of simplicity.

Suppose that each message belongs to one of three priority classes, and that the priority does not change with time. This means that on every channel we have three queues and that the queues of lower priority will be served only if the queues of higher priority are empty. We assume that in each queue the messages are served in first-in first-out order.

Updating

The routing information is stored in each node, in the form of three matrices, one for each priority. In each of these matrices the column numbers correspond to the number of outgoing channels and the row numbers correspond to the number of destination nodes. The element $r^m_{ij}(p)$ in the matrix M gives the estimated delivery time from node m to destination i through channel j (internal numbers for nodes) for a new message with priority p. Figure 1 which represents a very small network shows the use of these matrices.

For every priority we have a separate routing matrix. The matrix of a small priority is a function of higher priority matrices.

To clarify this, let us look at the first priority, i.e. $p = 1$. In Figure 6 beside each node is shown the routing matrix for $p = 1$, where the rows correspond to the destination nodes to be reached from this particular node, and the columns are the outgoing channels. In the example of Figure 1, the channel number corresponds to the node number at the far end of each channel. Each node also has a vector, and each element of it is the smallest value $\neq 0$ per row, representing the time needed to deliver a message from that node to the various destination nodes omitting the channel number, which is not important at this time.

When a sufficiently large queue forms at an outgoing channel of a node or, if the sum of the changes in the outgoing queues becomes more than a certain value the vector at that node adapts itself to the new situation.

After the adaptation, the new vector elements are again the minimum of each row, but the rows are now the sum of the old row's elements plus the queue of each channel. To every element in the row there corresponds a different queue (as queues and columns correspond to each other). The elements of the new vector are not the sum of the old matrix element plus queue length, but again only the old element, taking into consideration what would be the minimal if the queue lengths were added. This new vector is sent to the adjacent nodes who adapt the routing matrix for $p = 1$ accordingly, putting the new vector's element in the right place, and their vectors possibly changing too.

For $p = 2$ the routing matrix and, at the same time the vector are not only influenced by the queue of the packets with priority 2, but also by the queue for $p = 1$. The matrices for $p = 2$ relate always directly to the matrices with $p = 1$. In each node the elements of the matrix for $p = 2$ are larger or equal than the elements of $p = 1$. The same is true for $p = 3$. For the sake of simplicity the simulated algorithm only allows the transmission of the vector of one priority at a time, and starts to look for possible changes from $p = 1$ on (or from the last priority on, if one wants it), POROG(i), $i = 1, 3$ is the variable which decides if the changes are large enough to allow for an adaptation.*

An Example of a Simulation

The simulation is done on a 16 nodes and 60 channels network (Figure 3). After having generated 17,500 packets in a time of 1,592.39 simulation seconds with POROG(1) = 3, POROG(2) = 6, POROG(3) = 9, the routing matrices should have been updated 85,049 times, but 28,002 cases were collisions with other updatings at the same node in the same time interval.

*POROG can be constructed as adaptive, changing with the loading of the system in general, but here it is supposed to stay the same throughout the whole run.

The updating frequencies per node, INFUB(i), varied between 4,956 updatings at node 1 and 2,746 updatings at node 13, not counting collisions. As mentioned before only the matrix of one priority is updated at a time, but the cases of necessary updating for lower priorities are also counted.

The matrices of priority 1 were updated 21,259 times, on all nodes together, the matrices for priority 2 were updated 27,540 times and the matrices for priority 3 were updated 36,258 times. The collisions occuring were included in these figures. In the case where the matrices of priority 1 were updated, matrices of priority 2 should have been updated 7,013 times, and matrices for priority 3 should have been updated 34,910 times.

The time needed to transmit an updated vector to a neighboring node was 0.1 simulation seconds.* It was also assumed that the updating is sent over special control channels and does not disturb the transmission of messages in the data communication system.

Priority Philosophy

Most batch processing or time sharing systems have priority queues. The user can attach to his messages or jobs a certain priority in order to show its importance. He also thinks that jobs with highest priority will be run first and finished first, so that the turn-around time is the lowest. But due to the changing flow in a network it is impossible to predict the completion time of a job being put into a network. Therefore, the priorities do not always coincide with the user's interests.

The following three reasons make the design of the priority system extremely complicated:

1. There is no method, except simulation, to calculate the delivery time for each priority even for a relatively small net for an assumed loading;

*It is fixed constant in the program.

2. The assumptions for the expected loading are as a rule different from reality and a network mostly operates under other conditions than those planned;

3. When using the priority system it is not possible to compare different performances which originated by different methods (order of updating, frequency of updating, queueing strategy), thus it is impossible to know which method gives the best result.

The only guarantee a priority philosophy can give is that on the same channel messages with higher priorities will be sent earlier than the ones of lower priority, but that does not say anything about the performance of the whole network.

Pay Functions

We will now introduce the so called pay functions [3]. The main idea arises from the assumption that between the authorities responsible for running the network and the users of it, an agreement on charges per time unit can be achieved for the user, depending on the performance of each particular user's job.

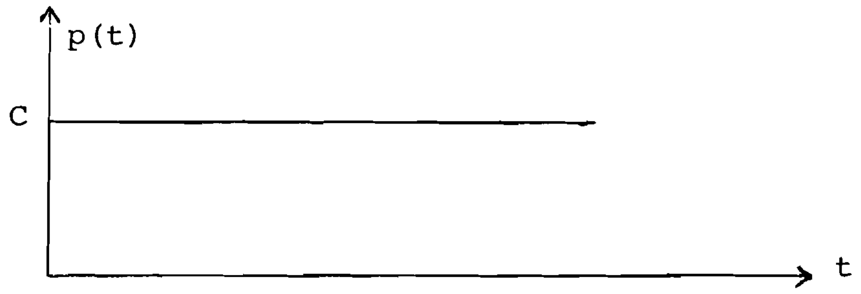
For example, the authorities running the commercial CDC 6600 System in Frankfurt, which is one of the systems used by IIASA scientists, have an agreement that the user can choose between 5 priorities, and the charge for each system second used will depend on the priority given to his job and not on the delivery or turn-around time. What we are looking for is a pay function depending on the delivery time, and the charge for the user will then depend on this too.

It is possible to think of various kinds of pay functions:

a) The pay function is a constant with time:

$$p(t) = C$$

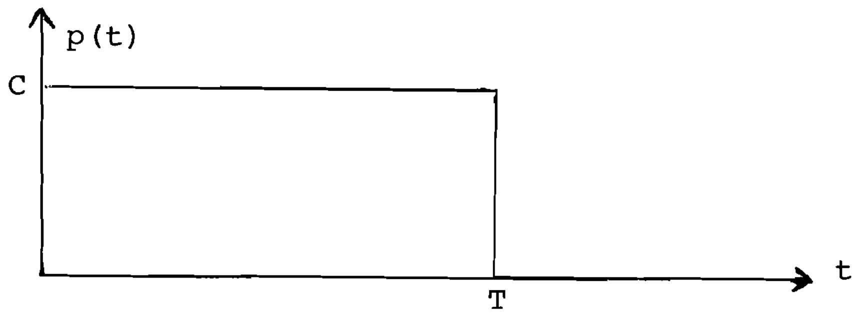
Such functions are used at CDC 6600 as mentioned above.



b) The pay function is a constant not equal to 0, up to a certain value of t, then it becomes 0 for all other values greater than that T.

$$p(t) = \begin{cases} C & t \leq T \\ 0 & T > t \end{cases}$$

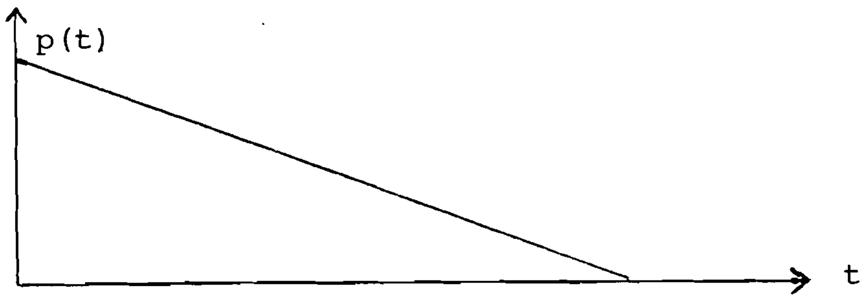
This function can apply to cases where a message is important until a certain time, when it becomes useless to the user.



c) The pay function is linear:

$$p(t) = d - k \cdot t$$

In this case the queueing strategy is independent of the age of the messages to be queued (which will be shown later).

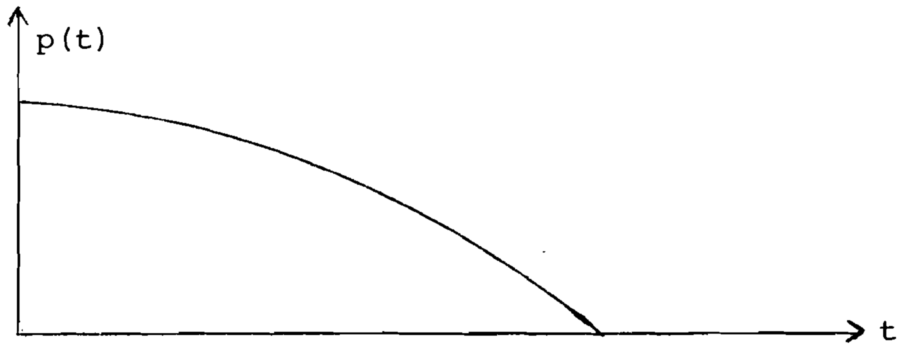


d) The value of a message decreases very slowly at first but rapidly at a later stage, i.e.:

$$\frac{dp^2(t)}{dt^2} \leq 0$$

A pay function of this kind could be:

$$p(t) = c - a \cdot e^{b \cdot t}$$

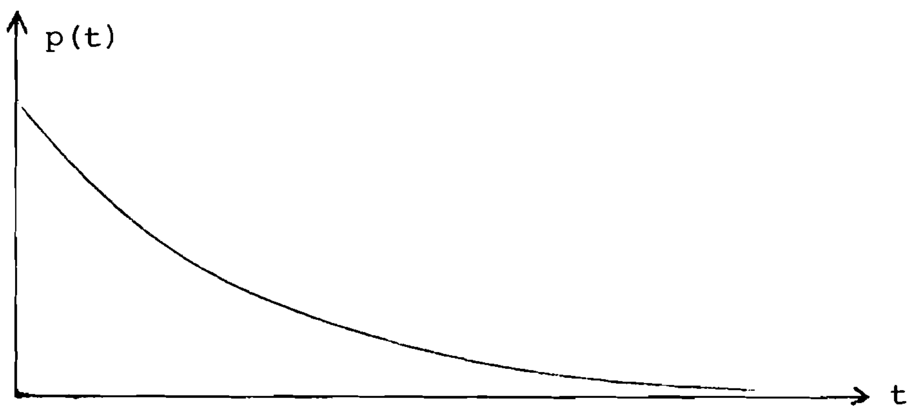


e) The value of the messages decreases very rapidly at first, and later less, not losing the importance completely, i.e.:

$$\frac{dp^2(t)}{dt^2} > 0$$

A pay function with this second deviate could be:

$$p(t) = a \cdot e^{-b \cdot t}$$



The reader can, of course, think of many more pay functions which coincide with his specific interests. In some cases the pay functions can be considered as "isointerest" functions of a user, since it is possible to establish a performance criterion for the network--the amount of its income--and in this case with the non-increasing pay functions the user's and the system's interests coincide. The maximization of this system's income is then an advantage to the user.

Let us point out two features of the pay functions:

1. If we assume that the storage of a message in its destination node does not require any network resources, then we can consider only non-increasing functions with time;
2. The function cannot fall indefinitely low. It means that there is an asymptotic value which can be considered as being "0" in a new system of coordinates.

Therefore we will only consider positive non-increasing pay functions.

Let us now look at the relationship between pay functions and fixed priorities. We want to determine what kind of pay functions can correspond to fixed priorities. No matter which function we take, it must allow for first-in first-out discipline in every priority class. As shown below linear pay functions are the only one's where the queueing strategy has no influence and therefore can be taken to compare the priority philosophy with the pay function idea.

As mentioned above the non-increasing pay functions allow for coincidence between the user's and the system's interests and we will therefore concentrate our interests now on non-increasing linear pay functions.

The proof follows that linear pay functions are the only one's where the queueing strategy has no influence.

Formulation of the problem:

t_i : age of a packet when arriving at the queue;

$f(z)$: pay function.

$$X_{ij} = \begin{cases} 1 \text{ package of age } i \text{ on place } j \text{ in queue} \\ 0 \text{ else} \end{cases}$$

and

$$\sum_i X_{ij} = 1 \quad , \qquad \sum_j X_{ij} \leq 1 \quad .$$

The queueing strategy has no influence, i.e.

$$\pi \left(\sum_{i=1}^{n} \sum_{j=1}^{m} X_{ij} \, f[t_i + (j-1)] \right) = C \quad , \quad \forall \pi \varepsilon \; P_i \times P_j$$

and

$$P_i = \begin{pmatrix} i_1, i_2, \dots, i_n \\ i_{v1}, i_{v2}, \dots, i_{vn} \end{pmatrix} \; ,$$

permutations of the $i_1, \dots, i_n$, the same for P_j, the permutations of the $j_1, \dots, j_m$. $P_i \times P_j$ is the cross product of them.

It must be shown that:

$$f(z) = kz + d \Leftrightarrow \pi \left(\sum_{i,j} X_{ij} \, f[t_i + j - 1] \right) = C \quad , \qquad \forall \pi \varepsilon \; P_i \times P_j$$

or

$$f(z) = kz + d \Leftrightarrow f(x + a) + f(y) = f(y + a) + f(x) \quad , \qquad \forall x, y, a$$

1. $f(z) = kz + d \Rightarrow$

$$f(x + a) + f(y) = d + kx + ka + d + ky$$
$$= f(x) + f(y + a)$$

and the right hand side is true.

2. $f(x + a) + f(y) = f(x) + f(y + a) \Longrightarrow f(x + a) - f(x)$
$= C$. $\forall x$

f(z) is piecewise continuous differentiable in any part interval of the whole interval.

Then

$$f(x + a) - f(x) = f(x) + \frac{a}{1!}(1 - \delta)\, f'(x + \delta a) - f(x) =$$

or

$$f'(x + \delta a) = \frac{C}{a(1 - \delta)} \quad .$$

From there it follows that $f(z) = zk + d$ per part interval. As f(x) is piecewise differentiable it means that there exists a x_o:

$$f(x) = k_1 x + d_1 \;, \qquad x \le x_o$$

$$f(x) = k_2 x + d_2 \;, \qquad x \ge x_o \quad .$$

Having $0 < a < \varepsilon$, x: $x_o < x + a < x_o + \varepsilon$, y: $y + a < x_o$, then

$$f(x + a) + f(y) = k_2(x + a) + d_2 + k_1 y + d_1 \;,$$

$$f(y + a) + f(x) = k_1(y + a) + d_1 + k_1 x + d_1 \;,$$

that is

$$x(k_2 - k_1) = a(k_1 - k_2) + d_1 - d_2 = C_1 \;,$$

$\forall x \in$ part interval $\Rightarrow k_2 = k_1$ is the solution and therefore $d_1 = d_2$ and the function is the same in two adjacent part intervals.

This can be done for all pairs of intervals and therefore the function must be a continuous differentiable in the whole interval and be of the form:

$$f(z) = kz + d \quad .$$

Simulation with Pure Priorities

The general idea is to compare the performances of a network under both, pure priority handling and pay functions routing. If it should be possible to compare it, the use of a linear non-increasing pay function is necessary.

Some simulation runs have been done with three different pay functions for each priority with different slopes, the highest priority having the steepest slope (see Figure 2).

The choice of the outgoing channel at a node depended only on the routing matrix elements and the queue length in front of the new packet at a channel. This implies that not only the sum of packets of the same priority as the new packet influences the length, but also the sum of packets of higher priority.

A first run has been done with a creation rate of 16.0115 packets per simulation time unit (ang = 262), and with no updating at all. The choice of a channel has only been done taking into consideration the minimal distance of actual node and destination, and the queues at a channel, but not passing this information to the adjacent nodes. After having created 19,518 packets, 17,513 had arrived, 2,000 were still in the system, and the queue lengths at certain channels increased tremendously.

A maximal number of 2,000 packets are allowed to be in the system at one time, when there are more the simulation stops.

Because of the very large queues at certain channels the time delay of the packets, especially of the third priority, is remarkable. Table 1 gives some values of that run.

If regular updating is allowed, dependent on POROG(i), i = 1, 3, which is the maximum number of changes allowed at the queues of a node before the vector changes to a new configuration, the performance is much better. The network contains a reasonable amount of packets and the time delay of the arrived packets seems to be also reasonable. The creation rate is again 16.0115 p/t.u., the values of POROG(i) are 8, 16, and 32, the simulation is carried out until 20,000 packets have been created, leaving 108 in the system. Table 2 shows some figures of that run.*

The values listed on Tables 1 -6 are:

delay	:	mean time taken by a packet to travel from source to destination;
confidence interval	:	an interval with 95% probability of covering the true mean delay;
largest queue	:	lengths are only measured at a few discrete moments;
income	:	$\sum_{i=1}^{nj} d^j - k^j \cdot t_i$ the income given to the system by all packets of priority j;
ideal income	:	$\sum_{i=1}^{nj} d^j - k^j \cdot tid_i$ the income which the system could have had if the packets were not delayed at all;
loss	:	$\sum_{i=1}^{nj} - k^j \cdot t_i$;

*The run is so short at first in order to allow for an exact comparison with the run on Table 1.

ideal loss	:	$\sum_{i=1}^{nj} - k^j \cdot tid_i$ both are a similar calculation as the income, but independent of the initial value (intersection of the function with the y-axis);
diff of losses	:	(loss - ideal loss) what the system actually loses;
% of ideal loss	:	(diff of losses/ideal loss) $\cdot$ 100 the relative percentage of actual loss of the system;
i loss/r loss	:	a quotient which tells something about the quality of the system. If it is 1, the quality cannot be better. If it is Ø the system is the worst.

All these figures are tabulated for each priority and for all packets together.

From the results of Table 1 and Table 2 it can be deduced that routing without updating causes a too unstable behaviour of the network and unnecessary delays which can be avoided by not routing along the shortest distances only, but also by taking care of the different queues to be met on the way.

In the listed tables the ideal loss or ideal income is one which would have appeared if all the packets had had no delay other than the one caused by the minimal distance necessary to reach their destinations. The closer the achieved loss is to the ideal (the smaller "% of ideal loss"), and the nearer i loss/ r loss is to 1, the better the performance of the network.

The size of POROG(i) has a big influence on the updating frequency. The smaller POROG(i) the more often the vectors are updated and at the same time the more the updating intervals overlap. There were two runs with equal parameters except for POROG(i) which in one case was 3, 6, and 9, and in the other 8, 16 and 32. The creation rate was in both cases 14.98 p/t.u. and the runs were only carried out until 5,000 packets had been generated. With POROG(i) = 3, 6, 9 there had been 15,639 updatings, and 8,232 collisions which resulted in an omission of the updating procedure. With POROG(i) = 8, 16, 32 there had been 5,668 updatings and 650 collisions.

It can be seen that the number of updatings decrease in the same proportion as POROG(i) increase, whereas the number of collisions are much fewer with larger POROG(i). For POROG(i) = 3, 6, 9, 34.49% of the actual updatings are collisions, for POROG(i) = 8, 16, 32 only 10.29%.

A further statistic exists about necessary updatings of other priority classes which have not been done because the procedure only provides for the updating of one priority at one node. For POROG(i) = 3, 6, 9 between 19.3% and 11.8% of the actual updatings should have been made on other priorities. With POROG(i) = 8, 16, 32 it would only have been between 10.5% and 6.8%.

Table 3 shows again a network with updating of the routing tables with POROG(i) = 8, 16, 32, but with a slightly smaller creation rate (15.83 p/t.u.).

The updating procedure first asks the highest priority if updating would be necessary, then the second one etc. If the matrix of one priority needs to be updated, the next priorities are not allowed to. On Table 4 all the values are the same, except that the updating questions start from the lowest priority. But there is no significant difference in terms of delay or "% of ideal loss". The quotient measuring the quality is the same for both methods. The first one will then be preferred because it is simpler.

Simulation with Pay Functions

Our aim is to compare the simple priority handling with the pay function philosophy. It was shown that only with linear pay functions the queueing has no influence. We therefore use those simple pay functions with the parameters listed on Figure 2.

The routing algorithm is such that when choosing the channel for a new packet, it selects the one where the income to the system is the highest. The delay suffered by the packet is then the smallest possible.

The income to the system depends not only on what the new packet is going to give, considering its possible delay, but also on disturbances caused to other packets which will have to wait till the new one has been sent. Those packets waiting are called "rest" and the simulation with pay functions gets the name "priorities with rest."

Table 5 shows some results of a routing with rest. The parameters are the same as for the run of Table 6 (without rest).*

It can be seen that with such a light loading (approx. 16 packets per time unit creation rate) the differences between priority and pay function routing are very slight. The values for the first priority hardly change, in this case the pay function idea has no advantage.

For the 2nd and 3rd priority an improvement can be seen in terms of the quality quotient (losses ideal/loss real) as well as in the time delay. The 95% confidence intervals of the mean time delays do not overlap. At first it seems as if the loading is distributed more evenly over the channels when routing with pay functions. The mean loading over all channels is $\mu_1 = 0.691326$ when routing with pay functions, the correspondent deviation $\sigma_1 = 0.192581$. The routing without pay functions has values $\mu_2 = 0.689277$ and $\sigma_2 = 0.203352$. There is a difference

*The run in Table 6 is the same as in Table 2, only longer.

between σ_1 and σ_2, but a test of the level 0.95 shows that the difference is not relevant:

$$\sigma_1^2/\sigma_2^2 = 1.1149 \quad ;$$

c = 1.48 for the 0.95 level, therefore the hypothesis $\sigma_1 = \sigma_2$ can be accepted.

Table 9 and 10 again show 2 runs with pay functions (rest) and without them. With a creation rate of 16.6 packets per time unit the system is reasonably loaded. After having created 120,000 packets statistics were made concerning income, loss, difference, quality quotient etc. In this case, it can be seen much better that the introduction of pay functions improves the behaviour of priority packets 2 and 3, than at a rate of 16.01. Packets of priority 1 are more delayed and therefore cause a greater loss to the system when the rest is considered. The lower priorities have more influence and thus the overall quality quotient is better when routing with pay functions.

A further run has been done with a very strong loading (30 packets per time unit creation rate). In this case both routing strategies with and without rest, did not permit a settling down of the system. After approximately 6,000 generated packets the system was overloaded, with 2,000 packets in it and the results were meaningless.

The two runs (with and without rest) with a creation rate of 21 packets per time unit again finished after approximately 20,000 generated packets. As it can be seen from Table 7 and 8 the values to compare do not differ greatly. This is mainly due to the fact that the system was unable to settle down after so few packets had been generated.

Conclusions

1. The relations between linear pay functions and pure priority systems have been studied in a particular. example.
2. It was shown that at low loading (for this particular network 16.0 packets per time unit creation rate) there was hardly any difference between the priority system and linear pay functions. That is actually the case when the queues are almost empty. With heavier loading the difference increases up to congestion.
3. The difference increases with the priority. There is no difference for the highest priority and messages with the highest decreasing pay function. It allows one to omit the pay function philosophy for messages with highest decreasing function and to consider them as messages with the highest priority, while messages with lower decreasing functions have to be dealt with taking into account the functions.
4. The performance of the network depends considerably on the selected POROG(i) (thresholds). A further study of the network behaviour with an adapting POROG(i) is planned.

Simulation with no updating

Creation rate 16.01 packets per time unit

	Overall	P1	P2	P3
Delay	22.55	3.7	9.39	67.03
Confidence interval certainty 95 %		3.65 3.75	9.03 9.76	63.74 70.32
Largest queue	144	2	1	141
Income	2 128.36	3 417.09	1 448.06	
Ideal income	10 100.27	4 848.8	3 757.0	
Loss	10 481.29	3 165.91	2 950.74	4 364.64
Ideal loss	2 509.38	1 734.2	641.8	133.38
Diff.of losses	7 971.91	1 431.71	2 308.94	4 231.26
% of ideal loss	317.7	82.56	359.76	3 172.84
i.loss/r.loss	0.24	0.55	0.22	0.03

Number of created packets 19 518

Number of packets in the system 2 000

Porog(i) = 8 , 16 , 32

TABLE 1

Simulation with updating of routing matrices

Creation rate 16.01 packets per time unit

	Overall	P1	P2	P3
Delay	6.43	3.24	4.49	11.57
Confidence interval certainty 95 %		3.2 3.28	4.41 4.56	10.94 12.19
Largest queue	15	1	2	12
Income	8 245.74	3 908.12	3 086.36	1 251.26
Ideal income	10 988.49	4 972.47	3 879.2	2 136.82
Loss	5 375.41	2 842.88	1 455.24	1 077.29
Ideal loss	2 632.66	1 778.53	662.4	191.73
Diff.of losses	2 742.75	1 064.35	792.84	885.56
% of ideal loss	104.18	59.84	119.69	461.88
i.loss/r.loss	0.49	0.63	0.46	0.18

Number of created packets 20 000

Number of packets in the system 108

Porog(i) = 8 , 16 , 32

TABLE 2

Simulation with updating from the first priority on

Creation rate 15.38 packets per time unit

	Overall	P1	P2	P3
Delay	6.39	3.27	4.36	11.49
Confidence interval certainty 95 %		3.25 3.29	4.32 4.39	11.18 11.79
Largest queue	11	2	1	8
Income	39 370.4	18 264.63	15 072.41	6 033.36
Ideal income	52 239.72	23 294.02	18 700.3	10 245.94
Loss	25 422.7	13 472.37	6 819.39	5 130.94
Ideal loss	12 553.38	8 442.98	3 191.5	918.9
Diff.of losses	12 869.32	5 029.39	3 627.89	4 212.04
% of ideal loss	102.52	59.57	113.67	458.38
i.loss/r.loss	0.49	0.63	0.47	0.18

Number of created packets 95 000

Number of packets in the system 91

Porog(i) = 8 , 16 , 32

TABLE 3

Simulation with updating from the last priority on

Creation rate 15.83 packets per time unit

	Overall	P1	P2	P3
Delay	6.34	3.27	4.37	11.33
Confidence interval certainty 95 %		3.25 3.29	4.33 4.4	11.04 11.61
Largest queue	10	0	1	9
Income	39 411.85	18 244.63	15 062.08	6 105.14
Ideal income	52 234.51	23 294.02	18 696.7	10 243.79
Loss	25 375.3	13 492.37	6 825.52	5 057.41
Ideal loss	12 552.64	8 442.98	3 190.9	918.76
Diff.of losses	12 822.66	5 049.39	3 634.62	4 138.65
% of ideal loss	102.15	59.81	113.91	450.46
i.loss/r.loss	0.49	0.63	0.47	0.18

Number of created packets 95 000

Number of packets in the system 102

Porog(i) = 8 , 16 , 32

TABLE 4

Simulation with updating and rest of the queues

Creation rate 16.01 packets per time unit

	Overall	P1	P2	P3
Delay	6.52	3.28	4.29	11.94
Confidence interval certainty 95 %		3.26 3.29	4.26 4.32	11.68 12.21
Largest queue	23	1	4	18
Income	49 553.87	22 967.69	19 229.85	7 356.33
Ideal income	65 982.27	29 367.41	23 687.85	12 927.01
Loss	32 284.48	17 052.31	8 506.95	6 725.22
Ideal loss	15 856.08	10 652.59	4 048.95	1 154.54
Diff.of losses	16 428.40	6 399.72	4 458.0	5 570.68
% of ideal loss	103.61	60.08	110.10	482.50
i.loss/r.loss	0.49	0.62	0.48	0.17

Number of created packets 120 000

Number of packets in the system 123

Porog(i) = 8 , 16 , 32

TABLE 5

Simulation with updating of routing matrices

Creation rate 16.01 packets per time unit

	Overall	P1	P2	P3
Delay	7.04	3.29	4.46	13.3
Confidence interval certainty 95 %		3.27 3.31	4.43 4.49	12.99 13.62
Largest queue	26	1	4	21
Income	48 391.71	22 910.99	18 891.04	6 589.67
Ideal income	65 981.11	29 365.93	23 687.1	1 292.08
Loss	33 445.34	17 107.0	8 845.06	7 493.28
Ideal loss	15 855.94	10 652.07	4 049.0	1 154.87
Diff.of losses	17 589.4	6 454.93	4 796.06	6 338.41
% of ideal loss	110.93	60.60	118.45	548.84
i.loss/r.loss	0.47	0.62	0.46	0.15

Number of created packets 120 000

Number of packets in the system 122

Porog(i) = 8 , 16 , 32

TABLE 6

Simulation with updating of routing matrices

Creation rate 20.95 packets per time unit

	Overall	P1	P2	P3
Delay	22.78	3.61	6.46	70.33
Confidence interval certainty 95 %		3.56 3.65	6.33 6.59	67.13 73.54
Largest queue	284	0	0	284
Income	3 114.08	3 864.23	2 633.28	
Ideal income	11 257.86	5 359.89	4 176.7	
Loss	10 917.57	3 412.77	2 255.52	5 249.28
Ideal loss	2 773.78	1 917.11	712.1	144.58
Diff.of losses	8 143.78	1 495.66	1 543.42	5 104.7
% of ideal loss	293.6	78.02	216.74	3 530.71
i.loss/r.loss	0.25	0.56	0.32	0.03

Number of created packets 21 592

Number of packets in the system 2 000

Porog(i) = 8 , 16 , 32

TABLE 7

Simulation with updating and rest of the queues

Creation rate 20.95 packets per time unit

	Overall	P1	P2	P3
Delay	21.73	4.0	6.56	67.04
Confidence interval certainty 95 %		3.94 4.06	6.42 6.7	64.10 69.98
Largest queue	256	1	2	253
Income	2 763.98	3 181.96	2 357.86	
Ideal income	10 196.97	4 884.45	3 790.5	
Loss	9 956.32	3 449.04	2 080.84	4 426.44
Ideal loss	2 523.32	1 746.55	648.2	128.57
Diff.of losses	7 433.0	1 702.49	1 432.64	4 297.87
% of ideal loss	294.57	97.48	221.02	3 342.82
i.loss/r.loss	0.25	0.51	0.31	0.03

Number of created packets 19 688

Number of packets in the system 2 000

Porog(i) = 8 , 16 , 32

TABLE 8

Simulation with updating of routing matrices

Creation rate 16.6 packets per time unit

	Overall	P1	P2	P3
Delay	25.27	3.33	4.65	68.14
Confidence interval certainty 95 %		3.31 3.34	4.61 4.68	66.38 69.91
Largest queue	233	2	0	231
Income	17 359.22	22 716.51	18 526.85	
Ideal income	65 759.32	29 366.06	23 685.4	
Loss	64 232.48	17 301.49	9 207.15	37 723.84
Ideal loss	15 832.38	10 651.94	4 048.6	1 131.84
Diff.of losses	48 400.1	6 649.55	5 158.55	36 592.0
% of ideal loss	305.70	62.43	127.42	3 232.97
i.loss/r.loss	0.25	0.62	0.44	0.03

Number of created packets 120 000

Number of packets in the system 820

Porog(i) = 8 , 16 , 32

TABLE 9

Simulation with updating and rest of the queues

Creation rate 16.6 packets per time unit

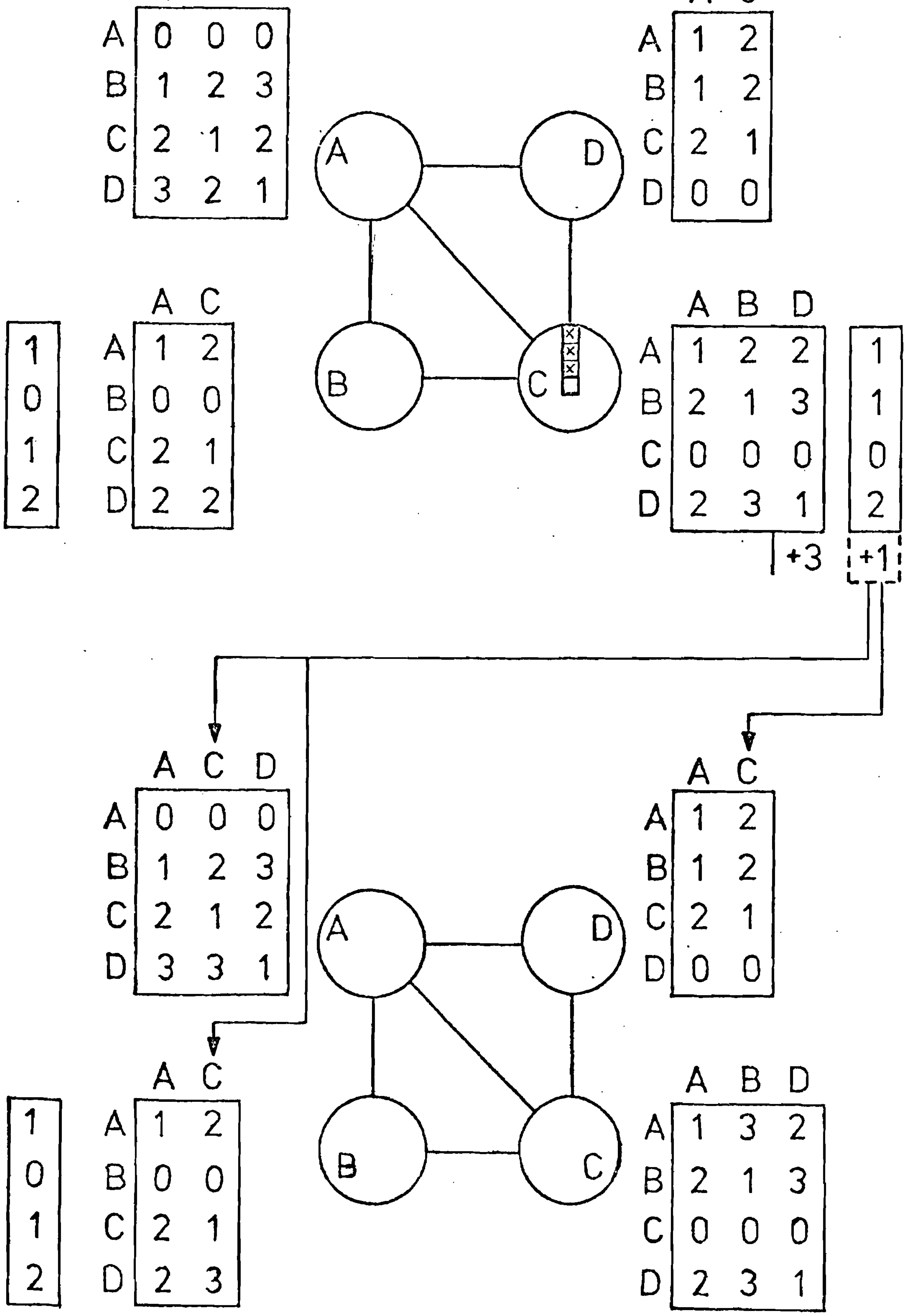
	Overall	P1	P2	P3
Delay	20.81	3.47	4.57	54.41
Confidence interval certainty 95 %		3.45 3.49	4.54 4.60	53.13 55.7
Largest queue	107	2	4	101
Income	24 261.31	21 977.49	18 677.35	
Ideal income	65 841.48	29 362.84	2 366.25	
Loss	57 420.89	18 036.51	9 057.35	30 327.04
Ideal loss	15 840.72	10 651.16	4 048.45	1 141.11
Diff.of losses	41 580.17	7 385.35	5 008.9	29 185.93
% of ideal loss	262.49	69.34	123.72	2 557.68
i.loss/r.loss	0.28	0.59	0.45	0.04

Number of created packets 120 000

Number of packets in the system 555

Porog(i) = 8 , 16 , 32

TABLE 10



	B	C	D
A	0	0	0
B	1	2	3
C	2	1	2
D	3	2	1

	A	C
A	1	2
B	1	2
C	2	1
D	0	0

1
0
1
2

	A	C
A	1	2
B	0	0
C	2	1
D	2	2

	A	B	D
A	1	2	2
B	2	1	3
C	0	0	0
D	2	3	1

1
1
0
2

	A	C	D
A	0	0	0
B	1	2	3
C	2	1	2
D	3	3	1

	A	C
A	1	2
B	1	2
C	2	1
D	0	0

1
0
1
2

	A	C
A	1	2
B	0	0
C	2	1
D	2	3

	A	B	D
A	1	3	2
B	2	1	3
C	0	0	0
D	2	3	1

FIGURE 1

$y_1 = 1 - 0.13t$

$y_2 = 0.7 - 0.05t$

$y_3 = 0.35 - 0.014t$

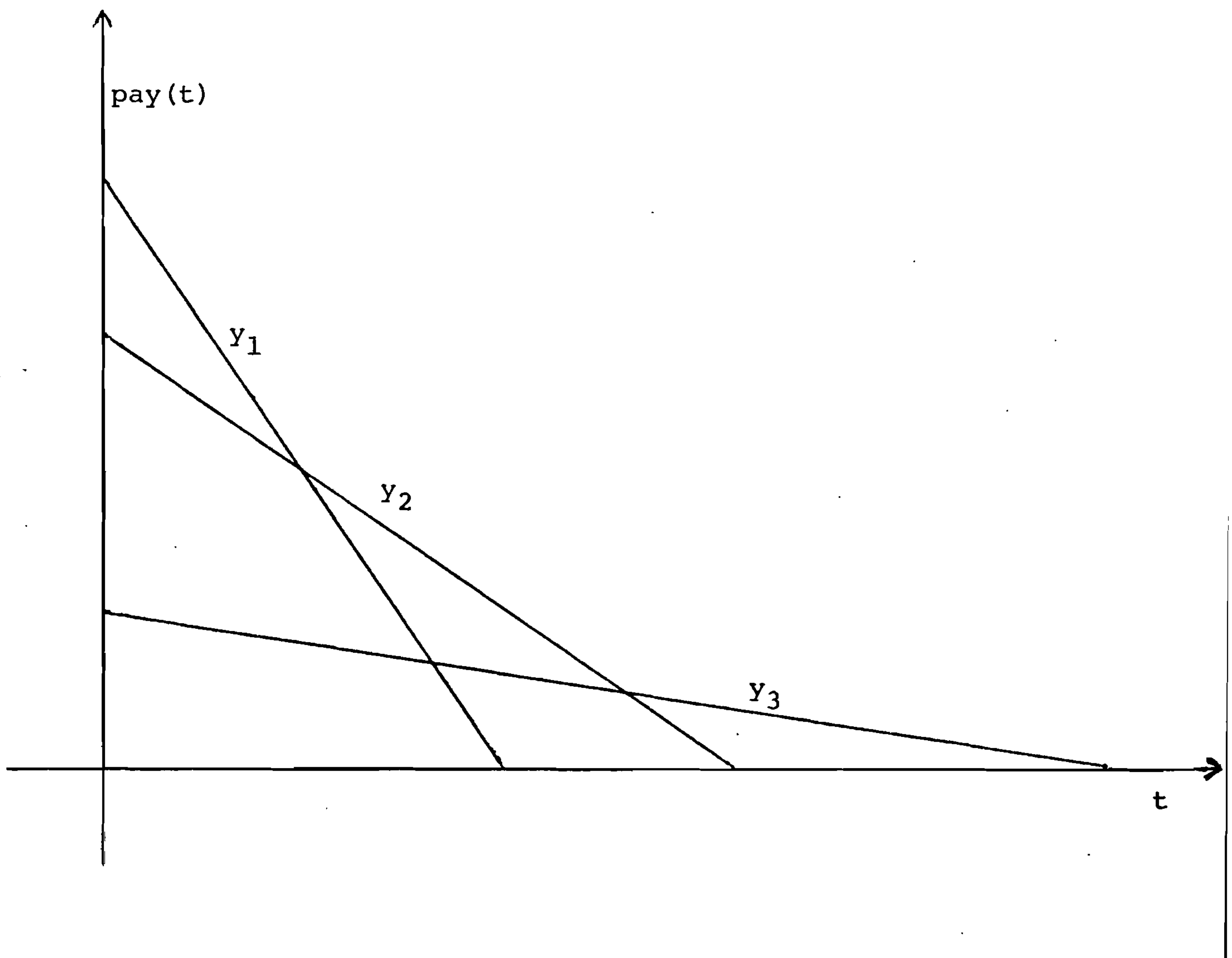


FIGURE 2

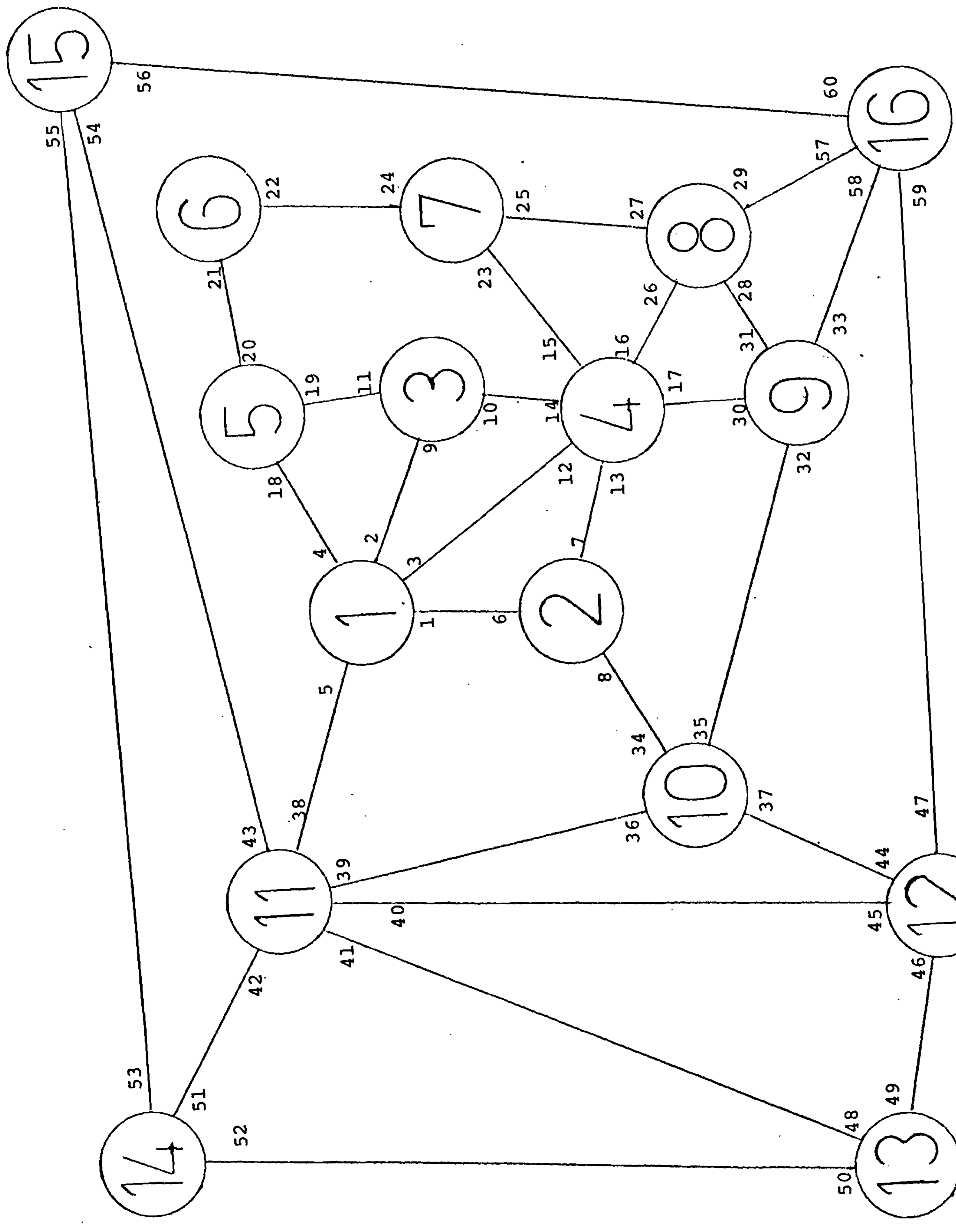

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60

FIGURE 3

References

[1] McQuillan, J.M. "Adaptive Routing Algorithms for Distributed Computer Networks," Bolt, Beranek and Newman, 1974.

[2] Frank, H. and Chuo, W. "Routing in Computer Networks," Networks, Vol. No. 2, 1971.

[3] Butrimenko, A. "Routing Technique for Message Switching Networks with Message Outdating," Symposium on Computer Communications Networks and Teletraffic, Brooklyn, 1972.