KYBERNETIKA — VOLUME 21 (1985), NUMBER 3

ADAPTIVE BIFURCATION ROUTING ALGORITHMS FOR COMPUTER-COMMUNICATION NETWORKS*

A. N. VENETSANOPOULOS, W. WAUNG

With the cost of computation constantly decreasing, packet switched computer-communication networks are becoming increasingly prevalent. To ensure robustness, mesh type topologies are often used. However, such network configurations require efficient and effective routing algorithms. In this paper, two adaptive bifurcation with threshold routing algorithms are considered. These algorithms use the usual update information available on standard computer-communication networks. Various traffic conditions are simulated to compare the proposed routing algorithms with others. The performance criteria used are average packet delay and link utilization at a specified load. Link failure effects are examined and observations are made about the algorithms' performance on actual networks.

1. INTRODUCTION

During the past two decades, the cost of communications has been steadily decreasing. However, the last few years have also witnessed a drop in the cost of computation, almost thirty times faster than that of communications [1]. This trend, expected to continue, has brought about a change in the direction of computer-communications networks (ARPA, TIDAS, Cyclades, Telenet, etc.). No longer are simple switching nodes the most cost effective and efficient methods of using network resources. Intelligent terminals, with the ability to make proper decisions on a more efficient distribution of network capacities, are now desirable.

A major means of interconnecting computers separated geographically is by the use of terrestrial links. Many different topological configurations can be used for such networks. When a compromise between reliability and cost is needed, mesh topologies with a connectivity of two usually result. These networks must have at least two link failures before a node, or group of nodes, is isolated from the rest of the system. However, since not every node is connected to all other nodes directly,

* The research presented in this paper was supported by a Natural Sciences and Engineering Research Council of Canada Research Grant No. G-0669.

messages have to be relayed to their destinations through intermediate nodes, thus requiring routing control.

Routing algorithms can be divided into three main categories: static, dynamic and quasistatic or adaptive routing schemes.

Static algorithms do not allow routing parameters to vary with time and are thus susceptible to changing traffic conditions. Dynamic algorithms, on the other hand, adjust routing parameters instantaneously, adjusting to network conditions. This flexibility is achieved at the expense of large and sometimes impractical information transfers. Adaptive algorithms adjust the routing parameters to accommodate small or slow changes in network conditions. To do so, some information about the network, but not the vast amount used in dynamic routing schemes, must be available. The ability to cope with changing network conditions, albeit slow changes, make these algorithms practical and useful in large systems [2–4].

The routing computations required to react to changing network conditions can be achieved either centrally or through distributed methods. With centralized computation, a network routing centre (NRC) receives global information about the system and routing decisions are made on these data. Distributed computation implies that routing calculations are performed at all nodes of the network. In most cases these calculations are performed using only local information available at the individual nodes. This local information may lead to some ambiguity about network conditions and thus preclude optimal behaviour. However, this may be tolerable if network conditions change slowly with respect to routing update intervals. The most important advantage of distributed computation is increased reliability. With decision making capabilities distributed, single node failures would not render the entire system inoperative. Centralized computation, conversely, has the undesirable characteristic that the network can become paralyzed in the event of the NRC failure. This reliability danger is contrary to presend trends for the development of robust communication networks.

In this paper, adaptive algorithms using the desirable characteristics of distributed computation are proposed. The next section briefly presents some well known adaptive routing algorithms. These algorithms are examined and their desired features are extracted and incorporated into the proposed routing schemes. The algorithms proposed are then described in detail and their performance is analyzed.

Sections 3 and 4 detail the simulations used to compare different algorithms, along with their results. The paper concludes with a discussion of the potential application of the two algorithms proposed to a number of actual networks.

2. ADAPTIVE ROUTING ALGORITHMS

2.1. The Original ARPANET Routing Algorithm [5]

The original ARPANET routing algorithm is an example of an adaptive algorithm which used simple updating methods. During updates, each node sends an update message to all its immediate neighbours, containing a vector indicating its expected delay to all nodes. Upon reception of these vectors from all its neigbours, a node processes this information by adding a bias value and the length of the queue to a particular neighbour. In addition, the expected delay to itself is set to zero. Then the neighbour with the lowest expected delay to a certain destination is chosen as the next node for all messages with that destination. At the time of the next update, the new expected delay vector is transmitted to all neighbours. Therefore, no information about the network need be collected between updates. In such an implementation, updates occur periodically three times every two seconds. During low traffic conditions, updates may occur more often.

This algorithm converges to the shortest path routing decision during low traffic conditions. In heavy and/or unbalanced traffic conditions, the algorithm attempts to seek alternate paths to avoid congested areas of the network. This algorithm, referred in the sequel as the ARPANET routing algorithm, considers only one output link for messages*.

2.2. The Bifurcation Routing Algorithm [6]

The bifurcation routing algorithm considers up to two output links for every message. At network initialization, the topology of the entire network is examined. First, the shortest delay path for each source-destination node pair is determined. Then, secondary paths are picked for some node pairs of the network. The secondary path is the shortest alternate path, less than two hops longer than the primary path. If no alternate path satisfies the criterion, no secondary path is chosen. During the operation of the network, a message is routed over the primary route if only one path is permitted. When two output paths are allowed for a message, a random choice is made. This random choice is achieved by using pseudo-random number generators and predetermined ratios. The ratios, set at the initialization of the network, control the proportion of the total traffic to be sent over the primary path. Simulation studies on an arbitrary ten node network showed that the bifurcation algorithm gives

* Recently the original ARPA algorithm was modified to alleviate looping and improve performance. Measured packet delays, averaged over some suitable interval, are used instead of a delay estimate computed by adding a constant for each hop to the instantaneous queue length at each node. These average packet delays are measured on each link in the network and the results are placed in updates, which flood the network. Each node then constructs a minimum delay path to each destination.

improved performance over single output link algorithms, during high unbalanced traffic conditions.

During low traffic conditions the bifurcation routing performs poorly, when compared to static shortest path routing. This behaviour is readily explained by the unneccessary routing of messages over longer delay paths.

2.3. The Delta Routing Algorithm [7]

Delta routing is an adaptive routing algorithm, which uses a combination of centralized and distributed computation. The centralized route calculation is done with less frequency than the distributed update. At the initialization of the network, up to two paths are considered for each source-destination node pair. These are recorded at a node acting as the network routing centre (NRC). During a centralized update, if two paths are allowed at a node, the expected delays on these two paths are examined. If the difference in the expected delays of the two paths is above a present threshold value, the lower delay path is chosen for all messages with that destination. This chosen path will then be the only allowed path, until the next centralized update decision. On the other hand, if the difference in the expected delays on the two paths is below the present threshold value, the path choice is left to the individual nodes. The node then chooses the lower expected delay path using only local queue length information. Unlike bifurcation algorithms, only one output path is allowed at all times.

This delta routing algorithm was compared with several others by simulation studies on small networks with different traffic conditions [7]. The results showed an improvement in the routing algorithm's ability to react to changing traffic situations. However, the results on a larger network showed only a minor improvement over standard routing algorithms. The centralized routing calculation required reduces the algorithm's reliability. Software or hardwate failures at the NRC can cause the system to become inoperative.

2.4. The Adaptive Bifurcation Routing Algorithm

A logical evolution combining the advantages of the previous routing algorithms is now proposed. The algorithm should use bifurcation techniques to spread traffic and use network resources more efficiently. However, it must not do so by splitting traffic during low network utilization. This condition can be met by the inclusion of a threshold level as in delta routing. Finally, the algorithm should incorporate an efficient computation method, without the need for a network routing centre. This last condition avoids the reliability problems characteristic of centralized computation.

The distributed computation method proposed can be viewed as an enhancement to the original ARPANET routing algorithm. The new adaptive routing using

bifurcation uses update information identical to that used in the original ARPANET. The justification for this choice is that numerous computer-communications networks were based on the approach taken in the original ARPANET routing algorithms, which has influenced routing research a great deal [1, 5]. In addition, the ideas described here can find application in other similar routing algorithms. For a bifurcation algorithm, the two best paths need to be chosen and the traffic to be divided between them. The secondary path should be the path with the lowest delay, when the primary path is excluded and can be chosen using the original update information, with little increase in complexity.

To avoid bifurcation of traffic at low utilization, a threshold involving the difference in expected delays of the primary and secondary path is practical. The ARPANET update information contains data about the number of hops to the destination and queue lengths along a path. During low traffic conditions, queue build-up is minimal, therefore the updating information contains numbers that are multiples of the bias value of 4. A difference between expected delays of 4 thus indicates a path length difference of a single hop.

A choice of a threshold of 3 is surmised as a good compromise providing early bifurcation, but reducing the possibility of unnecessary splitting of traffic. However, threshold values ranging from 2 to 7 are also examined here, to establish the validity of this hypothesis. In the algorithm proposed, if the difference in the expected delay between primary and secondary paths is less than the threshold value, then the traffic is bifurcated. Otherwise, the traffic is sent on the primary path only.

The algorithm splits all traffic allowed to be bifurcated evenly. Other methods of splitting are of course possible. Such an even split is considered for two reasons:

1) with the inclusion of the threshold, traffic is allowed to be sent on a secondary path only when the difference in expected delays is small, and
2) with an even bifurcation of traffic between two output links, simple deterministic techniques of splitting can be used.

In addition to the conservation of processing capacity and processing time, the alternate queue splitting technique produces a stream of traffic whose interarrival times have a lower variance than that of traffic subject to random splitting. A lower variance in interarrival times of customers will improve the performance of a queueing system [8], thus also improving the performance of the network.

Another deterministic switching method considered here is the one where a message joins the shortest of the two allowed queues. Since this 'join the shortest queue' discipline requires some current knowledge of the conditions at the node, it is expected to result in a slightly better performance compared to the simple alternate switching system.

In the following sections we shall examine these algorithms' capabilities to handle a variety of topologies and traffic conditions. The performance criteria used are average packet delay and link utilization at a specified load. The evaluation is per-

formed by conducting simulations of different networks and traffic conditions. The algorithms compared include:

1) A static routing scheme, designed as a shortest path algorithm. This algorithm gives excellent performance at low or balanced traffic conditions.
2) The ARPANET routing algorithm. On this simulation, simple loops are avoided by the 'hold-down' technique discussed in [9].
3) The bifurcation routing algorithm, using ARPANET update information, but without any threshold.
4) A bifurcation with threshold algorithm using a 'join the shortest queue' splitting discipline (Bif. w/thres. (s.q.)), and
5) A bifurcation with threshold algorithm using alternate queue switching (Bif. w/thres. (alt.)).

The alternate splitting bifurcation algorithm is also simulated with a range of threshold values. These are compared among themselves to determine the best threshold value.

Details on the network simulation models used are presented in the following section.

3. NETWORK SIMULATION MODELS

In the simulations performed, the delay encountered by each packet is recorded and its average is calculated. Related to the packet delay are two useful measures. These are the variance of the packet delays and the 95% delay level. Both of these measures are useful from the user's point of view and are provided for all simulations.

Another parameter examined is link utilization. This indicator, throughout the network, gives the proportion of the available bandwidth used for transmission of data. If some links are used heavily while others are idle, there is inefficiency and can appear as a large variance in link utilization.

Although the routing algorithms evaluated are designed for operation on large networks, the complexity of such networks makes a good understanding on the routing effects difficult. To circumvent this problem, small networks are first examined. Insight on the various algorithms is gained from their respective performances over these small networks. Finally simulation is performed on larger and more complex networks, to enhance our understanding of the behaviour of these algorithms in realistic situations. Different traffic conditions on all these networks are simulated to observe each algorithm's ability to react to changes in network conditions.

The arrival statistics used in the simulations were chosen after preliminary examination of the simulations of the static shortest path routing algorithm. The 'low traffic' condition produced link utilizations of less than 0·50 on all links. 'Moderate traffic' conditions produced at least one link with a utilization of greater than 0·50.

The small networks used in simulations are of four types. Schematically, these

networks are shown in Figure 1. The three four node networks are distinguished from each other by the different types of traffic carried. The four networks studied are similar to, but not identical, to the small networks used in [7]. In that work,

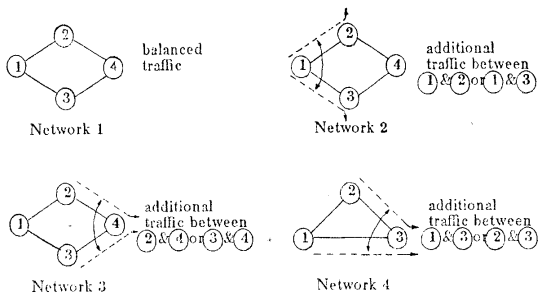

Fig 1. Four Small Simulated Networks.

the networks used simplex links only, while in the present paper, all links are full duplex. This change is made because simplex links limit the interrupting effect of the network resources. Full duplex links more accurately represent real networks and are thus used here.

After some insight, obtained from the performance of the various routing algorithms in the small networks, these algorithms are examined on larger networks. The first large network topology (network 5) is shown in Figure 2. This represents

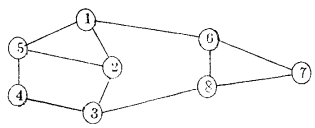

Fig. 2. Network 5 — Eight Node Balanced Topology.

an eight node balanced topology*. Three traffic conditions are simulated on this network. The first corresponds to a low balanced traffic. The second to an unbalanced traffic, where in addition to the balanced traffic of the previous simulation model, traffic between nodes 1 and 8 is introduced, which increases the traffic intensity between these two nodes to approximately 10 times that of any other node pair. This simulation thus examines the routing algorithms' ability to handle the resulting congestion. The remaining traffic condition simulated on this balanced large network is similar

* A balanced topology is one where all links are of the same length and capacity.

to the previous unbalanced traffic case except that the balanced background traffic is increased. This traffic condition is used to observe the routing algorithms' ability to handle increased load. The routing algorithms' robustness is examined by simulation on a large network. The main concern here is a routing scheme's ability to provide acceptable performance levels during link failures.

Network 6, used to investigate the effects of link failures, consists of 10 nodes. Its topology is shown in Figure 3. A large network is used here, because link failures

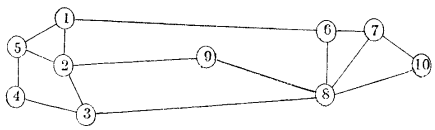

Fig. 3. Network 6 — Ten Node Unbalanced Topology.

in small networks often result in trivial topologies. An unbalanced topology is considered, because these topologies result in very uneven loading of network links. This uneven loading produces 'major links', heavily utilized, whose failures can cause major degradation of the network performance. The complement of these links are the minor links. This paper considers both major link and minor link failures. Therefore, the simulation of a major link failure is achieved by severing the connection between nodes 1 and 6. The traffic on the network is unbalanced. The minor link failure was simulated by disconnecting the link between nodes 3 and 4.

4. NETWORK SIMULATION RESULTS

In this section, the various simulation results are discussed and some are explained in both graphical and tabular form.

The first small network simulated is network 1 and the results are as expected. The bifurcation without threshold routing performs poorly, when compared to the other algorithms simulated. The remaining four algorithms show small differences in average packet delays. However, the bifurcation with threshold routing has a 33% smaller variance than that of static routing and a 16% small variance than that of the ARPANET routing algorithm. No significant difference is observed where the threshold is varied. Bifurcation without threshold is found to be the worst algorithm for this type of traffic conditions, because of the unnecessary splitting of the traffic.

Networks 2 and 3 were then simulated. Static routing in such fluctuating traffic conditions showed poor performance. Both the ARPANET routing and the bifurcation with threshold routing showed the ability to adapt and keep packet delays to a reasonable level. The bifurcation with and without threshold routing algorithms exhibit an improvement over the ARPANET routing algorithm. The average

packet delays for the complete simulation runs indicate that the bifurcation with threshold, using a threshold value of 3 has a reduced average packet delay compared to that of the ARPANET routing algorithm. Thresholds of 2 and 4 show minor

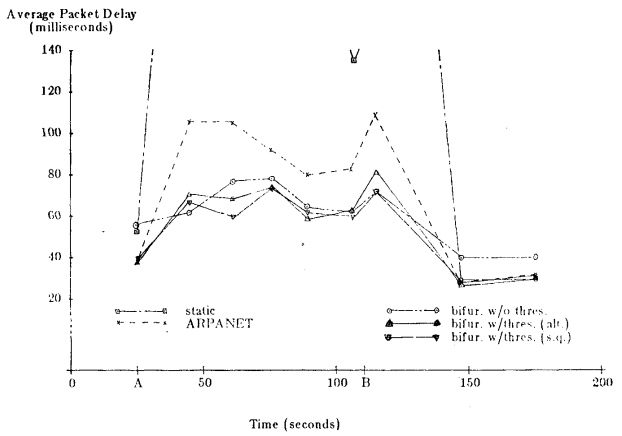

Fig. 4. Average Packet Delay Plot of Network 4.

Table 1. Network 4 simulation results and comparisons.

	Packet Delay			Link Utilization	
	Average (msec)	σ^2 (sec^2)	95% level ($\pm$10 msec)	Average	σ^2
Static	197·8	$9{\cdot}55\,(10^{-2})$	930	0·217	$3{\cdot}04\,(10^{-2})$
ARPANET alg.	72·4	$7{\cdot}24\,(10^{-3})$	270	0·219	$1{\cdot}35\,(10^{-2})$
Bif. w/o thres.	60·0	$2{\cdot}17\,(10^{-3})$	150	0·277	$8{\cdot}97\,(10^{-3})$
Bif. w/thres. $T = 2$ (alt.)	57·4	$3{\cdot}64\,(10^{-3})$	190	0·219	$7{\cdot}24\,(10^{-3})$
Bif. w/thres. $T = 3$ (alt.)	55·1	$3{\cdot}28\,(10^{-3})$	170	0·225	$1{\cdot}15\,(10^{-3})$
Bif. w/thres. $T = 4$ (alt.)	60·2	$2{\cdot}44\,(10^{-3})$	170	0·292	$6{\cdot}31\,(10^{-3})$
Bif. w/thres. $T = 5$ (alt.)	53·6	$2{\cdot}94\,(10^{-3})$	170	0·221	$7{\cdot}05\,(10^{-3})$

Comparison with ARPANET algorithm	Packet delay			
	Average		Variance	
	Δ%	Confidence	Δ%	Confidence
Bif. w/thres. $T = 3$ (alt.)	−24%	99·5%	−54·7%	97·5%
Bif. w/thres. $T = 3$ (s.q.)	−26%	99·5%	−59·3%	97·5%

Table 2. Network 5 (Balanced Traffic) simulation results and comparisons.

	Packet Delay			Link Utilization	
	Average (msec)	σ^2 (sec^2)	95% level ($\pm$10 msec)	Average	σ^2
Static	59·8	1·27 (10^{-3})	130	0·213	8·31 (10^{-5})
ARPANET alg.	59·4	1·21 (10^{-3})	130	0·213	6·61 (10^{-3})
Bif. w/o thres.	87·1	3·46 (10^{-3})	190	0·295	3·06 (10^{-3})
Bif. w/thres. $T = 2$ (alt.)	58·8	1·19 (10^{-3})	130	0·214	6·19 (10^{-3})
Bif. w/thres. $T = 3$ (alt.)	59·0	1·22 (10^{-3})	130	0·214	6·10 (10^{-3})
Bif. w/thres. $T = 4$ (alt.)	66·5	1·54 (10^{-3})	150	0·244	4·25 (10^{-3})
Bif. w/thres. $T = 5$ (alt.)	67·1	1·58 (10^{-3})	150	0·247	4·20 (10^{-3})
Bif. w/thres. $T = 6$ (alt.)	67·3	1·58 (10^{-3})	150	0·247	4·19 (10^{-3})
Bif. w/thres. $T = 7$ (alt.)	68·3	1·68 (10^{-3})	150	0·248	4·09 (10^{-3})
Bif. w/thres. $T = 3$ (s.q.)	58·9	1·16 (10^{-3})	130	0·214	6·57 (10^{-3})

Comparison with ARPANET algorithm	Packet delay			
	Average		Variance	
	Δ%	Confidence	Δ%	Confidence
Bif. w/thres. $T = 3$ (alt.)	−0·5%	99·0%	— ·–%	insignif.
Bif. w/thres. $T = 3$ (s.q.)	−0·8%	99·5%	−4·2%	90%

differences in performance. Similarly, bifurcation without threshold also showed only minor differences.

Network 4 is the three node network with unbalanced and switched traffic, shown in Figure 1d. The simulation results are shown in Figure 4. Here the additional traffic is introduced at time A and removed at time B. The static routing algorithm does not handle the additional traffic effectively. The ARPANET routing algorithm shows a reasonable performance, but is worse than any of the bifurcation routing algorithms. During unbalanced traffic conditions, all bifurcation routing algorithms show similar performance. When the additional traffic is removed, the bifurcation with threshold routing algorithms exhibited superior performance.

In all four cases, the bifurcation with threshold results in a smaller variance in packet delays than the ARPANET routing algorithm. These smaller variances, coupled with lower average values imply a lower maximum value. This is supported by the lower 95% delay level of the bifurcation with threshold ($T = 3$) algorithm in all cases.

The average link utilization of the static routing and bifurcation with threshold routing differs very little in the four cases examined. The bifurcation without threshold routing has average utilization values that are noticeably higher in all cases. This may indicate some inefficiency in the routing algorithm. Possibly some packets were sent along paths requiring more links unnecessarily, thereby raising the average link

Table 3. Network 5 simulation results and comparisons (unbalanced traffic, low background).

Overall Traffic	Packet Delay			Link Utilization	
	Average (msec)	σ^2 (sec^2)	95% level ($\pm$10 msec)	Average	σ^2
Static	79·1	4·76 (10^{-3})	210	0·245	2·47 (10^{-2})
ARPANET alg.	72·2	2·77 (10^{-3})	170	0·246	1·89 (10^{-2})
Bif. w/o thres.	99·5	4·87 (10^{-3})	230	0·339	1·02 (10^{-2})
Bif. w/thres. T = 2 (alt.)	70·3	2·43 (10^{-3})	170	0·247	1·78 (10^{-2})
Bif. w/thres. T = 3 (alt.)	69·6	2·25 (10^{-3})	170	0·255	1·61 (10^{-2})
Bif. w/thres. T = 4 (alt.)	76·0	2·29 (10^{-3})	170	0·288	1·15 (10^{-2})
Bif. w/thres. T = 5 (alt.)	76·3	2·20 (10^{-3})	170	0·293	1·19 (10^{-2})
Bif. w/thres. T = 6 (alt.)	76·3	2·25 (10^{-3})	170	0·294	1·20 (10^{-2})
Bif. w/thres. T = 7 (alt.)	77·5	2·25 (10^{-3})	170	0·299	1·22 (10^{-2})
Bif. w/thres. T = 3 (s.q.)	69·3	2·20 (10^{-3})	150	0·248	1·75 (10^{-2})

Node 1 to Node 8 Traffic	Packet Delay		
	Average (msec)	σ^2 (sec^2)	95% level ($\pm$10 msec)
Static	149·0	1·04 (10^{-2})	350
ARPANET alg.	118·9	5·22 (10^{-3})	270
Bif. w/o thres.	126·0	5·09 (10^{-3})	270
Bif. w/thres. T = 2 (alt.)	111·1	4·07 (10^{-3})	250
Bif. w/thres. T = 3 (alt.)	105·7	3·46 (10^{-3})	230
Bif. w/thres. T = 4 (alt.)	105·7	2·63 (10^{-3})	210
Bif. w/thres. T = 5 (alt.)	102·1	2·14 (10^{-3})	190
Bif. w/thres. T = 6 (alt.)	100·8	1·94 (10^{-3})	190
Bif. w/thres. T = 7 (alt.)	102·5	2·12 (10^{-3})	190
Bif. w/thres. T = 3 (s.q.)	104·2	3·72 (10^{-3})	230

Comparison with ARPANET algorithm (Overall Traffic)	Packet delay			
	Average		Variance	
	Δ%	Confidence	Δ%	Confidence
Bif. w/thres. T = 3 (alt.)	−3·5%	99·95%	−18·7%	99·9%
Bif. w/thres. T = 3 (s.q.)	−4·6%	99·5%	−20·7%	99·5%

Comparison with ARPANET algorithm (Node 1 to Node 8 Traffic)	Packet delay			
	Average		Variance	
	Δ%	Confidence	Δ%	Confidence
Bif. w/thres. T = 3 (alt.)	−11·1%	99·5%	−33·7%	97·5%
Bif. w/thres. T = 3 (s.q.)	−12·3%	99·95%	−28·7%	99·5%

Table 4. Network 5 simulation results and comparisons (unbalanced traffic, high background).

Overall Traffic	Packet Delay			Link Utilization	
	Average (msec)	σ^2 (sec^2)	95% level ($\pm$10 msec)	Average	σ^2
Static	123·1	3·30 (10^{-2})	450	0·318	3·24 (10^{-2})
ARPANET alg.	84·3	5·04 (10^{-3})	210	0·321	2·28 (10^{-2})
Bif. w/o thres.	120·2	8·30 (10^{-3})	290	0·439	1·10 (10^{-2})
Bif. w/thres. $T = 2$ (alt.)	80·6	4·01 (10^{-3}	210	0·323	2·14 (10^{-2})
Bif. w/thres. $T = 3$ (alt.)	81·3	4·06 (10^{-3})	210	0·328	2·00 (10^{-2})
Bif. w/thres. $T = 4$ (alt.)	89·8	4·01 (10^{-3})	210	0·376	1·44 (10^{-2})
Bif. w/thres. $T = 5$ (alt.)	90·3	4·38 (10^{-3})	210	0·385	1·42 (10^{-2})
Bif. w/thres. $T = 6$ (alt.)	91·3	3·86 (10^{-3})	210	0·392	1·44 (10^{-2})
Bif. w/thres. $T = 7$ (alt).	94·1	4·27 (10^{-3})	210	0·404	1·43 (10^{-2})
Bif. w/thres. $T = 3$ (s.q.)	79·6	3·86 (10^{-3})	190	0·325	2·14 (10^{-2})

Node 1 to Node 8 Traffic	Packet Delay		
	Average (msec)	σ^2 (sec^2)	95% level ($\pm$10 msec)
Static	351·5	9·18 (10^{-2})	>1000
ARPANET alg.	156·2	1·10 (10^{-2})	370
Bif. w/o thres.	166·3	1·01 (10^{-2})	370
Bif. w/thres. $T = 2$ (alt.)	138·4	7·27 (10^{-3})	290
Bif. w/thres. $T = 3$ (alt.)	137·2	7·00 (10^{-3})	290
Bif. w/thres. $T = 4$ (alt.)	135·5	5·38 (10^{-3})	290
Bif. w/thres. $T = 5$ (alt.)	131·7	4·72 (10^{-3})	270
Bif. w/thres. $T = 6$ (alt.)	130·7	4·51 (10^{-3})	270
Bif. w/thres. $T = 7$ (alt.)	132·5	5·45 (10^{-3})	270
Bif. w/thres. $T = 3$ (s.q.)	131·8	7·29 (10^{-3})	310

Comparison with ARPANET algorithm (Overall Traffic)	Packet delay			
	Average		Variance	
	Δ%	Confidence	Δ%	Confidence
Bif. w/thres. $T = 3$ (alt.)	−3·5%	99·5%	−19·5%	99·0%
Bif. w/thres. $T = 3$ (s.q.)	−5·5%	99·95%	−23·5%	99·5%

Comparison with ARPANET algorithm (Node 1 to Node 8 Traffic)	Packet delay			
	Average		Variance	
	Δ%	Confidence	Δ%	Confidence
Bif. w/thres. $T = 3$ (alt.)	−12·2%	99·5%	−36·1%	99·0%
Bif. w/thres. $T = 3$ (s.q.)	−15·6%	99·95%	−33·7%	99·0%

Table 5. Simulation Results with Major Link Failure.

Overal Traffic	Packet Delay				
	Average		%	Variance (sec^2)	95% level ($\pm$10 msec)
	Normal (msec)	Link Fail. (msec)			
ARPANET alg.	114·9	209·9	82·7%	2·57 (10^{-2})	490
Bif. w/thres. $T = 3$ (alt.)	103·9	173·9	67·4%	1·63 (10^{-2})	390
Δ%	+10·6%	+20·7%	—	+57·9%	—
Confidence	99·95%	99·95%	—	99·95%	—

Node 5 to Node 10 Traffic	Packet Delay				
	Average		%	Variance (sec^2)	95% level ($\pm$10 msec)
	Normal (msec)	Link Fail. (msec)			
ARPANET alg.	219·0	349·7	59·7%	2·90 (10^{-2})	590
Bif. w/thres. $T = 3$ (alt.)	187·1	288·7	54·3%	1·68 (10^{-2})	490
Δ%	+17·1%	+21·1%	—	+72·2%	—
Confidence	99·95%	99·95%	—	99·95%	—

Table 6. Simulation Results with Minor Link Failure.

Overall Traffic	Packet Delay				
	Average		%	Variance (sec^2)	95% level ($\pm$10 msec)
	Normal (msec)	Link Fail. (msec)			
ARPANET alg.	163·5	193·9	18·6%	2·88 (10^{-2})	530
Bif. w/thres. $T = 3$ (alt.)	147·5	164·2	11·3%	1·89 (10^{-2})	430
Δ%	+10·9%	+18·1%	—	+52·3%	—
Confidence	99·95%	99·95%	—	99·95%	—

utilization. The variance of link utilizations cannot be compared with confidence, due to the small number of links on these networks.

Network 5, with low balanced traffic, was then simulated. From Table 5, it can be seen that the bifurcation without threshold algorithms give poorer performance than all the other algorithms. Among the remaining routing algorithms shown in the figure, differences in average packet delays are insignificant. Also shown, the bifurcation with threshold values greater than 3 result in significantly larger average packet delays. These results confirm the findings of the simulations on network 1. On this larger network, the bifurcation without threshold routing algorithm performs poorly. This is attributed to unnecessary transmission of traffic over long delay paths. Unlike small networks, secondary paths in this large network may be much longer than the

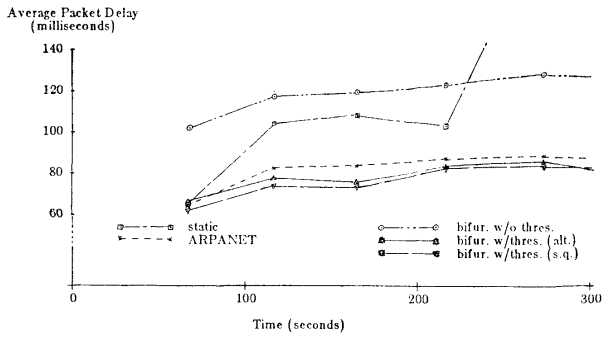

Fig. 5. Overall Traffic Average Packet Delay on Network 5. (Unbalanced Traffic, Low Background)

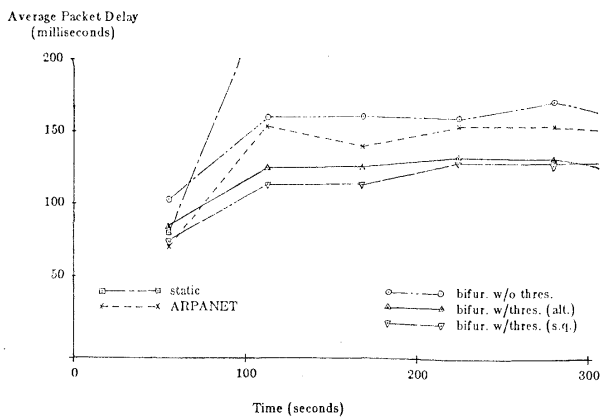

Fig. 6. Node 1 to Node 8 Average Packet Delay on Network 5. (Unbalanced Traffic, Low Background)

primary path, thus resulting in a greater discrepancy in the performance of bifurcation without threshold and other routing disciplines. This also explains the poorer performance of larger threshold valued bifurcation algorithms.

The same balanced network 5 was also simulated with low balanced traffic plus an additional load from nodes 1 to 8. Figure 11 shows the average packet delay of the entire network and shows bifurcation without threshold routing performing

poorly under these conditions. High threshold values produced approximately 10% higher average packet delays, when compared to the algorithm with a threshold of 3. Static routing accommodates the traffic loads with reasonable efficiency but is inferior to the ARPANET routing algorithm. The two bifurcation with threshold ($T = 3$) routing algorithms perform equally well and are slightly more efficient than the ARPANET routing algorithm. Table 3 indicates the same ordering of per-

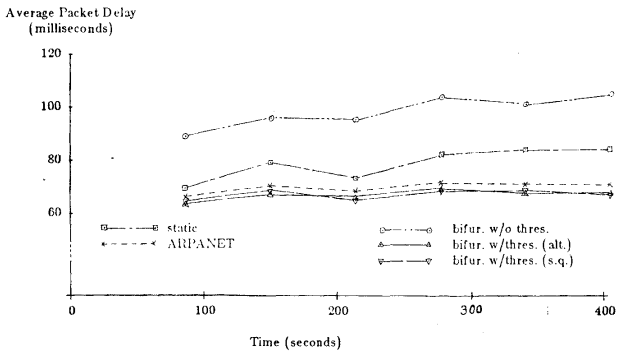

Fig. 7. Overall Traffic Average Packet Delay on Network 5. (Unbalanced Traffic, High Background)

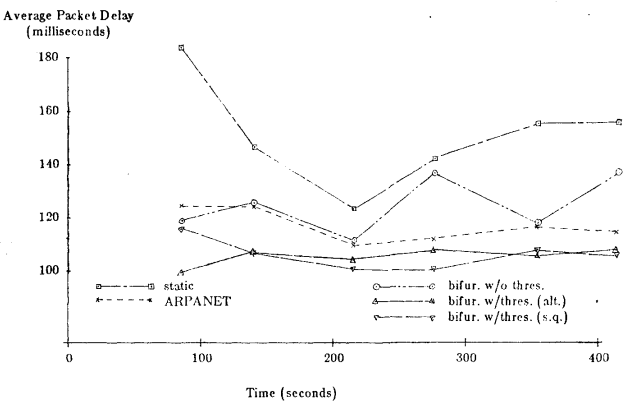

Fig. 8. Node 1 to Node 8 Average Packet Delay on Network 5. (Unbalanced Traffic, High Background)

formance in terms of average packet delay. The variance of packet delays further establishes that the bifurcation with threshold $(T = 3)$ routing results in a more compact distribution of packet delays. The bifurcation with threshold algorithms also exhibit a more even loading of network links by presenting smaller variances in link utilization.

These results indicate the advantages of the bifurcation with threshold algorithms.

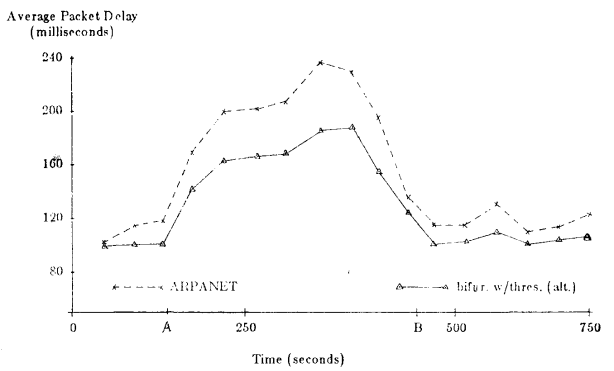

Fig. 9. Overall Traffic Average Packet Delay with Major Link Failure. (From A to B)

The two algorithms not only provide lower average delays, but also lower variance of the packet delays. The bifurcation without threshold algorithm does not handle traffic for the entire network effectively. This demonstrates the inefficiency of bifurcation in controlling low balanced traffic conditions. Figure 5 shows the node 1 to node 8 average packet delay for the same network.

In the next simulation, the intensity of the balanced traffic was raised, while the traffic from node 1 to node 8 remained the same. The bifurcation without threshold algorithm results in average packet delays, which were higher than those of the remaining algorithms. The three adaptive routing algorithms provide similar average packet delays throughout the simulation. However, the variance of packet delays using the bifurcation with threshold $(T = 3)$ algorithms is approximately 20% less than that with the ARPANET routing algorithm. The link utilization average and variance also indicates that the bifurcation with threshold routing provides more even and efficient loading than the other algorithms. Bifurcation with large threshold values however exhibits significantly worse performance than the ARPANET routing algorithm. This indicates the possible problems of early bifurcation, when considering all network traffic during unbalanced heavy usage. The higher balanced background traffic in these simulations does not affect the bifurcation with threshold

$(T = 3)$ algorithms' performance drastically. These two algorithms consistently provide routing with packet delays of low average value and low variance. The bifurcation without threshold algorithm shows its effectiveness in handling the heavily loaded portion of a network, while the static algorithm indicates a definite lack of such an ability.

Of the routing algorithms considered so far in this work, only two were chosen

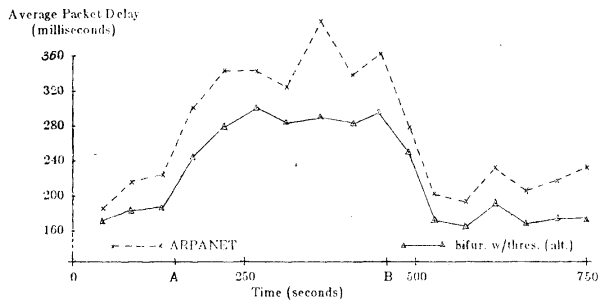

Fig. 10. Node 5 to Node 10 Average Paket Delay with Major Link Failure. (From A to B)

to examine their abilities to handle link failures. These were the ARPANET routing algorithm and the bifurcation with threshold $(T = 3)$ routing algorithm. The static routing algorithm was not considered because of its inability to react to any changes in network conditions. The bifurcation without threshold routing algorithm was omitted, because of its poor performance in the large network simulated. The bifurcation with threshold $(T = 3)$ routing algorithm using alternate splitting of traffic was chosen over the other bifurcation algorithms, because of its overall performance and reduced complexity. The ARPANET routing algorithm was examined to provide a comparison standard. The difference in the behaviour of the two routing algorithms is now summarized.

A plot of the transient behaviour of the two algorithms, when subjected to a link failure between nodes 1 and 6, is shown in Figure 9. The link is disconnected at point A and an increase in average packet delay is immediately evident. Note that the ARPANET routing algorithm results in a larger average delay at all times. The link between nodes 1 and 6 is reconnected at point B and the average packet delays using both algorithms quickly return to a lower value. Both algorithms appear to react to the link failure with similar swiftness. From the figures in Table 5, it can be seen that the ARPANET routing algorithm results in a higher average packet delay upon failure of the link. The bifurcation with threshold algorithm also provided lower variance of packet delay and a lower 95% delay level.

Traffic from node 5 to node 10 also experienced a similar pattern in average packet delay and variance of packet delay.

When considering a failure of the link between nodes 3 and 4, an increase in average packet delay is noted for both algorithms during the link failure but this increase is small. The variance and the 95% delay level are also higher using the ARPANET routing algorithm.

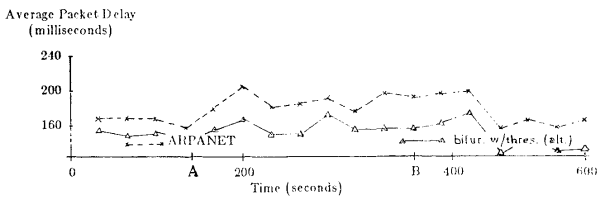

Fig. 11. Overall Traffic Average Packet Delay with Minor Link Failure. (From A to B)

When considering only the traffic from node 5 to node 10, a very similar trend appears. The results presented in Figure 12 as well as additional results, indicate that one significant difference is that the link failure causes the average packet delay to increase by only 2·4% when using the bifurcation algorithm. With the ARPANET

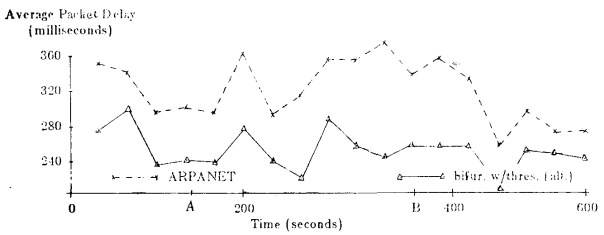

Fig. 12. Node 5 to Node 10 Average Packet Delay Plot with Minor Link Failure. (From A to B)

routing algorithm, the link failure increases the average packet delay by 13%. This shows the bifurcation algorithm's ability to effectively react to minor link failures. During normal network conditions, the ARPANET routing algorithm produces an average delay, that is 18% higher than that with the bifurcation routing technique. Under link failure conditions, this difference grows to 29%. The variance and 95% delay level are again lower with the bifurcation routing scheme.

5. CONCLUSIONS

In this paper, routing algorithms used on computer-communication networks were examined. Knowledge gained from the characteristics of other routing algorithms lead to the proposal of bifurcation with threshold algorithms.

From the simulation results, it is expected that bifurcation without threshold routing algorithms would perform poorly in large networks. Static routing algorithms are ineffective in handling heavy or unbalanced traffic conditions. The ARPANET routing algorithm performs very well under almost all traffic conditions, but is not as efficient as the bifurcation with threshold algorithms proposed. The best compromise for threshold appears to be a value around 3*. This choice provides good performance for the overall network traffic, without neglecting portions of the network with heavy traffic.

The link failure simulations demonstrate the bifurcation with threshold routing algorithms' ability to react to and handle topological changes. The algorithm examined appears to provide much smaller percentage degradation of service during failures, which makes it a very useful routing technique.

The 'join the shortest queue' discipline of traffic splitting exhibits a slight advantage over alternate queue splitting. This small improvement however, may not be worth the increased processing complexity required in sensing queue conditions in practical networks.

(Received October 5, 1983.)

* The choice of the threshold is naturally related to the bias of 4 used in the routing updates.

REFERENCES

[1] L. G. Roberts: The evolution of packet switching. Proceedings of the IEEE *66* (1978), 1307—1313.

[2] R. G. Gallager: A minimum delay routing algorithm using distributed computation. IEEE Trans. Comm. *COM-25* (1977), 73—85.

[3] A. Segall: The modeling of adaptive routing in data-communication networks. IEEE Trans. Comm. *COM-25* (1977), 85—95.

[4] T. E. Stern: A class of decentralized routing algorithms using relaxation. IEEE Trans. Comm. *COM-25* (1977), 1092—1102.

[5] J. M. McQuillan: Routing algorithms for computer networks. In: Proceedings of the 1977 National Telecommunications Conference, 1977.

[6] W. L. Price: Adaptive routing in store-and-forward networks and the importance of load spliting. In: Information Processing 77, IFIP, North Holland, Amsterdam 1977.

[7] H. Rudin: On routing and delta routing: a taxonomy and performance comparison of techniques for packet-switched networks. IEEE Trans. Comm. *COM-24* (1976), 43—59.

[8] L. Kleinrock: Queueing Systems, Volume II: Computer Applications. J. Wiley and Sons, New York 1976.

[9] J. M. McQuillan, G. Falk and I. Richer: A review of the development and performance of the ARPANET routing algorithm. IEEE Trans. Comm. *COM-26* (1978), 1802—1811.

Prof. Anastasios N. Venetsanopoulos, Department of Electrical Engineering, University of Toronto, Toronto M5S 1A4. Canada.

Mr. W. Waung, Microtel Pacific Research Limited, Burnaby. Canada.