South African Computer Journal
Number 6
March 1992

Suid-Afrikaanse Rekenaar-tydskrif
Nommer 6
Maart 1992

Computer Science and Information Systems

Rekenaarwetenskap en Inligtingstelsels

The South African Computer Journal

An official publication of the South African Computer Society and the South African Institute of Computer Scientists

Die Suid-Afrikaanse Rekenaartydskrif

'n Amptelike publikasie van die Suid-Afrikaanse Rekenaarvereniging en die Suid-Afrikaanse Instituut vir Rekenaarwetenskaplikes

Editor
Professor Derrick G Kourie
Department of Computer Science
University of Pretoria
Hatfield 0083
Email: dkourie@dos-lan.cs.up.ac.za

Subeditor: Information Systems
Prof John Schochot
University of the Witwatersrand
Private Bag 3
WITS 2050
Email: 035ebrse@witsvma.wits.ac.za

Production Editor
Prof G de V Smit
Department of Computer Science
University of Cape Town
Rondebosch 7700
Email: gds@cs.uct.ac.za

Subscriptions

	Annual	Single copy
Southern Africa:	R45_00	R15_00
Elsewhere	$45_00	$15_00

to be sent to:
Computer Society of South Africa
Box 1714 Halfway House 1685

Computer Science Department, University of Pretoria, Pretoria 0002
Phone: (int)+(012)+420-2504 Fax: (int)+(012)+43-6454 email: dkourie@rkw-risc.cs.up.ac.za

Sigma Press

Guest Contribution

This guest contribution is a slightly edited report to the Foundation for Research and Development (FRD) drawn up by Ed Coffman. Ed was an FRD-sponsored guest at the 6th South African Computer Research Symposium. The report was not originally intended for general distribution. Rather, it was specifically compiled for the FRD and its staff. I am therefore grateful to both the FRD and the author for agreeing to its publication in SACJ. I believe that it contains several incisive observations that merit further thought and discussion amongst South African computer scientists. (Editor)

Impressions of Computer Science Research in South Africa

E. G. Coffman, Jr.
AT&T Bell Laboratories, USA

In commenting on the cross section of computer science research in South Africa, I will use the classification in the table of contents of the "Summary of Awards: Fiscal Year 1989," a document published recently by the US National Science Foundation. Of the 5 categories, I will treat Numeric and Symbolic Computation as inappropriate for the discussion below. In this category I noted no research in the computer science setting in South Africa. It is also common in the US and elsewhere to place this effort in other departments, e.g., departments of mathematics or applied mathematics.

Of the remaining categories I found South Africa to be strongest in software systems and engineering, to have a substantial investment in computer systems and architecture, and to be weakest in computer and computation theory.

The coverage in software systems and engineering (SSE) was broad, topical, and similar in scope to that in US universities. Technology transfer and the corresponding relations with industry seemed to be in place or developing along promising lines. I comment in passing that this was rather surprising to me. In the US the development of SSE within university departments has lagged behind almost all other disciplines of computer science. A primary problem has been the insatiable appetite of industry for all Ph.D. graduates in the SSE field.

The investment in parallel processing, computer networks, and distributed computing appears sound, although I expected to see a greater emphasis on mathematical foundations (see my remarks below), particularly in the parallel algorithms area. Given current resources, South African institutions are doing remarkably well in computer science research. But computer science is a fundamentally important course of study, beginning at an early age and extending through graduate Ph.D. research; I take this as sufficiently obvious that I need not dwell on justifications. With this in mind, and with the necessary resources in hand, South Africa should, in my opinion, expand and consolidate its computer science research effort, increase its visibility in the international arena, and correct the rather thin distribution of graduate research among universities.

I can see much of this proceeding along present lines, but I would strongly recommend a concerted development in computer and computation theory (CCT), education and research; this is mainstream computer science and forms the basis for virtually all other fields of study within computer science. It is by no means absent in South Africa curricula, but it appears to be under--represented in advanced studies and Ph.D. level research.

At the graduate level CCT is heavily mathematical. I understand that mathematical foundations are supplied by mathematics departments in certain cases. This is not ideal, but workable and it is justified by limited resources. However, it is important that mathematics departments not regard this as a mere service; faculty will have to make a major commitment to theoretical computer science, publishing in its leading journals (e.g. SIAM Journal of Computing, Journal of the ACM, Journal of Algorithms, Algorithmica, Journal of Computer and Systems Sciences, Theoretical Computer Science, etc.), and providing the supervision of theses sponsored by computer science departments and leading to degrees in computer science. I would also encourage active participation in the international computer science "theory" societies and their meetings; two highly prestigious examples of the latter are the annual Symposium on the Theory of Computing and the Foundations of Computer Science conference.

Returning to the thin distribution of computer science research, I would make the following point. If the current situation is only a stage of development - i.e., if further resources (both human and financial) can be counted on to bring at least a few of the departments to a critical mass - then little needs to be said beyond the earlier remarks. Critical mass is hard to define, but calls for adequate, expert coverage of mainstream computer science re- search. In view of the breadth of this research, 8-10 Ph.D. full- time-equivalent faculty would seem to be barely adequate; with the usual clumping of faculty in specific research areas, more would be expected. South Africa has a talent base such that there is little doubt that such departments would achieve a much wider international recognition.

On the other hand, if resources remain fixed at current or even slightly retrenched levels, then I would recommend consolidation to achieve the same goals on a smaller scale. Within a university this can often be done by establishing interdisciplinary, degree-granting laboratories or institutes of computer science, which bring together the computer science efforts located in various departments other than computer science, such as electrical engineering, industrial engineering, business/-management science, mathematics, and operations research. The idea is to enjoy the advantages (opportunity, synergy, awareness, etc.) to both students and faculty of reasonably large computer science programs. There are many examples of such intramural laboratories in North America and Europe.

This approach could also be considered among universities within a confined geographical area, admittedly with greater difficulty perhaps. The Institute of Discrete Mathematics and Computer Science connecting Princeton University, Rutgers University, AT&T Bell Laboratories, and Bell Communications Research is a possible model. Examples in South Africa might consist of universities and research institutions on the Reef or those in the Western Cape (just to mention those with which I'm a little familiar).

As a final comment, I should note that my impressions have been based on limited information which may not give a representative picture. I am sure that my reactions will be appropriately discounted where I have been off target.

Editor's Notes

Prof John Schochot has graciously accepted to be SACJ's subeditor for papers relating to Information Systems. Authors wishing to submit papers in this general area should please contact him directly. I look forward working with John, and to a significant increase in IS contributions in future.

The hand of the new production editor, Riël Smit, will be clearly evident in this issue. Those papers not prepared in camera-ready format by the authors themselves were prepared by him in TEX. He will be announcing revised guidelines for camera-ready format in a future issue. If you use TEX or one of its variations, Riël would be happy to provide you with a styles document to SACJ format.

At last some Department of National Education committee has decided that SACJ should now be on the list of approved journals. This places it amongst the ranks of some 6800 other journals. These include not merely a number of ACM and IEEE Transactions but also such journals as *Ostrich*, *Trivium*, *Crane Bag*, *Koers*, *Mosquito News*, *Police Chief*, *Connoisseur*, *Lion and the Unicorn*, *About the House* and *Ohio Agricultural Research and Development Center Department Series ESS*. You will recall that in 1990 this same committee decided that, if judged on its own merits, SACJ did not deserve to be on the illustrious list. In the absence of other evidence, we must assume that the sole reason for its revised decision is that SACJ's predecessor, Quæstiones Informaticæ, was there. (I have a secret suspicion that the committee liked that name.)

It is my understanding that for official purposes, all journals on this list are regarded as *equally* meritorious, and all of them are more meritorious than *any* conference proceedings. What does all of this mean?

The momentous implication of the committee's deliberations is that the State will not give your institution a single cent for anything that you publish in SACJ. Instead, the State and your institution will scrupulously keep a score of the annual number of publications that count - but actually don't - because someday they might! And to encourage your enthusiastic participation in this Alice in Wonderland exercise, your institution might actually give you some of the standard subsidy funding that the State should have provided according to its own formulae, but didn't.

You will not be allowed to use this money to buy yourself a car - not even a casual meal. You may only use it to finance activities that are provably directed towards producing more papers in approved journals. The great consolation, of course, is that you will not be required to pay income tax on this money. The only tax involved will be the VAT component when you spend it in an approved manner. As a good computer scientist who enjoys recursion, my vote would be that all such revenue collected by the State should be earmarked to be placed in the pay packets of committee members who decided that SACJ should be approved.

If you publish in these approved journals with sufficient regularity and enthusiasm you will almost deserve to be regarded as a researcher. What you additionally need to do, is to ensure that you befriend and impress at least three overseas referees. You then apply to the FRD for official recognition as a researcher, and if they are sufficiently impressed, they will give you more of the non-taxable kind of money that you need to spend on research to publish in approved journals.

Derrick Kourie
Editor

Analysing Routing Strategies in Sporadic Networks

S W Melville

Department of Computer Science, University of Natal, Durban, 4001

Abstract

Communications networks with sporadically available links pose particular problems in terms of delay analysis. This paper presents an approach to determining the effectiveness of various routing strategies when applied to such networks, the strategies considered being minimum-hop routing, shortest-path routing, and flood routing. The implementation of a simulator package for sporadic networks is discussed, with particular attention to a branch-and-bound algorithm devised for the efficicient simulation of flood routing. Meteor-Burst Communications networks are used as primary examples of sporadic networks, and some preliminary results are presented.

Keywords: Sporadic networks, routing, meteor-burst communications.
Computing Review Categories: 3.24, 3.81, 8.1.

1 Introduction

In most forms of communication networks links between stations, once established, are constantly available. Thus the delay involved in sending a message from one station to another is essentially invariant if factors such as contention, buffer overflow and so forth are excluded from consideration.

Some networks, however, have links which are only sporadically available. Henceforth we shall describe such networks as 'sporadic networks'. A typical example is Meteor-Burst Communications, where signals between stations are scattered off the trails of ionisation created by meteors entering the atmosphere at high speed. While billions of meteor trails are formed every day, a trail must occur in the region of sky illuminated by both the transmit station and the receive station for the link between them to be viable. Different links will have different mean delays between usable meteor trails, due to such factors as transmitter and receiver power, galactic noise, size of common area of sky, and so on.

Modelling throughput over such sporadic links is not problematic (see [4],[5]) as the channel can merely be perceived as a low data-rate one rather than a collection of 'bursts' at high data rates. This is not valid in terms of delay analysis however (see [8]), as modelling the high data-rate sporadic channel as a low data- rate continuous one would result in gross underestimation of average delay for 'short' messages.

Some useful work has been done in analysing routing strategies for fast-changing networks (see [1]), and there has been a belief that this work might be applicable to sporadic networks ([6]). However there is a fundamental difference between these two types of networks. In the fast-changing network, typically one with mobile stations, the actual stations and established links change over time. In the sporadic network stations and the links between them are constant, but the links are only viable at certain times.

2 Routing Strategies Considered.

There is of course a plethora of potential routing strategies. In this work three of the more basic approaches are considered. These are minimum-hop routing, which seeks to route messages along the path containing the least number of intermediate stations; shortest-path routing, where messages are routed along the path estimated to have the least total delay; and flood routing, where the originating and all intermediate stations forward copies of messages to each of their neighbours which are not on the path the current copy originated from. In flood routing a record is kept of what messages have been forwarded, so duplicate copies arriving later may be ignored.

If there was no variance associated with delay in a sporadic network, and no possibility of channel noise causing more than one attempt to be needed for each send, the time taken for a message using shortest-path routing would be identical to that taken using flooding (barring considerations like contention, buffer storage, etcetera), as the best flood path would of course be the shortest path.

In sporadic networks which do have a non-zero variance of delay between usable links, this is not always the case. While the path with the shortest **mean** delay can be established, there is no guarantee that this will be the shortest path **for a particular send**. Thus it is quite possible for the flood routing to find a path for a particular message which is 'shorter' (less delay) than the path of least mean delay.

3 Simplifying Assumptions.

A number of simplifying assumptions are made in order to constrain the problem of comparing different routing strategies. Firstly, it is assumed that there will be no contention within the network. While this may seem an extreme assumption in terms of conventional networks, it is somewhat more tenable for those sporadic networks in which the availability of any given link is independent of the availability of all others (bar the special case of reciprocal links). Certainly collisions will be infrequent is such networks, however there could still be some queuing contention. The circumstances in which this contention would arise, as well as its effects , shall be discussed in detail later in the paper.

Further, it is assumed that all stations have unlimited buffer storage, so no overflow problems can occur, that there is a constant transmission error rate across all links in the network, with a simple ARQ feedback scheme being employed, and that traffic flow between stations is uniform (ie. the network does not have any overly 'loquacious' or 'reticent' stations).

In addition, it is assumed that all messages are small enough to be sent within the time that a sporadic link is enabled (valid for a command network) and are not subdivided into packets.

Finally it is assumed that all delays (eg. propagation delays, switching delays) are insignificant in comparison to the delay experienced waiting for links to become 'enabled'.

4 The SEER System.

A software package, System for the Evaluation of Efficieny of Routings (SEER) was designed and implemented in Pascal, using a clone of the IBM 286 AT. It accepts as input a file describing a network in terms of stations and links, with each link having two numbers associated with it. These numbers define the mean delay of link availability, as well as the standard deviation of this delay.

The system is presented with some number of message sends to be simulated, and randomly assigns sources and destinations for these messages. It then proceeds to simulate both the send of each message and the return of an acknowledgement of its receipt, by each of the three routing strategies. Statistics are kept so that the overall mean delay and delay standard deviation for each strategy can be determined.

5 Sending a Message

Computation of the time needed to send a message and to receive an acknowledgement is achieved by a simple summing of the delays found for each link on each of the two routes being considered (source to destination and destination to source).

Determining the delay for an attempted send over a particular link is achieved by using the following formula :

$$d_l = \max [\, md_l + (r \,.\, \sqrt{3} \,.\, sd_l) , 0 \,]$$

where :

d_l is the delay for a send over link l
md_l is the mean delay associated with link l
sd_l is the standard deviation of delay associated with link l
and r is a random number in the range -1 to 1.

The decision to allow for a range of root three standard deviations on either side of the mean is based on this ensuring that the standard deviation of the generated delays will be the same as the standard deviation associated with the link. (The derivation of this result is given in Appendix A.) The max function is used in order to avoid negative delay time.

An actual send is determined to have occured if a randomly generated number in the range zero to one is greater than the error rate specified by the user. If the random number is less than or equal to this rate, the attempt fails. The time taken for the send across a link is then taken to be the sum of the delays for each attempted send plus the delay for the actual send.

6 Route Determination

Determining shortest-path and minimum-hop routes is simple enough using Dijkstra's algorithm ([2]) for shortest paths. (For the minimum-hop case edge weights of one are given to all links, for shortest-path the mean delay associated with each link serves as its edge weight).

Determining flood routing is a far more complex problem. Clearly determining delays along all possible routes in a network would be impractical for all but the smallest networks. (Consider the thousands of different routes between any given source and destination that would be encountered in a hundred node fully-connected network, for example.) However if the simulation is to be valid it is necessary to consider all routes which have the potential to yield the least delay in sending the message.

This problem was resolved by using what is essentially a branch- and-bound search as described by Lawler and Wood [7], and making use of what Ibaraki [3] refers to as 'pruning by dominance'.

The algorithm to send a message (or acknowledgement) from source to destination is as follows :

Set shortest completed path time to infinity
Create an incomplete path from source to destination consisting of just the source node. Set 'time taken to here' of this path to be zero
Make this path the first (and only) element in a queue of paths
While the path queue is not empty do
  Set current path to be the front path in the queue
  Remove the current path from the queue

For each neighbour of the last station on the current path do
If the neighbour is already on the current path, ignore it (cycle)
Else
Determine the delay over the link from last station to the neighbour
Set NewTime to the current path's 'time to here' plus this delay
If NewTime is greater than shortest completed path time then do nothing further (not a candidate for the quickest flood route)
Else
If the neighbour is the destination station then
Set shortest completed path time to NewTime
Else
If the neighbour is on no other path then
Make a duplicate of current path, and then add the neighbour to the path. Set the path's 'time to here' to NewTime, and place it on the queue
Else
Compare Newtime to the time taken to 'time to here' at the point at which the neighbour was reached on the other path.
If NewTime is greater then do nothing further (duplicate copy being received, so can ignor path), else delete the other path from the queue, and construct and add onto the queue the path created by adding the neighbour to current path (in this case the other path would have carried the duplicate rather than the first copy received, and so would not be pursued)
End (of 'for each neighbour' loop)
Sort queue by 'time to here' of paths, with least-cost paths at front of the queue
End (Of 'while queue not empty' loop)

This algorithm guarantees finding the shortest route of the flood, as will be explained in the following paragraphs. In addition, it prunes the search space in such a way as to allow computation time to meet reasonable constraints.

By considering all incomplete paths until they become longer (greater delay) than some completed path, or until some station on the path is found to have been reached earlier by another route (in which case the station would have ignored the 'duplicate copy' of the message and so the path would not have been continued on to the destination station), we ensure that no candidate for the shortest flood path is ignored.

At the same time, by sorting the queue of incomplete paths by time taken (path cost) to date, we ensure that the most promising alternates are explored first. This will tend to allow 'expensive' paths to be safely excluded as candidates early in the search. (These paths will be excluded when some other candidate yields a quicker route to a station on the path, or when the cost of the incomplete path exceeds that of the shortest complete path.)

The structures used to implement the queue and paths are pointer records, as shown below :

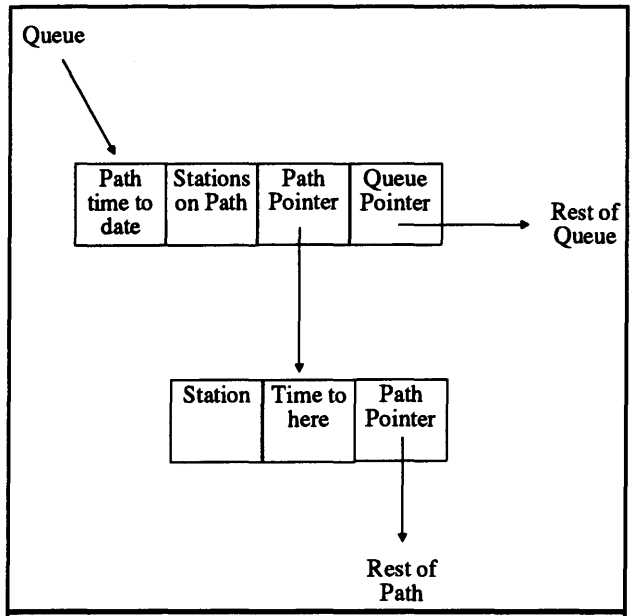

Figure 1 - Pointer Structure used For Flood Routing

7 Results

Results are given both for a 6-station network with varying mean delays and delay standard deviations, with delay time much as might be found in Meteor-Burst networks and for a fully-connected 50-station network with all links having identical delay characteristics.

The 'Meteor-model' network is shown below. The numbers on links indicate the mean delay and, in brackets, the standard deviation of delay, in seconds, for each link. Note that the network has been constructed to have reciprocal links, so that only one pair of numbers is given on each connection between each pair of stations, despite there being two links between them.

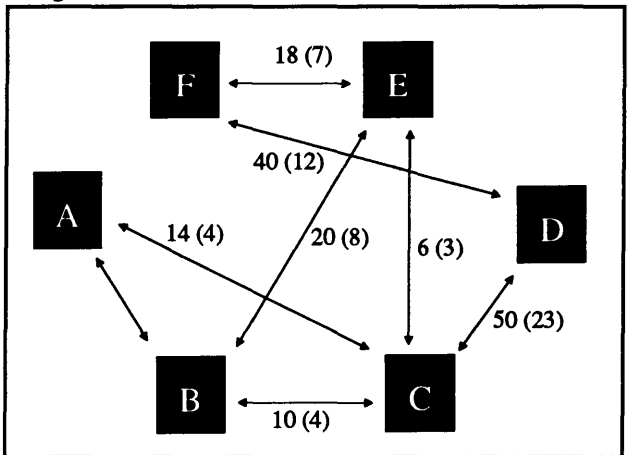

Figure 2 - A 'Meteor-Burst Model' Network

Results obtained by the SEER system simulating the transmission of one thousand messages on this network at various error rates are shown in Figures 3 and 4 below (the full numeric results appear in Appendix B) :

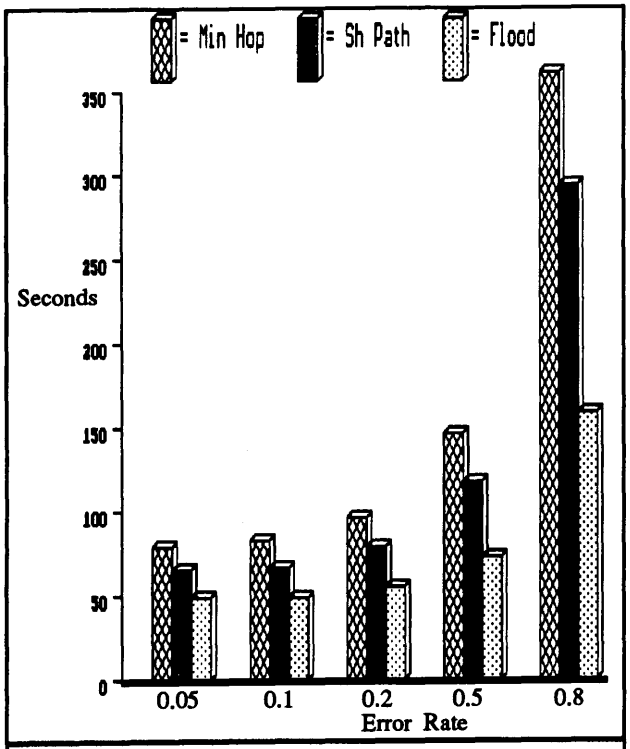

Figure 3 - Mean delays of routing strategies at various error rates

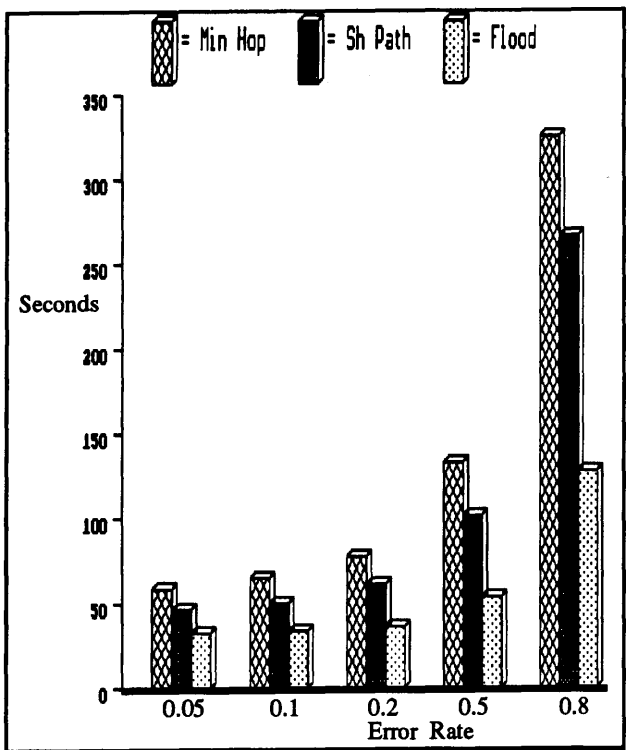

Figure 4 - Standard deviations of delay of routing strategies at various error rates

Considering the lower error rates first, it is obvious that shortest-path outperforms minimum-hop. (In this network not all least mean delay paths are also least intermediary paths, obviously if this were the case the two strategies would perform equally well.) Flooding is significantly better than shortest- path, having a mean delay of only about 75% that of shortest- path, and a standard deviation of delay of about 70% that of shortest path, for the 0.05 error rate. At this low error rate the gains achieved by flooding can probably be attributed to its finding shorter delay paths for particular sends than achieved by using paths with the shortest mean delays.

Now what is most interesting is how robust the flood routing is in the face of higher error rates. At the 0.2 error rate we see that flood now has a mean delay of around 69% of that of shortest-path routing, and a standard deviation of only about 60% of that of shortest-path routing. This relative gain increases as higher error rates are encountered, as can be seen in Figures 5 and 6 below.

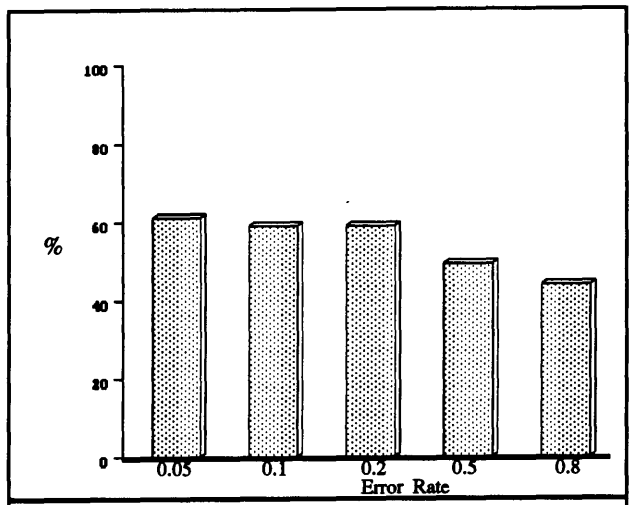

Figure 5 - Flood mean delay as a percentage of minimum-hop mean delay.

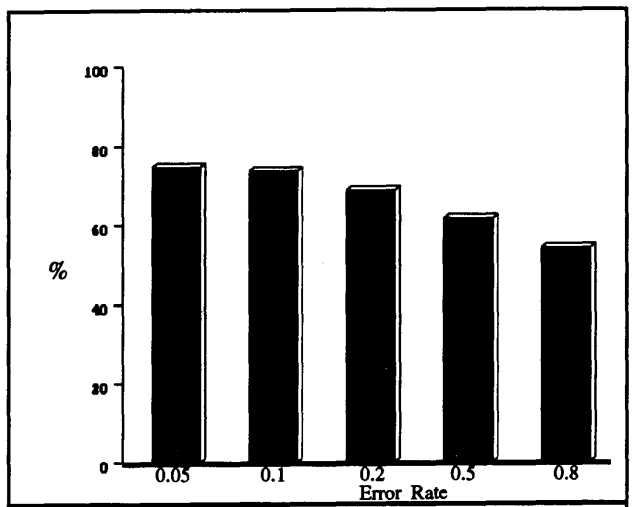

Figure 6 - Flood mean delay as a percentage of shortest-path mean delay.

This evidence that flood routing becomes relatively more attractive the noisier a channel gets can be attributed to the fact that in shortest-path or minimum-hop routing a failed attempt to send will generally result in considerable delay, while in flooding the failure will not have a great effect on delay if the link can be 'bypassed' by an alternate route taking the same time, or only a little more.

This 'bypass potential' also has considerable effect on the standard deviation of delay, as can be seen from the results. This is especially important to situations where 'worst-case behaviour' of a sporadic network is an important concern, such as would be the case in military networks.

The second network tested was a fully connected 50-station network, with each link being assigned a mean delay of 12 seconds and a delay standard deviation of 7 seconds. The simulation was run on 200 messages for a total of 400 messages and acknowledgements being sent.

Results are shown below :

Error rate = **0.05**

Routing strategy	Mean Delay	Delay Std Deviation
Minimum-Hop	24.95	11.63
Shortest-Path	25.61	10.63
Flooding	3.47	1.26

Shortest-path routing and minimum-hop routing are equivalent in this case, with both strategies utilising the direct links between sources and destinations (the minor difference in results being due to the random factors in the computation).

Flood routing gives remarkably better results than the other strategies on this network, with less than 14% of the mean delay and less than 12% of the standard deviation encountered with the other strategies.

The mean delay does seem exceptionally low at first glance, however on considering the topology of the network it becomes obvious that such a result should be expected.

The range in which generated delays can fall is from zero seconds to a fraction over 24 seconds. This means that we would only expect the delay on the direct link between source and destination to be 'low' (say two seconds or less) about $^1/_{12}$ of the time.

Two-link paths will have delays in the range zero seconds to 48 seconds, and would thus have delays of two seconds or less about $^1/_{24}$ of the time. However, there are 48 two-link paths in this fully-connected 50-node network, so we would expect two of these ($48 * {}^1/_{24}$) to have such delays.

Thus without even considering the other potential paths (3-link, 4-link, etcetera), it is clear that such low delays for flood routing in a highly-connected topology are not only possible but should in fact be expected.

8 A Note on Speed

Clearly the most time-consuming section of code in the SEER system is that dealing with the generate-and-test involved in the flooding simulation. The performance of the algorithm used shall be briefly discussed.

The algorithm has a best case scenario occurring when there is a direct link between source and sink, and this link has the least delay amongst all the links emanating from the source. Here the algorithm will detect the oprimal flood path after inspecting only the links connecting the source to its neighbours, which gives a gain over exhaustive search equal to E_s/E_n, where E_s is the out-degree of the source, and E_n is the number of edges on all source-sink paths.

The worst-case scenario would occur in the unlikely situation where all paths between source and sink are node-disjoint (no common intermediaries would prevent any pruning by dominance) **and** all path delays from source to the station immediately preceding the sink are greater than the shortest complete path from source to sink. This latter condition means that no partial path could be discarded until its entire delay was computed. In this worst-case scenario the search algorithm would give no gain whatsoever over exhaustive search.

Determining an average case would be a particularly difficult exercise, if indeed it is possible to do so. Clearly the topology of the network and the pattern of delays on links between source and sink (do 'bad' paths become distinctly bad early?) would have to be considered in any such determination. However the algorithm presented here has constrained the problem well for the networks tested - for 400 floods on the fully-connected 50-station network discussed the simulation ran to completion in under an hour, while the simulations on the smaller network were handled in seconds. This on a 286 PC/AT clone.

9 Flooding and Contention

Queing contention arises when a station has a number of messages to send, and the order in which they are sent will affect the delay associated with their transmission. Such contention can arise under any of the routing strategies - one needs only consider the hub of a star topology network to see this. However it is clear that the flood mechanism will be most prone to create such contention, and so we will consider only this case in determining at what point such contention would degrade network performance.

An important point as regards sporadic networks is that queing contention is only significant if the time period that the link is enabled between some transmitter T and receiver R is too short to allow **all** messages T has for R to be sent. (Given the earlier assumption that the delay between link availability is our only significant delay, it follows that there is no significant delay between a message transmitted immediately the link becomes viable, and one transmitted immediately before it is no longer viable.) In the worst case scenario of flooding, T would have a copy of every message

in the network waiting to be sent to R when the link becomes enabled, and so contention would arise if the link was not enabled for a sufficient time to allow all these messages to be sent.

From this, we can see that a number of factors affect whether contention arises in a sporadic network or not, (and how severely it becomes a problem). Firstly, the amount of data in the network as a whole is a concern. The network can avoid contention if it is loaded to any level up to any point where the throughput possible on a link is greater than or equal to the amount of data being transmitted in the network. As loading becomes heavier than this so contention could arise, and the greater the loading the higher the risk, and the more severe the effect, of queing contention.

Conversely, the greater the throughput potential of enabled links, (the amount of time they're enabled for, times the data-rate they can support), the higher the loading that can be achieved before contention arises.

Finally, of course, the topology of the network and the pattern of link availability will determine how severely loaded individual stations become. (ie. how bad our worst-case for a particular network can get.)

As an example, in the case of Meteor-Burst Communications, a usable trail would be roughly 250 ms, and could support a data rate of about 32 kbps. This means that an enabled link would be capable of sending around 8 kilobits of data. For application networks with an average message length of 40 bytes or so this would be about 25 messages. Thus the network could be at risk of queing contention if more than 25 messages were 'in transit' at any given time.

Determining the probability of such contention actually occurring when the network is loaded beyond this 'safe' level, and to what degree it would affect the efficiency of routing strategies, is a subject that needs further work.

10 Conclusion

The design of the SEER system, in particular in terms of its handling of the problem of sporadic link availability and the branch-and-bound algorithm used to simulate flooding, has allowed for effective modelling and fast simulation of routing in a sporadic network environment.

Flood routing has clearly yielded significantly better results for sporadic networks than the other two alternates considered, particularly in terms of both lower mean delay in the network as well as far lower standard deviation of delay. It becomes even more attractive in situations where transmission error rates are high.

Shortest-path routing will have significant benefits over minimum-hop routing in some networks, but will obviously be equivalent in networks where the minimum-hop route and the shortest-path route are the same.

There are of course other factors to be considered in contrasting the routing strategies - overheads such as the construction of routing tables, queing contention, problems such as buffer overflow and so on. However it does seem that the particular nature of sporadic networks will still make flood routing the most effective routing strategy for these networks.

Further work is required in two areas. The effect of queing contention on delays experienced, as has been discussed, requires more study. In addition, a study of the effects of network topologies on routing efficiency would be of great interest. It is clear from the results that a high degree of connectivity will greatly increase the relative efficiency of flood routing over the two other strategies considered. A generalised formulation of the decrease in delay as a function of the increase in connectivity would be of great use to network designers, as would be the investigation of routing performance over a wider range of basic network topologies. (Stars and linked-stars, hierarchical networks, and so on.)

References

[1] S Corson and A Ephermides, [1989], A Distributed Routing Algorithm for Mobile Radio Networks, *Conference Record of the 1989 IEEE Conference on Military Communications* (MILCOM89), Volume 1, 11.2.1 - 11.2.4.

[2] E W Djikstra, [1959], A Note on Two Problems in Connection with Graphs, *Numerische Mathematik I*, 269-271.

[3] T Ibaraki, [1977], The Power of Dominance Relations in Branch-and-Bound Search Methods, *JACM*, **24** (2), 264-279.

[4] J D Larsen, S W Melville and R S Mawrey, [1990], Adaptive Data Rate Capacity of Meteor-Burst Communications, *Conference Record of the 1990 IEEE Conference on Military Communications* (MILCOM90) at Monterey, California, Volume 2, 40.1.1 - 40.1.5.

[5] J D Larsen, S W Melville, R S Mawrey, R Y Letschert, and W D Goddard, [1990], Throughput Capacity of Meteor-Burst Communications, *Transactions of the SAIEE*, **81** (3), 20 - 30.

[6] J D Larsen, [1991], Meteor Communications Corporation, Seattle, Washington, U.S.A. Personal communication.

[7] E W Lawler and D E Wood, [1966], Branch-and-Bound Methods : A Survey, *Operations Research*, **14** (4), 669-719.

[8] S W Melville and J D Larsen, Wait Time in Meteor-Burst Communications, submitted for publication in *Transactions of the SAIEE.*

Acknowledgements: Sincere thanks to Salbu (Pty) Ltd, the FRD and the University of Natal for making funding available without which this work would not have been possible.

Appendix A - Statistical Derivation

Having a mean μ and a standard deviation σ of delay associated with each link, the simulator required that random numbers be generated over an interval in such a way as to ensure that the numbers generated had the same mean and standard deviation as were associated with the link. The derivation of this interval is described here.

Now a random distribution over an interval will have the same mean and standard deviation as a uniform distribution, given that we have an infinite number of samples. Clearly then the centre of the interval must be the mean associated with the link, and we seek to determine k such that an infinite number of random numbers generated in the range $(\mu - k\sigma, \mu + k\sigma)$ will have a mean equal to μ and a standard deviation of σ. As we have a uniform distribution, we may consider the equivalent range of $(0, 2k\sigma)$ and corresponding mean $k\sigma$.

We now use a discrete uniform distribution with $n + 1$ points to derive k, and then determine the result as n tends towards infinity. We shall work with variance rather than standard deviation for ease of computation.

The $n + 1$ points occurring uniformly across the range 0 to $2k\sigma$ will be of the form :

$$0, \frac{2k\sigma}{n}, 2.\frac{2k\sigma}{n}, 3.\frac{2k\sigma}{n}, \ldots, n.\frac{2k\sigma}{n}$$

Their mean, $k\sigma$, can be written as $\frac{n}{2}.\frac{2k\sigma}{n}$, giving us the following formula for their variance :

$$\frac{1}{n+1}\sum_{i=0}^{n}[\frac{2k\sigma}{n}(i-\frac{n}{2})]^2$$

$$= \frac{1}{n+1}\sum_{i=0}^{n}[\frac{4k^2\sigma^2}{n^2}(i^2 - ni + \frac{n^2}{4})]$$

$$= \frac{1}{n+1}[\frac{4k^2\sigma^2}{n^2}\frac{n^2}{4} + \sum_{i=1}^{n}(\frac{4k^2\sigma^2}{n^2}(i^2 - ni + \frac{n^2}{4}))]$$

$$= \frac{1}{n+1}[k^2\sigma^2 + \frac{4k^2\sigma^2}{n^2}[\frac{n(n+1)(2n+1)}{6} - n\frac{n(n+1)}{2} + \frac{n^3}{4}]]$$

$$= \frac{1}{n+1}[k^2\sigma^2 + \frac{4k^2\sigma^2}{n^2}[\frac{4n^3 + 6n^2 + 2n - (6n^3 + 6n^2) + 3n^3}{12}]]$$

$$= \frac{1}{n+1}[k^2\sigma^2 + \frac{4k^2\sigma^2}{n^2}(\frac{n^3 + 2n}{12})]$$

$$= \frac{1}{n+1}[k^2\sigma^2 + \frac{k^2\sigma^2(n^2+2)}{3n}]$$

$$= k^2\sigma^2[\frac{1}{n+1}(1 + \frac{n^2+2}{3n})]$$

Let this equation equal σ^2, then

$$k^2[\frac{1}{n+1}(1 + \frac{n^2+2}{3n})] = 1$$

$$\Rightarrow k^2[\frac{1}{n+1}(\frac{n^2+3n+2}{3n})] = 1$$

$$\Rightarrow k^2[\frac{1}{n+1}\frac{(n+1)(n+2)}{3n}] = 1$$

$$\Rightarrow \frac{k^2(n+2)}{3n} = 1$$

$$\Rightarrow k^2 = \frac{3n}{(n+2)} = 3\frac{n}{n+2}$$

Now $\lim_{n\to\infty} \frac{n}{n+2} = 1$

so $\lim_{n\to\infty} k^2 = 3$

$$\Rightarrow k = \sqrt{3}$$

Appendix B - Numerical Results

Error rate = **0.05**

Routing strategy	Mean Delay	Delay Std Deviation
Minimum-Hop	75.82	56.74
Shortest-Path	63.98	43.11
Flooding	50.30	34.56

Error rate = **0.1**

Routing strategy	Mean Delay	Delay Std Deviation
Minimum-Hop	76.11	57.12
Shortest-Path	64.49	49.38
Flooding	51.50	34.70

Error rate = **0.2**

Routing strategy	Mean Delay	Delay Std Deviation
Minimum-Hop	99.83	69.53
Shortest-Path	81.79	62.80
Flooding	59.22	34.85

Error rate = **0.5**

Routing strategy	Mean Delay	Delay Std Deviation
Minimum-Hop	134.05	121.94
Shortest-Path	125.78	112.09
Flooding	74.58	54.87

Error rate = **0.8**

Routing strategy	Mean Delay	Delay Std Deviation
Minimum-Hop	344.23	323.26
Shortest-Path	256.63	219.20
Flooding	144.72	124.99

Notes for Contributors

The prime purpose of the journal is to publish original research papers in the fields of Computer Science and Information Systems, as well as shorter technical research papers. However, non-refereed review and exploratory articles of interest to the journal's readers will be considered for publication under sections marked as a Communications or Viewpoints. While English is the preferred language of the journal papers in Afrikaans will also be accepted. Typed manuscripts for review should be submitted **in triplicate** to the editor.

Form of Manuscript

Manuscripts for review should be prepared according to the following guidelines.

- Use double-space typing on one side only of A4 paper, and provide wide margins.
- The first page should include:
  - title (as brief as possible);
  - author's initials and surname;
  - author's affiliation and address;
  - an abstract of less than 200 words;
  - an appropriate keyword list;
  - a list of relevant Computing Review Categories.
- Tables and figures should be on separate sheets of A4 paper, and should be numbered and titled. Figures should be submitted as original line drawings, and not photocopies.
- Mathematical and other symbols may be either handwritten or typed. Greek letters and unusual symbols should be identified in the margin, if they are not clear in the text.
- References should be listed at the end of the text **in alphabetic order** of the (first) author's surname, and should be cited in the text in square brackets. References should thus take the following form:

  [1] E Ashcroft and Z Manna, [1972], The translation of 'GOTO' programs to 'WHILE' programs, *Proceedings of IFIP Congress 71*, North-Holland, Amsterdam, 250-255.

  [2] C Bohm and G Jacopini, [1966], Flow diagrams, Turing machines and languages with only two formation rules, *Comm. ACM*, **9**, 366-371.

  [3] S Ginsburg, [1966], *Mathematical theory of context free languages*, McGraw Hill, New York.

Manuscripts *accepted* for publication should comply with the above guidelines, and may be provided in one of the following formats:

- in a **typed form** (i.e. suitable for scanning);
- as an **ASCII file** on diskette; or
- as a **WordPerfect**, **TEX** or **LATEX** or file; or
- in **camera-ready** format.

A page specification is available on request from the editor, for authors wishing to provide camera-ready copies. A styles file is available from the editor for Wordperfect, **TEX** or **LATEX** documents.

Charges

Charges per final page will be levied on papers accepted for publication. They will be scaled to reflect scanning, typesetting, reproduction and other costs. Currently, the minimum rate is R20-00 per final page for camera-ready contributions and the maximum is R100-00 per page for contributions in typed format.

These charges may be waived upon request of the author and at the discretion of the editor.

Proofs

Proofs of accepted papers will be sent to the author to ensure that typesetting is correct, and not for addition of new material or major amendments to the text. Corrected proofs should be returned to the production editor within three days.

Note that, in the case of camera-ready submissions, it is the author's responsibility to ensure that such submissions are error-free. However, the editor may recommend minor typesetting changes to be made before publication.

Letters and Communications

Letters to the editor are welcomed. They should be signed, and should be limited to about 500 words.

Announcements and communications of interest to the readership will be considered for publication in a separate section of the journal. Communications may also reflect minor research contributions. However, such communications will not be refereed and will not be deemed as fully-fledged publications for state subsidy purposes.

Book reviews

Contributions in this regard will be welcomed. Views and opinions expressed in such reviews should, however, be regarded as those of the reviewer alone.

Advertisement

Placement of advertisements at R1000-00 per full page per issue and R500-00 per half page per issue will be considered. These charges exclude specialized production costs which will be borne by the advertiser. Enquiries should be directed to the editor.

South African Computer Journal

Number 6, March 1992
ISSN 1015-7999

Suid-Afrikaanse Rekenaar-tydskrif

Nommer 6, Maart 1992
ISSN 1015-7999

Contents