May 1980

DESIGN CONSIDERATIONS OF A DISTRIBUTED PACKET RADIO NETWORK USING THE RADIO AMATEUR BAND

Iréne E Persson
Robert Forchheimer

INTERNAL REPORT
LiTH-ISY-I-0408

Abstract

The paper outlines an experimental radio based data network to be used by radio amateurs and computer hobbyists. Various aspects of network design is discussed with the intention to provide a foundation on which a more detailed specification can be based. Underlaying principles and qualitative rather than quantitative results are stressed.

Keywords: packet radio, distributed data network, personal computing

Table of contents

cont.

1. INTRODUCTION

This paper outlines a radio based data network. The network is considered to be used by radio amateurs and computer hobbyists and will reside on a suitable radio amateur band in the frequency spectrum. The paper discusses various aspects of network design with the intent of producing a sound foundation on which a more detailed specification can be based. Underlaying principles and qualitative - rather than quantitative - results will be stressed.

1.1 Radio Amateurs and Computer Hobbyists

Amateur radio has been a phenomena as old as radio itself. The radio amateurs have played a significant role in the development of radio technology mostly because they were pushed from commercially interesting long and medium wave frequency range into the unexplored and little understood short wave range. The vast possibilities of this frequency range was soon discovered by the amateurs, leading to regulations that forced them to move to the VHF and UHF bands, now highly exploited by FM-radio and television stations. The amateurs have however been remembered and there are now slots reserved for amateur traffic in all major frequency bands. Contrary to the more and more chaotic situation on the commercial bands, amateur traffic is disciplined and truly worldwide, knowing no geographical nor political boarders. Through the series of "Oscar" satellites, which carry transponders for the amateur bands, the radio amateurs have also shown that they are still capable of staying in the forefront of todays communication technology.

The development of Personal Computing shows remarkably few similarities with that of amateur radio. Computer hobbyists did not appear on the scene until the early 70's, after 20 years of increased use of computers in the professional field. Moreover, after a relatively short transition time the interest of the computer hobbyists shifted from building hardware into writing programs. As a result the typical personal computer of today is delivered as a "turn-key" system with hardware facilities and system support programs comparable to large main frames only 10 years ago. The computer hobbyist also differs from the radio amateur in that the computer system is largely regarded as a tool that may be used to enhance some other activity such as game playing or stock trading. It is most certain however, that the massive amount of personal computers now being sold simply marks the beginning of the computer as a new home consumer product. Gradually we will be able to distinguish a fraction of "true" computer hobbyists, namely those oriented towards the computer science field. These people will very likely play the same part

in the development of their field as did the early radio amateurs. The prediction is based on the fact that the CS field does not rely heavily on any other fields, not even mathematics,which makes it easy to "join" the game. Since the serious hobbyists will probably outnumber the established research community by at least an order of magnitude [1] (BYTE magazine has already more readers than all of the professional computer publications taken together) it is obvious that new ideas and concepts will emerge from such a large collection of dedicated minds.

1.2 Why an Amateur Data Network

As home computers are entering the mass market there will be a growing federal and private interest to offer various kinds of database services. Several European PTT's are already carrying out field tests toward this goal ("VIEWDATA" etc) and some American companies offer special services for home computer owners (such as commodity prices etc). In both cases the telephone network is used. As these services expand, there will be an increased need for better data communication. When such a network appears, it needs to be very well designed since any major changes after its introduction are not likely to take place. An experimental network built by and for hobbyists will provide several important design criteria. It can be used as a testbed for routing and access algorithms, while at the same time provide the experimenter with a large number of active participants. As will be evident later on, such an experimental network will be just as useful for the testing of routing algorithms and protocols even if the target network is not intended to be radio based.

Finally, for the radio amateurs and computer hobbyists, such a data network will provide a new and rich experience promising great intellectual rewards to all those involved in its development and use.

1.3 Some basic network notations

A Data Network can be represented by a connected graph in which the nodes represent edges and the links represent connections that exist between the nodes. Figure 1 shows the basic structure of the graphs for three "classical" networks, ARPANET [2], the ALOHA system [3] and ETHERNET [4]. In these networks the digital information is sent as packets of limited size (in the order of 1 kbit) and each packet contains its destination address so that it can be easily processed and distributed. ARPANET is a multi hop

network in that the packets hops from node to node in order to reach the destination. In contrast, the Aloha network and ETHERNET are examples of one-hop networks. These are also called broadcast networks, since a transmitted packet will be intercepted by all those nodes connected to the transmitting node. Physically the Aloha network is radio based while ETHERNET consists of a coaxial cable onto which all users are connected.

ARPANET structure ALOHA structure ETHERNET structure

Figure 1

Sending information in packets (ie Packet Switching) is a fairly modern concept developed to suit "bursty" sources such as computers. In contrast to such a shared use of the communication channels we have the circuit switching technique in which an undisturbed reserved route exists between transmitter and receiver. The telephone network is a typical example of a circuit switched network. Physically the channel can be a wire, a time slot in a time division multiplex system (TDM) or a frequency band in a frequency division multiplex system (FDM).

2. BASIC NETWORK REQUIREMENTS AND OUTLINE

It is obvious that the requirements on a non-commercial network for experimental use differ from those on other public or private data networks. A service like accounting need not be supported at all while other aspects of technical and economical nature may need to be more heavily weighted. Below follows what is believed to be reasonable assumptions regarding node complexity and requirements on the network.

2.1 Node complexity

A typical personal computer system of today consists of an 8 or 16 Bit micro computer and less than 32 kbyte of memory. Typical execution speed ranges from 1 to 5 μs for simple register to register and memory to register instructions. It can be expected that there will usually be memory available to hold even very complex node programs. A bottleneck is more likely to occur with regards to speed limitations and great care must be exercised to ensure a reasonable tradeoff between e.g. the "smartness" of a routing algorithm and its execution time.

The modem and radio parts are slightly modified off-the-shelf products of relatively low cost. This is true as long as bit rates are low (<10 kbit/s) and the bit stream is modulated onto a carrier in the popular 2 meter band (14 4 MHz). For moderate bit rates (<100 kbits/s) it is still possible to keep the modem part fairly simple but radio equipment will become more expensive as we will have to move to amateur bands in the UHF region (>300 MHz). For still higher bit rates the modems will become increasingly expensive as the modulation technique gets more complicated. Adaptive equalization and matched filters based on SAW-devices [5] are techniques which are probably too costly for most amateurs.

Apart from the standard units described above, it may be necessary to include a dedicated hardware unit for buffering/sending of data and checksum calculations prior to transmission. Several micro computer manufacturers offer easily interfaceable I/O chips that can perform such tasks.

2.2 Network topology

A general requirement on a data network is that it is robust against failures in its nodes. Faulty nodes must not be hazardous to the operation of the network. Specifically, if repeaters break such that the network can split into several non-connected subnets, a failure should only affect traffic going between the parts. All other messages should be unaffected.

The concept of stations [5] which assumes a hierarchy among the nodes is probably not practical in our case, since there is no natural hierarchy of users. This lesson can be learned from the experiences with voice repeaters for the VHF amateur band. Although organizational efforts led to the construction of these repeaters, it has been shown very hard to motivate people to maintain them efficiently. The reason is of course that all work is based on voluntary efforts which diminishes rapidly when the task is no longer intellectually rewarding. This property of the

human mind has to be fully understood and taken into consideration in the network design, otherwise the project will simply never be realized. Thus, as parts are built they have to be "incrementally rewarding" in the sense that they can be put to work immediately.

Based on such considerations the conclusion is that the network should be totally decentralized and stationless. Since nodes will attach and detach in a quite random manner, the network is characterized by dynamic topology which is further stressed by the fact that some nodes may be mobile. Thus, a message will be carried in a multi hop fashion from the transmitting node (S) to the receiving node (R) using which ever intermediate nodes that are available at the time of transmission.

2.3 Switching

It is desirable to use the network both for interactive terminal-computer communication as well as computer-computer data transfer. The first case corresponds to transmission of single characters or short bursts of characters at a very low duty cycle. In the second case we typically have file transfers corresponding to (very) long messages.

It seems appropriate to chose packet switching in the first case and some kind of circuit switching in the second. However, since the network changes dynamically, we can not guaranty that a particular path will be available all through the transfer of a long file. The real time properties normally associated with circuit switching is not really demanded with regard to file transfers. To simplify network control it is suggested that packet switching is used here as well. A suitable mechanism need to be implemented to insure that multi packet files are reassembled correctly at the destination. This is a well known problem however and several approaches to file transfer protocols (FTP) exist (see e.g. [6]). Furthermore, by allowing variable packet size, the overhead associated with each packet can be made small in comparison with the data. Since it is felt that the major delays in transmitting packets will be due to processing time rather than transmission time, a packet can safely grow in size until these two terms approximately balance each other. Assuming a fairly complex routing algorithm (5-50 ms) and a bitrate of 100 kbit/s this rule of thumb gives packet sizes of a few thousand bits.

2.4 Local and global levels

The network as we have deduced it so far is shown in figure 2. Nodes which can hear (and disturb) each other are connected. The figure shows a fairly regular graph which is a highly simplified model of connectivity relations that may occur in a real situation. This is done on purpose however and more complex patterns will be used subsequently as the need for them appears. In the figure is indicated a possible path taken by a packet on its way from node S to node R.

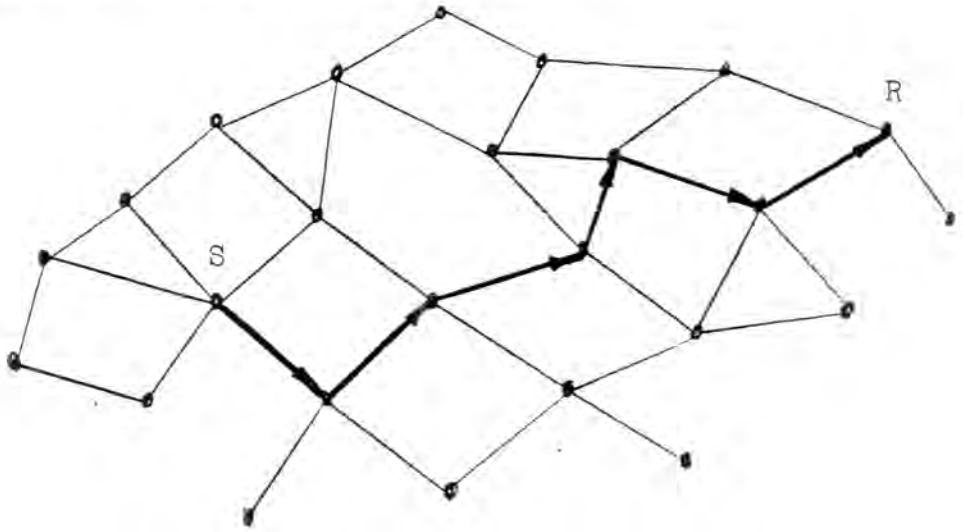

Figure 2

It is evident that one can distinguish between a local and a global level. On the global level we immediately find a "routing problem" namely how to direct the packet to its destination in shortest possible time. On the local level there is the problem of forwarding a packet to a neighboring node in a broadcasting mode while not interfering with packets travelling nearby.

In the following we will keep strictly to this hierarchical view on the network. The next three chapters will assume the high-level model and discuss algorithms for best packet distribution under the assumption that nodes can send packets to anyone of their neighbors without interference as were they connected by wires. If the reader finds difficulties in accepting this view he may consider a possible realization in which each node-pair communicates over a separate frequency channel chosen such that interference does not occur. Later on, in chapter 6, will we specifically address the "access problem" of the local level and relate techniques that give good broadcast efficiencies.

3. ROUTING

3.1 Introductory remarks

In a real network, nodes will have different amount of knowledge about other nodes and their position in the network. A particular transmitting node may have more information about the network than has anyone of the other repeating nodes. In that case the best path can be computed prior to transmission. The path forms a list of node identities and may be coded into the packet. As the packet is interpreted by a node n_i, that node checks the top item on the list and may find its own identifying code. If that is the case, the node removes the top element of the list and retransmits the packet. Thus the repeating node uses very little apriori information. It does not even have to know anything about its neighbors.

It is obvious however, that the related case is somewhat unrealistic. Since there does not exist a hierarchy between the users, we can expect that, at any instance of time, one of the repeating nodes in the previous example may want to originate a message itself. Either it does not have sufficient knowledge of the network, in which case it needs to rely on other nodes, or it has full knowledge in which case it was unnecessary by the first node in the previous example to transmit the full list.

In the light of this example it seems reasonable to study a "suboptimal" class of routing schemes (distributed adaptive routing) in which each repeating node makes its own judgement about the best route , sending the packet to the next node according to this route. In its simplest interpretation such a scheme may easily lead to dead-lock situations in which a packet oscillates between a number of nodes. It will be shown however that such a situation can be managed if we allow repeating nodes to communicate among each other by adding some extra bits to the packet.

To ensure that we do not exclude completely, the possibility for a node to take global responsibility for the routing of the packet , we will assume that the "envelope principle" is applied. According to this principle, a packet is wrapped in an "envelope" that bears the destination address. When the envelop reaches this address it is opened and examined. If another envelope is found inside it will be "posted" etc. In this way a packet can be partially or fully controlled through the network regardless of what particular routing algorithm is used. The envelope is really a level of the protocol structure to be further discussed in chapter 5. Knowing that it exists however, helps to accept the restrictions that we will assume throughout the rest of this chapter.

3.2 Node and packet information

Upon interception of a packet, a node has to make a judgement on whether to discard the packet, to accept it or to retransmit it. If the packet is retransmitted the node need also decide which of its neighbors should receive it. The judgements are based on previously stored information in the node (the apriori information) and information supplied by the packet. This information can be input to a "routing function" which is evaluated for each of the neighbors to find out the best repeater. The node and packet information complement each other but they may also substitute each other as in the following case. Assume that a repeating node n_{i-1} does not specify the next repeater n_i. In certain cases the neighbors of n_{i-1} may deduce which of them is the intended n_i if they all know the neighbors of n_{i-1}. Thus, by having knowledge of neighbors' neighbors, a node n_i can reevaluate the routing function using his knowledge about n_{i-1} to find out whether he is the intended recipient.

As there is very little overhead associated with sending a clarifying address to the next intended repeater, this seems to be an obvious choice. In the following we will take it for granted that a packet is addressed to only one neighboring node as is also the assumption underlaying the "global level" as described in section 2.4.

As was stated earlier a repeating node should base its decision regarding the best route on all the information that it has so far. This may include information gathered from previous packets. To simplify matters somewhat we will assume that repeating nodes just retransmits passing ~~packages~~ without remembering anything about them. This way we need not worry about an unknown amount of memory being used up to remember old packets.

The rest of the chapter will discuss routing algorithms based on various assumptions on the amount and type of information that is stored in the nodes and carried by the packets. To be able to see the gross effect of a certain algorithm we will assume that all nodes follow the same rules. We will particularly look at the cases where the nodes have i) no knowledge of the network, ii) full knowledge and iii) partial knowledge. The first two cases does not fit nicely into our assumptions and model and will therefore not be treated in any depth. The third case will be splitted up in two new cases, namely where a node have detailed knowledge about its neighbors (local routing information) and where nodes possess some global information.

3.3 Local routing information

The minimum amount of information to be stored in a node is obviously its own identity (ID). Likewise, the minimum routing information in a packet is the ID of the destination node. With only this information available, a repeating node can only make the decision to retransmit a received packet in the hope that it will propagate to its destination. Unfortunately, this will eventually lead to a situation in which all nodes keep sending the message for ever. The "wave-front" propagation can be controlled by either adding the IDs of the repeating nodes to the packet or using a hop-counter that is decremented for each hop. Another solution would be to relax our assumption about having no temporary storage of passing packets and allow the nodes to store and compare incoming packets to avoid retransmitting the same packet twice.

Packet distribution according to the above technique is very robust but extremely inefficient. it will not be recommended for use although we may leave such a possibility open when designing the protocols. Instead we will try to define parameters which can capture the essential of a good routing scheme. Since the nodes are making an incremental decision, each node trying to get the packet closer to its destination R, it would be nice to have some kind of distance function that could be evaluated at each node. Then a node would simply evaluate the distance function for each neighbor to find the next suitable repeater. Examples of relevant distances would be: i) the number of nodes to reach R or ii) the total delay to get to R. Unfortunately these functions are very difficult to evaluate as they are controlled by parameters that are net-dependent and therefore need to be updated continuously for all nodes. One way to get around this problem is to use a fixed net-independent metric and distance function such as euclidean distance as a rough approximation of what we really want. This approximation is fairly good for evenly distributed networks but may fail miserably in many other cases. Interestingly enough, although the approximation may be bad it will be shown that reasonable routing decisions can still be made.

3.4 "Shortest Distance" algorithm

Having access to the coordinates, in a fixed coordinate system of the destination node as well as the coordinates of itself and its neighbors a node may proceed according to the following simple algorithm upon reception of a packet:

Routing Algorithm R1:

1: calculate the distance between each of the neighbors and the destination node.

2: transmit packet to neighbor giving shortest distance. If own node is nearest, the packet is considered undeliverable.

This algorithm will work for evenly distributed networks and a packet will tend to follow a straight line drawn from the transmitting node to the receiver. In some cases it may have to make a large detour e.g. when the network contains holes as shown in figure 3. For such cases the distance function does not give a good estimate of topological relations. However, as long as the hole is convex, algorithm R1 will safely route the packet to its destination.

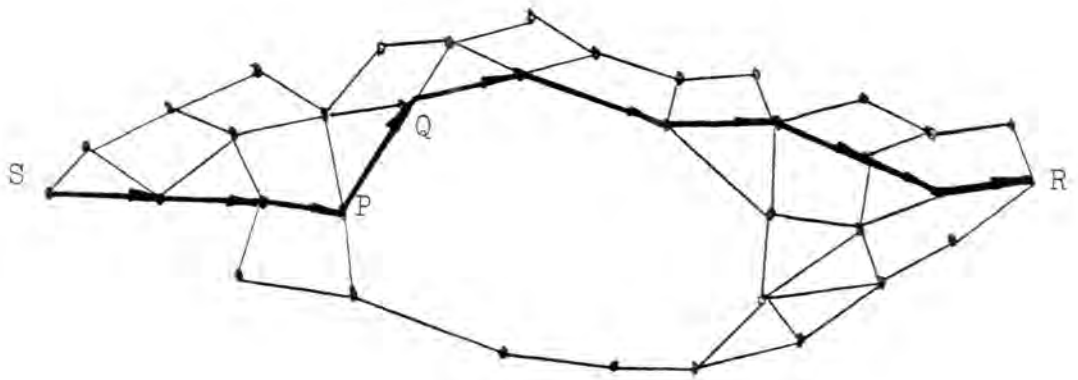

Figure 3

As shown in figure 4 the algorithm will experience severe problems if the hole has a concave "dent" along its border. The packet will be trapped in the concavity eventually being declared as undeliverable by the node (P) which is closest to R.

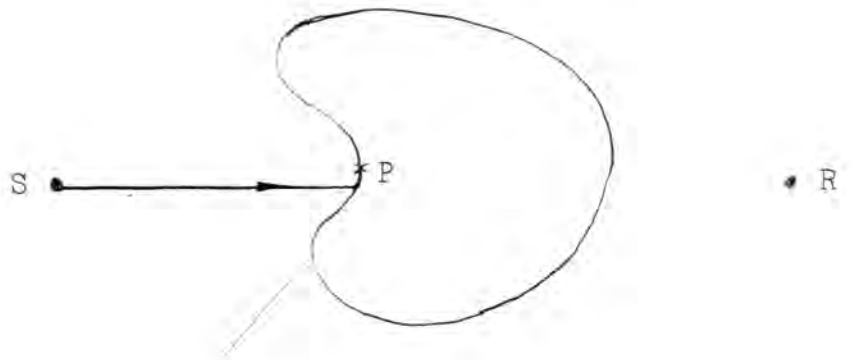

Figure 4. Hatched area indicates availability of nodes

By augmenting the algorithm slightly it is possible to come up with a new algorithm that can handle the situation in figure 4 to a certain extent.

3.5 Augmented "Shortest Distance" algorithm

We mentioned earlier that algorithm R1 tends to have a packet following a straight line connecting the transmitting node with the receiver. The line did not have any particular significant in itself. The chosen neighbor may lie on any side of this line and there may also be neighbors which are closer to the line than the one chosen. In our next algorithm this line plays an important role.

Routing Algorithm R2:

1: Draw a line , L, to the destination node. Calculate the angles, α_i , between this line and corresponding lines to the neighbors (positive direction = counter-clockwise).

2: Transmit packet to neighbor corresponding to the smallest positive angle (α_{min}). If there is no neighbor available with a corresponding positive angle $<= \pi$ rads, then packet is considered undeliverable.

Routing according to this algorithm is visualized in figure 5. It is easily verified that a packet reaching the concavity point P will indeed continue on a route that eventually will lead to R.

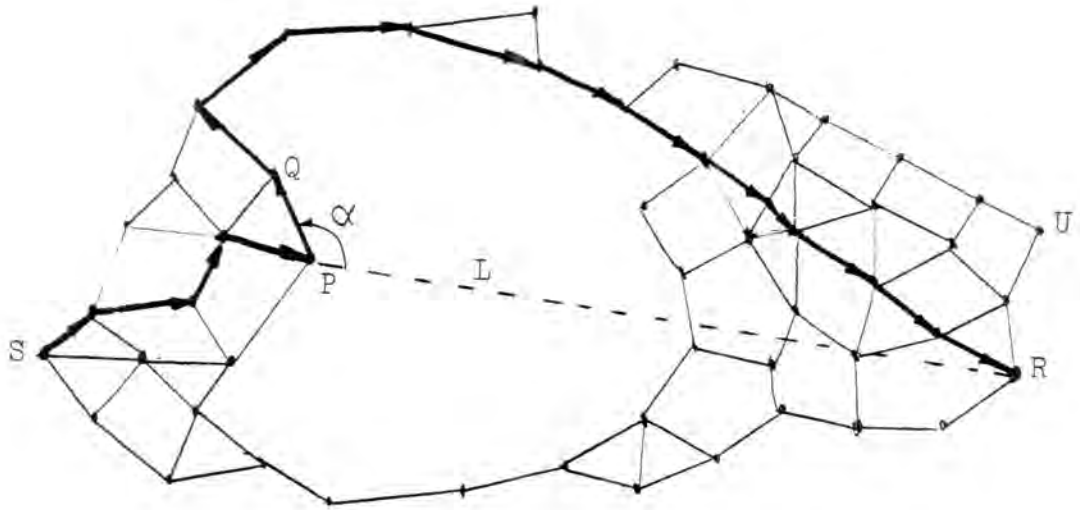

Figure 5

Although this algorithm may find its way out of a concavity it is not entirely foolproof as can be understood from figure 6.

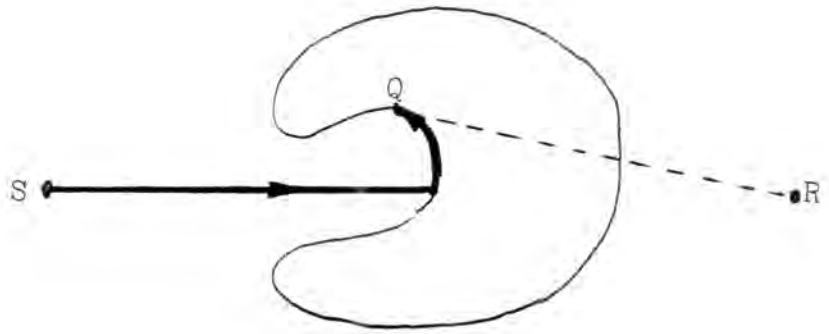

Figure 6

It is evident that a packet reaching Q will be considered undeliverable. Applying the angle criterion will namely yield a node that will no longer be on the boarder but rather below and to the right of Q. Thus, such a packet would find itself caught in an infinite loop.

3.6 Border following

It seems obvious that in order to securely manage a hole in the network we need to develop a reliable border following algorithm. The problem is very similar to the Picture Processing

problem of locating edges of binary objects. There are some differences however. The nodes (or pixels) are usually restricted to lie on a rectangular or hexagonal grid. Furthermore, a pixel which corresponds to inactivated background may at any time be activated to hold temporary values as part of the evaluation. In our case "background" corresponds to the lack of nodes and this emptiness can never be instructed to hold any temporary values!

Fortunately, the standard border following algorithm used in the picture processing field (see e.g. [7] pp 342) does not rely too heavily on the above differences and can easily be adapted to suit our needs.

Border-following Algorithm BF:

Given two border points P, Q such that a packet travelling from P to Q has the empty region on its right side proceed as follow from Q:

1: Calculate the angels βi between the line Q-P and corresponding lines to the remaining neighbors (positive direction as before).

2: Transmit packet to neighbor corresponding to the smallest positive angle (βmin).

As is seen in figure 7, the border following algorithm is closely related to algorithm R2. The difference is that now we use the direction to the previous node as a reference rather than the direction to the destination.

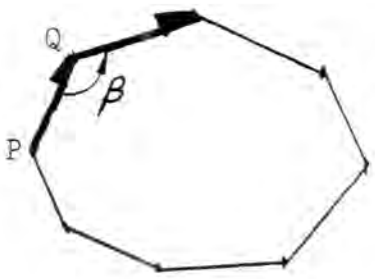

Figure 7

3.7 Algorithms based on border following

There are two problems which must be resolved when designing routing algorithms using border following. The first is the problem of identifying two adjacent nodes P, Q on the border. The second problem is to establish a criterion that tells when to leave the border.

The first problem can be easily solved by applying algorithm R2 on the first node P that is found to be on the border (figure 5).

Next we present two routing algorithms which differ only in their criterion for how to leave the border.

Routing Algorithm R3:

1: If own node is closer to R than previous node, use algorithm R1. If undeliverable, use algorithm R2.

2: If own node is not closer to R than previous node use algorithm BF.

This algorithm has an intuitively pleasant behavior (figure 8) in that it uses the "shortest distance" until it hits a hole with a concavity (P). It now follows the border until the next border point (Q) is again closer to R making it return to "shortest distance" mode.

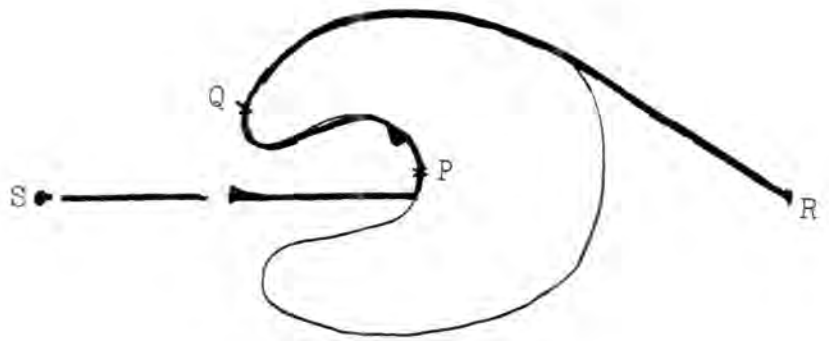

Figure 8

In order to see the limitations of this algorithm we will have to distort our hole into the shape shown in figure 9.

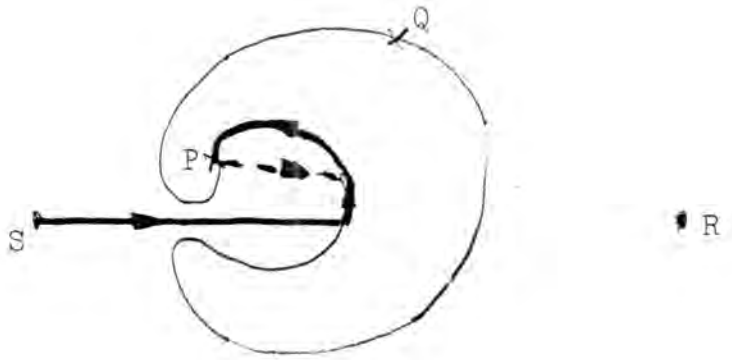

Figure 9

At point P there exists border points which brings the packet closer to R. Therefor the packet will enter the dotted line leading to a never ending cycle! Contrary to the previous algorithms it is not possible to detect such a degenerated state without adding some temporary information to the packet.

In our next algorithm, the "mode-switching" algorithm, we will more strictly control the point where we leave the border.

Routing Algorithm R4:

We assume that the packet holds a logical variable (MODE) and has space to store a temporary position.

1: If MODE is false then proceed according to algorithm R1. If undeliverable, set MODE true, load own position into packet and proceed according to algorithm R2.

2: If MODE is true, compare own position with the temporary position stored in the packet. If own node is closer to R, change MODE to false, delete the temporary position data and proceed according to algorithm R1.
If own node is not closer to R then proceed according to the border following algorithm BF.

The MODE switch is used to indicate when a packet is controlled by R1 (MODE=false) or by BF (MODE=true). The packet will not leave the border following mode until it arrives at a node which is closer to the destination than any previously encountered node (node Q in figure 9). This algorithm will find its way through rather awkward situations like the one shown in figure 10.

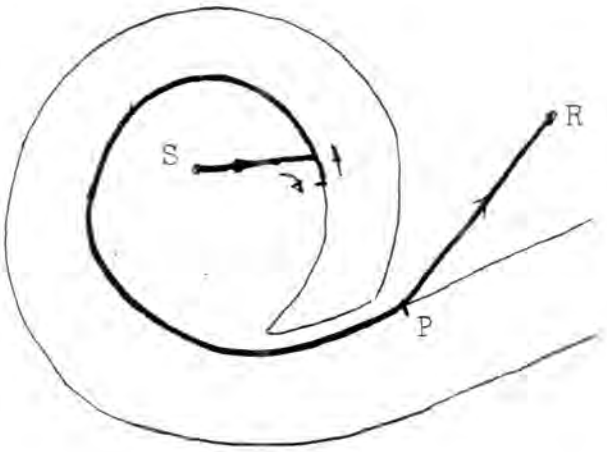

Figure 10

Unfortunately, not even this algorithm is without its flaws. Consider the simple graph in figure 11.

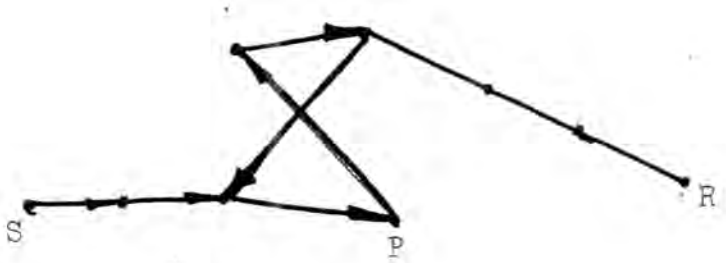

Figure 11

Although there is a simple route between S and R the algorithm will not find it. At point P border following will start and the packet will travel on a Moebius ring eventually ending up where it started.

It may be argued that the cases described represent pathological cases which will rarely occur in a real life situation. This is not true however. The holes may very well represent lakes or other geographical bodies in areas with otherwise dense distribution of nodes. Furthermore, even in a fairly homogeneous population of nodes may topologies as the ones described occur on a microscopic scale. Consider for example figure 3 with node P slightly moved to the right. This will create a local concavity with devastating effect if routing algorithm R1 is used.

3.8 Partial global information - the Skeleton

Although routing algorithms based on local routing information, such as knowing the position of the neighbors, may work in most cases, it is inevitable that the lack of global information will eventually lead to bad routes (figure 12).

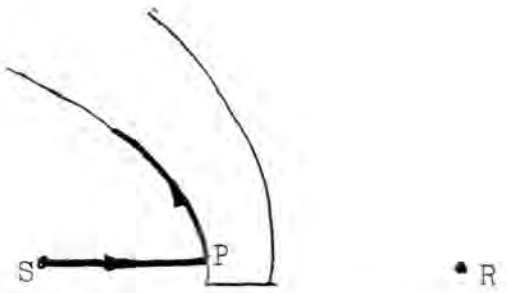

Figure 12

Following the border algorithm to the right instead of to the left when reaching P would yield a significantly shorter route. We have previously declared it not practical to assume that nodes will have full topological knowledge. The remaining part of this chapter will therefor present ways to acquire and distribute partial global information.

We will particularly look at properties which can be computed simultaneously by all nodes. This will speed up the calculations significantly and makes it possible to collect results faster than in "linear time". By "linear time" we mean that the total computation time is proportional to the number of nodes in the network.

The set of points of an object whose distances from the nearest border are locally maximum (i.e. no neighboring point has greater distance to the nearest border constitutes the skeleton of the object (figure 13). The skeleton is sometimes called the "medial axis" or "symmetry axis" of the object [7, pp 357-362].

Efficient parallel algorithms for calculating the skeleton of objects in digitized images can be found in the picture processing literature. The algorithm described here is a modified version suitable for our type of "array machine".

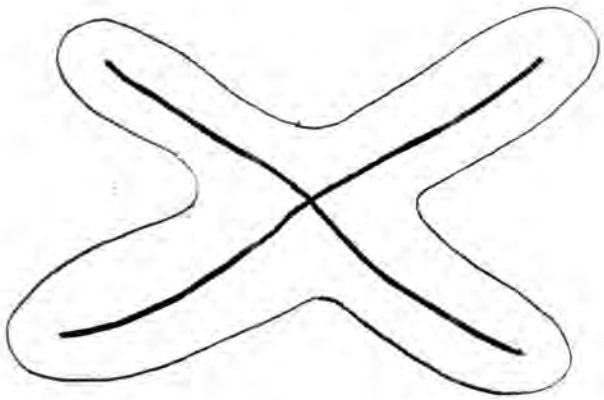

Figure 13. An object and its skeleton.

Algorithm for constructing the skeleton:

1: A skeleton initializing packet is sent to the border. If the node sending this package does not know where the nearest border is, it can chose a virtual destination point well outside the network in any direction.

2: When the packet reaches the border its status is changed into border following mode, making it traverse nodes along the "edge" of the network. These nodes recognizes the packet and labels themselves as "border nodes". Eventually the packet will return to its first border position where it was labeled by the node that changed the status. If that node is no longer available, the packet can be "timed-out" after a reasonable number of hops.

3: In the next step all border nodes tell their neighbors about the fact that they are on the "edge" of the network. A neighbor who is not a border node labels itself as being 1 hop from the border and notifies the neighbors in turn. Nodes which receives several such messages discards all but the one containing the lowest hop number.

4: A node that finds itself having a higher (or equal) hop number than its neighbors constitutes a local maximum point. It is thus a member of the skeleton and signals this fact to those neighbors having lower hop numbers. They in turn distributes the information to lower level nodes.

Thus the algorithm consists of 4 parts:

1: initialize
2: mark border points in sequence
3: propagate border information (inward propagation)
4: propagate result (outward propagation)

If the border and skeleton nodes submit their position in the transmitted packet, then all nodes in the network will know how they are located with respect to the nearest border and skeleton node.

Each of the four parts can be performed in times which are either proportional to the network's diameter (1,3,4) or its perimeter (2). For equally distributed and "compact" networks, the computation time will thus grow as the square-root of N, the total number of nodes. For networks which has a dense border and few internal nodes, the computation time will increase more linearly with N.

3.9 Thinning and object labelling

It is beyond the scope of this paper to dwell further into the use of skeleton information for efficient routing. One natural interpretation however, is to view the skeleton as a "highway" system spanning the entire network. A routing algorithm based on this view may want the skeleton to be a "connected set", i.e. the skeleton nodes should form a fully connected network by themselves. This is not always guaranteed by our simple skeleton algorithm, although it can be modified to yield fully connected sets.

A conceptually simpler approach to achieve an object similar to the skeleton is to use iterative border following in such a way that layer after layer of the network is stripped off. This "thinning" procedure can be performed such that the result object is fully connected. Its computation time is linear with N for any shape of the network. Interesting applications related to thinning includes the possibility to send a packet through all the nodes of the network such that it will only visit each node once. Such a packet can be used to count the number of nodes or collect statistics etc.

In certain cases it may be desirable to define subnets within the network. In picture processing terms these can be considered as "objects" within the network "object". Using similar approaches as above, it is possible to calculate several properties of such objects including their size, the number of nodes involved and connectivity relations. Such algorithms can in fact be more sophisticated since the option now exists to use an "active" background.

4. ACK and NAK

From chapter 3 it is obvious that a packet may end up at many places other than the intended receiver. Apart from pure routing error, the packet may be lost due to bad radio reception or faulty repeaters. In less severe situations, the packet may be delivered to its destination but may contain errors. In this case the receiver can request retransmission of the packet. This is equivalent to sending a negative acknowledgement (NAK) message. To inform the transmitter about lost packets, it is obvious that positive acknowledgement (ACK) should also be used. In fact, in our type of network, ACK seems to be of fundamental importance in obtaining reliable communication.

The use of acknowledgement messages is critically dependent on actual application. Single character communication in an interactive environment, needs probably not be acknowledged at the network level, since higher level response (such as echoing) acts just as efficiently. Transmittance of a file however, will probably need a rather "tight" handshaking procedure to ensure that all parts of the file have been received properly.

This latter situation lends itself to a nice analysis regarding how often an ACK message should be sent in relation to the number of transmitted packets. If every transmitted packet is acknowledged, an error in one of them means that only one packet needs to be retransmitted. The total number of packets sent and total transmission time may however become unnecessary large, particularly if the error rate is low. On the other hand, if only one ACK message is sent for a group of packets, then this whole group needs to be retransmitted in case of error. The proper balance is found easily by looking at three different situations which describes what may happen to a group.

Define: P_e = probability of losing a packet
k = number of packets in a group
x = average number of packets sent per user packet

	probability	no. of packets
1: The group is properly delivered and ACK is received.	$(1-P_e)^{k+1}$	k+1
2: The group is properly delivered but ACK is not received. Retransmit till OK.	$(1-P_e)^k P_e$	k+1+kx
3: The group is not properly delivered. Retransmit till OK.	$1-(1-P_e)^k$	k+1+kx

Thus we have:

$$kx=(1+P_e)^{k+1}(k+1)+(1-P_e)^k P_e(k+1+kx)+[1-(1-P_e)^k](k+1+kx) \qquad (4.1)$$

solving for x yields

$$x=\frac{k+1}{k(1-P_e)^{k+1}} \qquad (4.2)$$

taking the derivative with respect to k

$$\frac{dx}{dk}=\frac{k-(k+1)[k\ln(1-P_e)+1]}{k^2(1-P_e)^{k+1}} \qquad (4.3)$$

and minimizing gives

$$k_{opt}=\sqrt{\frac{1}{4}-\frac{1}{\ln(1-P_e)}}-\frac{1}{2} \qquad (4.4)$$

This relation is shown graphically in figure 13.

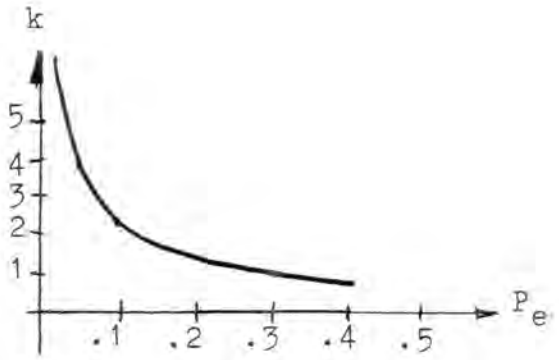

figure 13

5 PROTOCOLS

5.1 Thoughts about protocols

If it is at all relevant to consider the network as an array machine, then the implemented protocols represent the available instruction repertoire for this machine. As an illustrating example, we may take the routing algorithms presented in chapter 3. They define actions that each node has to obey upon reception of input data (the packets). Thus if e.g. algorithm R3 has been implemented among other node functions, then the control field of a typical data packet would initialize execution of that algorithm. A qualitatively different example is given by our description of the skeleton algorithm. The algorithm is actually not described in the same sense as the previous routing algorithm, since we have not specified what the individual nodes should do but rather what the overall behavior should be! The reader should have no difficulty figuring out the specific node behavior however, since the particular operations were explained earlier in the text. The skeleton algorithm is furthermore interesting in that its purpose is not to transmit packets but rather to compute some values to be stored in the individual nodes, such that these values may eventually be used for routing purposes.

If we know exactly all the algorithms that this array machine would execute, then the most compact instruction repertoire would consist of just these operations. This is typically what is the case in most implemented networks. In an experimental network like ours this can not be accepted since one routing algorithm may be replaced by another etc. In the most general case, the nodes present the users with a general purpose computer giving him full control of all node functions. The protocol then corresponds to the instruction repertoire of that machine. If the language used is very low level, then the user is faced with submitting a long program with each packet, since he can not rely on the nodes remembering anything about his previous transmission. One reason being that the network is a shared multiuser resource.

Fortunately, in our case, the aim is not to develop protocols for general purpose use but specifically for transferring messages, files etc. between the nodes. Introduction and changing of facilities will be relatively occasional events that may well use a lot of transmission capacity. This leads to the criterion that, as long as we leave a facility opened to get access to the micro computer controlling the node, the remaining part of the protocol can be oriented towards some specific implementation (compare [5]).

It is still important though, to find common denominators among the various algorithms we want to implement. Such a set of operations may form the kernel of a language, specifically oriented towards the kind of operations that the users might want the network to perform.

One example that illustrates such a common denominator is to view the instruction of sending an ACK upon reception of a packet as simply an example of the previously described "envelope" principle, i.e. the ACK message is actually supplied in the packet but wrapped in an envelope addressed to the source. Thus we think of the receiving node as a machine that opens envelopes keeps messages addressed to itself and forwards messages addressed to other nodes.

It may be argued that the packet would be unnecessary large if it also had to contain the ACK message. This need not be the case however. The source must in all cases agree with the receiver about an unique identifier to be included in the ACK message.Furthermore, by using appropriate coding, e.g. default code for the addresses, the overhead may be kept to a minimum.

Likewise, with respect to the different data types there may also exist logically identical operations such as store-and-forward file, which corresponds to the basic repeater function usually associated with packets. In such a context it would be logical to put a whole file into an envelope. Unfortunately, attempts to include such general functions in other networks has not been very successful, as they tend to be incompatible with existing operating systems [8]. The work on development of good file transfer protocols is advancing however and it may be a good idea to incorporate a well known FTP (such as TCP) already in the initial specification.

5.2 Layered Protocols

A basic philosophy in modern protocol construction is to use layered protocols. By this is meant that the hierarchy of logical functions that we want to control should also be reflected as identifiable parts of the protocol. In this way a certain subfunction can be changed without influencing any other part of the protocol. Usually subfunctions are visualized as machines or "physical" systems. A typical example is the end-to-end transmission. These levels represent communication lines of various sophistication , each of which could have been implemented physically.

Our model of the network fits very nicely with the idea of layered protocols. The lower level (local level) corresponds to the neighbor level and has the only purpose of forwarding the

packet one hop. The next level (global level) would thus correspond to the various aspects of routing a packet from source node to destination node. From there on would the more exotic message and file transfer protocols emerge.

5.3 Addressing

We have previously described a simple address decoding principle based on putting packets in envelopes. This mechanism is very powerful when a node wants to send the messages to groups of receivers. Depending on their position in the network, the originating node has the choice to group messages together or to send them separately. If a grouping of messages are done they will eventually need to split up as they come closer to their destinations. The originating node can fully control this process by putting several envelopes into one larger envelope all of them having different destination addresses. If the message to be sent is identical this can be indicated by a default code as mentioned previously, so that the message needs only to be coded once.

Furthermore, an anonymous ID could be allowed and meaning that the packet is distributed to the node nearest the given position or distributed to all nodes within a circle of some specified size.

The object labelling technique discussed in section 3.9 will also offer the possibility to spread messages to well defined subgroups of the users. In this case it is sufficient to deliver the message to one node which is a member of the group. A propagation algorithm active within the group will quickly distribute the message to all members.

6 ACCESS METHODS

6.1 Possible schemes

We have so far neglected to discuss what techniques can be used to achieve interference free transmission between neighboring nodes. The method of assigning a different frequency channel to each node pair, as suggested in the introduction, is inefficient but technically quite possible. An administrative problem regarding the assignments of channels need to be solved. A relatively large amount of channels will be needed, since nodes who share the same neighbor(s) must use different sets of frequencies.

Instead of FDM we could use TDM which would simplify the radio part somewhat. It is usually considered that such fixed allocation of the channel is bad for bursty type of traffic with low duty cycle. Although it is true that the traffic generated by an originating node may be bursty, this need not hold for nodes working as repeaters in the network. The flow of fast and slow streams of packets mixes and becomes averaged over many repeating nodes.

We will however not advocate the use of the above techniques. The reason is that there exist probabilistic channel sharing methods which can be shown to perform fairly close to maximum rate [9,10,11,12] Probabilistic channel sharing means that all nodes use the same channel. Since the channel can be more broad banded than in the previous case, a packet will occupy the channel a significantly smaller amount of time. By simply transmitting packets over the channel as they arrive, a statistical amount of them will come through while others will collide. Colliding packets are retransmitted so all packets will eventually be communicated.

The parameters that control the sharing of the channel are mainly the individual probabilities governing the rate at which the nodes will attempt to transmit. The nodes may improve their strategies by measuring the activity on the channel and adjust themselves accordingly. Based on these assumptions and a decentralized optimization criterion, it is possible to achieve an optimal solution for the transmission policies [11].

6.2 Carrier sense

It was soon realized that sensing the channel prior to a planned transmission could be very valuable. If the channel was already occupied then the packet could be rescheduled in the same way as

had collision been detected [12]. This technique of carrier sensing or CSMA can achieve up to 85 % throughput.

In our system one has to be careful when evaluating the significance of the carrier signal. Given that a carrier is actually detected this does not necessarily mean that a packet can not be sent anyway, namely if it is sent to a node that is not disturbed by the sensed carrier. Likewise, if a carrier is not sensed, the intended recipient of the packed can still be disturbed by a signal that the transmitting node did not hear.

This problem can be attacked in several ways. The most obvious is just to neglect it. The sensed carrier still contributes useful information and will improve the performance compared to the case where it is not used at all.

The second approach is to generate an artificial carrier that fulfills the role that is expected from it. Assume that a node that is currently decoding a message also puts out a carrier strong enough to be sensed by his neighbors. This carrier must of course use a different frequency but its information content is extremely low, leading to low bandwidth requirement.

Finally, if a node could detect the signal reaching any of its neighbors than half of the problem will disappear. This can be achieved by having two receivers, one very sensitive for carrier detection and one less sensitive for message signal detection. The sensitive receiver can in fact be further utilized if it is equipped with full decoding facilities. By monitoring packets which are intended for neighbors, an over all view of the local channel activity can be gathered.

Using a system with two receivers (or one switched receiver) effectively means that we use different neighborhood sizes when listening. This complements a technique to be described in the next chapter in which we will use different neighborhood sizes when transmitting.

All three techniques described above lends themselves to CSMA algorithms as described in [12]. Whether these algorithms should be modified to account for the peculiarities of the "new carriers" remains to be investigated.

To summarize, we thus recommend the use of CSMA possibly with an augmented carrier sense function. Difficulties to synchronize a network that may extend over a large area and contain thousands of nodes will probably prohibit the use of slotted algorithms.

7 CHOOSING NEIGHBORHOOD SIZE

A very important design parameter is the power of the transmitters. In a modern transistorized transmitter, which is e.g. frequency modulated, the output power can be easily controlled. This fact can be utilized to the benefit of the whole system as will be shown below. When a node uses high power it reaches far and can transmit packets with few hops (low delay). On the other hand, it will disturb many nodes causing the total traffic to become lower. In the extreme case where all nodes are neighbors, only one packet at a time can be transmitted. The opposite extreme is for all nodes to have only one neighbor. Many parallel packets can be sent simultaneously but delivery time for each packet will be very long.

What we really want to control is the number of neighbors. To find the optimal value, we need to know the relationship between total delay when sending a packet a certain distance and the number of neighbors. This relation will depend critically on the traffic so it would be desirable to have that as a parameter.

We will make the following assumptions:

S1. The network is evenly distributed and behaves symmetrically.
S2. A neighborhood area , consisting of n-1 neighbors, follows the same relations between throughput, delay and offered traffic as that of a one-hop network with n nodes.
S3. Offered traffic in a neighborhood area grows proportionally with the number of nodes.

Assumption 2 is a key assumption. It can be justified on the basis that the network is symmetrical, so that on the average the same number of nodes (n) will be disturbed for each transmission in both systems.

Define:

S = normalized throughput for 1-hop, n-node network
D = delay per node -"-
T = message delay for average message
g = normalized user offered traffic at each node
n = number of neighbors + 1
d = average distance per message/
average distance between nodes
m = average number of hops

From the literature we get for any particular access method

$$D = D(S) \qquad (7.1)$$

(see e.g. [12] for CSMA-algorithms)

Since the network is evenly distributed the average number of hops decreases with $\sim\sqrt{n}$ as the neighborhood area grows.

Thus for the average number of hops

$$m \approx d/\sqrt{n} \qquad (7.2)$$

The traffic to be handled by the nodes is g * m due to S1 and according to S2 we may equate

$$S = n * g * m \qquad (7.3)$$

Thus for the total delay

$$T = D(m) * m \qquad (7.4)$$

which is the desired result.

To verify this result against our intuition we proceed by choosing a simple example.
A simple analytical expression for the delay D can be obtained by viewing the neighborhood area network as an M/M/1 queue with utilization factor S. This gives

$$D = 1/(1 - S) \qquad (7.5)$$

(Compare this relation with [12, fig 12] for the optimum p-persistent CSMA algorithm.)

Combining (7.2), (7.3), (7.4) and (7.5) gives

$$T = d/(\sqrt{n} - ngd) \qquad (7.6)$$

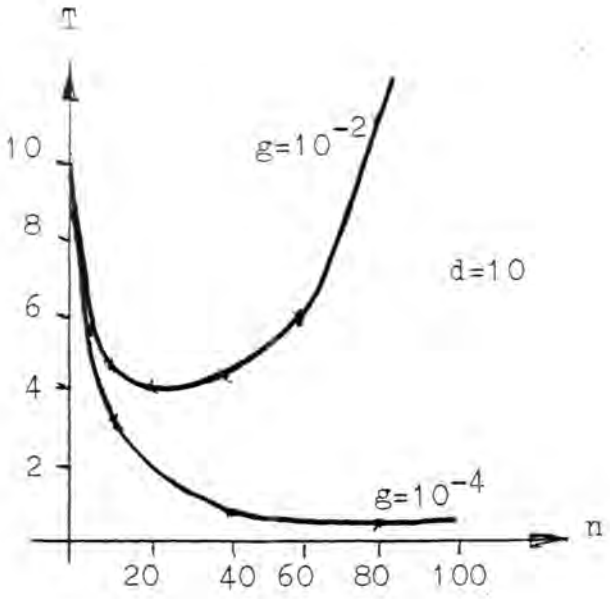

Figure 14

In figure 14 we have plotted the delay as a function of the number of neighbors for a network with d = 10 and for two different rates.

The balancing between channel utilization and neighborhood size is evident for the higher value of g. When the traffic rate is lowered, the total delay is minimized by expanding the transmitting range as much as possible.

Taking the derivative of T with respect to n yields

$$n_{opt} = 1 / 4(dg)^2 \qquad (7.7)$$

This function is shown in figure 15.

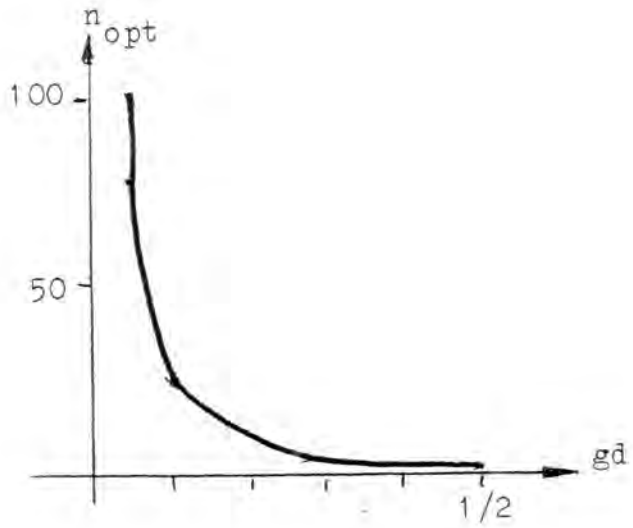

Figure 15

It is evident that a network that changes its neighborhood area according to the traffic may gain considerably in lowering its delay. Using the assumption that traffic flow is evenly distributed (at least locally), the nodes can themselves adjust their range using throughput estimations and relation (7.7).

8 SUMMARY

We have touched on some of the basic properties regarding a fully distributed experimental data network for radio and computer hobbyists. By some rather fundamental assumptions we were led to a stationless packet switching network. A model was introduced that separated the local broadcasting level from the global topological and hierarchical level. The two levels were then treated separately.

The routing problem was studied in some detail, as no reference could be found that was directly applicable to this type of network. The notation of the network as an array processor was introduced to enable the transfer of algorithms from the picture processing field and to give a different view upon the system. Protocols are available in a rich variety, several being standardized for world wide use. In the light of this development it was felt that a philosophical view upon protocols as seen from the machine perspective would be more entertaining than a study in bits and fields.

Turning to the local broadcasting level, the interesting technique of probabilistic multiplexing is suggested as the basic method of use. Earlier analysis did not utilize the flexibility of the radio system. By recognizing the "capture effect" and the possibility to vary the input threshold of receivers and power of transmitters, it seems likely that a more efficient use of the medium will be achieved.

The particular influence of transmitter power was then discussed. Due to the lack of theoretical results in this area, some simplified assumptions were made to calculate a relation between the number of neighbors of a node and delay as a function of input traffic rate. It was found that the total delay can be significantly lowered by choosing appropriate transmitting power.

Many questions which have not been treated here will require some attention prior to an implementation. These include the subjects of flow control, routing techniques in the repeaters of long-distance repeaters, the capability of automatic position detection, system and data integrity, priorities and internetting. The validity of our two-level model is also an important issue. Finally, as were mentioned in the introduction, much effort must be spent to ensure that the network can be bootstrapped step by step, each step being highly rewarding to motivate its implementation.

Acknowledgement

An ACK is forwarded to Dr. John Silvester, University of Southern California without whose encouragement this report would never have been written.

References

[1] Scientific American, September 1977

[2] L.G.Roberts, B.D.Wessler,"Computer network development to achieve resource sharing", AFIPS Spring Joint Computer Conf, vol.36, May 5-7, 1970, pp 543-549

[3] R.Binder et al.,"ALOHA packet broadcasting - A retrospect",Proc. national Computer Conference 1975, pp 203-215

[4] R.M.Metcalfe.,"Ethernet:Distributed packet switching for local computer networks", Communications of the ACM,July 1976, volume 19, number 7, pp 395-403

[5] R.E.Kahn et al.,"Advances in packet radio technology", Proc. of IEEE, Vol.66, no.11, november 1978, pp1468-1496

[6] IEEE Computer: special issue on computer networks, september 1979

[7] A.Rosenfeld,A.C.Kak.,"Digital Picture Processing",Academic Press 1976

[8] D.C.Walden,A.A.McKenzie,"The evolution of host-to-host protocol technology", IEEE Computer, September 1979, pp 29-36

[9] N.Abramson," The throughput of packet broadcasting channels ", IEEE trans on Communication,Vol. COM-25,No.1, January 1977, pp 117-127

[10] L.Kleinrock,Y.Yemini,"An optimal adaptive scheme for multiple access broadcast communication", Proc. ICC 1978

[11] Y.Yemini,L.Kleinrock.,"On a general rule for access control or, silence is golden...", Flow Control in Computer Networks, J.-L.Grange and M.Gien, eds, IFIP, North Holland Publishing Company, 1979

[12] L.Kleinrock,F.A.Tobagi.,"Packet Switching in Radio Channels:Part 1...",IEEE trans.Communications, Vol COM-23, No.12, December 1975, pp 1400-1416