DELAY CONSIDERATIONS IN PACKET RADIO NETWORKS WITH CAPTURE

by
Charles Rossi, Jr.

B.S. Electrical Engineering, Cornell University (1983)

SUBMITTED TO THE DEPARTMENT OF ELECTRICAL ENGINEERING AND COMPUTER SCIENCE IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF

MASTER OF SCIENCE IN ELECTRICAL ENGINEERING AND COMPUTER SCIENCE

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY
September, 1986

Copyright (c) Charles Rossi, Jr.

The author hereby grants to M.I.T. permission to reproduce and to distribute copies of this thesis document in whole or in part.

Signature of Author

Department of Electrical Engineering
August, 1986

Certified by

Professor Robert Gallager
Thesis Supervisor

Accepted by

Professor Arthur C. Smith, Chairman
Committee on Graduate Students
DELAY CONSIDERATIONS IN
PACKET RADIO NETWORKS WITH CAPTURE

by
Charles Rossi, Jr.

Submitted to the Department of Electrical Engineering and
Computer Science on 4 August 1986 in partial fulfillment
of the requirements for the Degree of Master of Science in
Electrical Engineering and Computer Science

ABSTRACT

This thesis studies the effect of routing strategy on the delay incurred in lightly
loaded random access packet radio networks with capture. It does so by analyzing
the "velocity" of packets, i.e. the progress towards the destination divided by the
average delay. The networks considered in this thesis are stabilized by choosing
the retransmission probability so that the packet handling capacity of the network
is maximized when all nodes are backlogged.

An approximate delay model is developed which allows the performance of
specific routing strategies in specific networks to be compared. Two routing
strategies are compared for a sample 10-node network. The results of this
comparison suggest that as load increases in a network, the average transmission
radius should be made lower to maximize average velocity.

The delay model is then applied to more general networks of randomly
distributed terminals. The first case considered is the very lightly loaded case
where it is assumed that many nodes are within the optimum transmission radius
so that all transmissions are approximately to the optimal place. In this case it is
found that there is an optimum transmission radius which maximizes velocity. The
velocity obtained by always using this optimum transmission radius is an upper
bound for the more heavily loaded case.

An approximate analysis is then developed for the more heavily loaded case. In
this case velocity is found to be a function of the mean distance travelled towards
the final destination and the second moment of the transmission radius. Using the
results of this analysis, more specific routing strategies for networks of randomly
placed nodes can be compared with each other and with the upper bound. The
Closest to Optimum and Most Forward within R_p strategies are then compared
with each other and with the upper bound. It is found that the Closest to
Optimum strategy performs better than Most Forward within R_p and within 10% of
the upper bound in lightly loaded networks.

Thesis Supervisor: Dr. Robert G. Gallager
Title: Professor of Electrical Engineering and Computer Science
Acknowledgements

I would like to thank my thesis advisor, Professor Robert Gallager, for his help in doing the research and writing the report for this thesis. I would also like to thank M/A-Com Government Systems and particularly the management at the Boston Engineering Center for the scholarship that made my graduate education possible. Finally, I would like to thank my wife Sheryl for her constant support while I was doing my thesis as well as while I was fulfilling the rest of the requirements for a Master of Science degree.
# Table of Contents

1 Introduction ................................................................. 7
2 Delay Model ................................................................. 14
3 Very Lightly Loaded Case ................................................. 21
4 More Heavily Loaded Case ............................................... 27
5 Comparison of Transmission Strategies ............................... 31
   5.1 Closest to Optimum .................................................. 31
   5.2 Most Forward within $R_p$ ......................................... 32
6 Conclusion ........................................................................ 37
1 Computer Programs .......................................................... 39
   1.1 Program to Find Retransmission Probabilities and Velocity Given a Specific Network and Flows to Support the Network Load .................. 39
   1.2 Program to Find the Service Time in the Very Lightly Loaded Case ........................................... 48
   1.3 Program to Compare Transmission Strategies .......................... 52
## List of Figures

| Figure 1–1. | Progress towards Destination ........................................ | 10 |
| Figure 2–1. | CE Successful even though it Collides with AB .................. | 15 |
| Figure 2–2. | Placement of E so that AB does not interfere with CE .......... | 16 |
| Figure 2–3. | Retransmission Probabilities and Maximum ........................ | 18 |
| Figure 2–4. | Velocity Comparison for a Simple 10–Node Network ............... | 20 |
| Figure 3–1. | Area where there Are No Terminals ............................... | 22 |
| Figure 4–1. | Nodes around the Nodes around the Receiver ..................... | 28 |
| Figure 5–1. | Closest to Optimum Transmission .................................. | 32 |
| Figure 5–2. | Velocity vs. $\eta D$ for the 3 Transmission Strategies .......... | 34 |
| Figure 5–3. | Most Forward Within $R_p$ Transmission .......................... | 35 |
List of Tables

Table 3-1. Comparison of Velocity for the Lightly Loaded Case .......... 26
Table 5-1. Comparison of Transmission Strategies ....................... 33
1 Introduction

Packet radio combines the networking concepts developed for point-to-point networks with those developed for single-hop multi-access networks. A packet radio network consists of a group of nodes in a wide geographical area that communicate using a broadcast radio channel [1]. Each node generates communication traffic which is characterized by a low duty cycle in which a short burst of data is sent followed by a longer period in which no traffic is generated. Additionally each node acts as a repeater for multi-hop transmissions between other nodes. Each node can transmit to some subset of other nodes in the network, and thus most packets must be relayed through several nodes to get to their final destination. A packet transmission is successful only if no other nodes within interfering range of the receiver (including the receiver) transmit. Due to the relaying of packets, packet radio networks have many of the attributes of point-to-point networks. However, since packets are transmitted on a multi-access channel, packet radio networks have many of the attributes of single-hop multi-access networks.

The primary advantages of packet radio networks over point-to-point networks are mobility and rapid and convenient deployment [1]. Since packet radio uses a radio channel, terminals may move around rather than staying in one place. Likewise adding a node to a network requires just setting up a terminal rather than installation of a communications line between two terminals. For some applications, it may be advantageous to be able to quickly set up an entire network by setting up a group of terminals in a geographical area.

Additionally, packet radio offers several advantages over single-hop multi-access networks [1]. Since multiple transmissions can take place at the same time, the throughput of a packet radio network may be higher than that of a multi-access network. Also, since terminals need only transmit to a small subset of terminals around them, they require less power and smaller antennae. Thus, they are more mobile than terminals for single-hop multi-access networks.

A packet radio network should provide several basic capabilities in order to efficiently support real-time communication between terminals. The internal
operation of the network should be transparent to the user of the network. The network should provide error-free communication between any two nodes within the coverage area. The system should support mobile terminals at normal ground speeds within the network coverage area. It should support the required network throughput with acceptable average delay and should be stable so that average delay is bounded even during periods of peak usage. It should support flexible addressing options to allow users to send messages to specific subsets of network nodes.

This thesis studies how the routing strategy within the network should be chosen to minimize the average delay from initial transmitter to final destination. Given the load and topology of a network, one approach to this problem is for each node to transmit to the neighbor that minimizes the expected time that it takes for the packet to get to its final destination. The delay depends on contention in the radio channel, the routing used by all the other nodes, and on the queueing delay at the intermediate destinations. This thesis will examine the relationship between routing strategy and the delay introduced due to contention on the radio channel under the assumptions that each node knows only the location of the nodes around it and the final destination and that all other nodes use the same transmission strategy. Another approach to this problem would be to attempt to globally optimize delay given the topology and the end-to-end packet flow of the network.

Very little work is in the literature which discusses how to optimize the transmission radius with respect to delay in packet radio networks. Kleinrock [2] presents a simple argument which suggests that there is a critical transmission radius which optimizes delay. Takagi and Kleinrock [3] show how to pick R (subject to many assumptions) to maximize packet handling capacity of the network when using the Most Forward within R transmission strategy. Hou and Li [4] indicate that throughput is higher using the Nearest Forward strategy rather than the Most Forward within R strategy.

In this thesis, we assume that our network is operating well below its maximum packet handling capacity, and thus we are interested in minimizing end-to-end
delay rather than maximizing packet handling capacity. Since end-to-end delay depends on the distance travelled towards the final destination and the delay on each hop, the quantity to be maximized in this work is the average "velocity", V, of packets in the network. Velocity is defined as $E[R(\cos \theta)]/E[T]$ where $R$ is the distance from transmitting station to receiving station, $\theta$ is the angle between the line from the transmitter to the receiver and the line from the transmitter to the final destination, and $T$ is the amount of time between when a packet is received at the transmitter (either from outside the network or from another node) and when it is successfully received by the receiver.

The model to be considered consists of a large number of terminals randomly distributed according to a Poisson point process with rate $\lambda$ terminals per unit area. The terminals communicate with each other using a slotted radio channel. Thus, all packets are assumed to be of the same length and all terminals are perfectly synchronized in time. Each terminal acts as a repeater for multi-hop packet transmissions and also generates new packets according to a Poisson process with rate $\eta$ packets per slot. Each new packet is equally likely to go in any direction and must be transmitted an average total distance $D$ to get to its final destination. The terminal knows the location of the destination and of all terminals within its maximum transmission radius. Thus the transmitter will either transmit to the final destination or to another terminal that is in the direction of the final destination. We assume that $D$ is large compared to the distance the packets travel in one hop. With this assumption, most packet transmissions are to repeaters and not to the final destination and the progress towards the destination for most transmissions can be approximated by $R\cos \theta$ as shown in Figure 1-1. Stations transmit with constant power in a noiseless environment with perfect FM capture. Thus if a station transmits to a terminal at distance $R$ from it, the transmission is successful if and only if no terminal at a distance less than $R$ from the receiver (including the receiver) attempts to transmit in that slot. A transmitting station knows immediately at the end of a slot whether it was successful or not. When a node receives a packet that needs to be forwarded to another node, it transmits the packet in the next slot.

Ideally, we would like to schedule transmissions perfectly on a first-come first-
Assume that each node can be in one of two states: idle or backlogged. When it
is idle and it receives a packet it transmits it immediately in the next slot. If the
transmission is successful, the station enters the idle state. If the transmission is
unsuccessful, it enters the backlogged state. When node i is backlogged
attempting to transmit to node j, it attempts to transmit in each slot with
probability $p_{ij}$ and it buffers all new arrivals. The retransmission probabilities, $p_{ij}$,
are chosen so that the network is stable and so as to maximize throughput when
all nodes are backlogged. Since a node transmits with the same power no matter
what node it is transmitting to, the effect on the rest of the network is
independent of the receiving node j. Thus, for the rest of this work we will assign
a retransmission probability, $p_i$, to each node and let $p_{ij} = p_i$ for all j.

As long as D is large enough so that most transmissions are to intermediate
destinations, the load of the network can be specified by the product of $\eta$ and
D. We can then choose our units of distance such that $\lambda = 1$ so that the only
parameter in the problem is $\eta D$. All the derivations in this thesis include $\lambda$;
however, $\lambda$ is set to 1 for most of the examples.

Given our constant network load, five factors influence how large the
transmission distance should be. First as $E[R]$ gets larger, presumably the average
progress that a packet makes towards its destination during a successful
transmission gets larger. If $\theta$ is the angle between the line from the transmitter to
the final destination and the line from the transmitter to the receiver and D is
large, then the progress is $R \cos \theta$ as shown in Figure 1-1. Second, as $E[R \cos \theta]$
gets larger (as long as $D/E[R \cos \theta]$ is large), less packets need to be successfully
transmitted per slot in order to support the network load. The total packet
success rate per unit area must be $\eta \lambda D/E[R \cos \theta]$ to support our network load. We
use the independence assumption and assume that the total load of originating
and relayed packets per unit area is Poisson with rate $\eta \lambda D/E[R \cos \theta]$. Thus, offered
load to the network is roughly inversely proportional to the expected transmission
distance. However, third, as $E[R^2]$ gets larger, the number of terminals whose
transmissions could collide with a transmission increases, and thus the probability
of success of a transmission decreases. The expected number of terminals that
could potentially interfere with a transmission of radius $R'$ is $\lambda \pi R'^2$ and the
expected rate of successful transmissions within this area must be
\[ \lambda \pi R^2 \eta D/E[R \cos \theta] \] in order to support our network load. In addition to lowering the probability of success, as \( E[R^2] \) gets larger, the retransmission probability after an unsuccessful transmission, \( p \), must get smaller in order to maximize the network throughput during periods of peak loading. Finally as \( E[R] \) increases, we are more likely to find a station that is directly towards the destination thus decreasing \( E[Rd] \). This effect leads to less circuitous routes and less overall load in the network.

Because the loading in a packet radio network depends on our routing strategy (we can always find a routing strategy such that \( E[R^2] \) is high enough to make our probability of success low), we need to define our lightly loaded assumption in strategy-independent terms. Intuitively, since the expected number of nodes that can interfere with a transmission is \( \lambda \pi E[R^2] \), we would expect our network to behave approximately like an Aloha-type network [5] with arrival rate \( \lambda \pi E[R^2] \eta D/E[R \cos \theta] \). Since the maximum throughput of an Aloha-type network is 1/e and our network is lightly loaded, we want

\[ \lambda \pi E[R^2] \eta D/E[R \cos \theta] \ll 1/e. \]  

(1)

Additionally, with our light loading assumption, we assume there are many nodes that can potentially interfere with our transmissions, so

\[ \lambda \pi E[R^2] \gg 1. \]  

(2)

If we let \( E[R^2] = R^2 \) and \( E[R \cos \theta] = R \) and we solve for the values of our parameters where both equations 1 and 2 are true we find that our network is lightly loaded if

\[ \eta D \lambda^{1/2} \ll \pi^{1/2}/e. \]  

(3)

For the remainder of the thesis we assume that under our lightly loaded assumption equations 1, 2, and 3 are true.

The rest of this thesis is organized as follows. Chapter 2 develops the basis for an approximate delay analysis using our random access model. Chapter 3 is an approximate delay analysis for the very lightly loaded case where it is assumed that \( \lambda \pi E[R^2] \) is large enough that we can assume that the variation in the number
of terminals around the receiver and the transmission radius between different transmissions is insignificant. Chapter 4 develops an approximate delay analysis for more heavily loaded networks where the transmission radius and the number of terminals around the receiver varies. Chapter 5 applies this delay model to several transmission strategies. Chapter 6 summarizes the results and suggests several areas for further study.
2 Delay Model

In the lightly loaded case, collisions between more than two nodes are extremely rare and we can assume that whenever a node is backlogged at most one other node’s transmission collided with it. In this case, we can define $P_{STij}$ to be the probability of success on the first transmission from node $i$ to node $j$ and $P_{SRIj}$ to be the probability of success on each subsequent retransmission. With this assumption the expected service delay incurred by a packet going from node $i$ to node $j$ ($W_{ij}$) is 1 slot for the first transmission plus the probability that the first transmission fails times the expected wait when the node is backlogged. The probability that a node successfully transmits when backlogged is $P_{SRIj}$; thus the expected delay once backlogged is $1/(P_{SRIj})$. Thus, we obtain:

$$W_{ij} = 1 + (1 - P_{STij})/(p_i P_{SRIj}).$$

(4)

In this equation, and throughout, we ignore the expected 1/2 unit of extra delay incurred between the initial arrival of a packet in the network and the first subsequent slot boundary.

In most random-access networks (see for example [6] or [7]), if a transmission from node $A$ to node $B$ is unsuccessful due to a collision, then the colliding transmission from node $C$ to $E$ is also unsuccessful. However, in a packet radio network with perfect capture, the transmission from $C$ to $E$ can interfere with the transmission from $A$ to $B$ and still be successful. Figure 2-1 shows several possible node placements for which transmission $CE$ is successful even though it collides with transmission $AB$ and figure 2-2 shows regions where $E$ may be located such that $AB$ does not collide with $CE$. Thus, since $E$ is equally likely to be in any direction away from $C$ relative to $A$, we see that

$$Pr(AB \text{ collides with } CE \mid CE \text{ collides with } AB) < 1/2,$$

(5)

Thus, it is reasonable to make the assumption that if we pick the retransmission probabilities large enough, then

$$P_{STij} = P_{SRIj} = P_{Sij}$$

(6)

so that
\[ W_{ij} = 1 + (1 - P_{Sij})/(p_i P_{Sij}) \] \hspace{1cm} (7)

In order to determine the \( W_{ij} \)'s, we must now determine the retransmission probabilities, \( p_i \), and the steady-state success probabilities, \( P_{Sij} \).

![Diagram of networks A-B, A-B-E, and A-B-C-E](image)

**Case 1**

**Case 2**

**Case 3**

**Figure 2-1.** CE Successful even though it Collides with AB

Given a network topology and a transmission strategy that supports the network throughput, we can determine the link flows, \( f_{ij} \). We can then write \( L \) simultaneous equations for the steady-state success probabilities where \( L \) is the number of links for which \( f_{ij} \) is non-zero. First, define the set, \( S(i,j) \), to be the set of nodes \( k, k \neq j \), for which a transmission from \( k \) to any other node will collide with a transmission from \( i \) to \( j \). Then, using our independence assumption,

\[ P_{Sij} = (1 - \sum_{l} f_{jl}/P_{Sjl}) \prod_{k \in S(i,j)} (1 - \sum_{l} f_{kl}/P_{Skl}) \] \hspace{1cm} (8)

This set of equations may have multiple solutions. Since we want a set of probabilities for the steady-state case, a solution such that \( 0 \leq P_{Sij} \leq 1 \) for all \( i \) and \( j \). If no solution exists between zero and one, then the set of link flows given are too high to be supported by our network. This set of equations can be solved
Figure 2-2. Placement of $E$ so that $AB$ does not interfere with $CE$ iteratively to give the solution that we desire [5]. For link flows too high to be supported by the network the algorithm produces values for $P_{Si}$ that are greater than one or less than zero.

In order to clear backlogged nodes as fast as possible when the network has a run of bad luck and every node is backlogged, we would like to pick our retransmission probabilities such that throughput is maximized when all nodes are backlogged. Additionally, we require that the network be stable. If we use the independence assumption, then the network is stable as long as the throughput when all nodes are backlogged is greater than the average link flows. Thus, if it is
possible to support our link flows, then picking retransmission probabilities to maximize throughput when all nodes are backlogged will produce retransmission probabilities that make the network stable.

An extension of the algorithm outlined above can be used to determine these retransmission probabilities given the link flows. We would like to maximize the probability of successful transmission on all links when all the nodes are backlogged. To find this maximum, we want to determine the maximum packet handling capacity of the links when all nodes are backlogged. The maximum packet handling capacity is just the set of maximum link flows that could be supported by the network while keeping the relative link flows constant. We then set the retransmission probability for each node $i$ to be the probability that node $i$ would transmit if the network were supporting the maximum flow. Mathematically, we would like to maximize $\alpha$ under the constraint that $f_{ij}^\ast = \alpha f_{ij}$ for all $f_{ij}$. We can then calculate the probabilities of success if the network were supporting the maximum flow, $P_{Sij}^\ast$. We can then set the retransmission probabilities to be

$$p_i = \sum_j f_{ij}^\ast / P_{Sij}^\ast.$$  \hfill (9)

Thus if each node $i$ transmits with probability $p_i$ when backlogged, link packet handling capacity will be maximized when all nodes are backlogged and the backlog will be cleared quickly.

To get the $f_{ij}^\ast$'s, we start by getting an upper bound on the $f_{ij}^\ast$'s by making $f_{ij_{\text{upper}}}^\ast = 1$ for some $i$ and $j$ and scaling the rest of the $f_{ij_{\text{upper}}}^\ast$'s appropriately. We get an initial lower bound on the $f_{ij}^\ast$'s by making $f_{ij_{\text{lower}}}^\ast = 0$ for all $i$ and $j$. We can then do a binary search to find the set of maximum $f_{ij}^\ast$'s that will produce a stable network. Figure 2–3 show the retransmission probabilities and maximum flows which can be supported for a simple three node network.

Now given a specific network and a routing strategy which supports the given end-to-end packet flows we can calculate the steady-state success probabilities by iterating equation 8 and we can calculate the retransmission probabilities using the binary search algorithm and equation 9 given above. We can then calculate the service delay for each transmission by substituting into equation 7. Since we
\[ p_1 = 0.33 \quad \text{Maximum flow matrix:} \]
\[ p_2 = 0.40 \quad \begin{bmatrix} 0.0000 & 0.0934 & 0.0934 \\ 0.0934 & 0.0000 & 0.0934 \\ 0.0934 & 0.0934 & 0.0000 \end{bmatrix} \]
\[ p_3 = 0.46 \quad f_{ij}^{\text{max}} = 0.0934 \]

Note: Retransmission probabilities are not the same due to the asymmetry of the distances in the network.

**Figure 2-3. Retransmission Probabilities and Maximum Flows for a Simple Network**

are dealing primarily with lightly loaded networks, packets will very rarely be queued behind other packets at a terminal, and the transmission delay, $T$, is approximately equal to the service delay, $W$. Thus, once we compute the service delay on each link we can compute an approximation for the average velocity in the entire network. Appendix I.1 shows a computer program for doing these calculations for simple networks. The program takes as input the distance matrix, the end-to-end packet flow matrix, and the link flow matrix, and calculates the retransmission probabilities, the success probabilities, and the total average velocity.

Figure 2-4 shows a velocity comparison for a simple 10-node network using two different transmission strategies and two different values of $\eta$. In this network each node wants to transmit to each other node with the same end-to-end flow. Using transmission strategy 1, each packet arriving at a node is transmitted
directly to the final destination. Using transmission strategy 2, each node transmits to one of its nearest neighbors. The load is balanced to keep the relative link flows as close as possible. We see that for the smaller of the two values of $\eta$, strategy 1 results in a larger value for average velocity. For the larger of the two values of $\eta$, strategy 2 results in a larger value for average velocity. Thus, this example suggests that as our network load gets larger we want to decrease our average transmission radius to improve the average delay.

The above approach allows us to compare any two transmission strategies given a specific routing strategy. However, because this approach is iterative, we have no way of determining the performance of a general routing strategy (such as Most Forward Within R) given a random placement of nodes.
Figure 2-4. Velocity Comparison for a Simple 10-Node Network

Network Topology

Relative Flow for Routing Strategy 2

<table>
<thead>
<tr>
<th>$\eta$</th>
<th>Strategy</th>
<th>Velocity</th>
</tr>
</thead>
<tbody>
<tr>
<td>.0450</td>
<td>1</td>
<td>1.344</td>
</tr>
<tr>
<td>.0450</td>
<td>2</td>
<td>1.265</td>
</tr>
<tr>
<td>.0612</td>
<td>1</td>
<td>0.658</td>
</tr>
<tr>
<td>.0612</td>
<td>2</td>
<td>0.838</td>
</tr>
</tbody>
</table>

Each transmitter is equally likely to transmit each packet generated to any other node. In routing strategy 1, each node transmits each originating packet directly to its final destination. In routing strategy 2, the relative link flows are as shown in the diagram above.
3 Very Lightly Loaded Case

In the very lightly loaded case, where there are many terminals within the optimum transmission radius, we expect nodes to be able to transmit all packets approximately to the optimum place and we expect the number of terminals whose packets could possibly interfere with a transmission to be approximately constant. In this case the probability of success, $P_s$, the retransmission probability, $p$, the transmission radius, $R$, the number of terminals whose transmission could cause a collision, $N$, and the progress towards destination, $E[R\cos\theta]$ will be approximately constant. With these approximations, equation 8 becomes

$$P_s = (1 - \eta D/\eta P_s)^N, \quad (10)$$

and equation 7 becomes

$$W = 1 + (1 - P_s)/(pP_s) = 1 + 1/(pP_s) - 1/p. \quad (11)$$

Here $N$ is the average number of terminals (including the receiver) that are within $R$ of the receiver. Since we transmit to the most optimum node, $N$ is one (for the receiver) plus the expected number within $R$ of the receiver. But as shown in Figure 3-1 there is a small region of area $A$ in which there are no terminals since any node within this area would be "more optimum" than the actual receiver. Since the expected number of terminals within $R$ of the receiver is large, we can make the assumption that the area $A$ is totally included in the circle of radius $R$ around the receiver to get:

$$E[N|A] = \lambda (\pi R^2 - A) + 1. \quad (12)$$

Now $E[N] = E_A[E[N|A]]$, so

$$E[N] = \lambda \pi R^2 - \lambda E[A] + 1. \quad (13)$$

But $E[A]$ is just the average amount of area one would have to search to find the first terminal. Since our terminals are distributed according to a Poisson process, $E[A] = 1/\lambda$ independent of the shape of the region of area $A$. Thus we find that:

$$E[N] = \lambda \pi R^2. \quad (14)$$

Using our assumption that $N$ is large and does not vary much, we have:
\[ P_S = (1 - \frac{\eta D}{R P_S})^{\pi R^2 \lambda}. \] (15)

![Diagram](image)

**Figure 3-1.** Area where there are No Terminals

Like equation 8 in Chapter 2, this equation may have multiple real-valued solutions. In this case the equation for \( P_S \) has either zero, one, or two solutions between \( P_S = 0 \) and \( P_S = 1 \). In the case where it has zero solutions, the link flows cannot be supported for the given values of \( R, \eta, D, \) and \( \lambda \). In the case where it has one solution, the link flows can be supported but the network is extremely heavily loaded. In fact any increase in \( \eta \) or \( D \) would make the link flows too large to be supported. In the case where the equation has two solutions, the network flow can be supported by two values of \( P_S \), one close to 1 and one not so close to 1. Since we want our network to be stable, we will set our retransmission probability low enough so that our \( P_S \) is the higher of the two solutions to this equation that are between 0 and 1.
Now since \( \frac{(\pi D/(RP_S))\pi R^2\lambda}{1} \ll 1 \) and \( \pi R^2\lambda \gg 1 \) in our model, we can use the approximation \((1 - a)^b = 1 - ab\) for small \(a\) and \(ab\) in equation 15 to get

\[
P_S = 1 - \left(\frac{\eta D\pi R^2\lambda}{RP_S}\right) = 1 - \left(\frac{\eta D\pi R\lambda}{P_S}\right).
\]

(16)

Thus,

\[
P_S^2 - P_S + \eta D\pi R\lambda = 0.
\]

(17)

So,

\[
P_S = \frac{1}{2} \pm \frac{1}{2}(1 - 4\eta D\pi R\lambda)^{1/2}.
\]

(18)

The solution we want is the higher value of \(P_S\). Now, since \(4\eta D\pi R\lambda\) is small we can use the approximation \((1 - a)^{1/2} = 1 - (1/2)a\) for small \(a\), to get

\[
P_S = 1 - \eta D\pi R\lambda.
\]

(19)

Now the probability of successful transmission when all nodes are backlogged is:

\[
P_{\text{successful \ trans. when backlogged}} = pP_S^* = p(1 - p)^N.
\]

(20)

Picking \(p\) to maximize this quantity we find that

\[
p = \frac{1}{(N + 1)} = \frac{1}{(\pi R^2\lambda + 1)}.
\]

(21)

If we rewrite equation 11 in terms of \(P_F\) where \(P_F = 1 - P_S\) is the probability that a transmission fails, we have

\[
W = 1 + \frac{1}{p(1 - P_F)} - 1/p.
\]

(22)

Since \(P_F\) is small, we can use the approximation \(1/(1 - P_F) \approx 1 + P_F\) to get

\[
W = 1 + (1 + P_F)/p - 1/p \approx 1 + (1 - P_S)/p.
\]

(23)

Substituting the value for \(p\) given by equation 21 and the value for \(P_S\) given by equation 19 into our formula for delay given by equation 23 we get

\[
W \approx 1 + \pi^2\lambda^2\eta DR^3 + \pi\lambda \eta DR.
\]

(24)

Since \(E[R\cos\theta] = R\), we can divide the service delay, \(W\), by \(R\) to get an approximation for \(1/W\). We can then solve for the minimum with respect to \(R\) to determine the optimal transmission radius. Doing so, we find that
\[
1/V = (1/R) + \pi^2 \lambda^2 \eta DR^2 + \pi \lambda \eta D.
\]  
(25)

\[
\frac{d}{dR}(1/V) = (-1/R^2) + 2\pi^2 \lambda^2 \eta DR
\]  
(26)

and

\[
\frac{d^2}{dR^2}(1/V) = (2/R^3) + 2\pi^2 \lambda^2 \eta D > 0.
\]  
(27)

Setting our approximation for \(\frac{d}{dR}(1/V)\) to 0, we find a minima

\[
R = (1/(2\pi^2 \lambda^2 \eta D))^{1/3}.
\]  
(28)

Thus the optimum transmission radius for the case when \(R\) is large enough so that it is approximately constant is

\[
R_{opt} = (2\pi^2 \lambda^2 \eta D)^{-1/3}.
\]  
(29)

Now with \(R = R_{opt}\) our average velocity is:

\[
V = 1/((2\pi^2 \lambda^2 \eta D)^{1/3} + \pi^2 \lambda^2 \eta D(2\pi^2 \lambda^2 \eta D)^{-2/3} + \pi \lambda \eta D).
\]  
(30)

We can check some of our approximations using two variations to the iterative approach to finding the probability of success for specific values of \(\lambda, \eta, D,\) and \(R\). In the first method, we check our mathematical approximations by iteratively solving equation 15 for \(P_S\), substituting the result into equation 10 for delay, and dividing \(R\) by delay to get velocity. This result for velocity is compared to the result given by equation 25. In the second method, we check our assumption that \(P_S\) is the same for transmissions as it is for retransmissions by assuming that one other node is backlogged during retransmissions. Thus we have the following equations to solve iteratively:

\[
P_{ST} = (1 - \eta D/(RP_S))^N.
\]  
(31)

\[
P_{SR} = (1 - \eta D/(RP_S))^{N-1}(1 - p).
\]  
(32)

and

\[
P_S = P_{SR}/(P_{SR} + 1 - P_{ST}).
\]  
(33)

The last of these equations is obtained by taking the sum of the the probability of
success on first transmissions multiplied by the fraction of total transmissions that are first transmissions and the probability of success on retransmissions multiplied by the fraction of total transmissions that are retransmissions. Once we have solved equations 31, 32, and 33, we can substitute the resulting values of $P_{ST}$ and $P_{SR}$ into equation 4 to get the average delay. We then divide $R$ by the average delay to get velocity and compare it to the result given by equation 25.

Appendix I.2 shows a computer program that calculates velocity given some representative values of $\lambda$, $\eta$, and $D$, and Table 3-1 shows some sample output from this program. $V_1$ is the value of velocity using equation 30, $V_2$ is the value of velocity obtained by solving equation 15 iteratively, and $V_3$ is the value of velocity obtained by solving equations 31, 32, and 33 iteratively. The three cases considered are $N_{opt} = 177$, $N_{opt} = 38$, and $N_{opt} = 8$. The $N_{opt} = 8$ case violates our assumption of very light loading; however, it is shown here to indicate that our other assumptions are reasonable even in a more heavily loaded network, and thus are reasonable in the lightly loaded case.

We can see that the values we get for velocity using our approximate analysis are within 5% of those given by either of the two other methods for the cases considered. Furthermore, we see that our maxima's are fairly broad maxima's. Even when $R$ is 20% higher or lower than the optimum, velocity is within 5% of the optimum velocity. This suggests that for the lightly loaded case, the fact that we only approximately transmit to the optimum point on each transmission is not going to affect velocity very much.

Thus, for the very lightly loaded case, we conclude that our approximations are valid and the optimal transmission radius is:

$$R_{opt} = (2\pi^2\lambda^2\eta D)^{-1/3}.$$  \hspace{1cm} (34)

The velocity is:

$$V \approx 1/((2\pi^2\lambda^2\eta D)^{1/3} + \pi^2\lambda^2\eta D(2\pi^2\lambda^2\eta D)^{-2/3} + \pi\lambda\eta D).$$  \hspace{1cm} (35)

As will be seen later, this result for expected velocity also serves as an upper bound for the velocity in the more heavily loaded case.
Table 3-1. Comparison of Velocity for the Lightly Loaded Case

$V_1$ is velocity using equation 30.
$V_2$ is obtained by solving equation 15 iteratively.
$V_3$ is obtained by solving equations 31, 32, and 33 iteratively.

$nD = 0.0001200000, \lambda = 1.00$

<table>
<thead>
<tr>
<th>N</th>
<th>FLOW</th>
<th>R</th>
<th>$V_1$</th>
<th>$V_2$</th>
<th>$V_3$</th>
</tr>
</thead>
<tbody>
<tr>
<td>150.00</td>
<td>1.73664E-05</td>
<td>6.91</td>
<td>4.95</td>
<td>4.96</td>
<td>4.95</td>
</tr>
<tr>
<td>155.00</td>
<td>1.70840E-05</td>
<td>7.02</td>
<td>4.96</td>
<td>4.97</td>
<td>4.96</td>
</tr>
<tr>
<td>160.00</td>
<td>1.68150E-05</td>
<td>7.14</td>
<td>4.97</td>
<td>4.98</td>
<td>4.97</td>
</tr>
<tr>
<td>165.00</td>
<td>1.65582E-05</td>
<td>7.25</td>
<td>4.98</td>
<td>4.98</td>
<td>4.97</td>
</tr>
<tr>
<td>170.00</td>
<td>1.63129E-05</td>
<td>7.36</td>
<td>4.98</td>
<td>4.99</td>
<td>4.98</td>
</tr>
<tr>
<td>175.00</td>
<td>1.60782E-05</td>
<td>7.46</td>
<td>4.98</td>
<td>4.99</td>
<td>4.98</td>
</tr>
<tr>
<td>180.00</td>
<td>1.58533E-05</td>
<td>7.57</td>
<td>4.98</td>
<td>4.99</td>
<td>4.98</td>
</tr>
<tr>
<td>185.00</td>
<td>1.56376E-05</td>
<td>7.67</td>
<td>4.98</td>
<td>4.99</td>
<td>4.98</td>
</tr>
<tr>
<td>190.00</td>
<td>1.54305E-05</td>
<td>7.78</td>
<td>4.97</td>
<td>4.98</td>
<td>4.97</td>
</tr>
<tr>
<td>195.00</td>
<td>1.52314E-05</td>
<td>7.88</td>
<td>4.97</td>
<td>4.97</td>
<td>4.96</td>
</tr>
<tr>
<td>200.00</td>
<td>1.50398E-05</td>
<td>7.98</td>
<td>4.96</td>
<td>4.97</td>
<td>4.96</td>
</tr>
</tbody>
</table>

$nD = 0.001200000, \lambda = 1.00$

<table>
<thead>
<tr>
<th>N</th>
<th>FLOW</th>
<th>R</th>
<th>$V_1$</th>
<th>$V_2$</th>
<th>$V_3$</th>
</tr>
</thead>
<tbody>
<tr>
<td>20.00</td>
<td>4.75599E-04</td>
<td>2.52</td>
<td>2.10</td>
<td>2.10</td>
<td>2.08</td>
</tr>
<tr>
<td>24.00</td>
<td>4.34161E-04</td>
<td>2.76</td>
<td>2.18</td>
<td>2.19</td>
<td>2.17</td>
</tr>
<tr>
<td>28.00</td>
<td>4.01955E-04</td>
<td>2.99</td>
<td>2.24</td>
<td>2.25</td>
<td>2.23</td>
</tr>
<tr>
<td>32.00</td>
<td>3.75994E-04</td>
<td>3.19</td>
<td>2.27</td>
<td>2.28</td>
<td>2.26</td>
</tr>
<tr>
<td>36.00</td>
<td>3.54491E-04</td>
<td>3.39</td>
<td>2.28</td>
<td>2.29</td>
<td>2.27</td>
</tr>
<tr>
<td>40.00</td>
<td>3.36299E-04</td>
<td>3.57</td>
<td>2.28</td>
<td>2.29</td>
<td>2.27</td>
</tr>
<tr>
<td>44.00</td>
<td>3.20649E-04</td>
<td>3.74</td>
<td>2.26</td>
<td>2.28</td>
<td>2.26</td>
</tr>
<tr>
<td>48.00</td>
<td>3.06998E-04</td>
<td>3.91</td>
<td>2.24</td>
<td>2.26</td>
<td>2.24</td>
</tr>
<tr>
<td>52.00</td>
<td>2.94954E-04</td>
<td>4.07</td>
<td>2.21</td>
<td>2.24</td>
<td>2.22</td>
</tr>
<tr>
<td>56.00</td>
<td>2.84225E-04</td>
<td>4.22</td>
<td>2.18</td>
<td>2.21</td>
<td>2.19</td>
</tr>
<tr>
<td>60.00</td>
<td>2.74587E-04</td>
<td>4.37</td>
<td>2.14</td>
<td>2.17</td>
<td>2.15</td>
</tr>
</tbody>
</table>

$nD = 0.012466800, \lambda = 1.00$

<table>
<thead>
<tr>
<th>N</th>
<th>FLOW</th>
<th>R</th>
<th>$V_1$</th>
<th>$V_2$</th>
<th>$V_3$</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.00</td>
<td>1.10484E-02</td>
<td>1.13</td>
<td>0.91</td>
<td>0.92</td>
<td>0.88</td>
</tr>
<tr>
<td>4.80</td>
<td>1.00858E-02</td>
<td>1.24</td>
<td>0.94</td>
<td>0.96</td>
<td>0.92</td>
</tr>
<tr>
<td>5.60</td>
<td>9.33761E-03</td>
<td>1.34</td>
<td>0.96</td>
<td>0.98</td>
<td>0.94</td>
</tr>
<tr>
<td>6.40</td>
<td>8.73454E-03</td>
<td>1.43</td>
<td>0.97</td>
<td>1.00</td>
<td>0.96</td>
</tr>
<tr>
<td>7.20</td>
<td>8.23500E-03</td>
<td>1.51</td>
<td>0.97</td>
<td>1.01</td>
<td>0.96</td>
</tr>
<tr>
<td>8.00</td>
<td>7.81241E-03</td>
<td>1.60</td>
<td>0.97</td>
<td>1.01</td>
<td>0.96</td>
</tr>
<tr>
<td>8.80</td>
<td>7.44884E-03</td>
<td>1.67</td>
<td>0.96</td>
<td>1.00</td>
<td>0.96</td>
</tr>
<tr>
<td>9.60</td>
<td>7.13172E-03</td>
<td>1.75</td>
<td>0.95</td>
<td>1.00</td>
<td>0.95</td>
</tr>
<tr>
<td>10.40</td>
<td>6.85193E-03</td>
<td>1.82</td>
<td>0.93</td>
<td>0.99</td>
<td>0.94</td>
</tr>
<tr>
<td>11.20</td>
<td>6.60269E-03</td>
<td>1.89</td>
<td>0.91</td>
<td>0.97</td>
<td>0.93</td>
</tr>
<tr>
<td>12.00</td>
<td>6.37880E-03</td>
<td>1.95</td>
<td>0.89</td>
<td>0.96</td>
<td>0.92</td>
</tr>
</tbody>
</table>
4 More Heavily Loaded Case

As we increase $\eta$, the optimal transmission radius gets smaller. As $E[R]$ gets smaller the relative variance of $R$ ($\text{var}(R/R_{\text{opt}})$) and the variance of $\theta$ increases. Thus, we wish to relax our assumption that $R$ and $N$ are approximately constant and find a method of determining the expected value of the velocity given the joint probability distribution of $R$ and $\theta$. Then, for any routing rule we can in theory calculate the joint distribution of $R$ and $\theta$ and determine the average velocity. Various different routing strategies can be compared with each other and with the upper bound obtained using the results in Chapter 3. Since the maxima obtained in Chapter 3 were fairly broad, we expect to be able to find a routing strategy that performs fairly well as compared with the upper bound.

We need to determine values for the retransmission probability and for the probability of success. The best way to determine the retransmission probability for any given network and routing strategy is to go through the procedure discussed in Chapter 2. However, in any real packet radio network, nodes are going to be moving and/or entering and leaving the network, so it will be difficult to maintain a carefully controlled retransmission probability for each individual terminal [5]. Thus, it seems reasonable to assign the same retransmission probability to all nodes. The value we choose is

$$p = 1/E[N]=1/(\lambda \pi E[R^2])$$

since this is the value that would maximize throughput under fully backlogged conditions if our terminal distribution were uniform rather than Poisson. This value can produce some instability in sections of the network where there is an extremely high concentration of nodes; however, these places are extremely rare. The instability in these sections can be handled by adjusting the retransmission probability; however, since these sections of the network are rare, the adjusted retransmission probability will not affect average velocity very much.

Next we assume that the probability that each node around a given receiver transmits in a given slot, $P_{Tr}$, is independent of the number of nodes around the receiver. Figure 4-1 shows a typical transmitter (node A), receiver (node B), and
another node that could interfere with a transmission from A to B (node C). The probability that C will transmit in a given slot depends on the number of nodes around C. Clearly, some of the nodes around node C are not within the transmission radius of B leading to some independence. Additionally, as the number of nodes around C increases, the probability of success decreases, but also the number of packets routed through node C decreases; thus it seems plausible to assume that the probability of transmission stays approximately the same.

Figure 4-1. Nodes around the Receiver

Using this assumption, the probability of success, given the transmission radius \( R \), is \( P_{S|R} = (1 - E[P_T])^N \), where \( P_T \) is the probability a terminal transmits in a given slot and \( N \) is the number of terminals that can interfere with the transmission. Now since the probability that a terminal transmits is small, we can use the approximation \( (1 - a)^b \approx 1 - ab \) for \( a \) and \( ab \) small to get

\[
P_{S|R} \approx 1 - E[P_T](N)
\] (37)

and
\[ E[P_{SR}] = 1 - E[P_T]E[N|R] = 1 - E[P_T]\lambda \pi R^2. \] (38)

Now, \( E[P_T] = \text{(transmissions per node per slot)}/E[P_S] = \eta D/E[R \cos \theta]E[P_S] \) so,
\[ E[P_{SR}] = 1 - \lambda \pi R^2 \eta D/(E[R \cos \theta]E[P_S]). \] (39)

But \( E[P_S] = E_R[E[P_{SR}]], \) so
\[ E[P_S] = 1 - \lambda \pi E[R^2] \eta D/(E[R \cos \theta]E[P_S]) \] (40)

or
\[ \{E[P_S]\}^2 - \{E[P_S]\} + \lambda \pi E[R^2] \eta D/E[R \cos \theta] = 0. \] (41)

Solving this we find that
\[ E[P_S] = (1/2) \pm (1/2)(1 - 4\lambda \pi E[R^2] \eta D/E[R \cos \theta])^{1/2}. \] (42)

Once again, since our retransmission probability is set to keep the expected probability of success high, we are only interested in the larger of the two solutions. Now since \( 4\lambda \pi E[R^2] \eta D/E[R \cos \theta] \) is relatively small we can use the approximation \((1 - a)^{1/2} = 1 - (1/2)a\) to get
\[ E[P_S] = 1 - \lambda \pi E[R^2] \eta D/E[R \cos \theta]. \] (43)

Now we can substitute our results for \( p \) given by equation 36 and \( P_S \) given by equation 43 into equation 23 for delay and \((1/V)\). We find that
\[ E[W] = 1 + \lambda \pi E[R^2]\{(\lambda \pi E[R^2] + 1) \eta D/E[R \cos \theta]. \] (44)

Dividing this result by the expected progress towards the destination, we get
\[ E[W]/E[R \cos \theta] = 1/E[R \cos \theta] + \lambda \pi E[R^2]\{(\lambda \pi E[R^2] + 1) \eta D/E[R \cos \theta]^2 \] (45)

and
\[ V = \frac{1}{\{1/E[R \cos \theta] + \lambda \pi E[R^2]\{(\lambda \pi E[R^2] + 1) \eta D/E[R \cos \theta]^2 \}. \] (46)

We see that if we let \( E[R^2] = R^2 \) and \( E[R \cos \theta] = R \) then this result reduces to the result given by equation 25 in Chapter 3.

This formula gives the average velocity in terms of the expected forward
progress of successful transmissions (E[Rcosθ]) and the second moment of the
transmission radius (E[R^2]). If we rewrite E[R^2] as E[R]^2 + var[R] in equation 45 we have

\[ 1/V = 1/E[Rcosθ] + λπ(E[R]^2 + var[R])λπ(E[R]^2 + var[R] + 1)ηD/E[Rcosθ]. \]

Rearranging the terms we find,

\[ 1/V = 1/E[Rcosθ] + λπE[R]^2(λπE[R]^2 + 1)ηD/E[Rcosθ]^2 + λπvar[R]λπ(2E[R]^2 + var[R] + 1)ηD/E[Rcosθ]. \]

We see that given any value of E[R], 1/V is smallest if var[R] is zero and E[Rcosθ] = E[R] (since var[R] > 0 and E[Rcosθ] < E[R]). Thus, an upper bound on velocity is obtained by setting E[Rcosθ] = R and E[R^2] = R^2 and optimizing with respect to R. When these values are substituted into equation 46, the result reduces to the velocity obtained for the very lightly loaded case given by equation 35. Thus, equation 35 gives an upper bound on the velocity in a network.

Now given a transmission strategy, we can in theory calculate E[R^2] and E[Rcosθ] using our Poisson terminal distribution model. We can then compute the approximate average velocity and compare it with other transmission strategies and the upper bound given by the result developed in Chapter 3.
5 Comparison of Transmission Strategies

5.1 Closest to Optimum

One possible transmission strategy is to transmit to the closest terminal to the light loading optimum point of \( R = R_{\text{opt}}, \theta = 0 \), where \( R_{\text{opt}} \) is given by equation 29. Figure 5-1 shows a sample transmission using this transmission strategy. Now given \( R_d \), we are equally likely to transmit anywhere around the circle of radius \( R_d \) centered at the optimum point. Thus \( E[R\cos\theta|R_d] = R_{\text{opt}} \), so

\[
E[R\cos\theta] = E_{R_d}[E[R\cos\theta|R_d]] = R_{\text{opt}}
\]  

(47)

In order to get \( E[R^2] \), we use the law of cosines to give us

\[
R^2 = R_{\text{opt}}^2 + R_d^2 - 2R_{\text{opt}}R_d\cos\phi
\]  

(48)

so that \( E[R^2] \) is

\[
E[R^2] = R_{\text{opt}}^2 + E[R_d^2] - E[2R_{\text{opt}}R_d\cos\phi].
\]  

(49)

Now clearly \( E[\cos\phi] \) is 0 and \( \phi \) and \( R_d \) are independent so \( E[2R_{\text{opt}}R_d\cos\phi] = E[2R_{\text{opt}}R_d]E[\cos\phi] = 0 \). Thus to determine \( E[R^2] \) we need to solve for \( E[R_d^2] \) which is just the expected distance from the optimal transmission point to the closest terminal to the optimal transmission point.

If we let \( A \) be the area within \( r \) of the optimal transmission point then we have a probability density in terms of \( A \) given by our Poisson terminal distribution:

\[
p_{\text{closest terminal}}(A) = \lambda e^{-\lambda A}.
\]  

(50)

Now since \( A = \pi r^2 \) and \( dA = 2\pi rdr \), the second moment of \( R_d \) is given by:

\[
E[R_d^2] = \int_0^\infty r^3 e^{-\lambda \pi r^2} 2\pi dr = 1/(\lambda \pi).
\]  

(51)

Thus we have:

\[
E[R^2] = R_{\text{opt}}^2 + 1/(\lambda \pi).
\]  

(52)
Figure 5-1. Closest to Optimum Transmission

Table 5-1 shows the delay for this strategy compared with upper bound given by equation 35 and delay for the most forward within $R$ strategy which is discussed next. Figure 5-2 shows a graph of velocity vs. $\eta D$ for these transmission strategies.

5.2 Most Forward within $R_p$

Using the most forward within $R_p$ strategy, a terminal transmits to the terminal which is most forward within $R_p$ where $R_p$ is a parameter which is set to optimize performance, in this case delay. Figure 5-3 shows a sample transmission using this transmission strategy. To solve for $E[R \cos \theta]$ we first solve for the area $A$ as a function of $R \cos \theta$. Letting $x = R \cos \theta$, we have

$$A = R_p^2 \left( \cos^{-1} \left( \frac{x}{R_p} \right) - \frac{x}{R_p} \right) \left( 1 - \left( \frac{x}{R_p} \right)^2 \right)^{1/2}. \quad (53)$$

Now,

$$E[x] = \int_0^{R_p} x \lambda e^{-\lambda x} \frac{dA}{dx} dx \quad (54)$$

so.
Table 5-1. Comparison of Transmission Strategies

- $V_{\text{OPT}}$ = velocity for upper bound on obtained using equation 30.
- $V_{\text{CLOSEST}}$ = velocity for the closest to optimum strategy.
- $V_{\text{MFR}}$ = velocity for the Most Forward within R_P strategy.

<table>
<thead>
<tr>
<th>$\eta_D$</th>
<th>$R_{\text{OPT}}$</th>
<th>$M_{\text{OPT}}$</th>
<th>$R_P$</th>
<th>$V_{\text{OPT}}$</th>
<th>$V_{\text{CLOSEST}}$</th>
<th>$V_{\text{MFR}}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.0252313</td>
<td>1.26157</td>
<td>5.00</td>
<td>1.75499</td>
<td>0.7885</td>
<td>0.6857</td>
<td>0.6030</td>
</tr>
<tr>
<td>0.0137342</td>
<td>1.54510</td>
<td>7.50</td>
<td>2.00682</td>
<td>0.9862</td>
<td>0.8995</td>
<td>0.8123</td>
</tr>
<tr>
<td>0.0089206</td>
<td>1.78412</td>
<td>10.00</td>
<td>2.21796</td>
<td>1.1510</td>
<td>1.0748</td>
<td>0.9859</td>
</tr>
<tr>
<td>0.0063831</td>
<td>1.99471</td>
<td>12.50</td>
<td>2.40768</td>
<td>1.2953</td>
<td>1.2265</td>
<td>1.1365</td>
</tr>
<tr>
<td>0.0048558</td>
<td>2.18510</td>
<td>15.00</td>
<td>2.57987</td>
<td>1.4251</td>
<td>1.3619</td>
<td>1.2711</td>
</tr>
<tr>
<td>0.0038533</td>
<td>2.36017</td>
<td>17.50</td>
<td>2.74221</td>
<td>1.5440</td>
<td>1.4854</td>
<td>1.3937</td>
</tr>
<tr>
<td>0.0031539</td>
<td>2.52313</td>
<td>20.00</td>
<td>2.89396</td>
<td>1.6545</td>
<td>1.5995</td>
<td>1.5070</td>
</tr>
<tr>
<td>0.0026431</td>
<td>2.67619</td>
<td>22.50</td>
<td>3.03684</td>
<td>1.7581</td>
<td>1.7061</td>
<td>1.6127</td>
</tr>
<tr>
<td>0.0022568</td>
<td>2.82095</td>
<td>25.00</td>
<td>3.17357</td>
<td>1.8559</td>
<td>1.8065</td>
<td>1.7121</td>
</tr>
<tr>
<td>0.0019561</td>
<td>2.95864</td>
<td>27.50</td>
<td>3.29957</td>
<td>1.9488</td>
<td>1.9016</td>
<td>1.8062</td>
</tr>
<tr>
<td>0.0017168</td>
<td>3.09019</td>
<td>30.00</td>
<td>3.39197</td>
<td>2.0375</td>
<td>1.9923</td>
<td>1.8955</td>
</tr>
<tr>
<td>0.0015225</td>
<td>3.21638</td>
<td>32.50</td>
<td>3.54304</td>
<td>2.1225</td>
<td>2.0790</td>
<td>1.9813</td>
</tr>
<tr>
<td>0.0013624</td>
<td>3.33779</td>
<td>35.00</td>
<td>3.62463</td>
<td>2.2042</td>
<td>2.1623</td>
<td>2.0633</td>
</tr>
<tr>
<td>0.0012284</td>
<td>3.45494</td>
<td>37.50</td>
<td>3.77884</td>
<td>2.2830</td>
<td>2.2424</td>
<td>2.1423</td>
</tr>
<tr>
<td>0.0011151</td>
<td>3.56825</td>
<td>40.00</td>
<td>3.84702</td>
<td>2.3592</td>
<td>2.3199</td>
<td>2.2183</td>
</tr>
<tr>
<td>0.0010181</td>
<td>3.67807</td>
<td>42.50</td>
<td>3.98517</td>
<td>2.4330</td>
<td>2.3948</td>
<td>2.2920</td>
</tr>
<tr>
<td>0.0009345</td>
<td>3.78470</td>
<td>45.00</td>
<td>4.08038</td>
<td>2.5046</td>
<td>2.4675</td>
<td>2.3633</td>
</tr>
<tr>
<td>0.0008617</td>
<td>3.88841</td>
<td>47.50</td>
<td>4.18080</td>
<td>2.5742</td>
<td>2.5381</td>
<td>2.4324</td>
</tr>
<tr>
<td>0.0007979</td>
<td>3.98942</td>
<td>50.00</td>
<td>4.27967</td>
<td>2.6420</td>
<td>2.6068</td>
<td>2.4996</td>
</tr>
<tr>
<td>0.0007416</td>
<td>4.08794</td>
<td>52.50</td>
<td>4.37138</td>
<td>2.7081</td>
<td>2.6737</td>
<td>2.5650</td>
</tr>
<tr>
<td>0.0006916</td>
<td>4.18414</td>
<td>55.00</td>
<td>4.46557</td>
<td>2.7726</td>
<td>2.7390</td>
<td>2.6286</td>
</tr>
<tr>
<td>0.0006470</td>
<td>4.27818</td>
<td>57.50</td>
<td>4.55183</td>
<td>2.8357</td>
<td>2.8028</td>
<td>2.6908</td>
</tr>
<tr>
<td>0.0006070</td>
<td>4.37019</td>
<td>60.00</td>
<td>4.64120</td>
<td>2.8974</td>
<td>2.8652</td>
<td>2.7514</td>
</tr>
<tr>
<td>0.0005709</td>
<td>4.46031</td>
<td>62.50</td>
<td>4.72601</td>
<td>2.9578</td>
<td>2.9262</td>
<td>2.8107</td>
</tr>
<tr>
<td>0.0005383</td>
<td>4.54864</td>
<td>65.00</td>
<td>4.81072</td>
<td>3.0170</td>
<td>2.9860</td>
<td>2.8687</td>
</tr>
<tr>
<td>0.0005087</td>
<td>4.63529</td>
<td>67.50</td>
<td>4.89105</td>
<td>3.0750</td>
<td>3.0447</td>
<td>2.9255</td>
</tr>
<tr>
<td>0.0004817</td>
<td>4.72035</td>
<td>70.00</td>
<td>4.96927</td>
<td>3.1320</td>
<td>3.1022</td>
<td>2.9812</td>
</tr>
<tr>
<td>0.0004570</td>
<td>4.80390</td>
<td>72.50</td>
<td>5.04316</td>
<td>3.1879</td>
<td>3.1586</td>
<td>3.0357</td>
</tr>
<tr>
<td>0.0004343</td>
<td>4.88603</td>
<td>75.00</td>
<td>5.12460</td>
<td>3.2429</td>
<td>3.2141</td>
<td>3.0893</td>
</tr>
<tr>
<td>0.0004135</td>
<td>4.96679</td>
<td>77.50</td>
<td>5.20446</td>
<td>3.2970</td>
<td>3.2687</td>
<td>3.1418</td>
</tr>
<tr>
<td>0.0003942</td>
<td>5.04627</td>
<td>80.00</td>
<td>5.27788</td>
<td>3.3502</td>
<td>3.3223</td>
<td>3.1935</td>
</tr>
<tr>
<td>0.0003765</td>
<td>5.12451</td>
<td>82.50</td>
<td>5.35221</td>
<td>3.4026</td>
<td>3.3751</td>
<td>3.2442</td>
</tr>
<tr>
<td>0.0003600</td>
<td>5.20157</td>
<td>85.00</td>
<td>5.42254</td>
<td>3.4542</td>
<td>3.4271</td>
<td>3.2941</td>
</tr>
<tr>
<td>0.0003447</td>
<td>5.27751</td>
<td>87.50</td>
<td>5.49913</td>
<td>3.5050</td>
<td>3.4783</td>
<td>3.3432</td>
</tr>
<tr>
<td>0.0003304</td>
<td>5.35237</td>
<td>90.00</td>
<td>5.56668</td>
<td>3.5551</td>
<td>3.5288</td>
<td>3.3915</td>
</tr>
<tr>
<td>0.0003171</td>
<td>5.42620</td>
<td>92.50</td>
<td>5.63816</td>
<td>3.6045</td>
<td>3.5785</td>
<td>3.4391</td>
</tr>
<tr>
<td>0.0003047</td>
<td>5.49904</td>
<td>95.00</td>
<td>5.70579</td>
<td>3.6532</td>
<td>3.6276</td>
<td>3.4859</td>
</tr>
<tr>
<td>0.0002930</td>
<td>5.57093</td>
<td>97.50</td>
<td>5.77358</td>
<td>3.7013</td>
<td>3.6760</td>
<td>3.5321</td>
</tr>
<tr>
<td>0.0002821</td>
<td>5.64190</td>
<td>100.00</td>
<td>5.84025</td>
<td>3.7488</td>
<td>3.7238</td>
<td>3.5776</td>
</tr>
</tbody>
</table>
Figure 5-2. Velocity vs. $\eta D$ for the 3 Transmission Strategies

Graph of Velocity vs. $\eta D$ for the Closest to Optimum and Most Forward within $R_p$ Transmission Strategies and for the Upper Bound Obtained by Assuming all Transmissions are to the optimum Point.

Closest to Optimum: +
Most Forward within $R_p$: 0
Upper Bound: X
\[
E[x] = \int_0^R x \lambda e^{-\lambda R_p^2 \left(\cos^{-1}(x/R_p) - (x/R_p)(1 - (x/R_p)^2)^{1/2}\right)}(2/R_p)(1 - (x/R_p)^2)^{1/2}dx.
\]

This does not look like it is easily solved directly, however, it can be solved using numerical integration.

![Diagram of transmitter and receiver](Image)

**Figure 5-3. Most Forward Within R_p Transmission**

To solve for \(E[R^2]\) we use iterated expectation, i.e.

\[
E[R^2] = E_x[E[R^2|x]].
\]

(55)

Now since \(R^2 = x^2 + y^2\),

\[
E[R^2|x] = E[x^2|x] + E[y^2|x] = x^2 + E[y^2|x],
\]

(56)

so we need to solve for \(E[y^2|x]\). Given \(x, y\) is equally likely to be anywhere from 0 to its maximum value which is \(y_{max} = (R_p^2 - x^2)^{1/2}\), so

\[
E[y^2] = \int_{(R_p^2 - x^2)^{1/2}}^{(R_p^2 - x^2)^{1/2}} z^2/(2(R_p^2 - x^2)^{1/2})dz = (R_p^2 - x^2)/3
\]

(57)

and

\[
E[R^2|x] = (2x^2 + R_p^2)/3.
\]

(58)

Thus we find:
\[ E[R^2] = (2E[x^2] + R_p^2)/3 \] (59)

where the expected value of \( x^2 \) is given by

\[ E[x^2] = \int_0^R x^2 \lambda e^{-\lambda R_p^2 \cos^{-1}(\kappa/R_p)} - (\kappa/R_p)(1 - (\kappa/R_p)^2)^{1/2}(2/R_p)(1 - (\kappa/R_p)^2)^{1/2}dx. \]

Thus in order to find \( E[x] \) and \( E[R^2] \) we have to numerically evaluate two integrals. Appendix I.3 shows a program which calculates the two integrals for values of \( R_p \) around \( R_{opt} \) and determines the value of \( R_p \) that maximizes the velocity. Additionally, this program computes the upper bound for velocity obtained using the results of Chapter 4 and the velocity using the closest to optimum strategy. Table 5-1 shows the results from this computer program for various values of \( \eta D \) with \( \lambda \) held constant and Figure 5-2 shows a graph of velocity vs. \( \eta D \).

As we see in Table 5-1, the closest to optimum strategy performs better than the most forward within \( R_p \) over all values \( \eta D \). Closest to optimum is within 10% of the upper bound for all values of \( \eta D \) such that \( N_{opt} \) is greater than 7.5. As \( \eta D \) gets smaller, the performance of the closest to optimum strategy approaches the upper bound fairly quickly. The velocity for the most forward with \( R_p \) strategy, on the other hand, is an almost constant amount less than the upper bound as \( \eta D \) gets smaller.

These results suggest that closest to optimum is a better transmission strategy than most forward within \( R_p \). This seems reasonable since when using most forward within \( R_p \), \( R_p \) must be higher than \( R_{opt} \) in order to minimize delay. Thus even if there is a terminal exactly at the optimum point, the transmitter may transmit elsewhere if there is a terminal more forward with \( R_p \).
6 Conclusion

This thesis has developed methods of approximating the delay in random access packet radio networks with capture. First we developed a method of determining the optimum retransmission probabilities, the probability of success, and the average velocity of packets given a specific network topology and the link flows required to support the required network throughput. We found that while this method was the best way to compare specific transmission strategies given a specific network configuration, it did not allow us to compare general transmission strategies given a random network configuration. Next, we assumed that our network was so lightly loaded that there were many nodes within the optimum transmission radius, and thus all transmissions were approximately to the optimum place. Using this assumption, we developed a formula for the optimal transmission radius and the velocity of packets when all transmissions are of this optimal radius. This formula for velocity serves as an upper bound for velocity in more heavily loaded networks. Next we considered the more heavily loaded network case in which we assumed that the transmission radius varied. We then developed a formula for velocity which depended on the expected forward progress of packets (E[Rcosθ]) and the second moment of the transmission radius (E[R^2]). Finally we used this formula for velocity to compare two transmission strategies, closest to optimum and most forward within R, with each other and the upper bound obtained by assuming all transmissions go to the optimum place. We found that closest to optimum performed better than most forward within R.

Several areas of future study are suggested by this work. One possibility would be to simulate a packet radio network to verify the validity of the assumptions made in this approximate analysis. Another possibility would be to extend the results to the case of imperfect capture, i.e. where a transmission is successful only if no other node within αR of the receiver transmits where α is a constant greater than 1. The math presented in this work is easily extended; however, some of the assumptions made in this paper may not be reasonable if α is much greater than 1. In particular, with imperfect capture, equation 5 is not true, and thus it is more likely that if node A's transmission was unsuccessful because C transmitted, node C's transmission was also unsuccessful. Third, methods of
changing the transmission radius depending on network load need to be studied. Such methods could be fixed (such as using a different transmission radius at night than during the day) or dynamic (such as sensing the load in the network, estimating the load, and updating the transmission radius accordingly). Finally, improved methods for picking the retransmission probabilities need to be studied. If we look at equation 46, our equation for velocity, we see it has two terms, the first due to the one-slot delay when the packet is being transmitted and the second due to the delay waiting for a successful retransmission. In the latter term, we have a factor roughly proportional to \( E[R^2] \) due to collisions on each transmission and a factor roughly proportional to \( E[R^2] \) due to waiting to retransmit. Thus, if the retransmission probability could be made independent of \( E[R^2] \), the result we get for optimum transmission radius would be larger. There are algorithms for scheduling retransmissions for single-hop networks in which the time to resolve collisions is a function of the number of nodes involved in the collision rather than the number of nodes in the network [8]; however, they are not easily extendable to packet radio with capture since they assume that when a collision occurs both terminals are backlogged.
I Computer Programs

I.1 Program to Find Retransmission Probabilities and Velocity Given a Specific Network and Flows to Support the Network Load

(* This program takes as input a distance matrix, an end-to-end packet flow matrix, and a link flow matrix and produces as output the maximum link flow with relative link flow constant matrix, the maximum total throughput value, the retransmission probabilities, the success probability matrix, and the total average velocity. It does so by first using a binary search to scale up the relative flows to find the maximum flows. It uses these maximum flows to compute the maximum total throughput and the retransmission probabilities. It then calculates the success probabilities and from these calculates average delay on each link. From the delay matrix it calculates the average total velocity. *)

Program FIND_P_AND_DELAY ( INPUT, OUTPUT, IN_FILE, OUT_FILE );

type
   MATRX = array[0..10] of array[0..10] of real;
   VCTR = array[0..10] of real;

var
   IN_FILE: Text;
   OUT_FILE: Text;
   FILE_NAME: Varying[ 132 ] of char;
   IN_FILE_NAME: Varying[ 132 ] of char;
   DISTANCE: MATRX;
   ATTEMPT_RATE: MATRX;
   SUCCESS_PROB: MATRX;
   STOP, N, I, J, K: Integer;
   ERROR: real;
   HI_FLOW: MATRX;
   LO_FLOW: Matrix;
   MID_FLOW: MATRX;
   FLOW: MATRX;
   END_TO_END_FLOW: MATRX;
   SCALE_VALUE: Real;
   RETRANS_PROB: VCTR;
   TOTAL_TIME: Real;
TOTAL DISTANCE: Real;
VELOCITY: Real;
MAX_TOTAL_THROUGHPUT: Real;
END_TO_END_THROUGHPUT: Real;
LINK_THROUGHPUT: Real;

(* The FIND_TRANS_PROB routine iteratively calculates the *)
(* steady-state probability of success on each link in the *)
(* network given the flow and distance matrix. If the flow cannot *)
(* be supported by the network then it sets the global variable *)
(* STOP to 1. Otherwise it up the ATTEMPT_RATE matrix. *)
(*
])(

Procedure FIND_TRANS_PROB( var N: Integer;
                           DISTANCE: matrix;
                           FLOW: matrix);

var
  I, J, K: Integer;
  ERROR: REAL;
  TOTAL_TRANS_PROB: VECTOR;
  NEW_ATTEMPT_RATE: real;

begin

(* Initialize variables. *)

  For I := 0 to N-1 do
    For J := 0 to N-1 do
      ATTEMPT_RATE[ I, J ] := 0;
  For I := 0 to N-1 do
    For J := 0 to N-1 do
      SUCCESS_PROB[ I, J ] := 0;
  ERROR := 1;
  STOP := 0;

  while (* we haven't yet converged. *)
  (* ( ERROR > 0.0000001 ) and ( STOP = 0 ) )do
begin

  (* Calculate the total transmission *)
  (* probabilities for each node. *)
  (*

  The following code calculates the steady-state probability of success on each link in the network given the flow and distance matrix. If the flow cannot be supported by the network, it sets the global variable STOP to 1. Otherwise, it updates the ATTEMPT_RATE matrix.

  The procedure begins by initializing the ATTEMPT_RATE and SUCCESS_PROB matrices to zero. It then enters a loop that continues until convergence is achieved, as indicated by the condition ERROR > 0.0000001 and STOP = 0. Within the loop, the total transmission probabilities are calculated for each node.
for I := 0 to N-1 do
begin
  TOTAL_TRANS_PROB[ I ] := 0;
  for J := 0 to N-1 do
    TOTAL_TRANS_PROB[ I ] :=
      TOTAL_TRANS_PROB[ I ] + ATTEMPT_RATE[ I, J ];
  if ( TOTAL_TRANS_PROB[ I ] >= 1 ) then
begin
    STOP := 1;
    TOTAL_TRANS_PROB[ I ] := 0.9;
end;
end;

(* Calculate the success probabilities on each link. *)

for I := 0 to N-1 do
begin
  for J := 0 to N-1 do
  begin
    if FLOW[ I, J ] <> 0 then
      SUCCESS_PROB[ I, J ] := 1 - TOTAL_TRANS_PROB[ J ];
    for K := 0 to N-1 do
    begin
      if ( DISTANCE[ K, J ] < DISTANCE[ I, J ] )
      and ( K <> J ) then
        SUCCESS_PROB[ I, J ] := SUCCESS_PROB[ I, J ]
        * ( 1 - TOTAL_TRANS_PROB[ K ] );
      if ( DISTANCE[ K, J ] = DISTANCE[ I, J ] )
      and ( K <> I ) then
        SUCCESS_PROB[ I, J ] := SUCCESS_PROB[ I, J ]
        * ( 1 - TOTAL_TRANS_PROB[ K ]/2 );
    end;
  end;
end;

(* Update the attempt rate based on the flow *)
(* and the new success probabilities. *)

ERROR := 0;
for I := 0 to N-1 do
begin
  for J := 0 to N-1 do
  begin
    if FLOW[ I, J ] <> 0 then
      NEW_ATTEMPT_RATE := FLOW[ I, J ]/
      SUCCESS_PROB[ I, J ];
    else NEW_ATTEMPT_RATE := 0;
    if (NEW_ATTEMPT_RATE > 1) or (NEW_ATTEMPT_RATE < 0)
then STOP := 1;
ERROR := max( ERROR,
    NEW_ATTEMPT_RATE - ATTEMPT_RATE[ I, J ] );
ATTEMPT_RATE[ I, J ] := NEW_ATTEMPT_RATE;

end;

End; (* Find transmission probabilities. *)

(* Start of main procedure. *)

Begin

(* Input the file names. *)

Write( 'Type file name: ' );
Readln( FILE_NAME );

(* Open input and output files. *)

Open( IN_FILE, FILE_NAME + '.DAT', readonly );
Open( OUT_FILE, FILE_NAME + '.RES' );
Reset( IN_FILE );
Rewrite( OUT_FILE );

(* Read in initialization parameters. *)

Read( IN_FILE, N );

For I := 0 to N-1 do
    For J := 0 to N-1 do
        Read( IN_FILE, DISTANCE[ I, J ] );

For I := 0 to N-1 do
    For J := 0 to N-1 do
        Read( IN_FILE, END_TO_END_FLOW[ I, J ] );

For I := 0 to N-1 do
    For J := 0 to N-1 do
        Read( IN_FILE, FLOW[ I, J ] );

(* Find first non-zero flow value in the flow matrix. *)

I := 0;
J := 0;
while( FLOW[ I, J ] = 0 ) do
  begin
    While( ( FLOW[ I, J ] = 0 ) and ( J < N ) ) do
      begin
        J := J + 1;
        end;

    If ( FLOW[ I, J ] = 0 )
      then I := I + 1;
  end;

  SCALE_VALUE := FLOW[ I, J ];

(* Now find upper and lower bounds on flow. The upper bound *)
(* is obtained by scaling all flows to the first non-zero *)
(* flow found above. The lower bound is 0. *)

  For I := 0 to N-1 do
  For J := 0 to N-1 do
    Begin
      HI_FLOW[ I, J ] := FLOW[ I, J ]/SCALE_VALUE;
      LO_FLOW[ I, J ] := 0.0;
    End;

(* Now do the binary search to find the maximum flow matrix *)
(* that can be supported. HI_FLOW is always too high for *)
(* the network to support. LO_FLOW is can always be *)
(* supported. On each iteration of the loop, either HI_FLOW *)
(* or LOW_FLOW is set to the flow mid-way beteen the two. *)
(* The loop terminates when the difference between HI_FLOW *)
(* and LOW_FLOW is sufficiently small. *)

  ERROR := 1.0;
  while ( ERROR > 0.00001 ) do
   begin
     (* Set up MID_FLOW. *)

     For I := 0 to N-1 do
       for J := 0 to N-1 do
         MID_FLOW[ I, J ] := ( HI_FLOW[ I, J ] +
                              LO_FLOW[ I, J ] )/2;

     (* Find if MID_FLOW is possible. *)
FIND_TRANS_PROB( N, DISTANCE, MID_FLOW );
If ( STOP = 1 ) then
   For I := 0 to N-1 do
      for J := 0 to N-1 do
         HI_FLOW[ I, J ] := MID_FLOW[ I, J ]
   Else
      For I := 0 to N-1 do
         for J := 0 to N-1 do
            LO_FLOW[ I, J ] := MID_FLOW[ I, J ];
(* Set up to determine if done. *)
   For I := 0 to N-1 do
      for J := 0 to N-1 do
         If HI_FLOW[ I, J ] <> 0 then
            ERROR := ( HI_FLOW[ I, J ] - LO_FLOW[ I, J ] )/
                     HI_FLOW[ I, J ];
   End;
(* Find the maximum success probabilities, etc. *)
FIND_TRANS_PROB( N, DISTANCE, LO_FLOW );
(* Put the input in the output file. *)
WRITELN( OUT_FILE, 'Distance Matrix:' );
For I := 0 to N-1 do
begin
   For J := 0 to N-1 do
      Write( OUT_FILE, DISTANCE[ I, J ]:5:5, ' ' );
      WriteLn( OUT_FILE );
End;
WriteLn( OUT_FILE );
WRITELN( OUT_FILE, 'End to End Packet Flow Matrix:' );
For I := 0 to N-1 do
begin
   For J := 0 to N-1 do
      Write( OUT_FILE, END_TO_END_FLOW[ I, J ]:5:5, ' ' );
      WriteLn( OUT_FILE );
End;
WriteLn( OUT_FILE );
WRITELN( OUT_FILE, 'Link Flow Matrix:' );
For I := 0 to N-1 do
begin
  For J := 0 to N-1 do
    Write( OUT_FILE, FLOW[ I, J ]:5:5, ' ' );
    Writeln( OUT_FILE );
  End;
  Writeln( OUT_FILE );

(* Now output maximum link flow matrix. *)

WRITELN( OUT_FILE,
  'Maximum Link Flow Matrix with Relative Flow Constant:' );
For I := 0 to N-1 do
begin
  For J := 0 to N-1 do
    Write( OUT_FILE, LO_FLOW[ I, J ]:5:5, ' ' );
    Writeln( OUT_FILE );
  End;
  Writeln( OUT_FILE );

(* Now calculate maximum total throughput. Since we are only *)
(* interested in throughput towards the receiver this is not *)
(* straightforward. What we do is to scale up the end to end *)
(* link flow matrix given to us as input. *)

MAX_TOTAL_THROUGHPUT := 0;
END_TO_END_THROUGHPUT := 0;
LINK_THROUGHPUT := 0;
For I := 0 to N-1 do
begin
  For J := 0 to N-1 do
  begin
    MAX_TOTAL_THROUGHPUT := MAX_TOTAL_THROUGHPUT +
      LO_FLOW[ I, J ] * DISTANCE[ I, J ];
    END_TO_END_THROUGHPUT := END_TO_END_THROUGHPUT +
      END_TO_END_FLOW[ I, J ] * DISTANCE[ I, J ];
    LINK_THROUGHPUT := LINK_THROUGHPUT +
      FLOW[ I, J ] * DISTANCE[ I, J ];
  end;
end;

MAX_TOTAL_THROUGHPUT :=
  END_TO_END_THROUGHPUT * MAX_TOTAL_THROUGHPUT/LINK_THROUGHPUT;

(* Write the maximum throughput to the output file. *)

WRITELN( OUT_FILE,
'Maximum total throughput in packets X ',
'distance towards destination per slot');
WRITELN( OUT_FILE, MAX_TOTAL_THROUGHPUT:5:5 );
Writeln( OUT_FILE );

(* Now calculate and output the retransmission probabilities. *)

WRITELN( OUT_FILE, 'Retransmission Probabilities: ' );
For I := 0 to N-1 do
begin
  RETRANS_PROB[ I ] := 0;
  For J := 0 to N-1 do
    RETRANS_PROB[ I ] := RETRANS_PROB[ I ]
    + ATTEMPT_RATE[ I, J ];
  Write( OUT_FILE, RETRANS_PROB[ I ]:5:5, ' ' );
End;
Writeln( OUT_FILE );
Writeln( OUT_FILE );

(* Now find the success probabilities for our given flow. *)

FIND_TRANS_PROB( N, DISTANCE, FLOW );

(* Write out the success probabilities. *)

WRITELN( OUT_FILE, 'Success Matrix: ' );
For I := 0 to N-1 do
begin
  For J := 0 to N-1 do
    Write( OUT_FILE, SUCCESS_PROB[ I, J ]:5:5, ' ' );
  Writeln( OUT_FILE );
End;
Writeln( OUT_FILE );

(* Now calculate the total average velocity. Again since we *)
(* are only interested in velocity towards the receiver we *)
(* must scale things from the end to end throughput given to *)
(* us as an input parameter. *)

TOTAL_DISTANCE := 0;
TOTAL_TIME := 0;

For I := 0 to N-1 do
  For J := 0 to N-1 do
    begin
TOTAL_DISTANCE := TOTAL_DISTANCE +
    END_TO_END_FLOW[ I, J ] * DISTANCE[ I, J ];
If ( SUCCESS_PROB[ I, J ] <> 0 ) then
    TOTAL_TIME := TOTAL_TIME + FLOW[ I, J ] *
        ( 1 + ( 1 - SUCCESS_PROB[ I, J ] ) / 
        ( RETRANS_PROB[ I ] * SUCCESS_PROB[ I, J ] ) ) ;
End;

(* Write out the average velocity. *)

VELOCITY := TOTAL_DISTANCE/TOTAL_TIME;
write( OUT_FILE, 'Total Average Velocity: ' );
write( OUT_FILE, VELOCITY:5:5 );

End. (* Find p and V given f. *)
I.2 Program to Find the Service Time in the Very Lightly Loaded Case

(* ********************************************** *)
(* This program finds the average service time of network in the *)
(* large N case. It takes five parameters as input. The first *)
(* parameter is the product of ETA and D where ETA is the rate of *)
(* arrival of packets at a node and D is the average distance that *)
(* the packets must travel. The second parameter is the average *)
(* number of terminals per unit area. The third and fourth *)
(* parameters are the lower and upper bounds on N where N is the *)
(* average number of terminals within a transmission radius of the *)
(* receiver. The final parameter is the number of values of N *)
(* between the minimum and maximum N for which the program is to *)
(* calculate the following: *)
(* N: Number of terminals within the transmission radius. *)
(* Flow: Average number of successful transmissions that must *)
(* occur in each slot for each terminal. *)
(* R: Transmission radius. *)
(* V1: Velocity using the approximate analysis. *)
(* V2: Velocity using the iterative approach. *)
(* V3: Velocity using different probabilities of success for *)
(* transmissions and retransmissions. On *)
(* retransmissions we assume one other node is *)
(* backlogged. *)
(* *)
(* ********************************************** *)

Program FIND_SERVICE_TIME( INPUT, OUTPUT, IN_FILE, OUT_FILE );

var
    IN_FILE:        Text;
    OUT_FILE:       Text;
    FILE_NAME:      Varying[ 132 ] of char;
    IN_FILE_NAME:   Varying[ 132 ] of char;
    SAMPLES:        Integer;
    N:              Real;
    MIN_N:          Real;
    MAX_N:          Real;
    MAX_RADIUS:     Real;
    I:              Integer;
    STEP_SIZE:      Real;
    A:              Real;
    SERVICE_4:      Real;
ETAD: Real;
LAMBDA: Real;
TRANSMISSION_RADIUS: Real;
RETRANSMISSION_PROB: Real;
SS_SUCCESS_PROB: Real;
APP_SS_SUCCESS_PROB: Real;
FIRST_SUCCESS_PROB: Real;
RETRANS_SUCCESS_PROB: Real;
OVERALL_SUCCESS_PROB: Real;
NUM_WITHIN_RADIUS: Real;
FLOW: Real;

SERVICE_1: Real;
SERVICE_2: Real;
SERVICE_3: Real;

VELOCITY_1: Real;
VELOCITY_2: Real;
VELOCITY_3: Real;

Procedure CALCULATE_SERVICE_TIME;

var OLD_PROB: Real;
    TEMP: Real;
Begin

(* Calculate N and retransmission probability. *)

NUM_WITHIN_RADIUS := 3.1415926 * LAMBDA * 
( TRANSMISSION_RADIUS ** 2 );
RETRANSMISSION_PROB := 1 / ( NUM_WITHIN_RADIUS + 1 );

(* Calculate approximate steady state success probability. *)

TEMP := ( 1 - 4 *
    ETAD * NUM_WITHIN_RADIUS/TRANSMISSION_RADIUS ) ;
IF TEMP > 0 THEN APP_SS_SUCCESS_PROB := ( 1 + TEMP ** 0.5 )/2
    ELSE APP_SS_SUCCESS_PROB := 1;

(* Calculate approximate steady state success probability using the *)
(* iterative approach. *)

SS_SUCCESS_PROB := 1;
OLD_PROB := 0;
while( ( ( SS_SUCCESS_PROB - OLD_PROB )/SS_SUCCESS_PROB )
Begin
OLD_PROB := SS_SUCCESS_PROB;
SS_SUCCESS_PROB := ( 1 - ETAD /
( TRANSMISSION_RADIUS * OLD_PROB ) ) ** NUM_WITHIN_RADIUS;
End;

(* Now assume that one node is backlogged on retransmissions. *)

OVERALL_SUCCESS_PROB := 1;
OLD_PROB := 0;
while( (( OVERALL_SUCCESS_PROB - OLD_PROB )/
OVERALL_SUCCESS_PROB ) > 0.00001 ) do
Begin
OLD_PROB := OVERALL_SUCCESS_PROB;
FIRST_SUCCESS_PROB := ( 1 - ETAD /
( TRANSMISSION_RADIUS * OLD_PROB ) ) ** NUM_WITHIN_RADIUS;
RETRANS_SUCCESS_PROB := ( 1 - RETRANSMISSION_PROB )
* ( 1 - ETAD /
( TRANSMISSION_RADIUS * OLD_PROB ) ) ** NUM_WITHIN_RADIUS;
OVERALL_SUCCESS_PROB := RETRANSMISSION_PROB /
(RETRANS_SUCCESS_PROB + 1 - FIRST_SUCCESS_PROB );
End;

(* Calculate the average service times. *)

SERVICE_1 := 1 + ( 1 - APP_SS_SUCCESS_PROB ) /
(RETRANSMISSION_PROB * APP_SS_SUCCESS_PROB );
SERVICE_2 := 1 + ( 1 - SS_SUCCESS_PROB ) /
(RETRANSMISSION_PROB * SS_SUCCESS_PROB );
SERVICE_3 := 1 + ( 1 - FIRST_SUCCESS_PROB )/
(RETRANSMISSION_PROB * RETRANSMISSION_PROB );

(* Calculate the average velocities. *)

VELOCITY_1 := TRANSMISSION_RADIUS / SERVICE_1;
VELOCITY_2 := TRANSMISSION_RADIUS / SERVICE_2;
VELOCITY_3 := TRANSMISSION_RADIUS / SERVICE_3;
End;

(* Start of main procedure. *)

Begin
(* Input the file names. *)

Write( 'Type file name: ' );
Readln( FILE_NAME );

(* Open input and output files. *)

Open( IN_FILE, FILE_NAME + '.DAT', readonly );
Open( OUT_FILE, FILE_NAME + '.RES' );
Reset( IN_FILE );
Rewrite( OUT_FILE );

(* Read in initialization parameters. *)

Read( IN_FILE, ETAD, LAMBDA, MIN_N, MAX_N, SAMPLES );
WRITELN( OUT_FILE, ' σD = ', ETAD:10:9,
          ', λ = ', LAMBDA:5:2 );
WRITELN( OUT_FILE, ' 
', ' N     FLOW     R     V_1',
          ',
          ' V_2     V_3' );

STEP_SIZE := ( MAX_N - MIN_N )/SAMPLES;
MAX_RADIUS :=
  1/( \exp( 1 ) * 3.1415926 * LAMBDA * ETAD );
MAX_N := MIN( MAX_N, ( 3.1415926 * MAX_RADIUS *
                  MAX_RADIUS * LAMBDA ) );

For I := 0 to SAMPLES do
  Begin

    N := MIN( MAX_N, ( I * STEP_SIZE + MIN_N ) );
    TRANSMISSION_RADIUS := ( N / ( 3.1415916 * LAMBDA ) ) ** 0.5;
    CALCULATE_SERVICE_TIME;
    FLOW := ETAD / TRANSMISSION_RADIUS;

    Writeln( OUT_FILE, ' ', N:7:2, ' ',
              FLOW, ' ',
              TRANSMISSION_RADIUS:7:2, ' ',
              VELOCITY_1:7:2, ' ',
              VELOCITY_2:7:2, ' ',
              VELOCITY_3:7:2 );

  End;

End.
1.3 Program to Compare Transmission Strategies

(*******************************************************************************************)
(*
(* This program compares the Closest to Optimum and Most Forward
(* within Rp transmission strategies with each other and the upper
(* bound obtained by assuming that all transmissions go to the
(* optimum place. In order to compute velocity for the Most Forward
(* within Rp strategy we must numerically evaluate two integrals. The
(* numerical integration is done using the trapezoidal rule.
(* Additionally, we must determine the value of Rp to use to maximize
(* velocity. This program maximizes velocity by doing a search of
(* values between Ropt/2 and 1.5*Ropt.
(*
(* The output of this program is the velocity for each delay model
(* with ETAD set to make the optimum N between 5 and 100. The output
(* is formatted to be included in a Scribe file.
(*
*******************************************************************************************)

Program COMPARE_TRANS_STRATEGIES ( INPUT, OUTPUT, OUT_FILE );

var OUT_FILE: Text;
var J: Integer;
var RP: Real;
var ETAD: Real;
var ROPT: Real;
var NOPT: Real;
var VOPT: Real;
var RCLOSEST: Real;
var VCLOSEST: Real;
var I: Integer;
var K: Integer;
var VELOCITY: array [0..4] of Real;
var RP_TEST: array [0..4] of Real;
var FLAG: Integer;

*******************************************************************************************)
(*
(* The function F implements the two functions that have to be
(* integrated. If the flag is set to 1 then it computes the function
(* required to compute E[X] and if the flag is set to 0 it computes
(* the function required to compute E[X*X].
(*
****************************************************************************************%%%%
Function F( var X: Real;
    flag: integer ): Real;

var TEMP1: Real;
    TEMP2: Real;
    RES: Real;

begin

    TEMP1 := SQRT(1 - (X/RP)*(X/RP));
    TEMP2 := arctan( sqrt( rp * rp - x * x )/x );
    If ( FLAG = 1 ) THEN
        RES := X * EXP( - RP * RP *( TEMP2 - (X/RP) * TEMP1 ) ) * 2 * RP * TEMP1
    ELSE
        RES := X*X * EXP( - RP * RP *( TEMP2 - (X/RP) * TEMP1 ) ) * 2 * RP * TEMP1;
    F := RES;
end; (* F *)

(******************************************************************************)
(*
(* Integrate does the numerical integration. It keeps trying smaller *)
(* step sizes until it does two in a row that differ by less than .1 *)
(* % It uses the trapezoidal rule. *)
(*
(******************************************************************************)

Function INTEGRATE(
    X_MIN: Real;
    X_MAX: Real;
    FLAG: Integer ): Real;

var
    N: Integer;
    STEP_SIZE: Real;
    SUM: Real;
    I: Integer;
    TEMP: Real;
    RESULT: Real;
    OLD_RESULT: Real;
    ERROR: Real;

begin
(* Set up N to do the integration. *)

N := 100;
ERROR := 1;
RESULT := 0;

(* Now repeat the integration until we are accurate enough. *)

while( ERROR > 0.001 ) do
begin

(* Do the integration. *)

STEP_SIZE := ( X_MAX - X_MIN )/N;
SUM := 0;
For I := 0 TO N do
begin
TEMP := MIN( (X_MIN + STEP_SIZE * I), X_MAX );
SUM := 2 * F( TEMP, FLAG ) + SUM;
end;
SUM := SUM - F( X_MIN, FLAG ) - F( X_MAX, FLAG );
OLD_RESULT := RESULT;
RESULT := STEP_SIZE * SUM/2;
ERROR := ABS( (RESULT - OLD_RESULT )/RESULT );
end;

INTEGRATE := RESULT;

End; (* Integrate. *)

FUNCTION MFR_VELOCITY( RP: real ): real;

(********************************************************************)
(*
(* The function MFR_VELOCITY determines the velocity when using the  *
(* MFR strategy. It takes the parameter radius as parameter and    *
(* returns the velocity when using this strategy.                  *
(*
(* Note that the lower bound on our integration must be slightly   *
(* higher than zero since the functions we are integrating go to   *
(* infinite at 0.                                                  *
(*
(********************************************************************)
var RCOS: Real;
var R2: Real;
var RCOS2: Real;
var FLAG: Integer;
var TEMP: Real;

begin

   RCOS := INTEGRATE( 0.000001, RP, 1 );
   RCOS2 := INTEGRATE( 0.000001, RP, 0 );
   R2 := ( 2 * RCOS2 + RP*RP )/3;
   TEMP := 1/((1/RCOS) +
            3.1415926*R2*(3.1415926*R2+1)*ETAD/(RCOS**2));
   MFR VELOCITY := TEMP;

end;

(* Beginning of main program. *)

begin

   Open( OUT_FILE, 'compare.res' );
   Rewrite( OUT_FILE );

   writeln( OUT_FILE, ' nD ',
            ' ROPT ',
            ' NOPT ',
            ' RP ',
            ' V OPT ',
            ' V CLOSEST ',
            ' v MFR ');

   writeln( OUT_FILE );

   for J := 2 to 40 do
   begin
      (* Set up the load parameter. *)

      NOPT := 2.5 * J;

      writeln( OUT_FILE, NOPT, RP, V OPT, V CLOSEST, v MFR );
   end;

end;
ROPT := SQRT( NOPT/3.1415926 );
ETAD := 1/( 2 * ( 3.1415926 ** 2 ) * ( ROPT ** 3 ) );

(* Now calculate the optimum velocity. *)
VOPT := 1/((1/ROPT) + (3.1415926**2) * ETAD * (ROPT**2)
+ 3.1415926*ETAD);

(* Now calculate the velocity using *)
(* closest to optimum algorithm. *)
RCLOSEST2 := ROPT**2 + 1/3.1425926;
VCLOSEST := 1/( (1/ROPT) +
3.1415926*RCLOSEST2*(3.1415926*RCLOSEST2+1)*ETAD/(ROPT**2) );

(* Finally calculate the velocity using the MFR algorithm. *)
(* We have to do a search to find the optimum value of RP. *)
(* We do this by doing a binary type search. RP_HI is *)
(* always higher than the optimum RP and RP_LO is always *)
(* lower than the optimum RP. On each iteration we compute *)
(* the velocity for three values of RP between RP_HIGH and *)
(* RP_LO. We then adjust RP_HI and RP_LO to keep RP_HI *)
(* higher than optimum and RP_LO lower than optimum. We *)
(* are finished when RP_HI and RP_LO are sufficiently close *)
(* that velocity does not vary much between the two. *)

RP_TEST[ 4 ] := 1.5 * ROPT;
RP_TEST[ 0 ] := ROPT/2.0;

RP := RP_TEST[ 0 ];
VELOCITY[ 0 ] := MFR VELOCITY( RP );

RP := RP_TEST[ 4 ];
VELOCITY[ 4 ] := MFR VELOCITY( RP );

K := 1;
I := 0;

while( ( RP_TEST[ 4 ] - RP_TEST[ 0 ] )/ RP_TEST[ 0 ] > 0.001 )
and ( ( K <> 4 ) or ( I <> 0 ) ) do
begin
for I := 1 to 3 do
begin
RP_TEST[ I ] := RP_TEST[ 0 ] +
end
RP := RP_TEST[ I ];
VELOCITY[ I ] := MFR VELOCITY( RP );
end;

I := 0;
while( VELOCITY[ I ] < VELOCITY[ I + 1 ] ) and
( I < 3 ) do
I := I + 1;
if I > 0 Then I := I - 1;
VELOCITY[ 0 ] := VELOCITY[ I ];
RP_TEST[ 0 ] := RP_TEST[ I ];
end;

K := 4;
while( VELOCITY[ K ] < VELOCITY[ K - 1 ] ) and
( K > 1 ) do
K := K - 1;
if K < 4 then K := K + 1;
VELOCITY[ 4 ] := VELOCITY[ K ];
end;

writeln( OUT_FILE, ETAD:9:7, ' ',
ROPT:7:5, ' ',
NOPT:7:2, ' ',
RP_TEST[ 0 ]:7:5, ' ',
VOPT:7:4, ' ',
VCLOSEST:7:4, ' ',
VELOCITY[ 0 ]:7:4 );
end;

End.
References


