A Practical Scheme for Wireless Network Operation

Radhika Gowaikar, Amir F. Dana, Babak Hassibi, Michelle Effros

June 21, 2004

Abstract

In many problems in wireline networks, it is known that achieving capacity on each link or sub-network is optimal for the entire network operation. In this paper we show that achieving capacity on the sub-networks of a wireless network does not guarantee achieving the capacity of the entire network. This implies that operating wireless networks in a multi-hop manner, where each relay node decodes what it receives is not necessarily the right approach. We demonstrate this with some examples. Next we consider Gaussian and erasure wireless networks where nodes are are permitted only two possible operations – nodes can either decode what they receive (and then re-encode and transmit the message) or simply forward it. We present a simple greedy algorithm that returns the optimal scheme from the exponential-sized set of possible schemes. This algorithm will go over each node atmost once to determine its operation and hence is very efficient.

1 Introduction

In a wireline network having a single source and a single destination, we can think of information flow in the same terms as fluid flow and obtain a max-flow min-cut result to get capacity. This treatment closely follows that of the Ford-Fulkerson [1] algorithm to give us a neat capacity result. This has been well understood for many years. However, until recently, similar min-cut capacity results were not known for any other class of network problems. Before we describe the recent results obtained in network problems, let us understand the general network problem. This can be stated in the context of a multi-terminal network [2] as follows. We have a set of nodes and the "channel" between these is specified by a probability transition function which governs the relationship between the signals transmitted by the nodes and how these are received by the other nodes. Every node can have messages that it wants to send to every other node. Because of the generality of this model, it can be tailored to describe many practical systems easily. For instance, several wireless as well as wireline systems, (stationary) ad hoc and sensor networks, etc., can be modeled by choosing a suitable probability transition function.

Not surprisingly, finding the capacity region in this general setting is extremely challenging. In [2] outer bounds on the capacity region can be found. These have the form of "min-cut" upper bounds. Such an upper bound formalizes the intuitively satisfying notion that the rate from node a to node b cannot exceed the rate that any cutset of edges from a to

b can support. However, determining whether schemes of network operation that reach this upper bound exist or not has proven to be very difficult. Even in simple relay networks, i.e., networks having one source node, one destination node and a single other node (called the relay node), the answer to this question is not known in general [2]. Only in special cases of the probability transition function (defined as "degraded" distributions) do we know schemes that can reach the upper bounds and thus attain capacity.

In this context, the results in [3], [4] are remarkable. They say that, in a wireline network setting, we can indeed achieve the min-cut upper bounds for a special case of a certain class of problems called multicast problems. In this problem, we have one source node and several sink nodes that want to receive the same message from the source. It turns out that using network coding techniques we can achieve the min-cut capacity of the network. Further, [5] put this problem in an algebraic framework and presented *linear* schemes that also achieved this capacity. In addition, for some more general multicast problems capacity has been shown to be achievable using linear network coding [5]. The work of [6, 7, 8] demonstrates the strengths of this algebraic approach.

We motivate the work presented in this paper by first examining a feature of the recent results in wireline networks and trying to determine if this feature is applicable in more general networks, viz., that in all the capacity achieving schemes we have referred to above, the min-cut upper bounds are reached through separate channel and network coding. This means that there exists an optimal strategy in which each link in the wireline network can be made error-free by means of channel coding and network coding can be employed separately on top of this to determine which messages should be transmitted on which link. This is quite unexpected and leads us to wonder if such a separation can be optimal in more general network settings.

In the early sections of this paper, we will present simple *wireless* networks where this principle of separation fails. Thus we will show that operating wireless networks in a multi-hop manner, where each relay node decodes the message it receives is not necessarily the right approach. We will suggest some schemes of operation that will outperform those that require the ability of relay nodes to decode.

We will focus attention on two specific wireless network models. The important features that characterize a wireless network are broadcast and interference. We will look at Gaussian Wireless Networks and Erasure Wireless Networks. The former has Gaussian channels as links and incorporates broadcast as well as interference. The second model has erasure channels as links and incorporates broadcast, but not interference. For these models, we will show that making links error-free can sometimes degrade the performance. In fact, asking nodes to simply forward their data rather than decoding it is sometimes more advantageous. This tells us that wireless networks need to be understood differently from wireline networks. We will see some explanations as to why this is the case later in the paper.

In our study of wireless networks, we propose a scheme of network operation that permits nodes only two operations. One is decoding to get the original data and then resending the same message as the source. The other is forwarding the data as is received. Since each node has two options, we have an exponential-sized set of possible operations. We will present an algorithm that goes over each node at most once to find the optimal operation among this set of restricted operations. This will be a greedy algorithm that avoids searching over the exponential-sized set of possible operation allocations. We also present an algorithm that can approach the best rate arbitrarily closely in an iterative manner. This will be a "decentralized" algorithm in the sense that each node needs only one bit of information from the destination in every iteration and no knowledge of the rest of the network in order to determine its own operation.

The organization of this paper is as follows. In Section 2 we present two wireless network models. These will be the Gaussian and Erasure Wireless Networks. In Section 3 we show that with these wireless models, making links or sub-networks error-free can be sub-optimal. In Section 4 we will formally state the two operations that nodes will be permitted to perform. With this setup, we will state our problem of allocating appropriate operations in Section 5. In Section 6 we will see how rates are calculated for all nodes in the network and how asking certain nodes to decode and others to forward can affect the rate of the network. In Section 7 we will state our algorithm to find the optimal policy. In Section 8 we will prove optimality of the algorithm. We will see some examples in Section 9 that will show that the gap between the "all nodes decode" strategy and our method can be significant. In Section 10 we will discuss the decentralized algorithm. We present upperbounds on the rate achievable by our scheme in Section 11. Conclusions and future work is presented in 12.

2 Two wireless network models

In this section we formalise two wireless network models. These are Gaussian Networks and Erasure Networks. In both cases the network consists of a directed, acyclic graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ where $\mathcal{V}$ is the set of vertices and $\mathcal{E}$ is the set of directed edges where each edge is a communication channel. We will denote $|\mathcal{V}| = V$ and $|\mathcal{E}| = E$. Also, we will have $\mathcal{V} = \{v_1, \cdots, v_V\}$ and $\mathcal{E} = \{(v_i, v_j) : (v_i, v_j) \text{ is an edge}\}$. We will assume, without loss of generality, that $s = v_1$ is the source node and $d = v_V$ is the destination. We will assume that every edge is on some directed path from s to d. If we have edges other than these, we remove them and what remains is our graph $\mathcal{G}$. We will denote the message transmitted by vertex v_i by $X(v_i)$ and that received by node v_j by $Y(v_j)$.

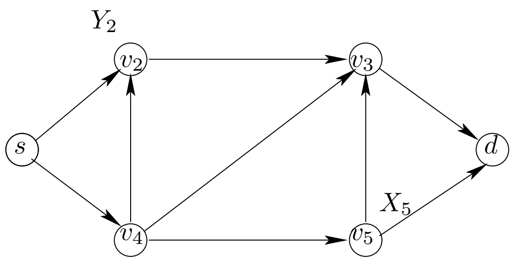


Figure 1: Example of a network

Figure 1 represents a network with 6 vertices and 9 edges where v_1 is the source s and v_6 is the destination d. $X(v_5)$ is the message transmitted by v_5 and $Y(v_2)$ is that received by v_2.

1. **Gaussian Wireless Networks** In these networks, each edge (v_i, v_j) of the network is a Gaussian channel. We will assume that nodes broadcast messages, i.e., a node

transmits the same message on all outgoing edges. Assuming that Figure 1 represents a Gaussian wireless network, $X(v_5)$ is the message transmitted on edges (v_5, v_3) and (v_5, v_6). We will also assume interference, i.e., the received signal at node v_i is the sum of all the signals transmitted on edges coming in to it and additive white Gaussian noise n_i of variance σ_i^2. Therefore, in general, we have

$$Y(v_i) = n_i + \sum_{v_j:(v_j,v_i)\in\mathcal{E}} X(v_j).$$

All n_is are assumed independent of each other as well as the messages. For Figure 1 this implies that $Y(v_2) = X(v_1) + X(v_4) + n_2$. We will assume that all transmitting nodes have a power constraint of P.

2. **Erasure Wireless Networks** In these networks, each edge (v_i, v_j) of the network is a binary erasure channel with erasure probability $\epsilon_{i,j}$. In addition, we assume that nodes (other than the source node) can transmit erasures and they are received as erasures with probability 1. Denoting erasure by $*$, this assumption means that edges can also take $*$ as input and this is always received as $*$. In short, the channel for edge (v_i, v_j) (for $v_i \neq s$) is modified as in Figure 2. We incorporate broadcast in the model, i.e., each transmitting node must send out the same signal on each outgoing edge. Now assuming that Figure 1 represents a wireless erasure network, v_5 transmits $X(v_5)$ on edges (v_5, v_3) and $(v_5, 4_6)$.

   However, we do not permit interference. This means that a node having several incoming edges sees messages from each edge without their interfering with each other. In general, if v_i has $\gamma_I(i)$ incoming edges, it will see $\gamma_I(i)$ messages that do not interfere with each other. [1]

   Finally, we mention that instead of the regular binary erasure channel, we can consider a channel with any finite alphabet $\mathcal{A}$ as the input alphabet and get a more general erasure wireless network model. Our results go through for this also, but for simplicity, we restrict ourselves to binary inputs.

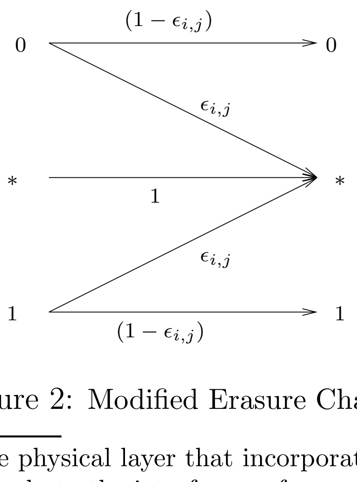


Figure 2: Modified Erasure Channel

[1]There exist network models in the physical layer that incorporate interference, which, when abstracted to an erasure network model act similarly to the interference-free model we have described here.

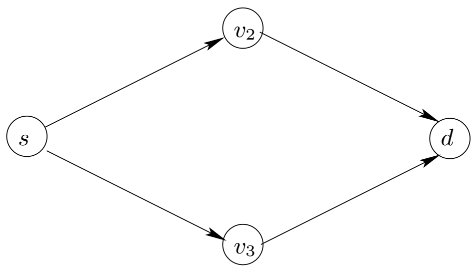
In Figure 1, we see that $Y(v_2)$ consists of two received messages – the message coming in on edge (v_1, v_2) (which is $X(v_1)$ with some bits erased) as well as the message coming in on edge (v_4, v_2) (which is $X(v_4)$ with some bits erased).

For both networks, we will assume instantaneous transmission on all links.

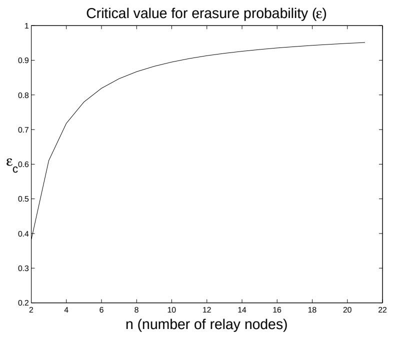
3 Optimizing over sub-networks does not work

Theorem 1. *For the wireless networks described in Section 2, making sub-networks error-free can be suboptimal.*

Proof. We give some examples to demonstrate this.



(a) Graph representation of a relay network with two relay nodes.



(b) Critical value for erasure probability as a function of the number of nodes in the relay network.

Figure 3: Proof of Theorem 1

- **Gaussian Relay Networks**: Consider a Gaussian parallel relay network consisting of two relay nodes and one source-destination pair. See Figure 3(a). The relay nodes v_2 and v_3 are solely to aid communication from source to destination. We assume that the noise power at each receiver is σ^2 and the transmit power at each node is P. Let $\rho \triangleq \frac{P}{\sigma^2}$ be Signal to Noise Ratio (SNR).

  One way to view the network is as a cascade of a broadcast channel (from s to $\{v_2, v_3\}$) and a multiple access channel (from $\{v_2, v_3\}$). This is equivalent to assuming that the relays decode their messages correctly and code them again and transmit. If the relays are receiving independent information at rates R_1 and R_2, we have $R_1 + R_2 \leq \log(1+\rho)$ as the capacity region. These rate pairs (R_1, R_2) can be supported by the multiple access channel and hence the maximum rate from s to d is no greater than $\log(1+\rho)$. If the relays are receiving exactly the same information from the source, the maximum rate of this is $\log(1+\rho)$. In this case, the multiple access channel is used for correlated information and can support rates upto $\log(1 + 4\rho)$. In either case, asking the relay nodes to decode limits the rate from s to d to $\log(1+\rho)$. (We note also that the broadcast sub-network is the bottleneck in both cases.)

Now consider another strategy in which the relay nodes do not decode but only normalize their received signal to meet the power constraint and transmit it to the destination. In this case the received signal at the destination is

$$Y(v_4) = \sqrt{\frac{P}{P+\sigma^2}}(2X(v_1) + n_2 + n_3) + n_4$$

where $X(v_1)$, $Y(v_4)$, n_2, n_3, n_4 are, respectively, the transmitted signal from the source, the received signal at the destination and the noises introduced at v_2, v_3 and d. Thus, the signal received by d is a scaled version of $X(v_1)$ with additive Gaussian noise. The maximum achievable rate, denoted by R_f, is

$$R_f = \log\left(1 + \frac{\frac{4P^2}{P+\sigma^2}}{\sigma^2 + \frac{2P\sigma^2}{P+\sigma^2}}\right) = \log\left(1 + \frac{4\rho^2}{3\rho+1}\right).$$

where ρ is as before. Here, the subscript f stands for *forwarding.*

Comparing R_d and R_f, we can see $\rho = 1$ is a critical value in the following sense. For $\rho > 1$, we have superior performance in the forwarding scheme and for $\rho < 1$ we have better rate with relay nodes decoding and re-encoding. This implies that making a sub-network error-free (in this case, the broadcast section, or the links (v_1, v_2) and (v_1, v_3)) can sometimes be sub-optimal.

We note that decoding at one of the relay nodes and forwarding at the other is always sub-optimal.

In general, if we have $k(\geq 2)$ relay nodes in parallel rather than two, it can be easily checked that

$$R_d = \log(1+\rho) \quad \text{and} \quad R_f = \log\left(1 + \frac{k^2\rho^2}{(k+1)\rho+1}\right)$$

With this we get a critical value of $\rho = \frac{1}{k^2-k-1}$ below which decoding is better and above which forwarding is better. Clearly, this goes to zero for large k. Therefore in the limit of $k \to \infty$ it is always favorable to forward.

It turns out that this fact is also true for Gaussian relay networks in the presence of fading. The work of [9] shows that for fading Gaussian relay networks with n nodes, the asymptotic capacity achievable with the relay nodes decoding (and re-encoding) scales like $O(\log\log n)$ whereas with the forward scheme it scales like $O(\log n)$.

- **Erasure Relay Network**: Consider, once again, the network of Figure 3(a), where, now, each link represents an erasure channel with erasure probability $\epsilon_{i,j} = e$. Since we have broadcast, node s transmits the same messages to relay nodes v_2 and v_3. If the relay nodes decode and re-encode, the rate is bounded by the sum-rate capacity of the broadcast system, which gives

$$R_d = 1 - e.$$

If the relay nodes simply forward what they receive, it is easy to see that the destination sees an effective erasure probability of $(1-(1-e)^2)$. (We will spell out how to do this calculation for a general network in Section 6.) Forwarding erasures is possible since we are assuming the modified erasure channel of Figure 2. With this we have $R_f = 1-(1-(1-e)^2)^2$. Comparing R_f and R_d, we can see that $e = \frac{3-\sqrt{5}}{2}$ is a critical value, above which decoding and re-encoding is better and below which forwarding is better.

Thus we see that for this network also, making the broadcast sub-network error-free is not always optimal.

In general, if we have k relay nodes in parallel rather than two, we have

$$R_d = 1-e \quad \text{and} \quad R_f = 1-(1-(1-e)^2)^k$$

and the critical value of e is as plotted in Figure 3(b). Below this, forwarding is better and above this decoding is better. In the limit of large k, it is always better to forward.

From this we see that making links or sub-networks error-free does not ensure optimal network operation. It can sometimes be provably sub-optimal. $\square$

In this proof a simple operation like forwarding the received data proved to be better than decoding it. We understand this as follows. Because of the broadcast present in wireless networks, the same data naturally gets passed on to the destination along many different paths. Therefore some nodes receive better versions of the data on incoming links than other nodes and are automatically in a better position to decode. Forcing all the nodes to decode and be error-free only imposes additional bottle-necks on the rate. Therefore it is beneficial to carefully check the quality of the effective signal that various nodes get to see and then decide whether to ask them to decode or not.

4 A Possible Set of Network Operations

It follows from the previous discussions that to obtain the optimum rate over wireless networks, the nodes must perform operations other than just decoding. Determining what the optimum operation at each node should be, especially for a general wireless network appears to be a daunting task. [2] We shall therefore simplify the problem by allowing one of only two operations at every node. One will be the decode and re-encode operation as before. The other is the far simpler operation of forwarding the received data as is. The first operation, viz., decode and re-encode is typically the only operation used in multi-hop networks and many wireline networks. In effect, we are attempting to attain higher rates by introducing the additional operation of forwarding.

We will assume that the network operates in blocks of length n. We assume that the source s has a set of message indices

$$\Omega = \{1, 2, \ldots, 2^{\lfloor nR \rfloor}\}$$

[2]We should remark that for the broadcast erasure networks studied here, provided information on the location of erasures is available at the receiver, the optimum operation of the nodes and the optimum capacity has been found in [10]. However, we shall not delve into this here.

and an encoding function

$$f : \Omega \to \mathcal{X}^n$$

where $\mathcal{X}$ is $\mathbb{R}$ for the Gaussian wireless network and $\{0, 1\}$ for the erasure wireless network. To transmit message $i \in \Omega$, the source transmits $f(i)$. With this the source operates at rate R. $\{f(1), f(2), \ldots, f(2^{\lfloor nR \rfloor})\}$ is the set of codewords or possible transmitted messages. This set is called the codebook and is denoted by $\mathcal{C}$. We assume that all nodes have the codebook. For the Gaussian network we will assume that the codebook meets the power constraint, i.e., $E\|f(i)\|^2 \leq P$.

In this paper, we restrict the relay nodes to two operations. These have been introduced in the examples of Section 3, viz., "forward" and "decode and re-encode". We now state them formally.

1. **Decode and Re-encode**: This operation implies that when node v_i receives message $Y(v_i)$ it performs ML decoding of $Y(v_i)$ to determine which message index was transmitted by s. Since it has the codebook, it re-encodes the message using the same codeword that the source s would have used and transmits the same codeword. In short, it should act like a copy of the source.

   However, for this to happen, we need that the decoding be error-free. This implies that the rate R at which the source operates should be no greater that the maximum rate at which node v_i can decode. We will see the relevance of this constraint in Section 5.

2. **Forward**: We will describe this operation separately for the two network models. In the Gaussian network, node v_i receives message $Y(v_i)$ given by

$$Y(v_i) = n_i + \sum_{v_j : (v_j, v_i) \in \mathcal{E}} X(v_j) \tag{1}$$

   "Forwarding" implies that the node normalizes this signal to meet the power constraint and then transmits the message. Therefore it transmits $X(v_i)$ given by

$$X(v_i) = \sqrt{\frac{P}{E\|Y(v_i)\|^2}} \; Y(v_i).$$

   We will assume that $E\|Y(v_i)\|^2$ is known to v_i.

   For the erasure network, nodes either decode without error and transmit the original codeword or "forward" the received data. Consider node v_i which sees data coming in on several edges, in the form of n-length blocks of bits and erasures. For the b-th bit of such a block, it either sees erasures on every edge (and sees an *effective* erasure) or gets to see the bit on at least one incoming edge. (It cannot happen that the node sees 1 on a particular edge and 0 on another edge for the b-th position. This is because of our assumption that whenever an earlier node decodes it does so without error.) Therefore in our interference free model, every relay node sees an effective erasure channel from the source, i.e., it sees the codeword transmitted by the source with some bits erased. "Forwarding" means broadcasting this sequence of bits and erasures.

Note that the effective erasure probability seen by node v_i is a function of the network topology and parameters, $\epsilon_{i,j}$. We will see in Section 6.3 how this effective erasure probability can be calculated.

By restricting ourselves to only two operations, we have ensured that all nodes in the network see a Gaussian channel (with some effective SNR) or erasure channel (with some effective erasure probability) with respect to the transmitted codeword. Therefore, they can do ML decoding or typical set decoding if R is no greater than the rate that they can support. We will always ensure that R satisfies this constraint.

Having described the two operations permitted to the relay nodes in the two networks, we are now ready to formally state the problem.

5 Problem Statement

Since we allow only two operations to nodes, viz., "decode and re-encode" and "forward" and every relay node must perform one of these, it is enough to specify the set of relay nodes that "decode and re-encode" in order to completely specify the working of the network. The source and destination will always be excluded from this set.

If a set $D \subseteq \mathcal{V} - \{s, d\}$ is the set of nodes that "decode and re-encode", we will call D a **policy** for network operation.

Under policy D, each node of the network sees an effective (Gaussian or erasure) channel from the source. Let the effective SNR that node v_i sees under policy D be denoted by $\rho_D(v_i)$ for Gaussian networks. For erasure networks we denote the effective erasure probability seen by node v_i under policy D by $e_D(v_i)$. Therefore the rate that node v_i can support under policy D is $\log(1 + \rho_D(v_i))$ or $(1 - e_D(v_i))$ for Gaussian or erasure networks, respectively. In general we will call this $R_D(v_i)$. Nodes in D as well as the destination must be able to perform error-free decoding. This means that the rate at which the source transmits must be no greater than the rates at which these nodes can decode. This tells us that under policy D, the rate R at which we can operate the network is constrained by

$$R \leq \min_{v_i \in D \cup \{d\}} R_D(v_i). \tag{2}$$

We denote this minimum by R_D.

$$R_D = \min_{v_i \in D \cup \{d\}} R_D(v_i). \tag{3}$$

Intuitively, asking some nodes to decode means that there are more copies of the source in the network and hence the rate which the destination can support increases. On the other hand, asking a node to decode introduces a constraint on the rate R. This is the tradeoff for any policy D. For instance, in Figure 1 consider nodes v_2 and v_4. If v_4 forwards, node v_2 sees an effective erasure probability of $\epsilon_{4,2}\epsilon_{1,2} + \epsilon_{1,4}\epsilon_{1,2}(1 - \epsilon_{4,2})$. (We will see how this has been calculated in Section 6.3.) On the other hand, if v_4 decodes, node v_2 is at an advantage since it sees a lower effective erasure probability, viz., $\epsilon_{1,2}\epsilon_{4,2}$. However, asking v_4 to decode puts a constraint on the rate as seen by (2) since the rate that v_4 can support is only $(1 - \epsilon_{1,4})$. This constraint is $R_D \leq 1 - \epsilon_{1,4}$.

Our problem is to find the policy that gives the best rate, i.e., to find D such that R_D is maximised, viz.,

$$\max_{D} \min_{v_i \in D\{d\}} R_D(v_i).$$

First we need to address the question of finding $R_D(v_i)$, i.e., of finding the rate at node v_i under policy D. Recall that $X(v_i)$ and $Y(v_i)$ are the transmitted and received messages at node v_i. If we are using policy D, we will denote these by $X_D(v_i)$ and $Y_D(v_i)$. We may drop the subscript D if it is clear which policy we are referring to. Note that for the source, the transmitted message is $X(v_1)$ irrespective of the policy.

6 Determining the rate at a node – $R_D(v_i)$

In this section we describe a method to find the rate at an arbitrary node v_i when the set of decoding nodes is given by D. Therefore, we need to find the effective SNR or erasure probability of the received signal $Y_D(v_i)$. In order to do that, we need the concept of a partial ordering on the nodes.

6.1 Partial Ordering of nodes

Consider two distinct nodes v_i and v_j of the network. Exactly one of the following three will occur:

1. There is a directed path from v_i to v_j. In this case we will say that $v_i < v_j$.

2. There is a directed path from v_j to v_i. In this case we will say that $v_j < v_i$.

3. There is no directed path from v_i to v_j or from v_j to v_i. In this case we will say that v_j and v_i are incomparable.

Note that since we assume acyclic networks, we cannot have directed paths both from v_i to v_j and from v_j to v_i. Thus we have a partial ordering for nodes in the network. For example, in Figure 1, we have $v_4 < v_3$ but v_2 and v_5 are incomparable. Note that the partial ordering gives us a (non-unique) sequence of nodes starting with s such that for every v_i, all the nodes v_j that satisfy $v_j < v_i$ are before it in the sequence [11]. Call such a sequence $\mathcal{S}$. A possible sequence $\mathcal{S}$ for Figure 1 is s, v_4, v_2, v_5, v_3, d.

Next we address the issue of determining the rate under a particular policy. We discuss this separately for Gaussian wireless networks and Erasure wireless networks.

6.2 Finding the rate in Gaussian Wireless Networks

Recall that $Y_D(v_j)$ is the received signal at v_j under policy D. Once we know $Y_D(v_j)$ we can determine the signal power and the noise power in it. Denote these by $P_D(v_j)$ and $N_D(v_j)$ respectively. Consider node v_j. If it is decoding, $X_D(v_j) = X(v_1)$. If it is forwarding,

$$X_D(v_j) = \sqrt{\frac{P}{E\|Y_D(v_j)\|^2}}\ Y_D(v_j) = \sqrt{\frac{P}{P_D(v_j) + N_D(v_j)}}\ Y_D(v_j).$$

For the Gaussian wireless networks, we now outline a method for finding the rate for all the nodes by proceeding in the order given by $\mathcal{S}$. Without loss of generality, assume that the nodes are already numbered according to a partial ordering. Therefore $\mathcal{S} = (v_1 = s, v_2, \ldots, v_V = d)$. Then for v_2, we only have an edge coming in from s and hence

$$Y_D(v_2) = X(v_1) + n_2.$$

Let our induction hypothesis be that we know $Y_D(v_j)$ for $j = 1, \ldots, i-1$. For $Y_D(v_i)$ we now have

$$\begin{aligned} Y_D(v_i) &= n_i + \sum_{v_j:(v_j,v_i)\in\mathcal{E}} X_D(v_j) \qquad (4)\\ &= n_i + \sum_{v_j:(v_j,v_i)\in\mathcal{E}, v_j\in D\cup\{s\}} X(v_1) + \sum_{v_j:(v_j,v_i)\in\mathcal{E}, v_j\notin D\cup\{s\}} X_D(v_j)\\ &= n_i + \sum_{v_j:(v_j,v_i)\in\mathcal{E}, v_j\in D\cup\{s\}} X(v_1) + \sum_{v_j:(v_j,v_i)\in\mathcal{E}, v_j\notin D\cup\{s\}} \sqrt{\frac{P}{P_D(v_j)+N_D(v_j)}}\, Y_D(v_j). \end{aligned}$$

By our hypothesis, we know all the $Y_D(v_j)$ that occur in the last summation, Substituting for these, we get $Y_D(v_i)$. Careful observation indicates that this will be a linear combination of $X(v_1)$ and the noise terms $n_2, \ldots, n_i$.

In general, if this linear combination is given by

$$Y(v_i) = a_D X(v_1) + \sum_{j=2}^{i} a_{D,j}(v_i) n_j,$$

we have $P_D(v_i) = a_D^2 P$ and $N_D(v_i) = \sum_{j=2}^{i} a_{D,j}^2(v_i)\sigma_j^2$. Once these are known, the SNR is simply $\rho_D(v_i) = \frac{P_D(v_i)}{N_D(v_i)}$ and the rate can be calculated as $R_D(v_i) = \log(1 + \rho_D(v_i))$.

6.3 Finding rate in Erasure Wireless Networks

We first put this problem in a graph theoretic setting. We are given a directed, acyclic graph where certain nodes act as sources. For us, the set $D \cup \{s\}$ is the set of source nodes. All the edges of the graph have certain probabilities of failing, i.e., of being absent. With this setup, for every node v in the network (excluding s, but including those in D) we need to find the probability that there exists at least one directed path from some source node to this node. This is the Network Reliability problem in one of its most general formulations [12, 13]. This is a well studied problem and is known to be $\#P$-hard [13]. Although no polynomial time algorithms to solve the problem are known, efficient algorithms for special graphs are known. An overview of the network reliability problem can be found in [14]. In the rest of this section we propose two straightforward methods to compute the probabilities of connectivity that we are interested in. We will also mention some techniques that can reduce the computation involved in these methods.

Assume we have a policy D. Consider a node v_i of the network. To find $R_D(v_i)$ we need to find $e_D(v_i)$. A bit is erased at node v_i if it is erased on all incoming links. With each edge (v_i, v_j) in the graph, associate a channel random variable $z(i, j)$. This takes the value 0 when a bit is erased and the value 1 when a bit is not erased. Thus, it is a Bernoulli random variable with probability $(1 - \epsilon_{i,j})$.

Consider all the directed paths from s to v_i. Let there be k_i paths. Denote the paths by $B_1, \ldots, B_{k_i}$. Let path B_j consist of l_j edges. We specify path B_j by writing in order the edges it traverses, i.e., with the sequence $((v_{j_1}, v_{j_2}), (v_{j_2}, v_{j_3}), \ldots, (v_{j_{l_j}}, v_{j_{l_j+1}}))$. We know that $s = v_{j_1}$ and $v_i = v_{j_{l_j+1}}$. Consider the set of vertices excluding v_i that are on path v_j, i.e., $\{v_{j_i} : i = 1, \ldots, l_j\}$. Some nodes in this set may belong to D, i.e., they are decoding nodes. In this case we know that they transmit the original codeword exactly. Let t be the largest index in this set such that v_{j_t} decodes. Therefore, v_i will not receive bit b along path B_j only if an erasure occurs on an edge that comes after v_{j_t} in the path. We associate with path B_j the product of the random variables that affect this, viz.,

$$Z_j = z(j_t, j_{t+1}) \cdot z(j_{t+1}, j_{t+2}) \cdot \cdots \cdot z(j_{l_j}, j_{l_j+1}).$$

This product is zero if one of the z random variables takes value zero, which, in turn, means that an erasure occurred on that edge.

Now, v_i sees an erasure only when none of the paths from s to itself manage to transmit the bit to it. Therefore, v_i sees an erasure when $Z_j = 0$ for *all* the paths $B_j, j = 1, \ldots, k_i$. Therefore we have

$$\begin{aligned} R_D(v_i) &= 1 - e_D(v_i) \\ &= 1 - P(\bigcap_{j=1}^{k_i}(Z_j = 0)) \\ &= P(\bigcup_{j=1}^{k_i}(Z_j \neq 0)) \end{aligned}$$

One way to evaluate this is by checking all possible combinations of values that the z variables can take and finding the total probability of those combinations that satisfy $\cup_{j=1}^{k_i}(Z_j \neq 0)$. This procedure has complexity $O(2^E)$. One observation that can make this procedure more efficient is the following – if we know that setting a certain subset of the z variables to 1 is enough to make the event $\bigcup_{j=1}^{k_i}(Z_j \neq 0)$ happen, then for every superset of this subset, setting all the z variables in that superset to 1 is also enough to make the event $\bigcup_{j=1}^{k_i}(Z_j \neq 0)$ happen. With this, we may have to check out fewer than the 2^E possible combinations of values for the z variables and reduce the complexity.

Another way to evaluate this is by using the Inclusion Exclusion Principle [11]. This gives us

$$P(\bigcup_{j=1}^{k_i} Z_j \neq 0) = \sum_{r=1}^{k_i} \sum_{1 \leq j_1 < \cdots < j_r \leq k_i} (-1)^{r+1} P(Z_{j_1} \neq 0, \ldots, Z_{j_r} \neq 0)$$

Since we have k_i paths, the above expression has $2^{k_i} - 1$ terms. A general term of the form $P(Z_{j_1} \neq 0, \ldots, Z_{j_r} \neq 0)$ can be evaluated by first listing all the z variables that occur in at least one of the r terms. Say these are $z(i_1, j_1), \cdots, z(i_q, j_q)$. Now $P(Z_{j_1} \neq 0, \ldots, Z_{j_r} \neq 0)$

is given by the product $(1-\epsilon_{i_1,j_1}) \times \cdots \times (1-\epsilon_{i_q,j_q})$. This procedure has complexity $O(E2^k)$ where k is the $\max_i k_i$. In this procedure, the complexity of listing all the variables in a certain set of r terms can be reduced by storing the lists that one makes for sets of $(r-1)$ terms and simply adding on the z terms from the r-th term to the appropriate list.

7 Algorithm to find Optimum Policy

In general, since we have $V-2$ relay nodes and each node has two options, viz., "forwarding" and "decoding and re-encoding", we have 2^{V-2} policies. To find the optimum policy we can analyze the rate for each of these policies and determine the one that gives us the best rate. This strategy of exhaustive search requires us to analyse 2^{V-2} policies.

Here, we propose a greedy algorithm that finds the optimum policy D which maximises the rate. This algorithm require us to analyze at most $V-2$ policies. In the next section we will give a proof of correctness for this algorithm.

---

1. Set $D = \emptyset$.
2. Compute $R_D(v_i)$ for all $v_i \in \mathcal{V}$. (Use techniques of Section 6.)
   Find $R_D = \min_{v_i \in D \cup \{d\}} R_D(v_i)$.
3. Find $M = \{v_i | v_i \notin \{s,d\} \cup D, R_D \leq R_D(v_i)\}$.
4. If $M = \emptyset$, terminate. D is the optimal strategy.
5. If $M \neq \emptyset$, find the largest $D' \subseteq M$ such that $\forall v \in D'$, $R_D(v) = \max_{v_i \in M} R_D(v_i)$.
   Let $D = D \cup D'$.
   Return to 2.

At each stage of the algorithm, we look for nodes that are seeing a rate as good as or better than the current rate of network operation. If there are no such nodes, the algorithm terminates. If there are such nodes, we choose the best from among them. Thus, in every iteration, the nodes we add are such that they do not put additional constraints on the rate of the network. Therefore, the rate of the network can only increase in successive iterations.

Note that since we assume a finite network, this algorithm is certain to terminate. Also, since D cannot have more than $(V-2)$ nodes, the algorithm cycles between steps 2 to 5 at most $(V-2)$ times. This is significantly faster than the strategy of exhaustive search that requires us to analyze 2^{V-2} policies.

The complexity of the algorithm depends on how fast the computation of $R_D(v_i)$ can be done. We have seen techniques for this computation in Section 6.

8 Analysis of the Algorithm

We first prove a Lemma regarding the effect of decoding at a particular node on the rates supportable at other nodes.

Lemma 1. *When node v is added to the decoding set D, the only nodes v_i that may see a change in rate are $v_i > v$. This change can only be an increase in rate, i.e., $\forall v_i$ such that $v_i > v$ we have, $R_D(v_i) \leq R_{D\cup\{v\}}(v_i)$. Every other node v_j is unaffected, i.e., $R_D(v_j) = R_{D\cup\{v\}}(v_j)$.*

Proof. We give separate proofs for the Gaussian Network and the Erasure Network.
Gaussian Network : Recall the computation of $\rho_D(v_i)$ described in Section 6.2. The computation for $Y_D(v_i)$ depends only on (some of) the $Y_D(v_j)$ where (V_j, v_i) is an edge. Therefore, inductively, it is clear that $Y_D(v_i)$ (and hence $\rho_D(v_i)$) depends only on the nodes v where $v < v_i$. Therefore, the only nodes that are affected when v changes its operation (from "forwarding" to "decoding and re-encoding") are $v_i > v$. The rest are unaffected.

Consider one of the $X_D(v_j)$ terms in (4). Note that each of these are of power P of which some power is the signal power and the rest is the noise power. If v_j changes its operation from forwarding to decoding, $X_D(v_j) = X(v_1)$, i.e., the signal power increases to P and the noise power goes to 0. If v_j is forwarding, $X_D(v_j)$ is only a scaled version of $Y_D(v_j)$. Since it is always of power P, if the SNR at node v_j increases, the signal power in $X_D(v_j)$ increases while the noise power decreases. From (4) we see that in both these cases, there is an increase in the signal power of $Y_D(v_i)$ and a decrease in the noise power. This implies an increase in the SNR.

Therefore, when v is added to D, by induction, for all nodes $v_i > v$, the SNR, if affected, can only undergo an increase. Naturally, we have the same conclusion for the rate.
Erasure Network : Recall the computation of $e_D(v_i)$ described in Section 6.3. Consider the product Z_j corresponding to path R_j. Adding v to D will not affect Z_j in any way if v does not occur on R_j. Therefore the only nodes that may be affected are v_i such that v is on some path from s to v_i, i.e., $v < v_i$. For these nodes, consider the product Z_j in the two cases when the policy is D and when the policy is $D \cup \{v\}$. The only way Z_j may be affected is if v occurs on the path R_j after v_{j_t} (where v_{j_t} is as defined in Section 6.3. In this case, Z_j for policy $D \cup \{v\}$ will have fewer terms than Z_j for policy D. This will happen for every path on which v occurs. Since $e_D(v_i) = P(\cap_j(Z_j = 0))$, we have that $e_{D\cup\{v\}}(v_i) \leq e_D(v_i)$. This means that $R_D(v_i) \leq R_{D\cup\{v\}}(v_i)$ for all $v_i > v$ and $R_D(v_j) = R_{D\cup\{v\}}(v_j)$ for all other nodes. □

This Lemma tells us that adding nodes to the set of decoding nodes can only increase the rate to other nodes. While this sounds like a good thing, it also puts a constraint on the rate as indicated by (2). It is this tradeoff that our algorithm seeks to resolve by finding the optimal set of decoding nodes.

8.1 Proof of Optimality

Theorem 2. *The algorithm of Section 7 gives us an optimal set of decoding nodes.*

Proof. Let S be an optimal set of decoding nodes. Let D be the set returned by the algorithm. We will prove that $R_D \geq R_S$. Then, since S is optimal, we will have $R_D = R_S$.
We prove $R_D \geq R_S$ in two steps. First we show that $R_{S\cup D} \geq R_S$. Then we show that $S \cup D - D = \emptyset$, i.e., $S \cup D = D$. This will complete the proof.
Step 1: In every iteration, the algorithm finds subsets D' and adds them to D. Denote by

D_i the subset that is added to D in the i-th iteration. Assuming the algorithm goes through m iterations, we have $D = D_1 \cup \cdots \cup D_m$ where the union is over disjoint sets. In the algorithm, when D_i is added to D, all the nodes in it are decoding at the same rate which is $R_{D_1 \cup \cdots \cup D_{i-1}}(v)$ for $v \in D_i$. We will call this rate $R_{\text{algo},i}$. Consider the smallest i such that $D_i \nsubseteq S$, i.e., D_i is not already entirely in S.
Claim: Adding D_i to S does not decrease the rate, i.e., $R_{S \cup D_i} \geq R_S$.

Proof. Because of the acyclic assumption on the graph, we will have some nodes $v \in S$ such that $\forall u (\neq v) \in S$ we either have $v < u$ or v and u are incomparable. Let L be the set of all such nodes v. Note that by Lemma 1 node v supports a rate $R_S(v) = R_\emptyset(v)$. By 3, for every $v \in L$ we have the necessary condition

$$R_S \leq R_S(v) = R_\emptyset(v). \tag{5}$$

Also note that $D_1, \ldots, D_{i-1}$ are all in S and by the definition of L and Lemma 1 we have

$$R_\emptyset(v) = R_{D_1 \cup \cdots \cup D_{i-1}}(v). \tag{6}$$

We now consider two cases.

- If for some $w \in L$, we also have $w \in D_i$, then from (5) and (6) we have $R_S \leq R_S(w) = R_\emptyset(w) = R_{D_1 \cup \cdots \cup D_{i-1}}(w) = R_{\text{algo},i}$.
- On the other hand, if none of the nodes in L are in D_i, pick any node $v \in L$. We have $v \notin D_i$. We now consider two subcases.
  - Let $v \notin D_1, \ldots, D_{i-1}$. We note from Steps 3 and 5 of the algorithm that it picks out from the set of nodes not in D, all nodes with the best rate. Since v does not get picked, we have $R_{\text{algo},i} > R_{D_1 \cup \cdots \cup D_{i-1}}(v)$. This alongwith (5) and (6) gives us $R_S \leq R_{D_1 \cup \cdots \cup D_{i-1}}(v) < R_{\text{algo},i}$.
  - The other possibility is that $v \in D_1 \cup \ldots \cup D_{i-1}$. Since the D_is are disjoint, there is a unique j such that $v \in D_j$. Since $v \in L$, by Lemma 1, $R_{\text{algo},j} = R_{D_1 \cup \cdots \cup D_{j-1}}(v)$. With the same argument as that for (6), we have $R_\emptyset(v) = R_{D_1 \cup \cdots \cup D_{j-1}}(v)$. But since the algorithm never decreases rate from one iteration to the next, we have $R_{\text{algo},i} \geq R_{\text{algo},j}$. Putting these together we get $R_{\text{algo},i} \geq R_{\text{algo},j} = R_{D_1 \cup \cdots \cup D_{j-1}}(v) = R_\emptyset(v)$. With (5) this gives us $R_S \leq R_S(v) = R_\emptyset(v) \leq R_{\text{algo},i}$.

Therefore, in every case, we have shown that $R_S \leq R_{\text{algo},i}$. This implies that adding the rest of the nodes from D_i to S will not put additional constraints on R_S and hence cannot decrease the rate. Therefore we have $R_{S \cup D_i} \geq R_S$. □

Since S is optimal, this proves that $S \cup D_i$ also achieves optimal rate. We can now call this set S and for the next value of i such that $D_i \nsubseteq S$, we can prove that $S \cup D_i$ has optimal rate. Continuing like this we have that $S \cup D$ is optimal, or, in other words, $R_{S \cup D} \geq R_S$.
Step 2: Next we wish to show that $S \subseteq D$, i.e., $S \cup D - D = \emptyset$. Let us assume the contrary. Let $T = S \cup D - D$. Therefore, $T \cap D = \emptyset$ but $T \subseteq S$. Thus, $D \cup S = D \cup T$ where D and T are disjoint. Consider $v \in T$ such that $\forall u (\neq v) \in T$ we either have $v < u$ or v and

u are incomparable. We have $R_{D\cup T}(v) = R_{D\cup S}(v)$. By Lemma 1, $R_{D\cup T}(v) = R_D(v)$. Also, the constraint of (2) tells us that $R_{D\cup S} \leq R_{D\cup S}(v)$. Finally, note that since the algorithm terminates without adding v to D, we have $R_D > R_D(v)$. Putting these inequalities together we have $R_D > R_D(v) = R_{D\cup S}(v) \geq R_{D\cup S}$. But this contradicts the fact that $S \cup D$ is optimal. Thus we have $S \subseteq D$, i.e., $S \cup D = D$.
From Steps 1 and 2 we have $R_D \geq R_S$. But since S was an optimal policy, D is also an optimal policy. This proves that the algorithm does indeed return an optimal set of decoding nodes.
The only case in which this proof does not go through is when the algorithm returns $D = \emptyset$ and $S \neq \emptyset$. In this case, consider node $v \in L \subseteq S$, where L is as defined earlier. Since the algorithm does not pick up v, we have $R_\emptyset > R_\emptyset(v)$. But $R_S \leq R_S(v) = R_\emptyset(v)$ from (5). Thus, $R_S < R_\emptyset$. But this contradicts the optimality of S. Therefore, if there exists an optimal, non-empty S, the algorithm cannot return an empty D. □

Corollary 1. *The algorithm of Section 7 returns the largest optimal policy D.*

Proof. In the proof above, we have shown that for any optimal policy S, we have $S \subseteq D$. This implies that D is the largest optimal policy. □

9 Examples

In this section we present some examples of networks and show how the algorithm runs on them.

9.1 Multistage relay networks

In Figure 4(a) we have depicted a multistage relay network. In this we have a single source and destination and k layers of relay nodes. The i-th layer consists of l_i nodes. Between the i-th and the $(i+1)$-th layer we have a complete bipartite graph where all the edges are directed from the i-th layer to the $(i+1)$-th. We assume that each of these edges has erasure probability ϵ_i. The source is connected to all the nodes in the first layer by erasure channels with erasure probability ϵ_0 and all the nodes in the k-th layer are connected to the destination by erasure channels with erasure probability ϵ_k. We will also call d the $(k+1)$-th layer and $l_{k+1} = 1$.

Because of the structure of this network, finding the rate under a particular policy is easier than indicated in Section 6.3. Denote by $Q_{i,j}$ the probability that in layer i there are j nodes that do not see an erasure. This defines $Q_{i,j}$ for $i = 1, 2, \ldots, (k+1)$ and $j = 0, 1, \ldots, l_i$. With this, for $i = 1$ we obtain,

$$Q_{1,k} = \binom{l_1}{k} \epsilon_0^{l_1-k}(1-\epsilon_0)^k. \tag{7}$$

For $i > 1$, we can show the recursion below.

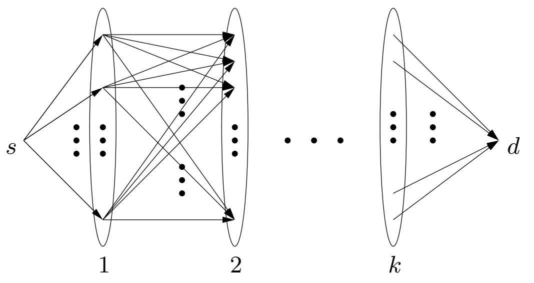
$$Q_{i,k} = \binom{l_i}{k} \sum_{t=0}^{l_{i-1}} \epsilon_{i-1}^{t(l_i-k)}(1-\epsilon_{i-1}^t)^k Q_{i-1,t}. \tag{8}$$

Denote by e_i the probability that the at least one node in the i-th layer does not see an erasure. We can show that

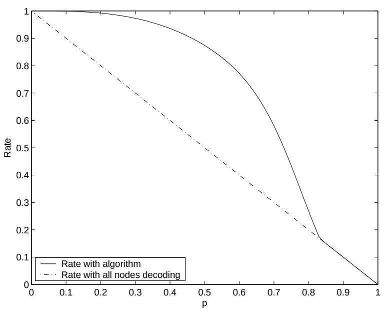
$$e_i = \sum_{k=0}^{l_i} Q_{i,k} \left(1 - \frac{k}{l_i}\right)$$

Note that, by symmetry, whenever a node decides to decode, all the nodes in that layer decode. When layer i decides to decode, we set $Q_{i,l_i} = 1$ and $Q_{i,j} = 0$ for $j \neq l_i$ and continue with the recursion of (8) for the other layers. This also extends to the case when more than one layer decodes.

Now, our algorithm proceeds as before, but operates on layers rather than nodes and the effective erasure probability at layer i is e_i.



(a) Model of a multistage relay network



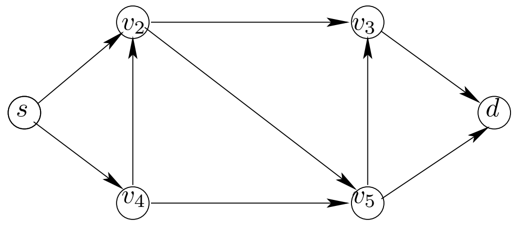
(b) Maximum rate as given by the algorithm

Figure 4: Multistage relay network

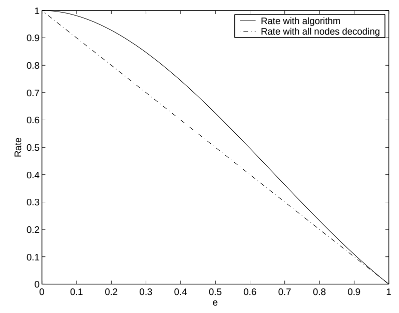
As an explicit example, consider a multistage relay network with four layers between the source and destination. Let $l_1 = 3, l_2 = 6, l_3 = 4, l_4 = 5$ and $\epsilon_0 = p, \epsilon_1 = p^2, \epsilon_2 = p, \epsilon_3 = p^3, \epsilon_4 = p$ where p is any number in the interval $[0, 1]$. For a fixed value of p, we can find the optimum policy for the network and this will give us the optimal rate. Figure 4(b) shows this optimal rate for the parameter p going from 0 to 1. This is not a smooth curve. The point where the right and left derivatives do not match is where either the optimum policy or the rate-determining layer changes. The rate with all nodes decoding has also been plotted. It is $1 - p$ and we see that the algorithm gives us dramatically higher rates.

9.2 Erasure network with four relay nodes

Consider the relay network of Figure 5(a). All the links have the same erasure probability p, where p is any number between 0 and 1. For this range of p, the algorithm has been used to find the optimum rates and policies. The rate is plotted in Figure 5(b). Throughout, the optimal policy is $D = \{v_2, v_3, v_5\}$. The rate with all nodes decoding is $1 - p$ and is also plotted. As expected, the algorithm outperforms the all-decoding scheme.



(a) Erasure network with four relay nodes.

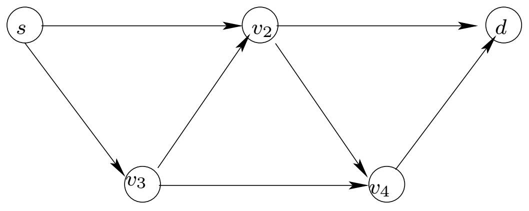


(b) Maximum rate as given by the algorithm

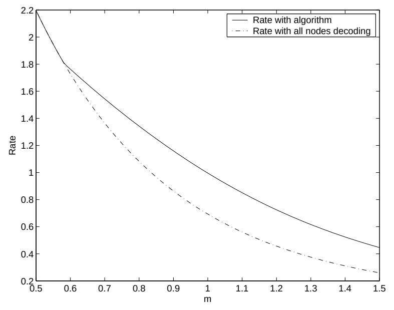
Figure 5: Erasure network with four relay nodes

9.3 Gaussian network with three relay nodes

In Figure 6(a) we see a Gaussian network with three relay nodes. We assume that each node is restricted to use power $P = 1$. Let the additive noise variances be $\sigma_2^2 = m, \sigma_3^2 = m^3, \sigma_4^2 = m^2, \sigma_5^2 = m^1$ where m can be an arbitrarily chosen real number. In Figure 6(b) we see the



(a) Gaussian network with three relay nodes.



(b) Maximum rate as given by the algorithm

Figure 6: Gaussian network with three relay nodes

rate returned by the algorithm for the optimal policy for $m \in [0.5, 1.5]$. The rate with all nodes decoding is also plotted. In the region $m \in [0.5, 0.58]$ we see that the optimal policy is infact that of decoding at all nodes and the two curves match. After that, the optimal policy changes and hence we see that the optimal rate curve is not smooth.

10 A Distributed Algorithm for the Optimal Policy

The algorithm as proposed in Section 7 requires that the network parameters (viz., noise variances or erasure probabilities) be known before the network operation begins so that the optimum policy is known beforehand. With the algorithm in its current form the nodes cannot determine for themselves if they should decode or forward. In this section we propose a scheme that can permit nodes to determine their own operation.

The algorithm works iteratively to converge to a rate. In each iteration, the rate of operation of the network is incremented or decremented depending on whether the previous transmission was successful or not. In every iteration, all the nodes get to decide their operation for themselves.

Let R^* be the maximum rate of the network. This is not known beforehand. We assume that parameters R, δ and N are known to all the nodes beforehand. The blocklength n is also predetermined and known to all the nodes. In addition, we require that the nodes have a common source of randomness so that they can generate the *same* random codebook individually. With this, consider the following algorithm.

---

1. All nodes generate the (same) codebook for rate R. They all set $k = 0$.

2. s transmits a randomly chosen codeword $X(v_1)$.

3. Every relay node v_i attempts to decode the received message $Y(v_i)$.
   If it can decode to a unique codeword, it transmits that codeword.
   Else, it forwards the received message (with appropriate scaling, for the Gaussian network).

4. The destination attempts to decode the received message.
   If it decodes to a unique codeword, it sends back bit 1 to all the other nodes to indicate this.
   Else, it sends back bit 0 to all other nodes.

5. All nodes increment k. $k = k + 1$.
   If transmitted bit was zero, all nodes set $R = R - \delta/2^k$.
   If transmitted bit was one, all nodes set $R = R + \delta/2^k$.

6. While $k \leq N$, go to Step 1.

Theorem 3. *If the maximum rate of the network, viz. R^* is in the range $[R - \delta, R + \delta]$, the algorithm above converges to it with an accuracy of $\frac{1}{2^N}$.*

Proof. The source starts by transmitting at rate R. Each relay node receives messages on all incoming links and decodes the message if it can. If it cannot, it simply forwards what it has received. With this procedure, nodes decide their own operation. (The order in which they decide this is a partial order in the sense defined in Section 6.1.) After the destination receives all its incoming messages, it tries to decode. If $R > R^*$, the destination will definitely not be able to decode. If $R \leq R^*$, we claim that the destination will be able to decode. This is because when a node decodes, it only improves the rates for other nodes. Also, note that an arbitrary node v decides whether to decode or not only after all the nodes before it in the partial order have already determined if the rate they can support is greater or smaller than R. Since, by Lemma 1, these are the only nodes that affect the rate for v and they decode whenever they can, node v always gets to see the best situation it can as far as rate R is concerned. This is true for the destination also.

Therefore, depending on whether the destination can decode or not, we can say if R^* is greater or smaller than R. If this bit of information is transmitted back to the source and other nodes, they can accordingly decide whether to increase or decrease the rate for the next transmission. Thus we have a decision tree of rates such that the ability or inability of the decoder tells us which path to traverse in that tree we can finally converge on a rate sufficiently close to the actual rate R. $\square$

This algorithm provides a very natural mode of network operation that obviates the need for a central agent to know the entire network and decide the optimum policy. Although some communication from the destination to the source and other nodes is required, this is minimal and should be easily possible in a practical network setting.

We mention that a sightly more sophisticated algorithm than the above can be devised that works for all values of R^*, rather than those in the interval $[R-\delta, R+\delta]$. We omit it in the interests of brevity.

11 Upperbounds on the maximum rate

The algorithms of Section 7 as well as Section 10 converge to the maximum rate possible with the decode/forward scheme, but we have no way of simply looking at the network and saying what this maximum rate will be. In this section, we present upperbounds on the rate achievable with the limited operations that we use in this paper.

11.0.1 Definitions

An $s-d$ *cut* is defined as a partition of the vertex set $\mathcal{V}$ into two subsets $\mathcal{V}_s$ and $\mathcal{V}_d = \mathcal{V} - \mathcal{V}_s$ such that $s \in \mathcal{V}_s$ and $d \in \mathcal{V}_d$. Clearly, an $s-d$ cut is determined simply by $\mathcal{V}_s$. For the $s-d$ cut given by $\mathcal{V}_s$, let the *cutset* $\mathcal{E}(\mathcal{V}_s)$ be the set of edges defined below

$$\mathcal{E}(\mathcal{V}_s) = \{(v_i, v_j) | (v_i, v_j) \in \mathcal{E}, v_i \in \mathcal{V}_s, v_j \in \mathcal{V}_d\}$$

Finally, we define $X(\mathcal{V}_s)$ and $Y(\mathcal{V}_s)$ as below.

$$X(\mathcal{V}_s) = \{v_i | (v_i, v_j) \in \mathcal{E}(\mathcal{V}_s)\}$$

$$Y(\mathcal{V}_s) = \{v_j | (v_i, v_j) \in \mathcal{E}(\mathcal{V}_s)\}$$

Thus $X(\mathcal{V}_s)$ and $Y(\mathcal{V}_s)$ denote the nodes transmitting and receiving messages across the cut respectively.

11.0.2 Upperbound for Gaussian networks

For Gaussian networks, it is evident that making the additive noise zero at certain nodes can only increase the maximum rate available at d. In particular let us make the additive noise zero at all nodes except $Y(\mathcal{V}_s)$. Therefore, the received messages (and the transmitted messages) at all nodes in $\mathcal{V}_s$ are exactly the same as that transmitted by the source. Now, if we permit the nodes in $Y(\mathcal{V}_s)$ to decode co-operatively, we will get an upperbound on the rate that the destination can get.

Note that the SNR at node $v_j \in Y(\mathcal{V}_s)$ is $\gamma_I^2(j)P/\sigma_j^2$ where $\gamma_I(i)$ is the number of edges coming in to node v_i. Since our codebook and noise are Gaussian distributed, the optimum scheme for decoding co-operatively is taking a suitable linear combination of received messages and then decoding that. For optimal decoding, we find the linear combination that gives us the best SNR. It is easy to show that the best SNR possible is the sum of the SNRs seen by each node in $Y(\mathcal{V}_s)$.

Therefore an upperbound on the rate is

$$R \leq \log\left(1 + P \sum_{v_j \in Y(\mathcal{V}_s)} \frac{\gamma_I^2(j)}{\sigma_j^2}\right)$$

for every cut $\mathcal{V}_s$.

11.0.3 Upperbound for Erasure networks

For erasure networks, it is evident that making certain links perfect, i.e., with zero erasure probability, can only improve the performance. Therefore we can obtain an upperbound on the rate by making all edges other than those in $\mathcal{E}(\mathcal{V}_s)$ perfect. With this all the received (and transmitted) messages in $\mathcal{V}_s$ are exactly the same as the codeword transmitted by the source. Now, it is clear that the rate at which the nodes in $Y(\mathcal{V}_s)$ can decode co-operatively is an upperbound on the rate available at the destination.

Clearly, the effective erasure probability seen by the set of nodes $Y(\mathcal{V}_s)$ is $\prod_{(v_i,v_j)\in\mathcal{E}(\mathcal{V}_s)} \epsilon_{i,j}$ This gives us an upperbound on the rate. We have

$$R \leq 1 - \prod_{(v_i,v_j)\in\mathcal{E}(\mathcal{V}_s)} \epsilon_{i,j}$$

for every cut $\mathcal{V}_s$.

Note that in [10], a different min-cut upperbound is proposed and is shown to be achievable. This gives the capacity of the network under the assumption that the destination has perfect side-information regarding erasure locations from across the network. This is very different from the setup of this paper.

12 Conclusions and Further Questions

To summarize, we have shown that making each link error-free in a wireless network is sub-optimal. Thus a multi-hop approach, in which every relay node decodes the received message, is not necessarily the correct approach for wireless networks. We have proposed a scheme for network operation that is of use in practical networks and in which operations performed by a node are restricted to decoding and forwarding – both of which are common operations performed in a network setting. We have suggested an algorithm that finds the optimum policy without exhaustive search over an exponential number of policies and also proposed a method to converge to the correct policy without having a central decision-making agent.

The algorithm of Section 7 can find the maximum rate and optimum policy for any Gaussian or erasure wireless network. In addition, the bounds presented in Section 11 give us some idea of what sort of optimal rates to expect. However, we still do not know what sort of policies are optimal in what ranges of erasure probabilities or SNR. The examples of Section 3 suggest that when the links are poor (high erasure probabilites low SNR) it is better to decode. It would be interesting to know if this is true for general networks and what thresholds exist below which a certain operation is always preferred.

Also, Corollary 1 tells us that the algorithm returns the largest decoding set. Since decoding is the more costly of the two operations considered here, an algorithm that finds the smallest decoding set such that the maximum rate is obtained is of interest.

Finally, we note that the capacity of the erasure wireless networks with side-information was recently found [10] to be given by a min-cut expression. Comparing this capacity with the upperbounds we have presented, it is clear that our scheme does not reach those bounds. Of course, our schemes also do not assume the side-information regquired in [10]. Finding practical schemes that reach this capacity is an interesting avenue for future work.

References

[1] J. L. R. Ford and D. R. Fulkerson, *Flows in networks.* Princeton, NJ: Princeton University Press, 1962.

[2] T. M. Cover and J. A. Thomas, *Elements of information theory.* New York: Wiley, 1991.

[3] R. Ahlswede, N. Cai, S.-Y. R. Li, and R. W. Yeung, "Network information flow," *IEEE Transactions on Information Theory*, vol. 46, no. 4, pp. 1204–1216, 2000.

[4] S.-Y. R. Li, R. W. Yeung, and N. Cai, "Linear network coding," *IEEE Transactions on Information Theory*, vol. 49, no. 2, pp. 371–381, 2003.

[5] R. Koetter and M. Médard, "An algebraic approach to network coding," *IEEE/ACM Transactions on Networking*, vol. 11/5, pp. 782–795, Oct. 2003.

[6] T. Ho, R. Koetter, M. Médard, D. Karger, and M. Effros, "The benefits of coding over routing in randomized setting," *IEEE International Symposium on Information Theory, 2003*, vol. 1, pp. 122–130, 2003.

[7] M. Effros, M. Médard, T. Ho, S. Ray, D. Karger, and R. Koetter, "Linear network codes: a unified framework for source channel, and network coding," *invited paper to the DIMACS workshop on Network Information Theory*, 2003.

[8] S. Jaggi, P. Sanders, P. A. Chou, M. Effros, S. Egner, K. Jain, and L. Tolhuizen, "Polynomial time algorithms for multicast network code construction," *submitted to the IEEE Transactions on Information Theory.*

[9] A. F. Dana, M. Sharif, B. Hassibi, and M. Effros, "Is broadcast plus multiaccess optimal for Gaussian wireless networks with fading?," *37th Asilomar Conference on Signals, Systems and Computers, 2003.*

[10] A. F. Dana, R. Gowaikar, B. Hassibi, and M. Effros, "On the capacity of erasure wireless networks," *in preparation for submission.*

[11] J. H. van Lint and R. M. Wilson, *A Course in Combinatorics.* Cambridge University Press, 2001.

[12] J. S. Provan and M. O. Ball, "The complexity of counting cuts and of computing the probability that a graph is connected," *SIAM Journal of Computing*, vol. 12/4, pp. 384–393, 1983.

[13] L. G. Valiant, "The complexity of enumeration and reliability problems," *SIAM Journal of Computing*, vol. 8, pp. 410–421, 1979.

[14] H. L. Bodlaender and T. Wolle, "A note on the complexity of network reliability problems," *downloadable from http://www.cs.uu.nl/research/techreps/aut/thomasw.html*, 2003.