Mathematical methods in communication June 5th, 2012

Lecture 10

Lecturer: Haim Permuter *Scribe: Oron Sabag and Tal Kopetz*

I. Network Coding

Example 1 *(Simple Relay)* Consider the following bi-directional communication system. Nodes A and B make use of relay C in order to transfer messages X^n and Y^n respectively where the alphabet $\mathcal{X} = \mathcal{Y} = \{0,1\}$. What is the maximum bandwidth that can be used between nodes A and B, where each link has a bandwidth of *1KHz*.

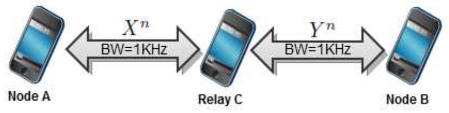


Fig. 1. This is a simple relay problem where bi-Directional communication between nodes A and B can be achieved only with the use of node C as a relay.

A simple approach to this problem would be that the maximum bandwidth between C to A and C to B is $500Hz$ each, since node C transmits the same message to both A and B. However, if $X^n \bigoplus Y^n$ is transmitted by relay C before transmitting to A and B, a *1KHz* bi-directional channel between A and B can be achieved.

Example 2 *(Multicast Network)* Consider the next communication network, where S wants to transmit 2 *bits*$\{b_1, b_2\}$ to node 5 and node 6 via relays nodes. Each link has the capacity of 1*bit*.

A few solutions may be applied here. One of them is as follows: Node S transmits b_1, b_2 to nodes 1 and 2 respectively. Nodes 1 and 2 transmit the received bits to nodes $\{3,5\}$ and $\{3,6\}$ respectively without any change. Transmitting $b_1 \bigoplus b_2$ from node 3 to node 4 will yield $\{b_1, b_1 \bigoplus b_2\}$ and $\{b_2, b_1 \bigoplus b_2\}$ in nodes 5 and node 6 respectively. In Fig. 4 we can see the suggested solution where node 5 receives $\{b_1, b_1 \bigoplus b_2\}$ and node 6

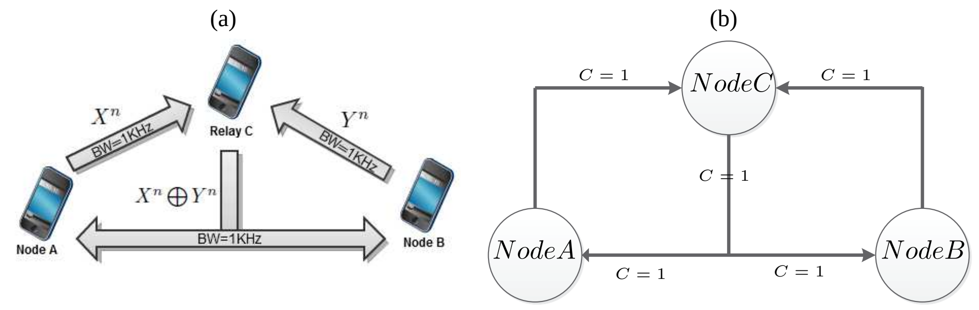


Fig. 2. (a) The suggested solution for the problem above. Relay C encodes $X^n \bigoplus Y^n$ and transmits this message to A and B. (b) Model for the given network where the capacity of each link is $C = 1$.

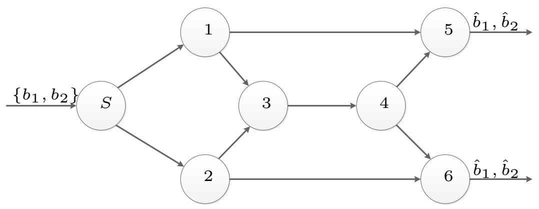


Fig. 3. This figure depicts a model for the problem described above.

receives $\{b_2, b_1 \bigoplus b_2\}$, XOR decoding at these nodes will achieve the required data in the destinations.

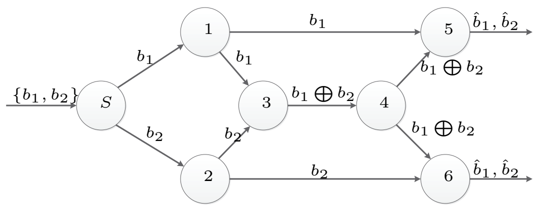


Fig. 4. This figure depicts the suggested solution.

Definition 1 *(Acyclic Network)* A network is acyclic if there is no a closed loop. Namely, there is no path that starts at a node and return to the same node.

Definition 2 *(Noiseless Relay Network)* Consider a acyclic noiseless relay network

$\mathcal{G}(\mathcal{N}, \mathcal{E}, \mathcal{C})$ where $\mathcal{N} = [1, ..., N]$ is a set of nodes, $\mathcal{E} \subset [1, ..., N] \times [1, ..., N]$ is a set of edges and $\mathcal{C} = \{C_{jk} \in \mathbb{R}^+ : (j,k) \in \mathcal{E}\}$. Each edge represents a noiseless communication link with a capacity of C_{jk}.

Example 3 In Fig. 5 we can see an example for noiseless relay network, where node 1 is the source and node N is the destination. The capacity links are defined as C_{jk} for an edge $(j,k) \in \mathcal{E}$.

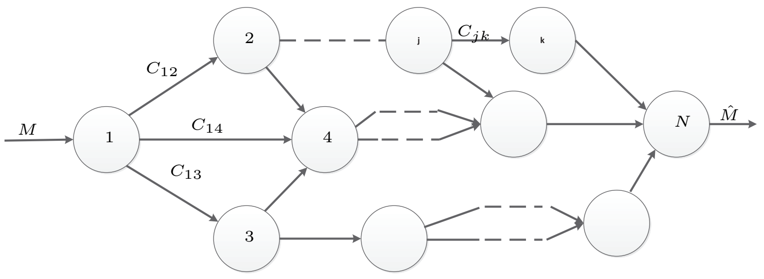


Fig. 5. Noiseless relay network with single source and single destination.

Definition 3 *(Codebook for noiseless relay network)* A $(2^{nR}, n)$ code for noiseless relay network $\mathcal{G}(\mathcal{N}, \mathcal{E}, \mathcal{C})$ consists of:

1) A source message set $[1, ..., 2^{nR}]$ and uniform distribution of message M.
2) A source encoder that assigns $m_{1j} \in [1, ..., 2^{nC_{1j}}]$ to each message $m \in [1, ..., 2^{nR}]$ for each edge $(1,j) \in \mathcal{E}$.

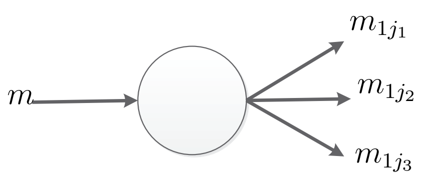


Fig. 6. Source encoder with input message m and output messages m_{1j_i}.

3) A set of $(N-2)$ encoders:
   Encoder k assigns an index $m_{kl} \in [1, ..., 2^{nC_{kl}}]$ to each received message $\{m_{jk} : (j,k) \in \mathcal{E}\}$ for each $(k,l) \in \mathcal{E}$.

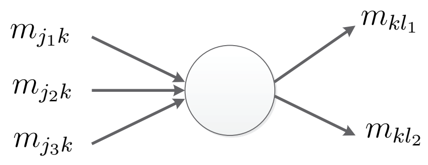


Fig. 7. Relay encoder with input messages m_{j_ik} and output messages m_{kl_i} .

4) A destination decoder that assigns a message $\hat{m} \in [1, ..., 2^{nR}]$ as a function of input messages $\{m_{jN} : (j, N) \in \mathcal{E}\}$.

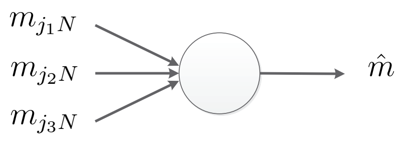


Fig. 8. Destination decoder with input messages m_{j_iN} and its outcome $\hat{m}(m_{j_iN} : (j_i, N) \in \mathcal{E})$.

Definition 4 *(Probability error of a code book)* The probability error of codebook $(2^{nR}, n)$ is $P_e{}^{(n)} = \Pr(M \neq \hat{M})$.

Definition 5 *(Achievable rate of a noiseless relay network)* A rate R is achievable if there exists a sequence of codes $(2^{nR}, n)$ such that $\lim_{n\to\infty} P_e{}^{(n)} = 0$.

Definition 6 *(Capacity of a noiseless relay network)* The capacity of $\mathcal{G}(\mathcal{N}, \mathcal{E}, \mathcal{C})$ is the supremum over the set of all achievable rates.

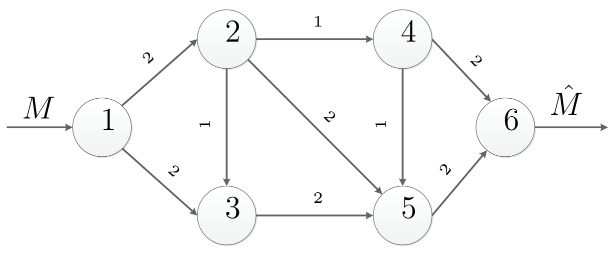
Theorem 1 *(Max-flow Min-cut)* The capacity of noiseless relay network is

$$C = \min_{\{S\subset\mathcal{N}:1\in S, N\in S^c\}} C(S), \tag{1}$$

where

$$C(S) = \sum_{\{(j,k)\in\mathcal{E}:j\in S, k\in S^c\}} C_{jk}. \tag{2}$$

Example 4 *(Min-cut in noiseless relay network)* Consider noiseless relay network, that is shown in Fig. 9. The capacity of this network is $C = 3$ with the minimum cut $S = \{1, 2, 3, 5\}$ and $S^c = \{4, 6\}$, the cut between S and S^c is depicted in Fig. 9. The



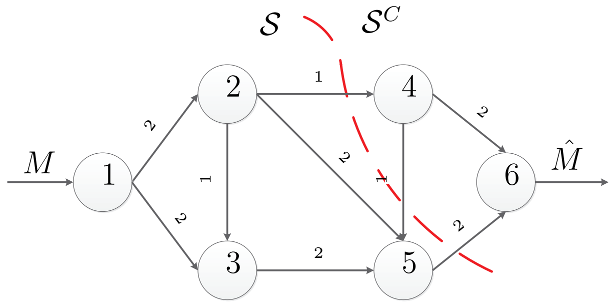


Fig. 9. Minimum cut S in a noiseless relay network.

capacity is achieved by sending 1 *bit* along the path $1 \to 2 \to 4 \to 6$ and another 2 *bits* along the path $1 \to 3 \to 5 \to 6$.

(Proof of converse for Theorem 1)

In this proof we want to show that $P_e{}^{(n)} = 0$ implies that $R \leq C$, where R is the rate used to transmit a message $m \in [1, ..., 2^{nR}]$ from source to destination. Let's fix $S \subset \mathcal{N}$ such that $1 \in S$ and $N \in S^c$, and let $T_1, T_2, ..., T_k$ be all messages on edges from S to S^c where $T_i \in [1, ..., 2^{nC_{jk}}]$ for some $j \in S, k \in S^c$.

$$\begin{aligned}
nR &\overset{(a)}{=} H(M) \\
&= I(M;\hat{M}) + H(M|\hat{M}) && (3)\\
&\overset{(b)}{\leq} I(M;\hat{M}) + (1 + P_e^{(n)} nR) && (4)\\
&\leq H(\hat{M}) + n\epsilon_n && (5)\\
&\overset{(c)}{\leq} H(T_1, T_2, ..., T_k) + n\epsilon_n && (6)
\end{aligned}$$

$$\overset{(d)}{\leq} \sum_{i=1}^{k} H(T_i) + n\epsilon_n \tag{7}$$

$$\overset{(e)}{\leq} \sum_{j\in S, k\in S^c} nC_{jk} + n\epsilon_n \tag{8}$$

$$\tag{9}$$

Where

a follows from the uniform distribution of $M \in [1, ..., 2^{nR}]$ as defined in (3).

b follows from Fano's inequality where $\log|M| = nR$.

c follows from Markov chain $M - (T_1, T_2, ..., T_k) - \hat{M}(T_1, T_2, ..., T_k)$.

d follows from chain rule of entropy where $H(T_i|T^{i-1}) \leq H(T_i)$.

e follows from upper bound on entropy for a finite alphabet, $H(T_i) \leq \log|\mathcal{T}_i|$.

A. Achievability via Ford-Fulkerson Algorithm

We will continue with finding the capacity of the Noiseless Relay Network. We have proved the converse, and now we will prove the achievability:
We want to show that

$$R = \min_{\mathcal{S}:1\in\mathcal{S}, N\in\mathcal{S}^c} C(S) \tag{10}$$

is achievable. In order to prove the achievability we would construct an algorithm called Ford-Fulkerson [1] that finds the maximum flow possible in a network.

Definition 7 (Flow) A *flow* from node 1 to node N is a function f that maps each edge $e \in \mathcal{E}$ to a non-negative real number.

$$f \quad : \quad \mathcal{E} \to R^+ \tag{11}$$

$$\mathcal{E} \quad := \quad \{e = (1, ..., N) \times (1, ..., N)\} \tag{12}$$

A flow must satisfy the following conditions:

1. $f(j,k) \leq C_{jk}$ (Capacity constraint)
2. $\sum_{e\in in(v)} f(e) = \sum_{e\in out(v)} f(e),\ v \neq 1, N$ (Conservation law).

Another Notation:

$$\sum_{(j,k)\in in(v)} f(j,k) = \sum_{(j,k)\in out(v)} f(j,k) \tag{13}$$

where $in(v)$ denote the edges that enter the node v and $out(v)$ denote the edges that exit the node v.

We assume in our lecture that the destination node has only edges that enter it.

Definition 8 (Value of flow) We defind the *value of flow* as follows:

$$val(f) = \sum_{k\in(2,\ldots,N)} f(1,k) \tag{14}$$

Every commodity that exits the source, $v = 1$, has to arrive at terminal N, hence

$$val(f) = \sum_{j\neq N} f(j,N) \tag{15}$$

The Maximum-Flow Problem: Our goal is to find the maximum value of the flow

$$\max_f val(f) \tag{16}$$

and we will see that this yields the capacity of the network. In order to find $\max_f val(f)$ we will introduce an algorithm. The algorithm will be iterative so that each iteration increases the flow. We will see that the algorithm indeed achieves the min-cut upper bound. First, we need to define a the *Residual Graph*:

Definition 9 (Residual Graph) Given a network G and a flow f, we define *residual graph* $G_f = (\mathcal{V}, \mathcal{E}_f, C_f)$ with respect to f as follows:

1) The set of the vertices is the same as in G.
2) For each $(j,k) \in \mathcal{E}$ if $f(j,k) < C_{jk}$ then introduce an edge $(j,k) \in \mathcal{E}_f$ with $C^f_{jk} = C_{jk} - f(j,k)$. We denote such an edge as a Forward Edge.
3) For $(j,k) \in \mathcal{E}$ of $f(j,k) > 0$ then include an edge from k to j $C^f_{kj} = f(k,j)$. We denote such an edge as a Backward Edge.

Example 5 *(Achievability using Algorithm 1)* Consider the network G and its flow f in Figure 10.a. We denote on each edge its flow and capacity. For instance, 1/2 means that $f = 1$ and $C = 2$. Figure 10.b depicts the corresponding residual graph G_f according to

Algorithm 1 Ford-Fulkerson Algo: achieves the maximum value of flow

```
Start with a flow, for instance, $f = 0$ for all edges.
loop
    Compute $G_f$.
    Find a path from node $1$ to node $N$ in $G_f$.
    Compute the bottleneck $\delta$ of the path in $G$.
    (The bottleneck of a path is the smallest capacity from all edges on the path.)
    for $i = 1 : n$ ($n$ - Number of edges on the path) do
        if $e_i$ is a forward edge ($e_i$ - Edge $i$ on the path) then
            $f(e_i) = f(e_i) + \delta$
        else if $e_i$ is a backward edge then
            $f(e_i) = f(e_i) - \delta$
        end if
    end for
    if $\delta = 0$ then
        Break
    end if
end loop
```

its definition. For instance, $f(2,3)$ is translated in to G_f as follows: The forward edge is $C_{23}^f = C_{23} - f(2,3) = 3 - 2 = 1$ while the backward edge is $C_{23}^f = f(2,3) = 2$.

In order to achieve maximum rate, we execute algorithm 1 on the system G. We start off with the red path and compute it's bottleneck δ=1. Now we go back to the network graph G and change the flow on the corresponding path in the following way: forward edges are increased by 1 and backward edges are decreased by 1.

Figure 11 depicts the modified network graph G and the corresponding residual graph G_f Respectively.

One can see that there are no paths from 1 to N on G_f. Thus, $\delta = 0$, the algorithm terminates and we obtain the maximum flow which equals the capacity of the network.

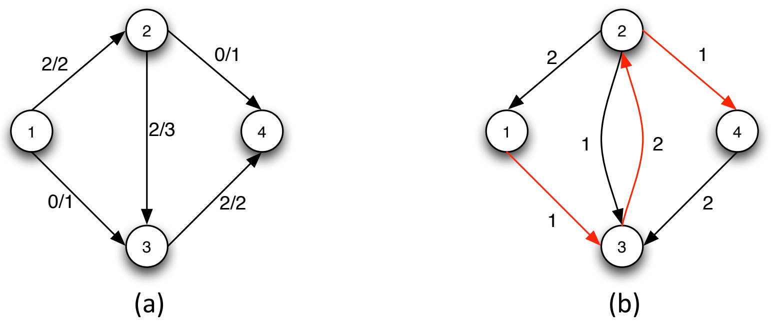


Fig. 10. Figure (a) depicts an arbitarary network flow G. Figure (b) depicts the corresponding residual graph G_f

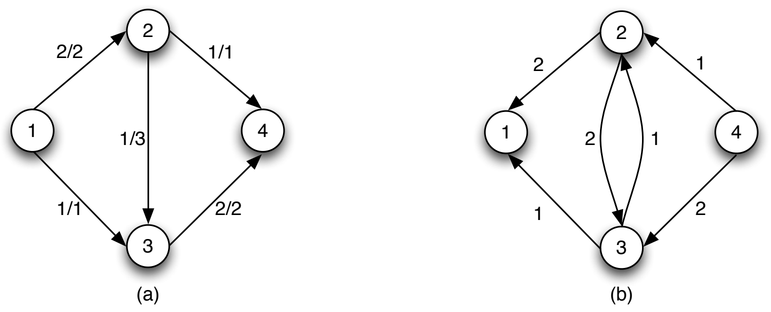


Fig. 11. Figure (a) depicts the modified network flow G after execution of the algorithm. Figure (b) depicts the corresponding residual graph G_f

Questions:

1) Let us denote f' as the new flow obtained after one iteration in Algorithm 1. Is f' a valid flow? Yes. In order to prove this, let us recall that a valid flow has to satisfy two constraints, the capacity constraint and the conservation constraint. Let us first prove that f' can't be larger then the Capacity. If $e(j, k)$ is a forward edge, then

$$\delta \leq C_e - f(e) \tag{17}$$

Now consider,

$$f'(e) = f(e) + \delta \tag{18}$$

$$\leq f(e) + (C_e - f(e)) \quad (19)$$

$$= C_e \quad (20)$$

Hence we obtained that $f'(e) \leq C_e$ and since $f'(e) = f(e) + \delta$ also $f'(e) \geq 0$. Now if $e(j,k)$ is a backward edge, then

$$\delta \leq f(e) \quad (21)$$

$$(22)$$

Consider,

$$f'(e) = f(e) - \delta \quad (23)$$

$$\geq f(e) - f(e) \quad (24)$$

$$= 0 \quad (25)$$

Hence $f'(e) \geq 0$ and since $f'(e) = f(e) - \delta$ and $f(e) \leq C_e$ then $f'(e) \leq C_e$. Thus we have proven that the flow isn't greater than the capacity and always non-negative.

The second condition of a flow is fulfilled due to the fact that all edges on the path are increased/decreased by the same δ. If a path from source to destination on G_f runs through node L, then all incoming and outgoing edges to node L are increased by the same δ thus the conservation law holds.

2) Is the new value of flow larger than the previous one? Yes, we have edges entering the destination node, so when we increase the flow on the path by δ, those edges will be increased by δ thus achieving a larger value of flow.

$$val(f') = val(f) + \delta > val(f) \quad (26)$$

3) Assume all capacities are integers, does the algorithm converges? From questions (1) and (2) we can derive that the algorithm does converge. The maximum value of flow is at most $\sum_{e \in out(S)} C_e = C_S$. Now, since the algorithm increases the value of the flow by at least one unit each time, it is clear that the algorithm can run for at most C_S iterations thus converges.

The algorithm terminates when there exists G s.t. no path from S to S^c exists in G_f. Before we conclude this with a theorem, we need the following lemma regarding the value of flow:

Lemma 1 (Value of flow) Let f be any $(1, N)$ flow and $1 \in S, N \in S^c$. Then

$$val(f) = \sum_{e \in out(S)} f(e) - \sum_{e \in in(S)} f(e) \tag{27}$$

where $out(S)$ denotes all edges with their tail in the group of nodes S and their head in the group of nodes S^c (all edges that exit the group of nodes S) and $in(S)$ denotes all edges with their head in the group of nodes S and their tail in the group of nodes S^c (all edges that enter the group of nodes S).

We defined flow in Definition (8)

$$val(f) = \sum_{k \in (2,\ldots,N)} f(1, k) \tag{28}$$

and we assume that no edges enter the source node, i.e,

$$\sum_{k \in (2,\ldots,N)} f(k, 1) = 0 \tag{29}$$

Then,

$$val(f) = \sum_{k \in (2,\ldots,N)} f(1, k) - \sum_{k \in (2,\ldots,N)} f(k, 1) \tag{30}$$

By the conservation law of flow, we know that for every $v \in S$ that

$$\sum_{e \in out(v)} f(e) - \sum_{e \in in(v)} f(e) = 0, \;\; v \neq 1 \tag{31}$$

Thus,

$$val(f) = \sum_{v \in S} \left[\sum_{e \in out(v)} f(e) - \sum_{e \in in(v)} f(e) \right] \tag{32}$$

Since only for node $v = 1$ we obtain a non-zero value (from defintion). Now, if an edge e has both end points in S, then $f(e)$ appears twice in the above sum, once with a positive sign and once with a negative sign i.e. the contribution of e to the above sum is 0. On the other hand if e only has its head in S then $f(e)$ appears only once with a negative sign and if e only has its tail in S then $f(e)$ appears only once with a positive sign. Of course if e has neither its head nor its tail in S then it does not participate in the sum at all. Therefore, we have

$$\sum_{v\in S}\left[\sum_{e\in out(v)} f(e) - \sum_{e\in in(v)} f(e)\right] = \sum_{e\in out(S)} f(e) - \sum_{e\in in(S)} f(e) \tag{33}$$

finally using (32) and (33) we obtain the result of the lemma in (27).

Now we conclude with the following theorem:

Theorem 2 (Achievability) If f' is a flow on network G and no path from S to S^c exists in G'_f then there exists a cut between (j,k) where $j \in S$ and $k \in S^c$ s.t.

$$\begin{aligned} f'(j,k) &= C_{jk} && (34)\\ f'(k,j) &= 0 && (35)\\ val(f') &= C && (36)\end{aligned}$$

Where

$$C = \sum_{j\in S, k\in S^c} C_{jk} \tag{37}$$

We first prove (34). Let us consider an edge $e = (j,k) \in G$ such that $j \in S$ and $k \in S^c$. Now lets Assume that $f(e) < c_e$. If so, then $e(j,k)$ would be a forward edge in $G_{f'}$. This would mean that we could extended a path from the source to k in G_f which contradicts our assumption that k belongs to S^c. Thus (34) holds.

Secondly, we prove (35). Let us consider an edge $e' = (k,j) \in G$ such that $j \in S$ and $k \in S^c$. Now lets Assume that $f(e') > 0$. If so, then $e(k,j)$ would be a backward path in $G_{f'}$. This would mean that we could extended a path from the source to k in G_f which contradicts our assumption that k belongs to S^c. Thus (35) holds. Now we calculate the Value of flow:

$$\begin{aligned} val(f) &\overset{(a)}{=} \sum_{e\in out(S)} f(e) - \sum_{e\in in(S)} f(e) && (38)\\ &\overset{(b)}{=} \sum_{e\in out(S)} C_e - 0 && (39)\end{aligned}$$

$$= \quad C \tag{40}$$

where (a) follows from (27) and (b) follows from (34) and (35) which were proved.

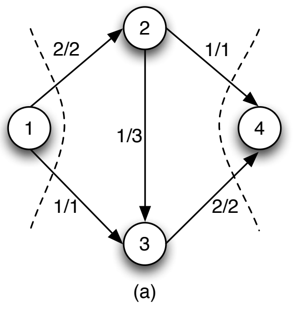


Fig. 12. The modified network flow G after execution of the algorithm. The figure also depicts the cut at the source or destination in order to derive C.

Comment: Once the algorithm is terminated we can easily find the value of the flow which equals to the capacity. We just need to consider the cut at the source (where we separate the source from the rest of the network) or at the cut at the destination and obtain the flow of the network. In Example 1, we cut at the source as shown in Figure 12 and derive that $C = 3$.

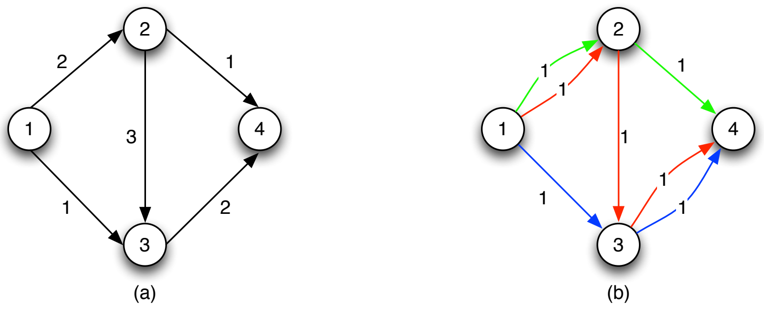


Fig. 13. Figure (a) depicts an arbitrary network G with integer links. Figure (b) illustrates transfering the network G into a network with links of 1.

We would like to emphasis the fact that flow is transmitted through disjoint paths.

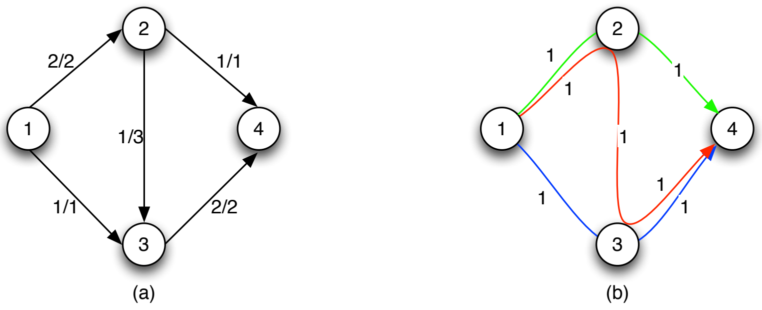


Fig. 14. Figure (a) depicts network G and a flow f. Figure (b) depicts the same network G only split in to Disjoint Paths.

Definition 10 (Disjoint Paths) A set of paths is said to be edge disjoint if no edge is common between any two paths.

Example 6 (Disjoint Paths) Given the network G in Figure 13.a, we can divide each edge to f edges of $f' = 1$ and receive the Disjoint Paths as in Figure 14.b. To do so, we need first to consider an equivalent network where each edge is with capacity link of 1.

II. Noiseless Multicast Network(one source to many destinations)

Now we consider a multicast extension of the noiseless relay network. The source initiates the transmission of message M to a set of destinations $\mathcal{D}$.

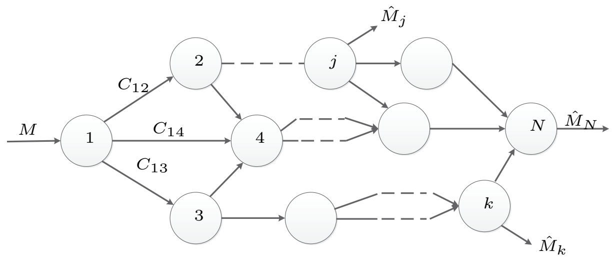


Fig. 15. Noiseless multicast network.

Definitions of achievability and capacity in noiseless multicast network are defined the same as for the single destination case; whereas, the probability of error is defined as ${P_e}^{(n)} = \{\Pr(M \neq \hat{M})$ for some $j \in \mathcal{D}\}$.

Exercise 1 (All nodes are destinations) Prove that if $\mathcal{D} = [2, ..., N]$, then routing-only will achieve the upper bound of

$$C \leq \min_{\{S \subset \mathcal{N}: 1 \in S, j \in S^c\}} C(S) \forall j \in \mathcal{D}. \tag{41}$$

Theorem 3 The capacity of noiseless multicast network $\mathcal{G}(\mathcal{N}, \mathcal{E}, \mathcal{C})$ is

$$C = \min_{i \in \mathcal{D}} \min_{S \subset \mathcal{N}, 1 \in S, i \in S^c} C(S) \tag{42}$$

where

$$C(S) = \sum_{(j,k) \in \mathcal{E}, j \in S, k \in S^c} C_{jk}. \tag{43}$$

proof of converse The converse for Theorem 3 follows from the case of one source and one destination. From the one source and one destination converse in Theorem 1 it follows that for each destination $j \in \mathcal{D}$

$$C \leq \min_{S \subset \mathcal{N}, 1 \in S, j \in S^c} C(S). \tag{44}$$

This inequality stands since if there exists a greater achievable rate we would find it for the single destination case. By minimizing this group of upper bounds we conclude that the rate for a multicast network is bounded by

$$C \leq \min_{i \in \mathcal{D}} \min_{S \subset \mathcal{N}, 1 \in S, j \in S^c} C(S) \tag{45}$$

We would next prove the achievability using linear network coding.

III. Linear Network Coding

For simplicity, we first consider noiseless multicast network with integer link capacities, represented by $\mathcal{G}(\mathcal{N}, \mathcal{E})$ with links of 1*bit* capacity. Therefore, each link of the multigraph $\mathcal{G}$ can carry *n* bits of information (a symbol from $\mathbb{F}_{2^n}$) per *n*-transmission block.

Furthermore, we assume that R is an integer so that the message can be represented as $M = [M_1, ..., M_R]$ with $M_j \in \mathbb{F}_{2^n}, j \in [1, ..., R]$.

Example 7 Let's assume $R = 3$ so the equivalent multigraph of network $\mathcal{G}$ can be shown as:

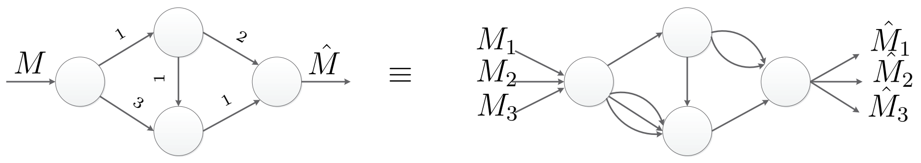


Fig. 16. Transformation of noiseless multicast network into its equivalent multigraph Network.

Definition 11 *(Edges vectors for node k)* Given a network modeled by multigraph $(\mathcal{N}, \mathcal{E})$, we define the set of outgoing edges from a node $k \in \mathcal{N}$ by $out(k)$ and the set of all incoming edges to a node $k \in \mathcal{N}$ by $in(k)$.

Definition 12 *(Definitions for linear code)* For this setup, a $(2^{nR}, n)$ linear code consists of:

1) A message set $\mathbb{F}_{2^n}^R$, each message is represented by a vector in the R-dimensional vector space over the finite field $\mathbb{F}_{2^n}$.
2) A linear source encoder that assigns an index tuple $m(out(1)) := \{m_e \in \mathbb{F}_{2^n} : e \in out(1)\}$ to each $(m_1, ..., m_R) \in \mathbb{F}_{2^n}^R$ via a linear transformation with coefficient $\alpha_{jk} \in \mathbb{F}_{2^n}$.

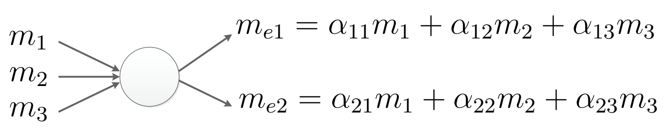
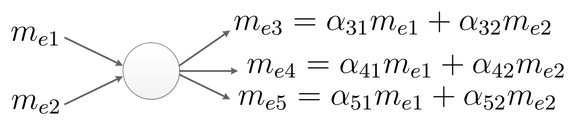

m_1
m_2
m_3
$m_{e1} = \alpha_{11}m_1 + \alpha_{12}m_2 + \alpha_{13}m_3$
$m_{e2} = \alpha_{21}m_1 + \alpha_{22}m_2 + \alpha_{23}m_3$

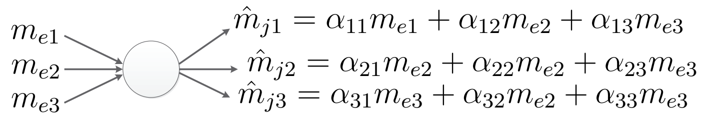
Fig. 17. A source encoder with R=3.

3) A set of linear relay encoders: Encoder *k* assigns an index tuple $m(out(k))$ to each $m(in(k))$ via a linear transformation.
4) A set of linear decoders,the j decoder $\{j : j \in \mathcal{D}\}$ assigns $\hat{m}_j^R$ to each $m(In(j))$.



m_{e1}
m_{e2}
$m_{e3} = \alpha_{31}m_{e1} + \alpha_{32}m_{e2}$
$m_{e4} = \alpha_{41}m_{e1} + \alpha_{42}m_{e2}$
$m_{e5} = \alpha_{51}m_{e1} + \alpha_{52}m_{e2}$

Fig. 18. A relay encoder with $|in(j)| = 2$ and $|out(j)| = 3$.



m_{e1}
m_{e2}
m_{e3}
$\hat{m}_{j1} = \alpha_{11}m_{e1} + \alpha_{12}m_{e2} + \alpha_{13}m_{e3}$
$\hat{m}_{j2} = \alpha_{21}m_{e2} + \alpha_{22}m_{e2} + \alpha_{23}m_{e3}$
$\hat{m}_{j3} = \alpha_{31}m_{e3} + \alpha_{32}m_{e2} + \alpha_{33}m_{e3}$

Fig. 19. This figure depicts an example of a linear decoder in source.

Thus, for each destination node $j \in \mathcal{D}$ a linear code induces a linear transformation

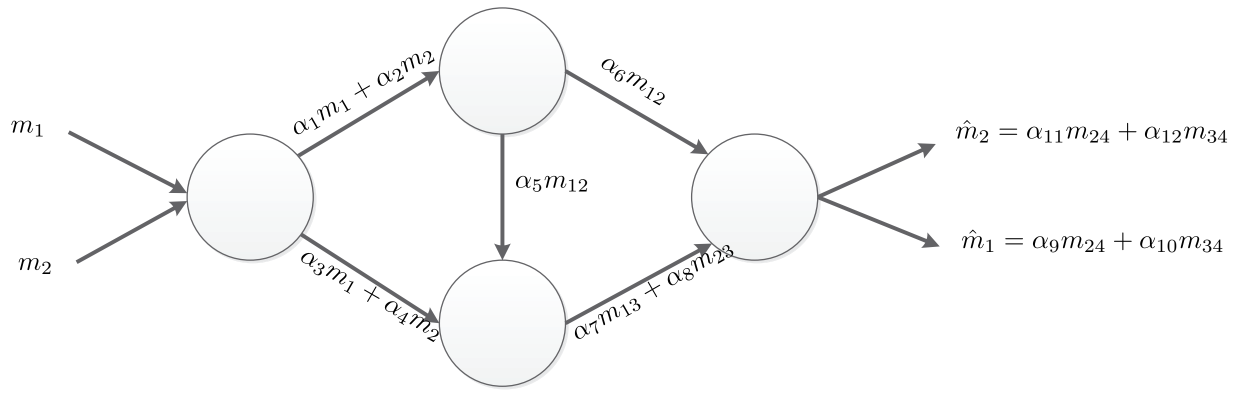
$$\hat{m}_j^R = A_j(\bar{\alpha})m^R, \tag{46}$$

where $\bar{\alpha}$ is the coefficients vector of $j \in \mathcal{D}$.

The rate R is achievable with zero error if there exists an $n \in \mathcal{N}$ and $\bar{\alpha}$ such that $A_j(\bar{\alpha}) = I_R \; \forall j \in \mathcal{D}$.

Note that any invertible $A_j(\bar{\alpha})$ is sufficient for restoration of m^R by multiplying $A_j^{-1}(\bar{\alpha})$ with $\hat{m}_j^R$.

Example 8 Let's consider the multicast network depicted in Fig. 20 with $\mathcal{D} = \{4\}$: One



m_1
m_2
$\alpha_1 m_1 + \alpha_2 m_2$
$\alpha_3 m_1 + \alpha_4 m_2$
$\alpha_5 m_{12}$
$\alpha_6 m_{12}$
$\alpha_7 m_{13} + \alpha_8 m_{23}$
$\hat{m}_2 = \alpha_{11} m_{24} + \alpha_{12} m_{34}$
$\hat{m}_1 = \alpha_9 m_{24} + \alpha_{10} m_{34}$

Fig. 20. Example for linear network coding with R=2.

can see that linear code book with $R = 2$ induces the linear transformation:

$$\begin{pmatrix} \hat{m}_1 \\ \hat{m}_2 \end{pmatrix} = \begin{pmatrix} \alpha_9 & \alpha_{10} \\ \alpha_{11} & \alpha_{12} \end{pmatrix} \begin{pmatrix} m_{24} \\ m_{34} \end{pmatrix} \tag{47}$$

$$= \begin{pmatrix} \alpha_9 & \alpha_{10} \\ \alpha_{11} & \alpha_{12} \end{pmatrix} \begin{pmatrix} \alpha_6 & 0 \\ \alpha_5\alpha_8 & \alpha_7 \end{pmatrix} \begin{pmatrix} m_{12} \\ m_{13} \end{pmatrix} \tag{48}$$

$$= \begin{pmatrix} \alpha_9 & \alpha_{10} \\ \alpha_{11} & \alpha_{12} \end{pmatrix} \begin{pmatrix} \alpha_6 & 0 \\ \alpha_5\alpha_8 & \alpha_7 \end{pmatrix} \begin{pmatrix} \alpha_1 & \alpha_2 \\ \alpha_3 & \alpha_4 \end{pmatrix} \begin{pmatrix} m_1 \\ m_2 \end{pmatrix} \tag{49}$$

We can see that for this multicast network,

$$A_j(\bar{\alpha}) = \begin{pmatrix} \alpha_9 & \alpha_{10} \\ \alpha_{11} & \alpha_{12} \end{pmatrix} \begin{pmatrix} \alpha_6 & 0 \\ \alpha_5\alpha_8 & \alpha_7 \end{pmatrix} \begin{pmatrix} \alpha_1 & \alpha_2 \\ \alpha_3 & \alpha_4 \end{pmatrix}. \tag{50}$$

Restoration of the original messages can be done only by multiplying $A_j^{-1}(\bar{\alpha})$ with $\hat{m}_j^R$. $A_j(\bar{\alpha})$ is inveritable if $|A_j(\bar{\alpha})| \neq 0$, therefore we need to find a coefficients vector $\bar{\alpha}$ such that $|A_j(\bar{\alpha})| \neq 0$.

REFERENCES