

Lecture 9

As we saw from the previous lecture a new packet format is introduced to solve the problem of communicating the multiplicative coefficients, however this comes with a loss in bandwidth by adding a $1 \times h$ overhead field in the packet. The total overhead will be $\frac{h}{N}$ where h is the number of sources at the source node and N is the number of symbols communicated in a single packet. We can see that the overhead can be reduced by choosing large N (see Figure 1).



Figure 1: Packet Format

With this packet format we have solved the problem of communicating the multiplicative coefficients, however we are still left with other problems that have not been addressed. These problems, as we mentioned in the previous lecture, are cycles, random delays, asynchronous transmission and losses. The assumption of synchronous transmissions is a problem that arises from the fact that in practical networks it is not reasonable to assume that at every clock cycle all the inputs at a node are ready and the outputs are assigned a linear combination of the inputs at that clock cycle. This is due to the random delays in the network, where different routes may impose different delays on the packets and sometimes a single route may impose different delays. A solution to this problem can be arrived at by the use of buffers and use of packet generations. Note that in general routers have buffers that are used to store the packets before transmission. Under the practical approach to network coding we randomly combine packets that exist in the buffer and belong to the same generation. Basically we tag every packet with a generation number (1 to 2 bytes). At any given time a certain generation number is declared to be current. During the current generation, packets with that current generation number will be randomly linearly combined from the buffer. Also the current generation number stays for some period of time and after that the packets with that generation number that remain in the buffers are flushed and the current generation number is updated to the next generation number and so on. In practice this is achievable since network routers have buffers that hold packets that arrive at a different points in times.

Linear Information Flow Algorithm

We now present a centralized algorithm for multicast network code construction. We assume a directed acyclic graph G with source node S having h data sources and a set of terminals \mathcal{T} . We assume that the $max-flow(S, T_i)$ is always greater than or equal to h for all $T_i \in \mathcal{T}$.

Preprocessing Step:

1. Find h edge-disjoint paths from S to T_i for all $T_i \in \mathcal{T}$ and denote this as the set \mathcal{P}_i . Also $\mathcal{P}_{i,j}$ is the j^{th} path in this set.

2. Form a new graph G' consisting of all of these paths where the set of edges E' in G' is equal to all edges in $\cup_{i=1}^{|T|} \mathcal{P}_i$
3. Number the edges in G' in a way that if e_i is connected to e_j then $i < j$.

We feed the output of this preprocessing step as an input to the algorithm. Before we go into the details of the algorithm, we will explore some of the concepts that we will rely on.

Local vs Global coding vectors

Local coding vector is a vector that consists of the coefficients assigned to the incoming edges at a local node. lets examine the following figure:

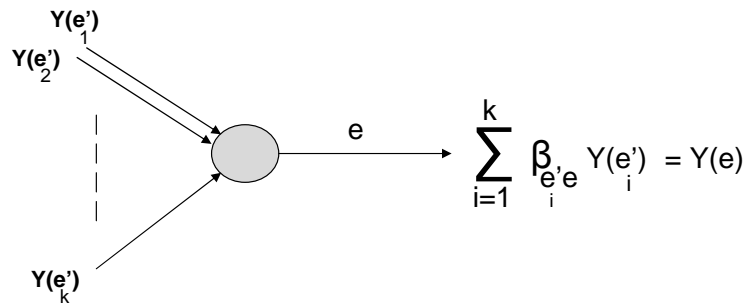


Figure 2: Local Coding Vector

In this case the local coding vector is $\left[\beta_{e'_1 e} \quad \beta_{e'_2 e} \quad - \quad - \quad - \quad - \quad \beta_{e'_k e} \right]$, since the $Y(e)$ can be expressed as

$$Y(e) = \left[\beta_{e'_1 e} \quad \beta_{e'_2 e} \quad - \quad - \quad - \quad - \quad \beta_{e'_k e} \right] \begin{bmatrix} Y(e_1) \\ Y(e_2) \\ - \\ - \\ - \\ - \\ Y(e_k) \end{bmatrix}.$$

However if we relate the input vector $\begin{bmatrix} X(s, 1) \\ X(s, 2) \\ - \\ - \\ X(s, h) \end{bmatrix}$ at source node S directly to $Y(e)$ then we end up with a global coding vector $[\zeta(e)]$ for edge e and we have:

$$Y(e) = \left[\zeta_1 \quad \zeta_2 \quad - \quad - \quad - \quad - \quad \zeta_h \right] \begin{bmatrix} X(s, 1) \\ X(s, 2) \\ - \\ - \\ X(s, h) \end{bmatrix}$$

We can see that the global coding vector for an edge e_k can be derived from the local coding vectors assigned for edges e_i where $i < k$. Finally we will state a definition that we will use frequently in the algorithm:

Definition 1. *Let the β be the set of vectors $\{v_1, v_2, \dots, v_k\}$ where v_i has components belonging to $GF(q)$. Then we define the span of β denoted $\mathcal{L}(\beta)$ to be the set of all vectors that can be expressed as a linear combination of the vectors in β i.e. $\mathcal{L}(\beta) = \{x | x = \sum_{i=1}^k \alpha_i v_i, \text{ where } \alpha_i \in GF(q)\}$*

The Algorithm (Basic Idea)

The LIF algorithm is a greedy algorithm that tries to find good global coding vectors for each edge in G' . Formally the algorithm visits each edge in G' and assigns it a coding vector such that the multicast property is satisfied. In other words the choice of the local coding vector should make the matrix invertible locally and it should be from the span of the coding vectors of the incoming edges, also we want to make sure that the transfer matrix M_i is always full rank for all $T_i \in \mathcal{T}$.