國 立 交 通 大 學

電信工程學系碩士班

碩 士 論 文

冗餘複播圖上之分散式網路編碼

Distributed Network Coding
over Redundant Multicast Graphs

研 究 生： 李欣勳

指導教授： 蘇育德　博士

中 華 民 國 九 十 三 年 七 月

冗餘複播圖上之分散式網路編碼

Distributed Network Coding over Redundant Multicast Graphs

研 究 生：李欣勳　　Student：Hsin-Hsun Li

指導教授：蘇育德　博士　　Advisor：Dr. Yu T. Su

國 立 交 通 大 學

電信工程學系碩士班

碩 士 論 文

A Thesis Submitted to
the Institute of Communication Engineering
College of Electrical Engineering and Computer Science
National Chiao Tung University
in Partial Fulfillment of the Requirements
for the Degree of Master of Science
in
Communication Engineering
July 2004

Hsinchu, Taiwan, Republic of China

中 華 民 國 九 十 三 年 七 月

冗餘複播圖上之分散式網路編碼

研究生 ： 李欣勳　　指導教授 ： 蘇育德 博士

國立交通大學電信工程研究所

中文摘要

對於多重傳播來說，其傳輸速率上限往往被 max-flow bound 給限制住。為了改善傳統單一訊源多重傳播的傳輸速率，在每個網路節點增加一個網路編碼的新功能。藉由網路編碼的功能，這個傳輸速率的上限就可以達得到。更進一步地，使用線性網路編碼就足以讓單一訊源多重傳播速率能夠達到其上限速率，並且，網路碼往往可以在一個有限的場找到。

在覆蓋式網路上，虛擬網路拓撲可以經由不同的使用目的建構出來，而且每個虛擬網路節點可以具備更高階的處理能力。借重網路編碼和覆蓋式網路的優點，可以建造出 2 階冗餘複播圖以提供單一訊源多重傳輸。在 2 階冗餘複播圖上，網路碼的建構可以不需要網路拓撲的資訊。基於這個特別的多重傳播圖形，我們推導出在一個 2 的延伸場上最大能用量的編碼向量數目，並且提出產生編碼向量的演算法。相較於 Zhu 她們所提的方法，此演算法更加的容易，並且，用於多重傳輸的網路碼會需一更少的處理時間。由於在現實世界中，資訊的儲存與傳送都是二位元形式，如果網路碼是從質數場上建構出的，則將會損失一些資訊。我們所提出的方法，是建構在 $GF(2^m)$ 的場上，所以不會有資訊流失。在 2 階冗餘複播圖上，從 $GF(2^m)$ 建構出的網路碼終將增加端點到端點間的流量，是個合理的推測。

Distributed Network Coding over Redundant Multicast Graphs

Student : Hsin-Hsun Li Advisor : Yu T. Su

Institute of Communication Engineering

National Chiao Tung University

Abstract

Multicast transmission over a network is upper bounded by the network capacity (rate) called the *max-flow bound*. In order to improve the capacity of a conventional single source multicast, a new function called network coding is introduced to network nodes. It has been shown that, by means of network coding, the capacity is always achievable. Moreover, linear network codes are sufficient to attain the capacity for all single source multicast transmissions and they cam always can be found over finite fields.

In an overlay network, virtual network topology over an existing physical network is constructed to meet some special networking requirements. For the purpose of implementing network coding, all virtual network nodes must be capable of handling higher level operations. Taking the advantages of network coding and overlay networks, a two-redundant multicast graph is constructed so that network codes over the resulting overlay network can be designed without the knowledge of the network topology. Based on this special multicast graph, we derive the maximum available encoding vectors over an extension filed of GF(2) and propose an encoding vector generation algorithm. The algorithm is simpler and the resulting multicast network codec renders lower complexity. It is believed that network codes over GF(2^m) for redundant multicast graph can also enhance the end-to-end throughput.

誌謝

本論文得以順利完成，首先要感謝指導老師蘇育德教授兩年來不管在專業上的指導或者生活態度上的表現，都讓我精進、受益良多。並且感謝口試委員楊谷章教授、王忠炫助理教授與翁芳標助理教授提供寶貴意見，以彌補論文的缺失不足。另外，要感謝實驗室學長、同學和學弟妹，都曾經在學業上與生活上提供我不少意見與幫助，特別是鄭延修學長與陳彥志學長，常常撥空指導我一些研究方法，加上兩位，所學又廣又精，足以作為我人生路上的目標。

最後，要感謝的，就是一直關心我、鼓勵我的家人和朋友們，僅獻上此論文，代表我最最最深的謝意！

Contents

List of Figures

Chapter 1

Introduction

Multicast technology is useful to a number of emerging network applications require the delivery of packets from one or more senders to a group a of receivers. These applications include bulk data transfer (for example, the transfer of a software upgrade from the software developer to users needing the upgrade), streaming continuous media (for example, the transfer of audio, video, and text of live lecture to a set of distributed lecture participants), shared data applications (for example, a whiteboard or teleconferencing application that is shared among many distributed participants), data feeds (for example, stock quotes), web cache updating, and interactive gaming (for example, distributed interactive virtual environments or multiplayer games such as Quake). The notion of a multicast is the sending of a packet from one sender to multiple receivers with a single send operation. This will save a lot of bandwidth in comparison to sending packet using unicast.

Due to the highly increase of data size, a complete transfer of data in the Internet requires more time or more available bandwidth. From a point-to-point transfer standpoint, the transmission rate of each pair has an theoretic upper bound. From the multicast abstraction standpoint, the bandwidth of a channel can not be shared by different input data when it just allow one to pass. This will decrease the maximum achievable rate of multicast by using only routing and replication, and relatively requires more time to complete the data retrieving. In [5], Ahlswede *et al.* derive that theoret-

ically the *max-flow bound* of multicast rate is always achievable if the network routers and switches have capable of not only routing and replication but also the other function called network coding. Further, Li *et al.*[6] shown the sufficiency of linear network coding and the linear network codes can be found in finite fields. The linearity of network coding simplify the network coding operation. Then, in [8] and [9], both show linear network code construction algorithms, one from the flow perspective and the other using an algebraic characterization. Both of them derive that the linear network codes can be found in a finite field GF(2^m), where the size of m is just correlated to the number of receivers.

Taking the advantages of network coding and overlay network, Zhu *et al.* construct a two-redundant multicast graph. Base on this special graph, the linear network code can be constructed by using a distributed algorithm, which opposes to the centralized algorithms proposed in [8] and [9]. Since their codes are constructed over prime fields, this will make the coding operation in the computer requiring a mapping of table-looking from prime filed to a extension field of order 2. It will needs more processing time and effectively decrease the end-to-end throughput. Also, linear network code construction over a prime filed decreases the effective end-to-end throughput by a factor $\frac{1}{m+1}$, where m is the smallest number that the size of GF(2^m) larger than the prime filed.

Base on the two-redundant mlulticast graph, we propose a simple algorithm that generates linear network code over GF(2^m), and conjecture that eventually the effective ene-to-end throughput will larger than Zhu's proposal [11].

The rest of this thesis is organized as follows. In Chapter 2, we introduce the notion of *max-flow bound* for multicast and the achievability of the bound by using network coding. Then, introduce the sufficiency of linear network code derived by Li *et al.* [6]. In chapter 3, we introduce the construction algorithm for linear network code. In chapter 4, we introduce network coding over overlay networks and our proposal in the last section. In chapter 5, we make a conclusion.

Chapter 2

Network Information Flow

2.1 Network Flows

A communication network is a collection of directed links connecting transmitters, receivers, switches, and routers. It can be represented by a directed graph $G = (V, E)$ with a node set V and an edge set $E \subseteq V \times V$. In most cases of practical concern, G is finite, i.e., $|V| < \infty$. Let node v_i and node v_j be two elements in V, then the edge (link or channel) from node v_i to node v_j can be denoted by a round bracket $(v_i, v_j) \in E$. The edge (v_i, v_j) is directed whose head and tail are denoted by $v_i = tail(e)$ and $v_j = head(e)$, respectively. Define $In(v)$ as the set of edges that end at the node $v \in V$ and $Out(v)$ as the set of edges originating at v. Formally, we have

$$In(v) = \{e \in E : head(e) = v\}, \tag{2.1a}$$

and

$$Out(v) = \{e \in E : tail(e) = v\} \tag{2.1b}$$

For a network with one information source, the node at which information is generated is referred to as the *source node* or *transmitter*, denoted by s, and the information destination nodes are referred to as the *sink nodes* or *receiver nodes*, denoted by $t_1, t_2, \cdots, t_N$. Here, we denote T as the set of all the sink nodes.

For a communication channel from node v_i to node v_j, let the nonnegative integer C_{ij} be the *rate constraint*, i.e., the maximum number of bits which can be sent per unit time over the channel. The C_{ij} is also referred to as the *capacity* of edge (v_i, v_j). Then the set

$$\mathbf{C} = [C_{ij} : (v_i, v_j) \in E] \tag{2.2}$$

represents the rate constraints of the network G.

2.1.1 Flow and cut

Consider a network G in which $t_n \in T$ is a sink node and s be the only one source node. One can regard an edge in G as a water pipe and the network G as a network of water pipes. We shall assume that there is no leakage in the network, so that information flow is conserved at every node other than s and t_n.

Definition 1 *An $s - t_n$* flow $\mathbf{R_n}$ *in a directed network G is an integer-valued function $\mathbf{R_n}$ defined on each edge (v_i, v_j) – R^n_{ij} is the edge flow in (v_i, v_j)– together with a source node s and a sink node t_n satisfying the following three conditions:*

$$(a)\ 0 \le R^n_{ij} \le C_{ij},\ \forall\ (v_i, v_j) \in E, \tag{2.3a}$$

$$(b)\ R^n_{ij} = 0,\ \textit{if}\ (v_i, v_j) \in In(s)\ \textit{or}\ (v_i, v_j) \in Out(t_n), \tag{2.3b}$$

$$(c)\ \textit{For}\ v \ne s\ \textit{or}\ t_n, \sum_{(v_i,v_j)\in In(v)} R^n_{ij} = \sum_{(v_i,v_j)\in Out(v)} R^n_{ij} \tag{2.3c}$$

Condition (a) ensures that the edge flow is nonnegative and would not exceed the edge capacity. Condition (b) assures that the flow goes from node s to node t_n, not in the reverse direction. Condition (c) is called the conservation condition. Intuitively, for any $s - t_n$ *flow* $\mathbf{R_n}$ in a network G, the total edge flows out of source node s should be equal to the total edge flows entering the sink node t_n so that

$$\sum_{(v_i,v_j)\in Out(s)} R^n_{ij} = \sum_{(v_i,v_j)\in In(t_n)} R^n_{ij} \tag{2.4}$$

For a proof see [1] or [2].

Let $|\mathbf{R_n}|$ be the source generating rate associated with an $s - t_n$ *flow* $\mathbf{R_n}$

$$|\mathbf{R_n}| = \sum_{(v_i,v_j)\in Out(s)} R_{ij}^n. \tag{2.5}$$

It is sometimes referred to as the value of an $s-t_n$ *flow* $\mathbf{R_n}$. We denote by $maxflow(s,t_n)$ the maximal value of all possible $s-t_n$ *flow*'s. If an $|\mathbf{R_n}|$ equals to $maxflow(s,t_n)$, then we call the corresponding $\mathbf{R_n}$ a *maximal* $s-t_n$ *flow*.

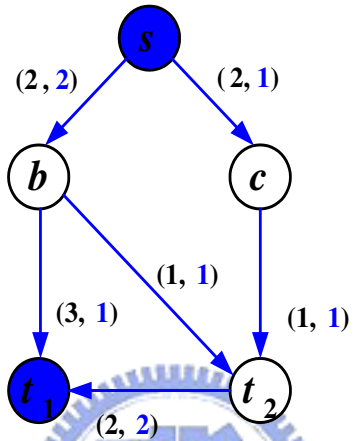


Figure 2.1: A *maximal* $s-t_1$ *flow* $\mathbf{R_1}$

Fig. 2.1 illustrates a *maximal* $s-t_1$ *flow* $\mathbf{R_1}$, the first term and second term in a round bracket beside an edge represent edge capacity and edge flow, respectively, and the value of the $s-t_1$ *flow* is $|\mathbf{R_n}| = 3$.

Definition 2 *Let* $P \subset V$ *and* $\overline{P}$ *be the complementary set of* P *with respect to* V*, i.e.,* $V = P \cup \overline{P}$*. A cut* $(P,\overline{P})$ *is the set of edges* (v_i,v_j) *with node* $v_i \in P$ *and* $v_j \in \overline{P}$*, i.e.,*

$$(P,\overline{P}) = \{(v_i,v_j) \in E : v_i \in P \text{ and } v_j \in \overline{P}\} \tag{2.6a}$$

The capacity of a cut is defined by

$$C(P,\overline{P}) = \sum_{(v_i,v_j)\in(P,\overline{P})} C_{ij} \tag{2.6b}$$

Definition 3 *For* $P_n \subset V$ *and* $\overline{P_n}$ *is the complementary set of* P_n*, then* $(P_n,\overline{P_n})$ *is called an* $s-t_n$ cut *, if* $s \in P_n$ *and* $t_n \in \overline{P_n}$.

In most cases, there are more than one $s-t_n$ cut, each has its own capacity. We denote by $mincut(s,t_n)$ the minimal capacity of $s-t_n$ *cut*. If $C(P_n,\overline{P_n})$ equals to $mincut(s,t_n)$, then we call such $(P_n,\overline{P_n})$ a *minimal* $s-t_n$ *cut*.

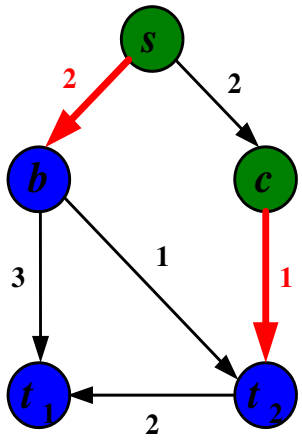


Figure 2.2: A *minimal* $s-t_1$ *cut*

Fig. 2.2 illustrates a *minimal* $s-t_1$ *cut* $(\{s,\ c\},\{t_1,\ b,\ t_2\})=\{(s,\ b),(c,\ t_2)\}$. The number beside an edge represents the edge capacity, and the capacity of the $s-t_1$ *cut* is $C(\{s,c\},\{t_1,b,t_2\})=3$.

2.1.2 Upper bound of an $s-t_n$ *flow*

Let us consider the question of how large $|\mathbf{R_n}|$ can be. Obviously, we can obtain the following two upper bound from (2.4) and (2.5)

$$|\mathbf{R_n}| = \sum_{(v_i,v_j)\in Out(s)} R_{ij}^n \le \sum_{(v_i,v_j)\in Out(s)} C_{ij} \tag{2.7a}$$

$$|\mathbf{R_n}| = \sum_{(v_i,v_j)\in In(t_n)} R_{ij}^n \le \sum_{(v_i,v_j)\in In(t_n)} C_{ij} \tag{2.7b}$$

Eqs (2.7a) and (2.7b) give us a sense that the value of an $s-t_n$ *flow* cannot exceed the sum of the capacities of the edges originating at node s and the sum of the capacities of the edges that end at node t_n. Intuitively, $|\mathbf{R_n}|$ is also bounded by the capacity of any $s-t_n$ *cut*. We state the third upper bound in the following proposition [1].

Proposition 1 *For any $s-t_n$* flow $\mathbf{R_n}$ *and any $s-t_n$* cut $(P_n, \overline{P_n})$ *in network G,*

$$|\mathbf{R_n}| = \Big(\sum_{(v_i,v_j)\in(P_n,\overline{P_n})} R_{ij}^n - \sum_{(v_i,v_j)\in(\overline{P_n},P_n)} R_{ij}^n\Big) \le C(P_n, \overline{P_n}) \tag{2.8}$$

■ Max-Flow Min-Cut

An $s-t_n$ *flow* $\mathbf{R_n}$ is a *maximal $s-t_n$ flow* or called a *max-flow* from node s to node t_n if $|\mathbf{R_n}|$ is greater than or equal to the value of any other $s-t_n$ *flow*, and an $s-t_n$ *cut* $(P_n, \overline{P_n})$ is a *minimal $s-t_n$ cut* or called a *min-cut* between s and t_n if $C(P_n, \overline{P_n})$ is less than or equal to the capacity of any other $s-t_n$ *cut*. From (2.8), we can derive the following corollary [1].

Corollary 1 *For any $s-t_n$* flow $\mathbf{R_n}$ *and any $s-t_n$* cut $(P_n, \overline{P_n})$ *in a network G, $|\mathbf{R_n}| = C(P_n, \overline{P_n})$ if and only if:*

$$(i)\ \textit{For each edge } (v_i, v_j) \in (P_n, \overline{P_n}),\ R_{ij}^n = C_{ij}, \tag{2.9a}$$

$$(ii)\ \textit{For each edge } (v_i, v_j) \in (\overline{P_n}, P_n),\ R_{ij}^n = 0 \tag{2.9b}$$

Furthermore, when $|\mathbf{R_n}| = C(P_n, \overline{P_n})$, the $\mathbf{R_n}$ is a max-flow from node s to node t_n and $(P_n, \overline{P_n})$ is a min-cut between s and t_n.

A *min-cut* between node s and node t_n can be thought of as a flow bottleneck between s and t_n. Therefore, it's intuitively clear that the value of a *max-flow* from node s to node t_n cannot exceed the capacity of a *min-cut* between node s and node t_n. The following theorem, known as the *max-flow min-cut* theorem, states that this bottleneck always can be achieved.

Theorem 1 *(*max-flow min-cut *theorem [4]) Let G be a directed network with a source node s and sink nodes $t_1, t_2, \cdots, t_N$, and $\mathbf{C}$ be the rate constraints. Then for all $t_n \in \{t_1, t_2, \cdots, t_N\}$, the value of max-flow from s to t_n is equal to the capacity of min-cut between s and t_n, i.e., $maxflow(s, t_n) = mincut(s, t_n)$.*

2.1.3 Ford-Fulkerson labelling algorithm

In order to achieve the capacity of the *min-cut* between node s and node t_n for the value of $s - t_n$ *flow*. Ford and Fulkerson propose an algorithm [1] that will eventually result a *max-flow* from s to t_n.

Before starting the algorithm, some notations must be established. A *chain* in a directed graph is represented by $(v_{c_0}, e_{c'_1}, v_{c_1}, e_{c'_2}, \cdots, v_{c_{L_c-1}}, e_{c'_{L_c}}, v_{c_{L_c}})$, which $e_{c'_k} = (v_{c_{k-1}}, v_{c_k})$ or $e_{c'_k} = (v_{c_k}, v_{c_{k-1}})$ for $1 \leq k \leq L_c$. If $e_{c'_k} = (v_{c_{k-1}}, v_{c_k})$, then the edge is called a *forward edge* of the chain. Otherwise, $e_{c'_k} = (v_{c_k}, v_{c_{k-1}})$, the edge is called a *reverse edge* of the chain. When $v_{c_0} = s$ and $v_{c_{L_c}} = t_n$, we call such a chain an $s-t_n$ *chain*. For two nodes v_i and v_j connected by an edge $(v_i, v_j) \in E$, node v_j is the successor of node v_i. On the contrary, node v_i is the predecessor of node v_j.

The algorithm has two parts called Routine A and Routine B. The first is a labelling process that searches for a *flow augmenting* $s - t_n$ *chain* for which $R^n_{ij} < C_{ij}$ on all forward edges of the chain and $R^n_{j'i'} > 0$ on all reverse edges of the chain. If a *flow augmenting* $s - t_n$ *chain* is found, then Routine B changes the edge flows accordingly. Otherwise, no such chain exits, and the $s - t_n$ *flow* is a *max-flow* from node s to t_n.

The algorithm begins with any feasible $s - t_n$ *flow* $\mathbf{R_n}$, i.e., any $s - t_n$ *flow* that satisfies the flow conditions (2.3a), (2.3b) and (2.3c). Initially, all nodes are in the unlabelled state and during the labelling process each node is in one of the following three states: (1) *unlabelled*, (2) *labelled and unscanned* and (3) *labelled and scanned*. We now provide a complete description of the Ford-Fulkerson Algorithm.

Routine A (labelling process):

- Initially, assign the source node s the label $(-, \triangle(s) = \infty)$ and set s in labelled and unscanned state.
- **General step**:
  - Select any node v_i that is in labelled and unscanned state and let $(v^{\pm}, \triangle(v_i))$

be its label.

- To all v_i's unlabelled successor nodes, v_j, such that $R^n_{ij} < C_{ij}$, assign the label $(v_i^+, \triangle(v_j))$, where

$$\triangle(v_j) = min\{\triangle(v_i),\ C_{ij} - R^n_{ij}\} \tag{2.10a}$$

  To all v_i's unlabelled predecessor nodes, v_j, such that $R^n_{ji} > 0$, assign the label $(v_i^-, \triangle(v_j))$, where

$$\triangle(v_j) = min\{\triangle(v_i),\ R^n_{ji}\} \tag{2.10b}$$

- Set v_i in labelled and scanned state and v_j in labelled and unscanned state.

- Repeat the **General step** until one of the following two cases happened:
  - Case 1: Sink node t_n is in labelled and unscanned state, go to **Routine B**.
  - Case 2: No more labels can be assigned, terminate!

Routine B (flow change):

- Let sink node t_n has the label $(v^+_{c_{L_c-1}}, \triangle(t_n))$.
- According to the first term in the label of node t_n, we can find the node $v_{c_{L_c-1}}$. Similarly, we can find the node $v_{c_{L_c-2}}$ by the label of node $v_{c_{L_c-1}}$. Repeat this procedure until the source node s is found, and then we can find the corresponding *flow augmenting* $s - t_n$ *chain* $(s = v_{c_0}, e_{c'_1}, v_{c_1}, e_{c'_2}, \cdots, v_{c_{L_c-1}}, e_{c'_{L_c}}, v_{c_{L_c}} = t_n)$.
- Increase the flow on all the forward edges and decrease the flow on all the reverse edges of the $s - t_n$ *chain* by the amount of $\triangle(t_n)$, respectively.
- Then, discard all labels and return to **Routine A**.

For any given $s - t_n$ *flow* $\mathbf{R_n}$, a finite number of applications of the Ford-Fulkerson labelling algorithm yields a *max-flow* from node s to node t_n. Moreover, if P_n is the set

of nodes labelled during the final application of the algorithm, then $(P_n, \overline{P_n})$ is a *min-cut* between s and t_n.

A modified and improved version of the Ford-Fulkerson algorithm is the Edmonds-Karp algorithm [3], which is an $O(|V||E|^2)$ polynomial time algorithm.

2.2 Single-Source Multicast with Network Coding

Consider a point-to-point communication network on which an information source is to be *multicast*, which means by using network the sending of information from one source node to a certain set of sink nodes with a single send operation. A multicast rate is called *achievable* if all sink nodes can successfully receive the information generated by source node at this rate. Contrary to one's intuition, it is in general not optimal to simply route or replicate information for the multicast. Rather, by employing coding at the nodes, which is refer to as *network coding*, bandwidth can in general be saved. But, this needs a higher level processing of addition function design in switching systems. We focus on point-to-point communication networks satisfying the following:

1. the communication channels are free of error;
2. the information is received at the sink nodes with zero error;
4. the communication network contains no directed cycle, i.e., acyclic;
3. the network is delay-free.

2.2.1 Upper bound of multicast rate with only routing and replication

Let r be the rate at which information multicast from source node s to all sink nodes $t_1, t_2, \cdots, t_N$. From the *max-flow min-cut* theorem, it is easily to see that the multicast rate r with only routing and replication cannot exceed the *max-flow* from source node s to any sink node $t_n \in T$,

$$r \leq maxflow(s, t_n) \tag{2.11}$$

Intuitively, the multicast rate is bounded by the minimum value of $maxflow(s, t_n)$ for all $t_n \in T$,

$$r \leq \min_n maxflow(s, t_n) \tag{2.12}$$

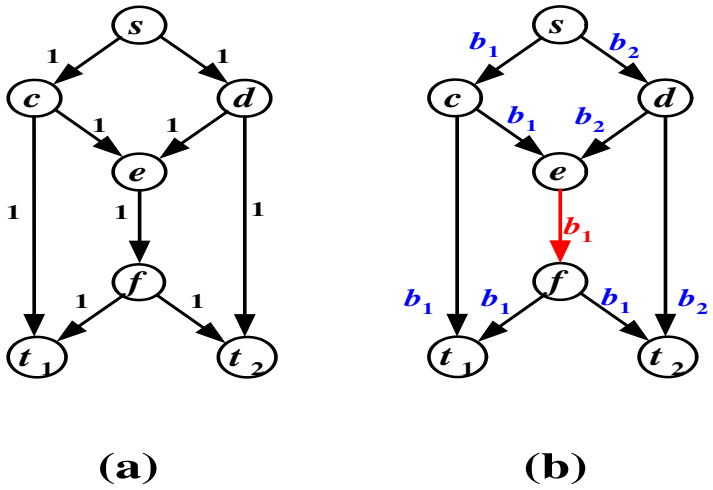


Figure 2.3: A multicast with only routing and replication

Given a network in Fig. 2.3(a), the number beside an edge represents the edge capacity. By the *max-flow min-cut* theorem, we can use the relation

$$mincut(s, t_n) = maxflow(s, t_n) \tag{2.13}$$

to derive the values of *max-flow* from source node s to sink nodes t_1 and t_2, respectively. A *min-cut* between node s and node t_1 is $(\{s, c, d, e, f, t_2\}, \{t_1\})$ and

$$mincut(s, t_1) = 2 \tag{2.14}$$

Similarly, the capacity of *min-cut* between node s and node t_2 is

$$mincut(s, t_2) = 2 \tag{2.15}$$

Then the *max-flow bound* is 2. Assume that the multicast rate is two bits per unit time, and let b_1 and b_2 be the two information bits. In Fig. 2.3(b), all nodes in network are only capable of routing and replication, the source node s tries to multicast two

bits of data, b_1 and b_2, to both sink nodes. Node e allows only one of received bits on to its outgoing edge, suppose b_1 be chosen. From the figure, we can see that sink node t_2 can receive both b_1 and b_2, and unfortunately t_1 receives two duplicated b_1 but b_2 cannot recovered. Thus for this network, the *max-flow bound* can not be achieved by only routing and replicating bits. Rather, the achievable multicast rate with only routing and replication is 3/2.

2.2.2 Network coding

Before talking about network coding, we first say something about why using network coding. In a network, routers or switches (i.e., nodes) traditionally have capable of *routing* and *replication* to their incoming data. As we can see in the above example, for multicast cases only these two functions sometimes the rate of *max-flow bound* is not achievable. Therefore, add the routers or switches a new function, called *network coding*, to improve the multicast rate and hopefully achieve the *max-flow bound*.

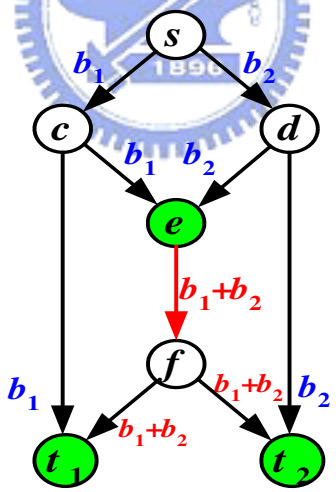


Figure 2.4: A multicast with network coding

In Fig. 2.4, coding is allowed at nodes. The node e encodes two received data streams, b_1 and b_2, for example, by using modulo 2 addition and transmits the resulting stream $(b_1 + b_2)$ over its outgoing edge (e, f). At node t_1, b_1 and $(b_1 + b_2)$ are received,

and b_2 can be recovered by adding b_1 and $(b_1 + b_2)$, because

$$b_2 = b_1 + (b_1 + b_2). \tag{2.16}$$

Similarly, b_2 and $(b_1 + b_2)$ are received at node t_2, and b_1 can be recovered by adding b_2 and $(b_1 + b_2)$. Therefore, the multicast rate achieves the *max-flow bound* by adding coding function at network nodes. The coding operation at nodes in a network is refer to as *network coding*.

We describe a coding scheme for the network which is referred to as a network coding. Assume that there is no input edge at node s, i.e., $(v_i, s) \notin E$ for all $v_i \in V \backslash \{s\}$. We consider a block code of length m. Let X denote the information source with rate τ and assume that x, the outcome of X, is obtained by selecting an index from a set χ according to the uniform distribution. The elements in $\chi = \{1, 2, \cdots, \lceil 2^{m\tau} \rceil\}$ are called messages. For $(v_i, v_j) \in E$, node v_i sends information to node v_j which depends only on the information previously received by node v_i.

2.2.3 Upper bound of multicast rate with network coding

Naturally, we are interested in the maximum possible value of the multicast rate r in a given network and hopefully achieve the *max-flow bound* or higher by using network coding. Assume that each edge (v_i, v_j) in G with capacity C_{ij} is decomposed into C_{ij} unit capacity edges. Given a network G, the upper bound of the multicast rate with network coding is derived by using an $(m, (\eta_{ij} : (v_i, v_j) \in E), \tau)$ $\alpha - code$ on G, which is defined by the components listed as below.

(1) A positive integer K (i.e., K transactions)

(2)

$$u_1 : \{1, 2, \cdots, K\} \to V \tag{2.17a}$$

and

$$u_2 : \{1, 2, \cdots, K\} \to V \tag{2.17b}$$

such that $(u_1(k), u_2(k)) \in E$ and it is a unit capacity edge between node $u_1(k)$ and node $u_2(k)$. (K transactions take place in chronological order)

(3) $A_k = \{1, 2, \cdots, |A_k|\}$, $1 \leq k \leq K$, such that

$$\prod_{k \in T_{ij}} |A_k| = \eta_{ij}, \tag{2.18a}$$

where

$$T_{ij} = \{1 \leq k \leq K : (u_1(k), u_2(k)) = (v_i, v_j)\}. \tag{2.18b}$$

The set A_k represents all possible index that can be send onto $(u_1(k), u_2(k))$, the set T_{ij} represents the indices of all the transactions for which information is sent form v_i to v_j, and η_{ij} is the number of possible index tuples that can be sent from node v_i to node v_j.

(4) If $u_1(k) = s$, then

$$Y_k : \chi \to A_k, \tag{2.19a}$$

where

$$\chi = \{1, 2, \cdots, \lceil 2^{m\tau} \rceil\}. \tag{2.19b}$$

If $u_1(k) \neq s$, if

$$Q_k = \{1 \leq k' \leq k : u_2(k') = u_1(k)\} \tag{2.19c}$$

is nonempty, then

$$Y_k : \prod_{k'} A_{k'} \to A_k \tag{2.19d}$$

Otherwise, let Y_k be an arbitrary constant taken from A_k.

Here, Y_k is the *encoding function* at node $u_1(k)$ and in *kth* transaction, node $u_1(k)$ encodes according to encoding function Y_k and sends an index in A_k to node $u_2(k)$.

(5)

$$g_n : \prod_{k' \in W_n} A_{k'} \to \chi, \tag{2.20a}$$

$n = 1, 2, \cdots, N,$ where

$$W_n = \{1 \leq k \leq K : u_2(k) = t_n\} \tag{2.20b}$$

Such that for all $n = 1, 2, \cdots, N,$

$$\widetilde{g_n}(x) = x \tag{2.20c}$$

for all $x \in \chi$.

Here, g_n is the *decoding function* at node t_n, the set W_n represents the indices of all transactions for which information is sent to t_n, and x is the multicast information transmitted from source s. The x in (2.20c) has different meaning, x in the left-hand side is the source input, and x in the right-hand side of is the decoder output.

Definition 4 *[4], [5] For a network G with rate constraints $\mathbf{C}$, an multicast rate $r \geq 0$ is asymptotically achievable if for any $\varepsilon > 0$, there exists for sufficiently large m an $(m, (\eta_{ij} : (v_i, v_j) \in E), \tau)$ $\alpha - code$ on G, such that*

$$m^{-1} log_2 \eta_{ij} \leq C_{ij} + \varepsilon \tag{2.21a}$$

for all $(v_i, v_j) \in E$, where $m^{-1} log_2 \eta_{ij}$ is the average bit rate of the code on channel (v_i, v_j), and

$$r - \varepsilon \leq \tau \tag{2.21b}$$

The following theorem [4],[5] states that the multicast rate with network coding is also bounded by the *max-flow bound*.

Theorem 2 *[4] For a network G with rate constraints $\mathbf{C}$ if the multicast rate r is achievable, then*

$$r \leq \min_n max flow(s, t_n) \tag{2.22}$$

For a proof of the theorem please see Appendix A.

2.2.4 Achievability of the *max-flow bound* for network coding

After reviewing the above fundamental theorems one might ask whether the multicast rate at the *max-flow bound* with network coding is always achievable. The answer is affirmative. This follows from proving the existence of an $(m, (\eta_{ij} : (v_i, v_j) \in E), r - \varepsilon)$ $\beta - code$ on G. We need the following proposition.

Proposition 2 [4] *If $G = (V, E)$ is a finite acyclic graph, then it is possible to order the nodes of the network G in a sequence such that if there is an edge from node v_i to node v_j, then node v_i is before node v_j.*

An $(m, (\eta_{ij} : (v_i, v_j) \in E), r)$ $\beta - code$ on network G is defined by the following components listed as below:

(1) for all $(s, v_j) \in E$, an encoding function

$$Y_{sj} : \chi \to \{1, 2, \cdots, \eta_{ij}\}, \tag{2.23a}$$

where

$$\chi = \{1, 2, \cdots, \lceil 2^{mr} \rceil\}; \tag{2.23b}$$

(2) for all $(v_i, v_j) \in E$ such that $v_i \neq s$, an encoding function

$$Y_{ij} : \prod_{v_{i'}:(v_{i'}, v_i) \in E} \{1, 2, \cdots, \eta_{i'i}\} \to \{1, 2, \cdots, \eta_{ij}\} \tag{2.24}$$

(if $\{v_{i'} : (v_{i'}, v_i) \in E\}$ is empty, conventionally Y_{ij} is an arbitrary constant taken from $\{1, 2, \cdots, \eta_{i'i}\}$);

The encoding function Y_{ij} is applied before $Y_{i'j'}$ if the order of node v_i is small than the order of node v_j and before $Y_{ij'}$ if the order of node v_j is small than the order of node v'_j.

(3) For all $n = 1, 2, \cdots, N$, a decoding function

$$g_n : \prod_{v_i:(v_i,t_n)\in E} \{1, 2, \ldots, \eta_{it_n}\} \to \chi \tag{2.25a}$$

such that

$$\widetilde{g_n}(x) = x \tag{2.25b}$$

for all $x \in \chi$.

The x in Eqs (2.25b) has different meaning, x in the left-hand side is the source input, and x in the right-hand side of is the decoder output. The following theorem states that for multicast rate less than or equal to the *max-flow bound* is always achievable.

Theorem 3 *[4], [5] For a network G with rate constraints $\mathbf{C}$, if the multicast rate*

$$r \leq \min_n maxflow(s, t_n), \tag{2.26}$$

then r is achievable.

A detailed proof is given in Appendix B.

2.2.5 Sufficiency of linear network codes

The proof of the above theorem (Appendix B) indicates that a multicast rate r is always achievable if it does not exceed the *max-flow bound*. In other words, there exits an $(m, (\eta_{ij} : (v_i, v_j) \in E), r)$ $\beta - code$ on the network G and some encoding functions for a sufficiently large m.

For a given acyclic network G with rate constraints $\mathbf{C}$, each edge of capacity C_{ij} can, without loss of generality, be decomposed into C_{ij} unit capacity edges between node v_i and node v_j. Consider the multicast rate

$$r = \min_n maxflow(s, t_n) \tag{2.27}$$

In order to explain the following description using *generic linear-code multicast (generic LCM)*–a coding scheme proposed by Li *et al.* [6]–one can remove some unit capacity edges such that

$$maxflow(s, t_n) = \min_n maxflow(s, t_n) \tag{2.28}$$

for all $t_n \in T$.

A generic LCM is an assignment of an r-dimensional vector space $\mathbf{D}(v)$ to every node $v \in V$ and an r-dimensional column vector $\underline{d}(e)$, called encoding vector, to every unit capacity edge $e \in E$ over a base filed, which is in an infinite field or a large enough finite field, such that

1. $\dim(\mathbf{D}(s)) = r$;
2. $\underline{d}(e) \in \mathbf{D}(v)$ for all $e \in Out(v)$, where $v \in V$;
3. for all $v \neq s$, the corresponding vector space $\mathbf{D}(v) =< \{\underline{d}(e) : e \in In(v)\} >$, which the notation $< \cdot >$ is for *linear span*;
4. for each $e' \in Out(v)$, the corresponding encoding vector $\underline{d}(e')$ is a linear combination of incoming encoding vectors $\{\underline{d}(e) : e \in In(v)\}$.

Suppose that the muticast information can be encoded into r information symbols and formed into an r-dimensional row vector, called a *source vector*, over the base field

$$\underline{X}(s) = (X_1(s), X_2(s), \cdots, X_r(s)). \tag{2.29}$$

The encoding function on each edge $e \in E$ is defined as

$$\begin{aligned} Y(e) &= \underline{X}(s) \cdot \underline{d}(e)^T \\ &= \sum_{i=1}^{r} d_i(e) X_i \end{aligned} \tag{2.30}$$

where

$$\underline{d}(e) = (d_1(e), d_2(e), \cdots, d_r(e))^T \tag{2.31}$$

Condition 4 of the generic LCM and the fact that encoding functions are all linear imply that for any given node $v \in V \setminus \{s\}$, the encoding function of each $e' \in Out(v)$ can be expressed as the linear combination of the received data

$$Y(e') = \sum_{e \in In(v)} \beta_{e,\, e'} Y(e) \tag{2.32}$$

where the coefficients $\{\beta_{e,\, e'}\}$, which are referred to as the network codes, are elements over the base field. Since the encoding operation is linear, the network codes are called linear network codes.

Moreover, it is proved [6] that a generic LCM always exits for an acyclic network, provided that the base field is an infinite field or a large enough finite field, such that

$$\dim(\mathbf{D}(t_n)) = r \tag{2.33}$$

for all $t_n \in T$. This implies that the source vector $\underline{X}(s)$ can be recovered by all the sink nodes and a multicast rate r equals to the *max-flow bound* is achievable.

In conclusion, linear network coding is *sufficient* for a multicast to achieve a rate at the *max-flow bound* and network coding does reduce the operation complexity significantly.

Chapter 3

Construction of Linear Network Codes over Finite Fields

As mentioned in Chapter 2, it has been proved that given a directed, delay-free, acyclic and finite communication network G with rate constraints $\mathbf{C}$, there exists a linear network code over an infinite field or large enough field if the multicast rate

$$r \leq \min_n maxflow(s, t_n) \tag{3.1}$$

Without loss of generality, we assume that the field over which the network code is constructed is GF(2^m), for some large m. Both [7] and [8] propose algorithms for constructing linear network code from a graph theory point of view. Representing a multicast with rate r by sending r GF(2^m) symbols per unit time to each sink node, they show that it is sufficient to perform coding over edges in r edge-disjoint paths from s to t_n (if exists), for all $t_n \in T$. Two linear coding schemes are suggested in [8]. The first algorithm requires a running time of order $O(|E| \cdot |T| \cdot r \cdot (r + |T|))$ and, to guarantee the existence of a linear network code, it requires a field size that satisfies

$$2^m > |T| \tag{3.2}$$

The second algorithm has a faster running time $O(|E| \cdot |T| \cdot r + |T| \cdot O(r^{2.367}))$ but requires a larger lower bound for m

$$2^m > 2 \cdot |E| \cdot |T| \tag{3.3}$$

Another approach is recently proposed by Koetter and Médard [9]. Their algebraic approach can easily be applied to solve problems related to coding over general multicast networks. For the convenience of ensuing discussion for multiple source multicasting, we need to generalize some old and introduce new notations.

To begin with, let's assume that a source node v generates a set of discrete random processes

$$\mathcal{X}(v) = \{X_1(v), X_2(v), \cdots, X_{\mu(v)}(v)\}, \tag{3.4}$$

where random processes $X_l(v),\ \forall\ l \in \{1, 2, \cdots, \mu(v)\}$ and $\forall\ v \in V$, are independent of each other and have a constant and integral entropy rate, e.g., one bit per unit time,

$$H(X_{l_1}(v)) = H(X_{l_2}(v)) = 1 \tag{3.5}$$

for $l_1,\ l_2 \in \{1, 2, \cdots, \mu(v)\}$. The $H(X)$ is the entropy rate of a random process X. If v is not a source node, then $\mathcal{X}(v)$ is a empty set.

A *connection* c is defined as a triple

$$(v, v', \mathcal{X}(v, v')) \in V \times V \times \mathcal{P}_{\mathcal{X}(v)}, \tag{3.6}$$

where $\mathcal{P}_{\mathcal{X}(v)}$ denotes the power set of $\mathcal{X}(v)$ and $\mathcal{X}(v, v') \subseteq \mathcal{X}(v)$. It indicates to replicate, by means of the network G, a subset of the random processes in $\mathcal{X}(v)$ at some different node v'. For a given connection $c = (v, v', \mathcal{X}(v, v'))$, call v a *source* node and v' a *sink* node of the connection c and write $v = source(c)$ and $v' = sink(c)$. The rate $r(c)$ of a connection $c = (v, v', \mathcal{X}(v, v'))$ is defined as

$$\begin{aligned} r(c) &= \sum_{l:\ X_l(v) \in \mathcal{X}(v,v')} H(X_l(v)) \\ &= |\mathcal{X}(v, v')| \end{aligned} \tag{3.7}$$

If v' is the sink node of connections, the collection of $\nu(v')$ random processes

$$\mathcal{Z}(v') = \{Z_1(v'), Z_2(v'), \cdots, Z_{\nu(v')}(v')\} \tag{3.8}$$

denotes the decoding output at v'. A connection $c = (v, v', \mathcal{X}(v, v'))$ is *established successfully* if a copy of $\mathcal{X}(v, v')$ is a subset of $\mathcal{Z}(v')$. This is because node v' can be a sink node of different connections. Similarly, $\mathcal{Z}(v')$ is a empty set if v' is not a sink node of any connection. Assume that communication in the network is performed by transmission of vectors of bits and the length of the vectors are equal in all transmissions. Any binary vector of length m can be interpreted as an element in GF(2^m), the finite field with 2^m elements. It is frequently considered the *algebraic closure* $\overline{\mathbb{F}}$ of GF(2^m), which is defined as the union of all possible algebraic extensions of GF(2^m). Also, assume that all links are synchronized with respect to the symbol timing.

Let $G = (V, E)$ be a delay-free acyclic communication network. Then G is a GF(2^m)-linear network, if for all links, the random process $Y(e)$ on a link $e = (v, v_j)$ satisfies

$$Y(e) = \sum_{l=1}^{\mu(v)} \alpha_{e,\ l} X_l(v) + \sum_{e':head(e')=tail(e)} \beta_{e',\ e} Y(e'), \tag{3.9}$$

where $\alpha_{e,\ l}$ and $\beta_{e',\ e}$ are elements of GF(2^m) and $\mu(v) = 0$ if v is not a source node. It is also sufficient for the output process $Z_{l'}(v')$ at a sink node v' to be a linear combination of the random processes $Y(e')$ for all $e' \in In(v')$

$$Z_{l'}(v') = \sum_{e' \in In(v')} \varepsilon_{e',\ l'} Y(e') \tag{3.10}$$

where the coefficients $\varepsilon_{e',\ l'}$ are elements of GF(2^m).

For a directed, acyclic and finite network G, the nodes can be ordered to a sequence such that if there is an edge from node v_i to v_j, then node v_i is before node v_j and $i < j$. Since the nodes are ordered, the edges also can be ordered in a way that $e_{i'}$ is before $e_{j'}$ if $head(e_{i'})$ before $head(e_{j'})$. Suppose that a network contains μ information sources generated from source nodes, then form the sources into a row vector

$$\begin{aligned} \underline{X} &= (X_1, X_2, ..., X_\mu) \\ &= (X_1(v_1), X_2(v_1), \cdots, X_1(v_2), \cdots, X_1(v_{|V|}), ..., X_{\mu(v_{|V|})}(v_{|V|})) \end{aligned} \tag{3.11}$$

which $\underline{X}$ is the vector of input random processes on all nodes in V. If a node $v_i \in V$ is not a source node, set $\mu(v_i)$ to be zero and the length of $\underline{X}$ is

$$\mu = \sum_{v_i \in V} \mu(v_i). \tag{3.12}$$

Define the entries of a $\mu \times |E|$ matrix A as

$$A_{i,j} = \begin{cases} \alpha_{e_j,\, l}, & X_i = X_l(tail(e_j)) \text{ for some } l; \\ 0, & \text{otherwise.} \end{cases} \tag{3.13}$$

Similarly, suppose that a network contains ν information sinks which retrieved at sink nodes, then form the sinks into a row vector

$$\begin{aligned} \underline{Z} &= (Z_1, Z_2, ..., Z_\nu) \\ &= (Z_1(v_1), Z_2(v_1), \cdots, Z_1(v_2), \cdots, Z_1(v_{|V|}), \cdots, Z_{\nu(v_{|V|})}(v_{|V|})) \end{aligned} \tag{3.14}$$

which $\underline{Z}$ is the vector of output random processes on all nodes in V. If a node $v_j \in V$ is not a sink node of any connection, set $\nu(v_j)$ to be zero and the length of $\underline{Z}$ is

$$\nu = \sum_{v_j \in V} \nu(v_j). \tag{3.15}$$

Define the entries of a $\nu \times |E|$ matrix B as

$$B_{i,j} = \begin{cases} \varepsilon_{e_j,\, l'}, & Z_i = Z_{l'}(head(e_j)) \text{ for some } \ l'; \\ 0, & \text{otherwise.} \end{cases} \tag{3.16}$$

The "directed labelled line graph" of $G = (V, E)$ is defined as ß($\ddot{\text{V}}$,$\ddot{\text{E}}$) with node set $\ddot{\text{V}}$ $= E$ and edge set $\ddot{\text{E}} = \{(e_i, e_j) \in E^2 : head(e_i) = tail(e_j)\}$, which edge ë$\in$ $\ddot{\text{E}}$ is labelled with a corresponding symbol β_{e_i,e_j}. Define the $|E| \times |E|$ adjacency matrix F of the graph ß with elements $F_{i,j}$ given as

$$F_{i,j} = \begin{cases} \beta_{e_i,e_j}, & head(e_i) = tail(e_j); \\ 0, & \text{otherwise.} \end{cases} \tag{3.17}$$

Since the network coding is linear, the relation between $\underline{X}$ and $\underline{Z}$ can be describe by a transfer matrix M based on the matrices A, B, and F. That means

$$\underline{Z} = \underline{X}M, \tag{3.18}$$

where

$$M = A(I - F)^{-1}B^T \tag{3.19}$$

and I is the $|E| \times |E|$ identity matrix. Since the network G is acyclic, the graph ß is also acyclic, then F is a strict upper-triangular matrix and elements in the diagonal are all zero. Hence, F is a nilpotent matrix, and $(I - F)^{-1}$ exists in the ring of polynomials, i.e., the elements in $(I - F)^{-1}$ are belong to $\mathrm{GF}(2^m)[\cdots, \beta_{e_i,e_j}, \cdots]$. Further, it can be derived that elements in M are belong to the ring of polynomials

$$\mathrm{GF}(2^m)[\cdots, \alpha_{e_i,l}, \cdots, \beta_{e_i,e_j}, \cdots, \varepsilon_{e_j,l'}, \cdots] \tag{3.20}$$

and for simplicity, give the linear network codes $\alpha_{e_i,l}$, β_{e_i,e_j}, and $\varepsilon_{e_j,l'}$ another notations

$$\begin{aligned} \underline{\xi} &= (\xi_1, \xi_2, \cdots) \\ &= (\cdots, \alpha_{e_i,l}, \cdots, \beta_{e_i,e_j}, \cdots, \varepsilon_{e_j,l'}, \cdots) \end{aligned} \tag{3.21}$$

The following Lemma is the foundation used to prove the existence of linear network codes of given connections.

Lemma 1 [9] *Let $\mathbb{F}[\xi_1, \xi_2, \cdots, \xi_n]$ be the ring of polynomial over an infinite field $\mathbb{F}$ in variables $\xi_1, \xi_2, \cdots, \xi_n$. For any non-zero element $f \in \mathbb{F}[\xi_1, \xi_2, \cdots, \xi_n]$ there exists an infinite set of n-tuples $(a_1, a_2, ..., a_n) \in \mathbb{F}^n$ such that*

$$f(a_1, a_2, ..., a_n) \neq 0. \tag{3.22}$$

TABLE I
LINEAR NETWORK CODE SEARCH ALGORITHM

Input:
Given a polynomial f in indeterminants $\xi_1, \xi_2, ..., \xi_n$
Set integers: $i = 1$, $t = 1$

Iteration:
(1) Find the maximal degree δ of f in any variable ξ_j
and let m be the smallest number such that $2^m > \delta$
(2) Find an element a_t in $\mathrm{GF}(2^m)$ such that $f(\underline{\xi})|_{\xi_t = a_t} \neq 0$
and let $f \leftarrow f(\underline{\xi})|_{\xi_t = a_t}$
(3) If $t = n$ then halt, else $t \leftarrow t + 1$ go to (2)

Output: $(a_1, a_2, ..., a_n)$

Base on the existence of linear network codes, a simple algorithm, see Table I, is used to find a vector $\underline{a}$ over a suitable field GF(2^m) such that $f(\underline{a}) \neq 0$.

For a given network G and a given set of connections ℓ, it is formally to define a network coding problem as a pair (G, ℓ). The problem is to give algebraic conditions under which a set of desired connections ℓ is feasible or equivalently find elements $\alpha_{e_i,l}$, β_{e_i,e_j} and $\varepsilon_{e_j,l'}$ in a suitably chosen field GF(2^m), such that all desired connections can be established successfully in G. For such a set of elements $\alpha_{e_i,l}$, β_{e_i,e_j} and $\varepsilon_{e_j,l'}$ that makes the desired connections feasible will be called a *solution* to the network coding problem (G, ℓ). If a solution exists, the network coding problem will be called *solvable*.

3.1 Point-to-Point Connection

First, consider a point-to-point connection $c_n = (s, t_n, \mathcal{X}(s))$, which s is the source node and t_n be the sink node on network G and

$$\mathcal{X}(s) = \{X_1(s), X_2(s), \cdots, X_{r(c_n)}(s)\} \tag{3.23}$$

Let

$$\underline{X}(s) = (X_1(s), X_2(s), \cdots, X_{r(c_n)}(s)) \tag{3.24}$$

denote the vector of input processes generated at source node s and

$$\underline{Z}(t_n) = (Z_1(t_n), Z_2(t_n), \cdots, Z_{r(c_n)}(t_n)) \tag{3.25}$$

denote the vector of output processes at sink node t_n. Given an $r(c_n) \times |E|$ matrix A, a $|E| \times |E|$ matrix $(I - F)^{-1}$ and a $|E| \times r(c_n)$ matrix B_n^T. A corresponding $r(c_n) \times r(c_n)$ transfer matrix M_n can describe the relationship between $\underline{X}(s)$ and $\underline{Z}(t_n)$.

$$\begin{aligned} \underline{Z}(t_n) &= \underline{X}(s)A(I - F)^{-1}B_n^T \\ &= \underline{X}(s)M_n \end{aligned} \tag{3.26}$$

where elements in M_n are polynomials over a ring GF(2^m)[$\underline{\xi}$]. The connection can be establish successfully if the inverse of matrix M_n exists. Let $f(\underline{\xi})$ be the determinant of

M_n. If $f(\underline{\xi})$ is a nonzero element in a ring of polynomials GF$(2^m)[\underline{\xi}]$ over an infinite field GF(2^m), by Lemma 1, there exists infinite number of $\underline{a}$ over an infinite filed GF(2^m), such that $f(\underline{a}) \neq 0$. Further, by algorithm in Table I, a set of network code in a suitable filed GF(2^m) can be found. The following theorem ties the relationship between the network transfer function M_n (an algebraic quantity) and the Max-flow Min-cut theorem (an graph-theoretic tool):

Theorem 4 [9] *Give a linear network. The following three statements are equivalent:*

1. *A point-to-point connection* $c_n = (s, t_n, \mathcal{X}(s, t_n))$ *is established successfully;*
2. *The* max-flow bound *is satisfied for a rate* $r(c_n)$.
3. *The determinant of the* $r(c_n) \times r(c_n)$ *transfer matrix* M_n *is nonzero over the ring* $GF(2^m)[\underline{\xi}]$

It is further derived by Ho *et al.*[10] that the determinant of a transfer matrix between a connection c_n can be expressed as

$$\begin{aligned} f(\underline{\xi}) &= det(A(I-F)^{-1}B_n^T) \\ &= (-1)^{(|E|+1)r(c_n)} det(\begin{pmatrix} A & 0 \\ I-F & B_n^T \end{pmatrix}) \end{aligned} \tag{3.27}$$

It is intuitive that the maximal degree of $f(\underline{\xi})$ in any variable is 1. So a suitable filed can be chosen to be GF(2).

3.2 Single Source Multicast Connections

For the multicast case, it consists of the distribution of the information, $\mathcal{X}(s)$, generated at a single source node s to a set of sink nodes $t_1, t_2, \cdots, t_N$, such that all sink nodes receive all source bits. In other words, the set of desired connections is given by

$$\ell = \{(s, t_1, \mathcal{X}(s)), (s, t_2, \mathcal{X}(s)), \cdots, (s, t_N, \mathcal{X}(s))\}. \tag{3.28}$$

Then the rate of each connection $c_n = (s, t_n, \mathcal{X}(s))$ is given by $r(c_n) = |\mathcal{X}(s)|$, for all $t_n \in T$. For simplicity, let r denotes the multicast rate, hence $r = r(c_n)\ \forall\ t_n \in T$. Similarly, let

$$\underline{X}(s) = (X_1(s), X_2(s), ..., X_r(s)) \tag{3.29}$$

denote the vector of input processes generated at source node s and

$$\underline{Z} = (\underline{Z}(t_1), \underline{Z}(t_2), \cdots, \underline{Z}(t_N)) \tag{3.30}$$

where the $1 \times r$ vector $\underline{Z}(t_n)$ denotes the output processes at sink node t_n. Given an $r \times |E|$ matrix A, an $|E| \times |E|$ matrix $(I - F)^{-1}$ and an $|E| \times rN$ matrix B^T. A corresponding $r \times rN$ transfer matrix $M = A(I - F)^{-1}B^T$ can describe the relationship between $\underline{X}(s)$ and $\underline{Z}$.

$$\begin{aligned} \underline{Z} &= (\underline{Z}(t_1), \underline{Z}(t_2), \cdots, \underline{Z}(t_N)) \\ &= \underline{X}(s)A(I - F)^{-1}B^T \\ &= \underline{X}(s)A(I - F)^{-1}[B_1^T, B_2^T, \cdots, B_N^T] \\ &= \underline{X}(s)[M_1, M_2, \cdots, M_N] \end{aligned} \tag{3.31}$$

where $r \times r$ submatrix $A(I - F)^{-1}B_n^T$ describes the relationship for a connection c_n.

By the theorem[5] derived by Ahlswede *et al.*, they guarantee the existence of a coding strategy that ensures the feasibility of the desired connections if the rate of each connection $c_n,\ \forall\ n \in 1, 2, \cdots, N$, satisfies the *max-flow bound*.

Let a delay-free acyclic network G and a set of desired connections ℓ be given. The following theorem states the necessary and sufficient condition of multicast connections to be established successfully by using an algebraic characterization.

Theorem 5 [9] *The network coding problem* (G, ℓ) *is solvable if and only if the* max-flow bound *is satisfied for all connections in* ℓ.

It means that for the multicast at rate r satisfying the *max-flow bound*, there exists a feasible set of network code over a infinite filed GF(2^m). Such that the determinant of

transfer matrix M_n are nonzero in the ring of GF$(2^m)[\underline{\xi}]$ for all $t_n \in T$. Let

$$\begin{aligned} f(\underline{\xi}) &= \prod_{n=1}^{N} det(A(I-F)^{-1}B_n^T) \\ &= (-1)^{rN(|E|+1)} \prod_{n=1}^{N} det(\begin{pmatrix} A & 0 \\ I-F & B_n^T \end{pmatrix}) \end{aligned} \tag{3.32}$$

It is intuitive that the maximal degree of $f(\underline{\xi})$ in any variable is at most N. By using algorithm in Table I, there always exists a set of network codes over GF(2^m), which

$$m > log_2 N \tag{3.33}$$

Further, a feasible set of network codes in a certain finite field GF(2^m), where

$$m \leq \lceil loge_2(N+1) \rceil, \tag{3.34}$$

can also be found. The determination of the coefficients a_i in the algorithm renders a network code that all the transfer matrices between the single source node and any sink node are invertible. Also, the algorithm provides a upper bound on the degree of the extension of GF(2) that needs to consider.

■ Linear Network Codes For Robust Networks

After accomplishing the construction of network codes, it is annoying to re-construct the network codes if link fail happens. In order to avoid this situation, the network codes is better chosen to prevent the re-construction if some links fail. In the other words, this network codes still works even link is disconnected. In the following, we introduce such code design in detail.

Edges in a network may fail. You may ask that under which failure pattern a successful network usage is still guaranteed. Here, assume that link failure is a link either working perfectly or is effectively deleted from the network. A link failure pattern can be identified with binary vectors $\underline{h}$ of length $|E|$ such that each position in $\underline{h}$

$$\underline{h} = (h_1, h_2, \cdots, h_{|E|})^T \tag{3.35}$$

is associated with one edge in G. If a link fails, the corresponding position in $\underline{h}$ equals one. Otherwise, the entry in $\underline{h}$ corresponding to the link equals zero. A network is solvable under link failure pattern $\underline{h}$, if it's solvable once the links corresponding to $\underline{h}$ have been deleted. Such network codes is important because

1. no new network codes has to be found and distributed in the network if a failure pattern $\underline{h} \in \mathcal{H}$ occurs,

2. the individual nodes in the network can be oblivious to the failure pattern.

Suppose that e_i is the failing link, the effect of a failing link can be achieved by setting parameters β_{e',e_i}, $\beta_{e_i,e''}$ and $\alpha_{e_i,\ l}$ to zero for all e', e'' and l. Define $\mathbb{B}_{e_i}$ be the set of parameters ξ_i that are identified with $\beta_{e',e_i}, \beta_{e_i,e''}$ and $\alpha_{e_i,\ l}$ to zero for all e', e'' and l

$$\mathbb{B}_{e_i} = \{\xi_i : \ \xi_i \text{ is identifed with } \beta_{e',e_i},\ \beta_{e_i,e''} \text{ and } \alpha_{e_i,\ l} \text{ to zero for all } e',\ e'' \text{ and } l\}. \tag{3.36}$$

For any particular link failure pattern $\underline{h}$, define $\mathbb{B}(\underline{h})$ as

$$\mathbb{B}(\underline{h}) = \bigcup_{e_i:\ h_i=1} \mathbb{B}_{e_i} \tag{3.37}$$

For a network G and $M(\underline{\xi})$ be the corresponding system matrix.

$$M(\underline{\xi}) = [M_1(\underline{\xi}), M_2(\underline{\xi}), \cdots, M_N(\underline{\xi})] \tag{3.38}$$

The network $G_{\underline{h}}$ is obtained by deleting the failing links $\underline{h}$ and the corresponding $M_{\underline{h}}(\underline{\xi})$

$$\begin{aligned} M_{\underline{h}}(\underline{\xi}) &= M(\underline{\xi})|_{\xi_j=0:\ \xi_j \in \mathbb{B}_{\underline{h}}} \\ &= (M_{\underline{h},1}(\underline{\xi}), M_{\underline{h},2}(\underline{\xi}), \cdots, M_{\underline{h},N}(\underline{\xi})) \end{aligned} \tag{3.39}$$

Let $\mathcal{H}$ be the set of failure patterns, such that $(G_{\underline{h}}, \ell)$ is solvable for any $\underline{h} \in \mathcal{H}$.

Given any $\underline{h} \in \mathcal{H}$

$$f_{\underline{h}}(\xi) = \prod_{n=1}^{N} det(M_{\underline{h},n}(\underline{\xi})) \tag{3.40}$$

Since $(G_{\underline{h}}, \ell)$ is solvable, $f_{\underline{h}}(\xi)$ is nonzero element over the polynomial ring $\mathbb{F}[\underline{\xi}]$. It can be further derive that

$$\begin{aligned} f_R(\xi) &= \prod_{\underline{h}\in\mathcal{H}} f_{\underline{h}}(\xi) \\ &= \prod_{\underline{h}\in\mathcal{H}} \prod_{n=1}^{N} det(M_{\underline{h},n}(\underline{\xi})) \end{aligned} \tag{3.41}$$

$f_R(\xi)$ is still a nonzero element over $\mathbb{F}[\underline{\xi}]$. By lemma 1, there exist a set of linear network codes, such that $f_R(\xi) \neq 0$ over the infinite field GF(2^m). Intuitively, the codes can be found by the above the algorithm over a certain finite field GF(2^m), where

$$m \leq \lceil log_2(|\mathcal{H}| \cdot N + 1) \rceil \tag{3.42}$$

The factor $|\mathcal{H}|$ in (3.42) will make the upper bound of the filed GF(2^m) bigger than the one without robustness concerning and complicate the the searching algorithm. And this is the price to organize the robust multicast.

3.3 General Network Coding Problem

Given a network G and an arbitrary set of connections, ℓ. The following theorem states a succinct condition under which a network problem (G, ℓ) is solvable.

Theorem 6 *[9](GeneralizedMin−cutMax−flowcondition) Let an acyclic, delay-free linear network problem (G, ℓ) be given. Since the relationship between $\underline{X}$ and $\underline{Z}$ can be expressed as*

$$\underline{Z} = \underline{X}M \tag{3.43}$$

Then each connection can also be expressed a linear relation. Let $M = \{M_{i,j}\}$ be the corresponding transfer matrix relating the set of input nodes to the set of output nodes. The network problem is solvable if and only if there exists an assignment of numbers to $\underline{\xi}$ such that

1. *$M_{i,j} = 0$ for all pairs (v_i, v_j) of nodes such that $(v_i, v_j, \mathcal{X}(v_i, v_j)) \notin \ell$*

2. *If* $\{(v_{i_1}, v_j, \mathcal{X}(v_{i_1}, v_j)), (v_{i_2}, v_j, \mathcal{X}(v_{i_2}, v_j)), \cdots, (v_{i_L}, v_j, \mathcal{X}(v_{i_L}, v_j))\} \subseteq \ell$ *the submatrix* $[M^T_{i_1,j}, M^T_{i_2,j}, ..., M^T_{i_L,j}]$ *is a non singular* $\nu(v_j) \times \nu(v_j)$ *matrix.*

It means that for each sink v_j, the corresponding vector $\underline{Z}(v_j)$ can be expressed as

$$\underline{Z}(v_j) = \underline{X}_{d_j} M_{d_j} + \underline{X}_{I_j} M_{I_j} \tag{3.44}$$

where $\underline{X}_{d_j}$ contains the desired information of v_j, submatrix $M_{d_j} = [M^T_{i_1,j}, M^T_{i_2,j}, ..., M^T_{i_L,j}]$ is $\nu(v_j) \times \nu(v_j)$, $\underline{X}_{I_j}$ is the interference with respect to v_j and M_{I_j} is a interference transform matrix that induce noise term into $\underline{Z}(v_j)$.

$$\underline{X}_{d_j} = M^{-1}_{d_j}(\underline{Z}(v_j) - \underline{X}_{I_j} M_{I_j}) \tag{3.45}$$

For each sink v_j to recover desired information, it needs M_{d_j} nonsingular and interference transform matrix M_{I_j} to be zero matrix.

However, checking the two conditions is a tedious task as to find a solution $\underline{\xi}$ and the theory of Gröbner bases provides a structured approach to this problem. For more detail, see [9].

Chapter 4

Linear Network Coding in Overlay Networks

Traditionally network architectures distinguish between two types of entities: end systems (hosts) and the network (switches and routers). In the internet architecture, the IP layer implements a minimal functionality − a best-effort unicast datagram service, and the end system implements all other important functionality such as error, congestion, and flow control. For a network standpoint, the multicast abstraction can be implemented in several ways:

1. *One-to-all unicast (or called naive unicast)*: This approach implements a single source multicast abstraction using an underlying unicast network layer, required no explicit multicast support from the IP layer. The source uses a separate unicast transport connection to each of the receivers. The approach here is simple and needs no support from the routers. But the drawback is that the links near the source node are likely to experience high *link stress*, which is referred to as the number of identical copies of a packet carried by a physical link and eventually decreases the multicast rate.

2. *IP multicast*: The approach is based on the explicit multicast support at the IP layer. That is a new multicast protocols at IP layer such as IGMP, DVMRP, PIM and MOSPF, and new routers that can handle multicast abstraction. The

approach is efficient and higher multicast rate than any other methods, but requires a significant amount of infrastructures to set up and maintain. There are several drawbacks that have so far prevented the service from being widely deployed. First, IP layer multicast requires routers to maintain per group state, which not only violates the "stateless" architecture but also introduces high complexity and serious scaling constraints. Second, the network vulnerable to flooding attacks by malicious sources. Third, the requirement of dynamically obtain a globally unique address. Finally, the IP multicast calls for the changes at the infrastructural level, a new multicast routers need to be used, this slows down the pace of deployment. A brief introduction of IP multicast see [14].

3. *Application-level multicast (or called end-system multicast)*: A model in which multicast related features, such as group membership management, multicast routing and packet duplication are implemented at end systems and base on only unicast IP service. The end systems (source and receivers) form an overlay network, which is a virtual network. The nodes on the virtual network are all end systems and the link on the virtual network is actually a path in the physical network. For a multicast abstraction the end systems construct an multicast tree on overlay network and the transmission between node and node is accomplished by using IP unicast. In addition, the reliable dilivery, flow control congestion control, and security can be significantly handle by end systems. However, end system multicast introduces duplicate packets on physical links (increase link stress) and incur larger end-to-end delay (increase link stretch, which is defined as the ratio of path length from the source to the multicast receiver along the overlay to the length of the direct unicast path), hence decrease the multicast rate and increase delay. In [12], they propose an Narada protocol, which end system self-organize in to an overlay structure using a fully distributed protocol.

Due to the lack of widely available Internet protocol multicast service, lots of research suggest using application-layer multicast by taking all of the advantages of application-layer overlay networks. Application-layer overlay networks have two properties: (1) the network nodes in a overlay network are end systems and have capabilities far beyond basic operations of storing and routing, and (2) the overlay topology, residing above a densely connected Internet wide-area network, can be constructed to suit one's purposes.

Base on the advantages of end-system (application-layer), Zhu *et al.* implements the multicast abstraction with network coding[11] and attempts to increase the end-to-end throughput. Taking the two benifits of application-layer multicast over IP multicast: (1) multicast support in IP layer is not required, and (2) data is transmitted between virtual nodes via unicast, efficiently exploiting all existing security, flow control, and reliable delivery mechanism that are already available and mature. They use two novel view as oppose to traditional multicast abstraction. First, they depart from conventional view that data can only be routing by overlay nodes. Rather, these overlay nodes also have the full capability of encoding and decoding data at the message level using linear codes. Second, they also depart from traditional wisdom that the multicast topology from source s to sink nodes to be tree, and propose an distributed algorithm to construct a two-redundant multicast graph as mlulticast topology that embedded in the virtual overlay network. In the two-redundant multicast graph, the network codes could be designed without the knowledge of network topology using a distributed algorithm.

Base on this framework, we propose a new code design algorithm that is easier than the algorithm proposed by Zhu *et al.* and the code will accomplish higher end-to-end throughput than theirs. In the following, we give a detailed description of [11].

4.1 Two-Redundant Multicast Graphs

A k-redundant multicast graph is a directed acyclic graph (DAG) which has the following two properties:

1. The set of all nodes V is the union of three disjoint subsets $\{s\} \cup V_I \cup T$ and the total number of nodes are

$$N_{total} = |V| = |\{s\}| + |V_I| + |T| = 1 + N_I + N, \tag{4.1}$$

where s is source node, V_I is the set of intermediate nodes, and T is the set of receiver nodes.

   (a) The in-degree and out-degree of s are $In(s) = 0$ and $Out(s) > 0$, respectively.

   (b) Let v_i denote the intermediate node,
   which $1 \leq In(v_i) \leq k$ and $Out(v_i) \geq 0$, $\forall\ v_i \in V_I$.

   (c) Let t_n denote the receiver node,
   which $In(t_n) = k$ and $Out(t_n) \geq 0$, $\forall\ t_n \in T$.

2. Assume that each link has unit bandwidth. For any receiver node whose indegree is k, then

$$maxflow(s, t_n) = k,\ \forall\ t_n \in T \tag{4.2}$$

The intermediate nodes are dedicated high-degree relay nodes, which are end hosts or proxy servers connected to high-bandwidth physical links in the overlay network, which do not belong to the set of receiver nodes, and such a pool of dedicated nodes is the price to exploit network coding to increase end-to-end throughput. Why they use two-redundant multicast graph rather than k-redundant mlulticast graph? As k increase, it's difficult to find multiple good paths (i.e., large path bandwidth) from source s to each receiver with limited intermediate nodes and increased link stress and as k increase, network code assignment algorithm (introduce latter) become more complex and averse to the dynamics of node joints and departures. The following three propositions establish the achievablility of network coding, the sufficiency of a maximum in-degree of 2 and the necessity of non-empty intermediate nodes in a two-redundant acyclic multicast graph.

Proposition 3 *For any receiver node t_n with $In(t_n) = 2$, it has two disjoint path from source node s if and only if t_n has $maxflow(s, t_n) = 2$.*

Proposition 4 *It's not necessary for any node in a two-redundant multicast graph to have in-degree of greater than 2 to obtain two disjoint paths for each receiver from s.*

Proposition 5 *If a two-redundant multicast graph contains only source and receivers, i.e., $V_I = \emptyset$, then two-redundant multicast graph contains a directed cycle.*

Each receiver has two disjoint paths from the source and both path should be carefully chosen to maximize the throughput to the receiver. The degree of a node represents the total number of neighbors in the virtual network. For optimizing the virtual link bandwidth, both the rudimentary tree and the two-redundant multicast graph have the degree constraint: every virtual node has degree $\leq \triangle$; i.e., $\triangle$ is the maximum virtual node degree and in the following set $\triangle = 4$. Minimizing link stress is paramount since it determine the available bandwidth in a virtual link and the minimization of the link stress is cover by imposing a maximum virtual node degree constraint, $\triangle$. The graph is constructed by the following three steps:

- Step 1: Rudimentary graph construction;
- Step 2: Rudimentary tree construction;
- Step 3: Two-redundant multicast graph construction.

and the construction algorithm is distributed.

4.1.1 Rudimentary graph construction

When a new node joints the group, it's given a set of nodes already in the group by a bootstrapping node. These are its initial neighbors in the rudimentary graph. The node then contacts all its neighbor so they are made aware of it. Every node maintains two set of lists, one for storing the addresses of neighbors (neighbor lists) and another for

storing the addresses of all the nodes it knows about in the group (group lists). Node exchanges group information with its neighbor's group lists, and update accordingly. By the exchanging processes, the information about a node will eventually propagate through the rudimentary graph.

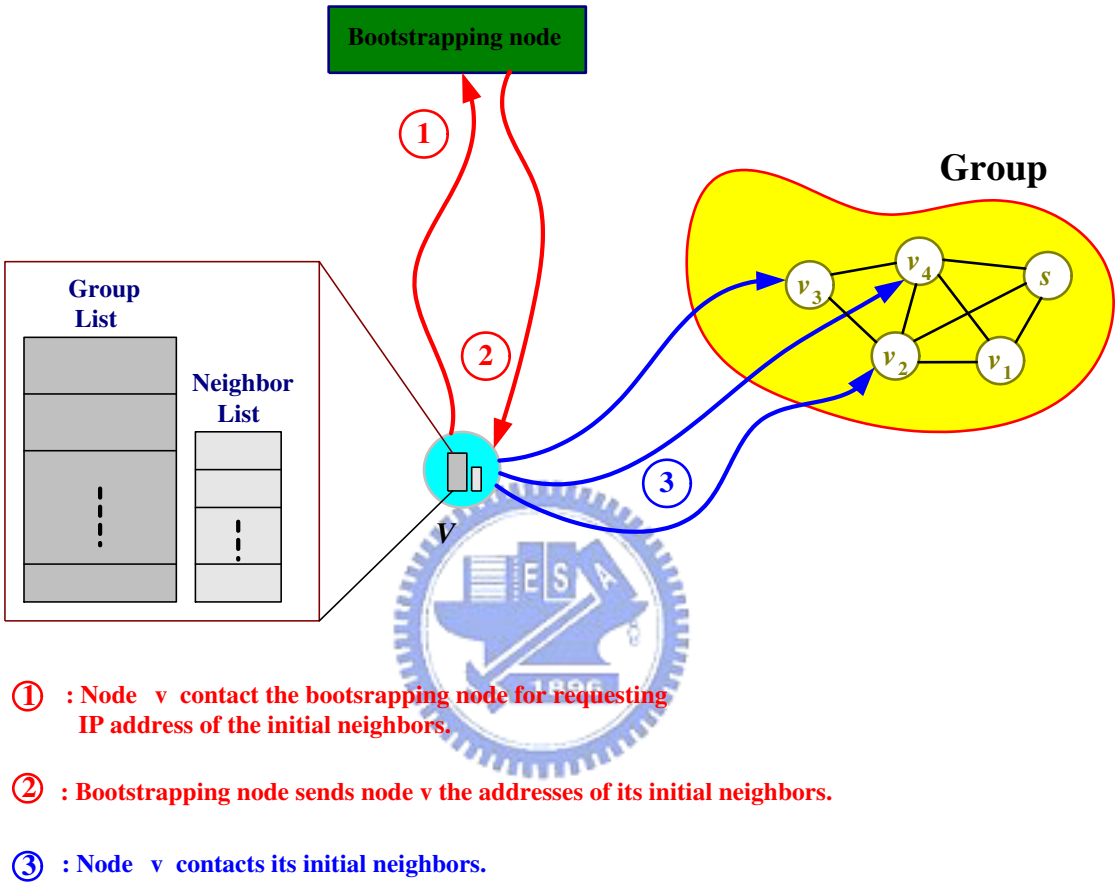


Figure 4.1: Join procedure for a new node.

Fig. 4.1 shows the procedures when a new node join the group .

Each virtual link e_i in the rudimentary graph has associated with a 2-tuple weight $w(e_i) = (\beta_i, \lambda_i)$, which β_i is the link bandwidth and λ_i is the link latency (delay). Each path $p = (e_1, e_2, \cdots, e_l)$ in the rudimentary graph also has an associated weight $w(p) = (\beta_p, \lambda_p)$, which $\beta_p = min(\beta_i, i = 1, 2, \cdots, l)$ and $\lambda_p = \sum_{i=1}^{l} \lambda_i$. Let $w(p_1) = (\beta_{p_1}, \lambda_{p_1})$ and $w(p_2) = (\beta_{p_2}, \lambda_{p_2})$ be the weights of p_1 and p_2, respectively. Then p_1 is better than

p_2 if $\beta_{p_1} > \beta_{p_2}$, or $\beta_{p_1} = \beta_{p_2}$ and $\lambda_{p_1} < \lambda_{p_2}$. In order to optimize the performance, the dynamics of adding high-quality links and dripping poor-quality links is vital to the performance of the multicast scheme.

(a) Neighbor-adding process:

- Periodically, each node $v_i \in V$ chooses randomly another node v_j in the group lists that are not a neighbor and by sending a probing message, estimate the bandwidth and latency of the virtual link, (v_i, v_j).
- If $w((v_i, v_j))$ is better than most of links of its current neighbors, and both v_i and v_j have not yet excess degree constraint, then v_j is added as a neighbor of node v_i and link (v_i, v_j) is added in the rudimentary graph.
- The degree of a node $v_i \in V$ in the rudimentary graph has the following properties:
  1. For source node s, the number of its neighbors that are intermediate nodes are no larger than 3.
  2. For every intermediate node v_i, the number of its neighbors that are intermediate nodes must no larger than 4.
  3. For every node, the total number of its neighbors can have more than 4.

(b) Neighbor-dropping process:

- If a current neighbor $v_{j'}$ of v_i has worse link than the links v_i has to its neighbors, and both v_i and $v_{j'}$ use link $(v_i, v_{j'})$ very rarely, then v_i drops $v_{j'}$ as its neighbor and $(v_i, v_{j'})$ is removed from the rudimentary graph .

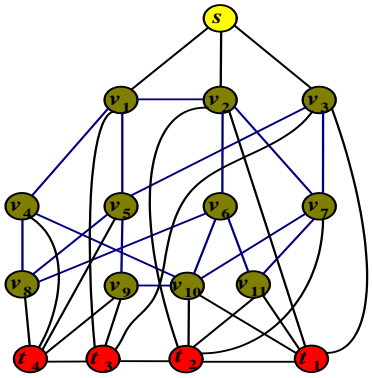
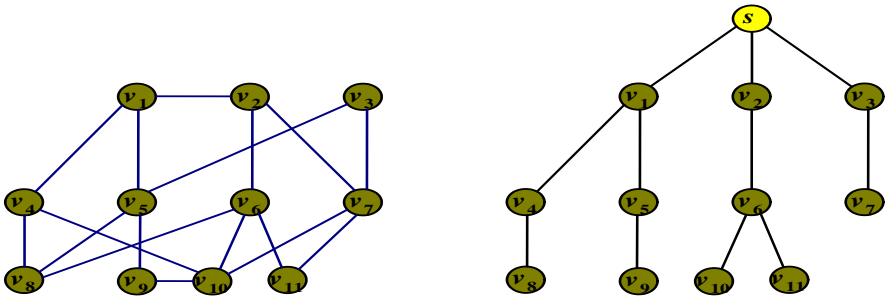


Figure 4.2: An example of rudimentary graph.

Fig. 4.2 shows an example of rudimentary graph.

4.1.2 Rudimentary tree construction

The core graph is the subgraph of the rudimentary graph with the set of intermediate nodes V_I and all the incident links. In order to avoid the connectionless of the two-redundant multicast graph, the core graph must be constructed as a connected graph. The rudimentary tree is built from the subgraph consisting of the source node s and the core graph by exploiting the distributed algorithm proposed in [13] based on distance vectors that finds the *shortest widest paths*.



(a) core graph **(b) rudimentary tree**

Figure 4.3: An example of core graph and rudimentary tree.

Fig. 4.3 shows a core graph and a rudimentary tree of Fig. 4.2, respectively.

4.1.3 Two-redundant multicast graph construction

The two-redundant multicast graph G_{2r} is constructed by taking the rudimentary tree as a basis and add links from the rudimentary graph or randomly establish another good virtual links, i.e., higher bandwidth link, that are not belong to the rudimentary graph. The construction of the two-redundant multicast graph adhere the following rules for source node s and every intermediate node $v \in V_I$:

1. Source s has ϕ intermediate nodes as children, i.e., $Out(s) = \phi$ and

$$2 \leq \phi \leq 3; \tag{4.3}$$

2. Total degree of node v is

$$degree(v) = (In(v) + Out(v)) \leq 4; \tag{4.4}$$

3. The in-degree of node v is

$$1 \leq In(v) \leq 2; \tag{4.5}$$

4. Number of children of v that are receivers is less than or equal to $(3 - In(v))$.

Rule 4 ensures that the construction of G_{2r} is always successful as long as the intermediate nodes are large enough.

An intermediate node v is called a *leaf intermediate node* if none of v's children in G_{2r} is an intermediate node. An intermediate node v is *saturated* if either $degree(v) = 4$, or v is a leaf intermediate node and $degree(v) = 3$. The breadth-first search is used for searching unsaturated nodes.

■ **Breadth-First Search Primitive**:

When a node firs becomes to saturated, it sends that information to its parent node of the rudimentary tree. Recursively, a node which is the root of subtree T_r, knows that T_r is saturated when all its children have sent saturation notification to it. The

construction of two-redundant multicast graph G_{2r} is initialized to the rudimentary tree, so initially it has no leaf receivers. Adding each receiver $t_n \in T$ two links to the intermediate nodes, so that construct two disjoint paths, p_f and p_s, from the source node s to receiver t_n.

(a) First Path Construction:

To construct p_f, receiver t_n first contacts all its neighbors in the rudimentary graph that are intermediate nodes and unsaturated and let $\{v_i\}$ denote the unsaturated neighbors of t_n. Receiver t_n compares the paths $\{p_i\}$, which consist of the tree paths from s to $\{v_i\}$ and the links $\{(v_i, t_n)\}$ from v_i to t_n, then selects the best path. If all of t_n's intermediate neighbors are saturated, then t_n initiates a breadth-first search of the tree to find the first ψ unsaturated intermediate nodes and let $\{v_{i'}\}$ denote the ψ unsaturated neighbors that found by the breadth-first search. Similarly, receiver t_n compares the ψ paths and select the best path. Here, p_f represents the best path and v_f be the corresponding unsaturated intermediate node.

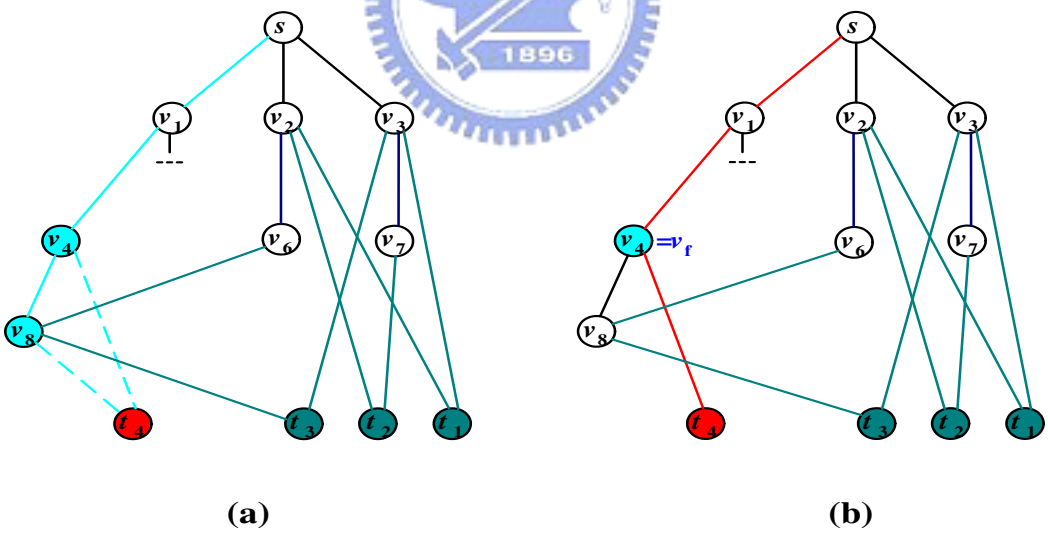


Figure 4.4: First path comparison and p_f selection of receiver t_4.

In Fig. 4.4(a), receiver t_4 compares two paths $p_4 \cup (v_4, t_4)$ and $p_8 \cup (v_8, t_4)$. In Fig. 4.4(b), assume that $p_4 \cup (v_4, t_4)$ is a better path, then t_4 decides $p_4 \cup (v_4, t_4)$ to be its p_f

and v_4 to be its v_f.

TABLE II
PROCEDURE FOR CONSTRUCTION p_f FOR t_n

<**Main**>
- Leaf receiver t_n:
  - · **If** not all t_n's intermediate node neighbors, $\{v_i\}$, are saturated
    - ‡ v=Find-best-path$(t_n, \{v_i\})$
    - ‡ $v_f = v$ and $p_f = p \cup (v, t_n)$
  - · **else if** all neighbors of t_n are saturated
    - ‡ Breadth-first search of tree, halt when ψ unsaturated nodes $\{v_{i'}\}$ found
    - ‡ v=Find-best-path$(t_n, \{v_{i'}\})$
    - ‡ $v_f = v$ and $p_f = p \cup (v, t_n)$

<**Subroutine**>
- Find-best-path$(t_n, \{v_i\})$:
  - · request $\{v_i\}$ for weights $\{w(p_i) = (\beta_i, \lambda_i)\}$ of their paths in the tree
  - · **for each** unsaturated node v_i
    - ‡ β=bandwidth of link (v_i, t_n)
    - ‡ compute $w(p_i \cup (v_i, t_n)) = (min(\beta_i, \beta), \lambda_i + \lambda)$
    - ‡ choose the best (shortest widest) path, p
    - ‡ return v that corresponds to p.

Table II shows the procedures to construct first path, p_f, of t_n.

(b) Second Path Construction:

To construct p_s, receiver t_n find ψ unsaturated intermediate node $\{v_i\}$, which are not in p_f, from its neighbors. If there are fewer than ψ such neighbors, then t_n randomly probes intermediate nodes that are unsaturated and not in p_f. For each v_i, it checks if its tree path p_i intersect the first path p_f. If it does not intersect, then v_i replies p_i and $w(p_i)$ to t_n, else, it finds an alternative path from source node s.

- If $In(v_i) = 2$, an alternative path may already exit, then v_i replace this new path as p_i and replies p_i and $w(p_i)$ to t_n.
- If $In(v_i) = 1$, then v_i sends a message to the a child node c of s that is different from the child who is upstream from v_i and request for c to do the breadth-first search of its own subtree. Node c replies to v_i the first unsaturated or leaf intermediate

node v_i', then v_i replies t_n with $w(p_i' \cup (v_i', t_n))$ and $p_i' \cup (v_i', t_n)$, which p_i' is the tree path from s to v_i'.

After t_n receives $w(p_i)$ and $w(p_i' \cup (v_i', t_n))$ from all the v_i, receiver t_n selects the best path among $p_i \cup (v_i, t_n)$ and $p_i' \cup (v_i', t_n)$. Here, p_s represents the best path and v_s be the corresponding unsaturated intermediate node or leaf intermediate node.

TABLE III
PROCEDURE FOR CONSTRUCTION p_s FOR t_n

<**Main**>
• Leaf receiver t_n:
  · Find ψ unsaturated intermediate nodes $\{v_i\}$ not in p_f,
    from its neighbors and/or random probing.
  · Send p_f to each v_i and request best path p_i from s to v_i
    that does not intersect with p_f, and its weight $w(p_i)$.
  · v=Find-best-path$(t_n, \{v_i\}, \{p_i\}, \{w(p_i)\})$

<**Subroutine 1**>
• Find-best-path$(t_n, \{v_i\}, \{p_i\}, \{w(p_i)\})$:
  · **For each** intermediate node v_i
    ‡ $(p_i, w(p_i))$=Check(v_i)
    ‡ β=bandwidth of link (v_i, t_n)
    ‡ compute $w(p_i \cup (v_i, t_n)) = (min(\beta_i, \beta), \lambda_i + \lambda)$
  · choose the best (shortest widest) path, p
  · return v that corresponds to p.

<**Subroutine 2**>
• Check(v_i):
  / Upon receiving p_f and request for path from s to v_i disjoint from p_f /
  · **If** v_i's tree path p_i doest not intersect with p_f
      then return p_i and $w(p_i)$
  · **else if** v_i's tree path intersects p_f then
    ‡ **if** v_i has an alternative path p_i' from s then
        return p_i' and $w(p_i')$
    ‡ **else if** v_i has $In(v_i) = 1$ then
      ◇ contact a different child c of s than the one whose subtree v_i is in.
      ◇ c conducts breadth-first search of its subtree
        and returns to v_i the first unsaturated or leaf intermediate node v
      ◇ return $p_i = p_v \cup (v, v_i)$ and $w(p_i)$

Table III shows the procedures to construct second path, p_s of t_n.

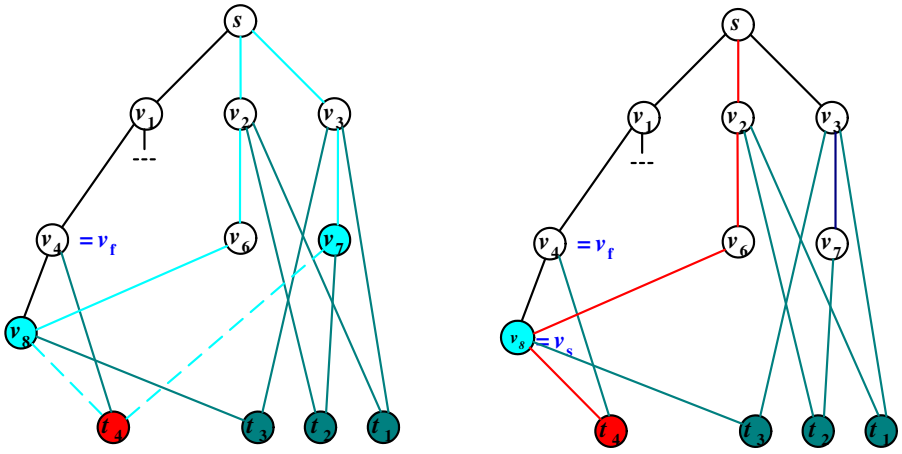


Figure 4.5: Second path comparison and p_s selection of receiver t_4.

In the left-hand side of Fig. 4.5, receiver t_4 compares two paths $p_7 \cup (v_7, t_4)$ and $p_6 \cup (v_6, v_8) \cup (v_8, t_4)$. In the other side of Fig. 4.5, assume that $p_6 \cup (v_6, v_8) \cup (v_8, t_4)$ is a better path, then t_4 decides $p_6 \cup (v_6, v_8) \cup (v_8, t_4)$ to be its p_s and v_8 to be its v_s.

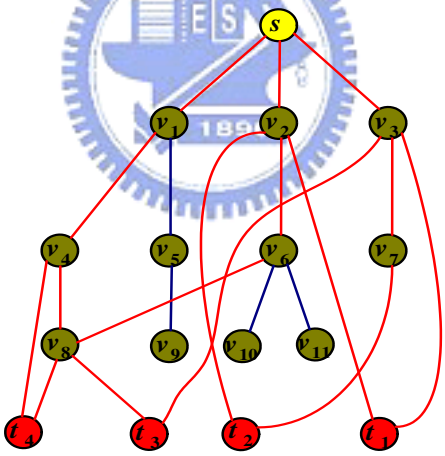


Figure 4.6: An example of two-redundant multicast graph.

Fig. 4.6 shows an two-redundant multicast graph after the construction of p_f and p_s for each sink node, and each node satisfies the properties of G_{2r}.

After finishing the construction of the two-redundant multicast graph, the multicast rate is then determined by the *minimum link bandwidth of the graph*.

With the restriction of maximum degree $\triangle = 4$ on the nodes in the multicast graph, the number of intermediate nodes needed increases with the number of receivers. For a given number of intermediate nodes N_I, Zhu *et al.* gives the maximum (N_{max}) and minimum (N_{min}) number of receivers that the intermediate nodes can support.

$$N_{max} = \lfloor((\triangle - 2)N_I + 1)/2\rfloor \tag{4.6a}$$

$$N_{min} = \lfloor((\triangle - 4)N_I + 1)/2\rfloor \tag{4.6b}$$

4.2 Linear Code Design over Prime Fields

Once the graph G_{2r} is constructed, a linear codes must be found to realize the linear coding multicast, such that doubling end-to-end throughput. The algorithm proposed in [9] and [8] are centralized and need to know the network topology, then generate network codes to each node. Due to the special graph G_{2r}, a distributed algorithm can be used to design linear network codes without the knowledge of the network topology. Further, the network coding is only needed at the source node s and the in-degree 2 intermediate nodes.

Since each receiver has two disjoint paths from source node s. Then s multicast the source vector $\underline{X}(s) = (X_1(s), X_2(s))$ to each receiver nodes. Both symbols $X_1(s)$ and $X_2(s)$ should be recovered by each receiver. Define $\underline{d}(e_i) = (d_1(e_i), d_2(e_i))^T$ be the encoding vector of link e_i. Let $e_j^{I_i}$ be the jth incoming link of node v_i and denote the transmitted symbol on the link as $Y(e_j^{I_i})$, and $e_{j'}^{O_i}$ be the $j'th$ outgoing link of node v_i and denote the transmitted symbol on the link as $Y(e_{j'}^{O_i})$. For every intermediate node $v_i \in V_I$ if $In(v_i) = 1$, then it just forwards the symbol received from its incoming link with no encoding onto all the outgoing links, i.e., $Y(e_j^{Oi}) = Y(e_1^{I_i})$, for all $e_j^{Oi} \in Out(v_i)$. Otherwise $In(v_i) = 2$, then it encodes the two received symbols using network code, which is denoted by $(\beta_{i_1}, \beta_{i_2})^T$ and forward the coded symbol onto all the outgoing

links. That means

$$Y(e_j^{Oi}) = (Y(e_1^{Ii}), Y(e_2^{Ii})) \cdot (\beta_{i_1}, \beta_{i_2}) \tag{4.7}$$

for all $e_j^{Oi} \in Out(v_i)$.

The network code vectors for in-degree 2 intermediate nodes are obtained from a distributed algorithm. There are two phases – *AssignCodes* phase and *DisseminateCodes* phase.

Before starting the algorithm, a function that generate a sequence of 2-dimensional pairwise linear independent vectors must be introduced. There is a function $gen(\delta)$ that generates a sequence of δ transformation vectors $\{(\rho_1, q_1)^T, (\rho_2, q_2)^T, \cdots, (\rho_\delta, q_\delta)^T\}$ such that:

- ρ_i, q_i are elements in a prime field;
- (ρ_i, q_i) and (ρ_j, q_j) are linear independent, $\forall\, i \neq j$;
- $(\rho_i, q_i)^T$ defines a linear transformation of source vector $(X_1(s), X_2(s))$:

$$Y_i = (X_1(s), X_2(s))(\rho_i, q_i)^T = \rho_i \cdot X_1(s) + q_i \cdot X_2(s) \tag{4.8}$$

Let $\mathcal{Y} = \{Y_1, Y_2, \cdots, Y_m\}$ and $gen(\delta) = \{(\rho_1, q_1)^T, (\rho_2, q_2)^T, \cdots, (\rho_\delta, q_\delta)^T\}$, which any two vectors in $gen(\delta)$ are pairwise independent. For each receiver t_n, it can recovers $(X_1(s), X_2(s))$ if it receives any two distinct elements in $\mathcal{Y}$ and also knows the corresponding transformation vectors.

TABLE IV
VECTOR SEQUENCE GENERATION ALGORITHM

- **Algorithm** $gen(\delta)$
  - $\cdot$ $i \longleftarrow 1$, $j \longleftarrow 1$
  - $\cdot$ primes$[i] \longleftarrow 2$, vectors$[j] \longleftarrow (2, 2)$
  - $\cdot$ Iteration:
    - ‡ Find the next prime p, smallest $p >$ primes$[i]$
    - ‡ **For** $k = 1$ to i
      - $\diamond$ $j \longleftarrow j + 1$
      - $\diamond$ vectors$[j] \longleftarrow$ (primes$[k], p$)
      - $\diamond$ **If** $j = \delta$ **then** halt
        - $\triangleright$ $j \longleftarrow j + 1$
        - $\triangleright$ vectors$[j] \longleftarrow$ (primes$[k], p$)
    - ‡ $i \longleftarrow i + 1$
    - ‡ primes$[i] \longleftarrow p$

Table IV shows the algorithm that generates δ pairwise linear independent transformation vectors.

4.2.1 AssignCodes phase

Source node s first broadcasts a message through the rudimentary tree to initiate the *AssignCodes* phase. If an intermediate node v_i has two incoming links in G_{2r}, then it sends a message containing its address to s requesting a transformation vector. Suppose L requests were received and s has ϕ children, then s generates $(L + \phi)$ transformation vectors using $gen(L + \phi)$. Source node sends to each requesting intermediate node one of the first L vectors and then each intermediate node v_i will receive a transformation vector (ρ, q). This vector will be the encoding vector of its outgoing links.

4.2.2 DisseminateCodes phase

The last ϕ transformation vectors are sent by s to its children, one to each child and these vectors are the symbol vectors of links between source node and its children. For each intermediate node, it passes its transformation vector to all the downstream nodes. For in-degree 1 intermediate node, its transformation vector is obtained from its parent node. Eventually, each node $v \in V$ has the knowledge about the transformation

vectors of its incoming and outgoing links, and then derives the network code vectors or decoding matrices. For each intermediate node $v_i \in V_I$ and $In(v_i) = 2$, it can derive the network code vector by using the transformation vectors of its parent nodes and its own transformation vector. Let (ρ_{v_i}, q_{v_i}) be the transformation vector of v_i, and (ρ_{i_1}, q_{i_1}), (ρ_{i_2}, q_{i_2}) be the two transformation vectors of its parent nodes. The network code vector will be

$$(\beta_{i_1}, \beta_{i_2})^T = \begin{pmatrix} \rho_{i_1} & \rho_{i_2} \\ q_{i_1} & q_{i_2} \end{pmatrix}^{-1} \begin{pmatrix} \rho_{v_i} \\ q_{v_i} \end{pmatrix} \tag{4.9}$$

The structure of G_{2r} will guarantee the parent nodes of each in-degree 2 intermediate node from different intermediate nodes, such that (ρ_{i_1}, q_{i_1}) and (ρ_{i_2}, q_{i_2}) are linear independent and the inverse of the matrix $\begin{pmatrix} \rho_{i_1} & \rho_{i_2} \\ q_{i_1} & q_{i_2} \end{pmatrix}$ exits.

For each receiver node $t_n \in T$, it can derive the decoding matrix by using the transformation vectors of its parent nodes. Suppose that (ρ_{n_1}, ρ_{n_1}) and (ρ_{n_2}, ρ_{n_2}) be the two transformation vectors of its parent nodes, and assume that Y_{n_1} and Y_{n_2} be the two incoming symbols of t_n. The decoding matrix will be

$$\begin{pmatrix} \varepsilon_{n_1} & \varepsilon_{n_3} \\ \varepsilon_{n_2} & \varepsilon_{n_4} \end{pmatrix} = \begin{pmatrix} \rho_{i_1} & \rho_{i_2} \\ q_{i_1} & q_{i_2} \end{pmatrix}^{-1} \tag{4.10}$$

Then the source vector can be recovered by

$$(X_1(s), X_2(s)) = (Y_{n_1}, Y_{n_2}) \begin{pmatrix} \varepsilon_{n_1} & \varepsilon_{n_3} \\ \varepsilon_{n_2} & \varepsilon_{n_4} \end{pmatrix} \tag{4.11}$$

The structure of G_{2r} will guarantee the parent nodes of each receiver from different intermediate nodes, such that (ρ_{n_1}, ρ_{n_1}) and (ρ_{n_2}, ρ_{n_2}) are linear independent and the inverse of the matrix $\begin{pmatrix} \rho_{i_1} & \rho_{i_2} \\ q_{i_1} & q_{i_2} \end{pmatrix}$ exits.

Due to the special multicast graph, all the receivers can always recover the source vector $(X_1(s), X_2(s))$. The elements in the transformation vectors belong to a prime field, the transmitted symbol on each link and data symbols also in a prime field.

Table V and Table VI give the pseudo codes of the two phases.

TABLE V
ASSIGNCODES PHASE

- The source s
  - · Multicast message <start-AssignCodes> in the rudimentary tree
  - · Set $Time$=(largest RTT)·2
  - · Set timer $t = 0$, initialize $L = 0$
  - · **While** $t < Time$ **do**
    - ‡ Upon receiving <request-code, address of v_i>:
      - ◇ $L = L + 1$
      - ◇ NodeAddr[L]= address of v_i
  - · ϕ = number of children of s
  - · Obtain $gen(L+\phi) = \{(\rho_1, q_1)^T, (\rho_2, q_2)^T, \cdots, (\rho_{L+\phi}, q_{L+\phi})^T\}$
  - · **For** $i = 1$ to L
    - ‡ send <code, $(\rho_i, q_i)^T$> to node at address NodeAddr[i]

- Upon receiving <start-AssignCodes>, each node $v_i \in V_I$:
  - · **If** v_i has 1 incoming links **then** do nothing
  - · **Else if** v_i has 2 incoming links **then**
    send message <request-code, address of v_i> to s

- Upon receiving <new-code, (ρ, q)>, each node $v_i \in V_I$:
  - · v_i sets its $\underline{w}_i$: $\rho_{v_i} = \rho$ and $q_{v_i} = q$

TABLE VI
DISSEMINATECODES PHASE

- The source s (has ϕ children and has $gen(L+\phi)$ from last phase):
  - · **For** $i = 1$ to ϕ
    - ‡ Send <code, $gen(L+\phi)$> to its ith child

- Each node $v_i \in V_I$ with in-degree 1:
  - · Upon receiving <code, $(\rho, q)^T$> from its parent:
    - ‡ Set its w_i: $\rho_{v_i} = p$ and $q_{v_i} = q$
    - ‡ Send <code, $(\rho_{v_i}, q_{v_i})^T$> out on all its outgoing links
    - ‡ Set linear transformation for all outgoing links to the identity

- Each node $v_i \in V_I$ with in-degree 2:
  - · Send <code, $(\rho_{v_i}, q_{v_i})^T$> out on all its outgoing links
  - · Upon receiving <code, $(\rho_1^{I_i}, q_1^{I_i})$>, <code, $(\rho_2^{I_i}, q_2^{I_i})$>, respectively, from its two upstream nodes:

$$(\beta_1^{v_i}, \beta_2^{v_i})^T = \begin{pmatrix} \rho_1^{I_i} & \rho_2^{I_i} \\ q_1^{I_i} & q_2^{I_i} \end{pmatrix}^{-1} \begin{pmatrix} \rho_{v_i} \\ q_{v_i} \end{pmatrix}$$

4.3 Linear Code Design over GF(2^m)

We propose an algorithm that generates transformation vectors, which elements are in the extension field of order 2, that is finite field GF(2^m)). And the algorithm that generates 2-dimensional pairwise linear independent vectors is quite simple.

In the real world, the data is represented in binary. For a signal processing standpoint, the operations over a prime field need a table mapping the prime elements into a corresponding elements over GF(2^m) and execute the corresponding operation over GF(2^m). After finish the operation, it has to make a re-mapping from GF(2^m) to the prime field. This slowing down the whole operation and eventually decreases the end-to-end throughput. Another drawback of network code construction over a prime field is that the effective end-to-end throughput will decrease by a factor $\frac{1}{m+1}$, where m correspond to GF(2^m) and the filed size of GF(2^m) larger than the prime filed.

The algorithm proposed by Zhu *et al.*, though simple but it can not find all the pairwise linear independent vectors in a prime field. Our proposal can find all the vectors for any GF(2^m). How many pairwise independent vectors are in a finite filed GF(2^m)? The following theorem answers that question.

Theorem 7 *There are* $(2^m + 1)$ *pairwise linear independent 2-d vectors over GF*(2^m).

Proof : Assume α be the primitive element in GF(2^m). Totally, there are $2^{2m} - 1$ nonzero 2-dimension vectors over GF(2^m) and can be divided into $2^m + 1$ disjoint sets with each set has $2^m - 1$ elements. We divide them into

$$S_j = \{\alpha^i(1, \alpha^j) : i = 1 \sim (2^m - 1)\}, \text{ for all } j = 1 \sim (2^m - 1); \quad (4.12a)$$

$$S_{2^m} = \{\alpha^i(1, 0) : i = 1 \sim (2^m - 1)\}; \quad (4.12b)$$

$$S_{2^m+1} = \{\alpha^i(0, 1) : i = 1 \sim (2^m - 1)\}. \quad (4.12c)$$

For each S_k, $k \in \{1, 2, \cdots, 2^m + 1\}$, the elements are all distinct and the size of the set is $|S_k| = (2^m - 1)$. It is easy to see that the sets indeed disjoint.

Vectors in the same set are linear dependent. Vectors come from different sets are pairwise linear independent and maximally there are $2^m + 1$ such vectors. This prove the maximal number of $(2^m + 1)$ 2-dimensional pairwise linear independent vectors. ■

We can randomly choose (2^m+1) vectors, each of them comes from different sets, and these vectors are pairwise linear independent. If there are L intermediate nodes requesting for transformation vector and $Out(s) = \phi$, then source node s first finds a suitable field, such that $(L + \phi)$ wouldn't exceed the maximum offering number $(2^m + 1)$ over GF(2^m). Then find a corresponding primitive element α. We use a function $Gen_2(L+\phi)$ to generate such vectors.

TABLE VII

VECTOR SEQUENCE GENERATION ALGORITHM OVER GF(2^m)

- **Algorithm** $Gen_2(\delta)$
  - · Let $m = \lceil log_2\delta \rceil$
  - · Find a primitive element α in GF(2^m)
  - · vectors[1] $= (0, 1)^T$ and vectors[2] $= (1, 0)^T$
  - · **If** $\delta > 2$ **then**
    - ◇ **for** $j = 1$ to $(\delta - 2)$

      vectors$[j + 2] = (1, \alpha^j)^T$

Table VII shows a simple algorithm for generating δ transformation vectors.

Chapter 5

Conclusion

Based on the max-flow min-cut theorem, Ahlswede *et al.* prove that the maximum achievable rate a network using coding is upper bounded by the *max-flow bound*. They also establish the existence of encoding function to achieve that bound. Li *et al.* then show that linear network coding is sufficient over a infinite field or a large enough finite field. From two different perspectives, linear network code construction algorithms for multicast applications are proposed and the sufficient conditions of the linear code over a finite field GF(2^m) whose dimension m depends only on the total number of receivers. We give detailed description of the algorithm proposed by Koetter *et al.* because their algorithm can be further extended to a more general multicast problem and their approach characterizes the coding problem from an algebraic viewpoint.

Although, theory has promised the existence of capacity-achieving codes for single source multicast and some linear coding algorithms had been proposed, it requires an infrastructure that contains widely deployed codec and supports new protocols to handle related issues like reliability, flow control, congestion control...etc. In order to avoid dealing with these difficulties, Zhu *et al.* use the network coding technique in overlay networks to accomplish the multicast abstraction. Their simulation results show a two-fold throughput improvement over the conventional approach using an application-layer multicast Narada protocol [12]. The overhead paid is the dedicated high-degree relay nodes, which are end hosts or proxy servers connected to high-bandwidth physical links.

This thesis propose a simple algorithm to construct linear network codes over two-redundant multicast graphs. We believe that the effective end-to-end throughput of our proposal is larger than that of Zhu's algorithm.

Network coding as a relative new area of both academic and industrial research interest naturally has many an unsolved and unexplored problems awaiting for the untamed minds. We mention just a few in the following paragraphs as the concluding remarks of our preliminary investigation effort.

It remains to be shown if one can extend our proposal to three- or higher (K-) redundant multicast graphs and find the corresponding higher dimensional independent vectors accordingly. Extension to the multi-source multicast also deserve much effort. For more realistic concern, one should consider nonuniform and non-delay-free cases and investigate related design issues like scheduling, minimum delay/memory-saving algorithm. Since at an integer multicast rate r, it is sufficient to choose r disjoint paths from source s to each sink t_n. The problem thus becomes one of choosing such paths so that minimum delays at each intermediate nodes and the sink nodes are obtained.

As mentioned before, implementation of network coding necessitates a widely deployment of powerful and specific routers that are capable of high-speed routing and (encoding) processing. One would like to know the minimum number of such routers required in a given network such that the full advantage of network coding can still be attained.

Another interesting issue is the application of network coding to wireless ad hoc networks with opportunistic multiple access. Considering such a scenario, one is tempted to ask if what is the capacity of such networks and what is the optimal combination of channel and network codes.

Appendix A

Upper Bound of Multicast Rate with Network Coding

Given a network G with rate constraint $\mathbf{C}$. Assume that for any $\varepsilon > 0$, there exists for sufficiently large m an $(m, (\eta_{ij} : (v_i, v_j) \in E), \tau)$ $\alpha - code$ on G such that

$$m^{-1} log_2 \eta_{ij} \leq C_{ij} + \varepsilon \tag{A.1}$$

for all $(v_i, v_j) \in E$.

The following derive the upper bound of an information source rate τ at a fixed ε and a sufficiently large m. Consider any $t_n \in T$ and any cut $(P_n, \overline{P_n})$ between node s and node t_n, let

$$Z_j(x) = (\widetilde{Y}_k(x) : k \in \bigcup_{v_i \in V} T_{ij}) \tag{A.2}$$

where $x \in \chi$ is a message and $\widetilde{Y}_k$ is the function induced inductively by Y'_k, $1 \leq k' \leq k$, such that $\widetilde{Y}_k(x)$ denotes the value of Y_k as a function of x. $Z_j(x)$ is all the information received by node v_j during the whole coding session when message is x. Since $\widetilde{Y}_k(x)$ is a function of the information previously received by $u_1(k)$, we see inductively that $Z_{t_n}(x)$ is a function of $\widetilde{Y}_k(x), \forall k \in \cup_{(v_i,v_j)\in(P_n,\overline{P_n})} T_{ij}$. Since every message in $\chi = \{1, 2, \cdots, \lceil 2^{m\tau} \rceil\}$ can be determined at sink node t_n, we have

$$\begin{aligned} H(X) &\leq H(X, Z_{t_n}(X)) \\ &= H(Z_{t_n}(X)) \end{aligned}$$

$$
\begin{aligned}
&\leq H(\widetilde{Y}_k(x) : k \in \cup_{(v_i,v_j)\in(P_n,\overline{P_n})} T_{ij}) \\
&\leq \sum_{(v_i,v_j)\in(P_n,\overline{P_n})} \sum_{k\in T_{ij}} H(\widetilde{Y}_k(X)) \\
&\leq \sum_{(v_i,v_j)\in(P_n,\overline{P_n})} \sum_{k\in T_{ij}} log_2|A_k| \\
&= \sum_{(v_i,v_j)\in(P_n,\overline{P_n})} log_2(\prod_{k\in T_{ij}} |A_k|) \\
&= \sum_{(v_i,v_j)\in(P_n,\overline{P_n})} log_2\eta_{ij} \qquad \text{(A.3)}
\end{aligned}
$$

Since sink node t_n can recover all $x \in \chi$, it is intuitive that $Z_{t_n}(X)$ contains the complete information about X, then (A.3) follows from (A.3). By the *independence bound for entropy theorem*[4], (A.3) can be derived from (A.3). Since $\frac{1}{m}\sum_{k\in T_{ij}} H(\widetilde{Y}_k(X))$ represents the information rate transmitted on edge (v_i, v_j) and $\frac{1}{m}\sum_{k\in T_{ij}} log_2|A_k|$ represents the edge capacity $(C_{ij} + \varepsilon)$, then (A.3) can be derived from (A.3). From (2.18a),

$$\sum_{k\in T_{ij}} log_2|A_k| = log_2\eta_{ij}, \qquad \text{(A.4)}$$

(A.3) can be derived. It is easy to derive that

$$m^{-1}H(X) \leq \sum_{(v_i,v_j)\in(P_n,\overline{P_n})} m^{-1}log_2\eta_{ij} \qquad \text{(A.5)}$$

and this will be used in the derivation of the upper bound of τ. Thus

$$
\begin{aligned}
\tau &\leq m^{-1}log_2\lceil 2^{m\tau}\rceil \\
&= m^{-1}log_2|\chi| \\
&= m^{-1}H(X) \\
&\leq \sum_{(v_i,v_j)\in(P_n,\overline{P_n})} m^{-1}log_2\eta_{ij} \\
&\leq \sum_{(v_i,v_j)\in(P_n,\overline{P_n})} (C_{ij}+\varepsilon) \\
&\leq (\sum_{(v_i,v_j)\in(P_n,\overline{P_n})} C_{ij}) + |E|\varepsilon \\
&= C(P_n,\overline{P_n}) + |E|\varepsilon \\
&\leq mincut(s,t_n) + |E|\varepsilon \qquad \text{(A.6)}
\end{aligned}
$$

The the message x is selecting from χ according to the uniformly distribution, then the entropy $H(X)$ is equal to $log_2|\chi|$. Since for any $t_n \in T$

$$\tau \leq mincut(s, t_n) + |E|\varepsilon, \tag{A.7}$$

the upper bound of the information source rate is

$$\tau \leq \min_n maxflow(s, t_n) + |E|\varepsilon, \tag{A.8}$$

For multicast at rate $r - \varepsilon \leq \tau$, we can select the multicast message x from $\chi_M \subseteq \chi$ according to uniform distribution with $m^{-1}log_2|\chi_M| = (r-\varepsilon)$, and each message $x \in \chi_M$ can be recovered by every sink nodes. In the same way, the mulicast rate $(r - \varepsilon)$ is also bounded by

$$r - \varepsilon \leq \min_n maxflow(s, t_n) + |E|\varepsilon. \tag{A.9}$$

Let $\varepsilon \to 0$, we obtain

$$r \leq \min_n maxflow(s, t_n). \tag{A.10}$$

We conclude that the achievable multicast rate is always bounded by the *max-flow bound*.

Appendix B

Achievability of the *Max-flow Bound* with Network Coding

Given a network G with rate constraint $\mathbf{C}$ and the mulitcast at rate $(r-\varepsilon)$, which

$$(r-\varepsilon) \leq \min_{n} maxflow(s,t_n). \tag{B.1}$$

The following try to prove that for any $\varepsilon > 0$, there exists for sufficiently large m an $(m, (\eta_{ij} : (v_i, v_j) \in E), r-\varepsilon)$ $\beta - code$ on G such that

$$m^{-1} log_2 \eta_{ij} \leq C_{ij} + \varepsilon \tag{B.2}$$

for all $(v_i, v_j) \in E$.

This will be done by constructing a Random code. In constructing this code, temporarily replace χ by

$$\chi' = \{1, 2, \cdots, \lceil c2^{m(r-\varepsilon)} \rceil\}, \tag{B.3}$$

where c is a positive integer greater than 1. Thus the domain of Y_{sj} is expanded from χ to χ' for all $(s, v_j) \in E$.

The encoding functions are constructed on as follows. For all v_j, such that $(s, v_j) \in E$, for all $x \in \chi'$,let $Y_{sj}(x)$ be a value selected independently from the set $\{1, 2, \cdots, \eta_{sj}\}$ according to the uniform distribution. For all $(v_i, v_j) \in E$, $(v_i \neq s)$, and for all

$$Z_i(x) \in \prod_{v_{i'}:(v_{i'}, v_i) \in E} \{1, 2, \ldots, \eta_{i'i}\} \tag{B.4}$$

where $Z_i(x)$ be the information received by node v_i and $Y_{ij}(Z_i)$ be a value selected independently from the set $\{1, 2, \ldots, \eta_{ij}\}$ according to the uniform distribution. Let

$$Z_s(x) = x \tag{B.5}$$

and for all $v_j \in V$ and $v_j \neq s$, let

$$Z_j(x) = (\widetilde{Y_{ij}}(x), (v_i, v_j) \in E) \tag{B.6}$$

where $Z_j(x)$ be the information received by node v_j when the message is $x \in \chi'$ and $\widetilde{Y_{ij}}(x)$ denotes the value of Y_{ij} as a function of x. For distinct x and x', both belong to χ', x and x' are *indistinguishable* at sink t_n if and only if $Z_{t_n}(x) = Z_{t_n}(x')$. For all $x \in \chi$, define

$$N(x) = \begin{cases} 1, & \text{if for some } n = \{1, 2, \cdots, N\}, \text{ there exists } x' \in \chi', \\ & x \neq x', \text{ such that } Z_{t_n}(x) = Z_{t_n}(x'); \\ 0, & \text{otherwise.} \end{cases} \tag{B.7}$$

$N(x)$ is equal to 1 if and only if *at least one* of the sink nodes cannot uniquely decode x and x'. Now fix $x \in \chi'$ and $1 \leq n \leq N$. Consider any $x' \in \chi'$ not equal to x and define two sets

$$P_d = \{v_i \in V : Z_i(x) \neq Z_i(x')\} \tag{B.8a}$$

and

$$P_{ind} = \{v_i \in V : Z_i(x) = Z_i(x')\} \tag{B.8b}$$

P_d is the set of nodes at which the two messages x and x' are distinguishable, and obviously $s \in P_d$.

Suppose $Z_{t_n}(x) = Z_{t_n}(x')$. Then $P_d = P_n$ for some $P_n \subset V$ (V has $2^{|V|}$ kind of subsets), which $(P_n, \overline{P_n})$ is a cut between s and t_n. For any $(v_i, v_j) \in E$,

$$Pr\{\widetilde{Y_{ij}}(x) = \widetilde{Y_{ij}}(x') | Z_i(x) \neq Z_i(x')\} = \eta_{ij}^{-1}. \tag{B.9}$$

It is the probability of the encoding function Y_{ij} of node v_i sending the same message onto edge (v_i, v_j) given the source messages are distinct and the received information of

node v_i is also distinct. Therefore,

$$\begin{aligned} Pr\{P_d = P_n\} &= Pr\{P_d = P_n, P_d \supset P_n\} \\ &= Pr\{P_d = P_n | P_d \supset P_n\} Pr\{P_d \supset P_n\} \\ &\leq Pr\{P_d = P_n | P_d \supset P_n\} \\ &= \prod_{(v_i,v_j)\in(P_n,\overline{P_n})} Pr\{\widetilde{Y_{ij}}(x) = \widetilde{Y_{ij}}(x') | Z_i(x) \neq Z_i(x')\} \\ &= \prod_{(v_i,v_j)\in E_{P_n}} \eta_{ij}^{-1} \end{aligned} \tag{B.10}$$

$Pr\{P_d = P_n\}$ is the probability of $P_d \subset P_n$ and $P_d \supset P_n$ for a fixed P_n.

Let ε be any fixed positive real value. For all $(v_i, v_j) \in E$, take η_{ij} such that

$$C_{ij} + \zeta \leq (m^{-1} log_2 \eta_{ij}) \leq C_{ij} + \varepsilon, \tag{B.11}$$

for some $0 < \zeta < \varepsilon$. Then

$$\begin{aligned} Pr\{P_d = P_n\} &\leq \prod_{(v_i,v_j)\in(P_n,\overline{P_n})} \eta_{ij}^{-1} \\ &\leq \prod_{(v_i,v_j)\in(P_n,\overline{P_n})} 2^{-m(C_{ij}+\zeta)} \\ &= 2^{-m(|(P_n,\overline{P_n})|\zeta + \sum_{(v_i,v_j)\in(P_n,\overline{P_n})} C_{ij})} \\ &\leq 2^{-m(\zeta + \sum_{(v_i,v_j)\in(P_n,\overline{P_n})} C_{ij})} \\ &\leq 2^{-m(\zeta + maxflow(s,t_n))} \\ &\leq 2^{-m((r-\varepsilon)+\zeta)} \end{aligned} \tag{B.12}$$

Eqn (B.12) follows because $|(P_n, \overline{P_n})| \geq 1$; Eqn (B.12) follows because for the capacity of any cut between node s and node t_n is bounded by $mincut(s, t_n)$ and

$$mincut(s, t_n) = maxflow(s, t_n). \tag{B.13}$$

Equation (B.12) follows because

$$\begin{aligned} r - \varepsilon &\leq \min_n maxflow(s, t_n) \\ &\leq maxflow(s, t_n) \end{aligned} \tag{B.14}$$

The upper bound of $Pr\{P_d = P_n\}$ does not depend on P_n. Then

$$\begin{aligned} & Pr\{Z_{t_n}(x) = Z_{t_n}(x')\} \\ &= Pr\{P_d = P_n \text{ for some cut } (P_n, \overline{P_n}) \text{ between } s \text{ and } t_n\} \\ &\leq \sum_{\text{for some } cuts} Pr\{P_d = P_n\} \\ &\leq 2^{|V|} 2^{-m((r-\varepsilon)+\zeta)} \end{aligned} \tag{B.15}$$

Eqn (B.15) follows by the union bound. Eqn (B.15) follows because there are total $2^{|V|}$ subsets of V. Further, fix x

$$\begin{aligned} & Pr\{Z_{t_n}(x) = Z_{t_n}(x') \text{ for some } x' \in \chi', x \neq x'\} \\ &\leq \sum_{\text{for some } x' \in \chi'} Pr\{Z_{t_n}(x) = Z_{t_n}(x')\} \\ &\leq (|\chi'| - 1) 2^{|V|} 2^{-m((r-\varepsilon)+\zeta)} \\ &\leq c 2^{m(r-\varepsilon)} 2^{|V|} 2^{-m((r-\varepsilon)+\zeta)} \\ &= c 2^{|V|} 2^{-m\zeta} \end{aligned} \tag{B.16}$$

Therefore

$$\begin{aligned} E\{N(x)\} &= Pr\{N(x) = 1\} \\ &= Pr\{\bigcup_{n=1}^{N} Z_{t_n}(x) = Z_{t_n}(x') \text{ for some } x' \in \chi', x \neq x'\} \\ &< N c 2^{|V|} 2^{-m\zeta} \\ &= \delta(m, \zeta) \end{aligned} \tag{B.17}$$

where

$$\delta(m, \zeta) = N c 2^{|V|} 2^{-m\zeta}. \tag{B.18}$$

For a given message x, if it can be uniquely decoded at all sink nodes, then $N(x) = 0$. Hence, the total number of messages which can be uniquely determined at all the sink nodes is equal to

$$\sum_{x \in \chi'} (1 - N(x)). \tag{B.19}$$

Since all the encoding functions are random, the Eqn (B.19) is a random number. By taking the expectation for the random code, then

$$\begin{aligned} E\{\sum_{x\in\chi'}(1-N(x))\} &= \sum_{x\in\chi'}(1-E(N(x))) \\ &> \sum_{x\in\chi'}(1-\delta(m,\zeta)) \\ &\geq (1-\delta(m,\zeta))c2^{m(r-\varepsilon)} \end{aligned} \tag{B.20}$$

where (B.20) follows from (B.17).

Therefore, there exits a deterministic code, for which Y_{ij} is a deterministic function for all $(v_i, v_j) \in E$, and the number of messages that can be uniquely decoded at all sink nodes is at least

$$(1-\delta(m,\zeta))c2^{m(r-\varepsilon)}, \tag{B.21}$$

which is greater than $2^{m(r-\varepsilon)}$ for m sufficiently large. Since $\delta(m,\zeta) \to 0$ as $m \to \infty$. Let χ be any set of $\lceil 2^{m(r-\varepsilon)} \rceil$ messages from χ' and for all $x \in \chi$ and for all $t_n \in T$

$$g_n(Z_{t_n}(x)) = x, \tag{B.22}$$

where

$$Z_{t_n}(x) \in \prod_{v_{i'}:(v_{i'},t_n)\in E} \{1, 2, \cdots, \eta_{i't_n}\}. \tag{B.23}$$

Eqn $(B.22)$ means that for every $x \in \chi$, all sink nodes can recover this message. Thus the multicast rate

$$(r-\varepsilon) \leq \min_n maxflow(s, t_n), \tag{B.24}$$

it is always achievable.

Let $\varepsilon \to 0$, there also exists an $(m, (\eta_{ij} : (v_i, v_j) \in E), r)$ $\beta-code$ on G for sufficiently large m. Such that for multicast rate

$$r \leq \min_n maxflow(s, t_n), \tag{B.25}$$

it is always achievable.

We conclude that the multicast rate at the *max-flow bound* is always achievable.

Bibliography

[1] L. R. Ford and D. R. Fulkerson, *Flows in Networks*, Princeton Univ. Press, 1962.

[2] A. Tucker, *Applied Combinatorics*, 3rd ed. New York: Wiley, 1995.

[3] J. Edmonds and R. M. Karp, "Theoretical improvements in algorithmic efficiency for network flow problems", *Journal of the ACM*, 19(2):248-264, Apr. 1972.

[4] R. W. Yeung, *A First Course in Information Theory*, New York: Kluwer Academic, 2002.

[5] R. Ahlswede, N. Cai, S.-Y. R. Li and R. W. Yeung, "Network information flow", *IEEE Trans. Inform. Theory*, vol. 46, pp. 1204-1216, Jul. 2000.

[6] S.-Y. R. Li, R. W. Yeung and N. Cai, "Linear network coding", *IEEE Trans. Inform. Theory*, vol. 49, pp. 371-381, Feb. 2003.

[7] S. Jaggi, P. A. Chou, and K. Jain, "Low complexity algebraic multicast network codes", *IEEE Int. Symp. Inform. Theory*, Yokohama, Japan, Jun. 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", *IEEE Trans. Inform. Theory*, submitted July 2003.

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

[10] T. Ho, D, Karger, M. Médard, and R. Koetter, "Network coding from a network flow perspective", *IEEE Int. Symp. Inform. Theory*, Yokohama, Japan, Jun. 2003.

[11] Y. Zhu, B. Li, and J. Guo, "Multicast with network coding in application-layer overlay networks", *IEEE J. Select. Areas Commun.*, vol. 22, pp.107 - 120, Jan. 2004.

[12] Y. Chu, S. G. Rao, S. Seshan, and H. Zhang, "A case for end system multicast", *IEEE J. Select. Areas Commun.*, vol. 20, pp. 1456-1471, Oct. 2002.

[13] Z. Wang and J. Crowcroft, "Quality-of-service routing for supporting multimedia applications", *IEEE J. Select. Areas Commun.*, vol. 14, pp.1228-1234, Sep. 1996.

[14] J. F. Kurose and K. W. Ross, "Computer Networking - A Top-Down Approach Featuring the Internet", 2nd ed. Addison Wesley, Reading, MA, 2003.

作者簡介

李欣勳，台灣省嘉義縣人，1980 年出生於布袋過溝半農半魚村，從此踏上不歸路---當人!
1992 年畢業於嘉義縣立過溝國小。小一到小四的時候，曾經被家人誤認成智障（當時的成績實在是不好意思拿出來），害我老爸老媽拼了老命也要賺錢把我送到補習班惡補一番（不過現在的情況似乎也沒有太大的改善，怪哉！）；約小五時，第一次跟班上一位女同學陷入所謂的感情模糊期，嗯，感覺挺好的。可是好景不常，畢業即失戀（不過現在還是跟這位女性，是個熟女囉，保持聯繫）。

1995 年畢業於嘉義縣立過溝國中；國一、國二沒啥好談的（阿不就考試、考試、考試這回事），不過國三可就刺激了，跟初戀情人重新搭上線（又再次陷入純純的愛裡面）加上幾乎是本校第一屆榮獲『嘉雲南數學競試---入選獎』（當時聽說可以直接保送本地的第一學府---嘉義高中），此時陷入了所謂的愛情事業兩得意；不過，古人說的好：『是福不是禍，是禍躲不過』，即將畢業之時，又再一次地被熟女拋棄，雪上加霜的是，好死不死的，本屆保送入學的優待---取消。嗯，只好乖乖的去參加聯考了。

1998 年畢業於省立嘉義高中。本高中可是出了名的『自由學府』，自由到什麼程度呢？自由到本人我，進去不到一學期就被學長拖到司令台海扁（我有這麼欠揍嗎？不解！），而此戰所留下的一攤鼻血，在隔天早上還被渲染成司令台殺人事件呢！不過流鼻血倒是小 case，比較大的 case 是---鼻梁被打斷了（淚　　）；至此之後，隱姓埋名，好好做個不快樂的高中生，考大學為己任！！

2002 年畢業於私立輔仁大學電子工程學系，告別了我最愛的台北。初嚐大學生活，大一可是可以用『糜爛』二字來形容，每天不是上 BBS，不然就是跟同學四處鬼混，再不然就四處找人聯誼（ㄟ…你們沒有嗎？？ 可能是本大學比較像『正常』的大學吧，別打我！）；而果然，聯誼就聯誼，好歹被我摸到一個女朋友（正式的喔，不是那種兩小無猜型的喔！），從此進入了我人生的第一個高峰期，愛情課業兩得意---而且持續了四年之久，不過從最上的兩個時期，大家應該可以再次的推測---沒錯，畢業之後，失戀 again！而在大四的時候，非常高興的甄試上了交大電信所，從此幻想著平步青雲的人生，直到

2002 年九月前往國立交通大學電信工程系就讀碩士班。有一句話說的好;『幻滅是成長的開始』，這一句話送給全國大學研究所的研究生；不過又有一句話說的更好：『壓力是成長的動力』（誰說的？我說的！），再一次與全國研究生共勉之！！

2004 年獲得碩士學位，暫別學生生涯當兵去。人生起起伏伏，在研究所兩年裡，就好像坐雲霄飛車一樣，雖然心裡面害怕，但又無時無刻的充滿著驚奇，總是在這裡得到些，而又在那裡失掉些。畢業，期望只是求知生涯的一個小小休止符；在未來裡，但願還能夠繼續回來彈奏這美妙動聽的學術樂章！

Graduate Course

1. Digital Communication
2. Detection and Estimation
3. Digital Signal Process
4. Adaptive Signal Process
5. Array Signal Process
6. Random Process
7. Coding Theory
8. Optimization Theory