파운틴 코드의 개요 (Overview of Fountain codes)

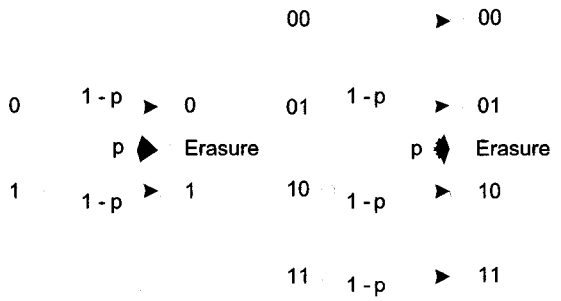
박도형 | 김민규 | 정세영
한국과학기술원

요 약

최근 들어 다중 사용자 환경에서 효율적인 정보 전송에 관한 연구가 진행되면서 새로운 방식의 채널 부호화 기술들이 탄생하였다. 파운틴 코드는 Erasure 채널로 표현되는 네트워크 상에서 다수에게 전송하는 브로드캐스팅 및 멀티캐스팅에 적합하도록 개발된 채널 부호화 기술로서, 정보를 무한히 많은 양으로 부호화하여 전송하는 것을 특징으로 한다. 이와 같은 방식의 채널 부호화 기술은 최근 몇 년간 이론적인 측면뿐만 아니라 실제 표준화 측면에서도 활발하게 연구되고 있다. 본 논문에서는 파운틴 코드에 대하여 소개하고 실제 개발된 몇 가지 파운틴 코드를 소개한다. 그리고 파운틴 코드의 향후 응용 가능성에 대해서도 논의한다.

I. 서 론

파운틴 코드는 채널 코딩의 rate를 (기존 정보의 양)/(부호화한 데이터의 양)으로 표현할 때, 부호화한 데이터의 양이 미리 정해지지 않기 때문에 이를 Rateless code라고도 부른다. 파운틴 코드는 송신단 측에서 수신단에 대한 정보가 부족하거나 수신단의 수가 매우 많을 때처럼 양방향 정보 전송이 어려울 때에도 (단방향 전송만으로) 에러 없이 완벽한 수신을 가능하게 한다는 장점을 가지고 있어 컴퓨터 네트워크 내에서의 멀티캐스트 등에서 필요성이 제기되었다.



(그림 1) Erasure 채널의 예

현재까지의 파운틴 코드는 주로 Erasure 채널에서 연구되었고, 그 밖의 채널에서의 적용이 활발하게 진행되고 있다. Erasure 채널은 그림 1에서 보는 것과 마찬가지로 어떤 정보가 송신되는지에 상관없이 일정한 확률로 그 정보가 지워져 무엇이 전송되었는지 알 수 없게 되는 채널이다. 이 채널이 소개된 1955년에는 이론적인 가상 채널로만 인식되어 왔으나, 최근 인터넷의 발전과 더불어 Erasure 채널은 인터넷을 모델링할 수 있는 채널로 주목받았다. 인터넷 상에서 각 패킷들은 제대로 전송되거나 그렇지 못하게 된다. 즉 어떤 패킷이 전송되는지에 상관없이 일정한 확률로 정확히 수신되거나 버려지는(Erasure) 경우만이 적용되므로 인터넷은 Erasure 채널로써 모델링 될 수 있다.

만약 인터넷망을 통해 방대한 양의 프로그램을 수십만 명의 다양한 수신자들에게 공급한다고 가정해보자. 그렇다면 유니캐스트 프로토콜(예: TCP/IP, UDP)을 위와 같은 상황에 적용시켰을 때, 수신 단에서는 그들이 받지 못한 패킷에 대한 재전송을 요청하는 피드백을 보내며, 송신단은 이에 따

른 패킷들을 재전송한다. 하지만 다중 사용자 통신환경에서는 각 수신단마다 수많은 재전송 요청이 생기므로 송신단에 가해지는 피드백의 양은 기하급수적으로 늘어나며, 또한 재전송 요청과 재전송된 패킷 등으로 인해 네트워크에 가해지는 부하는 더욱더 커지게 된다.

따라서 네트워크의 과부하를 제공하는 재전송요청을 없애고, 그리고 수신단에게 비동기 수신을 제공하기 위해서는 다음과 같은 통신방식을 생각할 수 있다. 우선 송신단에서는 전송할 파일을 이용하여 끊임없이 부호화된 패킷을 만들어서 전송한다. 그렇다면 각각의 수신단은 피드백이 필요없이 복호화가 가능할 정도의 패킷만을 수신하여 복호화한다. 파운틴 코드는 이러한 방식을 기초로 한다.

이와 같은 전송을 위한 채널 부호화 기법을 파운틴 코드라고 부르는 이유는 전송 방식이 마치 분수에서 물을 얻는 것과 비슷하기 때문이다. 분수에서는 거의 무한한 양의 물을 항상 뿜어내고 있으며 그 근처에 어느 곳에나 컵을 놓아도 원하는 양의 물을 채울 수 있다. 파운틴 코드에서도 역시 송신단에서는 일정한 양의 정보를 거의 무한히 부호화하여 전송하고 수신단은 그 중 어떤 것을 받는지에 상관없이 충분한 양을 수신하면 그 정보를 얻을 수 있기 때문에 이를 분수에 비유하여 명명되었다.

본 논문에서는 파운틴 코드의 부/복호화 방법을 설명하고 대표적인 파운틴 코드인 LT code와 Raptor code를 소개 및 그 성능에 대해서 다뤄본다. 파운틴 코드에 대해 이해하기에 앞서 필요한 개념에 대하여 2장에서 다룰 것이고, 3장에서는 파운틴 코드의 가장 단순한 예인 랜덤 파운틴 코드를 알아본다. 4, 5장에서는 LT code 및 Raptor code를 소개하고, 6장에서는 실제로 구현하였을 때의 결과를 살펴보며, 마지막으로 7장에서 현재 진행중인 파운틴 코드의 연구 흐름에 대해 알아본다.

II. 배경 지식

파운틴 코드에 들어가기에 앞서, 먼저 파운틴 코드를 이해하는 데 필요한 두 가지 개념을 먼저 설명하기로 한다.

2.1. 블록 부호화 기법 (Block code)

블록 부호화 기법은 채널 부호화 기법 중에서 가장 일반적으로 부호화 기법이다. 이 방식은 생성 행렬(Generator matrix)과 패리티체크 행렬(Parity-check matrix)로써 표현된다. 송신단에서 보내고자 하는 정보를 k 비트 단위로 끊어서 생성 행렬을 곱하여 얻어지는 n 비트의 정보를 코드워드라 한다. 또한 주어진 생성 행렬에 의해 생성된 코드워드만이 그것을 곱했을 때 0이 되는 패리티체크 행렬를 정의할 수 있다(그림 2 참조). 여기서 생성 행렬의 열벡터는 각각 코드워드 내의 대응하는 비트를 생성하며, 각각의 열벡터 내에는 k 비트의 정보 중에서 어떤 비트를 선택, XOR연산하여 얻을 것인가를 의미한다. 즉, 첫 번째 열벡터 내의 1, 2, 5번째 원소가 1일 경우 1, 2, 5번째 비트 정보를 취합하여 XOR연산함으로써 코드워드 내의 첫 번째 비트값을 얻는다. 또한 같은 방식으로 패리티 체크 행렬에서 각각의 행벡터는 n 비트의 벡터가 코드워드이려면 어떤 비트를 선택하여 XOR연산 할 때 0이어야 하는지 정의한다.

이러한 블록 부호화 기법의 복호화 방식으로 역행렬 연산을 이용하는 ML 복호방식과 그래프 상에서 반복하여 메시지를 주고 받는 Message-passing 복호방식이 있다. ML 복호방식은 높은 연산량을 요구하기 때문에 일반적으로 Message-passing 복호방식을 사용한다.

$$\underbrace{(1\ 1\ 0\ 0\ 1)}_{information\ \mathbf{u}} \times \underbrace{\begin{pmatrix} 0&0&1&0&0&0&0\\ 1&1&0&1&0&0&0\\ 0&1&0&0&1&0&0\\ 0&1&0&0&0&1&0\\ 1&0&0&0&0&0&1 \end{pmatrix}}_{Generator matrix\ \mathbf{G}} = \underbrace{(1\ 1\ 1\ 1\ 0\ 0\ 1)}_{codeword\ \mathbf{x}}$$

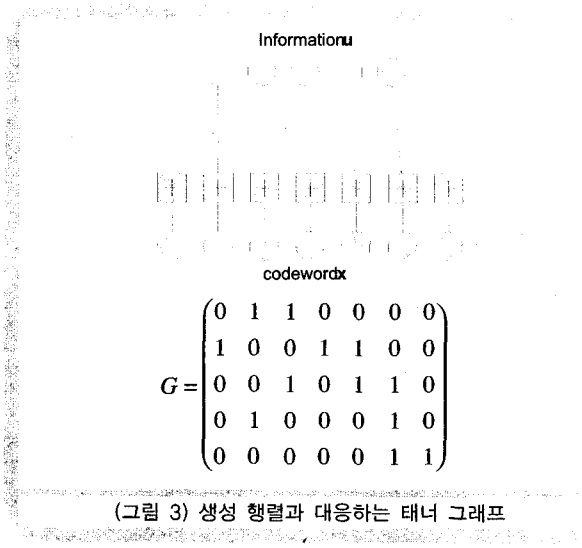
$$\underbrace{\begin{pmatrix} 1&0&1&1&0&0&1\\ 0&1&0&1&1&1&0 \end{pmatrix}}_{Parity-check\ matrix\ \mathbf{H}} \times \underbrace{(1\ 1\ 1\ 1\ 0\ 0\ 1)^T}_{codeword\ \mathbf{x}} = \underbrace{\begin{pmatrix} 0\\ 0 \end{pmatrix}}_{0}$$

(그림 2) 블록 부호화 기법

2.2. 태너 그래프 (Tanner graph)

태너 그래프는 블록 부호화 기법의 구조를 그림을 통해 표현할 수 있다. 태너 그래프는 변수 노드, 체크 노드, 에지 세 부분으로 구성된다. 변수 노드는 각각의 비트를 의미하며, 체크 노드는 에지로 연결된 변수 노드가 의미하는 비트의

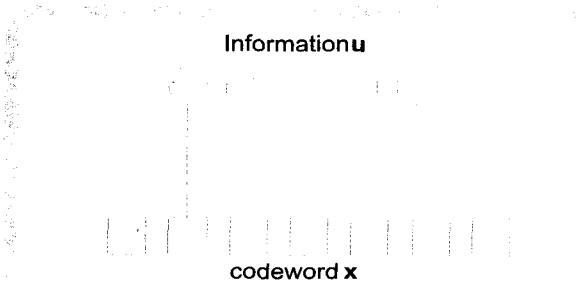
XOR 합이 0이 되도록 함을 의미한다.



(그림 3) 생성 행렬과 대응하는 태너 그래프

(그림 3)을 보면, 전송할 정보를 표현하는 변수 노드와 코드워드를 표현하는 변수 노드가 존재하며, 각각의 코드워드 비트는 서로 다른 체크 노드 하나와 연결되어 있으며, 코드워드 비트를 얻기 위해 어떤 정보 비트를 XOR연산할 것인지를 체크 노드를 통해서 연결하고 있다. 생성 행렬 G에 대해서 $G_{ij}=1$에 해당하는 i번째 정보 비트는 j번째 코드워드 비트와 연결된 체크 노드와 연결되어 있다. 의미상으로는 코드워드 비트는 1로 연결된 정보 비트의 XOR 합으로 나타나지만, 다르게 보면 그 합과 코드워드 비트를 XOR 합하였을 때 0이 되도록 한다는 의미로 해석할 수 있다.

일반적으로 태너 그래프를 간단하게 그리기 위하여 (그림 4)와 같이 체크 노드 자체를 코드워드 비트와 묶어서 간략화 할 수 있다.



(그림 4) 간략화된 태너 그래프

III. 랜덤 파운틴 코드

파운틴 코드의 부호화 기법은 주어진 k개의 소스 심볼 $(x_1,x_2,\cdots,x_k)$로 무한히 많은 양의 인코딩 심볼 $(y_1,y_2,\cdots)$을 생성시킨다. 여기에서 소스 심볼과 인코딩 심볼은 1비트가 될 수도 있지만, 여러 비트의 패킷을 나타내는 벡터가 될 수도 있다. 수신단에서는 부호화 과정을 모두 알고 있다고 가정한다. 또한 전송받은 인코딩 심볼들이 각각 어떠한 방식으로 생성되었는지 알고 있다고 가정한다. 이와 같은 정보는 각각의 인코딩 심볼들이 그 생성 정보를 포함하는 등의 방식으로 구현할 수 있다.

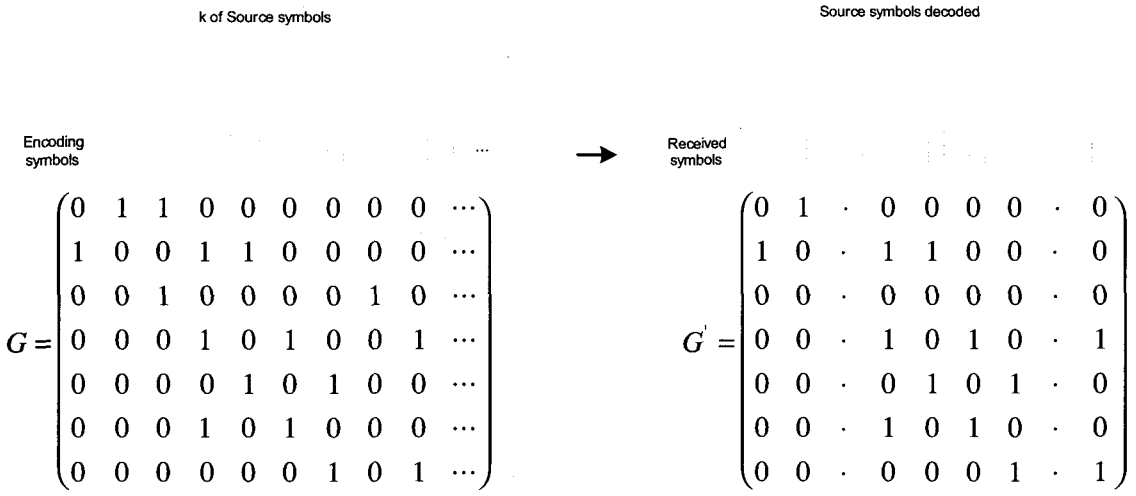
랜덤 파운틴 코드는 파운틴 코드의 가장 단순한 예로서, 파운틴 코드에 대해 이해하기 위한 가장 첫 발판이기도 하다[11].

3.1. 부호화

랜덤 파운틴 코드는 각각의 인코딩 심볼들을 생성하기 위하여 랜덤하게 소스 심볼을 선택하여 XOR합으로 만들어진다. 각각의 소스 심볼이 선택될 확률은 1/2의 확률을 갖는다.

이러한 방식의 부호화 기법은 랜덤 블록 부호화 방식과 비슷하며, 이 과정을 간단히 생성 행렬로 표현하면 다음과 같다(그림5 참조). 랜덤 파운틴 코드는 무한히 많은 인코딩 심볼을 생성하므로 생성 행렬은 k행과 무한한 수의 열을 갖는 행렬로 나타나는 것을 볼 수 있다. 각각의 열과 대응하는 인코딩 심볼은 XOR합으로 사용할 소스 심볼의 일부를 정하기 위해서 0 또는 1이 무작위로 선택된 열벡터를 사용한다. 이러한 확률 분포 는 총 2^k가지 가능한 열벡터가 동일한 확률을 가지고 선택되도록 한다. 즉, 하나의 인코딩 심볼을 생성하기 위해 소스 심볼의 일부를 선택하는 모든 경우의 확률이 동일하다는 것을 의미한다.

따라서 한 인코딩 심볼이 선택하는 평균 소스 심볼 수는 k/2이고, 이는 태너 그래프 상에서 평균 에지 수로 간주할 수 있다. 그리고 인코딩 심볼을 생성하기 위하여 필요한 XOR연산량은 k/2-1이 되므로, 한 인코딩 심볼을 생성하여 전송하는 데 필요한 시간복잡도는 $O(k)$의 선형 시간으로 얻어진다.

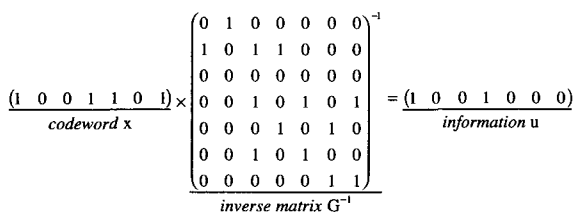


(그림 5) 파운틴 코드의 생성 및 전송

3.2. 복호화

랜덤 파운틴 코드의 복호화 과정에서는 앞에서 언급한 파운틴 코드의 기본적인 가정과 같이 인코딩 심볼이 k개 중 몇 번째 소스 심볼들을 사용하여 XOR합한 결과인지 모두 알고 있다고 가정한다. 따라서 수신단은 채널에 의해 일부가 지워지고 난 나머지 인코딩 심볼을 받아서 이들과 대응하는 생성 행렬의 열벡터만으로 구성된 새로운 생성 행렬 G'를 얻을 수 있다 (그림 5 참조). 수신단은 지워지지 않고 확실히 수신한 인코딩 심볼과 대응하는 생성 행렬 G'를 알고 있으므로 이를 이용하여 ML 복호화한다.

ML 복호화는 역행렬을 이용한다. ML 복호화는 가장 가능성 있는 코드워드를 찾는 방법으로, 역행렬을 사용함으로써 가능성 있는 유일한 코드워드를 찾는 것이라 생각할 수 있다. (그림 6)은 (그림 5)를 이용한 ML 복호화 과정을 표현하



(그림 6) ML 복호화 방식

고 있다.

랜덤 파운틴 코드는 복호화를 성공하기 위해서 k개보다 약간 많은 양의 인코딩 심볼을 필요로 한다. 수신단이 k개만의 인코딩 심볼을 받아 역행렬을 가질 확률(k개의 인코딩 심볼만으로 복호화 해낼 확률)은 k x k 행렬이 선형독립일 확률과 같으므로 다음과 같다.

$$\left(1-2^{-k}\right)\times\left(1-2^{-(k-1)}\right)\times\mathrm{L}\times\left(1-2^{-2}\right)\times\left(1-2^{-1}\right)$$

k가 매우 클 경우 k개의 인코딩 심볼만으로 복호화 해낼 확률은 0.289로 수렴한다. 따라서 랜덤 파운틴 코드는 소스 심볼의 개수만큼 인코딩 심볼을 가지고 복호화 할 경우 성능은 매우 좋지 않다. k가 클 때 인코딩 심볼의 개수를 k보다 약간 크게 하면 복호화를 성공할 확률을 크게 증가시킬 수 있다. 그러나 연산량 측면에서 ML 복호화 기법은 가우시안 소거를 계산하기 위하여 $O(k^3)$의 복잡도를 갖는다는 단점이 있다.

3.3 파운틴 코드의 성능 척도

일반적인 채널 부호화 기법은 제한된 양의 코드워드를 이용하여 전송하였을 때 최소한의 에러율을 얻을 수 있도록

하거나, 주어진 범위의 에러율을 허용하는 범위 내에서 최소한의 코드워드를 전송하는 것을 설계의 주 목적으로 본다.

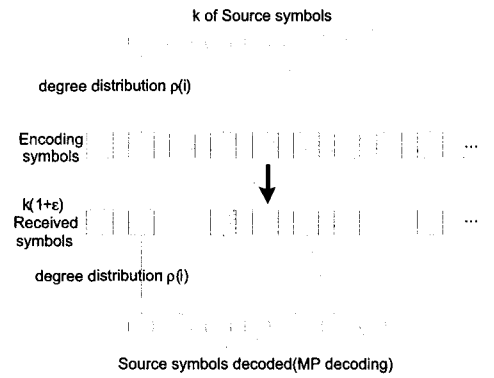
그러나 파운틴 코드는 무한한 양의 인코딩 심볼을 전송하기 때문에 수신단 측에서 항상 전송하고자 하는 정보를 완벽히 복호화할 수 있다고 가정한다. 따라서 다른 관점의 성능 척도를 사용하는 데, 두 가지가 있다. 하나는 얼마나 적은 양의 지워지지 않은 인코딩 심볼을 받아 기존의 소스 심볼을 완전히 복원할 수 있는지를 본다. 다른 하나는 얼마나 적은 양의 연산으로 소스 심볼을 복원할 수 있는지를 본다.

파운틴 코드가 실제 통신환경에서 사용되기 위해서는 수신단에서는 빠르게 복호화하기 위하여 최소한의 인코딩 심볼만으로도 복호화가 가능한 코드를 설계해야 한다. 이 때 소스 심볼의 수보다 적은 양의 인코딩 심볼로는 복호화가 불가능하기 때문에[1] 이 척도는 복호화 가능한 인코딩 심볼 수가 소스 심볼 수와 거의 같아지도록 하는 것을 목표로 한다. k개의 소스 심볼을 이용하여 파운틴 코드로 부호화 했을 때, 복호화에 필요한 코딩 심볼 수를 $k(1+\varepsilon)$로 표현할 수 있으며, 이 때 ε을 오버헤드(overhead)라 한다. 연산량 측면에서는 부/복호화에 필요한 XOR연산량을 줄이는 것이 목표이다.

이와 같이 파운틴 코드의 설계는 오버헤드가 0에 가깝고 연산량을 최대한 줄이기 위하여 어떻게 부/복호화할 것인가를 설계하는 것이 핵심이라 할 수 있다.

IV. LT code (Luby-Transform code)

LT code는 2002년 Michael Luby에 의하여 개발된 최초의 파운틴 코드이다[2]. 이 코드는 두 가지 파운틴 코드의 성능 척도를 이론적으로 가장 최적화한다. 매우 큰 양의 소스 심볼을 전송할 때 확률적으로 오버헤드가 0에 가깝도록 하며, 부/복호화에 필요한 계산을 최소화한다. LT code는 랜덤 파운틴 코드와 유사한 구조를 가지나 인코딩 심볼을 생성하는 데 있어서 소스 심볼을 선택하는 확률 분포를 잘 설계함으로써 성능을 높인 것이라 볼 수 있다.



(그림 7) LT code의 태너 그래프

4.1 부호화기

하나의 인코딩 심볼을 전송하는 방식은 다음 과정을 거친다.

1) 주어진 degree 분포 $\rho(i)$에 따라 선택할 소스 심볼, 즉 에지 개수를 결정한다.
2) 결정된 에지 개수만큼 소스 심볼을 랜덤하게 선택, XOR연산하여 인코딩 심볼로 정한다.

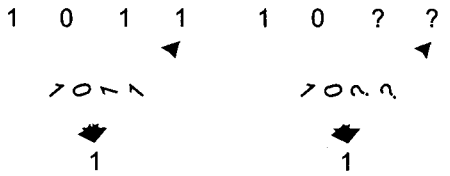
여기서 degree 분포 $\rho(i)$는 선택할 소스 심볼이 i개일 확률을 의미한다. LT code는 위와 같은 방식으로 무한히 많은 인코딩 심볼을 생성하여 전송한다. 하나의 인코딩 심볼을 생성하는 데에는 에지 개수에 비례한 XOR 연산이 필요하다. 따라서 인코딩 심볼 당 평균 연산량은 주어진 degree 분포의 평균 에지 수에 비례한다.

4.2 복호화기

LT 코드의 복호화는 Message-passing 복호화 과정을 사용한다. Message-passing 복호화 방식은 변수 노드와 체크 노드 사이에 메시지를 주고 받으면서 소스 심볼을 복원하는 방식이다. 변수 노드에서 체크 노드로 전달되는 메시지는

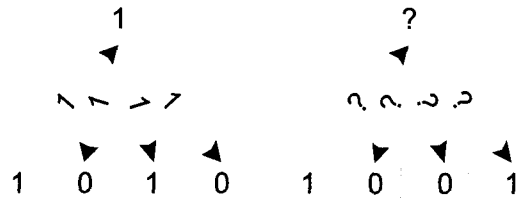
01_ 소스 심볼 수보다 적은 인코딩 심볼로는 k개의 소스 심볼 값을 얻을 수 있는 역행렬을 구성할 수 없음에 따라 이를 증명할 수 있다.

그 변수 노드에 대응하는 비트의 값이며, 체크 노드에서 변수 노드로 전달되는 메시지는 체크 노드가 예측하는 변수 노드의 비트 값이 된다. 일반적인 블록 부호에서 하나의 체크 노드에 연결된 모든 변수 노드와 대응하는 비트의 XOR 합은 0이 되어야 하므로, 체크 노드에서 특정 에지로 전송되는 메시지는 그 에지를 제외한 모든 에지들로부터 받아들인 메시지들의 XOR 연산 결과이다. 만약 적어도 하나의 메시지가 변수 노드의 값을 모른다는 메시지라면 체크 노드는 다른 변수 노드의 비트 값을 예측할 수 없으므로 연산결과는 '알수 없음' 이 된다(그림 8 참조).



(그림 8) 체크 노드에서의 과정

변수 노드에서는 연결된 어떤 체크 노드에서든지 메시지가 값이 정해진 0 또는 1로 전달되면 그것이 변수 노드의 비트값이 되므로, 그 값을 연결된 다른 체크 노드로 전달하는 메시지로서 사용된다(그림 9 참조).



(그림 9) 변수 노드에서의 과정

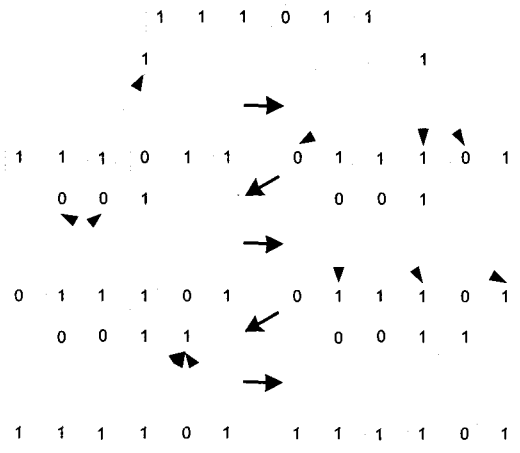
이와 같은 Message-passing 복호화 과정을 간략화하면 다음과 같다.

1) 단 한 개의 소스 심볼의 정보만을 가진 인코딩 심볼을 찾는다.
2) 그 인코딩 심볼은 소스 심볼값을 그대로 의미하므로 이를 복호화한다.
3) 복호화된 소스 심볼은 그것의 값을 이용하여 생성된 다른 인코딩 심볼에서 소스 심볼의 값을 뺀다.
4) 1)~3) 과정을 반복한다.

(그림 9)는 LT code 복호화 과정을 태너 그래프를 이용하여 표현한 간단한 예이다. 소스 심볼의 값이 빼진 인코딩 심볼은 더 이상 그 소스 심볼의 값을 포함하지 않으므로 연결된 에지를 지운다고 볼 수 있다.

이러한 Message-passing 복호화는 ML 복호화보다 떨어지는 성능을 가지며, 복호에 필요한 인코딩 심볼 수가 ML 복호화에 필요한 인코딩 심볼 수보다 작을 수 없다. 그러나 ML 복호화에 비해 훨씬 적은 양의 연산량을 요구한다는 장점이 있다.

위 과정에서 XOR 연산은 소스 심볼들을 복호화하는 데에 필요하다. 그 이유는 XOR 합으로 이루어진 인코딩 심볼의 값에서 복구된 소스 심볼의 값을 빼는 것 또한 XOR연산이기 때문이다. 따라서 복호화 과정에서 필요한 연산량은 소스 심볼의 평균 에지 수와 같은 오더를 갖는다.



(그림 10) 파운틴 코드의 Message-passing 복호화 예

4.3 Ideal Soliton distribution

Ideal Soliton distribution이라 불리는 LT 코드의 degree 분포는 다음과 같이 정의된다.

$$\rho(1) = 1/k$$
$$\rho(i) = 1/i(i-1)$$

위 식의 의미는, 하나의 인코딩 심볼을 통해 한 심볼의 정보만을 그대로 보낼 확률은 $1/k$ 되며, 그보다 많은 i개 심볼 수만큼의 정보를 XOR연산한 합을 보낼 확률이 각각 $1/i(i-1)$ 되는 것을 의미한다.

여기서 인코딩 심볼의 에지 개수를 정하는 확률이 주어졌을 때, 그 에지 개수만큼 소스 심볼들을 선택하며 소스 심볼이 선택될 확률은 모두 동일하다. 예를 들어, k개의 소스 심볼을 전송할 때, 어떤 인코딩 심볼이 3개의 에지를 가지게 된다면, 선택 가능한 경우의 수인 $\binom{k}{3}$가지 에지 조합들은 모두 선택될 확률이 동일하다고 볼 수 있다.

위와 같은 degree 분포가 정의되었을 때 Luby는 다음과 같은 이론을 증명하였다[2].

소스 심볼과 같은 수의 인코딩 심볼을 가졌을 때,
1. 복호화 최초 단계에서 한 개의 소스 심볼만 연결된 인코딩 심볼이 평균적으로 한 개 존재한다.
2. 하나의 소스 심볼을 복호화하고 나머지 연결을 모두 끊은 뒤에 한 개의 소스 심볼만 연결된 인코딩 심볼이 평균적으로 한 개 존재한다.

따라서 실제 현상이 평균적인 현상과 일치한다면 매 단계에서 복호화할 수 있는 소스 심볼의 수는 항상 한 개이다. 이는 Ideal Soliton Distribution으로 생성된 심볼 간의 연결이 한 개라도 없을 경우 복호화가 완전히 불가능하므로 최소한의 에지 수를 갖는다고 볼 수 있다. 따라서 Ideal Soliton distribution은 최소한의 인코딩 심볼수인 소스 심볼 수만큼을 이용하여 최소한의 에지 수로 소스 심볼을 복호화할 수 있기 때문에, 오버헤드가 없고, 복호화 연산량이 최소화된 가장 이상적인 degree 분포라고 할 수 있다.

그러나 이 degree 분포는 매우 나쁜 성능을 보인다. 그 이유는 소스 심볼수와 인코딩 심볼수가 같을 때 매 단계에서 복호화 가능한 소스 심볼이 평균적으로 단 한 개 뿐이므로 복호화가 진행될수록 복호화 실패할 확률이 1로 수렴한다. 이것은 7장에서 모의 실험을 통해 얻은 결과를 확인해 볼 수 있다.

이러한 문제를 해결하기 위하여 어느 정도의 복호화 과정의 성공률을 보장하는 degree 분포가 Luby에 의해서 함께 제안되었다. 이러한 degree 분포를 Robust Soliton distribution이라 한다.

4.4 Robust Soliton distribution

Robust Soliton distribution $\mu(i)$ 는 임의의 0보다 큰 적절한 상수 c에 대하여 다음과 같이 정의된다.

$$R = c \cdot \ln(k/\delta)\sqrt{k}$$
$$\mu(i) = (\rho(i) + \tau(i))/N, \quad N = \sum_i (\rho(i) + \tau(i))$$
$$\tau(i) = \begin{cases} R/k \cdot (1/i), & i = 1, \ldots, k/R - 1 \\ R/k \cdot \ln(R/\delta), & i = 1, \ldots, k/R \\ 0, & i = k/R + 1, \ldots, k \end{cases}$$

위 식에서 보는 것과 같이 Robust Soliton distribution은 기존의 Ideal Soliton distribution $\rho(i)$ 에서 추가적인 분포인 $\tau(i)$를 더한다. N을 나누는 것은 degree 분포의 총 합이 1이 되도록 normalize하는 것을 의미한다. 여기서도 주어진 degree를 이용하여 인코딩 심볼의 degree를 구하였을 때, 그 수만큼의 소스 심볼을 선택하는 데 있어서 모든 심볼은 동일한 확률로 선택된다.

위와 같은 degree 분포를 정의하였을 때 또한 다음과 같은 이론이 증명되었다[2].

어떤 적합한 파라미터 c에 대하여 kN 만큼의 인코딩 심볼을 받았을 때, k개의 소스 심볼을 완전히 복호화하는 데 성공할 확률은 최소 $1-\delta$이다.

여기서 c가 너무 작으면 degree-1인 인코딩 심볼이 생길 확률이 낮아서 복호화가 실패할 수 있고, c가 너무 크면 인코딩 심볼들의 degree가 전체적으로 낮아져서 어떤 인코딩 심볼에도 그 정보가 포함되지 못한 소스 심볼이 발생할 수 있다.

Robust Soliton Distribution은 복호 가능한 소스심볼의 수를 매 단계에서 단 한 개로 제한하지 않고 그 이상의 일정한 값을 허용하면서 복호화가 성공할 수 있도록 한다. 이 분포는 또한 약간의 오버헤드를 허용함으로써 복호화를 성공적으로 이끌어갈 수 있도록 한다. 이 분포에 대해서도 7장에서 모의 실험을 통해 얻은 결과를 정리하였다.

LT code에서 사용되는 Ideal Soliton Distribution과 Robust Soliton Distribution의 총 degree 수의 기대값은 부호화 및 복호화에 필요한 평균 인코딩 심볼 수와 평균 에지 수를 곱한 값이 된다.

$$k \cdot \sum_i i\rho(i) = O(k \log k)$$

$$kN \cdot \sum_i i\mu(i) = O(k \log k)$$

따라서 한 개의 인코딩 심볼을 부호화 하는 데 필요한시간 복잡도는 $O(\log k)$를 가지며, k개의 소스 심볼을 복호화하는 데 필요한 시간복잡도는 $O(k\log k)$를 갖는다.

V. Raptor code

Raptor 코드는 2004년 Amin Shokrollahi에 의하여 개발되었다[3]. 이 코드는 기존의 파운틴 코드인 LT 코드보다 복호화 과정의 연산량 측면에서 더 좋은 성능을 가지고 있다. LT code의 가장 큰 단점은 소스 심볼을 복구하는 데 필요한 연산량이 선형적이지 못하다는 것이다. Raptor code는 이러한 단점을 보완하여 부호화 연산량을 선형 시간 이내로 유지하면서 선형 시간의 복호화 연산량을 얻을 수 있게 한다.

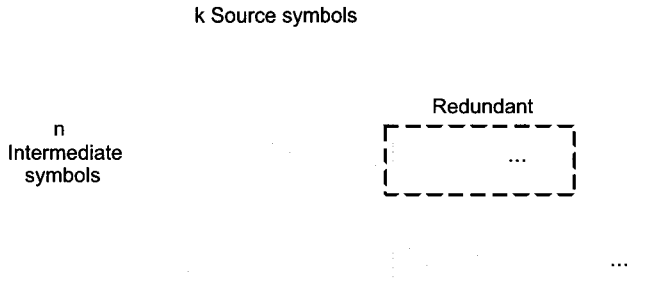
우선 기존의 소스 심볼을 Pre-code함으로써 중간단계 심볼(Intermediate symbol)을 생성하며, 이 심볼들을 LT 코드의 부호화기를 이용하여 인코딩 심볼을 생성한다. 이러한 방식의 성능 개선 요인을 직관적으로 살펴보자면, LT 복호화기가 선형 시간복잡도를 갖도록 연산량을 줄이게 되면, 이에 따라 중간 단계 심볼을 완전히 복원할 수는 없다. 그러나 이를 이용하여 소스 심볼을 완전히 복원할 수 있다. 여기서 Pre-code는 선형 시간복잡도가 보장되는 것을 사용함으로써 전체적인 복호화 과정의 복잡도가 선형이 되도록 만들 수 있다.

VI. 모의 실험

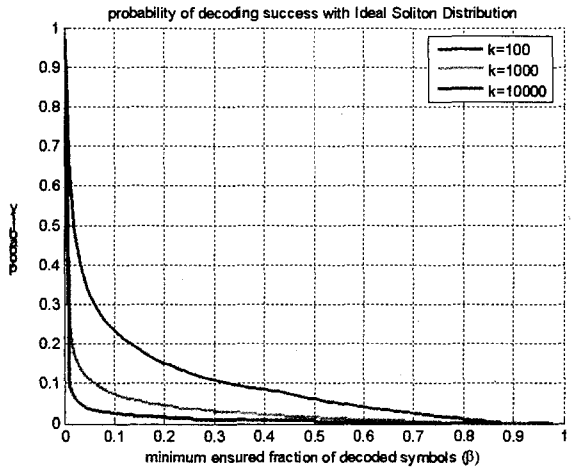
본 장에서는 실제로 파운틴 코드를 사용하여 정보를 전송하였을 때의 시뮬레이션 결과를 도시하였다.

(그림 12)는 Ideal Soliton Distribution을 사용한 LT code를 모의 실험하였다. k개의 소스 심볼을 부호화하여 전송했을 때 k개의 인코딩 심볼을 수신하여 최소 kβ개 이상의 소스 심볼이 복호된 비율을 나타내었다.

(그림 13)은 Robust Soliton Distribution을 사용하여 1000개의 소스 심볼을 LT code로 전송한 결과이다. 최소 99%의 복호 성공률을 갖도록 $\delta = 0.01$ 로 정하였으며, KN개의 인코딩 심볼을 수신하여 최소 $1000 \cdot \beta$개 이상의 소스 심볼이 복



(그림 11) Raptor code의 태너 그래프



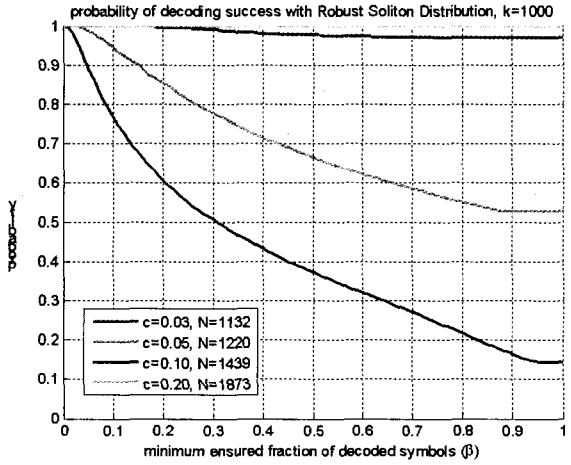
(그림 12) Ideal Soliton Distribution을 사용한 LT code의 복호 보장율

호된 비율을 c의 값을 다르게 하여 나타내었다. 충분히 큰 c에 대하여 목표 수준에 근접하는 복호 성공률을 얻을 수 있으나, 더 많은 인코딩 심볼을 받아야 한다는 단점을 갖는다.

VII. 관련 동향

7.1 파운틴 코드의 확장성

파운틴 코드에 관한 개념은 처음에는 Erasure channel에 적합한 코드를 연구하면서 제기되었으나, 서론에서 언급했던 파운틴 코드의 장점들을 다른 채널에도 적용해보는 연구가 Raptor 코드의 개발 및 실용성 검증 이후 활발하게 진행되었다[4]-[10]. Erasure 채널이 아닌 symmetric channel[4], Gaussian channel[5][7], Fading channel[9][10] 등 다양한 채널에서 파운틴 코드의 성능을 분석 및 검증하는 연구가 진행되었고, 파운틴 코드를 기존의 기술에 접목하는 연구 또한 진행되었다[6][8]. 이외에도 파운틴 코드가 다중 사용자 네트워크 상에서 필요한 채널 코딩 방식으로 알려져 있는 만큼, 다양한 다중 사용자 환경에서의 성능 연구에 대한 여지가



(그림 13) Robust Soliton 분포를 사용한 LT code의 복호 보장율

많이 남아 있으며, 전세계에서 이에 대한 연구가 진행되고 있다.

7.2 표준화 동향

파운틴 코드는 그것이 가지고 있는 장점을 필요로 하는 곳에서 표준화 논의가 이루어지고 있다. 현재까지는 2005년 Raptor code가 3GPP TS 26.346 MBMS와 DVB-H IP Datacast에서 사용되는 표준 방송용 채널 부호화 알고리즘으로서 채택되었다[12][13]. 표준에서 사용하고 있는 Raptor code는 Systematic[2]한 특성을 갖도록 정의되었다. 3GPP MBMS와 DVB-H에서는 이러한 특성의 Raptor code를 구현하기 위하여 pre-code 과정에서 LT decoding을 사용한다. 즉 기존의 소스 심볼을 Raptor code 내의 LT code 부분과 같은 소스 심볼 선택 패턴으로 부호화된 심볼들로 가정한다.

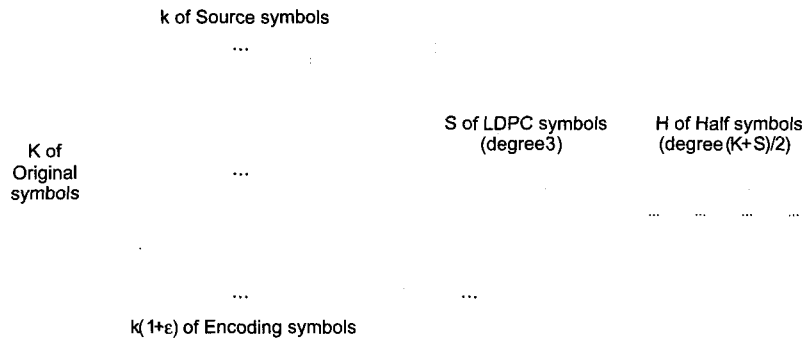
따라서 이를 LT decoding해서 얻은 중간단계 심볼을 LT encoding 하면 전송할 인코딩 심볼 내에는 소스 심볼이 그대로 포함된다.

VIII. 결 론

본 논문에서는 파운틴 코드의 구현과 기본 원리를 확인하였다. 파운틴 코드는 한정된 양의 정보를 이용하여 무한히 많은 패킷을 생성하여 전송할 수 있기 때문에 Rateless 코드라고도 불린다. 따라서 일반적인 채널 부호화 기법과 같이 제한된 전송률로 얼마나 낮은 에러율을 얻는지에 대해 분석하지 않고, 무한하게 전송하는 패킷 중에서 완벽하게 정보를 수신하기 위하여 얼마나 작은 양의 패킷과 복호화 과정의 복잡도를 갖게 하는지에 대해 분석한다.

파운틴 코드는 기본적으로 Erasure 채널에서 동작하도록 설계되므로 유선 네트워크 상에서의 통신에서도 적합하지만, 무선 통신에서도 MAC layer 상에서는 CRC가 적용되어 수신된 패킷이 옳은 정보인지 아닌지의 여부만이 결정되기 때문에 Erasure 채널과 같은 경우로 간주하여 파운틴 코드를 적용할 수 있다. MAC layer에서 사용되는 다른 코딩 기법인 H-ARQ와 비교해 볼 때, 파운틴 코드는 송신단과 수신단 사이의 상호작용 없이도 수신단에서 오류 없이 완벽하게 정보를 수신할 수 있게 한다는 특징이 있다. 따라서 양 단 사이에서 정보를 완벽하게 수신할 때까지 반복적인 상호작용을 통해 통신하는 H-ARQ(Hybrid Automatic Repeat Request) 전송기법에 비해 장점을 가진다.

[1] J. Byers, M. Luby, M. Mitzenmacher, and A. Rege, "A



(그림 14) 표준에서 사용되는 Raptor code 시스템

2_ 어떠한 코드가 systematic하다는 것은 부호화된 코드워드가 전송할 입력 심볼 그 자체를 포함하고 있다는 것을 말한다.

digital fountain approach to reliable distribution of bulk data," in Proc. ACM SIGCOMM' 98, Vancouver, Canada, Jan. 1998, pp. 56-67, see also "A Digital Fountain Approach to Asynchronous Reliable Multicast," IEEE Journal on Selected Areas in Communications, Vol. 20, No. 8, Oct. 2002

[2] M. Luby, "LT Codes," in Proc. 43rd Annu. IEEE Symp. Foundataions of Computer Science (FOCS), Vancouver, Canada, Nov. 2002, pp. 271-280.

[3] A. Shokrollahi, "Raptor Codes," IEEE Trans. on Information Theory, Vol. 52, No. 6, Jun. 2006, pp. 2551-2567.

[4] O. Etesami and Amin Shokrollahi, "Raptor Codes on Binary Memoryless Symmetric Channels," IEEE Trans. on Information Theory, Vol. 52, No. 5, May 2006, pp. 2033-2051.

[5] R. Palanki and J. S. Yedidia, "Rateless codes on noisy channels," in Proc. IEEE Int. Symp. Information Theory, Chicago, USA, July 2004, pp. 37.

[6] U. Erez, G. W. Wornell, M. D. Trott, "Rateless space-time coding," in Proc. IEEE Int. Symp. Information Theory, Adelaide, Austrailia, Sept. 2005.

[7] U. Erez, M. D. Trott and G. W. Wornell, "Rateless Coding and Perfect Rate-Compatible Codes for Gaussian Channels," in Proc. IEEE Int. Symp. Information Theory, Seattle, USA, Jul. 2006.

[8] A. W. Eckford and W. Yu, "Rateless Slepian-Wolf Codes," in Proc. the 39th Asilomar Conference on Signals, Systems and Computers, Oct. 2005, pp. 1757-1761.

[9] J. Castura, Y. Mao and S. C. Draper, "On Rateless Coding over Fading Channels with Delay Constraints," in Proc. IEEE Int. Symp. Information Theory, Seattle, USA, Jul. 2006.

[10] S. C. Draper, B. Frey and F. R. Kschischang, "Rateless Coding for Non-Ergodic Channels with Decoder Channel State Information," submitted to the IEEE Trans. on Information Theory.

[11] D. J. C. MacKay, "Fountain codes," IEE Proc.-Commun., Vol. 152, No. 6, Dec. 2005, pp. 1062-1068.

[12] "Technical Specification Group Services and System Aspects; Multimedia Broadcast/Multicast Services (MBMS); Protocols and Codecs (Release 6)," 3rd Generation Partnership Project (3GPP), Tech. Rep. 3GPP TS 26.346 V7.3.0, 3GPP, Mar. 2007.

[13] "Digital Video Broadcasting (DVB); IP Datacast over DVB-H: Content Delivery Protocols," Digital Video Broadcasting(DVB), ETSI TS 102 472 V1.2.1, DVB-IPDC, Dec. 2006.

약 력

박 도 형

2001년 ~ 2005년 한국과학기술원 학사
2006년 ~ 현재 한국과학기술원 석사과정
관심분야 : 정보이론, 오류 정정 부호

김 민 규

1999년~ 2007년 경북대학교 학사
2007년 ~ 현재 한국과학기술원 석사과정
관심분야 : 정보이론, 오류 정정 부호

정 세 영

1995년 ~ 2000년 MIT 박사
2000년 ~ 2004년 Airvana, Inc. principal engineer
2005년 ~ 현재 한국과학기술원 조교수
관심분야 : 정보이론, 오류 정정 부호, 무선통신