通信情報保護學會論文誌
第4卷 第2號, 1994. 12

자료보호를 위한 오류 탐지 및 교정

최 우 영*, 서 창 호**, 정 인 정***, 임 종 인****

Error Detection & Correction for the Data Protection

Woo-Young Choi, Chang-Ho Seo, In-Jeong Chung and Jong-In Lim

요 약

위성을 이용한 통신에서 오류의 검출과 교정은 중요한 과제중의 하나이다. 이 논문에서는, 위성통신에 많이 응용되는 컨벌루션 부호에 있어서, 전송되는 데이터에 패리티 비트를 부가하고 인터리브를 행하여, 채널상에서 나타나는 집단적 오류에 대해 능률적으로 대처할 수 있는 방법을 제시하고 있다. 이 방법은 부호화 과정에서 패리티비트를 얻기 위한 추가적 계산이 필요 없어서 계산의 복잡도를 증가시키지 않는다. 또, 부호해석 과정에서는 개선된 알고리즘의 적용을 통하여, 예상되는 오류의 발생량이 큰경우에만 패리티 정보를 참조케함으로써 불필요한 계산량의 증가를 줄이고 있다.

Abstract

This paper presents a new encoding and decoding method of convolutional code to correct the burst error of the transmitted data on the satellite communication system. The new proposed method encodes data by adding two redundant parity bits. In the decoding stage, we can avoid the incorrect decoding problems by extending the minimum distance of encoded codeword only when the number of erroneous bits equals to the threshold value.

1. 서 론

통신 상에서 발생하는 오류는 통신시스템 전체에 대해 어떠한 형태로든 부영향을 미칠 수 있고, 응답시간과 시스템의 성능 저하를 가져온다. 특별한 신뢰성이 요구되는 상황에서의 발견되지 않은

* 금성 정보통신 연구소 연구원
** 고려대학교 대학원 수학과 박사과정
*** 고려대학교 전산학과 부교수
**** 고려대학교 수학과 교수

오류의 부작용은 예측할 수 없다는 면에서 훨씬 더 심각할 수 있다.[18] 따라서, 통신 상에서 상존하고 있는 오류발생의 확률에 대해 효율적이고 신뢰성 있는 대책이 마련되어야만 한다.

1965년 인텔세트 1호의 발사로 본격적으로 시작된 위성통신은[20], 최근에는 국제간의 통신이나, 자국내 지상통신시스템의 보완책, CATV의 프로그램 분배, 기업내 통신망등으로 그 사용량이나 분야가 증가, 다양화되는 추세이다.[11,13] 그러나, 위성을 이용한 통신시스템에서도 전송되는 데이터는 여러가지 많은 장애를 받을 수 있는데, 이러한 장애는 점차로 저전력화 되어가는 디지탈 통신시스템에서 중요한 문제점으로 대두되고 있다.[12,20] 위성통신 시스템의 성능을 저하시키는 요인들로는 채널간의 간섭, 대기권이나 우주에서의 자속밀도 약화로 인한 채널상의 신호약화, 태양 간섭, 강우감쇄 등이 있다.[20] 이러한 장애요인들은 독립적이기보다, 상호복합적으로 작용하는 일이 많으므로 채널상에서 집단 오류(burst error)를 일으키는 요인이 될 수 있다.[8,10,14]

많은 위성통신 시스템에서 자료의 전송 형태는 단방향(simplex style)의 채널을 사용하며 역채널이 없는 경우가 많으므로[15] 재전송 요구를 할 수 없고, 또, 재전송이 가능하더라도 지연시간이 길다. 따라서, 오류가 발생하였을 때에 수신 측에서 능동적으로 교정할 수 있는 순방향 오류정정방식(foreward-error correction:FEC)을 마련해야만 한다.[2]

이 논문에서는 컨벌루션 부호(convolutional code)에 간단한 패리티 비트(parity bit)의 부가만으로 집단오류를 정정하는 알고리즘을 제안한다. 그리고 더 많은 집단 오류의 정정을 위해 인터리빙(interleaving) 방식을 도입함으로써, 부호해석 과정에서 적용되는, 개선된 알고리즘의 오류정정 효율을 높이는 방법에 대해 소개하고자 한다.

다음에 소개되고 있는 내용으로, 2장에서는 위성통신상에서 순방향 오류정정이 필요한 이유를 제시하였다. 3장에서는 간략하게 컨벌루션 부호화와 부호해석과정, 그리고 오류의 제어방법과 문제점에 대해 설명하고 있으며, 4장에서는 패리티 정보를 부가하는 부호화방법과 새로운 부호해석 알고리즘을 제안한다. 그리고 최종적으로 개선된 부호해석 알고리즘에 인터리빙 방식을 도입하여 효과적으로 집단오류를 정정하는 방법을 소개하였다. 5장에서는 웍스테이션(workstation)상에서의 시뮬레이션을 통하여, 각 방식들의 오류정정 능력을 비교하고 있으며, 마지막으로 6장에서는 결론과 문제점, 그리고 앞으로의 연구 방향을 제시하고 있다.

2. 순방향 오류 정정 방법의 필요성

통신상에서 발생하는 대부분의 오류는 백색 가우스형태(white gaussian style)[9,10] 이어서, 오류의 검출과 수정을 염두에 두고 통신시스템을 디자인할 때, 이러한 형태의 오류방지를 위해 노력하는 것만으로도 충분한 경우가 많다.[9] 위성통신의 경우에도 BER(bit error rate)은 10^{-8}정도이므로, 지상의 마이크로웨이브 통신의 BER 10^{-5}보다도 낮고, 대부분 백색 가우스 형태를 가진다.[13] 하지만, 위성통신은 지상의 통신시스템보다 오류를 일으킬 수 있는 위험요소들을 더 많이 가지고 있으며, 집단오류의 발생 가능성을 항상 염두에 두어야 한다.[13,14,15]

위성통신에서의 오류정정을 위해서 사용될 수 있는 방법으로는, 먼저 역방향 오류정정방식(backward-error correction)인 오류검출 후 자동 재전송요구(automatic repeat request:ARQ)방식[18,36]을 생각할 수 있다. 그러나, 위성통신의 경우는, 지상의 지구국과 통신위성과의 거리가 대략 35,800Km라고 할 때 최소한 71,600Km의 거리를 신호가 통과하여야 하므로, 통신위성에서 발생하는 지연까지 포함하여 대략 250ms의 전송지연(round-trip delay)이 발생하게 된다.[20] 이러한 지연시간과 함께 Stop-and-Wait ARQ방식을 사용하는 경우에는 ACK나 NAK신호를 받기 위

해 최소 0.5sec의 지연이 추가로 발생하게 된다.[8,13] 위성의 경우, 통신에 이용할 수 있는 최대범위가 최대 지연시간(maximum round-trip delay)과 밀접한 관계가 있다고 보면, ARQ방식을 쉽사리 사용할 수 없다는 것이 분명해진다. 또 ARQ는 시스템에 역채널이 존재하지 않는 경우에는 사용할 수 없다. 따라서, ARQ방식을 사용할 수 없는 경우에는 부호화율(code rate)에서 불이익을 감수하더라도 수신 측에서 송신 측에 대해 재전송 요구없이 오류를 교정할 수 있는 충분한 잉여 정보를 부가하여 전송하여야 한다.[3,21]

순방향 오류정정방식을 사용할 경우에는, 일정 수준 이상의 수정능력을 달성하기 위해서 부호화(encoding)방법과 부호해석(decoding)과정이 복잡해지고, 부가되는 기기들이 복잡해진다.[21] 또, 사용되는 오류정정부호는 단지 오류를 검출해내는 것보다 많은 양의 잉여비트가 필요하여, 실제 데이터의 부호화율이 낮다는 단점을 가진다.[1,3,21] 그러나, 순방향 오류정정방식은 재전송 요구가 없으므로 전송지연이 짧고, 연속적인 데이터의 흐름이 가능하며, ARQ방식을 사용할 수 없는 경우에 효율적인 대책이 될수 있으며, 영역제한도 ARQ방식에 비해 적다는 이점을 가지고 있다.[16,17]

순방향 오류정정을 위해서는 RS부호[11,2]와 같은 여러 가지 형태의 블록부호와 연속부호인 컨벌루션 부호가 많이 사용되며, 두가지를 함께 사용한 하이브리드(hybrid) 형태의 부호도 사용된다.[7] 특히 컨벌루션 부호는 실제 응용적인 측면에서 블록부호보다 유망하다.[2,21] 왜냐하면, 통신 장치의 복잡도가 비슷한 정도라면 컨벌루션 부호의 오류정정 능력이 더 뛰어나기 때문이다.[11,5,21]

본 논문에서는 컨벌루션 부호를 사용할 때, 집단적 오류정정 방법의 개선방안에 대해 자세히 다루고자 한다.

3. 컨벌루션 부호를 이용한 순방향 오류 정정방식

컨벌루션 부호의 사용은, 디지탈 통신에서 에너지 효율을 높이는데 좋은 장식이다. 이 부호는 일반적으로 전송 효율이 높아서 대역폭의 제한이 크지않은 경우에 효율적이며, 사용 전력의 제한이 많이 요구되는 위성통신 시스템에 적합한 오류정정방법이다.[20] 이 방식은 쉬프트 레지스터(shift register)와 배타적 논리합(exclusive-OR) 연산을 행하는 mod-2가산기만의 간단한 구성으로 부호화시킬 수가 있다. 또, 블럭코드에 비해 쉽게 순방향 오류정정이 가능하다는 이점이 있다.[1,2]

본 장에서 소개되는 컨벌루션 부호화 과정에서 얻어지는 부호들 역시 쉬프트 레지스터와 mod-2 연산기를 이용한 일반적인 형태이다. 그렇지만, 생성되는 부호가 짝(even)개의 비트들로 구성되는 경우에는 부호해석 과정에서 판단기준이 되는 해밍 거리(Hamming distance)의 판정이 불가능한 경우가 있다.[20] 따라서, 본 논문에서는 1비트의 입력에 대해 홀수(odd)개의 비트로 구성된 부호를 생성하는 방법을 택하였다. 홀수개의 비트로 구성된 부호에 대해서는 해밍 거리를 비교할 수 없는 상태가 일어나지 않기 때문이데, 이 방법에 대한 자세한 내용들은 4장의 개선된 부호해석 알고리즘에서 논의될 것이다.

3.1 컨벌루션 부호화와 최대 근사 부호해석 과정에서의 오류정정

3.1.1 부호화 과정 (Encoding Process)

일반적인 컨벌루션 부호는 $X = (X_1, X_2, \ldots, X_L)$의 원래의 정보를 사용하여 $Y = (X_1, X_2, \ldots, X_N)$의 전송 데이터를 만들어 낸다.(일반적으로 $L < N$이다.)[19,11]

다음은 쉬프트 레지스터의 상태 개수가 S_1, S_2, S_3, S_4의 4가지이고, 한번에 하나의 비트씩 쉬프트하는 경우에 mod-2 가산기를 이용한 부호화이다.

생성 다항식은 다음과 같이 나타낼 수 있다.

$Y_1 = S_1$

$Y_2 = S_1 \oplus S_2$

$Y_3 = S_1 \oplus S_3$

$Y_4 = S_1 \oplus S_2 \oplus S_3 \oplus S_4$

$Y_5 = S_1 \oplus S_3 \oplus S_4$

입력 X = (1101)의 부호화 과정의 예는 다음과 같다.

표 1 컨벌루션 부호화 과정에서 레지스터의 상태 및 출력의 보기

입력X \ 레지스터 상태	S_1	S_2	S_3	S_4	Y_1	Y_2	Y_3	Y_4	Y_5
1	1	0	0	0	1	1	1	1	1
1	1	1	0	0	1	0	1	0	1
0	0	1	1	0	0	1	1	0	1
1	1	0	1	1	1	1	0	1	1

3.1.2 부호해석 과정 (Decoding Process)

컨벌루션 부호를 부호해석하는 것은, 부호화하기 이전의 원래의 비트열(bit sequence)을 재생해 내는 것을 의미하는 것이다. 여기에서는 일반적으로 많이 이용되어지는 최대 근사 부호해석방법(Maximum Likelihood Decoding:MLD)[9,11]을 사용하기로 하자. MLD는 헤밍 거리를 비교하여 가장 근사한 값을 추정하는 부호해석 방법이다.

(보기1) 위에서 예로 든 X = (1,1,0,1) 를 Y = (11111,10101,01101,11011)의 전송부호를 만들어 전송한 경우의 이진 부호해석 트리는 다음과 같이 주어진다.

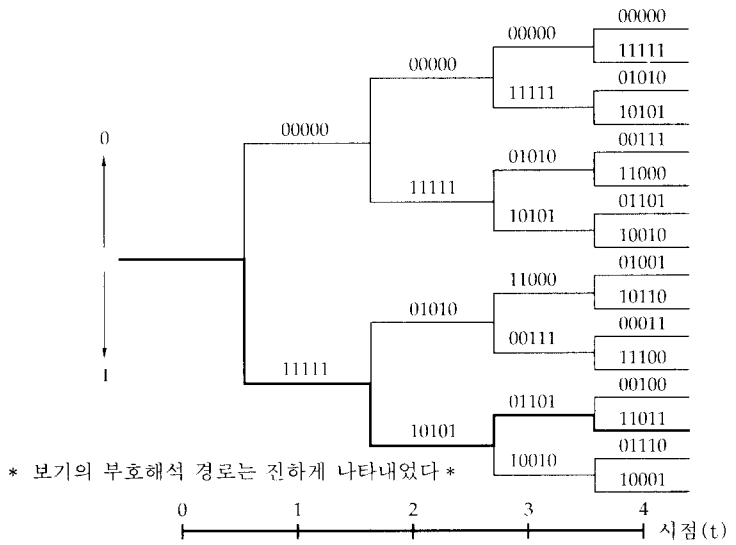


그림 1. 컨벌루션 부호의 이진 부호해석 트리 (binary decoding tree)

부호화된 비트 Y = (11111,10101,01101, 11011)가 전송되는 도중에, 잡음의 영향을 받아 변형된 경우가 생기면 부호해석 과정은 다음과 같은 경로(path)를 선택한다.

잡음요소를 E라고 정의하면 실제적으로 수신측의 부호해석기(decoder)에 전달되는 데이터는 R = Y + E이므로, E의 형태가 (10010,00011, 00000,00100)라면, 수신측에 전달되는 R = (01101,10110,01101,11111)가 된다.

R =(01101,10110,01101,11111)가 부호해석기의 입력으로 들어오면 이진 트리(binary tree)의 어느 경로를 선택할 것인가는 헤밍 거리(Hamming distance)를 사용하여, 거리가 짧은 경로를 선택하게 된다.[1,2,21]

이러한 방법으로 계속 노드(node)에서 경로를 판정해 나가면, 위의 그림에서 진하게 나타난 경로를 따라가게 된다. 그리고, 그 결과 우리가 원래 전송하고자 하였던 데이터 X = (1101)을 구할 수 있다.

3.2 헤밍 거리에 의존하는 부호해석 방법의 문제점

FEC를 위해 사용되는 컨벌루션 부호의 가장 큰 단점은 채널을 통해 부호해석기에 입력되는 R에 포함되어있는 오류가 집단적인(burst)경향을 띠는 경우에, 이러한 집단적 오류의 정정능력에 한계가 있다는 점이다.[11,16,20] - 이것은 5장의 시뮬레이션 결과를 통해서도 알수있다 - 왜냐하면, MLD 헤밍 거리값이 절대적으로 중요한 부호해석 정보가 되므로, 비교 단계에서 헤밍 거리값의 변화가 큰 경우에는 이진 부호해석 트리에서 결정할수 없게된다.[20] 이것은 Viterbi의 부호 해석 방법이나, 축차(sequential)부호해석 방법에서도 동일하게 나타나는 문제점이다. 특히, 축차 부호해석 방법에서는 Fano의 평가량을 따져서, 피드백(feed back)을 수행할 수 있다.[21] 그러나, 채널의 오류 발생 가능성이 큰 경우에는 피드백을 행한다 해도 잘못된 경로를 따라가는 것을 완전히 막을수 없다.[16,17,18]

다음의 보기는 실제적으로 다수개의 집단적 오류가 발생한 경우에, 부호해석 과정의 잘못된 전개를 보여주고 있다.

(보기1) 3개의 집단적 오류 발생시의 잘못된 부호해석과정

잡음요소 E =(11010,00011,00101,00100)라면,
전송 데이터 Y = (11111,10101,01101,11011)는 R = Y + E가 되어, 실제 수신 받는 전송데이터는 R = (00101,10110,01000,11111)가 된다. 이 가운데에서 처음으로 부호해석되는 부호열 (00101)은 부호해석 이진트리를 사용하여, 헤밍 거리로 경로를 결정한다면 다음과 같다.

R =(00101) → 00000 (3,5번째 비트가 다름. 헤밍거리는 2)
* 헤밍거리를 비교한 결과, 거리가 짧은 00000을 선택하게 된다.(틀린 결과를 얻게 됨)
→ 11111(1,2,4번째 비트가 다름. 헤밍거리는 3)

따라서, 트리의 맨 첫 단계에서 판단 오류를 일으키게 되고, 계속 경로를 따라 가보면, X값으로 (0100)을 얻게 되는데, 이것은 우리가 맨처음 부호화 단계에서 넣어준 X =(1101)과는 다른 것이다. 이것을 부호해석시 오류의 확산(decoding error propagation)이라고 한다.

위의 경우에 피드백을 일으킨다고 가정하면, 피드백이 일어나는 누적된 '복귀 헤밍 거리 값 (feed back Hamming distance)'의 기준치(threshold value)를 크게 잡아야 하고, 그 결과 거의 모든 경로의 가지(branch)를 탐색하게 되므로 부호해석 시간이 대단히 커질뿐더러[10,11], 이렇게 복잡한 피드백 후에 얻은 경로도 반드시 올바른 결과가 아닌 경우도 있다.

4. 패리티정보를 이용하는 새로운 부호 해석 알고리즘과 인터리빙을 사용하는 API(Added Parity & Interleaving Method) 방법

4.1 패리티정보의 부가

채널 상에서 집단 오류가 발생하였을 때, 수신측에서 일어날수 있는 잘못된 부호 해석을 막기 위한 방법으로써, 별도의 계산이 필요없는 패리티 정보를, 부가하여 사용하기로 하자. 부가되는 2개의 패리티 정보 가운데, 첫번째 패리티는 전체 비트 가운데 하나의 비트에 대한 패리티로 사용되며, 두번째 패리티는 전송되는 전체 데이터 비트에 대한 것이다. 이런 형태로 패리티를 사용하게 되면 전체 비트열에 대한 정보뿐 아니라, 특정 비트의 상태를 알고 있으므로 패리티 비트 자체의 오류발생 여부와 함께, 패리티 2비트를 포함하여 전체 비트 중에 3비트의 집단적 오류까지를 정확히 교정할 수가 있다.

첫번째 패리티 비트는 Y의 Y_1비트의 상태를 나타내며, 두번째의 패리티 비트는 전체에 대한 것이다. 여기서, 첫번째 패리티를 Y_1비트로 잡은 것은 부가적인 연산이 필요없이 레지스터 S_1의 값을 그대로 사용할 수가 있기 때문인데, 그밖의 Y_2, Y_3, Y_4, Y_5 중 하나를 선택한다해도 패리티 트리의 값에만 변동이 있을뿐, 부호해석 과정에는 아무 영향이 없다.

다음은 패리티 정보를 부가하는 경우의 생성 다항식이다.

$$Y_1 = S_1$$
$$Y_2 = S_1 \oplus S_2$$
$$Y_3 = S_1 \oplus S_3$$
$$Y_4 = S_1 \oplus S_2 \oplus S_3 \oplus S_4$$
$$Y_5 = S_1 \oplus S_3 \oplus S_4$$

$$P_1 = Y_1 = S_1$$
$$P_2 = Y_1 \oplus Y_2 \oplus Y_3 \oplus Y_4 \oplus Y_5$$

위에서 두번째 패리티 비트의 생성식을 살펴보면, 부호화기에서 별도의 mod-2 연산기를 추가할 필요가 없음을 알수 있다.

즉, $P_2 = Y_1 \oplus Y_2 \oplus Y_3 \oplus Y_4 \oplus Y_5$이므로, 다음과 같이 다시 쓸수 있다.

$$P_2 = S_1 \oplus (S_1 \oplus S_2) \oplus (S_1 \oplus S_3) \oplus (S_1 \oplus S_2 \oplus S_3 \oplus S_4) \oplus (S_1 \oplus S_3 \oplus S_4) = S_1 \oplus S_3 = Y_3$$

수식의 결과에 따라, 부호화기를 아래처럼 간단하게 만들 수있다.

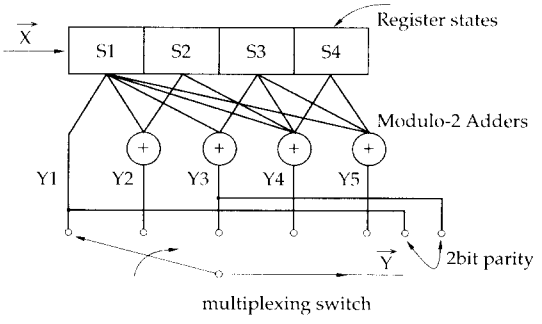


그림 2. 2비트의 패리티 정보를 부가하는 간략화된 부호화기

위의 그림 2와 같은 부호화 과정을 거치게 되면, 실제 전성되는 형태의 부호가 만들어진다. 새로운 것은 부호뒤에 패리티 비트가 붙어서 함께 전송되는 것이다. 여기서 우리는 수신 측에서 비교 부호의 이진 부호해석 트리를 가지듯이, 패리티 비트에 대한 이진 트리도 쉽게 생성할 수 있음을 알수 있다. 따라서 부호해석과정에서, 이진 부호해석 트리와 함께 패리트 트리도 같이 비교에 사용될 수 있다. 부호해석 트리는 3장의 그림 1과 동일하며, 패리티 트리는 다음의 그림 3과 같다.

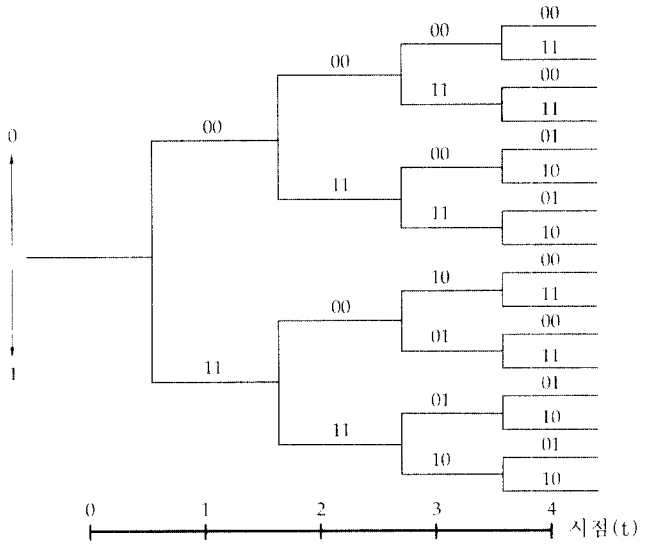


그림 3. 부가된 정보로 구성한 패리티 트리(parity tree)

비록 패리티가 부호화 과정에 포함되어 생성되고 다른 비트들과 구분없이 함께 전송이 되지만, 수신 측에서는 따로 분리시켜 별도의 트리형태로 만들어 보관함이 편리하다. 왜냐하면 패리티 비트가 부호해석 과정에 참여하는 경우도 있지만, 참조가 불필요한 경우도 있기 때문이다. 이것은 다음 절에 제시하는 개선된 부호 해석 알고리즘을 따라가보면 분명하다.

4.2 패리티 정보를 사용하는 새로운 부호해석 알고리즘

수신 측에서 부호해석 트리와 패리티 트리를 가지고, 하나의 부호열 당 3개까지의 집단적 오류를 교정하기 위해서, 다음과 같은 새로운 부호해석 알고리즘을 제안한다.

(규칙 1) 패리티 2비트를 제외한 나머지 부호들을 부호해석 트리(그림 2)의 두개의 비교 부호들과 헤밍거리를 계산하여 거리가 "0" 혹은 "1"인 경로가 있으면, 패리티 트리(그림 5)를 참조하지 않고 부호해석 트리에서 거리가 "0" 또는 "1"인 경로를 따라간다.

(규칙 2) 패리티 2비트를 제외한 나머지 부호들을, 부호해석 트리의 두개의 비교 부호들과 헤밍거리를 계산하여 거리가 2:3 혹은 3:2 과 같은 형태로 나타나는 경우는 다음 두가지 경우 가운데 한가지이다.

(경우 2-1) 패리티 트리를 참조해서, 해당 부호해석단계의 패리티 비트와 수신된 패리티 비트들을 비교했을 때, 패리티 비트들도 일치하는 경로가 없다면, 부호해석 트리의 헤밍거리가 짧은 쪽의 경로를 택한다.

(경우 2-2) 수신된 패리티 비트를 패리티 트리의 해당 단계와 비교해서 일치하는 것이 있다면, 패리티 트리에서 선택되는 경로 방향과 같은 방향을 택하게 된다.(부호해석 트리에서 위쪽 혹은 아래쪽)

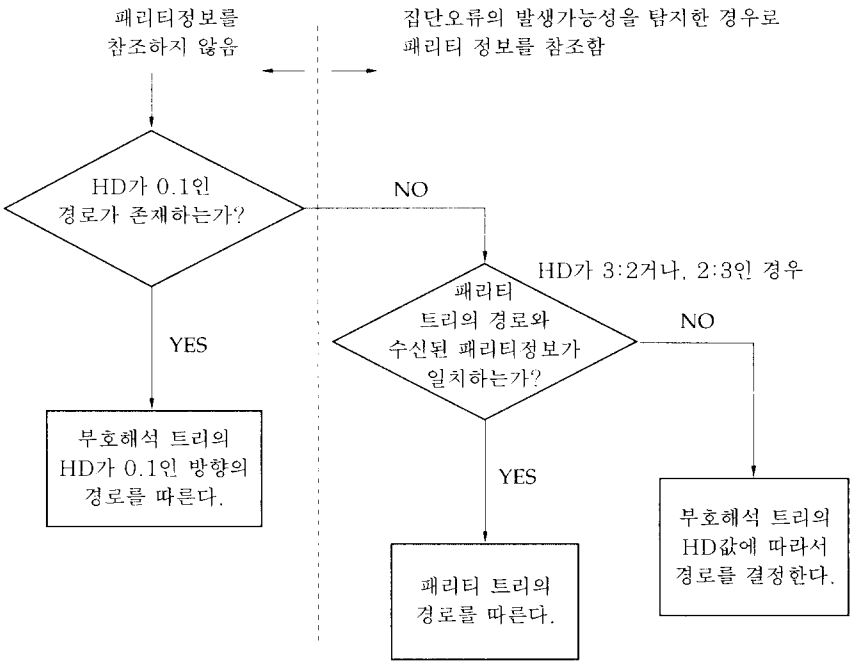


그림 4. 패리티 정보를 사용하는 개선된 부호해석 알고리즘

이 알고리즘을 사용하여 부호해석을 수행하는 경우, 최대 두번의 비교과정을 통하여 경로를 선택할 수 있다. 즉, 예상되는 오류의 개수에 따라 한번의 비교만으로 충분하고, 더이상의 계산이 필요 없을 수 있다는 것이다.

새로운 알고리즘은 오류가 포함된 정도를 예측하여, 탄력적으로 부호해석과정을 수행한다. 즉 알고리즘의 각 단계들을 따라가 보면 흐름도에서는 두번의 비교 단계가 존재한다. 그러나, 헤밍 거리가 0이나 1인 경우는 전송된 데이터에 오류가 없거나, 하나밖에 발생하지 않은 경우로 생각할 수 있다. 따라서 이러한 경우는 패리티 정보를 참조하지 않고서 부호해석을 행하게하여, 부호해석 단계의 불필요한 절차를 감소시키고 있다.

위의 부호해석 알고리즘의 신뢰성은 수신된 부호열 하나당, 최대 3개의 집단적 오류가 나타난 경우를 모두 생각해보면 분명하다. 하나의 부호열에 최대 3개까지의 오류가 발생하는 경우는 다음과 같이 나누어 생각해 볼 수 있다.

표 2. 전송되는 하나의 부호열당 가능한 오류 발생의 형태들(오류의 개수가 최대 3인 경우)

패리티 비트내의 오류 개수	데이터 비트의 오류 개수
0	3
1	2
2	1

표 2에서 나타나는 오류발생의 형태들에 대하여, 이진 부호새석 트리에 있는 16개의 모든 경로에 부호해석 알고리즘을 적용해 보면, 3개까지의 오류를 정확히 교정해내고 있다. 이것은 수신된 부호열에 대해 임의의 위치에 오류를 발생시켜 시뮬레이션 시킨 5장의 결과를 참조하면 명확하다.

새로 제안한 알고리즘을 다음의 보기 2에서 적용해 보기로 하자.

(보기 2) 원래 전송하고자하는 데이터 Y =(1111111,0101000,0011110,000110)일때

오류가 발생하여 전송된 데이터들	최대 비트 오류의 개수 (한 코드당)	부호해석과정에서 사용되는 알고리즘의 규칙과 경우	오류교정 여부
$\vec{R1}$ = (1011100, 1101011, 1011110, 0011111)	3	규칙 1	○
$\vec{R2}$ = (0011110, 1001010, 1111111, 1011101)	3	규칙 2, 경우 2-1	○
$\vec{R3}$ = (1000111, 0010000, 0000010, 1111100)	3	규칙 2, 경우 2-2	○
$\vec{R4}$ = (1000110, 0010010, 0000011, 1111101)	4	규칙 2, 경우 2-1	×

위의 보기들에서도 보여졌지만, 이 부호해석 방법은 하나의 부호열 당 오류가 3개 이하인 경우에 대해 대단히 효과적이다.

이 부호해석 방식에서 2비트의 패리티를 얻기 위해 부호화기에 부가되는 연산 장치는 아무것도 없다. 따라서 복잡한 연산이 필요 없으며, 수신 측에서도 패리티 비트를 따로 관리함으로써, 집단적 오류가 아닌 단일 오류의 경우에는 패리티 비트에 상관없이 부호해석함으로써, 잉여 정보의 사용으로 부가되는 부하(overhead)를 최소화하고자 하였다.

4.3 패리티 정보 부가방식의 문제점

4.1, 4.2절에서 제시한 방법의 문제점으로 가장 심각한 것은, 한 부호열(codeword)당 3개까지의 집단적 오류밖에 검출하고, 수정하지 못한다는 것이다. - 보기 2의 $\vec{R_4}$를 참조 - 그러나, 일반적으로 집단적 오류의 성질상 한 부호열 전체가 장애를 일으켜 변형될 수도 있고, 경우에 따라서는 이웃한 더 많은 부호 비트들이 변형될 수 있다.[7,11] 다음의 그림 5는 1200bps의 비교적 느린 속도로 데이터를 전송하는 경우에, 한개의 오류가 생긴 비트 다음에 오는 데이터 비트의 수에 따라 오류가 연속적으로 발생할 확률을 보여주고 있다.[10] 오류의 발생이 전송속도와 빌접한 관계가 있다는 것을 생각하면, 보다 빠른 전송속도 - 1.2GHz의 주파수 대역의 경우, 전송속도는 32Kbps - 를 가진 통신 시스템에서는 연속된 오류의 발생 가능성이 더욱 높아짐에 유의 하여야 한다.[10]

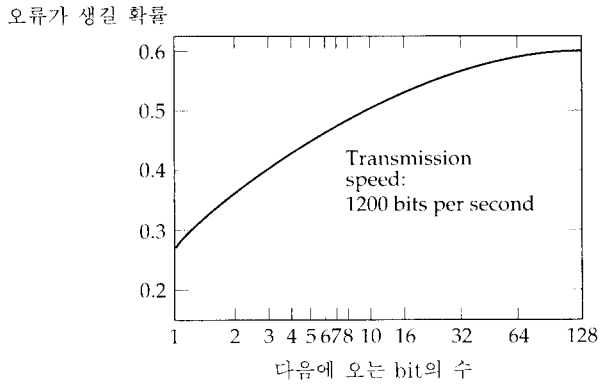


그림 5. 하나의 오류 다음에 오는 비트내에 오류가 생길 확률

3개 이상의 집단오류가 발생하는 상황에서는 4.2절에서 제시한 패리티 정보를 부가하여 전송하고, 수신 측에서는 이 패리티 정보를 이용하여 부호해석하는 새로운 알고리즘을 적용한다해도, 시뮬레이션 결과를 통해 알 수 있듯이 부호해석 오류의 증가를 막을 수가 없다. 따라서, 더 많은 집단 오류 발생에 대처하기 위해서는 별도의 방법이 필요함을 알 수 있다.

4.4 인터리빙(interleaving)을 패리티 정보와 함께 사용하는 개선된 알고리즘 (API Algorithm)

이 방법은 컨벌루션 부호를 부호화하여 2개의 패리티 비트를 추가하는 것은 같지만, 부호화된 데이터를 인터리빙시켜 전송하게 한다. 따라서 집단적 오류가 채널상에서 발생하여 전송되는 데이터가 변형되더라도, 디인터리빙(deinterleaving)을 통해 집단적 오류를 4.2절의 알고리즘으로 교정이 가능한 형태의 오류로 만들 수 있다. 사실, 인터리빙은 집단적 오류를 피하기위해 가장 많이 사용되는 방법 중의 하나이다.[21] 그러나, 인터리빙만을 사용했을 때는 5장의 시뮬에이션 결과에서 보듯이, 산발적 오류에 대해서 정정능력이 떨어지고, 집단적 오류에 대해서도 개선된 알고리즘에 비해, 대처하는 능력이 낮음을 알 수 있다.

새롭게 제안하는 인터리빙과 패리티 정보를 함께 사용하는 API 알고리즘은 4.2 절의 알고리즘이 가지는 집단 오류 정정의 한계를 충분히 극복하고 있다.

4.4.1 개선되어진 부호화와 부호해석 과정의 전체적인 단계

Step 1. 컨벌루션 부호를 부호화하고, 패리티 2비트를 부가한다.
Step 2. 부호화기의 버퍼에 저장된 인터리빙된 부호열은 채널을 통해 전송된다.
Step 3. 수신 측에서는 수신되어 버퍼에 저장된 부호열을 디인터브시킨다.
Step 4. 원래의 컨벌루션 부호들과 함께 패리티 정보를 얻는다.
Step 5. 얻어진 컨벌루션 부호를 이용하여 API 알고리즘을 적용한다.

위의 단계들을 그림으로 나타내면 다음과 같이 나타낼 수 있다.

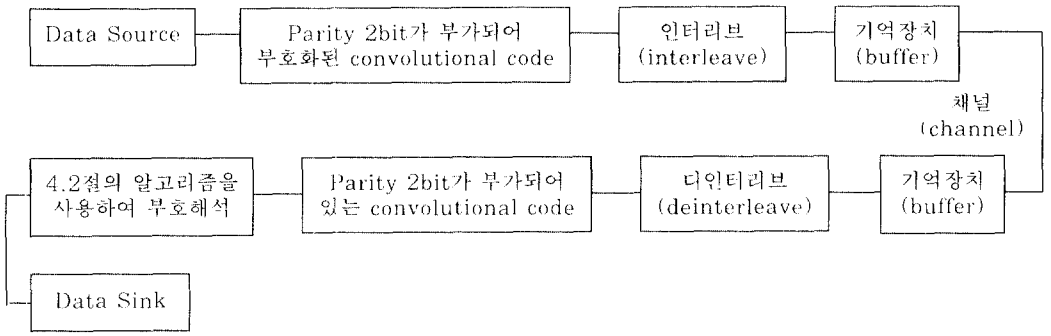


그림 6. 새로운 부호화방법과 부호해석 절차

그림 6의 채널을 통해 데이터를 수신한 이후 단계의 알고리즘을 좀더 자세히 나타내어보면 다음 그림 7과 같이 나타낼 수 있다. 인터리브된 데이터가 채널을 통해 들어오면, 수신 측에서는 디인터리브를 거친 뒤에 원래의 데이터와 패리티 정보를 분리한다. 그리고 새로 제안된 부호해석 알고리즘을 적용하는데, 이 단계에서는 수신 측의 부호해석 트리 정보와 mod-2연산을 행하여, 그 결과 나타난 1의 개수로 패리티 정보를 참조할 것인가 아닌가를 결정한다. 그림의 우측 빗금친 영역이 수신된 패리티 정보에 대해, 수신측이 이미 가지고 있는 패리티 정보와 비교하는 단계이다. 이 부분은 매번 부호해석시마다 수행되는 것이 아니고, 좌측의 데이터 비트에 대한 mod-2산 결과, 헤밍 거리값이 2이상을 나타내는 경우에 참조하게 된다.

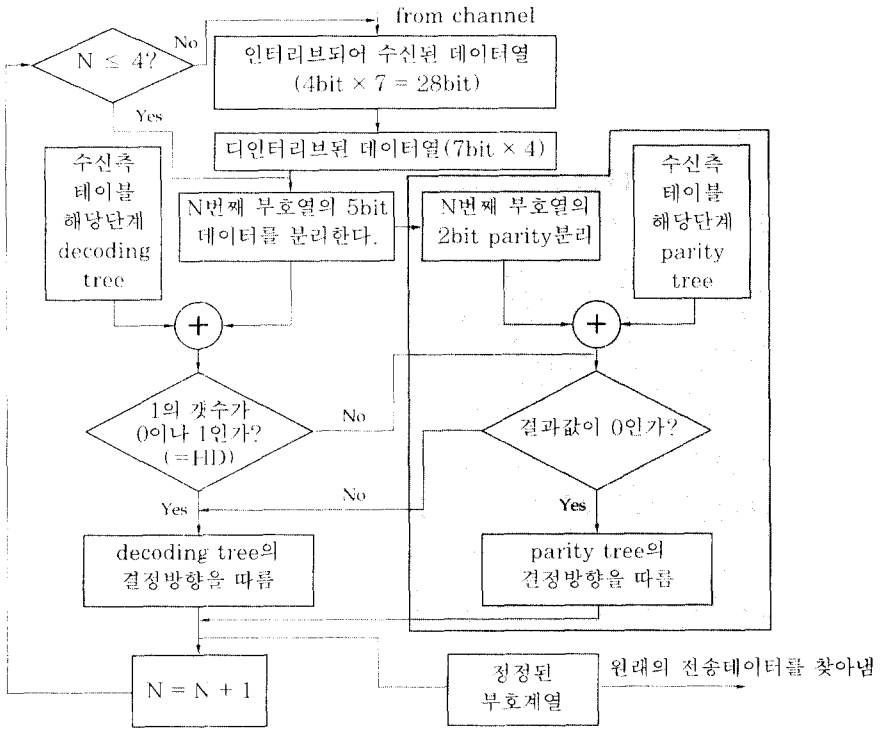


그림 7. API 부호해석 알고리즘

API(added parity & interleaving)방식은 그림 6에서 볼 수 있듯이 4.2절에서 제안한 패리티 정보를 부가 부호해석과정에서 이용하게 하는 방법보다 2단계를 더 거치게 된다. 즉, 전송 전에 부호화된 데이터를 인터리브하고, 채널을 통해 전송된 후에 디인터리브시켜 원래 부호화된 데이터의 형태로 복원한 후에 4.2 절의 알고리즘을 적용케 한다. 이렇게 함으로써 API는 채널상에서 발생되는 더 많은 집단 오류에 대해 정정할 수 있는 능력을 가지게 된다. 4.2절의 알고리즘만으로는 하나의 부호열 당 3개까지의 오류를 정정할 수 있었지만, API방식은 전체 부호열이 변형되는 경우는 물론, 최대 12비트의 집단 오류에 대해 정정할 수 있는 능력이 있다.

새롭게 제안된 알고리즘이 어떻게 적용되는지를 보기를 들어 설명하기로 한다.

API방법을 사용하여, 기존의 방법들로는 검출과 수정이 불가능했던, 전송되는 부호 열 전체에 집단오류가 발생한 경우를 새로운 알고리즘에 따라 부호화와 부호해석을 수행해보면 아래와 같다.

(보기 3) 전송할 데이터들이 패리티 비트 2개를 부가하여 다음과 같이 부호화되어 졌다고 가정하자.

$$Y = (1111111, 0101000, 0011110, 0001100)$$

(* 앞의 5비트가 데이터 비트, 뒤의 2비트가 패리티 비트 *)

ⓐ 인터리빙을 행하면 다음과 같다.

$$\overrightarrow{IY} = (1000, 1100, 1010, 1111, 1011, 1010, 1000)$$

인터리빙은 사실, 행렬(matrix)의 행(row)와 열(colum)을 바꾸어 놓은 것과 같다.

$$Y = \begin{vmatrix} 1111111 \\ 0101000 \\ 0011110 \\ 0001100 \end{vmatrix} \qquad \overrightarrow{IY} = \begin{vmatrix} 1000 \\ 1100 \\ 1010 \\ 1111 \\ 1011 \\ 1010 \\ 1000 \end{vmatrix}$$

※ 인터리빙 전후에 전체 부호열의 비트 수는 동일하다.

ⓑ 채널을 통해 $\overrightarrow{IY} = (1000, 1100, 1010, 1111, 1011, 1010, 1000)$ 전송하는 중에 다음과 같이 집단적 오류가 발생하였다면 다음과 같이 될 것이다.

$$S = (\boxtimes\boxtimes\boxtimes\boxtimes, 1100, 1010, 1111, 0\,0\boxtimes\boxtimes, \boxtimes\boxtimes\boxtimes\boxtimes, 1000)$$

(* ⊠ 는 오류가 발생한 것을 나타냄 *)

- 이 단계의 부호열에서 볼 수 있듯이, 전체 28개 비트 가운데 12비트의 오류가 발생하였고, 3개의 부호열 전체가 변형되었다.

ⓒ 따라서, 수신된 데이터는 다음과 같이 변형되었다고 하자.

$$\overrightarrow{ER} = (0111, 1100, 1010, 1111, 0100, 0101, 1000)$$

ⓓ 디인터리빙을 수행하면 다음과 같이 원래 형태로 돌아가지만, 원래의 Y와는 차이가 있다.

$Y = (1111111, 0101000, 0011110, 0001100)$ 원래의 데이터

$\overrightarrow{RY} = (0111001, 1101110, 1011000, 1001010)$ 수신된 데이터

오류가 발생한 위치

- 주의할 것은 집단적 오류가 발생한 $\overline{ER}$과 비교하여, 오류의 위치가 각 보호열에 고르게 퍼지게 함으로써, 한 부호열에 오류가 집중되는 것을 피할 수 있음을 알수 있다.

ⓒ 이제 3장의 부호해석 알고리즘을 적용하게되면, 다음과 같이 올바른 결과를 얻을 수가 있다.

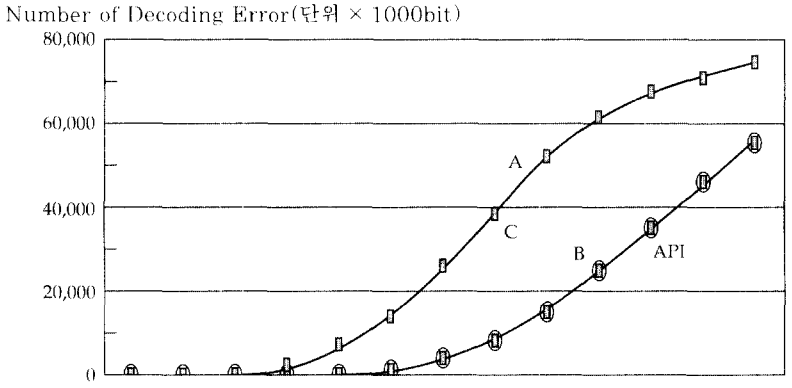
$$\dot{Y} = (11111,01010,00111,00011)$$

5. 시뮬에이션을 통한 각 오류정정 방식의 비교

본 장에서는 이제까지의 내용에서 제시했던 각각의 오류정정 방법들을 프로그램화하여, 웍스테이션상에서 시뮬레이션한 결과를 제시한다. 이 시뮬에이션은 SUN SPARC 2 웍스테이션에서 C언어를 사용하여 이루어졌다.

결과에서, 방법 A는 컨벌루션 부호를 최대 근사 부호해석 알고리즘만을 사용한 얻은 결과이며, 방법 B는 패리티 정보를 첨가하여 전송된 컨벌루션 부호에 대해 4.4절에서 제시한 개선된 알고리즘을 적용한 결과이다. 방법 C는 컨벌루션 부호를 인터리빙만을 시켜 전송하고, 디인터리빙을 거쳐 최대 근사 부호해석 과정을 거친 결과이다. 마지막으로 방법 API는 패리티 정보를 부가하여 인터리빙시켜 전송된 데이터를 디인터리빙하여 4.2절의 부호해석 알고리즘을 사용하는 형태이다.

테스트 1과 2는 10메가 바이트(Mbyte)의 이진 데이터를 전송하면서 산발오류를 임의의 위치에 발생시켜, 채널상의 오류가 증가함에 따라 수신측에서 바르게 부호해석하지 못하는 데이터의 증가를 보인 것이다. 그런데, 오류가 같은 위치에 두번 발생하게 되면 오류가 발생되지 않은 것과 같은 결과를 얻게 되므로, 임의의 위치의 한 비트가 오류로 변형된다면, 그 위치를 마스크(mask)하여 같은 위치에는 오류가 발생되지 않도록 프로그래밍하였다.



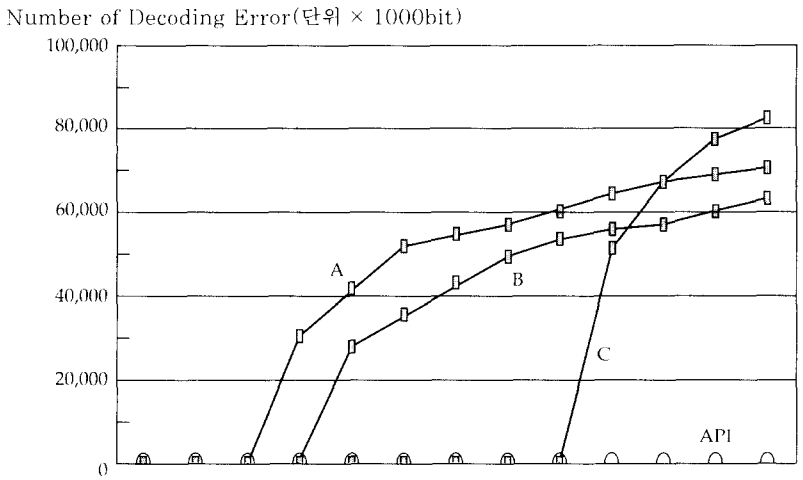
(단위 : bit)

Number of Errors	0	1	2	3	4	5	6	7	8	9	10	11	12
Method A ▮	0	0	0	1,832	6,264	14,619	26063	39,279	51,653	60,958	67,021	71,622	75,013
Method B ▮	0	0	0	0	388	1,576	4,428	8,795	15,650	24,625	34,637	45,680	54,599
Method C ⊟	0	0	0	1,823	6,643	14,863	26,591	39,270	51,822	61,090	67,211	71,518	74,849
API Method ⊖	0	0	0	0	402	1,580	4,468	8,878	15,653	24,663	34,987	45,848	54,888

그림 8. 테스트 1 - 산발오류에 대한 시뮬레이션 결과

위의 테스트 1의 결과를 분석하면, 오류의 형태가 산발적인 경우 방법 A, C보다 B와 API방법을 사용했을때 오류정정 능력이 우수함을 알 수 있다. 그러나, 오류의 개수가 증가하게되면 자연히 산발 오류가 집단적인 형태를 가지게 되므로, 각 방법들의 오류정정 능력이 많이 떨어지게 됨을 알 수 있다.

다음의 테스트 2는 10킬로바이트의 정보를 전송할 때, 오류의 형태가 집단적인 경우이다. 결과를 보면 오류가 한 부분에 집중하는 경우에는 예상한 바대로 인터리빙을 사용하는 방식인, 방법 C와 API가 좋은 결과를 나타내고 있다. 특히 API는 12개의 집단적 오류에 대해서도 완벽히 정정해낼 수 있음을 보여주고 있다. 인터리브만을 사용한 방법 C가 20비트당 8개까지 정정할 수 있음과 비교하여 볼때, API는 실제데이터 영역과 패리티 영역의 구분이 없이 전송되어지므로, 전체 28비트 가운데 데이터영역 20비트에만 12개의 오류가 발생한다고 하더라도 정정해낼 수 있는 능력을 가지고 있으므로, 실제 오류 정정 능력면에서 API방법이 방법C에 비해 더 나은 결과를 보인다.



(단위 : bit)

Number of Errors	0	1	2	3	4	5	6	7	8	9	10	11	12
Method A ▮	0	0	0	31,108	41,233	51,261	54,215	56,985	60,304	63,342	66,653	68,758	70,697
Method B ▮	0	0	0	0	29,378	36,625	43,734	51,155	53,411	55,674	58,008	60,213	62,385
Method C ⊟	0	0	0	0	0	0	0	0	0	51,129	66,555	76,799	81,920
API Method ⊖	0	0	0	0	0	0	0	0	0	0	0	0	0

그림 9. 테스트 2 - 집단 오류에 대한 시뮬레이션 결과

6. 결론과 앞으로의 연구방향

이 논문에서는 전송되는 데이터에 패리티 정보와 인터리브 기법을 도입하여, 컨벌루션 부호의 순방향 오류정정시에 심각한 제한 사항으로 존재하였던, 다량의 집단오류 발생에 대한 취약점을 보완하고 있다.

4.2절에서 제안한 알고리즘은 그 자체만으로도 사용할 수 있고, 하나의 부호열당 3개이하의 오류에 대해서는 우수한 성능을 보인다. 그러나, 더 많

은 집단 오류에 대처하기 위한 방법으로 4.4절에서 API방법을 제시하였다.

이 방법은 오류의 발생이 대부분 채널상에서 전송시에 생긴다는 점에 착안하여[1], 전송전에 부호를 인터리브시키고, 전송 받은 후 디인터리브시켜서 집단오류를 고르게 분포시켜 정정이 가능한 상태로 만드는 효과를 거둘수 있음을 보여 주었다. 전송 데이터 인더리브하는 방법은 집단적 오류를 피하기 위한 방법으로 많이 사용되는 방법 가운데 하나이지만, 단독으로 사용했을때보다 본 논문에서 제시한 패리티 정보를 병용하는 경우에 더 높은 효과를 얻을 수 있음을 시뮬레이션을 통한 결과에서 알수 있다.

사실, 28bit에 대해 12bit까지의 집단적 오류에 대해 처리할 수 있음은 인터리브방법을 사용하기 전후에 같다. 그러나, 채널에 발생하는 집단적 오류의 특성상, 4개이상의 오류가 한 부호열과 이웃한 부호열에 집중하지 않는다는 보장이 없다. 따라서, 전송되는 중에 하나의 부호열 전체가 변형되는 경우에도 API방법을 사용한다면 완전한 복구가 가능해진다.

또한 새로운 부호해석 알고리즘에서는, 이전에 제시하였던 2비트의 패리티를 사용하는 방식[9]에서 mod-2 연산기기가 추가로 필요했던 점을, 수식에 의한 간소화로 없애고 패리티 정보를 부호화과정에서 직접 얻고 있다. 따라서, 연산기의 증가로 인한 계산이나 장치의 복잡도(complexity)가 높아지는 문제를 피하고 있다. 또, 부호해석 단계에서도 경우에 따라서만 패리티 정보를 참조케 하여 패리티 정보의 추가에 따른 부호해석 단계에서의 부담을 줄이고 있다.

그러나, 이 방법은 오류의 정정 능력이 높은데 비하여, 부호열당 잉여정보(redundant information)의 수가 너무 많다는 부담을 가지고 있다. 이것은 통신 선로상의 트래픽(traffic)의 양의 증가를 가져올 수 있고, 단위시간 당 전송율이 떨어질수도 있다. 또, 인터리브와 디인터리브를 위한 버퍼(buffer), 그리고 패리티 정보를 가지는 추가의 기억장소가 필요로 하게 된다.

하지만, 부호화율이 낮은 것은 부호화단계에서 새로운 조화방법을 찾아냄으로써 어느 정도까지 줄일 수 있다.[3] 그리고, 추가되는 기억장치 문제 역시, 완전히 새롭게 추가되는 것이 아니라, 부호해석 단계에서 이미 어느 정도의 용량은 필요한 사항이므로, 하드웨어 가격의 하락에 비추어 비용등의 면에서 심각한 문제는 되지 못할 것으로 생각된다.

앞으로의 과제는, 이미 제시한 바대로 컴퓨터를 통한 계산으로 새로운 컨벌루션 부호의 조합 방법을 찾아내어 부호화율을 높이는 것이다. 또, 채널상의 오류발생 가능성을 감지하여 자동적으로 가장 적절한 오류 정정 방법을 통신 시스템이 선택하도록하여 불필요한 계산의 복잡성을 줄이는 방안도 기대할 수 있겠다.

참 고 문 헌

[1] Jiri Adamek, *Foundations of Coding*, pp 269-290, John Wiley and Sons, 1991

[2] Shu Lin, Daniel J. Costello, Jr. *Error Control Coding:Fundamentals and Applications*, pp 257-560, Prentice-Hall, 1983

[3] J. L. Massey, *Implementation of Burst-Correcting Convolutional Codes*, IEEE Trans. on Information Theory, July 1965

[4] W.W.Wu,*New Convolutional Codes-Part I*, IEEE Trans. on Communications, Vol. COM-23, No.9, Sep. 1975

[5] W.W.Wu,*New Convolutional Codes-Part II*, IEEE Trans. on Communica-tions, Vol. COM-24, No.1, Jan. 1976

[6] Harold S. Stone, *Spectrum of Incorrectly*

Decoded Bursts for Cyclic Burst Error Codes, IEEE Trans. on Information Theory, Vol. IT-17, No.6, Nov. 1971

[7] John P. Robinson, *Error Propagation and Definite Decoding of Convolutional Codes*, IEEE Trans. on Information Theory, Vol. IT-14, No.1, Jan. 1968

[8] D.W.Davies, *Computer networks and their protocols*, pp 257-270, John Wiley and Sons, 1979

[9] James. Martin, *Computer networks and distributed processing*, pp 464-489, Prentic-Hall, 1981

[10] T.Housley, *Data communications and teleprocessing system*, 2nd ed, pp 21-55, pp 178-199, Prentice-Hall, 1987

[11] R.W.Hamming, *Coding and information theory*, pp 20-50, Prentice-Hall, 1986

[12] L.Jordan, *Communications and networking for IBM PC and compati-bles*, 179-206, Brady, 1987

[13] J.Martin, *Data communication technology*, pp 275-286, Prentice-Hall, 1988

[14] W.Stallings, *Handbook of Computer communications*, Vol.2, pp 248-261, Howard W. Sams & Company, 1990.

[15] E.R.Berlekamp, *The application of error control to communications, IEEE Communications* Magazine Vol.25, No.4, pp 44-57, Apr. 1987

[16] A.J.Viterbi, *Error bounds for convolutional codes and an asymptotically optimum decoding algorithm*, IEEE Trans. Info. Theory, Vol.IT-1313, 60 - 369), Apr. 1967

[17] G.D.Froney, *Convolutional codes III: sequantial decoding*, Tech.Report 7004-1, Information Systems Lab.,Stanford Univ.

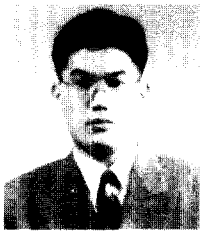
[18] A.J.Viterbi, and J. Omura, *Principles of digital communications and coding*, McGraw-Hill,1979

[19] 정인정, 최우영, *Convolution Code*의 집단 에러 교정을 위한 순차적 부호해석 방법의 개선, Journal of Institute of Science and Technology, pp 21- 31, 고려대학교 자연과학연구소, Dec, 1993

[20] 홍완표, *Artificial satellite and satellite commuications*, pp 353-412, Ohm, 1990

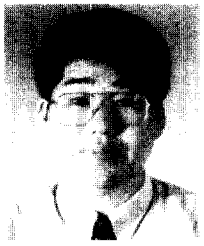
[21] 원동호 역, 宮川 洋, 今井秀樹, 原島 博, 情報と符號の理論,pp 241-271, Ohm, 1993

□ 著者紹介



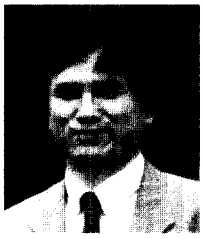
崔 佑 榮(學生會員)

1992年 2月 高麗大學校 電算學科 學士
1994年 8月 高麗大學校 大學院 電算學科 碩士
現在 : 金星 情報通信 硏究所 硏究員
主要 關心 分野 : 데이터 통신, 인공지능



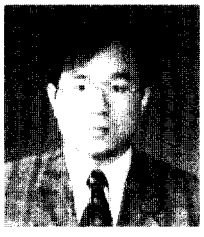
徐 彰 浩(學生會員)

1990年 2月 高麗大學校 數學科 學士
1992年 8月 高麗大學校 大學院 數學科 碩士
1993年 3月 高麗大學校 大學院 數學科 博士課程 在學中
主要 關心 分野 : 응용 대수학 및 정수론, 암호론



鄭 仁 禎(終身會員)

1978年 2月 서울 大學校 計算統計學科 學士
1980年 韓國科學技術院 電算學科 碩士
1980年 3月 - 1983年 2月 : 三星電子 컴퓨터 事業部 勤務
1983年 3月 - 1984年 8月 : 梨花女子大學校 電子計算學科 專任講師
1989年 12月 美國 Univ. of Iowa 大學校 卒業, 理學博士
1992年 3月 - 現在 : 高麗 大學校 電算學科 副敎授
主要 關心 分野 : 병렬처리, 인공지능



林 鐘 仁(正會員)

1980年 2月 高麗大學校 數學科 學士
1982年 2月 高麗大學校 大學院 數學科 碩士
1986年 2月 高麗大學校 大學院 數學科 理學博士
1986年 8月 - 現在 : 高麗大學校 數學科 敎授
主要 關心 分野 : 응용 대수학및 정수론, 암호론