이차원 코드를 위한 이차원 체크섬

안재원O
연세대학교 컴퓨터 과학과
chrisahnO@yonsei.ac.kr

Two-Dimensional Checksum for Two-Dimensional Codes

Jaewon AhnO
Dept. of Computer Science, Yonsei University

요 약

체크섬은 에러검출의 우수성 때문에 TCP등에서 널리 사용하고 있다. 이 체크섬 방식은 연산이 간단하고 빠르나, 미검출 에러가 생기는 경우가 있기 때문에 여러 다른 형태의 체크섬 방식등이 연구가 진행되고 있고 각 방식에 성능과 효율면에서 상관관계가 있어 어플리케이션에따라 선택에 어려움이 있는 경우가 있다. 이에 저자는 새로운 개념인 이차원 체크섬이라는 방식을 제안하고자 한다. 알고리즘이 단순하면서도 기존의 체크섬의 미검출 에러의 문제를 현저하게 줄일 수 있고 비용-이득 스펙트럼상에서 매우 효과적이고 효율적이다.

1. 서 론

기본적인 패리티 방식부터 체크섬과 CRC (Cyclic Redundancy Check) 방식등 다양한 에러검출 방식이 사용되고 있다. 이중 CRC는 강력한 에러 검출코드로서 사용되고 있지만, 빠른 소프트웨어 기법으로 구현될 필요가 있을 경우엔 non-CRC 방식인 패리티나 체크섬이 우선적으로 고려된다[4],[5],[6]. 특히, 통신의 TCP (Transmission Control Protocol) 에서 1의 보수 체크섬을 사용하고 있다. 이 체크섬 방식은 기본적으로 우수하나 특정한 경우의 미검출 에러의 경우 때문에 이를 보완하기 위해 여러 방법등이 고안되었으나 그러한 미검출에 개선에 제한적이다. 이에 저자는 새로운 개념인 이차원 체크섬이라는 방식을 제안하고자 한다. 알고리즘이 단순하면서도 기존의 체크섬의 미검출 에러의 문제를 현저하게 줄일 수 있는 것이 큰 장점이다.

2. 체크섬의 표현

체크섬 코드는 집합 Z_q (정수들 mod q) 로부터 (n+1) 심볼들의 벡터 집합이다[1]. 각 벡터는 자기를 제외한 다른 구성요소를 mod q 합인 체크 심볼과 정보 심볼이라는 부분으로 이루어진다.

$$\{(x_c, x_{n-1}, \ldots, x_0) \mid (x_i, x_c \in z_q), and (x_c = \sum_{0 \le i \le n-1} x_i \bmod q)\} \quad (1)$$

위 (1)식에 의해 체크섬 코드의 최소 해밍거리는 2임을 알 수 있다. 여기서 1보다 큰 b가 있을 때 q가 2^b와 같은 경우가 특별히 관심이 있다. 그리고, 일반적인 통신 인터페이스는 1바이트 (b = 8) 단위로 구성된다. 이 논문에서 언급하는 코드는 $Z_2{}^b$ 으로 부터의 각 심볼들은 b 비트로서 인코딩 된다. 따라서 각 코드워드는 $n \cdot b$ 정보 비트와 b 체크 비트들로 이루어진다. 여기서 에러는 b-비트 바이트 만큼 검출할 수 있다.

자 여기서 체크섬 코드로 두 입력벡터를 아래와 같은 형태로 된다.

$$A = (a_c, a_{n-1}, a_{n-2}, \ldots, a_0)$$
$$B = (b_c, b_{n-1}, b_{n-2}, \ldots, b_0) \quad (2)$$

위에서 a_c와 b_c는 체크심볼이고 나머지는 정보부분이다.

$$A_d \equiv (a_{n-1}, a_{n-2}, \ldots, a_0)$$
$$B_d \equiv (b_{n-1}, b_{n-2}, \ldots, b_0) \quad (3)$$

정보 부분 A_d와 B_d 다음과 같은 $[A_d]$와 $[B_d]$정수로 나타낼 수 있다.

$$[A_d] = \sum_{0 \le i \le n-1} a_i 2^{bi}$$
$$[B_d] = \sum_{0 \le i \le n-1} b_i 2^{bi} \quad (4)$$

위 식들에서 2의 보수 연산으로는 M은 2^b와 같고, 1의 보수연산에선 M은 2^b -1와 같다. 그리고 전자의 경우 b-비트를 넘어서는 오버플로우는 버려지고, 후자의 경우엔 " end-around" 로 캐리로 더해진다.

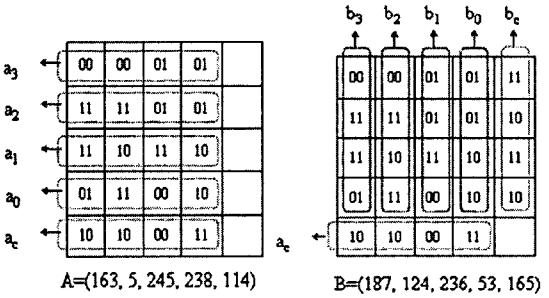
2.1 1의 보수 체크섬

미검출 에러 확률은 b가 체크섬 비트일 때 2^{-b} 로 정의된다[2]. 즉, 체크섬 비트가 8이면 1/256 만큼, 16비트를 사용하면 1/65536 만큼의 미검출 에러가 생긴다. 예를 들어 (2)식에서 n=4, b=8 이라고 가정하고 A=(163, 5, 245, 238, 114)을 살펴보면 (여기서 체크섬 163은 정보부분 5, 245, 238, 114를 1의 보수법으로 계산한 결과) 전송 중에 어떠한 심볼이 오류가 생겨도, 체크섬과 나머지 정보부분 연산의 결과와 같지 않게 되므로 1바이트 에러검출 코드가 된다[1]. 또한, 한 심볼 이상이 바뀌어도 에러를 검출할 수 있는 경우 경우가 있긴 하지만 (정보부분의 1의 보수 덧셈과 체크섬 정보부분의 비교의 의해) 만일, 수신측에서 받은 신호가 A' =(163, 10, 240, 238, 114) 이나 A'' =(163, 20, 230, 222, 130) 같은 경우처럼, 명시적으로 데이터가 에러가 발생 되었음에도 불구하고 우연히 데이터의 1의 보수합과 체크섬이 같은 경우에는 에러검출이 안 된다. 그래서, 1의 보수 체크섬을 사용하는 인터넷 체크섬처럼 체크섬 비트를 8비트 대신 16비트를 쓰면 미검출 에러확률이 줄어드는데 이는 체크섬의 성능을 향상시키는 방법중 체크섬 비트의 증가가 그 한 방법이기 때문이다[3].

2.2 새로운 접근방식으로서의 이차원 체크섬

앞에서 언급한 바와 같이 기존의 1의 보수 체크섬 방식 미검출 에러확률로 인해 인터넷 체크섬은 16비트로 늘려 미검출 에러확률을 줄이고 있지만, 여전히 어플리케이션에 따라 문제가 될 수도 있다. 다른 대안으로는 1의 보수 체크섬을 응용한 다른 형태로는 Fletcher 체크섬[4] 방식이 있다.

이러한 배경에서, 기존에 방식과는 새로운 발상으로 새로운 개념의 이차원 체크섬이라는 새로운 개념의 에러검출 방식을 살펴보고, 그의 장점을 살펴보고자 한다. 이 방식은 특징은 일단, 전체 벡터를 이차원적은 블록으로 가정한다. 앞의 예 A=(163, 5, 245, 238, 114)를 가지고 표현하면,



(a) (b)

그림 1. 2차원 체크섬의 예

즉, 보내려하는 정보부분이 A_d=(5, 245, 238, 114)일 때, 인코딩 방식은, 기본적으로 정수형 데이터를 데이터의 이진수형비트 클러스터로 표현할 수 있다는 점에서 출발한다. 혹은 정보부분이 처음부터 이진수 비트형 데이터일때도 당연히 적용할 수 있다.

- Step #1: 1의 보수 체크섬 방식으로 그림 1(a)처럼 수평 크섬 a_c를 구한다. 여기서 그림처럼 이진수 스트림이 수평적으로 묶여 있는 것처럼 가정한다.
- Step #2: 같은 정보부분을 이진수 비트스트림을 그림 1(b)처럼 묶는다면, 가상 정보부분 B_d =(124, 236, 53, 165) 를 얻을 수 있고 이를 통해 가상 체크섬인 b_c를 구할 수 있다. 여기서는 이진수 스트림이 수직적으로 묶여 있는 것처럼 가정한다.

2.3 일차원 체크섬과 이차원 체크섬의 비교

기존의 인터넷 체크섬과 같은 1의 보수 체크섬은 기학적인 모델로 보면 체크섬이 그림1(a)같은 1차원적인 구성으로 존재한다. 한 심볼 내에서 비트가 바뀌면 에러가 그 한 심볼에서만 영향을 미치므로, 수평적인 그림1(a)처럼 8비트 체크섬 방식은 1바이트 에러검출이 가능하고, 16비

트형인 인터넷 체크섬은 마찬가지고 계산하면 2바이트 에러검출이 가능하다. 그런 현상은 기하학적인 모델에서 보면 수평적인 방향의 에러패턴에서 에러검출이 이루어짐을 알 수 있고, 미검출 에러가 생기는 경우는 수직적으로 생길때, 물론 항상은 아니지만, 발생될 가능성이 매우 높다. 2.1절의 예를 들어 설명을 하면 A' 과 A'' 같은 경우엔 기존 1차원 체크섬 방식으론 에러검출이 되지 않지만, 2차원 체크섬 방식에선 B' =(187, 124, 236, 50, 162) 가 에러를 검출할 수 있고, B'' =(187, 188, 25, 53, 178) 로 에러검출이 가능하다.

3. 이차원 체크섬 알고리즘

여기서는 위에서 제시한 새로운 알고리즘을 인코딩/디코딩을 통해 구현하는 방식을 제시한다.

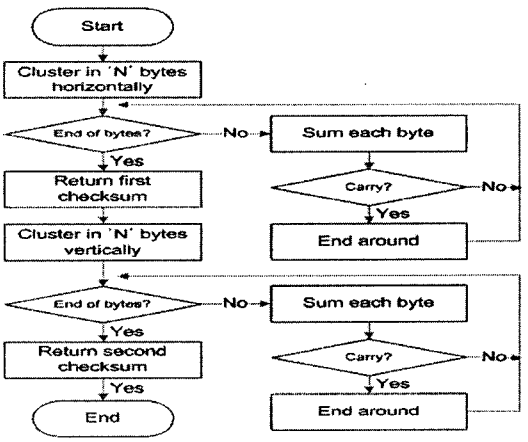


그림 2. 2차원 체크섬의 인코딩방법

인코팅 플로우차트에서 앞부분은 일반적인 1의 보수 체크섬 방식과 동일하나 비트를 수직적으로 다시 묶어서 (심볼크기로 여기서는 바이트) 다른 하나의 체크섬을 발생시킨다. 디코딩시에는 그런 인코딩의 역순으로 두개의 체크섬 심볼과 정보부분을 비교하면 정보부분에 에러가 있는지를 판별할 수 있고 에러가 있으면 송신측에 재전송을 요청하면된다.

4. 결론 및 고찰

인터넷 체크섬과 일반적인 형태의 1의 보수 체크섬은 각 심볼을 이진수로 변환시키면 마치 2차원적인 매트릭스 형태의 데이터가 되는데, 이때 수평축의 에러는 한 심볼에만 영향을 미치기 때문에 (2.3절 예 참조) 미검출 에러가 생

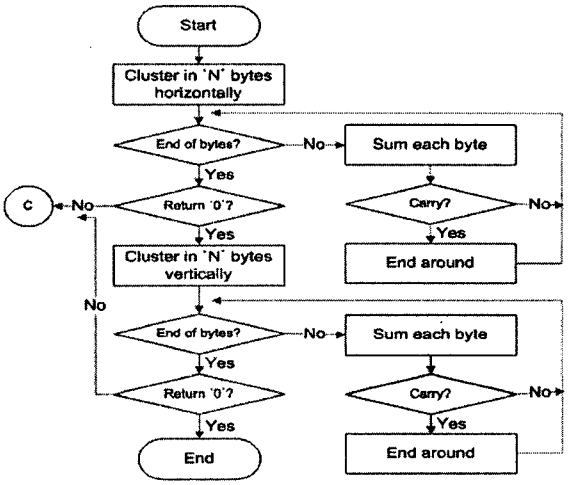


그림 3. 2차원 체크섬의 디코딩방법

기지 않는다. 하지만, 수직축으로는, 비트 반전이 다른 심볼과 균형이 이루어지는 경우엔 미검출 에러가 생길 수 있다. 이 점이 1의보수 체크섬의 약점인데, 기하적으로 1차원인 체크섬을 2차원으로 확장시켜 수직적인 가상 체크섬 (같은 데이터를 묶는 방식에 따라 다른 가상심볼을 통한 다른 체크섬 값을 얻음) 을 갖기 때문에 수직적으로 미검출 에러확률의 경우의 수를 현저하기 줄일 수 있다.

참고문헌

[1] T.R.N. Rao and E. Fujiwara, "Error-Control Coding for Computer Systems,"Prentice Hall, pp. 418-420, 1989.

[2] R. Braden, D. Borman, and C. Partridge, "Computing the Internet Checksum," ACM Computer Communication Review, 19, no. 2, Sept. 1988 (Internet RFC 1071).

[3] R. N. Williams, "A Painless Guide to CRC Error Detection Algorithm," Rocksoft Pty Ltd., 1993.

[4] J.G. Fletcher, "An Arithmetic Checksum for Serial Transmissions,"*IEEE Trans. Comm.*, vol. 30, pp. 247-252, Jan. 1982.

[5] D.C. Feldmeier, "Fast Software Implementation of Error Detection Codes," *IEEE/ACM Trans. Networking*, vol. 3, no. 6, pp. 640-651, Dec. 1995.

[6] G.D. Nguyen, "Error-Detection Codes: Algorithms and Fast Implementation,"*IEEE Trans. Computers*, vol. 54, no.1, pp. 1-11, Jan. 2005.