논문 15-40-09-14 The Journal of Korean Institute of Communications and Information Sciences '15-09 Vol.40 No.09
http://dx.doi.org/10.7840/kics.2015.40.9.1741

부분접속 복구 가능한 반복분할 부호

남 미 영◆, 김 정 현*, 송 홍 엽°

Locally Repairable Fractional Repetition Codes

Mi-Young Nam◆, Jung-Hyun Kim*, Hong-Yeop Song°

요 약

본 논문에서는 MBR 재생부호인 반복분할 부호의 부분접속수를 향상시킬 수 있는 방법에 대해 소개한다. 향상된 부분접속수를 갖는 반복분할 부호를 부분접속 복구 가능한 반복분할 부호라고 한다. 부분접속 복구 가능한 반복분할 부호의 서로 다른 두 가지 생성 방법을 소개하고 각각을 다양한 성능 척도를 통해 분석한다. 새로운 부호는 반복분할 부호에 비해 낮은 부분접속수를 갖는 대신 저장 가능한 최대 파일 크기나 필요한 저장 노드의 수에서 손해가 발생한다. 다른 부분접속 복구 부호와 비교해 향상된 안정성을 갖고 또한 복구시 심벌 연산이 필요 없는 단순전달 복구를 수행함으로써 복구 복잡도를 낮출 수 있다.

Key Words : Regenerating codes, MBR codes, Fractional Repetition codes, Locality, Locally repairable codes

ABSTRACT

In this paper, we introduce new locally repairable codes based on a Fractional repetition codes which is one of the MBR codes. We introduce two different constructions for different system parameters and compare these codes in terms of several performance metrics. There is some tradeoffs between the locality and other performance metrics. The newly introduced codes having the good locality should pay the price such as lower capacity or more storage nodes. And the proposed codes are more reliable than other locally repairable codes and have lower repair complexity since they can be repaired without any operations.

Ⅰ. 서 론

인터넷과 스마트폰 사용자의 폭발적인 증가, 그리고 소셜 네트워크 서비스(SNS)의 확산으로 빅 데이터(Big data) 시대가 열렸다. 전 세계 9억 5,000명이 가입한 것으로 알려진 페이스북은 SNS의 대표적인 예로서, 매월 평균 3억 장의 이미지 파일, 25억 개의 콘텐츠가 생성된다[4]. 이러한 생성 규모와 생성 속도를 특성으로 하는 빅 데이터는 기존의 저장 및 분석 방식을 통해 관리하는 것이 불가능하다. 빅 데이터의 효과적인 저장 관리 기술로 분산 저장 시스템이 주목받고 있다[3]. 분산 저장 시스템은 네트워크를 통해 연결된 다수의 노드들에 데이터를 분산 저장한다. 각 노드의 물리적 불완전성이나 네트워크의 불안정한 연결 상태 등으로 인해, 해당 노드에 저장되어 있는 데이터의 소실이 발생할 수 있다. 이러한 노드 장애로 인해 데이터가 영구적으로 소실되는 것을 막을 수 있는 안정적인 저장 방법이 필요하다. 가장 간단한 방법으로, 데

※ This research was supported by Basic Science Research Program through the National Research Foundation of Korea(NRF) funded by the Ministry of Education(No. 2013R1A1A2062061)
◆ First Author : School of Electrical and Electronic Engineering, Yonsei University, my.nam@yonsei.ac.kr, 학생회원
° Corresponding Author : School of Electrical and Electronic Engineering, Yonsei University, hysong@yonsei.ac.kr, 종신회원
* School of Electrical and Electronic Engineering, Yonsei University, jh.kim06@yonsei.ac.kr, 학생회원
논문번호 : KICS2015-04-135 Received April 21, 2015; Revised July 15, 2015; Accepted September 10, 2015

이터를 중복해서 저장하거나 좀 더 복잡하지만 효율적인 소실 부호(erasure code)를 적용하는 기법 등이 존재한다[2,5]. 이들은 모두 전통적인 오류정정부호를 적용한 것이다.

분산 저장 시스템은 기존의 오류정정부호가 적용되었던 통신 시스템과는 큰 차이점을 갖는다. 분산 저장 시스템에서는 안정성을 일정하게 유지하기 위해 장애가 발생한 노드의 복구가 수시로 이루어져야 한다. 이는 기존의 통신 시스템에 적용되는 오류정정부호가 부호화된 전체 심벌들로부터 원본 데이터를 복호하는 과정만을 고려했던 것과 큰 차이를 갖는다. 분산 저장 시스템에서는 소실된 일부의 심벌을 복구하는 과정이 수시로 이루어져야하기 때문에 낮은 복구 복잡도와 적은 복구 대역폭(repair bandwidth)이 필수적으로 요구된다. 복구 대역폭은 하나의 노드 장애로 인해 일부 부호화 심벌이 소실된 경우 해당 심벌의 복구를 위해 다른 노드로부터 받아와야 하는 심벌의 총 양으로, 분산 저장시스템의 전체 통신량의 상당 부분을 차지한다[13]. 따라서 복구 대역폭을 줄이는 것이 분산 저장 시스템에서 해결해야 할 중요한 문제가 된다. 이러한 복구 대역폭을 최적화하는 부호로 재생 부호(Regenerating code)가 소개되었다[7][11]. 재생 부호에서는 복구 대역폭과 한 노드의 저장용량 사이에 트레이드오프가 존재함이 잘 알려져 있다[7]. 이 트레이드오프 곡선의 양 끝단에 해당하는 부호를 각각 최소 저장공간 재생부호(minimum storage regenerating code; MSR code)와 최소 대역폭 재생부호(minimum repair-bandwidth regenerating code; MBR code)라고 한다. 이 두 영역에서의 부호의 설계 및 분석에 대한 연구가 주로 진행되어 왔다[8,10]. 반복분할(Fractional repetition; FR) 부호는 복구 과정에서 추가의 부호화 과정이 필요 없고 동일한 복구(exact repair)[12]를 할 수 있어, 복잡도가 낮은 특성을 갖는 MBR 부호로서 소개되었다[6].

최근에는 이러한 복구 대역폭 외에도 복구 성능의 척도로 부분접속수(locality)가 많이 연구되고 있다[1,9]. 부분접속수는 하나의 노드를 복구하는데 요구되는 최소 접속 노드 수를 의미한다. 일반적으로 작은 부분접속수를 갖는 부호를 부분접속 복구 부호(Locally Repairable Code; LRC)라고 한다.

본 논문에서는 MBR 부호로 제안된 반복분할 부호를 부분접속수 측면에서 분석한다. 이를 통해 재생 부호로서의 좋은 특성을 갖는 반복분할 부호의 부분접속수를 향상시킬 수 있는 두 가지 방법을 제안한다. 새로이 제안된 향상된 부분접속수를 갖는 반복분할 부호를 여러 성능척도를 통해 기존의 반복분할 부호 및 부분접속 복구 부호와 비교한다.

본 논문의 구조는 다음과 같다. Ⅱ장에서는 본 논문에서 사용할 시스템 모델 및 반복분할 부호에 대해 소개하고 이 부호의 부분접속수를 향상시키기 위한 새로운 생성방법을 제안한다. 제안한 새로운 부호를 부분접속 복구 가능한 반복분할 부호(locally repairable FR codes)로 명명하고 이 부호의 성능을 기존의 반복분할 부호 및 다른 LRC와 비교한다. Ⅲ장에서 결론을 내림으로써 본고를 마친다.

Ⅱ. 본 론

2.1 시스템 모델 및 반복분할 부호의 정의

본 논문에서는 (n,K,d,α) 분산 저장 시스템에 크기가 M인 파일을 저장하는 상황을 가정한다. 이는 n개의 노드를 갖고, 각 노드의 저장 공간의 크기가 α, 한 노드를 복구하는데 접속해야 하는 다른 노드의 수가 d인 시스템으로, 최종 사용자는 전체 n개 중 임의의 K개의 노드에 접속하여 받아온 데이터를 통해 원래의 파일을 얻을 수 있는 시스템이다. 이때 사용되는 부호에 따라 부분접속수 d가 달라진다. 일반적인 재생 부호에서는 임의의 d개에 접속해서 복구가 성공적으로 이루어지는 상황을 고려하므로, $d \geq K$를 만족해야만 한다[7]. 보통의 시스템에서는 복구 과정에서 하나의 노드로부터 받아오는 데이터의 양은 모두 β로 동일하다고 가정한다. 이때 복구 대역폭은 $\gamma = d\beta$로 표현할 수 있다.

복구 대역폭을 최소화 하는 MBR 부호를 이용해 저장할 수 있는 최대 용량은 접속하는 노드의 수 K에 대한 식으로 다음과 같다[7].

$$M(K) = \sum_{i=0}^{K-1}(d-i)\beta = \left(dK - \binom{K}{2}\right)\beta \quad (1)$$

[6]에서는 연산 없이 노드 복구가 가능한 재생 부호로, 반복분할 부호가 소개 되었다. 일반적인 반복분할 부호를 정의하고, 동시에 식 (1)의 용량을 달성하여 MBR 부호가 되는 구체적인 생성법도 함께 소개하였다[6].

반복분할 부호의 구조는 그림 1과 같다. 그림 1의 부호는 MDS 부호를 통해 부호화된 부호화 심벌들에 반복분할이라는 후처리 과정을 연접한 형태이다. 일반

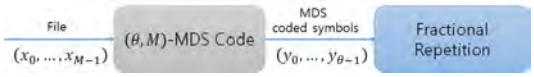


그림 1. 반복분할에 기반 한 MBR 부호의 구조
Fig. 1. A structure of an MBR code based on fractional repetition

적으로 반복분할은 θ개의 심벌을 ρ회 반복하여 생성된 $\rho\theta$개의 심벌들을 n개의 노드에 적절히 분산시키는 과정을 의미한다. 그림 2는 $\theta=6,\rho=3,n=7$인 경우의 반복분할의 예이다. 그림 2에서 $n=7$개의 노드에 저장되는 각 심벌들을 편의상 해당 심벌의 인덱스로 표현하였다.

정의 1.[6] (n,K,d,α) 분산 저장 시스템에서 각각의 크기가 α인 $\Omega=\{0,\ldots,\theta-1\}$의 n개의 부분집합 $V_0,\ldots,V_{n-1}$들의 컬렉션으로 Ω의 각 원소가 컬렉션 내의 ρ개의 집합에 속하는 조건을 만족하는 컬렉션을 반복도 ρ를 갖는 Fractional repetition(FR) 이라고 한다. (θ,M) MDS 부호와 이러한 FR을 연접한 전체 부호를 반복분할 부호라고 한다.

반복분할 부호 C가 주어졌을 때 이 부호 C의 용량은 부호 C를 이용해 저장할 수 있는 최대 파일의 크기로 정의한다[6]. 이때 파일을 얻고자 하는 사용자는 임의의 K개의 노드에 접속하는 것을 가정한다. 반복분할 부호에서 외부호(outer code)로 MDS 부호를 사용하므로, 반복분할 부호의 용량은 임의의 K개의 노드로부터 받아올 수 있는 서로 다른 심벌의 개수가 된다. 반복분할 부호가 소개된 이후, 식 (1)을 달성할 수 있는 다양한 생성방법들이 소개되었다[6,14-16]. 반복분할 부호는 기존의 MBR 부호가 만족해야 하는 제약사항보다 완화된 제약사항을 가정한다. 이로 인해 반복분할 부호의 최대 용량의 상한은 식 (1)의 MBR 용량보다 커질 수도 있다[15]. MBR 부호는 하나의 노드

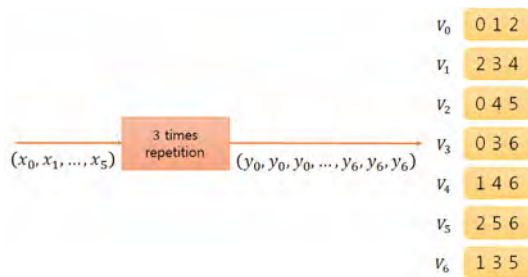


그림 2. (7,3,3,3) 분산 저장 시스템에서의 반복분할
Fig. 2. Fractional repetition for (7,3,3,3) DSS

를 복구하는 과정에서 나머지 $n-1$개 노드들 중 임의의 d개의 노드로부터 항상 해당 노드의 복구가 가능해야 한다고 가정한다. 즉 모든 노드는 크기가 d인 $\binom{n-1}{d}$개의 서로 다른 노드 집합으로부터 항상 복구될 수 있다. 반면 반복분할 부호에서는 한 노드의 복구를 돕는 특정 d개 노드 집합이 한 개 이상만 존재하면 된다고 가정한다. 이러한 복구 과정을 테이블 기반 복구(table-based repair)라고 한다[6]. 이러한 관점에서 반복분할 부호는 MBR 부호에서 요구되는 랜덤 복구의 특성을 만족시키지 못하기 때문에 엄밀히 말해서는 MBR 부호라고 할 수 없지만, 실제 시스템에 적용되어 사용될 때는 대부분 테이블 기반 복구를 주로 사용하므로 엄밀한 구분 없이, 용량이 식 (1)을 달성하는 반복분할 부호는 MBR 부호로 간주한다. 즉, 반복분할 부호는 그 구체적 생성방법에 따라 MBR의 용량보다 작을 수도 있고 MBR과 동일한 용량을 가질 수도 있고, MBR 용량보다 큰 용량을 가질 수도 있다. MBR 부호로 사용하기 위해 MBR 부호의 용량보다 크거나 같은 용량을 갖는 반복분할 부호를 생성할 수 있는 방법에 대한 연구가 중요하게 이루어지고 있다[6,14,16]. 더욱이 [15]에서는 반복분할 부호 용량의 상한계와 이를 달성하는 반복분할 부호의 생성에 대해 다루고 있다.

[6]에서는 반복분할 부호를 생성하는 두 가지 방법을 제안한다. 두 가지 생성법은 각각 $\rho=2$일 때 정규(regular) 그래프를 이용하는 방법과 $\rho>2$일 때 steiner system을 이용하는 방법이다. 두 방법은 모두 전체 n개의 노드 중 모든 두 노드들 사이에 최대 한 개의 심벌만을 공통으로 저장하도록 함으로써 생성된 부호가 식 (1)의 MBR 용량을 달성하는 반복분할 부호가 되도록 한다. 모든 두 노드 사이에 최대 한 개의 심벌만 공유하도록 하는 요구조건은 조합론에서의 디자인[17] 문제와 밀접한 관련이 있고 이러한 디자인을 적용하여 반복분할 부호를 설계하는 다양한 연구가 존재한다[14-16]. 이렇게 Steiner system을 포함하는 다양한 블록 디자인을 활용하는 부호의 생성은 각각의 디자인이 존재하는 파라미터가 한정적이다. 따라서 이를 통해 생성할 수 있는 부호의 파라미터가 제한적이라는 문제를 갖는다.

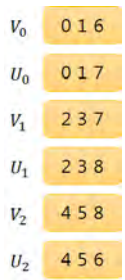
기존의 반복분할 부호에서 모든 두 개의 노드에 공통으로 저장될 수 있는 심벌의 개수를 최대 하나로 제한하기 때문에 한 노드에 발생한 장애를 복구하는 복구 과정에서의 부분접속수는 항상 해당 노드의 저장용량 α와 같다. 따라서 한 노드에 저장하는 심벌의

수 α가 증가하면 부분접속수 역시 커지게 된다. 본 논문에서는 한 쌍의 노드가 공통으로 저장할 수 있는 심벌의 최대 개수를 증가시켜 반복분할 부호의 부분접속수를 감소시키는 방식을 제안한다. 즉, α보다 작은 값의 주어진 부분접속수를 갖도록 하는 반복분할 부호의 생성방법을 제안한다.

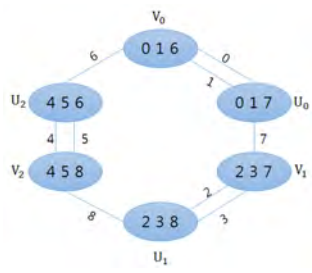
2.2 향상된 부분접속수를 갖는 반복분할 부호 제안

분산 저장 시스템에서 중요한 성능 척도 중 하나인 부분접속수가 작은 부호의 생성에 대한 연구가 활발히 이루어지고 있다[18-22]. 이러한 부호들은 대부분 복구 과정에서 복구를 수행하는 노드나 복구를 도와주는 헬퍼 노드들에서 심벌의 연산이 필수적이다. 이것이 복구 과정의 연산 복잡도를 증가시킨다. 이에 반해 반복분할 부호에서의 노드 복구는 단순 전달 복구(repair-by-transfer)[23]로서 d개의 헬퍼 노드는 저장된 심벌 중 일부를 바로 전달하고 복구를 수행하는 노드에서도 역시 전달 받은 심벌들을 그대로 저장하는 것만으로 복구가 가능하다. 본 논문에서는 이러한 반복분할 부호의 단순한 복구 과정을 그대로 유지할 수 있도록 반복분할 부호에 기반 한 부분접속 복구 부호를 제안한다.

기존의 반복분할 부호는 임의의 두 노드 사이에 공유되는 심벌이 존재하지 않거나 오직 한 개의 심벌을 공유하는 경우만 존재하였다. 여기서는 임의의 두 노드 사이에 공유되는 심벌의 개수가 더 다양한 경우를 고려한다. 이는 앞서 언급한 일반적인 경우와 달리, 각 노드 사이의 β가 모두 서로 다른 경우를 허용한다. 본 논문에서는 편의상 가장 간단한 $\alpha=3$인 경우만을 다룬다.



(a) symbols assignment



(b) graph representation of a code

그림 3. $\alpha=3, \rho=2$를 갖는 부분접속수 2인 반복분할 부호의 예
Fig. 3. An example of FR codes with locality 2 when $\alpha=3$ and $\rho=2$

정의 2. (n, K, d, α) 반복분할 부호 중, 부분접속수가 $d<\alpha$를 만족하는 부호를 부분접속 복구 가능한 반복분할 부호라고 한다.

그림 3은 본 논문에서 제안하는 부분접속 복구 가능한 반복분할 부호의 예이다. 제안하는 부호를 그래프로 표현한 그림 3(b)에서 각 노드는 분산 저장 시스템의 저장 노드에 해당한다. 이 예에서는 노드의 수가 $n=6$이다. 그래프 표현에서 노드는 V_i와 $U_i, i=0,1,2$의 두 그룹으로 나뉜다. 같은 그룹에 속하는 두 개의 V_i, V_j를 연결하는 에지나 U_i, U_j를 연결하는 에지는 존재하지 않는다. 그리고 각 V_i와 U_j, $0 \le i, j \le 2$, 사이에는 에지가 연결되지 않거나, 하나의 에지가 연결되거나 혹은 두 개의 에지가 연결되는 세 가지 경우가 가능하다. 그림 3에서는 동일한 인덱스 i를 갖는 두 노드 V_i와 U_i는 두 개의 에지로 연결되고 두 노드 $U_i, V_{(i+1)_3}$은 한 개의 에지로 연결된다. 본 논문에서 $(X)_m$은 $X \pmod m$을 의미한다. 이 그래프의 에지에는 0부터 8까지 서로 다른 값이 표기되어 있다. 이는 심벌의 인덱스를 의미하는 숫자로, 에지가 입사하는 양 끝 노드에 이에 해당하는 심벌들이 저장되는 것을 의미한다. 하나의 에지가 입사하는 노드는 두 개 존재하므로, 모든 심벌은 두 번 반복되어 나타나게 된다. 하나의 노드 장애를 복구하기 위해서는 해당 노드에 에지로 연결된 이웃노드에 모두 접속하여 에지에 해당하는 심벌을 받아오면 된다. 예를 들어, 노드 V_0을 복구하는 경우, 두 개의 이웃 노드 U_2와 U_0으로부터 노드 V_0에 연결된 에지에 해당하는 심벌 $\{6\}$과 $\{0,1\}$을 각 노드로부터 받아온다. 그림 3(b)의 그래프는 다양한 크기로 확장될 수 있다. 이를 통해 생성된 부호는 부분접속수가 $d=2$이고 반복도가 $\rho=2$인 반복분할 부호가 된다.

이러한 형태와 특성을 갖는 반복분할 부호의 일반적인 생성 방법은 다음과 같이 정리할 수 있다.

생성법 1. (θ, M) MDS 부호를 통해 생성된 부호화 심벌의 집합을 $\Omega=\{0,...,\theta-1\}$라고 하자. θ개의 심벌은 다시 반복도 $\rho=2$인 반복 부호를 통해 부호화 된다. 생성된 $\rho\theta$개의 심벌을 용량이 $\alpha=3$인 $n=2m$개의 노드 $V_0,...,V_{m-1}$과 $U_0,...,U_{m-1}$에 다음과 같이 저장한다.

$V_i=\{2i, 2i+1, 2m+i\}$,
$U_i=\{2i, 2i+1, 2m+(i+1)_m\}, i=0,...,m-1.$

이 부호는 반복도가 2이므로, 가용도(availability)[24]는 $\rho-1=1$인 부호가 된다. 즉, 최대 한 개의 노드에 장애가 발생한 경우에만 부분접속 복구가 가능하다. 두 개 이상의 노드 장애가 발생하면 부분접속 복구가 불가능한 경우가 생긴다.

생성법 1을 통해 생성된 부호는 항상 그림 3(b)와 같은 구조를 갖는 그래프로 표현된다. 모든 노드에 대해서 항상 두 개의 에지로 연결된 하나의 이웃 노드와 한 개의 에지로 연결된 하나의 이웃 노드가 존재한다. 이런 그래프로 표현되는 부호의 최대 저장 용량은 다음과 같이 표현할 수 있다.

$$\begin{aligned} M_1(K) &= 3+\underbrace{\{(3-2)+(3-1)+(3-2)+\cdots\}}_{(K-1)\text{회}} \\ &= 3+3(K-1)-(2+1+2+\cdots) \\ &= 3K-(2+1+2+\cdots) \end{aligned} \quad (2)$$

기존의 반복분할 부호는 식 (1)의 MBR 부호의 최대 용량을 $d=\alpha, \beta=1$의 파라미터로 달성한다. 식 (1)을 이 반복분할 부호의 파라미터를 이용해 다음과 같이 정리할 수 있다.

$$\begin{aligned} M_{MBR}(K) &= K\alpha-\binom{K}{2} \\ &= K\alpha-\frac{K(K-1)}{2} \end{aligned} \quad (3)$$

$$= 3K-\frac{K(K-1)}{2} \quad (4)$$

식 (4)는 생성법 1의 부호와의 비교를 위해 $\alpha=3$을 대입하여 구한 식이다.

생성법 1을 통해 생성된 부호의 최대 용량을 나타내는 식 (2)를 다시 정리하면 다음과 같다.

$$M_1(K)=\begin{cases} 3K-\dfrac{3(K-1)}{2} & , K\text{는 홀수} \\ 3K-\dfrac{3(K-1)+1}{2} & , K\text{는 짝수} \end{cases} \quad (5)$$

위의 두 식 (4)와 (5)를 통해, $M_{MBR}(K)$와 $M_1(K)$를 비교하면, $K<3$의 영역에서는 $M_{MBR}>M_1$이고 $K>3$의 영역에서는 $M_{MBR}<M_1$이 된다. 그리고 $K=3$일 때 두 값은 같아진다. 이는 $K\geq 3$의 영역에서는 생성법 1에 의해 생성되는 부분접속 복구 가능한 반복분할 부호가 MBR의 최대 용량을 달성하는 반복분할 부호가 됨을 보여준다.

생성법 1의 부호는 가용도가 $\rho-1=1$인 부호로, 최대 한 개의 노드 장애에 대해서만 부분접속 복구가 가능하다는 단점이 있다. 즉, 두 개 이상의 노드에 장애가 발생한 경우에는 K개의 노드에 모두 접속하여 원래의 파일을 만들어낸 후 이를 통해 다시 재 부호화의 과정을 통해 노드에 저장돼 있던 α개의 심벌을 만들어야 한다. 이러한 높은 복잡도를 피하기 위해 반복분할 부호에서 여러 개의 노드 장애가 발생한 상황에도 부분접속 복구가 가능하기 위해서는 반복도 ρ를 3 이상으로 늘려야 한다.

반복도가 3인 부호의 생성 방법에 대해 생각해 보자. 생성법 1에서 저장 용량의 최대화를 위해서 모든 노드는 두 개의 에지로 동시에 연결되는 이웃 노드를 오직 한 개만 갖도록 했다. 아래의 생성법 2는 이러한 조건을 반복도가 3인 경우에도 만족시키도록 하는 생성 방법이다.

생성법 2. $(12, M)$ MDS 부호를 통해 생성된 부호화 심벌 인덱스의 집합을 $\Omega=\{0,...,7, A, B, C, D\}$라고 하자. 12개의 심벌은 다시 반복도 $\rho=3$인 반복 부호를 통해 부호화 된다. 생성된 36개의 심벌을 용량이 $\alpha=3$인 12개의 노드 $V_0,..., V_{11}$에 그림 4와 같이 저장한다.

생성법 2에 의해 생성된 부호는 반복도가 $\rho=3$이면서, 동시에 모든 노드에 대해 두 개의 에지로 연결되는 이웃 노드가 오직 한 개만 존재한다는 조건을 만족시킨다. 예를 들어, 노드 V_0은 노드 V_1과 두 개의 심벌을 공유하고 나머지 이웃 노드 V_4, V_5, V_8, V_{10}과는 각각 하나의 심벌만을 공유한다. 다른 노드들도 모두 이러한 이웃 노드를 갖는다. 생성된 부호를 적용한 시스템의 최대 저장용량은 사용자가 접속하는 노드의 수 K에 따라 달라지는 값으로, $K=1$일 때 $M_2(1)=3$, $K=2$일 때, $M_2(2)=3+1$, 그리고 $K=3$일 때, $M_2(3)=3+1+2$로, 생성법 1에 의해 생성된 부호와 동일하다. 즉 생성법 2에 의해 생성된 부호의 최대 용량은

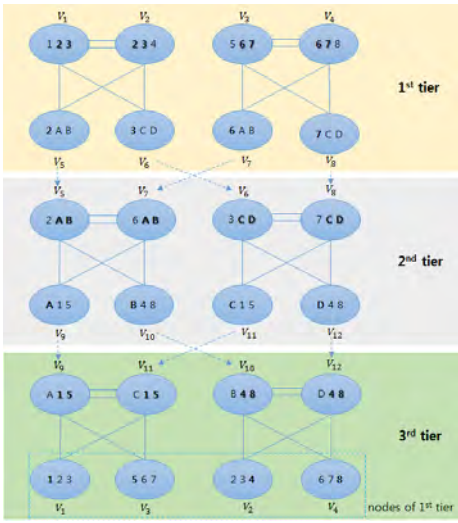


그림 4. $\alpha=3, \rho=3$을 갖는 부분접속수 2인 반복분할 부호의 생성 (생성법 2)
Fig. 4. Construction of FR codes with loclaity 2 when $\alpha=3, \rho=3$ (Construction 2)

$$M_2(K)=3+(K-1)+\left\lfloor \frac{K-1}{2} \right\rfloor \quad (6)$$

로 $M_1(K)$와 동일하다.

이를 기반으로 더 큰 크기를 갖는 부호를 쉽게 생성할 수 있다. 그림 4는 총 3단계(tier)로 이루어져 있다. 이 부호는 $K\leq 4$에서만 식 (6)의 최대 용량 $M_2(K)$를 유지할 수 있다. $K\geq 5$를 요구하는 시스템에 적합한 부호는 이 생성 단계를 4 이상으로 늘림으로써 가능하다.

생성법 2에 의해 생성되는 부호는 반복분할 부호나 생성법 1의 부호에 비해 동일한 크기의 파일을 저장하는데 필요한 노드의 수 n과 MDS 부호 심벌의 수 θ가 모두 증가한다. 이는 각각 부분접속수와 가용도를 향상시키기 위해 필요한 비용으로 생각할 수 있다.

지금까지 부분접속수가 2인 반복분할 부호의 두 가지 생성 방법을 제안하고 각각의 특성에 대해 논의하였다. 두 부호 모두 반복분할 부호와 마찬가지로 복구 과정에서 부호화 및 추가의 연산이 필요 없고 동일한 복구가 가능하다는 특성을 갖도록 함으로써 복구 과정에서의 낮은 복잡도를 유지하면서 부분접속수를 낮추었다.

본 논문에서 제안하는 두 부호는 모두 반복분할 부호의 부분접속수를 감소시키는 부호로, 부분접속수가 항상 2가 되도록 설계된 부호들이다. 두 개의 부호의 다른 성능이 기존의 반복분할 부호와 비교해 어떻게 달라지는지를 살펴보겠다. 생성법 1의 부호는 적절한 비교를 위해, 정규 그래프를 이용해 생성되는 반복도 2인 기존 반복분할 부호와 비교한다. 다음으로 생성법 2의 부호는 가용도가 3인 부호이므로, steiner system을 이용해 생성하는 기존의 반복분할 부호와 비교한다.

표 1은 $\rho=2$일 때 기존의 반복분할 부호와 생성법 1에서 제안하는 부호를 비교한 것이다. 앞서 언급한 생성법 1은 오직 $\alpha=3$인 경우에 대해서만 다루고 있다. 하지만 생성법 1의 그래프를 한 노드의 저장 공간의 크기가 $\alpha>3$인 경우로 쉽게 일반화 시킬 수 있다. 일반화된 부호의 그래프 표현에서는 한 노드에 저장된 심벌들을 각각 β_1개의 심벌과 β_2개의 심벌의 두 그룹으로 나누고 두 개의 이웃 노드와 각 그룹의 심벌을 공유하도록 한다. 편의를 위해 $\beta_1\geq\beta_2$라고 가정하자. 생성법 1에서 명시하고 있는 방법은 이 일반화된 부호의 $\beta_1=2, \beta_2=1$, $\alpha=\beta_1+\beta_2=3$인 예라고 볼 수 있다. 이러한 생성법 1을 일반화시킨 부호의 용량은 다음과 같다.

표 1. 반복분할부호와 제안하는 부분접속 복구 가능한 반복분할 부호의 비교 ($\rho=2$)
Table 1. Comparison between FR codes and Locally Repairable FR codes from construction 1 ($\rho=2$)

	FR codes from a regular graph[6]	Proposed code (construction 1)
locality	3	2
availability	1	1
max. file size	$K\alpha-\frac{K(K-1)}{2}$	$\begin{cases} K\alpha-\frac{\alpha(K-2)}{2}-\beta_1, K \text{ is even} \\ K\alpha-\frac{\alpha(K-1)}{2}, K \text{ is odd} \end{cases}$
comparison with MBR	always achieves the MBR capacity	achieves MBR capacity if $K\geq\alpha$
parameters	$\theta=\frac{n\alpha}{2}$, $n\geq K$	$\theta=\frac{n\alpha}{2}$, $n\geq K+1$

$$M_1(K,\alpha,\beta_1)=\begin{cases}K\alpha-\dfrac{\alpha(K-2)}{2}-\beta_1, & K\text{가 짝수}\\ K\alpha-\dfrac{\alpha(K-1)}{2}, & K\text{가 홀수}\end{cases} \quad (7)$$

두 부호의 용량을 비교하면 기존의 정규 그래프 기반 반복분할 부호는 항상 MBR 용량을 달성할 수 있는데 반해, 제안하는 생성법 1의 부호는 $K\geq\alpha$일 때만 MBR 용량을 달성할 수 있다. 두 부호는 모두 $\rho\theta=n\alpha$를 만족하는 파라미터만 가질 수 있다. 또한 두 부호 모두 노드의 수는 K보다 많아야 한다. 이에 더해, 생성법 1의 부호는 최적 용량을 달성하기 위해서는 부호의 그래프 표현에서 크기가 K 이하인 싸이클(cycle)이 존재하지 않아야 한다.

해당 부호는 모든 노드가 하나의 싸이클을 이룬다. 따라서 전체 노드의 수가 $K+1$ 이상이 되어야 한다. 또한 부호의 생성과정을 생각할 때, 생성법 1의 부호의 노드 수가 홀수가 되는 경우, n개의 노드 중 한 노드가 저장하는 심벌의 수가 나머지 $n-1$개의 노드와 달라진다. 이는 $K=1$일 때의 용량 $M(1)$의 값이 최대 용량을 달성하지 못하게 한다. 하지만 나머지의 $K>1$에서는 $M(K)$의 값은 여전히 최대 용량을 달성할 수 있다. 비교를 통해 제안하는 부호가 부분접속수를 항상 2로 유지할 수 있는 대신, 기존의 반복분할 부호에 비해 최대 용량에서 손해가 발생하는 것을 알 수 있다. 또한 동시에, 제안하는 부호가 가질 수 있는 파라미터 중 노드의 수가 기존의 반복분할 부호에 비해서 조금 더 제한적임을 확인할 수 있다.

표 2는 $\rho=3$인 기존의 반복분할 부호와 제안하는 부호를 비교한 것으로, 제안하는 부호는 생성법 2에 의해 생성된 부호이다. 기존의 반복분할 부호는 steiner system을 기반으로 생성된 부호를 비교 대상으로 하였다. 그리고 생성법 2의 부호가 $\alpha=3$에서만 생성될 수 있으므로, 동등한 비교를 위해 steiner system 기반 반복분할 부호도 $\alpha=3$으로 고정시킨다. 반복분할 부호의 부분접속수는 항상 α이므로, 3이 된다. 제안하는 부호는 이를 2로 감소시킨 부호이다. 두 부호 모두 가용도가 2이므로, 2-노드장애가 발생한 경우에도 부분접속 복구가 가능하다. 2-노드장애가 발생한 경우에는 두 부호의 부분접속수가 3으로 동일하다. 표 2의 용량은 $\alpha=3$인 경우의 각 부호의 용량을 나타낸다. 두 부호는 모두 $\rho\theta=n\alpha$라는 관계식을 만족해야 하고, 기존의 반복분할 부호는 steiner system이 존재하기 위한 조건도 동시에 만족해야 한다. 이를 고려하면 기존 반복분할 부호의 노드 수는 $n=\dfrac{\theta(\theta-1)}{\alpha(\alpha-1)}$이고, $\alpha=3$이므로 $n=\dfrac{\theta(\theta-1)}{6}$이 된다. 생성법 2의 부호는 $n=4\left(\max\left\{3,\left\lceil\dfrac{K}{2}\right\rceil+1\right\}\right)$의 조건을 만족해야 표 2의 다섯 번째 행에 주어진 부호의 용량을 달성할 수 있다. 또한 생성법 2가 갖는 가장 큰 약점은 $\alpha=3$인 경우만 생성할 수 있다는 점이다. 이 두 부호의 비교를 통해서도 기존의 반복분할 부호의 부분접속수의 성능을 향상시키는 대신 일부 영역에서의 용량의 감소와 생성 가능한 파라미터의 제한이 발생함을 확인할 수 있었다.

2.3 제안하는 부호의 평가

2절에서는 작은 부분접속수를 갖는 반복분할 부호의 두 가지 생성 방법을 제안하고 이를 기존의 반복분할 부호와 비교하였다. 이 절에서는 제안하는 부호를

표 2. 반복분할부호와 부분접속 복구 가능한 반복분할 부호의 비교 ($\rho=3,\alpha=3$)
Table 2. Comparison between FR codes and Locally Repairable FR codes from construction 2 ($\rho=3,\alpha=3$)

	FR codes from a Steiner system[6]	Proposed code (construction 2)
locality (1-failure)	3	2
localtiy (2-failure)	3	3
availability	2	2
max. file size	$3K-\dfrac{K(K-1)}{2}$	$\begin{cases}3K-\dfrac{3(K-2)}{2}-2, & K\text{ is even}\\ 3K-\dfrac{3(K-1)}{2}, & K\text{ is odd}\end{cases}$
comparison with MBR	always achieves the MBR capacity	achieves MBR capacity if $K\geq 3$
parameters	$n=\dfrac{\theta(\theta-1)}{6}$	$n=4\left(\max\left\{3,\left\lceil\dfrac{K}{2}\right\rceil+1\right\}\right)$, $\alpha=3$

부분접속 복구 부호들과 비교한다. 기존의 부분접속 복구 부호들은 부분접속수가 낮지만 복구 시에 심벌의 연산을 필요로 한다는 단점을 갖는다. 지금까지 알려진 대부분의 부분접속 복구 부호들은 큰 크기의 유한체 상에서 생성된다. 따라서 심벌의 연산은 높은 연산 복잡도를 갖게 된다. 연산이 필요 없고 부분접속수 1을 달성할 수 있는 반복부호가 사실상의 표준 (de facto standard)으로 사용되고 있지만, 안정성(reliability)이 떨어진다는 단점을 갖는다. 본 논문에서 제안하는 반복분할 부호에 기반 한 부분접속 복구 부호는 단순 전달 복구의 특성을 가지므로 연산이 필요 없는 노드 복구가 가능하여 기존의 부분접속 복구 부호들에 비해 연산 복잡도가 낮다는 장점을 갖는다. 또한 부분접속수 관점에서 최적인 반복부호에 비해 부분접속수는 2로 증가하지만 더 높은 안정성을 갖는다. 이 장에서는 제안하는 부호를 반복부호 및 기존의 간단한 부분접속 복구 부호[18]와 비교한다.

[18]의 부호는 가용도가 1인 부호이므로 이 부호와의 동등한 비교를 위해 2회 반복 부호와 본 논문에서 제안하는 부호 중 생성법 1의 부호를 비교한다. 또한 세 부호 모두 $\alpha=3$으로 동일하게 설정한다. 부호를 통해 저장되는 파일은 크기가 6이라고 가정하자. 2회 반복부호는 4개의 노드를 필요로 한다. 생성법 1의 부호는 6개의 노드를 필요로 하는 그림 3의 부호가 된다. 이는 $(9,6)$ MDS부호를 외부호로 사용하는 부호이다. 마지막으로 [18]에서 제안된 부호는 크기 6인 파일을 각각 크기가 3인 두 개의 그룹으로 나누어, $(6,3)$ MDS 부호를 각각 적용해 총 12개의 부호 심벌을 생성하고 이들의 XOR 연산을 통해 추가의 6개 패리티 심벌을 생성한다. 이때 필요한 노드의 수는 6이 된다.

세 부호를 다음의 성능 척도를 통해 비교함으로써 제안하는 부호를 평가한다: 1) 연산 복잡도; 2) 안정성; 3) 저장 공간 오버헤드; 그리고 4) 복구 대역폭.

2.3.1 연산 복잡도

부호화와 복구에 필요한 연산의 횟수를 비교한다. 반복부호는 부호화 및 복구 시에 연산을 필요로 하지 않는다. 먼저 부호화 과정을 비교한다. 생성법 1을 통해 생성된 그림 3의 부호와 [18]의 부분접속 복구 부호는 모두 MDS 부호를 통해 먼저 부호화되는 과정을 거친다. (n,k) MDS 부호는 부호화 과정에서 총 $k\times(n-k)$회의 곱 연산이 필요하고, $(k-1)\times(n-k)$회의 덧셈 연산이 필요하다. 심벌 당 필요한 연산의 횟수를 구하기 위해 메시지 심벌의 수로 나누어준 값을 비교한다. 제안하는 부호는 $(9,6)$ MDS 부호를 이용하므로, 심벌 당 3회의 곱 연산과 2.5회의 덧셈 연산이 필요하다. [18]의 부분접속 복구 부호는 $(6,3)$ MDS 부호를 두 개 사용하므로 심벌 당 3회의 곱 연산과 2회의 덧셈 연산이 필요하다. 또한 [18]의 부호는 부분접속 복구를 위한 추가의 패리티 심벌을 MDS 부호화 심벌 간의 덧셈을 통해 구하는 과정이 필요하므로, 심벌 당 1회의 덧셈이 추가적으로 더 필요하여 총 3회의 덧셈이 필요하게 된다.

다음으로 복구 과정에서의 연산을 비교한다. 부호화와 마찬가지로 반복부호는 연산 없이 복구가 가능하다. 제안하는 부호 역시 반복부호와 마찬가지로 연산이 필요 없이 단순 전달 복구가 가능하다. 하지만 [18]의 부호는 다른 대부분의 부분접속 복구 부호와 마찬가지로 복구 시 연산이 필요하다. 이 부호의 경우 한 노드를 복구하기 위해 3회의 덧셈 연산이 필요하다.

2.3.2 안정성

전통적으로 부호의 안정성을 나타내는 척도로 부호의 최소 거리(minimum distance)를 사용해 왔다. 따라서 가장 먼저 각 부호의 최소 거리를 비교한다. 2회 반복부호의 최소거리는 2이다. 제안하는 그림 3의 부호는 최소거리 4를 갖고 [18]의 부분접속 복구 부호 역시 최소 거리가 4이다. 이 비교를 통해 반복부호는 간단하다는 장점에도 불구하고 최소거리가 굉장히 작아서 부호의 안정성이 떨어진다고 할 수 있다. 조금 더 높은 안정성을 얻기 위해 3회 반복부호를 사용한다고 하더라도 최소 거리는 3이 된다. 최소거리는 해당 부호를 통해 항상 복구한 노드 장애의 최대 개수를 알려주는 척도이다. 이는 최악의 경우(worst case)의 시나리오를 가정한다. 제안하는 부호와 [18]의 부호는 4로 동일한 최소거리를 갖는다. 하지만 실제로는 동일한 최소거리를 갖는 부호라고 하더라도 최소거리 이상의 노드에 동시에 장애가 발생해도 복구 가능한 패턴의 수가 다를 수 있다. 다음으로는 이러한 장애 패턴에 따른 복구 확률까지 고려한 불안정성의 확률을 비교한다.

각 부호의 안정성은 노드 장애가 발생했을 때 적용된 부호에 따라 복구 가능한 확률을 통해 비교할 수 있다. 2회 반복부호의 경우, 한 개의 노드 장애는 항상 복구가 가능하므로 복구 가능 확률이 1이다. 두 개의 노드 장애가 발생한 경우, 어떤 두 개의 노드에 동시에 장애가 발생했는지에 따라 복구 가능 여부가 달라진다. 총 네 개의 노드 중 두 개의 노드에 장애가 발생할 수 있는 패턴은 6가지가 존재한다. 이때 복구 가능

한 패턴이 네 개이고 나머지 두 개는 복구할 수 없는 경우이다. 따라서 $\frac{4}{6}$의 복구 가능 확률을 갖는다. 3개 이상의 장애는 항상 복구할 수 없다. x개의 노드에 동시에 장애가 발생한 경우 복구할 수 있는 확률을 $r_p(x)$라고 하자. 각 부호의 구조에 따라 이 값을 모두 구할 수 있다. 예로 든 2회 반복부호의 경우 $r_p(0)=r_p(1)=1, r_p(2)=\frac{4}{6}, r_p(3)=r_p(4)=0$이다. 유사한 방법으로 제안하는 그림 3의 부호는 $r_p(0)=r_p(1)=r_p(2)=r_p(3)=1,\ r_p(4)=\frac{9}{15}$, 그리고 $r_p(5)=r_p(6)=0$이다. 마지막으로 [18]의 부호의 경우, $r_p(0)=r_p(1)=r_p(2)=r_p(3)=1$이고, $r_p(x\geq 4)=0$이다. 하나의 노드에 장애가 발생하는 확률은 $p_e=0.01$이라고 하자. 각 노드에 장애가 발생하는 것은 독립적이다. 모든 발생 가능한 노드 장애의 패턴에 대해서 각각의 부호를 통해 복구할 수 있는 확률은 $\sum_{x=0}^{n} r_p(x)\binom{n}{x}p_e^x(1-p_e)^{n-x}$으로 구할 수 있다. 각 부호가 복구할 수 없는 장애의 확률을 나타내는 불안정(unrecoverability)는 다음의 식을 이용해 구할 수 있다[25].

$$p_f = 1.0 - \sum_{x=0}^{n} r_p(x)\binom{n}{x}p_e^x(1-p_e)^{(n-x)} \qquad (8)$$

위 식을 통해 각 부호의 불안정성을 구하면 2회 반복부호가 가장 큰 값을 갖는다. 또한 동일한 최소거리를 가졌던 그림 3의 부호와 [18]의 부호의 경우 불안정성의 값은 다른 것을 확인할 수 있다. 제안하는 그림 3의 부호가 가장 낮은 불안정성의 값을 갖는다. 이는 제안하는 부호가 가장 많은 경우의 장애를 복구할 수 있음을 의미한다.

다음으로 조금 더 현실적인 시스템 환경을 고려한 안정성을 비교해 본다. 분산 저장 시스템은 주로 마르코프 연쇄 모델(Markov Chain Model)을 이용하여 모델링된다[13,26]. 마르코프 모델의 각 상태(state)는 장애가 발생한 노드의 수를 나타낸다. 최초 상태는 모든 노드가 정상 동작을 하는 상태로, 상태-0에서 시작한다. 한 노드의 장애 발생은 발생 빈도 λ를 갖는 지수분포(exponential distribution)를 따른다고 가정한다. 이것을 이용해 상태-1에서 다음 상태인 상태-2로 전이되는 전이율(transition rate)을 구할 수 있다. 반대 방향의 상태 전이는 장애 발생 노드를 복구함으로써 이루어지므로 복구율 ρ가 전이율이 된다. 노드 복구율은 노드의 크기 C_{node}, 복구 시 각 노드에서 사용 가능한 트래픽 B_{rep}, 시스템 전체에 존재하는 노드의 수 N, 사용된 부호의 부분접속수 d, 그리고 노드에 장애가 발생했음을 인지하고 복구를 수행할 때까지 걸리는 시간 T에 의해 결정된다[26,27]. 여기서는 [26]과 동일하게 $C_{node}=15\,TB, B_{rep}=0.1\,Gbps, N=3000$, 그리고 $T=30\min$으로 설정한다. [13]에서는 3000개의 노드로 구성된 페이스북의 프로덕션 클러스터를 한 달 동안 관찰하여, 평균 노드 장애 발생률을 조사하였다. 이를 이용해, 한달 평균 22개의 노드 장애가 발생한다고 가정하여, $\lambda=22 nodes/month$로 설정한다. 한 노드 장애 상황에서의 노드 복구율은 $\rho_1 \approx \frac{B_{rep}N}{C_{node}d}$이다. 2회 반복부호, 제안하는 부호, 그리고 [18]의 부호의 마르코프 모델을 구성하고 각각에 맞는 노드 장애율 및 복구율을 이용해 복구 불가능한 상태로 가기 전까지의 시간을 구한다. 이를 MTTDL(Mean Time To Data Loss)라고 하고, 분산 저장 시스템에서 각 부호의 안정성 척도로 사용한다. 2회 반복부호는 66.25일, [18]의 부호는 7.17년, 그리고 제안하는 그림 3의 부호는 32.69년의 MTTDL을 갖는 것을 확인할 수 있었다. MTTDL을 통한 안정성 비교를 통해서도 제안하는 부호가 우수함을 확인할 수 있다. MTTDL은 절대적인 시간의 값으로서의 의미보다는 분산 저장 시스템에 사용되는 부호 간의 성능 비교를 위해 사용하기에 충분한 척도이다[13].

2.3.3 저장공간 오버헤드

비교를 위해 예로 들고 있는 부호는 모두 6개의 데이터 심벌을 가정하고 있다. 2회 반복 부호의 경우 6개의 심벌이 추가로 필요하게 된다. [18]의 부호와 제안하는 부호는 모두 6개의 데이터 심벌에 12개의 심벌이 추가로 생성된다. 두 부호 모두 2회 반복부호에 비해 저장공간 오버헤드가 크다. 그에 반해 앞에서 살펴본 바와 같이 두 부호 모두 반복부호에 비해 높은 안정성을 갖는다. 더 높은 안정성을 갖는 반복부호로, 3회 반복부호를 가정하면, 저장공간 오버헤드가 [18]의 부호나 제안하는 그림 3의 부호와 동일해진다. 하지만 3회 반복부호 역시 MTTDL이 262일로, 크게 높아지지 않는다.

2.3.4 복구 대역폭

부분접속수와 달리, 복구 대역폭은 복구 과정에서

실제로 전송이 이루어지는 데이터의 양을 나타낸다. 동일한 부분접속수를 갖는 부호라 하더라도 복구 대역폭이 다를 수 있고, 이 차이 역시 전체 시스템의 성능에 영향을 미치는 요인이 된다. 먼저 하나의 노드 장애만 발생한 경우(1-노드 장애)를 생각해 보자. 반복부호와 제안하는 반복분할 부호는 모두 복구 과정에서 재생해야 하는 데이터와 동일한 양의 데이터 전송이 이루어진다. 반면 [18]의 부호의 경우, 하나의 노드를 복구할 때 부분접속수는 2이지만 실제 전송이 이루어져야 하는 심벌의 양은 6이다. 노드 하나에 저장된 심벌의 수는 3이므로, 심벌 하나의 복구에 필요한 평균 복구 대역폭은 2가 된다. 이는 나머지 두 개의 부호의 복구 대역폭이 1인 것에 비해 2배 크다.

이러한 부호들의 성능 척도 비교는 표 3에 정리하였다. 또한 가용도가 2인 부호의 비교를 위해, 3회 반복부호와 생성법 2에 의해 생성되는 부호에 대해서도 위의 성능 척도를 비교하여 표 4에 정리하였다.

이러한 비교를 통해, 반복부호는 부분접속수 측면에서 가장 우수한 것을 확인할 수 있다. 하지만 최소거리와 불안정성이라는 지표를 통해 안정성이 떨어진다는 것을 확인할 수 있다. 일반적으로 ρ회 반복부호는 $n=\frac{\rho M}{\alpha}$개의 노드를 필요로 하고, 최소 거리는 $d_{\min}=\rho$가 된다. 또한 대부분의 부분접속 복구 부호는 표 3에서 살펴본 부분접속 복구 부호[18]와 마찬가지로 복구 과정에서 유한체 상에서의 심벌 연산을 필요로 한다. 이는 복구 과정의 연산 복잡도를 증가시킴으로써 복구 성능을 저하시키는 요인이 된다. 일반적인 부분접속 복구 부호는 부분접속수에만 초점을 맞추어, 복구 대역폭 관점에서는 최적이 아니다. 표 3과 표 4를 통해, 본 논문에서 제안한 부호가 일반적인 부분접속 복구 부호와 비교해, 복구 과정에서의 연산을

표 3. 부분접속 복구 부호와 제안하는 생성법 1 부호의 비교 (M=6)
Table 3. Comparison among a 2x repetition code, an LRC of [18], and the proposed code of construction 1 (M=6)

	2x repetition code	proposed code (construction 1)	simple LRC[18]
number of nodes	4	6	6
locality (1-failure)	1	2	2
locality (2-failure)	cannot recover	1.6	1.8
repair bandwidth	3	3	6
number of computations per repair	-	-	3 XOR operations
number of nodes for acquiring a file (K)	3	3	3
unrecoverability probability	3.33×10^{-4}	5.94×10^{-8}	1.47×10^{-7}
minimum distance	$d_{\min}=2$	$d_{\min}=4$	$d_{\min}=4$
MTTDL	66.25 days	32.69 years	7.17 years
storage overhead	1x	2x	2x

표 4. 3회 반복부호와 제안하는 생성법 2 부호의 비교 (M=6)
Table 4. Comparison between a 3x repetition code and the proposed code of construction 2 (M=6)

	3x repetition code	proposed code (construction 2)
number of nodes	6	12
locality	1	2
repair bandwidth	3	3
computations per repair	-	-
number of nodes for acquiring a file(K)	4	3
minimum distance	$d_{\min}=3$	$d_{\min}=10$
MTTDL	262 days	3.52×10^{6} years
storage overhead	2x	5x

전혀 필요로 하지 않는 단순 전달 복구를 수행함으로써 복구 과정의 복잡도를 낮출 수 있음을 보았다. 동시에 MBR 부호의 특성을 그대로 물려받아, 한 노드의 복구대역폭 γ가 복구하는 노드에 저장하는 심벌의 개수 α와 동일하므로, 복구대역폭 측면에서도 최적임을 확인하였다. 또한 반복부호와 비교해서는 안정성을 높일 수 있다는 장점을 가짐을 확인하였다.

Ⅲ. 결 론

본 논문에서는 분산 저장 시스템에 적합한 재생 부호인 반복분할 부호를 부분접속수 측면에서 분석하고 이 부호의 부분접속수를 향상시키기 위한 두 가지 새로운 부호의 생성 방법에 대해 소개하였다. 제안하는 부호는 반복분할 부호에서 고려되지 않았던 부분접속수 성능을 고려한 것으로, 기존 반복분할 부호에 비해 향상된 부분접속수를 갖는 대신 저장할 수 있는 파일의 최대 크기가 기존 반복분할 부호에 비해 감소한다. 또한 제안하는 부호는 기존의 부분접속 복구 부호들에 비해서도 좋은 성능을 보인다. 제안하는 부호는 반복부호에 비해서 높은 안정성을 갖고, 일반적인 부분접속 복구 부호와 비교해서도 더 높은 안정성을 갖는다. 또한 제안하는 부호는 복구 과정에서 연산이 필요없어, 복구 복잡도가 낮고 복구대역폭 측면에서 최적이라는 장점을 갖는다.

앞으로는 본 논문에서 제안된 부호의 생성을 더 다양한 파라미터로 확장시키고, 저장공간 오버헤드도 더 낮출 수 있는 방향으로의 연구를 수행할 계획이다. 또한 본 논문에서의 연구 결과를 발전시켜 일반적인 부분접속 복구 부호의 생성에 대한 연구를 수행할 계획이다.

References

[1] J. S. Park, J.-H. Kim, K.-H. Park, and H.-Y. Song, “Average repair read cost of linear repairable code ensembles,” *J. KICS*, vol. 39B, no. 11, Nov. 2014.

[2] J.-H. Kim, M.-Y. Nam, and H.-Y. Song, “Construction of [2^k-1+k, k, 2^k-1+1] codes attaining Griesmer bound and its locality,” *J. KICS*, vol. 40, no. 03, Mar. 2015.

[3] T. Ahn, Y. Kim, and S. Lee, “Dynamic resource allocation in distributed cloud computing,” *J. KICS*, vol. 38B, no. 7, Jul. 2013.

[4] Samjong KPMG, “The future value creation through big-data analytics,” *Issue Monitor*, no. 17, Oct. 2013.

[5] J.-H. Kim, J. S. Park, K.-H. Park, M. Y. Nam, and H.-Y. Song, “Trends of regenerating codes for next-generation cloud storage systems,” *Inf. Commun. Mag.*, vol. 31, no. 2, pp. 125-131, Feb. 2014.

[6] S. E. Rouayheb and K. Ramchandran “Fractional repetition codes for repair in distributed storage systems,” *48th Annu. Allerton Conf. Commun., Control, Comput.,* pp. 1510-1517, 2010.

[7] A. G. Dimakis, P. B. Godfrey, Y. Wu, M. J. Wainwright, and K. Ramchandran, “Network coding for distributed storage systems,” *IEEE Trans. Inf. Theory*, vol. 56, no. 9, pp. 4536-4551, Sept. 2010.

[8] S.-J. Lin and W.-H Chung “Novel repair-by-transfer codes and systematic exact-MBR codes with lower complexities and smaller field sizes,” *IEEE Trans. Parallel Distrib. Syst.*, vol. 25, no. 12, pp. 3232-3241, Dec. 2014.

[9] P. Gopalan, H. Cheng, H. Simitci, and S. Yekhanin, “On the locality of codeword symbols,” *IEEE Trans. Inf. Theory*, vol. 58, no. 11, Nov. 2012.

[10] K. V. Rashmi, N. B. Shah, and P. V. Kumar, “Optimal exact-regenerating codes for distributed storage at the MSR and MBR points via a product-matrix construction,” *IEEE Trnas. Inf. Theory*, vol. 57, no. 8, Aug. 2011.

[11] N. B. Shah, K. V. Rashmi, P. V. Kumar, and K. Ramchandran, “Regenerating codes for distributed storage networks,” in *Proc. Int. Conf. Arithmetic of Finite Fields*, pp. 215-223, 2010.

[12] K. V. Rashmi, N. B. Shah, P. V. Kumar, and K. Ramchandran, “Explicit construction of optimal exact regenerating codes for distributed storage,” in *Proc. 47th Annu. Allerton Conf. Commun., Control, Comput.*,

pp. 1243-1249, Monticello, IL, Sept. 2009.

[13] M. Sathiamoorthy, M. Asteris, D. Papailiopoulos, A. G. Dimakis, R. Vadali, S. Chen, and D. Borthakur, "XORing elephants: Novel erasure codes for big data," in *Proc. Very Large Data Base Endowment*, vol. 6, no. 5, pp. 325-336, Mar. 2013.

[14] J. C. Koo and J. T. Gill, "Scalable constructions of fractional repetition codes in distributed storage systems," in *49th Annu. Allerton Conf. Commun., Control, Comput.*, pp. 1366-1373, Monticello, IL, 2011.

[15] N. Silberstein and T. Etzion, *Optimal fractional repetition codes based on graphs and designs*, [Online] Available: http://arxiv.org/abs/1401.4734, 2014.

[16] O. Olmez and A. Ramamoorthy, *Fractional repetition codes with flexible repair from combinatorial designs*, [Online] Available: http://arxiv.org/pdf/1408.5780v1, 2014.

[17] D. R. Stinson, *Combinatorial designs: Constructions and analysis*, Springer, 2004.

[18] D. Papailiopoulos and A. G. Dimakis, "Locally repairable codes," *IEEE Trans. Inf. Theory*, vol. 60, no. 10, pp. 5843-5855, Oct. 2014.

[19] W. Song, S. H. Dau, C. Yuen, and T. J. Li, "Optimal locally repairable linear codes," *IEEE J-SAC*, vol. 32, no. 5, pp. 1019-1036, May, 2014.

[20] I. Tamo and A. Barg, "A family of optimal locally recoverable codes," *IEEE Trans. Inf. Theory*, vol. 60, no. 8, Aug. 2014.

[21] F. Oggier and A. Datta, "Self-repairing homomorphic codes for distributed storage systems," in *Proc. IEEE INFOCOM,* pp. 1215-1223, 2011.

[22] F. Oggier and A. Datta, "Self-repairing codes for distributed storage-A projective geometric construction," in *Proc. IEEE ITW*, pp. 30-34, 2011.

[23] N. B. Shah, K. V. Rashmi, P. V. Kumar, and K. Ramchandran, "Distributed storage codes with repair-by-transfer and nonachievability of interior points on the storage-bandwidth tradeoff," *IEEE Trans. Inf. Theory*, vol. 58, no. 3, Mar. 2012.

[24] A. S. Rawat, D. S. Palailiopoulos, A. G. Dimakis, and S. Vishwanath, "Locality and availability in distributed storage," in *Proc. IEEE ISIT*, pp. 681-685, 2014.

[25] C. Huang, M. Chen, and J. Li, "Pyramid codes: Flexible schemes to trade space for access efficiency in reliable data storage systems," *ACM Trans. Storage*, vol. 9, no. 1, Mar. 2013.

[26] C. Huang, H. Simitci, Y. Xu, A. Ogus, B. Calder, P. Gopalan, J. Li, and S. Yekhanin, "Erasure coding in windows azure storage," in *Proc. USENIX Annu. Tech. Conf.*, vol. 37, no. 5, pp. 82-96, Boston, MA, Jun. 2012.

[27] M. Shahabinejad, M. Khabbazian, and M. Ardakani, "An efficient binary locally repairable codes for hadoop distributed file system," *IEEE Commun. Lett.*, vol. 18, no. 8, pp. 1287-1290, Jul. 2014.

남 미 영 (Mi-Young Nam)

2005년 2월 : 연세대학교 전기전자공학과 졸업

2005년 2월~2007년 8월 : 삼성전자 연구원

2009년 8월 : 연세대학교 전기전자공학과 석사

2009년 9월~현재 : 연세대학교 전기전자공학과 박사과정

<관심분야> 부호이론, 정보이론, 분산저장시스템

김 정 현 (Jung-Hyun Kim)

2006년 8월 : 연세대학교 전기전자공학과 졸업
2008년 8월 : 연세대학교 전기전자공학과 석사
2010년 7월~2013년 2월 : 한국전자통신연구원 연구원
2013년 3월~현재 : 연세대학교 전기전자공학과 박사과정
<관심분야> 통신공학, 정보이론, 부호이론, 분산저장시스템

송 홍 엽 (Hong-Yeop Song)

1984년 2월 : 연세대학교 전자공학과 졸업
1986년 5월 : University of Southern California Dept. of EE. Systems 석사
1991년 12월 : University of Southern California Dept. of EE. Systems 박사
1992년 1월~1993년 12월 : Post-Doc Research Associate, University of Southern California Dept. of EE. Systems
1994년 1월~1995년 8월 : Senior Engineer, Qualcomm Inc., San Diego, California.
2002년 3월~2003년 2월 : Visiting Professor, University of Waterloo, Canada
1995년 9월~현재 : 연세대학교 전기전자공학과 교수
<관심분야> 통신공학, 정보이론, 부호이론, 암호이론, 이산수학