논문 15-40-09-18 The Journal of Korean Institute of Communications and Information Sciences '15-09 Vol.40 No.09
http://dx.doi.org/10.7840/kics.2015.40.9.1786

랜덤 선형 네트워크 코딩의 실용적 설계 및 성능 분석

이 규 진◆, 신 연 철*, 구 종 회*, 최 성 현°

Practical Implementation and Performance Evaluation of Random Linear Network Coding

Gyujin Lee◆, Yeonchul Shin*, Jonghoe Koo*, Sunghyun Choi°

요 약

랜덤 선형 네트워크 코딩(Random Linear Network Coding, RLNC)은 멀티캐스트의 신뢰성을 높이는 방법으로 널리 사용되고 있다. RLNC 구현에 있어서 효율적인 연산을 위해 Galois Field (GF)를 사용한다. 상용 컴퓨터에서의 연산을 고려하였을 때 GF(2^m)의 크기 m이 32보다 작을 경우, 곱셈과 나눗셈에 대해 사전에 계산된 table을 사용하면 모든 사칙 연산이 m에 관계없이 상수 복잡도를 가진다. 이로부터 RLNC의 연산 복잡도는 m에 반비례하는 것을 보인다. 추가적으로, m이 커짐에 따라 발생하는 헤더 길이 증가, 메모리 사용량 증가 등의 추가적인 오버헤드를 고려하여 실용적인 GF의 크기를 선택한다. 이를 바탕으로 상용 컴퓨터에 RLNC를 구현하고 곱셈/나눗셈 연산 시에 사용되는 table의 종류와 한 번에 인코딩 되는 원본 패킷의 개수에 따른 성능을 실측한다.

Key Words : Random Linear Network Coding (RLNC), Galois Field, Complexity, Implementation

ABSTRACT

Random linear network coding (RLNC) is widely employed to enhance the reliability of wireless multicast. In RLNC encoding/decoding, Galois Filed (GF) arithmetic is typically used since all the operations can be performed with symbols of finite bits. Considering the architecture of commercial computers, the complexity of arithmetic operations is constant regardless of the dimension of GF m, if m is smaller than 32 and pre-calculated tables are used for multiplication/division. Based on this, we show that the complexity of RLNC inversely proportional to m. Considering additional overheads, i.e., the increase of header length and memory usage, we determine the practical value of m. We implement RLNC in a commercial computer and evaluate the codec throughput with respect to the type of the tables for multiplication/division and the number of original packets to encode with each other.

Ⅰ. 서 론

실시간 무선 멀티미디어 서비스를 제공함에 있어 하나의 멀티미디어 컨텐츠를 동시에 여러 단말에게 전송하는 경우 멀티캐스트 기법이 유니캐스트 기법에 비해 효율적이다. 하지만 멀티캐스트 기법의 경우 유

◆ First Author : Department of ECE and the Institute of New Media and Communications, Seoul National University, gjlee@mwnl.snu.ac.kr, 학생회원
° Corresponding Author : Department of ECE and the Institute of New Media and Communications, Seoul National University, schoi@snu.ac.kr, 종신회원
* Department of ECE and the Institute of New Media and Communications, Seoul National University, ycshin@mwnl.snu.ac.kr, jhkoo@mwnl.snu.ac.kr, 학생회원
논문번호 : KICS2015-06-169 Received June 7, 2015; Revised July 24, 2015; Accepted August 25, 2015

니캐스트 기법과는 달리 Acknowledgement와 재전송을 지원하지 않기 때문에 전송 신뢰성이 낮다. 이를 극복하기 위해서 잠재적으로 발생될 수 있는 전송 실패를 미리 고려하여 추가적인 패킷을 애초에 만들어 전송하는 랜덤 선형 네트워크 코딩 (Random Linear Network Coding, RLNC) 등의 기술들이 제안되어 왔다. RLNC 기술은 원본 패킷을 전송하는 대신 원본 패킷의 linear combination을 만들어 전송하는 기술로, 널리 사용되고 있는 기술이다.[1-3] RLNC 기술은 신뢰성이 높은 반면 인코딩 및 디코딩을 위한 추가적인 연산이 필요하다. 이 때, 사칙연산에 닫혀 있는 Galois field (GF)를 이용하면 보다 효율적으로 연산이 가능하다.

실시간 멀티미디어 전송을 위해서는 지연 시간을 최소화해야 하므로, RLNC에서의 인코딩 및 디코딩 과정에서 걸리는 시간을 줄여야 한다. 이를 위해 인코딩 및 디코딩 과정에서의 연산 복잡도를 최소화해야 한다. 본 논문에서는 상용 컴퓨터에서 32-bit 연산까지는 같은 복잡도로 계산할 수 있다는 사실을 고려하여, GF에서 사칙연산시 GF의 크기에 관계없이 상수 복잡도를 가지는 것을 보인다. 이를 통해 기존에 알려진 것과 달리[4] 작은 크기의 GF를 사용할 때 RLNC에서 더 높은 복잡도를 가진다는 것을 보인다. 또한 GF의 크기가 커짐에 따라 발생하는 헤더 길이 증가, 메모리 사용량 증가 등의 오버헤드를 고려하여 실용적인 GF의 크기로 8을 선택한다. 상용 컴퓨터에 RLNC를 구현하고 곱셈/나눗셈 연산 table의 종류와 한 번에 인코딩 되는 원본 패킷의 개수에 따른 인코딩 및 디코딩 수율을 실측하였다.

Ⅱ. 본 론

2.1 랜덤 선형 네트워크 코딩 (Random Linear Network Coding, RLNC)

랜덤 선형 네트워크 코딩 (Random Linear Network Coding, RLNC)는 보내려고 하는 패킷들을 k개씩 묶어서 인코딩하여 n개의 인코딩된 패킷을 생성하는 기술로, 그림 1은 RLNC의 개관을 보여준다. 사용하는 GF의 크기가 m bits일때, k개의 원본 패킷마다 각각 길이가 m bits인 coefficient를 랜덤하게 만든다. 길이가 d bytes인 원본 패킷을 m bits씩 나누어, 기생성한 coefficient를 곱해 패킷간의 linear combination 을 $\lceil 8d/m \rceil$ 번 수행함으로써 인코딩된 패킷을 1개 만들어낸다. 같은 과정을 n번 반복해, 총 n개의 인코딩된 패킷을 만들어 전송한다. 위의 인코딩 과정은 식(1)로 표현된다.

$$AX = Y \tag{1}$$

A는 길이가 m bits인 coefficient로 구성된 $n \times k$ 행렬, X는 길이가 d bytes인 k개의 원본 패킷을 m bits씩 나눈 $k \times q$행렬, Y는 인코딩 결과 만들어진 $n \times q$행렬이다. X의 각 행이 원본 패킷, Y의 각 행이 인코딩된 패킷을 의미한다. d가 m으로 나누어 떨어지지 않을 경우 추가적인 bit를 붙여서 인코딩하고 디코딩시 제거한다.

인코딩된 패킷을 만들 때 사용된 랜덤 coefficient를 패킷 헤더에 담아서 전송하게 된다. 전송과정에서 erasure channel에 의해서 패킷 손실이 발생하더라도 n개중 coefficient가 선형 독립인 k개의 패킷을 받게 되면, 식(2)를 통해 원본 패킷으로 복원해 낼 수 있다.

$$A'^{-1} Y' = X \tag{2}$$

A'는 받은 패킷의 coefficient로 구성된 $k \times k$ 행렬, Y'는 받은 k개의 인코딩된 패킷을 m bits씩 나눈 $k \times q$행렬, X는 원본 패킷에 해당되는 $k \times q$행렬이다 X의 각 행이 원본 패킷, Y'의 각 행이 받은 인코딩된 패킷을 의미한다.

2.2 랜덤 선형 네트워크 코딩에서 최적의 GF 크기

GF는 소수의 거듭제곱으로 표현이 되는 유한개의 원소를 가지며 사칙 연산에 대해 닫혀있는 집합이다.[5] 원소의 개수가 2^m인 GF를 GF(2^m)으로 표현하며, 이때 m이 GF의 크기이다. 다시 말해서, GF(2^m)에서는 모든 원소가 m bits로 표현된다. 예를 들어, m이 2일 때 GF(2^2)의 한 예는 식(3)과 같다.

$$GF(2^2) = \{0,1,2,3\} \tag{3}$$

GF(2^2)의 사칙 연산은 그림 2와 같이 정의될 수 있으며, 사칙 연산에 대해 닫혀있음을 확인할 수 있다.

2.2.1 GF 크기에 따른 덧셈 및 뺄셈의 연산 복잡도

GF내에서 원소간의 덧셈 및 뺄셈은 XOR 연산과 동일하다. 상용 컴퓨터에서는 32 bits 이하의 자료형에 대한 기본 연산(XOR, 덧셈 등)은 자료형의 크기에 관계없이 상수 복잡도를 가진다. 그림 3은 c언어에서

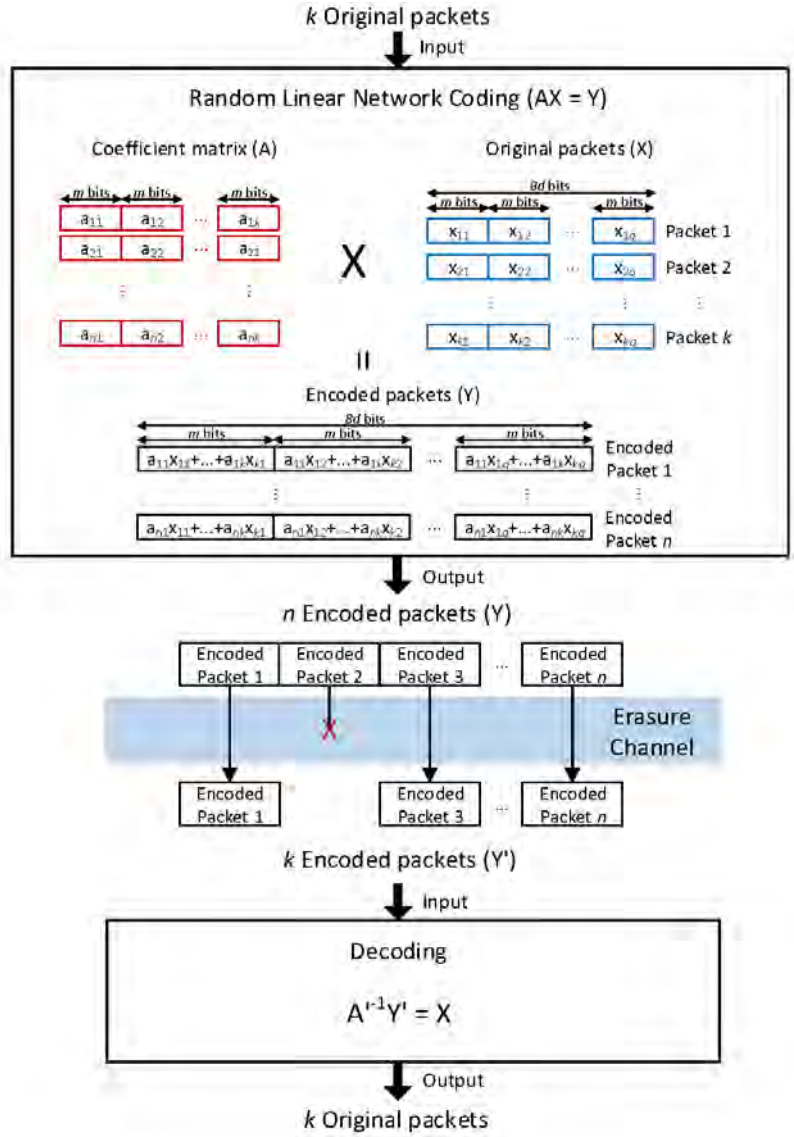

그림 1. 랜덤 선형 네트워크 코딩의 개관
Fig. 1. Overview of random linear network coding

+/-	0	1	2	3
0	0	1	2	3
1	1	0	3	2
2	2	3	0	1
3	3	2	1	0

x	0	1	2	3
0	0	0	0	0
1	0	1	2	3
2	0	2	3	1
3	0	3	1	2

/	0	1	2	3
0	NaN	NaN	NaN	NaN
1	0	1	2	3
2	0	3	1	2
3	0	2	3	1

그림 2. GF(2^2)의 사칙 연산
Fig. 2. Arithmetic operations of GF(2^2)

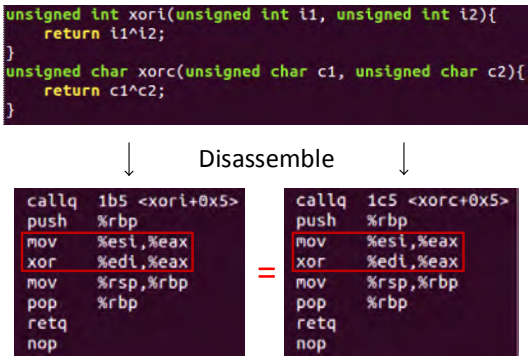

그림 3. 8-bit 와 32-bit XOR 연산의 디스어셈블 결과
Fig. 3. The disassemble result of 8-bit and 32-bit XOR operation

두 개의 char (8 bits) 형 변수와 두 개의 int (32 bits) 형 변수의 XOR 연산을 수행하는 함수를 디스어셈블(disassemble)한 결과로, XOR 연산과 관련된 부분(빨간 네모 박스)의 어셈블리어가 동일하다는 것을 확인할 수 있다. 어셈블리어가 동일하면 하드웨어에서 같은 연산을 하므로 8 bits와 32 bits의 XOR 연산은 같은 상수 복잡도를 가지게 된다. 따라서 $m \leq 32$인 GF(2^m)에서의 덧셈 및 뺄셈은 m값에 관계없이 상수 복잡도($O(1)$)를 가지게 된다.

2.2.2 GF 크기에 따른 곱셈 및 나눗셈의 연산 복잡도

GF에서의 덧셈과 뺄셈은 간단한 XOR 연산으로 수행될 수 있지만, 곱셈과 나눗셈은 간단한 연산으로 수행될 수 없다. 따라서 곱셈 및 나눗셈에서의 시간을 줄이기 위해서 사전에 연산해 결과를 기록해놓은 table이 이용된다.[6] 곱셈 및 나눗셈에서 이용되는 table은 1-D Table과 2-D Table, 2가지 종류로 나누어질 수 있다.

1) 1-D Table

식 (4)과 같이 곱셈 연산은 log 연산과 anti-log(alog) 연산 (log의 역연산) 의 성질을 이용하면 쉽게 계산될 수 있다. 연산 과정에서 덧셈 결과 GF의 범위를 벗어날 수 있는데, 이를 막기 위해서 벗어나게 되면 2^m-1 을 빼준다. 따라서 곱셈 연산은 최대 2번의 log 연산, 1번의 alog 연산, 1번의 덧셈 연산, 1번의 뺄셈 연산, 1번의 비교 연산으로 계산될 수 있다.

곱셈과 마찬가지로, 식 (5)와 같이 나눗셈 연산도 log 연산과 alog 연산의 성질을 이용해 쉽게 계산된다. 연산 과정에서 음수를 취할 때 GF의 범위를 벗어나는 것을 막기 위해서 2^m-1 을 더해준다. 나눗셈 연산은 1번의 곱셈 연산, 1번의 log 연산, 1번의 alog 연산, 1번의 뺄셈 연산으로 계산될 수 있다. 즉, 나눗셈 연산은 총 3번의 log연산, 2번의 alog 연산, 1번의 덧셈 연산, 1번의 뺄셈 연산으로 계산될 수 있다. (항상 $(2^m-1)-\log(b)$는 GF의 범위에 있으므로, 곱셈 연산시 비교 연산과 뺄셈 연산이 필요하지 않다.)

$$\begin{aligned} a \div b &= a \times b^{-1} \\ &= a \times \mathrm{alog}(\log(b^{-1})) \\ &= a \times \mathrm{alog}(((2^m-1)-\log(b))) \end{aligned} \tag{5}$$

곱셈 및 나눗셈 연산은 log, alog, 덧셈, 뺄셈, 비교 연산으로 계산될 수 있다. 이중 덧셈, 뺄셈, 비교연산은 XOR연산과 마찬가지로 상용 컴퓨터에서 기본적으로 제공하는 간단한 연산이므로, GF(2^m)의 모든 원소에 대한 log 연산과 alog 연산을 사전에 수행하고 그 결과 값을 기록한 table을 이용하면, 이후 곱셈 및 나눗셈 연산을 보다 빠르게 계산할 수 있다.

GF(2^m)의 모든 원소의 log 연산과 alog 연산 결과를 기록한 table을 1-D table로 정의한다. GF의 모든 원소는 특정 소수의 거듭제곱으로 표현되므로, 특정 소수를 밑으로 하는 log 연산과 alog 연산 결과 값은 m bits로 간단히 표현될 수 있다. 따라서 log 및 alog 1-D table의 크기는 각각 $2^m \times m$ bits 이다.

2) 2-D Table

앞에서는 log 연산과 alog 연산의 성질을 이용해 곱셈과 나눗셈을 계산하였다. 하지만 미리 모든 원소간의 곱셈과 나눗셈 연산 그 자체의 결과를 기록해 직접적으로 이용하면 log 연산과 alog 연산을 이용할 때보다 빠르게 계산이 가능하다. 이때의 곱셈과 나눗셈

$$\begin{aligned} a \times b &= \mathrm{alog}\{\log(a \times b)\} \\ &= \mathrm{alog}\{(\log(a)+\log(b)) \bmod (2^m-1)\} \\ &= \begin{cases} \mathrm{alog}\{(\log(a)+\log(b))\}, & (\log(a)+\log(b)) \leq 2^m-1 \\ \mathrm{alog}\{(\log(a)+\log(b)-(2^m-1))\}, & otherwise \end{cases} \end{aligned} \tag{4}$$

을 계산해 기록해 놓은 table을 2-D table로 정의한다. GF는 사칙 연산에 대해서 닫혀있기 때문에 곱셈 연산과 나눗셈 연산 결과 값은 m bits로 표현된다. 그러므로 곱셈 및 나눗셈 2-D table의 크기는 각각 $(2^m)^2 \times m$ bits 이다.

3) 1-D Table와 2-D Table을 사용하였을 때의 곱셈 및 나눗셈의 연산 복잡도

log 연산과 alog 연산은 한번의 table lookup으로 계산할 수 있다. Table lookup은 table에서 원하는 위치의 값을 얻어오는 것이기 때문에 log 연산과 alog 연산은 상수 복잡도를 가진다. 또한 덧셈, 뺄셈, 비교 연산은 앞에서 보였던 XOR연산과 마찬가지로 상용 컴퓨터에서 지원하는 기본적인 연산이므로 32bit 이하의 자료형에 대해서 같은 연산 복잡도를 가진다. 따라서 1-D table을 사용하였을 때 $m \leq 32$인 GF(2^m)에서의 곱셈 및 나눗셈 연산은 상수 복잡도를 가진다. 2-D table을 사용하였을 때 곱셈 및 나눗셈 연산은 한번의 table lookup으로 계산할 수 있다. 그러므로 곱셈 및 나눗셈 연산은 상수 복잡도를 가지게 된다.

따라서 1-D table을 사용하였을 때와 2-D table을 사용하였을 때 모두 $m \leq 32$인 GF(2^m)에서의 곱셈 및 나눗셈 연산은 상수 복잡도를 가진다.

2.2.3 GF 크기에 따른 RLNC 연산 복잡도

GF(2^m)을 이용해 RLNC를 수행할 경우, 한 symbol의 길이는 m bits이다. $m \leq 32$이고 원본 패킷의 개수 k와 만들고자 하는 인코딩된 패킷의 개수 n, 패킷의 길이 d bytes가 상수값으로 주어졌을 때, 식 (1)과 식 (2)을 풀기 위해 수행되어야 할 사칙연산의 총 수는 $O(1/m)$이다. $m \leq 32$인 GF(2^m)에서의 사칙연산이 m값에 관계없이 상수 복잡도를 가지므로, 식 (1)과 식 (2)의 총 복잡도는 $O(1/m)$이다. 따라서 $m \leq 32$인 경우, RLNC의 연산 복잡도는 GF의 크기 m에 반비례한다.

2.2.4 오버헤드를 고려한 실용적인 GF 크기

한편, GF의 크기 m이 커지게 되면 오버헤드가 증가하게 된다. 첫 번째로 RLNC에서는 coefficient를 헤더에 넣어서 보내게 되는데 헤더의 크기는 m에 비례하므로, m이 클 경우 헤더가 커지는 문제가 발생한다. 두 번째로 RLNC에서 m이 클 경우 곱셈 및 나눗셈에서 사용하는 table이 메모리를 너무 많이 차지하게 된다. 2.2.2에서 설명하였듯이, table의 크기는 m에 커짐에 따라서 지수적으로 증가하게 된다. 마지막으로 패킷의 길이가 GF(2^m)의 크기로 나누어떨어지지 않는 경우에는, 부가적인 bit를 추가해야 하는 문제가 생긴다. 위의 세 가지의 추가적인 오버헤드를 모두 고려하였을 때, 특히 헤더의 크기와 table의 크기를 고려하였을 때, 너무 큰 m값은 실용적이지 않다. 따라서 너무 크지 않으면서 항상 패킷 길이가 나누어떨어질 수 있는 최대 값인 8이 실용적으로 사용될 수 있는 GF의 크기일 것이다.

2.3 곱셈 및 나눗셈을 위한 table 종류에 따른 성능 분석

앞에서 설명하였듯이, 사전에 연산해 결과를 기록해 놓은 table을 이용하면 GF에서의 곱셈 및 나눗셈 연산 복잡도를 낮출 수 있어 RLNC의 성능을 향상시킬 수 있다. 한편, table 종류에 따라 연산 양과 table이 차지하는 메모리의 크기에 대한 tradeoff 관계가 있다. 어떤 table을 사용하더라도 곱셈 및 나눗셈 연산은 상수 복잡도를 가지지만, 1-D table을 사용할 때의 연산 양이 2-D table을 사용할 때 보다 많다. 한편 1-D table은 각 원소 log와 alog 결과 값을 가지는 반면, 2-D table은 원소끼리의 곱셈 및 나눗셈 결과 값을 가지므로 2-D table의 크기가 1-D table의 크기보다 크다. 따라서 실제 RLNC에서 사용시에는 이러한 tradeoff를 고려한 table 선택이 필요하다.

2.4 k에 따른 성능 분석

RLNC에서는 한 번에 묶어서 인코딩되는 k를 유동적으로 변경할 수 있으며, k값에 따라 성능이 변한다. 따라서 상황에 맞는 k값 선택이 필요하다.

k가 작은 경우, 적은 원본 패킷을 묶어서 인코딩하므로 한 개의 인코딩된 패킷을 만들기 위해서 k가 클 때에 비해서 적은 연산이 필요하다. 또한 디코딩 과정에서도 적은 인코딩된 패킷을 가지고 원본 패킷을 복원하므로, 디코딩에도 적은 연산이 필요하다. 따라서 인코딩 및 디코딩시 적은 연산이 필요하므로 수율이 증가하게 된다. 한편 k가 큰 경우, 같은 code rate에도 더 에러에 강하다는 것이 알려져 있다.[7] 따라서 이러한 tradeoff 관계를 고려한 k값 선택이 필요하다.

2.5 실측 결과

상용 노트북(Samsung Ativ XG700T1C-F53)에 GF를 이용한 RLNC를 실제 리눅스 커널에 구현하고 RLNC의 성능을 실측하였다. 이에 대한 특징적인 기능들을 요약하면 표 1과 같다.

표 1. 실측 장비의 특징적인 기능
Table 1. The specification of equipment

Features	Samsumg Ativ XG700T1C-F53
Chipset	Intel® Centrino® Advanced-N 6235
CPU	Intel® CoreTM i5 Processor 3337U
Memory	4GB DDR3 System Memory at 1600MHz

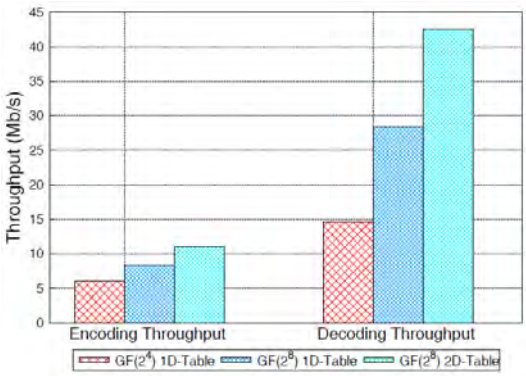

그림 4. GF 크기 및 table 종류에 따른 랜덤 선형 네트워크 코딩 수율
Fig. 4. Throughput of RLNC in GF size and type of table

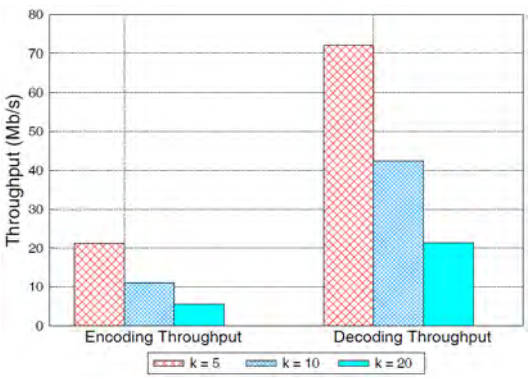

그림 5. k에 따른 랜덤 선형 네트워크 코딩 수율
Fig. 5. Throughput of RLNC in different k

그림 4는 GF의 크기와 곱셈과 나눗셈에 사용된 table의 종류에 따른 RLNC의 인코딩 및 디코딩 수율을 실측한 결과이다. $k = 10$, $n = 20$, $d = 1498$ bytes로 하였다. GF 크기가 4일 때 보다 8일 때 인코딩 및 디코딩 수율이 크므로, 이로부터 RLNC의 복잡도가 GF의 크기에 반비례하는 것을 확인할 수 있다. 또한 1D-table을 사용할 때에 비해 2D-table을 사용할 때 인코딩 및 디코딩 수율이 크다는 것으로부터, 2D-table을 사용할 때 1D-table을 사용할 때에 비해서 곱셈 및 나눗셈 연산 양이 적다는 것을 확인할 수 있다.

그림 5는 원본 패킷 개수에 따른 RLNC 수율을 실측한 결과이다. k값이 작을수록 하나의 인코딩된 패킷을 만들기 위해 적은 연산이 필요하므로 높은 수율을 가지는 것을 확인할 수 있다.

따라서 RLNC에서 GF의 크기는 8으로 하고, 곱셈 및 나눗셈을 위해 2D-table을 사용하며, 작은 k를 사용하는 것이 RLNC에서 가장 높은 인코딩/디코딩 수율을 얻을 수 있다.

Ⅲ. 결 론

본 논문에서는 실제 컴퓨터 연산을 고려하여 $m \leq 32$인 GF(2^m)에서의 사칙 연산은 GF의 크기 m에 관련 없이 상수 복잡도를 가지는 것을 보였다. 따라서 RLNC의 연산 복잡도는 GF의 크기 m에 반비례한다는 것을 보였고, 또한 헤더 길이 증가, 메모리 사용량 증가 등의 추가적인 오버헤드 또한 고려하여 8을 RLNC에서 실용적인 GF의 크기로 결정하였다. 곱셈 및 나눗셈 연산을 위해 사전에 계산해서 기록해놓은 table 종류에 따라 연산 양과 차지하는 메모리의 양에 대한 tradeoff 관계를 분석하였다. k에 따른 RLNC 성능을 분석하여 k가 작을수록 수율이 높아지는 것을 보였다. 상용 컴퓨터에 GF를 사용한 RLNC를 구현하여 인코딩 및 디코딩 수율을 실측해 분석한 내용을 확인하였다.

References

[1] S. Kim and J. Shin, “A joint sub-packet level network coding and channel coding,” *J. KICS*, vol. 40, no. 04, pp. 659-665, Apr. 2015.

[2] K. Lee, S. Cho, and J. Kim, “Feasibility analysis of network coding applied to IEEE802.11s wireless mesh networks,” *J. KICS*, vol. 37B, no. 11, pp. 1014-1021, Nov. 2012.

[3] M. Park and W. Yoon, “Optimized multipath network coding in multirate multi-hop wireless network,” *J. KICS*, vol. 37B, no. 9, pp. 734-740, Nov. 2012.

[4] K. T. Kim, C.-S. Hwang, and V. Tarokh, “Network error correction from matrix network coding,” in *ITA*, pp. 1-9, La Jolla, CA, Feb. 2011.

[5] E. Win, A. Bosselaers, S. Vandenberghe, P. D. Gersem, and J. Vandewalle, "A fast software implementation for arithmetic operations in GF(2n)," in *ASIACRYPT'96*, vol. 1163, pp. 65-76, Kyongju, Korea, Nov. 1996.

[6] Z. Wan, *Lectures on Finite Fields and Galois Rings*, World Scientific Publishing Co. Pte. Ltd., 2003.

[7] J. Heide, M. V. Pedersen, F. H. P. Fitzek, and T. Larsen, "Network coding for mobile devices-systematic binary random rateless codes," in *IEEE Int. Conf. Commun. Wksp. (ICC'09)*, pp. 1-6, Dresden, Germany, Jun. 2009.

이 규 진 (Gyujin Lee)

2013년 2월 : 서울대학교 전기정보공학부 학사 졸업
2013년 3월~현재 : 서울대학교 전기정보공학부 석박사통합과정
<관심분야> 무선 통신 네트워크, 멀티미디어 통신 네트워크

신 연 철 (Yeonchul Shin)

2010년 2월 : 서울대학교 전기정보공학부 학사 졸업
2010년 3월~현재 : 서울대학교 전기정보공학부 석박사통합과정
<관심분야> 무선 통신 네트워크, 멀티미디어 통신 네트워크

구 종 희 (Jonghoe Koo)

2011년 2월 : 서울대학교 전기정보공학부 학사 졸업
2011년 3월~현재 : 서울대학교 전기정보공학부 석박사통합과정
<관심분야> 무선 통신 네트워크, IoT

최 성 현 (Sunghyun Choi)

1992년 : 한국과학기술원 전기전자공학과학사 졸업
1994년 : 한국과학기술원 전기전자공학과 석사 졸업
1999년 : University of Michigan 전기컴퓨터공학과 박사 졸업
1999년~2002년 : 미국 Philips Research 선임연구원
2002년~현재 : 서울대학교 전기정보공학부 교수
<관심분야> 무선 통신 네트워크, 멀티미디어 통신 네트워크, IoT