한국정보전자통신기술학회논문지(jkiiect)'18-08, Vol.11 No.4
논문 18-11-04-421 http://dx.doi.org/10.17661/jkiiect.2018.11.4.421

IoT 및 금융 거래 실시간 데이터 정보의 압축 전송을 위한 새로운 고효율 유니버설 코드(BL-beta) 제안

김정훈*

New high-efficient universal code(BL-beta) proposal for compressed data transferring of real-time IoT sensing or financial transaction data

Jung-Hoon Kim*

요 약 IoT device 측정 데이터 또는 거래 데이터는 관측 정보가 실시간으로 전송되고 이를 처리하는 과정에서 많은 트래픽이 발생한다. 이를 실시간 무 손실 압축 기법인 universal code를 이용하면 효과적으로 압축 또는 전송할 수 있다. 본 논문은 측정 수치의 최대 범위를 예측하기 어렵고, 매우 짧은 시간 마다 비교적 일정한 범위 내에서 데이터가 발생하는 주식 거래량 데이터의 압축 전송을 위해, 본 연구진의 새롭게 개발한 유니버설 코드 BL-beta를 이용하여 압축 전송에 적용해보니, 고정 길이 비트 전송에 비해 최소 49.5%이상의 높은 압축 효율을 보였으며, 기존 유니버설 코드인 Exponential Golomb 코드 보다 16.6% 더 우수한 압축 전송 성능을 나타내었다.

Abstract While IoT device sensing data or financial transaction data is transmitted in real time, huge data traffic is generated in processing it. This huge data can be effectively compressed or transmitted using universal code, which is a real-time lossless compressor. In this paper, we propose our BL-beta code, which is newly developed universal code for compressing stock trading data, which the maximum range of measured values is difficult to predict and is generated within a relatively constant range over a very short period of time. For compressing real-time stock trading data, Compared with the fixed length bit transmission, the compression efficiency is at least 49.5% higher than that of the fixed length bit transmission, and the compression transmission performance is 16.6% better than the Exponential Golomb code.

Key Words : Exponential Golomb code, IoT, Lossless compression, Real-time data compression, Universal code

1. 서론

1.1 유니버설 코드의 정의와 특징

유니버설코드(Universal Code)는 임의의 정수 또는 자연수(이하 code-num 이라함)를 간단한 연산을 통해 변환된 유일 복호성(unique decodability)를 가진 이진수 코드로서, 데이터 압축 방법 중의 하나이다[1]. 구현 용이성, 실시간 압축 등의 장점이 있으나, 일반적으로 사전 기반 압축 방법에 비해서 압축 효율이 다소 떨어지는 단점이 있었다[2]. 그런데 최근 외부 환경 정보를 감지하여 수치화한 뒤 전송하는 IoT 측정 장비들이 다수 보급되면서 IoT 장비로부터의 실시간 수치 데이터의 전송량이 급격히 늘고 있다[3]. 특히 측정 장치의 특성상 비교적 짧은 시간 간격마다 데이터를 수집해서 전송해야 하는데[4-6], 이때 많은 연산 및 사전 공간을 필요로 하는 사전 방식(dictionary type) 압축 방법보다는 유니버설 코드를

* Corresponding Author : Graduate School of Public Health, Seoul National University (jhkim@binarylab.co.kr)
Received July 26, 2018 Revised July 30, 2018 Accepted August 22, 2018

이용한 무 사전 방식의 실시간 압축 방법을 활용하면 별도의 사전을 위한 공간이 불필요 하고, 간단한 연산을 통해 효과적으로 수치 데이터를 압축 전송할 수 있을 것으로 예상 할 수 있다. 또한 주식 거래량 데이터와 같은 금융거래 데이터의 통계적 특성도 유사한 모습을 보이는데 이러한 분야에 본 연구에서 제안한 BL-beta 유니버설 코드를 이용한 압축전송 방법을 활용할 수 있을 것으로 기대되었다.

1.2 연구의 목적

본 연구는 1.1에서 언급한 바와 같이 IoT 장비 또는 금융 거래 등에서 발생하는 실시간 수치 데이터를 효과적으로 압축 또는 전송하기 위하여 유니버설코드(Universal Code)의 장점을 살리면서, 압축률이 다소 낮은 단점을 보완하기 위하여 본 논문에서는 본 연구의 저자가 직접 개발한 새로운 유니버설 코드의 사용을 제안하고자 하였다. H.264등 비디오 압축 표준에서 사용되고 있는 Exponential Golomb 코드의 접두사 코드(Prefix code)를 본 연구에서는 reversed form 의 BL-alpha 로 대체하여 새롭게 생성하였고, 이 코드를 BL-beta 코드라고 명명하였고 이를 활용하여 기존 유니버설 코드보다 향상된 압축률을 보일 수 있음을 증명하고자 하였다.

2. 본론

2.1 바이너리 클러스터의 정의

임의의 '10'으로 시작하는 이진수의 경우, 예를 들어, '10111010100101011111101001…' 의 경우 최 상위 비트(the most significant bit)로 부터 최하위 비트(the least significant bit)로 이동하면서, '10'을 만날 때 마다 데이터를 분할하면, '1011/10/10/100/10/101111/10/1001' 로 분할되는데 각각의 분할되는 이진수를 바이너리 클러스터(Binary Cluster)라고 명명하였다. 이 바이너리 클러스터는 유일복호성이 있다. 따라서 각각의 바이너리 클러스터를 분할하여 전송하여도 수신 측에서 이를 원래의 바이너리 클러스터를 재조합 할 수 있다. 아래 표1은 바이너리 클러스터의 모든 생성 가능한 조합을 비트 수 길이에 따라 나열한 표이다. 바이너리 클러스터는 '1'로 시작하고 1개 이상의 '0'이 이어지며, 0개 이상의 '1'들로 구성된 이진수이며, 특징적인 것은 '11'로 시작하는 경우는 존재하지 않는다는 점이다[7-8]. Length Group Index는 Binary Cluster를 코드 길이를 기준으로 그룹으로 묶었을 때의 순차적인 그룹 일련 번호를 의미한다.

표 1. 바이너리 클러스터의 모든 가능한 조합
Table 1. All possible patterns of binary cluster

code num	Binary Cluster	Bit Length	Length Group Index
1	10	2	1
2	100	3	2
3	101	3	2
4	1000	4	3
5	1001	4	3
6	1011	4	3
7	10000	5	4
8	10001	5	4
9	10011	5	4
10	10111	5	4
11	100000	6	5
12	100001	6	5
13	100011	6	5
14	100111	6	5
15	101111	6	5
…	…	…	…

2.2 바이너리 클러스터를 이용한 유니버설 코드

2.2.1 Unary code와 BL-alpha 코드의 비교

표1에 따르면, 자연수 또는 정수로 이루어진 code-num에 대하여 1:1로 바이너리 클러스터가 할당되고, 각 바이너리 클러스터는 유일복호성이 있기 때문에 유니버설 코드로 활용할 수 있음을 알 수 있다. 이러한 코드를 본 연구에서는 BL-alpha 코드라고 명명하였는데, BL-alpha 코드와 아래 표2와 같이 가장 기본적인 유니버설 코드의 한 종류인 unary code와 비교하여 보면, 그림 1과 같은 code-num에 따른 코드의 길의 증가가 매우 완만하게 증가함을 알 수 있다. 이렇게 완만하게 증가하는 방식을 본 논문에서는 buffered elongation이라고 하였다.

표 2. Code-num 에 대응하는 Unary code
Table 2. Unary code responding to code-num

code num	Unary cpde	Bit Length
1	1	1
2	01	2
3	001	3
4	0001	4
5	00001	5
6	000001	6
7	0000001	7
8	00000001	8
...	...	...

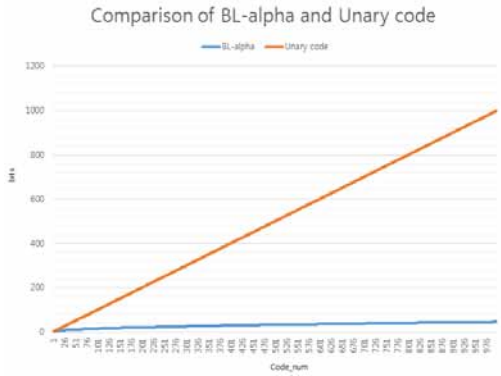

그림 1. Code-num에 대응하는 unary-code와 BL-alpha코드의 비트길이 비교
Fig. 1. Bit length comparison of unary-code and BL-alpha code corresponding to Code-num

한편 unary code는 H.264 영상압축 알고리즘에 채택되어 있는 Exponential Golomb code의 prefix로 사용되고 있는 코드이며[1], 실제 모습은 아래 표3과 같다. 그 외에도 Golomb code, Elias 코드에도 unary code를 채택하고 있다[9-12].

표 3. Code-num 에 대응하는 Exponential Golomb code(k=0, k는 외부변수)
Table 3. Exponential Golomb code(k=0) responding to code-num

code num	prefix	Exponential Golomb code	Bit Length
1	1	1	1
2	01	010	3
3	001	011	3
4	0001	00100	5
5	00001	00101	5
6	000001	00110	5
7	0000001	00111	5
8	00000001	0001000	7
...	...	...	...

표3의 Exponential Golomb code의 접두 코드 부를 BL-alpha 코드로 대체한다면 code-num 이 충분히 커지더라도 더 작은 비트에서 표현이 가능해질 것으로 예측되었다. 이때 유일 복호성 유지를 위해 Exponential Golomb code 의 접두 코드를 역순(Reversed form)의 BL-alpha를 사용하였다. 먼저 이와 같은 접두부 코드로서 BL-alpha코드의 생성 방법을 아래 절에 상술하였다.

2.2.2 BL-alpha 코드의 엔코딩

BL-alpha 코드는 특정 code-num 을 M이라고 했을 때, K 와 X 라는 두 개의 값을 본 논문의 저자가 직접 새롭게 개발한 수식을 사용하여 BL-alpha 코드를 생성할 수 있다[13].

(1) 먼저 위의 표1와 같이 압축하고자 하는 code-num 을 M이라고 할 때, 먼저 BL-alpha code의 group index를 나타내는 그림 3과 같은 식을 이용하여 K 값을 구한다. K값의 의미는, code-num M 에 대응하는 BL-alpha 코드의 코드길이-1을 나타낸다. code-num M 과 K값의 관계는 아래 수식 (1)과 같고,

$$\frac{K(K-1)}{2} < M \le \frac{K(K+1)}{2} \quad \cdots\cdots (1)$$

수식(1)은 아래 수식(2)와 동등하다

$$\frac{-1+\sqrt{1+8M}}{2} \le K < \frac{1+\sqrt{1+8M}}{2} \quad \cdots\cdots (2)$$

K는 수식(2)를 만족시키는 가장 큰 정수이므로, 아래 수식(3)으로 간단히 표현할 수 있다. K는 표1에서의 Length Group Index를 의미한다.

$$K = \left[\frac{1+\sqrt{1+8M}}{2} \right] \quad \cdots\cdots (3)$$

(단, $[p]$ 는 p보다 작은 정수 중에 최대의 정수를 돌려주는 함수이다)

(2) 수식(1)~(3)을 통해 M 및 K를 구한 뒤, 아래 수식(4)를 이용하여 X를 계산할 수 있다. X의 의미는 특정 M으로부터 생성되는 BL-alpha코드를 길이를 기준으로 동일 그룹으로 묶으면 K 번째 group 내에서 X번째 BL-alpha 코드라는 의미이다.

$$X = M - \frac{K(K-1)}{2} \quad \cdots\cdots (4)$$

(3) 이제 K 및 X를 알게 되면 표1과 같이 최 상위 비트(the most significant bit)가 '1'로 시작하고 이어서 K-(X-1)개의 '0' 이 채워지고 이어서 (X-1)개의 '1' 이 연속된 이진수를 생성할 수 있는데 이것이 code-num M에 1:1로 대응하는 BL-alpha 코드이다.

2.2.3 BL-alpha 코드의 디코딩(Decoding)

BL-alpha 코드의 디코딩은 연속된 BL-alpha 코드로 이루어진 이진수에 있어서, 예를 들어, '10111010100101011111101001' 의 경우, '10'을 만날 때마다 이진수를 분리하면, '1011/10/10/100/10/101111/10/1001' 와 같이 8개의 BL-alpha 코드로 분리된다. 아래 그림2를 보면, K는 BL-alpha 코드의 길이-1 로 구할 수 있으며, T는 '1'이 연속된 길이인데 9개이므로 T=9이다. 이때 아래 수식(5)를 이용하여 Code-num M을 역산할 수 있다.

$$CodeNum(M) = \frac{K(K-1)}{2} + T + 1 \quad \cdots\cdots (5)$$

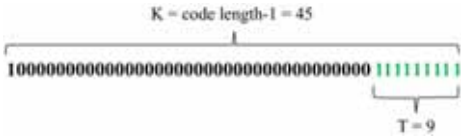

그림 2. BL-alpha 코드의 구조
Fig. 2. Structure of BL-alpha code

그림 2의 BL-alpha코드가 나타내는 code-num은 수식(5)에 따라 $M = \frac{45 \times 44}{2} + 9 + 1 = 1000$임을 알 수 있다.

2.3 유니버설 코드(BL-alpha)를 prefix로 치환한 Exponential Golomb Code(BL-beta)

Unary Code는 현재 다양한 유니버설 코드의 접두코드(prefix code)로 사용되고 있다. Exponential Golomb 코드는 접두코드로서 Unary code 에 이어 접미코드(Suffix code)로서 direct binary가 결합되어 있는 형태이다. Exponential Golomb 코드는 접두코드가 Unary code라는 특성으로 인하여 code-num이 매우 작은 경우에는 1비트만 차지하여 이점이 크지만 큰 경우에 급격히 코드의 길이가 늘어나는 단점이 있다. code-num의 분포가 영상압축에서는 Exponential Golomb 코드가 유리할 수 있지만, code-num의 상한 또는 하한을 예측하기 어렵고, peak 값을 가진 code-num의 출현 확률이 비교적 낮으면서, 특정한 범위 내에서 주로 균등하게 발생하는 금융거래 데이터 또는 IoT 센싱 데이터 등에 사용할 경우 압축효율이 급격히 떨어지는 문제가 있다. 이에 본 연구에서는 Exponential Golomb 코드의 접두 코드부를 BL-alpha로 대체하고 접미 코드로서 direct binary를 결합하여 새로운 유니버설 코드를 새롭게 개발하였고 이를 BL-beta 코드라고 명명하였다[13].

2.3.1 BL-beta 코드 개요와 의미

(1) Exponential Golomb Code의 접두코드 치환하여 생성한 BL-beta코드

표4에서처럼 Unary code로 구성된 Exponential Golomb Code의 접두코드를 BL-alpha 코드로 치환하였는데, 그 방식은 BL-alpha 코드를 그대로 치환하는 것이 아니라, BL-alpha 코드의 reversed form 형태로 치환하여 "01"을 기준으로 접두 코드부와 접미 코드부를 구분할 수 있도록 하였다.

표 4. Exponential Golomb Code의 접두부를 BL-alpha코드로 치환하여 생성한 BL-beta코드
Table 4. BL-beta code from replacement of the Exponential Golomb Code prefix with the BL-alpha

code num	prefix (unary)	Exponential Golomb Code	prefix (Reversed form BL-alpha)	BL-beta Code
1		1	01	010
2	01	010	01	011
3	01	011	001	00100
4	001	00100	001	00101
5	001	00101	001	00110
6	001	00110	001	00111
7	001	00111	101	101000
8	0001	0001000	101	101001
9	0001	0001001	101	101010
10	0001	0001010	101	101011
11	0001	0001011	101	101100
12	0001	0001100	101	101101
13	0001	0001101	101	101110
14	0001	0001110	101	101111
15	0001	0001111	0001	00010000
16	00001	000010000	0001	00010001
...	...	...	...	...

이렇게 생성된 BL-beta 코드와 Exponential Golomb코드간의 code-num에 따른 코드길이를 비교해 보면 그림 3 같이, code-num이 클수록 압축 효과가 더욱 커짐을 알 수 있다. 이때 유의할 부분은 유니버설 코드는 압축하고자 하는 code-num의 분포에 따라 선택 되어야 하는데 H.264 영상압축코덱과 같이 code-num '0'이 매우 높은 빈도로 출현 하도록 한 뒤 유니버설 코드로 압축을 하는 경우는 BL-beta코드보다는 Exponential Golomb 코드가 더 효과적일 수 있다. 본 연구에서는 code-num의 분포가 일정 범위 내에서는 비교적 균등하게 발생하면서도 상한과 하한 값을 특정하기 어려우면서도 이러한 특이값이 희귀하게 발생하는 수치 데이터인 예를 들어, 증권 거래 시 주식 거래량 데이터, 각종 거래 데이터, IoT device의 측정 데이터 등에 적절한 알고리즘으로 판단된다.

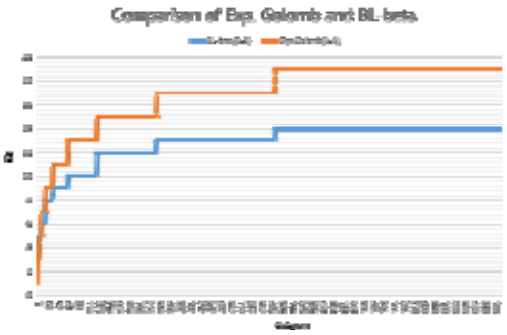

그림 3 Code-num에 따른 BL-beta코드와 Exponential Golomb코드의 코드길이 비교
Fig. 3. Comparison of code length between BL-beta code and Exponential Golomb code according to Code-num

2.3.2 BL-beta 코드 엔코딩

(1) 접두코드 BL-alpha코드 엔코딩

Code-num Z에 대하여, 1:1로 대응하는 BL-beta 코드는 먼저, 아래 수식(6)을 통해 M을 구하여야 한다. 이때 M은 BL-beta의 접두코드를 계산하기 위한 BL-alpha 코드의 code-num을 뜻한다. 수식(6)에서 S는 direct binary로 구성되는 접미 코드(suffix code)의 최소의 차지하는 비트 수이다. 표4에서의 경우, S=1 인 경우로서 접미 코드가 S=1이므로 1비트에서 부터 시작하여 접두코드가 바뀔 때마다 1비트씩 증가하는 경우이다. S=n 이라면 n비트에서 시작하여 1비트씩 증가하는 경우를 뜻한다.

$$M = \left\lceil \log_2 \left(\frac{Z + 2^S}{2^S} \right) \right\rceil \quad \ldots\ldots (6)$$

S 및 M을 알게 되면, BL-beta코드의 접미코드의 direct binary는 M+(S-1) bit의 크기로결정 된다. 후술하겠으나, S 및 M 을 알게 되면 접두 코드에 이어지는 접미코드의 비트 길이를 계산할 수 있다는 점은 BL-beta코드들이 연속되어 있을 때에도 유일 복호성을 유지하기 위하여 매우 중요하다. 이와 같이 M을 구하게 되면 2.2.2 절에서 사용된 수식(1)~(5)에 따라 K 및 X를 구하여 BL-alpha 코드를 생성한 뒤, 이의 reversed form을 BL-beta코드의 접두코드로 한다. reversed form을 접두코드로 사용해야 하는 이유는 예를 들어, 접두 코드가 BL-alpha 코드의 reversed form 이 아닐 경우 송신 측에서 '1011'을 접두코드로 하여 전송하였으나, 수신 측에서는 아래 그림 4와 같이 '10111' 또는 '1011' 와 같은 형태로 해석할 수 있는 모호성(ambiguity)이 발생하여 올바로 디코딩을 할 수 없는 문제가 발생한다.

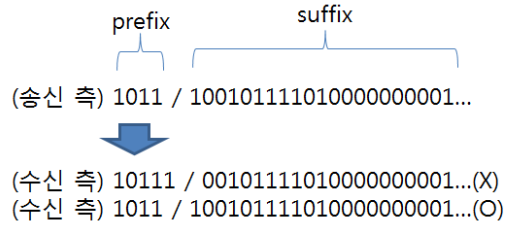

그림 4. Prefix로 쓰인 BL-alpha 코드의 ambiguity 문제
Fig. 4. Ambiguity problem of BL-alpha code used as prefix

따라서 아래 그림 5와 같이 '1011'을 reversed form 인 '1101'로 prefix를 만들어서 전송하면, MSB(the most significant bit)에서부터 LSB(the least significant bit) 방향으로 '01'을 처음 만날 때 '01'까지를 포함하여 prefix로 유일하게 분리할 수 있으며, reversed BL-alpha 코드의 특성상 '01'을 만나기 전에는 '01'이 절대로 나오지 못하는 특성이 있기 때문에 유일 복호성을 유지할 수 있게 된다[13].

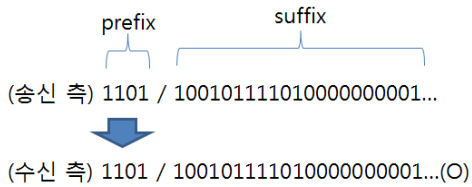

그림 5. Prefix로 쓰인 BL-alpha 코드의 reversed form 을 이용하여 유일 복호성 확보
Fig. 5. Ensure unique decodability by using reverse d form of BL-alpha code as prefix

(2) 접미코드 Encoding

이어서 S+(M-1)비트의 접미코드를 이루는 direct binary는 아래 수식으로 구한 뒤, S+(M-1)비트로 하기 위하여 '0'을 left padding하여 생성한다.

$$suffix = Z - 2^s \times (2^{(M-1)} - 1) - 1 \cdots\cdots (7)$$

(3) Encoding 사례

Exponential Golomb Code 는

[N개의 zero 비트][1][N bit 길이의 direct binary]로 구성되어 전체적으로 2N+1 비트 길이를 가지며, N은 다음수식으로 계산된다[14].

$$N = \lfloor \log_2(codenum + 1) \rfloor \cdots\cdots (8)$$

Code-num이 1,000,000 일 때 N=19이며 이때 전체 Exponential Golomb 코드의 길이는 39비트가 된다. 반면, 동일한 경우에 BL-beta를 엔코딩 하기 위해 수식 (6) 에 따라 M을 구하면 아래와 같다.

$$M = \left\lceil \log_2\left(\frac{1000000 + 2^1}{2^1}\right) \right\rceil = 19$$

다음으로, 수식(3) 및 (4) 에 따라 K 및 X는 아래와 같이 계산된다.

$$K = \left\lfloor \frac{1 + \sqrt{1 + 8 \times 19}}{2} \right\rfloor = 6,$$

$$X = 19 - \frac{6(6-1)}{2} = 4.$$

이렇게 생성된 접두코드 BL-alpha는 7비트임을 K로부터 알 수 있고, 접미코드는 S+(M-1)비트이므로, 19비트임을 알 수 있다. 접미코드가 나타내는 direct binary는 수식(7)에 따라 $1000000 - 2^1 \times (2^{(19-1)} - 1) - 1 = 475713$이고, 이 값을 19비트로 표현하면 다음과 같다. '1110100001001000001' 따라서, 접두코드와 접미코드를 결합하면 '1110001 / 1110100001001000001' 라는 26비트의 BL-beta 코드가 생성 된다[1]. 이때 1110001 은 BL-alpha의 reversed form이며 나머지는 19비트의 접두코드 부분이다. 이는 1,000,000 이라는 code-num 기준으로 Exponential Golomb code보다 34% 더 압축된 것임을 알 수 있다.

2.3.3 BL-beta 코드 디코딩(Decoding)

BL-beta코드로부터 code-num Z를 다시 구하는 것이 디코딩의 목적이며, S는 정확한 디코딩을 위해 엔코딩 및 디코딩 측에서 동일한 값으로 설정되어 있어야 한다. 본 연구에서는 S=1로 두고 실험을 진행하였다. Z는 수식(7)로부터 변형된 아래의 수식 (9)를 통해 구해지는데

$$Z = suffix + 2^s \times (2^{(M-1)} - 1) + 1 \cdots\cdots (9)$$

S는 알고 있으므로, M 과 suffix를 알면 BL-beta의 code-num Z를 구할 수 있다.

(1) M을 구하기 위해서는 BL-beta코드의 접두 코드부인 reversed form 의 BL-alpha 코드를 분리한다. 방법은 그림 6와 같이 BL-beta 코드의 최 상위 비트(the most significant bit)에서 최하위 비트(the least significant bit)방향으로 '01'을 처음 만날 때 최 상위 비트에서 '01'까지가 reversed form 의 BL-alpha 코드이다. 분리된 BL-alpha 코드의 code-num을 M 이라고 하고, 2.2.3에서 서술한 방법대로 구한다. 다음으로 접두 코드 이후로부터 S+(M-1)비트의 direct binary 가 BL-beta 코드의 접미 코드이며, 이 direct binary의 십진수 변환한 값이 suffix이다. 예를 들어, BL-beta 코드 '11100011110100001001000001' 에 있어서 위 규칙에 따르면, '1110001' 이 reversed form 의 BL-alpha 코드이며, 이의 normal form은 '1000111' 이며 2.2.3 절에서 서술한 방식에 따라 BL-alpha 코드를 decoding하면 K=6이고 T=3이므로 M=19이다. 따라서 접두 코드 이후

1) '/' 는 개념적으로 reversed form 의 BL-alpha인 prefix 와 suffix 를 구분하기 위한 표시임

S+(M-1)=19 비트의 '1110100001001000001' 가 접미 코드이며, 십진수로는 475713 이므로, BL-beta 코드의 디코딩 값(Z)는 수식 (9)에 따라

$475713+2^{1}\times(2^{(19-1)}-1)+1=1000000$와 같이 계산된다.

그림 6. BL-beta코드의 decoding 과정
Fig. 6. The decoding process of BL-beta code

2.3.4 BL-beta 코드의 압축 효율 비교

code-num이 클 경우에 따른 압축 효율은 아래 표5와 같다.

표 5. 큰 code-num 수치에 대한 BL-beta코드 압축 성능
Table 5. Compression performance of BL_beta code in case of big code-num(Z)

code num Z (decimal)	log of Z	BL - beta	Bit length
10	1	101011	6
100	2	1101100101	10
1000	3	11001111101001	14
10000	4	1100010011100010001	19
100000	5	00000011000011010100001	23
1000000	6	11100011110100001001000001	26
…	…	…	…

표5의 결과와 Fibonacci 코드, Elias Gamma 코드, Elias delta 코드와 압축 효율을 비교하면 아래 그림7 와 같이 BL-beta 코드는 code-num 이 매우 큰 경우에도 다른 유니버설 코드보다 더 높은 압축 효율을 나타내고 있음을 알 수 있다.

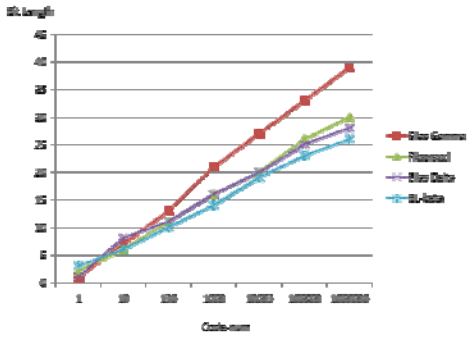

그림 7 Code-num에 따른 BL-beta코드와 다른 유니버설코드들과의 압축률 비교
Fig. 7. Comparison of compression ratio between BL-beta code and other universal codes according to code-num

따라서 실시간 무 손실 압축에서 code-num이 매우 큰 값이 들이 특이적으로 발생하면서도 대다수는 일정한 범위 내에서 발생하는 통계 특성을 가진 데이터의 압축에 있어서 기존 유니버설 코드들보다 더 좋을 성능을 나타낼 수 있을 것으로 예상 되었다[15-16].

2.3.5 실제 실시간 무손실 data 압축 전송을 통한 BL-beta 코드의 필요성

연구를 통해 새롭게 개발한 BL-beta코드를 이용하여 실시간으로 data 전송을 하는 경우 데이터 전송의 효율을 높일 수 있는 데, 구체적으로 아래 그림8과 같이 실제 국내 코스닥 상장사 B사의 2018년 8월 17일 기준 실시간 주식 거래량 데이터인 tick 거래량 data처럼 아주 짧은 시간 간격동안 발생하는 실시간 거래 데이터를 증권사 서버로부터 사용자 PC로 전송해야하는 상황의 경우 사전에 발생할 수 있는 최대 거래량의 peak 값을 고려하여 fixed length bit로 데이터를 전송한다고 했을 때, 그림6에서의 tick 단위에서 최대 거래량인 43,310주를 표현할 수 있는 16 bit로 모든 수치를 처리하여 전체 25054 개의 tick 에 대한 데이터이므로 400,864 bit의 데이터로 실시간 전송을 해야 한다. 게다가 실제로는 peak거래량이 어느 정도 크기인지 예측할 수 없기 때문에, fixed length bit로 전송 시 전송오류(overflow)의 위험에 항시 노출되게 된다. 또한 tick과 tick사이의 시간간격이 매우 조밀하고 불규칙하고, 발생하

는 거래량 수치에 대한 정보가 불확실하기 때문에 사전을 생성해야하는 huffman coding등 전통적인 엔트로피 부호화 알고리즘을 사용하기가 어렵다.

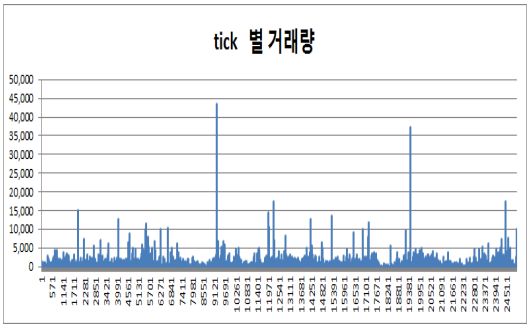

그림 8. 25054건의 실시간 tick 별 거래량 데이터
(가로축 : tick 순번, 세로축 : tick별 거래량)
Fig. 8. 25054 real-time ticks per transaction data
(Horizontal axis: tick sequence number, vertical axis: transaction amount value)

그런데, 위의 tick 별 거래량 데이터를 BL-beta코드 및 Exp. Golomb 코드로 압축 전송시 전송 데이터 량을 비교해 보면 아래 그림 9과 같다. 위와 같은 거래량 데이터를 전송 시 fixed length 로 보내는 경우보다는 BL-beta로 보낼 경우 49.5%의 까지 트래픽을 절감할 수 있음을 알 수 있다. 다양한 응용분야에서 정수데이터 전송 시 사용되는 Exponential Golomb 코드의 경우[17-18] 보다도 16.6% 더 작은 크기로 위의 tick 별 거래량 데이터를 전송할 수 있음을 알 수 있다.

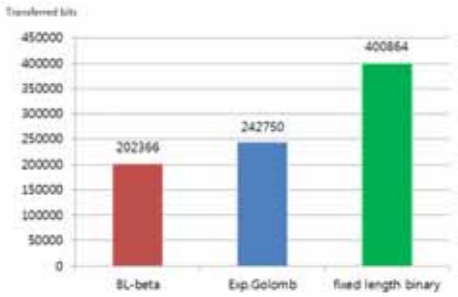

그림 9. 25054건의 실시간 tick 거래량 데이터를 BL-beta 코드, Exponential Golomb code, fixed length binary 로 전송 시의 전송 데이터 비트 수
Fig. 9. Number of transmitted data bits when transmitting 25054 real-time tick transaction data with BL-beta code, Exponential Golomb code and fixed length binary

3. 결론

본 연구에서는 저자가 새롭게 제안한 BL-beta코드를 이용하여 IoT 센싱 데이터, 금융거래 데이터 등 실시간 무손실 데이터 전송 압축에서 Exponential Golomb 코드와 비교하여 더욱 높은 압축률을 나타내었고, 그 구현 알고리즘 또한 간단하여, 하드웨어적으로 또는 소프트웨어적으로 구현이 용이할 것으로 생각되었다. 매우 짧은 간격으로 갱신되면서, 매우 편차가 큰 상한 값 또는 하한 값이 희귀하게 발생하면서도, 비교적 일정한 범위 내에 데이터가 집중해서 발생하는 발생 분포 유형에서 무 손실 압축 전송을 위하여 높은 압축률을 나타냈다. 구체적으로는 주식/환율 거래와 같은 각종 금융거래 데이터, IoT sensing data, 과학 계산용 측정데이터, 의료영상 데이터등 실제 산업분야에도 사용할 수 있을 것으로 판단되며, 추가 연구를 통해 매우 거대한 크기의 정수 데이터에 있어서도 BL-beta 코드와 비교하여 압축 효율을 연구할 예정이다.

REFERENCES

[1] Y. -S. Ho and H. Jin, “Information theory and lossless encoding method”, pp. 157-277, Seoul: Do-Yang Sa, 2009.
[2] D. A. Huffman, “A method for the construction of minimum redundancy codes,” in Proc. I.R.E., pp. 1098-1101, 1952.
[3] I.J. Hwang, Y.S. Lee, S.G. Lee, and I.J. Yeom,

"Naming-based Packet Header Compression Technique for Internet-of-Things", The Fall Conference of Korea Institute of Communication Science, 2015.
[4] Li X.; Hu G.; Gao S. A novel B-mode ultrasound image compression method based on beam forming data. in Proc. 20th Annu. Int. Conf. IEEE Engineering in Medicine and Biology Society, no. 3, pp. 1274-1276, 1998.
[5] Cheng P.; Shen C.; Li P. MPEG Compression of Ultrasound RF Channel Data for a Real-Time Software-Based Imaging System. IEEE Trans. Ultrason. Ferroelect. Freq. Contr. , no. 59, pp. 1413-1420, 2012.
[6] Lok U.; Fan G.; Li P. "Lossless Data Compression for Improving the Performance of a GPU-Based Beamformer", Ultrason. Imaging, no. 37, pp. 135-151, 2015.
[7] J. H. Kim, "Binary data compression and decompression apparatus and method thereof", , Korea Patent 10-1467684-0000, 2014.
[8] J. H. Kim, "Manchester coding of compressed binary clusters for reducing IoT healthcare device's digital data transfer time", The Journal of Korea Institute of Information, Electronics, and Communication Technology, Vol 8, No. 6, pp.460-469, 2015.
[9] S. Nakatsuka, T. Hamabe, Y. Takeuchi, and M. Imai, "An efficient lossless data compression method based on exponential-Golomb coding for biomedical information and its implementation using ASIP technology," IEEE Biomed. Circuits Syst. Conf. (BioCAS), pp. 382-385, Dec. 2013.
[10] S. W. Golomb, "Run-length encodings," IEEE Trans. Inf. Theory, vol. 12, no. 3, pp. 399-401, Jul. 1996.
[11] L. Li and K. Chakrabarty, "On using exponential-Golomb codes and subexponential codes for system-on-a-chip test data compression," J. Electron. Test., Vol. 20, No. 6, pp. 667-670, 2004.
[12] S. Kak, "Generalized unary coding," Circuits Syst Signal Process, Vol. 35, No. 4, pp. 1419-1426, 2016.
[13] Kim, J.H., Yeo, S., Kim, J.W., Kim, K., Song, T., Yoon, C., Sung, J. "Real-Time Lossless Compression Algorithm for Ultrasound Data Using BL Universal Code", Preprints 2018080309, 2018.
[14] Y. -S. Ho and S. -H. Kim, "Understanding and analysis of H.264/AVC algorithm and program.", pp. 197-198, Seoul: Do-Yang Sa, 2009.
[15] P. Elias, "Universal codeword sets and representations of the integers," IEEE Trans. Inf. Theory, vol. 21, no. 2, pp. 194-203, 1975.
[16] P. Fenwick, "Elias omega and Even-Rodeh codes," in Lossless Compression Handbook, K. Sayood, Ed. California: Elsevier, pp. 59-60, 2003.
[17] S. Nakatsuka, T. Hamabe, Y. Takeuchi, M. Iami, "An Efficient Lossless Data Compression Method based on Exponential-Golomb Coding for Biomedical Information and its Implementation using ASIP Technology", 2013 IEEE Biomedical Circuits and Systems Conference(BioCAS), 2013.
[18] E. Chua and W.-C. Fang, "Mixed Bio-Signal Lossless Data Compressor for Portable Brain-Heart Monitoring Systems", IEEE Transactions on Consumer Electronics, vol. 57, no. 1, pp. 267-273, 2011.

저자약력

김 정 훈(Jung-Hoon Kim) **[정회원]**

- 2002년2월 : 서울대학교 약학과 약학사
- 2010년2월 : 서울대학교 보건대학원 보건학 석사
- 2013년2월 : 서울대학교 보건대학원 보건학 박사수료
- 2016년3월~현재 : 바이너리랩(주) 대표이사

<관심분야>

의료정보, 약학정보, 데이터압축, 머신러닝