비접촉 IC카드 리더로의 UserID 전송을 위한 엔코딩 기법

조도연°, 박명순
고려대학교 컴퓨터과학기술대학원
bubugi1@hanafos.com°, myongsp@ilab.korea.ac.kr

Encoding Method for Transfer UserID to Contactless IC Card Reader

Do-Yeon Cho°, Myong-Soon Park
Graduate Schools of Computer Science and Technology, Korea University

요 약

최근 지하철이나 버스 등의 교통수단을 이용하기 위해 비접촉식 카드를 이용한 IC카드 Reader가 많이 사용되고 있다. 이러한 IC카드 Reader는 ID를 인증하기 위한 UserID를 Reader내부에 저장하고 있으며, UserID가 바뀔 때 마다 상위 시스템(서버)으로부터 전송받아 최신의 UserID를 가지고 있어야 한다.

교통시스템에서 사용되는 통신매체는 환경의 특수함으로 인해 낮은 전송속도를 이용하여 통신하고 있고, 그로인해 UserID를 전송할 때 시간이 많이 소모된다. UserID를 전송하는 동안에는 IC카드 Reader의 사용이 제한되기 때문에 UserID는 IC카드 Reader가 사용되지 않는 시간에 전송하고 있다. 많은 양의 UserID를 IC카드 Reader에 빠르게 전송하기 위해 UserID의 특성을 이용한 엔코딩(encoding) 기법을 적용하여 UserID의 크기를 줄여 전송함으로써 전송시간을 줄이는 방법을 제안하였다.

1. 서 론

최근 비접촉 IC카드를 사용하는 교통카드 시스템은 사용의 편의성, 데이터 관리의 정확성, 업무의 효율성 및 서비스 향상 등의 장점이 있어 많이 사용되고 있다.[1] 이러한 시스템에서 인증에 사용되는 UserID는 신용카드 번호로 상위 시스템으로부터 RS-485를 사용하여 38.4Kbps로 전송받아 IC카드 Reader 내부의 Flash Memory에 저장하게 되며, UserID는 100~200만개로 12~16Mbyte의 저장 공간을 필요로 한다.[2][3]

교통카드 시스템에서 사용하는 UserID는 사용자의 카드 분실, 카드의 갱신 등의 이유로 자주 변경된다. 변경된 UserID는 IC카드 Reader에 실시간으로 전송되어야 하나 낮은 전송속도로 전송에 걸리는 시간이 길고 전송시 Reader의 사용이 제한되기 때문에 Reader가 사용되지 않는 시간에 전송하고 있다.

IC카드 Reader는 인증 시 빠른 UserID 검색을 위해 binary search를 사용하며, 이를 위해 상위 시스템(서버)에서는 오름차순으로 정렬된 UserID를 전송한다.

UserID의 전송 시간을 줄이기 위해선 고속의 통신을 이용하거나 UserID의 크기를 줄여야 한다. 고속의 통신을 이용하기 위해선 많은 비용을 투자하여 하드웨어를 교체하여야 하지만 UserID를 전송할 때 엔코딩하여 UserID의 크기를 작게 만들어 전송하게 되면 하드웨어의 교체 없이 전송시간을 줄일 수 있다.

본 논문에서는 오름차순으로 정렬된 UserID를 비접촉 IC카드 Reader로 빠르게 전송하기 위해 UserID의 크기를 줄일 수 있는 엔코딩 방법을 제안한다. 2장에서는 기존 엔코딩 방법에 대하여 설명하고, 3장에서는 UserID의 특성을 이용한 엔코딩 방법을 제안하고 실험조건을 설정하였다. 4장은 기존 방법들과 비교한 결과를 나타내며, 5장에서 결론을 맺는다.

2. 기존 엔코딩 기법

허프만 부호기(Huffman coding)는 데이터 내의 각 문자에 대한 발생빈도를 조사하여 자주 나타나는 문자에는 보다 짧은 부호를, 잘 나타나지 않는 문자에는 긴 부호를 할당함으로써 압축 후의 부호의 길이를 원래 길이보다 축소시킬 수 있는 통계적 특성을 이용한 방법이다.[4]

런랭스 부호기(Run length coding)는 연속하는 "0" 또는 "1"의 개수를 하나의 부호로서 나타내는 방법이다. 연속하여 같은 값을 가지는 데이터가 있다면, 그 길이(run length)를 이용하여 부호화하는 방법이다. [5]

데이터에서 "0"이나 "1"이 연속적으로 존재하는 부분이 길 때에는 데이터의 크기를 줄이는데 효과적이지만, 데이터의 종류가 수시로 변하는 경우에는 압축 효율이 저하된다.

램펠지브 부호기(Lempel-Ziv coding)는 가장 많이 사용되는 압축 기술 중의 하나로 동적인 사전 부호화 기법으로서 동적인 사전 갱신을 통해서 입력 심벌들의 열인 현재의 메시지를 이전의 사전에서 발생된 참조 엔트리의

열로서 부호화되는 방법이다. 일반적으로 램펠지브 부호화 알고리즘은 입력 데이터를 사전에서 반복적으로 검색하여 사전에 존재하지 않는 데이터 열을 사전에 등록하고, 존재하는 경우는 그 사전의 주소와 길이를 전송하는 방법이다.[6]

3. UserID 특성에 따른 엔코딩 기법

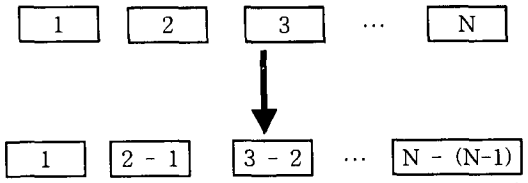
3.1. UserID의 데이터 특성

비접촉 IC 카드는 카드의 키값과 관계없이 4byte의 manufactor serial 번호를 가지고 있으며, 이는 카드의 데이터를 읽고 쓸 때 마다 사용되는 번호이다. 교통카드 시스템에서 사용되는 UserID인 카드 번호는 binary type으로 변경하면 최대 어떤 수라도 4byte안에 표현이 가능하여 현재 4byte로 표현하여 사용 중이다.[2] 그렇기 때문에 UserID는 4byte로 사용하기로 한다.

Reader에서는 내부의 UserID 저장소인 Flash memory로부터 UserID를 검색하기 위해 Binary Search를 사용하며, 이를 위해 정렬된 UserID가 필요하다. Reader에서 사용되는 CPU는 낮은 성능의 것으로 UserID의 정렬을 Reader에서 하게 되면 많은 시간이 소모되며, 같은 UserID 데이터를 이용하는 다른 Reader에서도 같은 정렬 작업을 하게 된다. 또한 정렬 작업을 하는 동안에는 Reader의 사용이 제한된다. Reader의 효율적인 사용을 위하여 높은 성능의 CPU를 사용하는 상위 시스템(서버)은 오름차순으로 정렬된 UserID를 전송한다.

3.2. UserID의 엔코딩

현재는 4byte의 오름차순으로 정렬된 UserID를 그대로 Reader로 전송하고 있으나 이를 엔코딩 하여 작은 byte로 표현하는 방법을 제안한다. [그림1]에서와 같이 첫번째 UserID는 그대로 사용하고 두번째 UserID부터는 앞의 UserID와의 크기의 차로 UserID를 표현하여 크기를 줄이는 방법이다.



[그림1] 엔코딩의 개요 (각 숫자는 순서를 나타냄)

UserID는 같은 것이 존재하지 않고, 오름차순으로 정렬되어 있기 때문에 뒤에 나오는 UserID는 반드시 앞의 UserID보다 크다. 따라서 항상 양의 정수로 표현되며, 그 크기에 따라 표현될 Byte 수가 결정된다. 두 UserID의 차이가 최소일 경우는 1byte로 표현이 가능하며, 최대일 경우는 4byte로 표현이 가능해진다. 그러나 차이가 1~4byte 중 몇byte로 표시될지 모르기 때문에 첫 번째 byte의 상위 3bit는 한 개의 UserID를 표현하기 위한 byte수를 나타내는 것으로 사용되며, 만일 1byte로 표현될 경우 상위 3bit는 byte를 표현하기 위한 bit이므로 나머지 5bit 범위의 숫자만 표현이 가능하다. 5bit의 범위를 벗어나게 되면 1byte를 추가하여 13bit로 표현 할 수 있는 수만큼의 차이가 나는 UserID를 표현 할 수 있다. 29bit 이상 차이가 나는 UserID를 표현하기 위해선 5byte로 표현되며, 첫 byte의 상위 3bit 는 byte의 크기만을 나타내고 뒤에 5bit는 사용되지 않는다. 이 경우 1byte가 원래보다 커지게 되나 최대 7가지 경우 밖에 존재하지 않으므로 전체적으로 UserID들의 크기를 줄이는데 크게 영향을 끼치지 않는다.

앞의 UserID와 뒤의 UserID의 크기의 차에 따라 표현되는 Byte수는 5가지로 구분되며 [표1]과 같다.

[표1] 크기 차에 따른 Byte 변화

상위3bit	byte수	크기 차	경우의 수
000	1	0x00000001 - 0x0000001F	4,294,967,295
001	2	0x00000020 - 0x00001FFF	134,217,727
010	3	0x00002000 - 0x001FFFFF	524,287
011	4	0x00200000 - 0x1FFFFFFF	2,047
100	5	0x20000000 - 0xFFFFFFFF	7

예를 들어 0x10000020, 0x10000034, 0x10005678 인 세 개의 UserID를 전송한다고 하자. 첫 번째 ID인 0x10000020은 그대로 4byte로 표현되고, 두 번째 ID인 0x10000034는 자신의 ID와 자기 앞의 ID의 차로 표현하여 0x10000034와 0x10000020의 차인 0x14로 표현된다. 0x14는 상위 3bit를 제외한 나머지 5bit 로 표현이 가능하여 1byte로 엔코딩 되며, 1byte이기 때문에 상위 3bit는 000으로 나머지 5bit는 두 번째 ID 첫 번째 ID의 차인 0x14 (이진수 1 0100)로 표현되어 0x14로 엔코딩된다. 세 번째 ID인 0x10005678은 두 번째 ID인 0x10000034와의 차인 0x5644로 표현이 가능하며, 이는 엔코딩 후 byte를 나타내는 상위 3bit를 제외한 나머지 15bit로 표현이 가능하여 상위 3bit는 010으로 3byte를 나타내고 나머지 15bit는 세 번째 ID와 두 번째 ID와의 차인 0x5644 (이진수 0 0000 0101 0110 0100 0100)로 표현되어 3byte 0x405644로 된다.

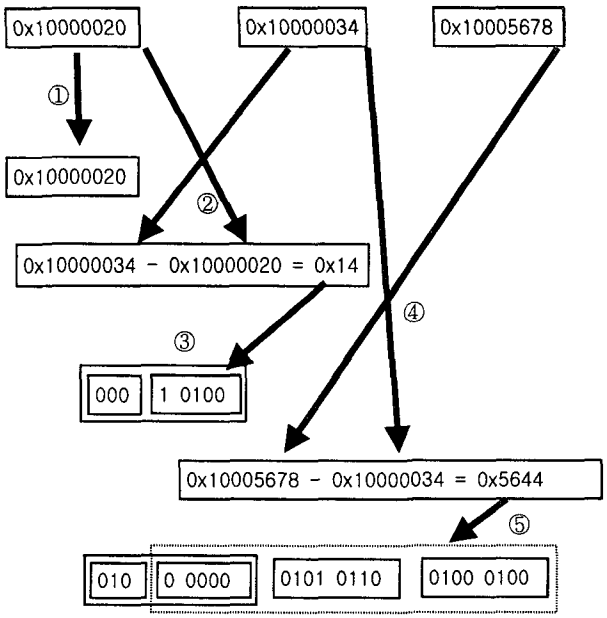
엔코딩을 하지 않은 3개의 UserID 12byte는 엔코딩을 하여 8byte로 표현이 가능하게 되었다. [그림2]는 위의 설명 과정을 보여준다.

3.3. 실험조건

본 논문에서는 데이터를 보내는 쪽의 엔코딩 시간과 받는 쪽의 디코딩(decoding) 시간을 제외한 통신 매체를 통한 물리적 전송 시간만을 기준으로 실험하였으며 이를 위해 엔코딩 전의 file 크기와 엔코딩 후의 file 크기를 비교함으로서 전송시간 단축을 보이는 것으로 하였다.

데이터 특성을 반영하여 오름차순으로 정렬된 4byte의 UserID를 사용하였으며, UserID의 개수는 100만개로 고정하여 4Mbyte의 UserID file을 사용하였다.

데이터의 공정성을 위하여 제안된 엔코딩 기법에 있어서의 압축률 변화에 따른 다양한 UserID 데이터를 적용하였다.



[그림 2] 엔코딩 과정

4. 성능평가

UserID file은 제안된 엔코딩 기법을 사용 시 압축율의 변화에 따라 이루어 졌으며, 그에 따른 구분은 [표1]과 같다.

[표1] File 의 구분에 따른 데이터의 종류

File	데이터의 종류
1	엔코딩 후 1byte로 표현 가능한 UserID 100만개
2	엔코딩 후 2byte로 표현 가능한 UserID 100만개
3	엔코딩 후 3byte로 표현 가능한 UserID 524,229개 엔코딩 후 1byte로 표현 가능한 UserID 475,771개
4	엔코딩 후 4byte로 표현 가능한 UserID 2,047개 엔코딩 후 1byte로 표현 가능한 UserID 997,953개
5	엔코딩 후 5byte로 표현 가능한 UserID 7개 엔코딩 후 1byte로 표현 가능한 UserID 999,993개
6	엔코딩 후 5byte로 표현 가능한 UserID 7개 엔코딩 후 2byte로 표현 가능한 UserID 999,993개

[표2]에서는 100만개의 UserID를 엔코딩 후 전송해야 할 Byte 수를 나타낸 것으로 UserID의 압축률을 나타낸 것이며, 제안된 엔코딩 방법이 기존의 데이터 압축에 사용되는 방법보다 우수한 것을 알 수 있다.

5. 결론

본 논문에서는 교통카드 시스템에서 사용되는 비접촉 IC 카드 리더로 UserID를 전송할 때 UserID의 크기를 줄이는 엔코딩 기법에 대하여 제안하였다.

기존의 통신매체의 변경 없이 전송효율을 높이기 위해 전송되는 데이터의 특성을 이용하는 엔코딩 기법을 제안, 테스트하였다. 압축률은 원본크기 / 엔코딩 후 크기로 표현하였다. 실험결과 UserID의 크기는 원본을 그대로 전송할 때 보다 평균 265.06% 압축되었으며, 기존에 많이 쓰이는 엔코딩 기법 중 가장 높은 압축 성능을 나타낸 LZ77보다 211.51% 더 높은 압축 성능을 나타내었다.

본 논문에서 제안한 엔코딩 기법은 전송 시 에러(Error)가 발생되면 이후의 데이터는 전부 잘못된 데이터가 되는 단점이 있어 보완되어야 하며, 차후 좀더 나은 성능의 엔코딩 방법을 찾아야 보아야 할 것이다.

[표2] 엔코딩 후의 File 크기 및 압축률 비교
(단위 : byte, 압축률(%)은 높을수록 좋음)

File	허프만	RLE	LZW	LZ77	제안방법
1	3,537,269	4,031,459	5,040,842	3,145,170	1,000,003
	113.08%	99.22%	79.35%	127.18%	400.00%
2	3,154,119	4,031,040	3,682,799	3,125,668	2,000,002
	126.82%	99.23%	108.61%	127.97%	200.00%
3	3,252,390	4,031,201	4,425,518	3,125,352	2,048,461
	122.99%	99.23%	90.38%	127.99%	195.27%
4	3,203,274	4,031,194	4,017,875	3,125,029	1,006,144
	124.87%	99.23%	99.56%	128.00%	397.56%
5	3,537,273	4,031,459	5,040,842	3,145,163	1,000,031
	113.08%	99.22%	79.35%	127.18%	399.99%
6	3,775,237	4,031,420	5,383,934	3,484,589	2,000,023
	105.95%	99.22%	74.30%	114.80%	200.00%
평균	3,409,927	4,031,296	4,598,635	3,191,829	1,509,111
	117.30%	**99.22%**	**86.98%**	**125.32%**	**265.06%**

6. 참고자료

[1]윤영수, "서울시 RF교통카드시스템에 관한 연구", 숭실대 석사논문, PP.41-42, 2000.
[2]황인성, "비접촉 스마트카드 단말기의 신용불량자 검출에 관한 연구", 서울시립대 석사논문, pp.2-41, 2002.
[3]황준흠, "RF-IC카드를 이용한 전철자동운임징수 시스템 분석", 영남대 석사논문, pp.16-22, 2002.
[4]차인숙, "혼합 부호화에 의한 압축률 개선에 관한 연구", 부경대 석사논문, pp.11, 2001.
[5]S.Gallager, "Run-length endocing," IEEE Trans. Inform Theory, vol. IT-12, pp.399-401, 1966.
[6]진용선, "실시간 데이터 압축을 위한 Lempel-Ziv 알고리즘의 하드웨어 설계", 한양대 박사논문, pp.17-25, 2000.