인터넷 프로토콜 5장

데이터 송수신 (1)

목포해양대 해양컴퓨터공학과

제 5장 데이터의 송수신

- 5.1 정수 인코딩
- 5.2 메시지 생성, 프레이밍, 그리고 파싱
- 5.3 마무리

들어가기 전

- TCP/IP는 사용자의 데이터를 검사하거나 변경하지 않고 그대로 전송
- 응용 프로토콜은 연속적인 필드로 구성된 메시지 형태로 정의
- 인코딩(encoding)/디코딩(decoding) 또는 파싱(parsing)
- 네트워크 응용 프로그램 개발 방법
  - 사용자가 모든 것을 새로 정의
  - 이미 존재하는 프로토콜 표준을 따르면서 남은 부분 구현

5.1 정수 인코딩

- 정수의 크기
  - 보내려는 정수의 크기를 미리 결정
- 바이트 순서화
  - 1 바이트가 넘는 값을 인코딩하는 경우 어떤 순서로 보낼 것인지(big-endian/little-endian) 상호 결정
- 부호화와 부호 확장
  - 보내는 값에 부호가 있는지(signed) 또는 없는지(unsigned) 결정

정수의 크기 (1)

- C 언어의 정수형 타입들
  - int
  - char
  - short
  - long
- C 언어의 표준
  - 크기에 대한 정의는 없으면서 단지 char형이 short보다 클 수 없고, short는 int보다 클 수 없고, int는 long보다 클 수 없다고만 기술
- 반드시 개발 플랫폼에서 sizeof 연산자를 사용하여 크기를 확인해야 함
  - 이때 sizeof 연산자 결과는 sizeof(char)를 1로 정의

정수의 크기 (2)

- C99언어 표준
  - Option 형태로 자료형에 비트 단위 크기를 나타냄
    - int8_t, int16_t, int32_t, int64_t (uint8_t)
  - 추가적으로 long long 타입도 있음

TestSizes.c

```c
#include <limits.h>
#include <stdint.h>
#include <stdio.h>

int main(int argc, char *argv[]) {
        printf("CHAR_BIT is %d\n\n",CHAR_BIT);         // Bits in a char (usually 8!)

        printf("sizeof(char) is %d\n", sizeof(char));  // ALWAYS 1
        printf("sizeof(short) is %d\n", sizeof(short));
        printf("sizeof(int) is %d\n", sizeof(int));
        printf("sizeof(long) is %d\n", sizeof(long));
        printf("sizeof(long long) is %d\n\n", sizeof(long long));

        printf("sizeof(int8_t) is %d\n", sizeof(int8_t));
        printf("sizeof(int16_t) is %d\n", sizeof(int16_t));
        printf("sizeof(int32_t) is %d\n", sizeof(int32_t));
        printf("sizeof(int64_t) is %d\n\n", sizeof(int64_t));

        printf("sizeof(uint8_t) is %d\n", sizeof(uint8_t));
        printf("sizeof(uint16_t) is %d\n", sizeof(uint16_t));
        printf("sizeof(uint32_t) is %d\n", sizeof(uint32_t));
        printf("sizeof(uint64_t) is %d\n", sizeof(uint64_t));
}
```

바이트 순서화

- Big-endian과 Little-endian
  - Big-endian(high order)
  - Little-endian(low order)
- 인터넷의 바이트 순서 : Big-endian
  - network bytes order
- 호스트의 바이트 순서 : 플랫폼에 따라 다를 수 있음
  - native bytes order 또는 host bytes order
- C 언어 플랫폼에서 제공하는 함수들
  - htons(), ntohs() : 16비트 정수
  - htonl(), ntohl() : 32비트 정수

부호화

- signed 와 unsigned 차이
- 부호 확장(sign extension)
  - int8_t가 int16_t로 확장되는 과정 이해
    - 0100 1110 (78) -> 0000 0000 0100 1110 (78)
    - 1110 0010 (-30) -> 1111 1111 1110 0010 (-30)
  - int8_t가 uint16_t로 확장되는 과정 이해
    - 0100 1110 (78) -> 0000 0000 0100 1110 (78)
    - 1110 0010 (-30) -> 1111 1111 1110 0010 (65506)
- 연산과정의 확장
  - char + char는 char이 아니다!

정수 인코딩을 직접 해보자

- BruteForceCoding.c
  - byte ordering
  - signedness

BruteForceCoding.c (1)

```c
#include <stdint.h>
#include <stdlib.h>
#include <stdio.h>
#include <limits.h>
#include "Practical.h"

const uint8_t val8 = 101; // One hundred and one
const uint16_t val16 = 10001; // Ten thousand and one
const uint32_t val32 = 100000001; // One hundred million and one
const uint64_t val64 = 1000000000001L; // One trillion and one
const int MESSAGELENGTH = sizeof(uint8_t) + sizeof(uint16_t) + sizeof(uint32_t) + sizeof(uint64_t);

static char stringBuf[BUFSIZE];

/*****************************************************************************/
char *BytesToDecString(uint8_t *byteArray, int arrayLength) {
        char *cp = stringBuf;
        size_t bufSpaceLeft = BUFSIZE;
        for (int i = 0; i < arrayLength && bufSpaceLeft > 0; i++) {
                int strl = snprintf(cp, bufSpaceLeft, "%u ", byteArray[i]);
                bufSpaceLeft -= strl;
                cp += strl;
        }
        return stringBuf;
}

```

BruteForceCoding.c (2)

```
27 /*****************************************************************************/
28 // Warning:  Untested preconditions (e.g., 0 <= size <= 8)
29 int EncodeIntBigEndian(uint8_t dst[], uint64_t val, int offset, int size) {
30         for (int i = 0; i < size; i++) {
31                 dst[offset++] = (uint8_t) (val >> ((size - 1) - i) * CHAR_BIT);
32         }
33         return offset;
34 }
35
36 /*****************************************************************************/
37 // Warning:  Untested preconditions (e.g., 0 <= size <= 8)
38 uint64_t DecodeIntBigEndian(uint8_t val[], int offset, int size) {
39         uint64_t rtn = 0;
40         for (int i = 0; i < size; i++) {
41                 rtn = (rtn << CHAR_BIT) | val[offset + i];
42         }
43         return rtn;
44 }
45
```

BruteForceCoding.c (3)

```
46 /*****************************************************************************/
47 int main(int argc, char *argv[]) {
48         uint8_t message[MESSAGELENGTH]; // Big enough to hold all four values
49
50         // Encode the integers in sequence in the message buffer
51         int offset = 0;
52         offset = EncodeIntBigEndian(message, val8, offset, sizeof(uint8_t));
53         offset = EncodeIntBigEndian(message, val16, offset, sizeof(uint16_t));
54         offset = EncodeIntBigEndian(message, val32, offset, sizeof(uint32_t));
55         offset = EncodeIntBigEndian(message, val64, offset, sizeof(uint64_t));
56         printf("Encoded message:\n%s\n", BytesToDecString(message, MESSAGELENGTH));
57
58         uint64_t value =
59                 DecodeIntBigEndian(message, sizeof(uint8_t), sizeof(uint16_t));
60         printf("Decoded 2-byte integer = %u\n", (unsigned int) value);
61         value = DecodeIntBigEndian(message, sizeof(uint8_t) + sizeof(uint16_t)
62                         + sizeof(uint32_t), sizeof(uint64_t));
63         printf("Decoded 8-byte integer = %llu\n", value);
64
65         // Show signedness
66         offset = 4;
67         int iSize = sizeof(int32_t);
68         value = DecodeIntBigEndian(message, offset, iSize);
69         printf("Decoded value (offset %d, size %d) = %lld\n", offset, iSize, value);
70         int signedVal = DecodeIntBigEndian(message, offset, iSize);
71         printf("...same as signed value %d\n", signedVal);
72 }
```

과제

- 본문 TestSizes.c 실행하여 확인
- 본문 BruteForceCoding.c 확인 및 정리
  - 특히 비트 연산자 활용 부분에 주의!