

## 과제 3 추가 설명

16bit ones' complement checksum

# Checksum algorithm

Checksum algorithm (송신측)

```
{  
    패킷 안의 모든 요소(data, acknum, seqnum, checksum) 32bit 으로 맞추  
    합 = 32bit 이 된 모든 요소를 bit-wise 로 합 계산  
    계산 결과 = 합에 wrap around carry 계산  
    checksum = 계산 결과의 1's complement  
}
```

Checksum algorithm (수신측)

```
{  
    수신한 패킷 안의 모든 요소(data, acknum, seqnum, checksum) 32bit 으로 맞추  
    합 = 32bit 이 된 모든 요소를 bit-wise 로 합 계산  
    계산 결과 = 합에 wrap around carry 계산  
    checksum = 계산 결과의 1's complement  
    checksum이 0 일 경우 손상되지 않은 패킷  
}
```

# 16bit Bit wise operation

Bit wise 연산자를 사용하여 연산

integer type , char type 모두 16bit 으로 맞추는 후 bit-wise operation 진행

ex) bit-wise operation

UINT16 Sum = A ^ B

UINT16 Carry = A & B

Char 의 경우 아래의 예시처럼 16bit 으로 맞추는 후 진행 할 수 있음

ex) char data[4] = AAAA

UINT16 sum = (left shift data[0] by 8 bit ^ data[1]) ^ (left shift data[2] by 8 bit ^ data[3])

UINT16 carry = (left shift data[0] by 8 bit ^ data[1]) & (left shift data[2] by 8 bit ^ data[3])

주의 사항 :

bit-wise 연산을 위해 char type 의 형변환이 필요 할 수 있음

# Bit-wise operation 을 사용하는 one's complement addition

수도코드 에 따른 4bit 연산 예시

one's complement addition:

```
{
    UINT16 Sum = 0
    Loop(until 패킷의 모든 요소에 대한 addition 완료 할 때까지)
    {
```

x = Sum

y= 16bit 으로 변환된 패킷의 요소

UINT16 Sum = x bit-wise XOR y

UINT16 carry = x bit-wise AND y

loop(carry != 0)

{

Logic에 따라, Carry의 맨 상위 bit 에 carry bit이 발생하면 곧 wrap around carry 다 발생한 것이므로 wrap around addition을 진행해줘야 하며, carry 안의 최상위 bit이 1인지를 검사하여 이를 알 수 있음

condition(carry bit-wise AND left-most-bit)

{

wrap around carry state

}

left shift carry by 1 bit

x = Sum

y = carry

Sum = x bit-wise XOR y

carry = x bit-wise AND y

Wrap around carry state on

}

condition (wrap around carry state){

unsigned variable 1

Sum = unsigned variable bit-wise addition Sum

}

}

Checksum = bitflip sum

}

left-most-bit = 1000

1011

0110

Bit-wise OP -----

Sum = 1101

Carry = 0010

left-most-bit = 1000

Carry bit-wise AND left-most-bit -----

Carry bit-wise AND left-most-bit = 0000

Left Shift carry by 1bit = 0100

Sum bit-wise OP left shift carry by 1 bit -----

Sum = 1001

Carry = 0100

left-most-bit = 1000

Carry bit-wise AND left-most-bit -----

Carry bit-wise AND left-most-bit = 0000

Left Shift carry by 1bit = 1000

Sum bit-wise OP left shift carry by 1 bit -----

Sum = 0001

Carry = 1000

left-most-bit = 1000

Carry bit-wise AND left-most-bit -----

Carry bit-wise AND left-most-bit = 1000

Left Shift carry by 1 bit = 0000

Sum bit-wise OP left shift carry by 1 bit -----

Sum = 0001

Wrap around bit = 0001

Sum bit-wise OP Wrap around bit -----

Sum = 0000

Carry = 0010

left-most-bit = 1000

Carry bit-wise AND left-most-bit -----

Carry bit-wise AND left-most-bit = 0000

Left Shift carry by 1 bit = 0100

Sum bit-wise OP left shift carry by 1 bit -----

Sum = 0010

Carry = 0000

# checksum

Checksum

패킷안의 모든 요소(data, seqnum, acknum )의 합 의 one's complement

one's complement를 이용한 corruption 판별

Checksum 은 bitwise 연산을 통해 corruption 을 판별

Wrap around carry 가 발생하여도 summation 값이  
(sum) 1001 1101 0011 1001  
1000 0000 0000 0101

일 경우 summation result =0001 1101 0011 1111  
(Checksum received) =1110 0010 1100 0000

Checksum ^ summation result =1111 1111 1111 1111  
(after one's complement)Checksum ^ summation result =0000 0000 0000 0000

패킷 손상되지 않음

# 필수 구현 사항

구현 방식 자유

가산점을 위한 필수 구현 사항

Bit-wise summation 을 통한 acknum, seqnum , data 의 합  
checksum = 합의 One's complement  
패킷 손상 여부 판별