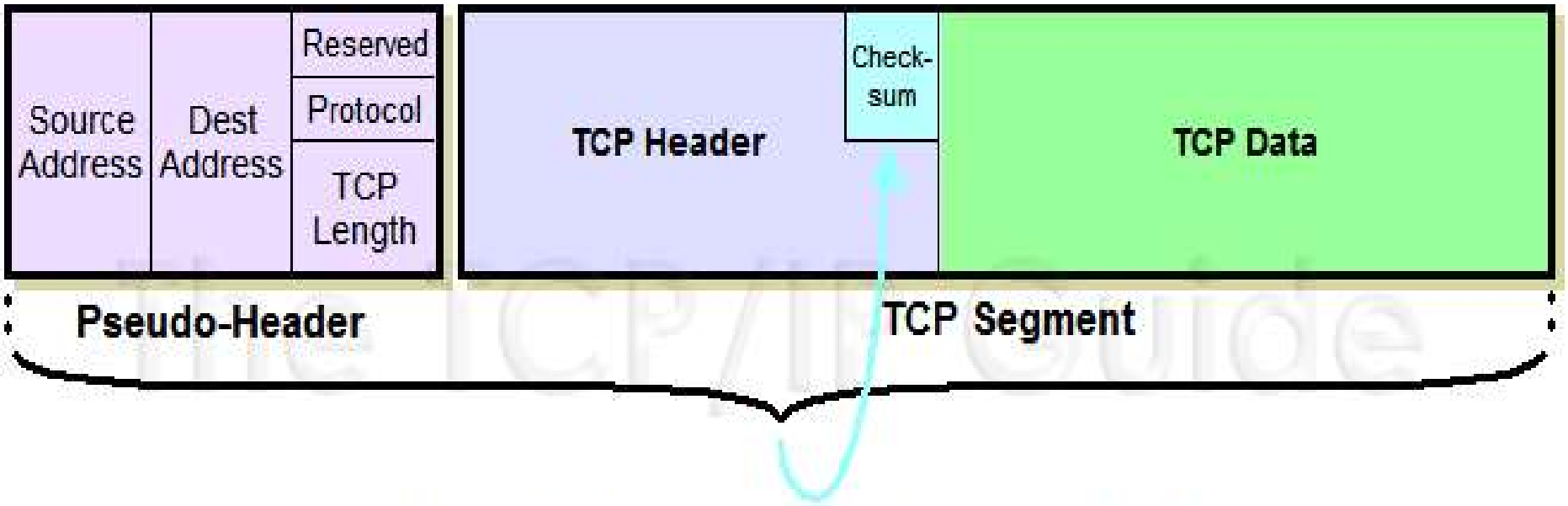
TCP Checksum Algorithm

Gert Pfeifer
gert.pfeifer@inf.tu-dresden.de

TCP Checksum

- used to detect errors after transmitting packet
- in case of error – packet is dropped
  - retransmission is cheaper than repair
- unlike UDP – TCP checksum is never optional

Computing TCP Checksum



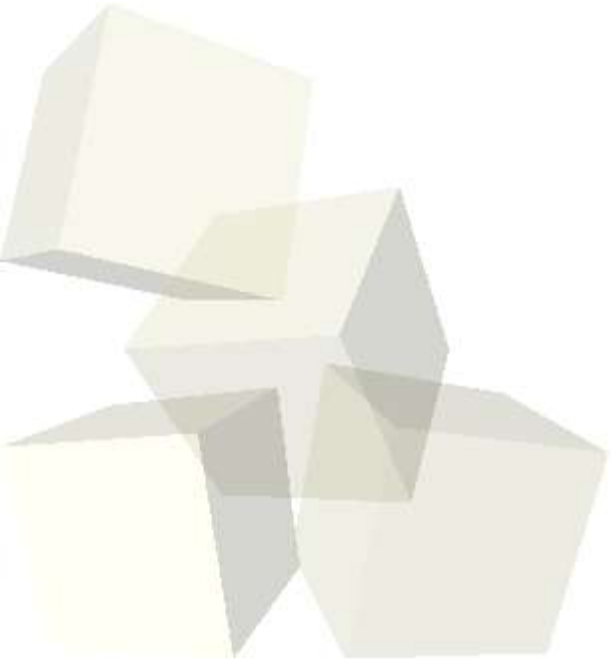
Picture property of
http://www.tcpipguide.com

Chicken and Egg

- TCP Checksum is a part of the fields over which the checksum is calculated
- This field contains
  - Zero ... for calculation
  - Checksum ... for transmission
  - Checksum ... for checking

Calculation Algorithm

- adjacent octets are paired
- checksum field = 0000 0000 0000 0000
- one's complement sum over these fields is calculated
- one's complement negation of this sum is placed into the checksum field



Checking the packet

- adjacent octets are paired
- one's complement sum over these fields is calculated
- If result == 1111 1111 1111 1111 (-0 in one's complement) the check succeeds.
- Otherwise the packet is dropped and will be retransmitted.

Example

- TCP packet contains octets A,B, ...,Y,Z
- [A,B] = A*256 + B

- one's complement sum is
  [A,B] (+) [C,D] (+) ... (+) [Y,Z]

- if number of octets is odd:
  [A,B] (+) [C,D] (+) ... (+) [Z,0]

Properties of one's complement

- Commutative and Associative
  - [A,B] (+) [C,D] = [C,D] (+) [A,B]
  - **([C,D] (+) [A,B])** (+) [E,F]
    == [C,D] (+) **([A,B] (+) [E,F])**

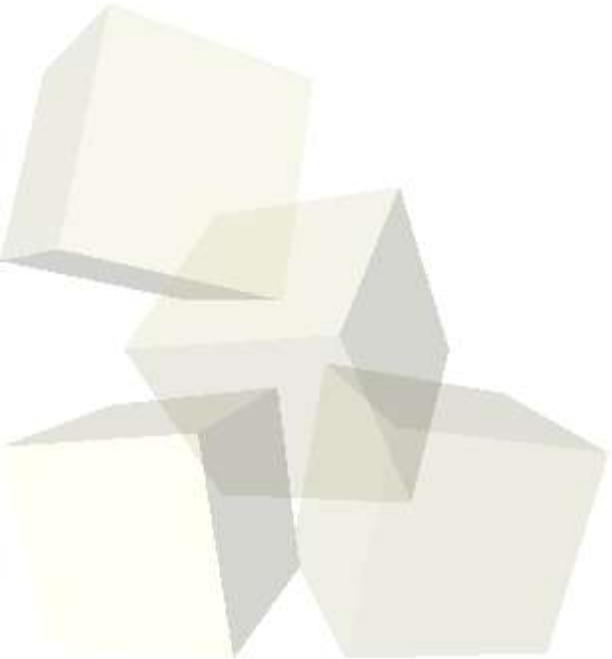
Properties of one's complement

- Byte Order Independance
  - [A,B] (+) [C,D] = [X,Y]
  - [B,A] (+) [D,C] = [Y,X]

  - Checksum is calculated in exactly the same way regardless of the byte order (“little endian” or “big endian”)
  - Example: IBM PC can calculate sum of data that is stored in network byte order without converting byte order before and after calculation

Properties of one's complement

- Parallel Summation
  - more efficient implementations possible on machine with word size that is a multiple of 16 bits (RFC from Sept. 1988)
  - Nowadays 64-bit machine can do 4 16-bit-summations in one step.



Examples

- Normal order:
  16 bit words

```
  0001
  f203
  f4f5
  f6f7
 (0000)
 -----
 2ddf0

  ddf0
     2
  ddf2
```

- Swapped order:
  16 bit words

```
  0100
  03f2
  f5f4
  f7f6
  0000
 -----
 1f2dc

  f2dc
     1
  f2dd
```

Examples

- Byte by Byte:

```
  00   01
  f2   03
  f4   f5
  f6   f7
 ---  ---
2dc  1f0

  dc   f0
   1    2
  dd   f2
```

- Word by Word:
  32 bit words

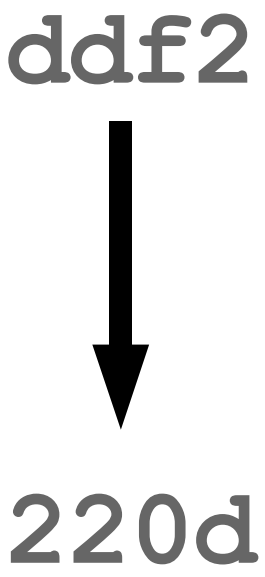
```
 0001f203
 f4f5f6f7
----------
 f4f7e8fa

      f4f7
       e8fa
----------
     1ddf1

      ddf1
         1
      ddf2
```

Calculate Checksum

- checksum is one's complement negation of sum



Using Checksum

- Calculate one's complement over all adjacent octets including checksum.

```
  0001
  f203
  f4f5
  f6f7
 (220d)
 -----
 2fffd

  fffd
     2
 ffff
```

Success ←