

Internet Engineering

241-461

Robert Elz

kre@munnari.OZ.AU

kre@coe.psu.ac.th

<http://fivedots.coe.psu.ac.th/~kre>

Delivery Address

- ◇ Mail to
 - `user@domain.name`
- ◇ We know how to deliver it
- ◇ We know the sender MTA
 - The one that currently has the message
- ◇ What system is the destination MTA?
- ◇ Is it `domain.name` ?
- ◇ Consider `user@psu.ac.th`
 - What system is `psu.ac.th` ?
 - There isn't one!
- ◇ Must all e-mail
 - be delivered to a specific host?
 - No, don't want that.
 - ▷ Need a better way

Mail Exchanges

- ◇ Mail Exchange
 - System that takes mail for a domain
 - Delivers to all users in that domain
 - ▷ directly or indirectly
- ◇ Need method to find Mail Exchange for domain
 - As well as address for hostname
- ◇ Domain Name System
 - Translation from domain name to
 - ▷ Address (v4 or v6)
 - ▷ Mail Exchange
 - ▷ Geographic Location
 - ▷ System & OS type
 - ▷ ...
 - Later

Network Requirements

- ◇ Reliable Communications
 - Nothing lost
 - Nothing added
 - Nothing out of order
 - Not arriving too quickly
- ◇ Clean Termination
 - Last data always arrives
 - End indication follows last data
- ◇ Transport Layer
 - TCP
 - ▷ Transmission Control Protocol

Reliable ?

- ◇ Reliable means
 - It must work
 - We can rely upon it working
 - No need to test
 - ▷ service guaranteed
- ◇ NO
- ◇ Reliable means
 - It works
 - Or we are told it failed

TCP

- ◇ RFC 793
 - IETF/ISOC Request For Comments
 - Internet Standards
- ◇ Transport Protocol
 - Arranges communications between applications
 - ▷ Between peers
 - client & server
 - or equal nodes
 - Recovers from errors
 - ▷ Lost data in network
 - ▷ Reordered data
 - ▷ Duplicated data
 - Delivers original data
 - ▷ In order
 - ▷ Without errors
 - Or reports Error

Underlying Network

- ◇ To come later, but
 - ethernetets deal with packets
 - applications (often) deal with files
 - ▷ One file ==> many packets
- ◇ data stream
 - ▷ file or whatever it is
 - must be broken into pieces
 - each sent individually
 - reassembled at receiver
- ◇ TCP does this
 - pieces called segments

Handling Loss

- ◇ Some segments might be lost in network
 - never arrive at destination
- ◇ How does TCP know?
 - TCP works for any data
 - It cannot look at image
 - ▷ "Look ma - person with no face"
- ◇ TCP sender adds identifiers to segments
 - TCP receiver looks for incoming identifiers
 - Notices any missing
- ◇ Identifiers?
 - 1 2 3 4 5 ...

Identify what?

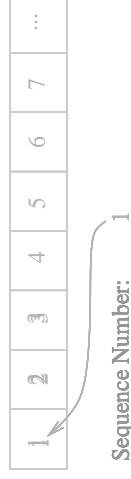
- ◇ TCP sends segments
 - So, one identifier for each segment ?
- ◇ How big is a segment?
 - Depends upon network underneath
 - Packet might pass through several networks
 - ▷ different maximum sizes
- ◇ TCP identifies octets
 - ▷ Octet == byte
 - One identifier for each octet
- ◇ Sequence Number

Sequence Numbers

- ◇ One number for each octet in data
 - Numbers monotonically increasing
 - ▷ They get bigger
 - never smaller
 - not even equal
 - ▷ Increase by 1 for each octet
- ◇ Data stream might be large
 - Many giga-bytes
- ◇ Sequence number will get big
 - If start at 1
 - Will end at big number
 - ▷ when transfer completes
- ◇ Two problems
 - sequence numbers take more space than data
 - ▷ waste lots of network bandwidth
 - we have no idea how big maximum will be

Sequence Numbers (2)

- ◇ Two problems
 - sequence numbers take more space than data
 - ▷ waste lots of network bandwidth
- ◇ Solution
 - Assume data in segment is a unit
 - ▷ all arrives
 - ▷ in same order
 - ▷ or none arrives
 - In segment
 - ▷ include sequence number of first octet
 - explicitly
 - ▷ calculate sequence numbers for others
 - using addition



Sequence Numbers (2)

- ◇ Two problems
 - sequence numbers take more space than data
 - ▷ waste lots of network bandwidth
 - we have no idea how big maximum will be
- ◇ Solution
 - Define sequence number space as a circle
 - ▷ 1 2 3 4 5 6 7 0 1 2 3 4 5 6 7 0 1 2 ...
 - Sequence continues indefinitely
 - ▷ Just like time on a clock
- ◇ Issue
 - How big is the circle?

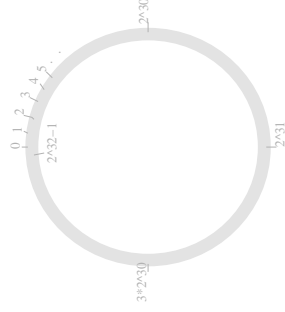
TCP Sequence Numbers

- ◇ **Issue**
 - How big is the circle?
 - How many different identifiers are needed?
- ◇ **Must have:**
 - different ID for every octet alive in network
 - ▷ actually alive
 - ▷ possibly alive
- ◇ **Consider**
 - 1 Gbit/sec network
 - ▷ ~100,000,000 bytes/second
 - If data might be alive 1 second
 - ▷ need at least 100,000,000 sequence numbers

TCP Sequence Numbers (2)

- ◇ 2^{32} was selected
 - considered big enough
 - 4,294,967,296 sequence numbers
- ◇ **Allows packets > 40 seconds @ 1Gb/sec**
 - or > 4 seconds @ 10Gb/sec
- ◇ **32 bit number**
 - convenient for computers to manipulate

TCP Sequence Number Space



- ◇ Sequence numbers run from 0 to $2^{32} - 1$
- ◇ 0, 1, 2, 3, ..., $2^{32} - 2$, $2^{32} - 1$, 0, 1, 2, ...