Lecture 12 (Feb 26, 2004)

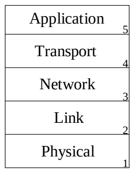
Outline

Intro to transport

UDP

Revisit Internet Architecture

- Recall Internet Architecture
- We are now going to layer 4
  - TCP and UDP

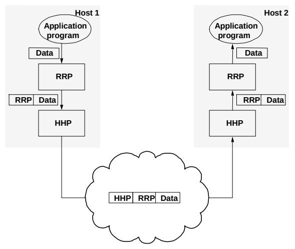


Layering and Encapsulation Revisited

- Each layer in the Internet architecture relies on layers below to provide services in black box fashion
  - Layers make complex system easier to understand and specify
  - Makes implementation more flexible
  - Can make implementation a bigger and less efficient
  - Layers are implemented by protocols – rules for communication
- Data from applications moves up and down protocol stack
  - Application level data is chopped into packets (segments)
  - Encapsulation deals with attaching headers at layers 2, 3, 4

Machinery

- Encapsulation (header/body)



End-to-End Protocols

- Underlying network is *best-effort*
  - drop messages
  - re-orders messages
  - delivers duplicate copies of a given message
  - limits messages to some finite size
  - delivers messages after an arbitrarily long delay
- Common end-to-end services
  - guarantee message delivery
  - deliver messages in the same order they are sent
  - deliver at most one copy of each message
  - support arbitrarily large messages
  - support synchronization
  - allow the receiver to flow control the sender
  - support multiple application processes on each host

Basic function of transport layer

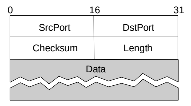
- How can processes on different systems get the right messages?
- *Ports* are numeric locators which enable messages to be demultiplexed to proper process.
  - Ports are addresses on individual hosts, not across the Internet.
- Ports are established using *well-known* values first
  - Port 80 = http, port 53 = DNS
- Ports are typically implemented as message queues
- Simplest function of the transport layer: multiplexing/demultiplexing of messages
  - Enables processes on different systems to communicate
  - End-to-end since only processes on end hosts invoke this protocol

Other transport layer functions

- Connection control
  - Setting up and tearing down communication between processes
- Error detection within packets – our first focus
  - Checksums
- Reliable, in order delivery of packets – our second focus
  - Acknowledgement schemes
- Flow control
  - Matching sending and receiving rates between end hosts
- Congestion control
  - Managing congestion in the network

User Datagram Protocol (UDP)

- Unreliable and unordered *datagram* service
- Adds multiplexing/demultiplexing
- Adds reliability through optional checksum
- No flow or congestion control
- Endpoints identified by ports
  - servers have *well-known* ports
  - see **`/etc/services`** on Unix
- Header format



- Optional checksum
  - Computed over psuedo header + UDP header + data

UDP Checksums

- Optional in current Internet
- Psuedoheader consists of 3 fields from IP header: protocol number (TCP or UDP), IP src, IP dst and UDP length field
  - Psuedoheader enables verification that message was delivered between correct source and destination.
  - IP dest address was changed during delivery, checksum would reflect this
- UDP uses the same checksum algorithm as IP
  - Internet checksum

Basics of dealing with errors

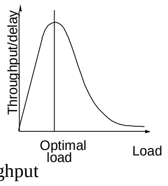
- Bit errors can be occur in packets
- This problem has been studied for a long time
  - Error detection (and correction) codes
  - Cyclic redundancy check (CRC) is a common error detection method
- Basic idea of any scheme is to add redundant data
  - Extreme example – send two identical copies of data
    - Poor for many reasons
  - A primary goal is to send minimal amount of redundant data
    - CRC used in Ethernet has 32 bits for each 1500 byte packet
  - Another goal is to make generation of checksum fast

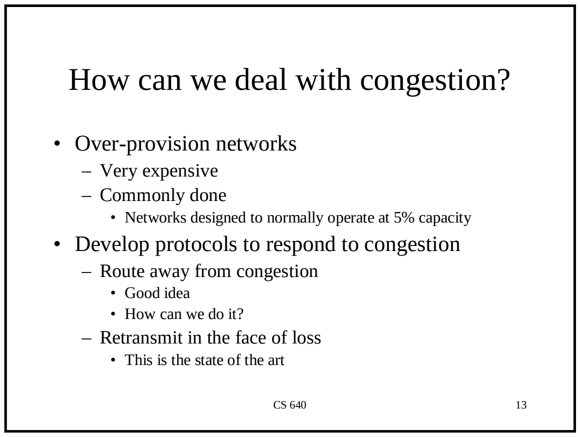
UDP in practice

- Minimal specification makes UDP very flexible
  - Any kind of end-to-end protocol can be implemented
    - Even TCP can be implemented using UDP
- Examples
  - Most commonly used in multimedia applications
    - These are frequently more robust to loss
  - RPC's
  - Many others…

Congestion in the Internet

- Checksums are effective for detecting bit errors but bit errors are not the only problem…
- We know that traffic has bursty characteristics
  - Statistical multiplexing of ON/OFF sources
  - Heavy-tailed file sizes
  - Routers have limited buffer capacity
  - Packets received after buffers are full are dropped
    - Buffers do protect from short bursts
- Congestion lengthens delays and lowers throughput
  - Standard throughput/load curve





How can we deal with congestion?

- Over-provision networks
  - Very expensive
  - Commonly done
    - Networks designed to normally operate at 5% capacity
- Develop protocols to respond to congestion
  - Route away from congestion
    - Good idea
    - How can we do it?
  - Retransmit in the face of loss
    - This is the state of the art

CS 640 13