


# CS 140: Operating Systems Lecture 26: End-to-End Layer

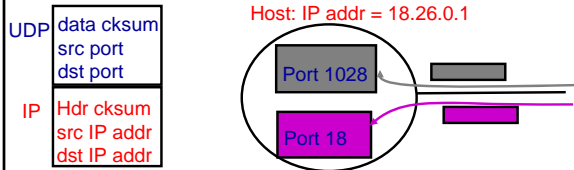
Mendel Rosenblum

## Past & Present


- ◆ Last time: the network layer  
Getting bits from one end of a collection of networks to the other.  

- ◆ The end-to-end layer  
Interface between applications and network some successful end-to-end protocols  
Turning "best effort" into a usable interface (or, deriving TCP from first principles)

## Successful story #1: UDP

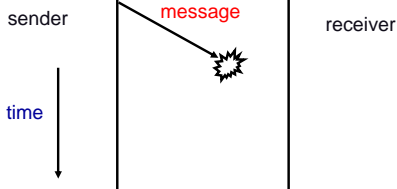
- ◆ User Datagram Protocol: best-effort, process-to-process  
Lives on top of IP: adds corruption detection and "ports" so packets can be addressed to a process on host.  
(note there are many other fields in header)



## Success story #2: TCP

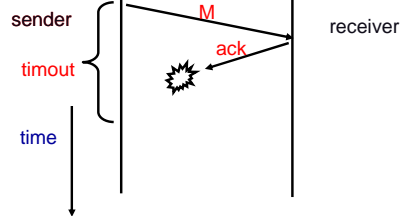
- ◆ "Transmission control protocol"  
Reliable, in-order, process-to-process, two-way byte stream.
- ◆ Like UDP:  
Layered on top of IP; adds ports and data checksums.  

- ◆ Unlike UDP in two ways  
Connection based.  
The important difference: recovers from packet loss, duplication, corruption, reordering.  
Next: answering "how???" from first principles.

## Lost packets



- ◆ Some causes:  
Traffic burst causes router buffers to overflow.  
Link corrupts packet (wireless up to 40% packet loss).  
How to fix?

## Lost acks

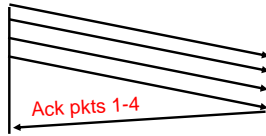


- ◆ What will the result be?



### Sliding window algorithm

- ◆ We've been sending data: send, ack, send, ack, ...  
This is called a "stop and go protocol".  
It's simple, but delivers really low bandwidth (each packet has a latency of at least 1 round-trip time).  
Sliding window allows multiple packets to be in flight.

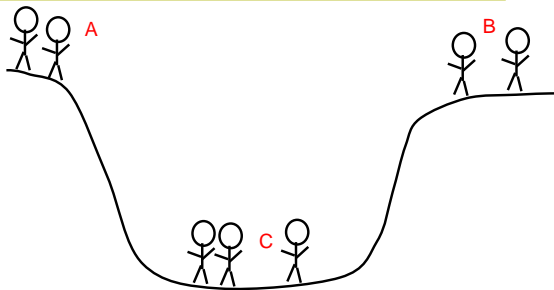


- ◆ Problem: have to make sure to not overwhelm receiver  
Have a "window" = # of bytes receiver is willing to buffer.

### What we can and cannot do

- ◆ We can
  - Synthesize reliability on an unreliable network.
  - Eliminate jitter on a bursty network (given enough buffering at the receiver)
- ◆ We cannot
  - Recreate low-latency responses on a network that can induce arbitrary latencies.  
Is there anything else?
- ◆ An impossibility: consensus on an unreliable network  
Consensus: you and I agree on 1 bit of information.  
A famous result: The two army problem.

### The two (stick figure) army problem



C can beat either A or B, but not both.  
A and B want to agree on a time to simultaneously attack

### Making names pretty

- ◆ IP addresses are not very memorable  
Internet uses the "domain name system" (DNS) to layer (sort of) human consumable names on top of them.  
DNS type names: lcs.mit.edu, leland.stanford.edu, etc.  
hierarchical. Gives scalability and distributed naming.
- ◆ Simplified resolving leland.stanford.edu to IP address:



### Protocol: naming + setup + guarantees

- ◆ Addressing: identifying receivers  
IP = host-to-host: every host has a unique IP address.  
UDP & TCP = process-to-process: extend IP with a port that is coupled to a process within a host.  
Telephone: "host-to-host": every connection point associated with a unique phone number .
- ◆ Connection model:  
IP, UDP, Ethernet: connectionless. Send packet.  
Telephone, ATM, TCP: connection-based. Explicitly set up connection before sending data. Tear down connection when done. (Makes richer service models easier.)
- ◆ Data delivery model: once data entrusted to protocol, what guarantees are made?

### Some successful data delivery models

- ◆ Best-effort:
  - IP: packets can be lost, delayed & packet data corrupted.
  - UDP: layers corruption detection on top of IP.
  - Ethernet: may lose packets, detects corruption.
  - ATM: can lose packets, detects corruption & makes reordering less likely.
- ◆ Reliable, in-order byte stream: TCP  
Two-way byte stream. Prevents: loss, duplication, corruption, reorder.
- ◆ Mostly reliable, quality-of-service: telephone  
Two-way voice service with small end-to-end delays.  
Guarantees accepted calls will run to completion.  
(easier to build on ATM than non-connection based)