SY110
Networking – Transport Layer

Major Brian Hawkins, USMC

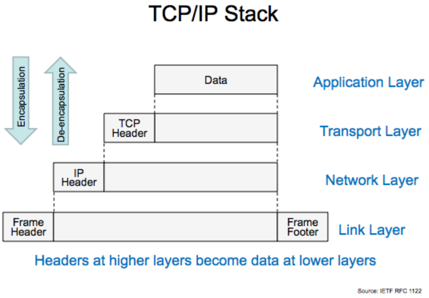
U.S. Naval Academy

Fall AY 2018

Outline

1. Review
2. Transport Layer - Basics / Terms
3. Transport Protocols
4. PracApp
5. NAT
6. Questions

Review



Source: IETF RFC 1122

Network Layer

- Routers!
- IP Addresses!
- Interconnects networks
- Protocols (IPv4, ICMP, IPv6)
- Tools (ipconfig, ping, & traceroute)

Transport Layer - Basics

So, the Network Layer gets data (packets) between hosts, but how do I know which packets go to which applications?

- Transport Layer provides a service to the application layer to ensure that the right data gets to the right process in the requested manner.

Some Important Terms

- Datagram - the name for data being sent at the Transport Layer.
- Port - a 16-bit number used to identify the application/process to which the data belongs. `http://en.wikipedia.org/wiki/List_of_TCP_and_UDP_port_numbers`

Two Primary Protocols

- Transport Control Protocol (TCP)
- User Datagram Protocol (UDP)

Transport Layer - Basics

So, the Network Layer gets data (packets) between hosts, but how do I know which packets go to which applications?

- Transport Layer provides a service to the application layer to ensure that the right data gets to the right process in the requested manner.

Some Important Terms

- Datagram - the name for data being sent at the Transport Layer.
- Port - a 16-bit number used to identify the application/process to which the data belongs. `http://en.wikipedia.org/wiki/List_of_TCP_and_UDP_port_numbers`

Two Primary Protocols

- Transport Control Protocol (TCP)
- User Datagram Protocol (UDP)

TCP – Transmission Control Protocol

- TCP sets up a reliable connection between hosts, but it takes time (overhead) to setup (and tear-down) the connection.
- We call this *connection-oriented*.
- But all that overhead gains us:
  - Lost packet retransmission
  - Out-of-order delivery handling
  - Flow control
  - Error detection

UDP – User Datagram Protocol

- UDP does not care about providing reliability, instead it trusts that the application itself will handle those issues, or that that application does not care about those specific issues.
- Because UDP is not concerned with providing error correction or packet loss detection, it does not need to setup an actual connection. That's why it is called *connection-less*
- UDP just fires off the packets and hopes that they all get to the other side. If something breaks the application service will deal with it.
- UDP is not faster then TCP!!! However, since UDP requires less overhead, the UDP process is shorter in a general sense. But UDP doesn't speed up delivery of individual packets, it just gets a head start on sending data packets on their way.

Putting it all together

So how do transport protocol, services, and ports relate?

- An application/process needs to decide what service it wants the transport layer to provide.
  - Reliability vs low-overhead
- Most application/process want/need the transport layer protocol to handle errors, lost packets, out-of-order packets, etc.
  - For example, what would happen if you only got a portion of the packets from a web server? (It would be bad, right)
- So most services choose to use TCP as its transport protocol.

cont'd...

So why even bother with UDP?

- Some services don't need (or don't *want*) the transport layer to provide reliability. Instead the service either handles it or it just doesn't care about reliability. Why might we want this?
- Domain Name System (DNS) uses UDP. (We'll learn more about DNS in our Application Layer lecture.)
    - DNS messages are very small, almost always sent in one individual packet!
    - The overhead of setting up the connection, sending one packet, and then tearing down the connection is a waste.
    - If that one packet is lost the client just resends again after not getting a response from the server.
- Most all "near real-time" communication processes use UDP.
    - Video chat/conferencing (FaceTime)
    - Voice over IP (VoIP)
    - Live video feeds (watching sports games for example)

Some useful tools from the command line

- What processes are communicating over which ports on my computer (host)?:
  - `netstat` - Provides very basic protocol
  - `netstat -bno` - Adds useful process info (see Task Manager)
  - To make it easier to read try:

    ```
    netstat -bno | more
    ```

    - or -

    ```
    netstat -bno > netstatInfoFile.txt
    ```

- Demonstrating TCP or UDP between your computers:
  `nk` (netkitten) is a custom USNA tool based on `nc` (netcat)
  - 1) Set up a "server" (listener): `nk -l <portNum>`
    - Pick a port number in the range 49152 - 65535.
  - 2) Send data from a "client": `nk <serverIP> <portNum>`
    - You can do this on your own computer with two shell windows and using "localhost" as the server's IP.

NAT

NAT - Network Address Translation

- Created to deal with the exhaustion of IPv4 addresses
- Uses the idea of non-routable IP addresses (Private IP addresses)
  - `10.0.0.0 - 10.255.255.255`
  - `172.16.0.0 - 172.31.255.255`
  - `192.168.0.0 - 192.168.255.255`

NAT Demo:

- http://rona.academy.usna.edu/~sy110/resources/netdemo/privateIP.html

What's my public IP?

- https://www.ipchicken.com

Questions

Questions?