ICS 351: Today's plan

- IP multicasting
- IGMP
- PIM
- TCP and UDP
- port numbers
- congestion control

IP Multicast Addresses

- IP multicast addresses are in class D, beginning with 224 through 239
- 1. the first byte for class A addresses is 0 through 127
- 2. the first byte for class B addresses is 128 through 191
- 3. the first byte for class C addresses is 192 through 223
- 4. the first byte for class D addresses is 224 through 239
- 5. the first byte for class E addresses is 240 through 255
- for example, 224.0.0.9 for RIP packets, 224.0.0.5 for OSPF packets

IP Multicasting

- In ICS 351, we use multicast addresses to forward routing packets within a local network
- IGMP manages group membership in multicast groups within local networks (MLD does the same on IPv6 networks)
- PIM (or MOSPF) are the equivalent of routing protocols for multicast, providing multicast routing when the multicast router is not local

IGMP

- Internet Group Management Protocol version 3, RFC 3376
- communication between a multicast router and local multicast hosts
- the router needs to know which hosts require which multicast stream(s), so as to only forward streams that are needed
- a host requests a stream from its router
  - the router should record this information
- requests expire if not refreshed: **soft state** in the router
- messages sent over IP (protocol number 2) with TTL 1
- IGMP routers send Membership Queries, IGMP hosts send Membership Reports

PIM

- Protocol Independent Multicast
  - protocol-independent means any routing protocol may provide the routes
- PIM dense mode (PIM-DM), RFC 3973
- in dense mode, multicast data is sent to all routers except those that send prune messages
- dense mode is only used within an autonomous system (with MSDP used to allow multicast among autonomous systems)
- PIM sparse mode (PIM-SM), RFC 4601
- in sparse mode, multicast data is broadcast over a tree rooted at a designated router called the **Rendezvouz Point (RP)**
- also PIM Source-Specific Multicast (PIM-SSM) and Bidirectional PIM (BIDIR-PIM), a variant of PIM-SM

TCP and UDP

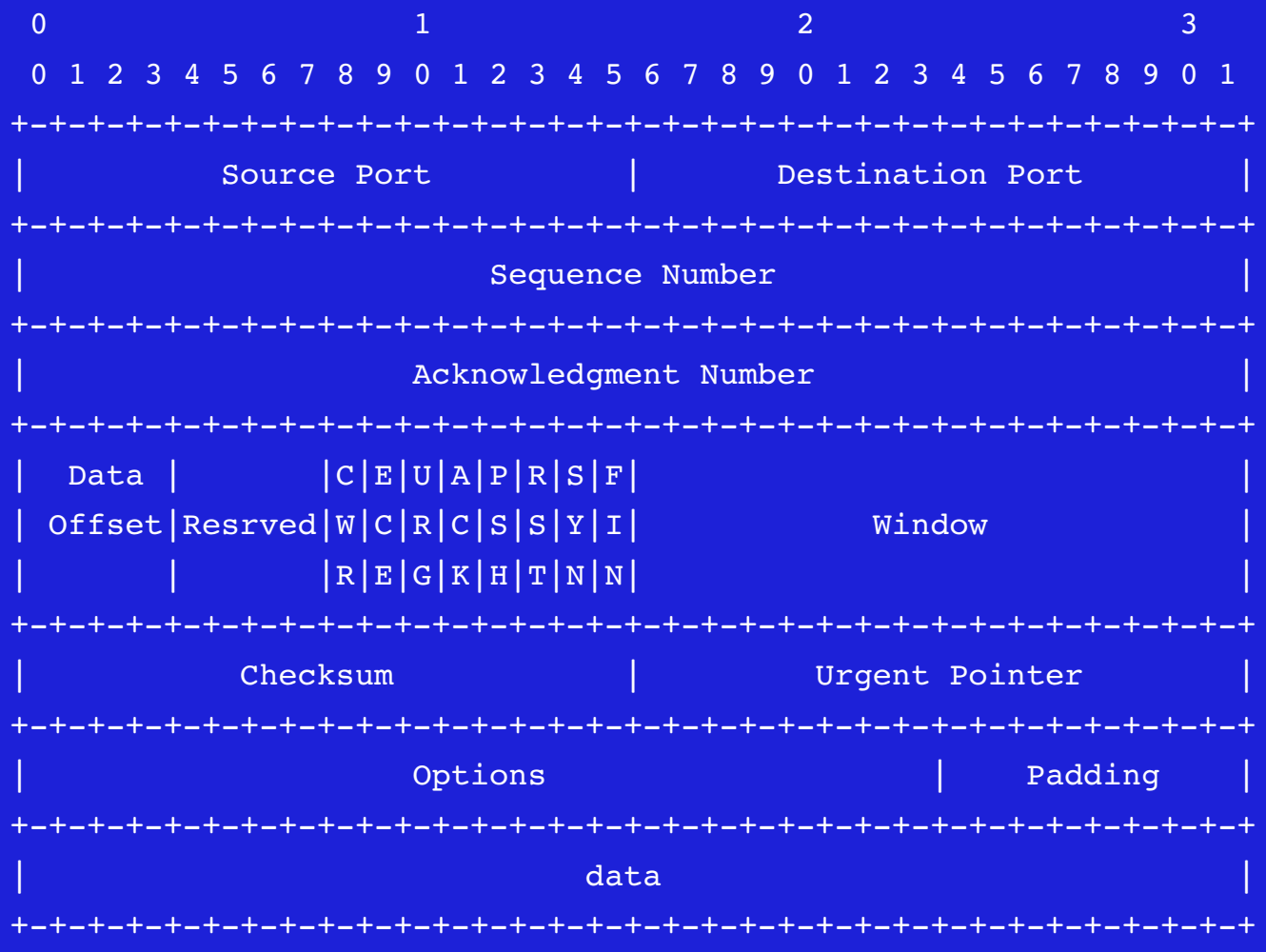
- layered above IP (header follows IP header)
- provide another layer of addressing: port numbers, which let us identify applications (sockets) within hosts

UDP

- UDP (RFC 768) essentially only provides port numbers and a checksum:

TCP Header

- RFC 793 and RFC 1122



TCP Behavior

- TCP control bits (SYN, FIN, ACK, RST) help maintain TCP connections
- three-way handshake is SYN, with SYN-ACK in answer, and a final ACK to confirm receipt of the second packet
- 32-bit sequence number, ack number count bytes rather than packets
- an ack is sent, almost for free (piggyback) in every packet except the first

TCP Window

- window tells the recipient how many more bytes (past the ack) the sender of this packet is willing to receive -- flow control, slowing down the sender to avoid overwhelming a slow receiver
- this is the flow control window
- setting the window to zero forces the sender to stop
- in general, TCP can send one window every RTT (round-trip time)

port numbers

- an IP address identifies an interface, and by extension a machine
- a port number identifies an application within a machine
- servers *listen* on specific, *well-known* ports
- each local port can be used for multiple *sockets*, as long as (at least) one of these is different: local/remote IP, local/remote port, protocol
- note:
  - a socket has a **local** and a **remote** port (and IP addresses)
  - a packet has a **source** and a **destination** port (and IP addresses)
- local and remote make sense on a host

  source and destination make sense for a packet

Congestion Collapse

- reminder: the network hardware might be working fine, but if the software fails, the network goes down
- e.g. if the routing tables include loops, packets will not get delivered
- imagine a retransmission mechanism where, when a packet is lost, I resend the lost packet and also a new one
- if a packet is lost due to congestion, the first little congestion experienced will likely lead to more congestion
- this happened a couple of times in the 1970's -- the network hardware was working fine, but almost no data would get through

TCP Congestion Control

- to control congestion, TCP slows down substantially (half the speed) when packets are lost
- TCP then slowly speeds up its transmission rate when no packets are lost.
- this is controlled by a window that (unlike the flow control window above) is maintained on each sender, and never communicated: the congestion window
- the effective window is the smaller of the flow control window and the congestion window

TCP Congestion Control: details

- when packets are lost, the congestion window shrinks to about half its previous size
  - actually it shrinks to one packet (one Maximum Segment Size)
  - then grows exponentially to half the previous window
- every RTT when no packets are lost, the congestion window grows by one packet (one MSS)
- since each TCP can send one window every RTT, shrinking the window slows down sending
- TCP also has other mechanisms to lessen congestion:
  - binary exponential backoff on retransmissions
  - adaptive timers to more reliably detect packet loss