# UDP, TCP, IP multicast

Dan Williams

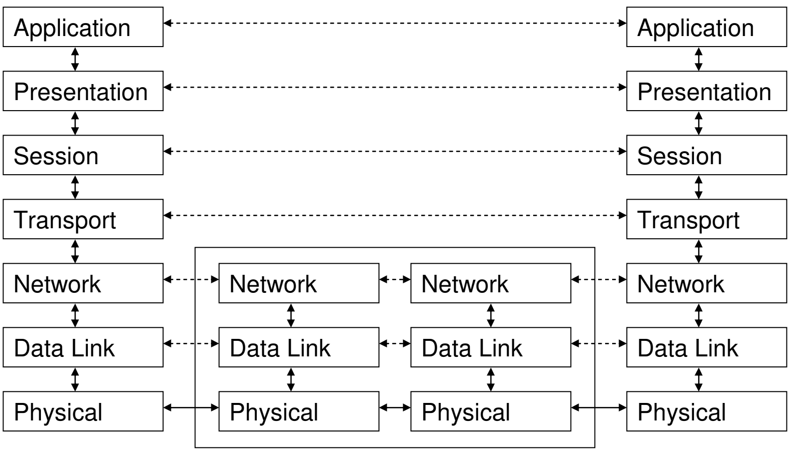
## In this lecture

- UDP (user datagram protocol)
  - Unreliable, packet-based
- TCP (transmission control protocol)
  - Reliable, connection oriented, stream-based
- IP multicast

# Process-to-Process

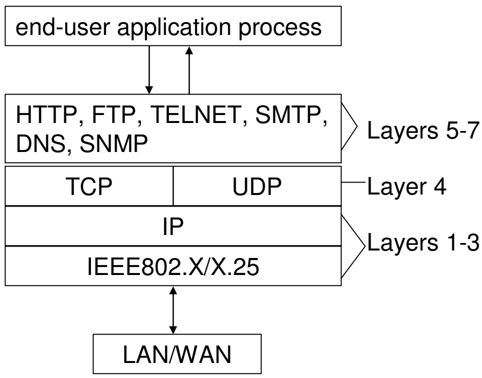
- We've talked about host-to-host packet delivery with IP
- Now we look at process-to-process communication
  - UDP (user datagram protocol)
  - TCP (transmission control protocol)
- Transport Layer Protocols
  - OSI model, TCP/IP model

# ISO OSI Reference Model




One or more nodes within network

# TCP/IP Architecture




# What we want from transport layer

- Guaranteed message delivery
- Same order message delivery
- Send arbitrarily large messages
- Allow receiver to apply flow control to sender
- Support multiple application processes on each host
- Network congestion control

# Remember Network may

- Drop packets
- Reorder packets
- Deliver duplicate copies of a packet
- Limit packets to some finite size
- Deliver packets after arbitrarily long delay

Note: things sent in the network layer are normally called packets, whereas things sent in the transport layer are normally called messages.

# UDP

- Simple – don't add much more than the best effort given by IP
- Demultiplex messages via port numbers
  - port number – number that kernel uses to deliver to appropriate application
  - For instance: HTTP is port 80, SMTP is port 25, Telnet is port 23, DNS is port 53, FTP is port 21
- Ensure correctness of message with checksum
- No connection established, datagram-based

# What we get from UDP

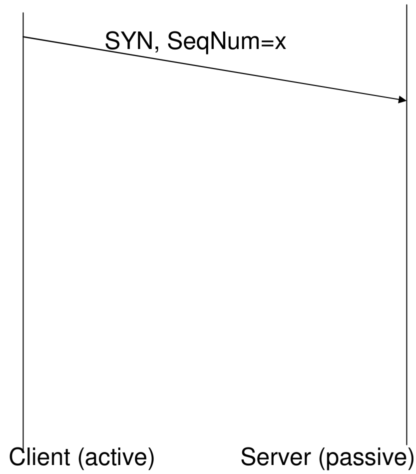
- NO guaranteed message delivery
- NO guaranteed same order message delivery
- NO ability to send arbitrarily large messages
- NO flow control
- Support multiple application processes on each host
- NO network congestion control

# TCP

- Stream based protocol allows us to send arbitrarily long messages
- Uses port numbers like UDP to send to multiple processes on each host
- Full duplex protocol
  - Both sides are senders and receivers
- We need each packet to have a sequence number to ensure reliable/in-order delivery

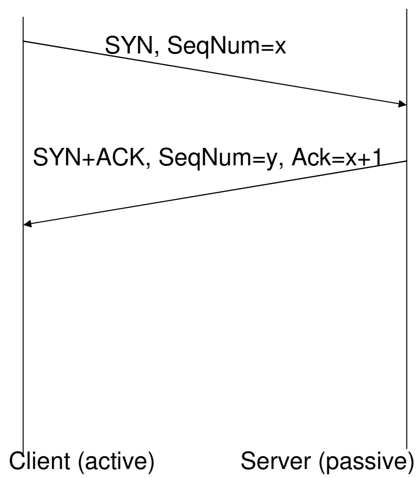
# TCP connection establishment

- Three-way handshake
  - SYN sent with initial sequence number of client



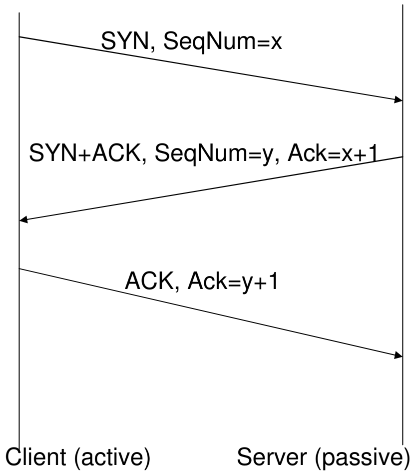

# TCP connection establishment

- Three-way handshake
  - SYN sent with initial sequence number of client
  - Server ACKs client's sequence number and sends its own sequence number for SYN



# TCP connection establishment

- Three-way handshake
  – SYN sent with initial sequence number of client
  – Server ACKs client's sequence number and sends its own sequence number for SYN
  – Client ACKs server's sequence number
  – Now we are synchronized and ready to communicate



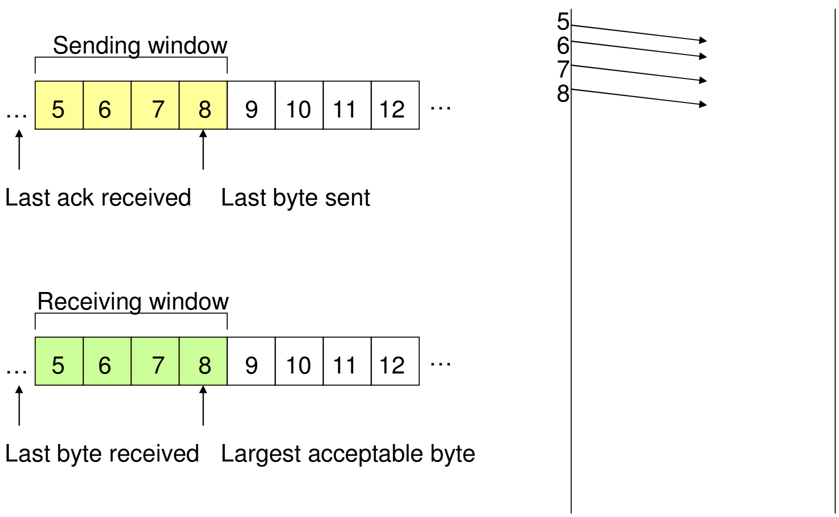

# TCP connection termination

- Each side needs to independently close its own half of the connection
- Send "Fin" pkt before terminating.
  - This is acknowledged by the other side.
  - On receiving acknowledgement, this direction is closed.
  - The other side follows exactly the same protocol.

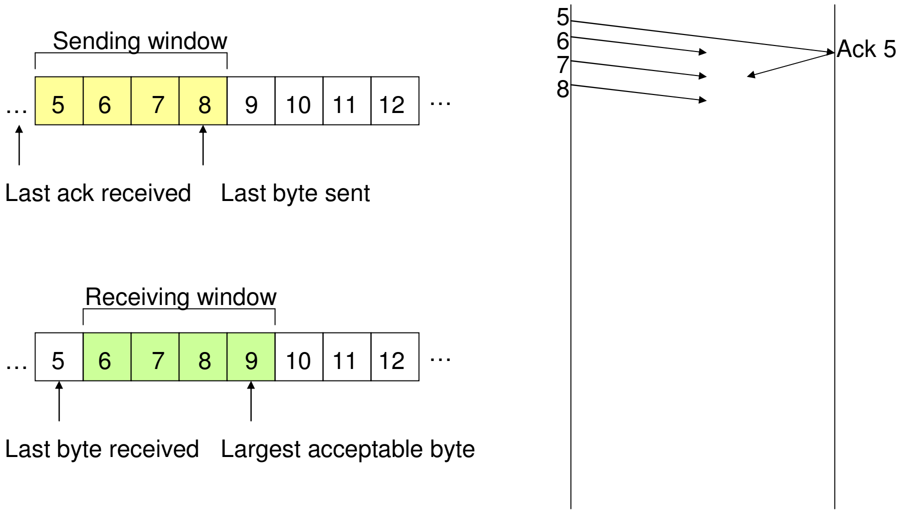
# Sliding window gives us

- Guaranteed message delivery
- Same order message delivery
- Send arbitrarily large messages
- Allow receiver to apply flow control to sender
- Support multiple application processes on each host
- Network congestion control

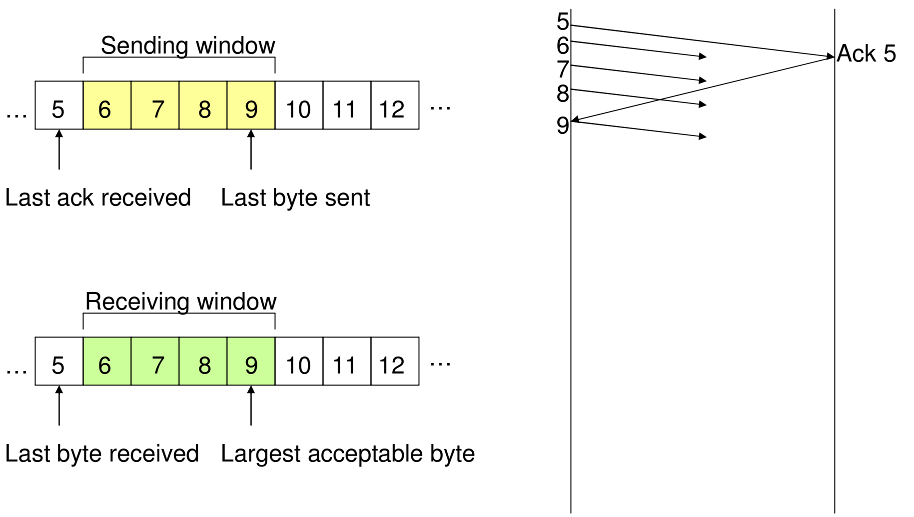
# Sliding Window



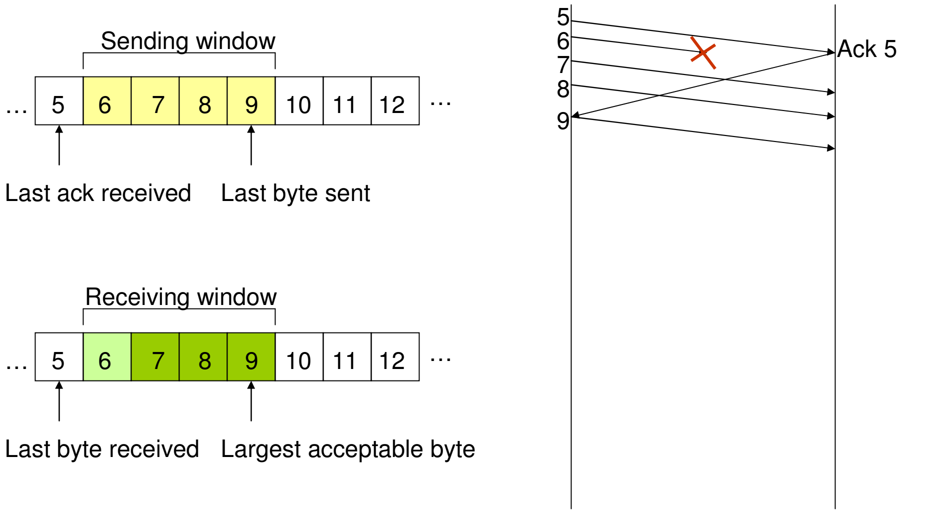
# Sliding Window



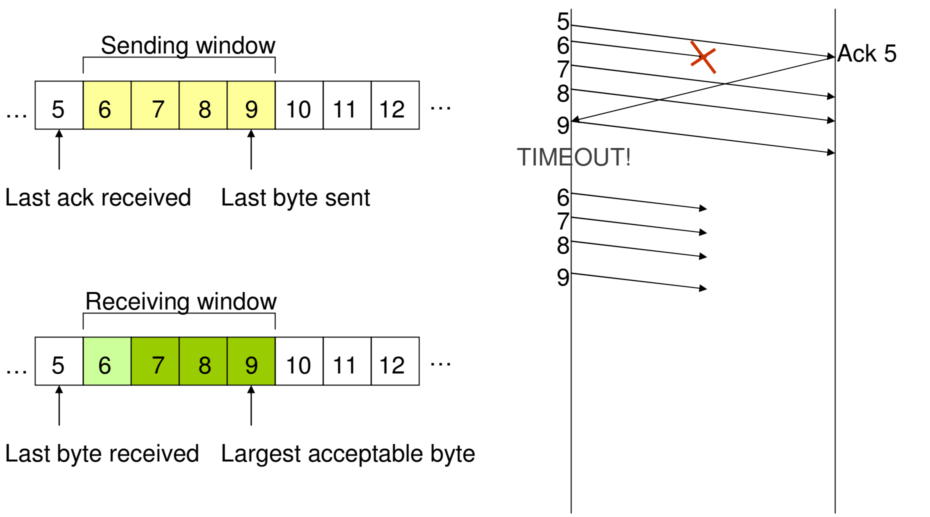

# Sliding Window



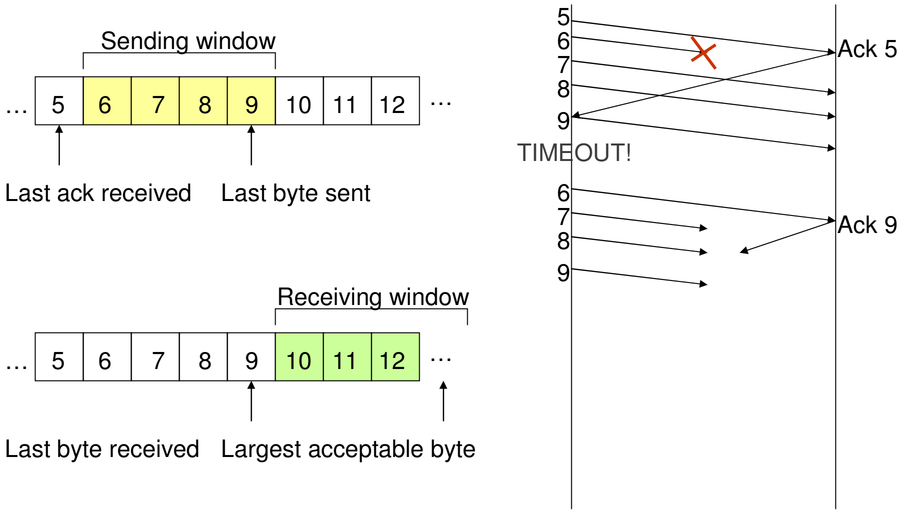
# Sliding Window



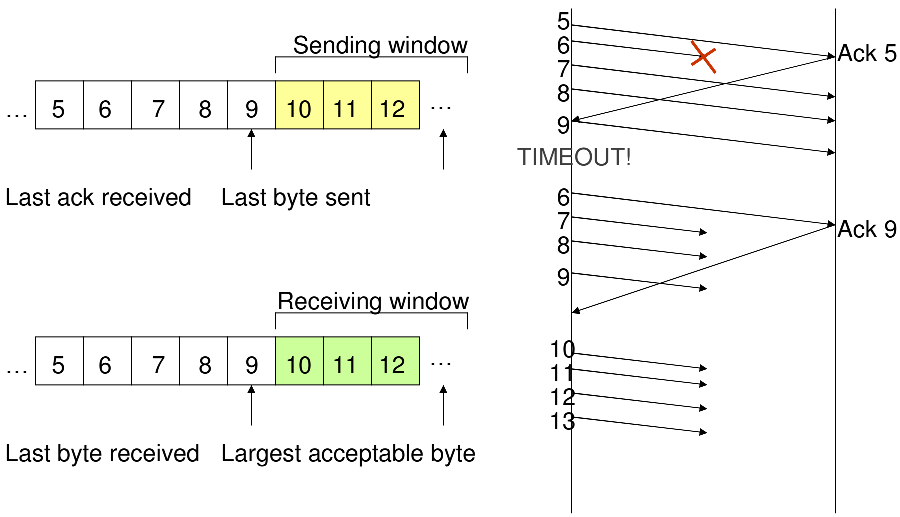

# Sliding Window



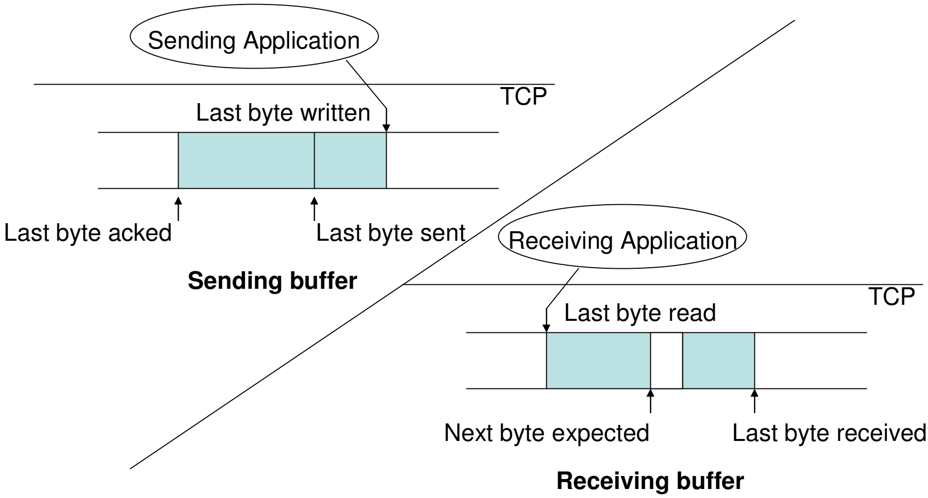
# Sliding Window




# Sliding Window



# Sliding Window with TCP




# Using sliding window for flow control

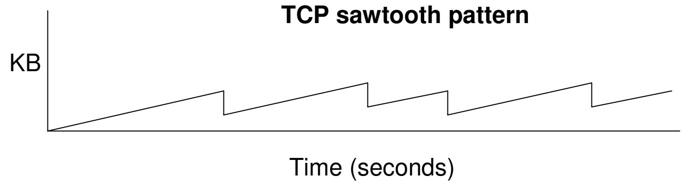
- Flow control:
  - make sure sender doesn't send more than receiver can handle
- Receive window can't slide if receiving process isn't ready to read from buffer
- Send advertised window size with ACK
  - Sender limits send window size based on advertised window size
- If send buffer fills up, TCP blocks sender

# What about congestion control?

- Introduced 8 years after TCP/IP had become operational
- Internet suffering from congestion collapse
- Idea:
  - Determine how many packets can be safely transmitted
  - ACK is a signal that a packet is out of network and can safely add another
- How do you find out capacity of network?

# Additive Increase Multiplicative Decrease

- Want to find point of congestion in network
- Maintain congestion window size
- TCP send window = min(congest, advert)
- Interpret dropped packets as congestion



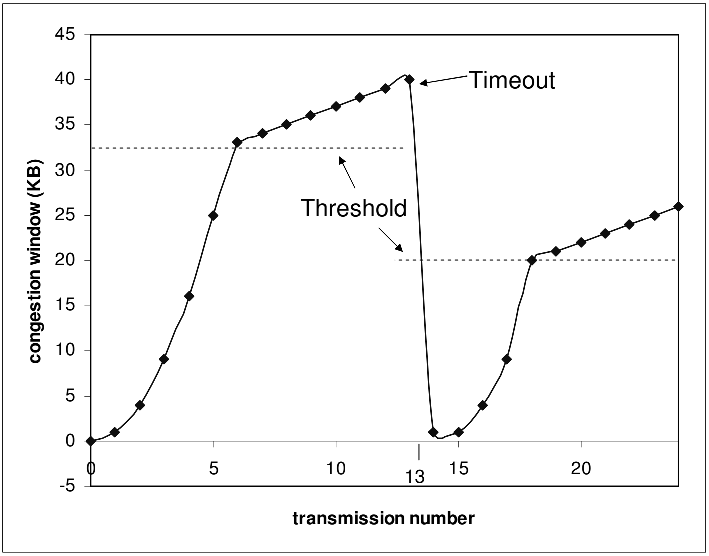
## Why decrease aggressively increase conservatively?

- Too large window worse than too small
  - Too large: dropped packets retransmitted making congestion worse
  - Too small: can't send as many packets without receiving an ACK
- What about at the start of connection?
  - Takes too long to get a large window

## Slow Start

- Double congestion window after every ACK until we get a loss or threshold is reached
- Threshold = half of the congestion window at which a packet was dropped
- Then do additive increase
- Called "slow start" because it starts slower than TCP without congestion control

# Slow Start congestion window




# Now we can see how RED works

- Recall RED (Random Early Detection)
  - Router monitors its own queue length
  - Router implicitly notifies sender of congestion by randomly dropping a packet
- TCP will note this as congestion and multiplicatively decrease congestion window

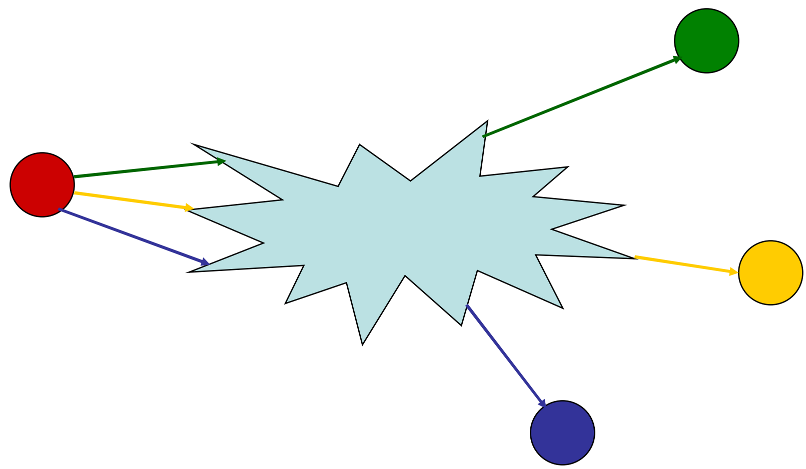
# What we get from TCP

- Guaranteed message delivery
- Same order message delivery
- Send arbitrarily large messages
- Allow receiver to apply flow control to sender
- Support multiple application processes on each host
- Network congestion control

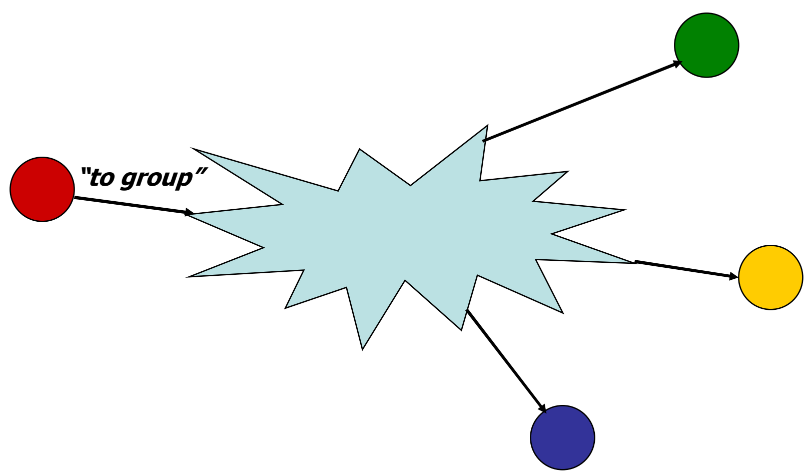
# IP multicast

- Host may want to send a packet to multiple hosts
- Send to a group, not individual hosts
  - Reduces overhead for sender
  - Reduces bandwidth consumption in network
  - Reduces latency in receiver (all see packet at same time)

## Unicast to multiple hosts



## Multicast to multiple hosts



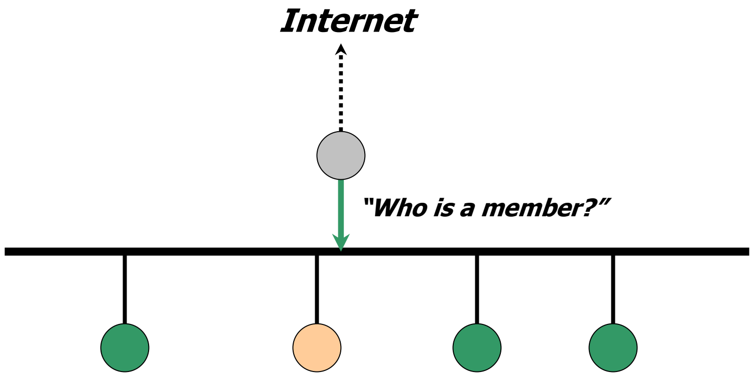
# Multicast Group

- Idea:
  - Send to group that receivers may join
  - Group has specially assigned address
- Class D IP addresses for group
  - 224.0.0.0 to 239.255.255.255
- How does a host join a group?
  - IGMP (Internet Group Management Protocol)

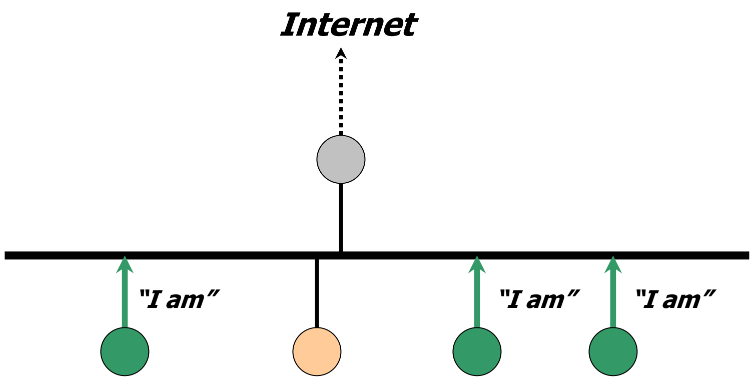
# IGMP

- Detects if a multicast group has any members within a LAN
- *Query* and *report* messages
  - router sends query of group membership periodically
  - hosts report groups they're in

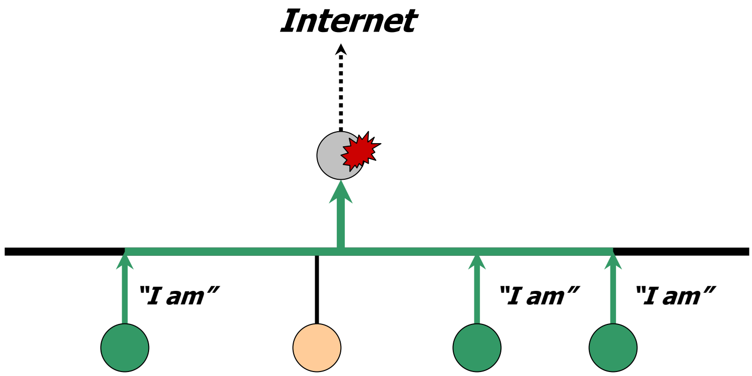
# IGMP




# IGMP



# IGMP




# Avoiding overloading

- Report packets may overload router
  - upon getting a query, each group member sets a timer
  - if it sees a report for its group before the timer expires, it suppresses its report
  - otherwise reports on expiration

# Multicast over Ethernet

- Trivial
- Ethernet is a broadcast medium – every host hears every packet
- Simply need to receive packets sent to group address
- Want to extend this between LANs connected by routers

# Naïve Approach

- We want to send multicasts everywhere where there are group members
  – use *flooding* to send multicast between routers
  – when we get to a LAN, use regular (Ethernet) multicast

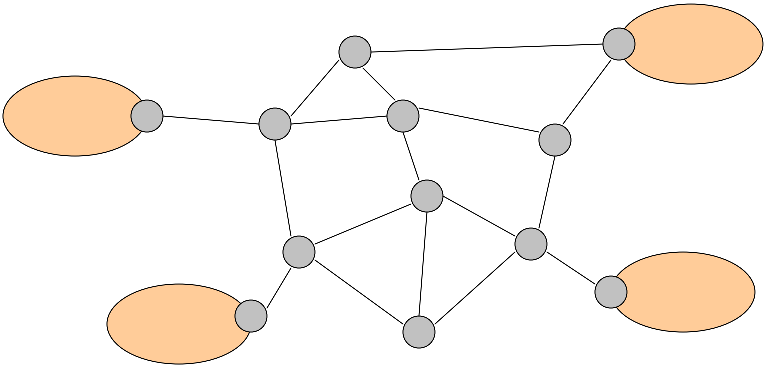
## Flooding not a good solution

- Not scalable
- LANs which do not have any group members will still receive multicast
- Need to know which LANs want to receive multicast message
  - Implement in routers
  - Create spanning tree

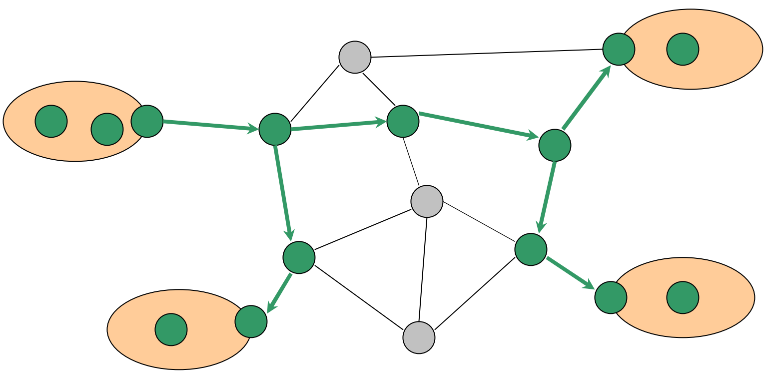
## Multicast trees

- Shortest-path tree to all multicast members, rooted at sender
- But must be computed independently by each router
- And must be dynamically adjusted for joins and leaves

# A multicast tree



# A multicast tree



THE END