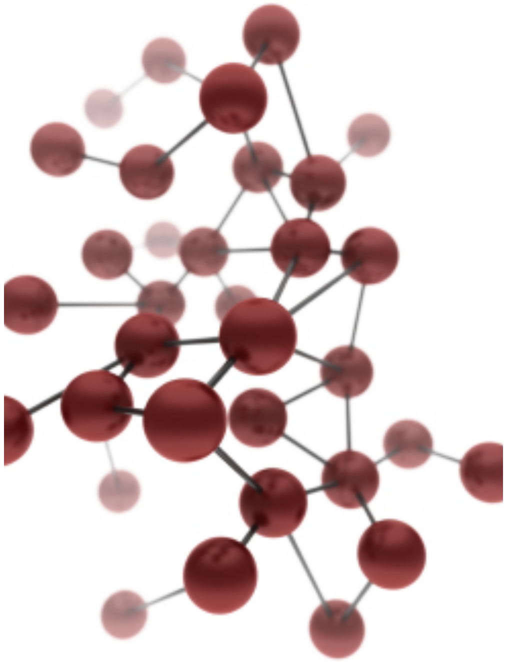


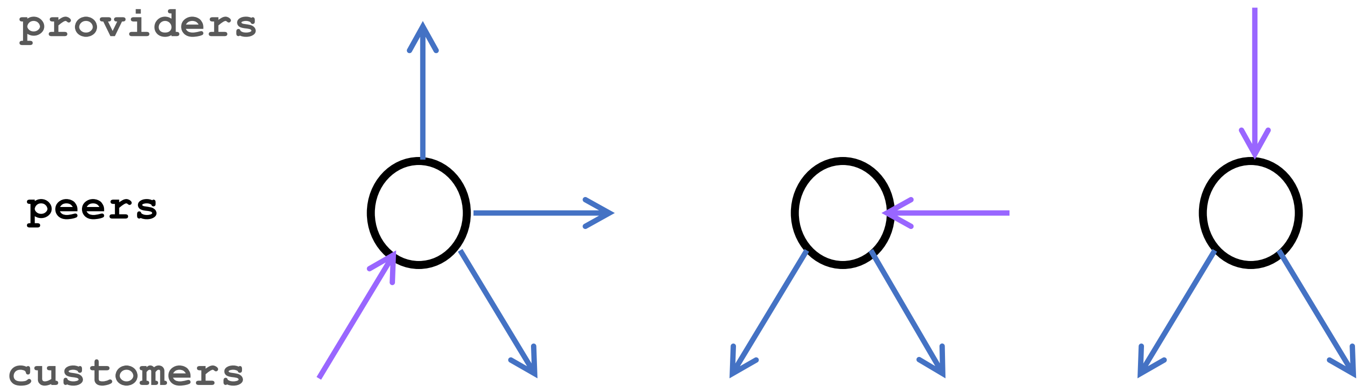
BGP, IP, TCP
Lecture 7

BGP Security

- An AS can claim to serve a prefix that they actually don't have a route to (blackholing traffic)
  - Problem not specific to policy or path vector
  - Important because of AS autonomy
  - Fixable: make ASes "prove" they have a path

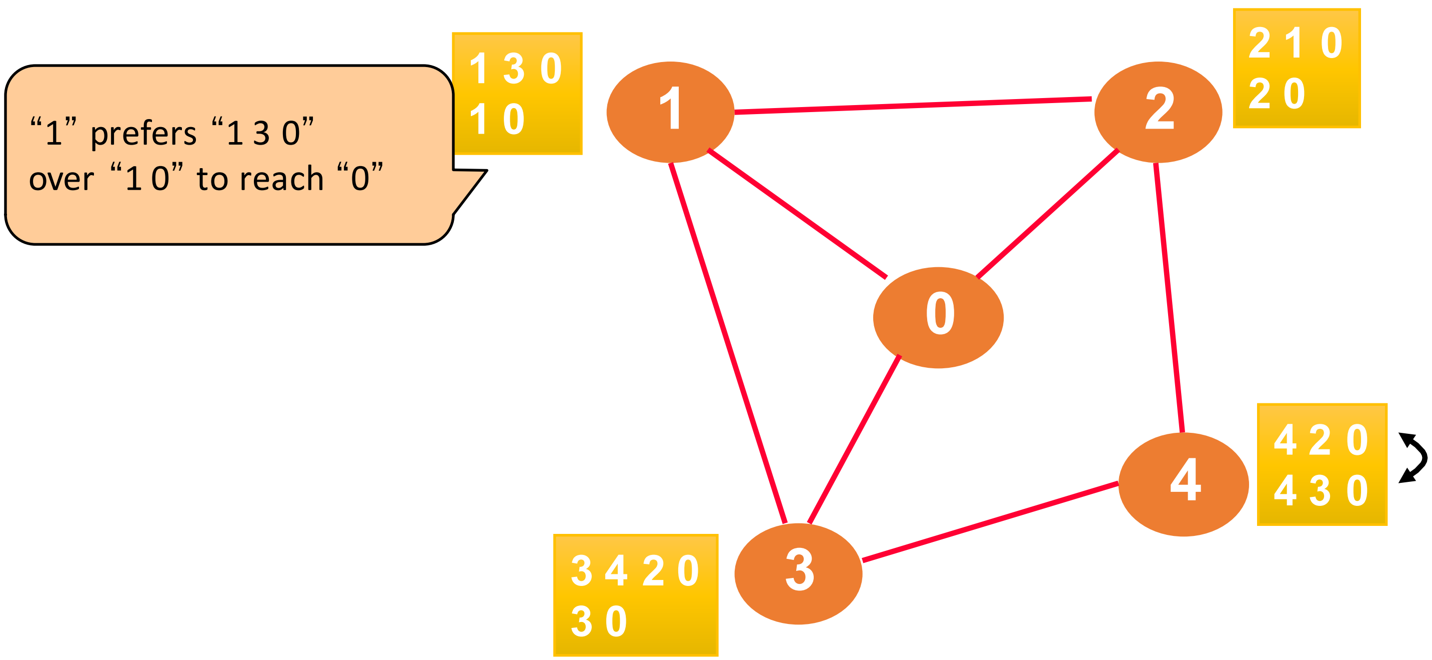
- Note: AS may forward packets along a route different from what is advertised
  - Tell customers about fictitious short path…
  - Much harder to fix!

Gao-Rexford

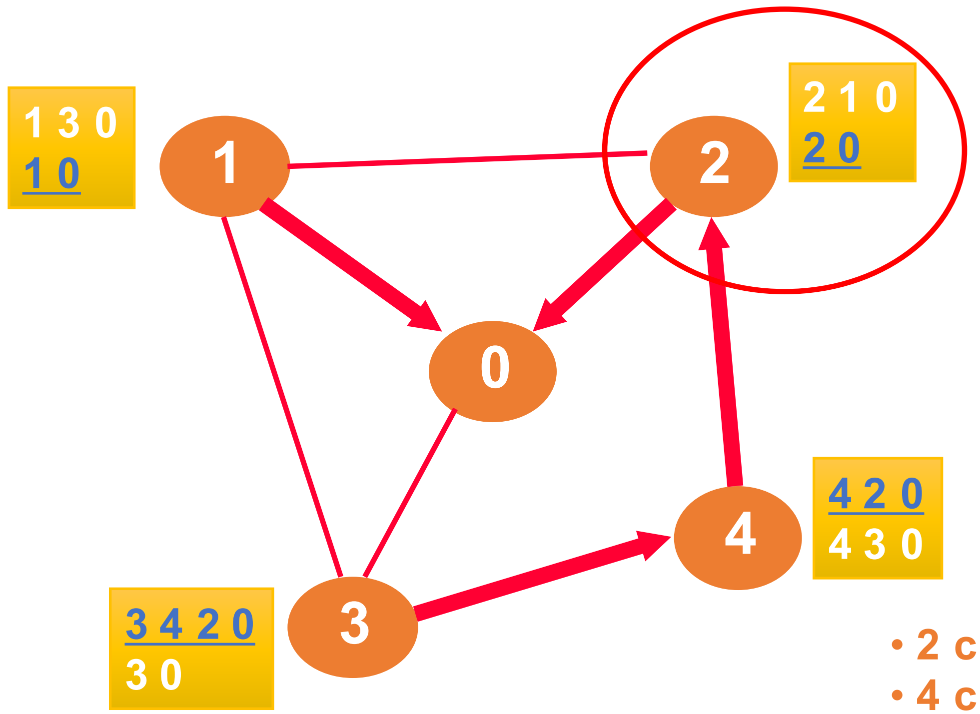


With Gao-Rexford, the AS policy graph is a
DAG (directed acyclic graph) and routes are "valley free"

Example of Policy Oscillation

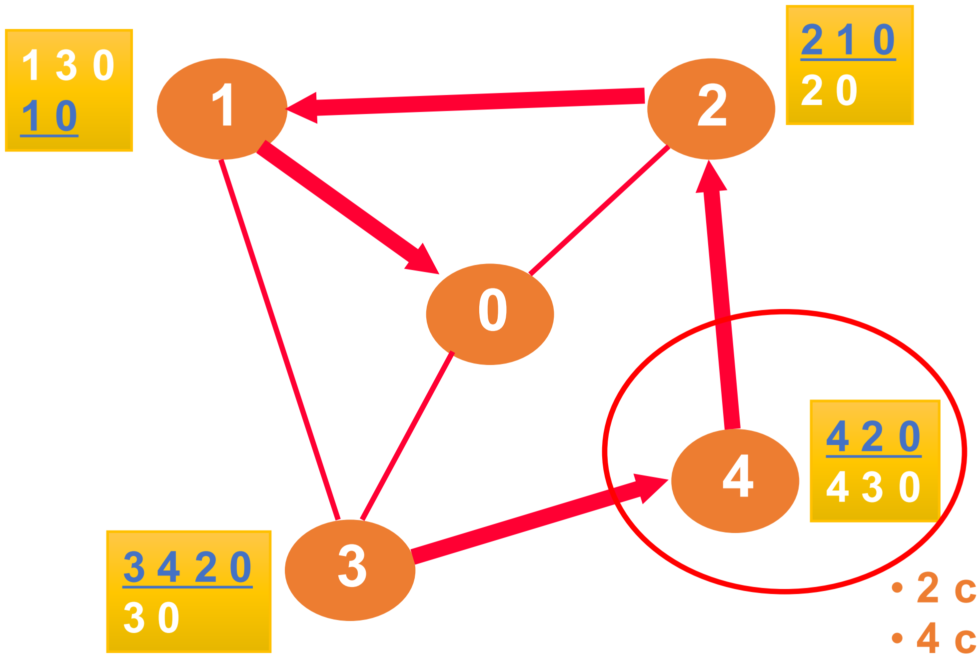


Example: Oscillation



- 2 chooses (2 0)
- 4 chooses (4 2 0)
- 3 chooses (3 4 2 0)
- 1 chooses (1 0)

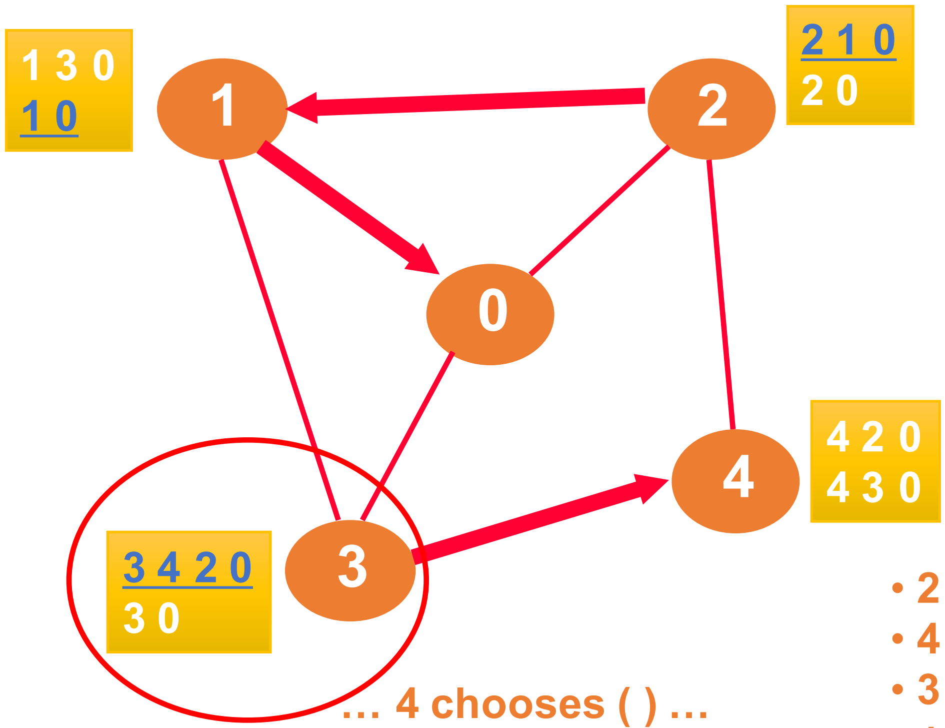
Example: Oscillation



… 2 chooses (2 1 0) …

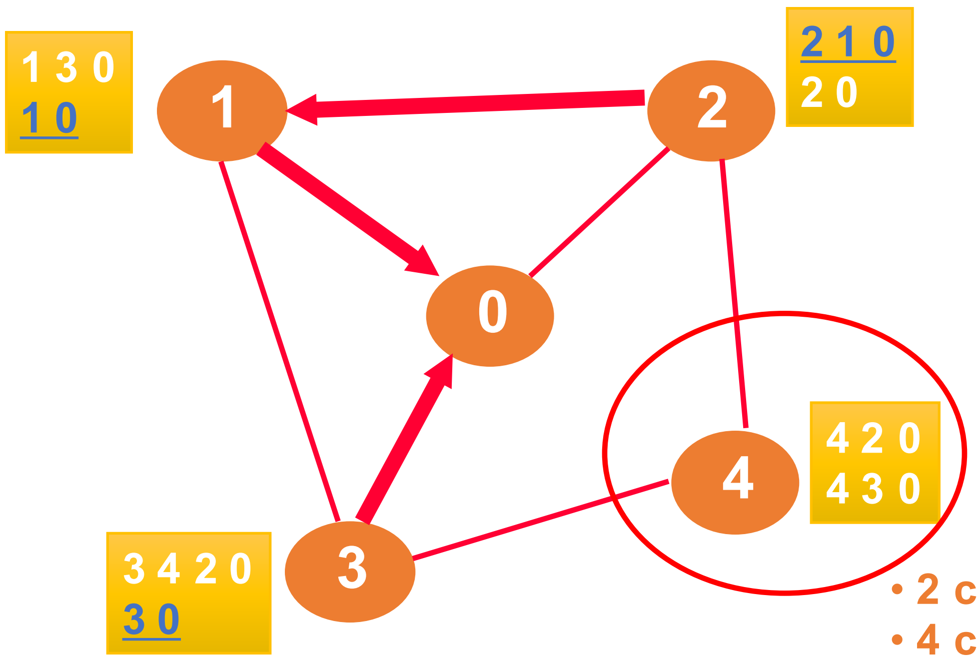
- 2 chooses (2 0)
- 4 chooses (4 2 0)
- 3 chooses (3 4 2 0)
- 1 chooses (1 0)

Example: Oscillation



- 2 chooses (2 0)
- 4 chooses (4 2 0)
- 3 chooses (3 4 2 0)
- 1 chooses (1 0)

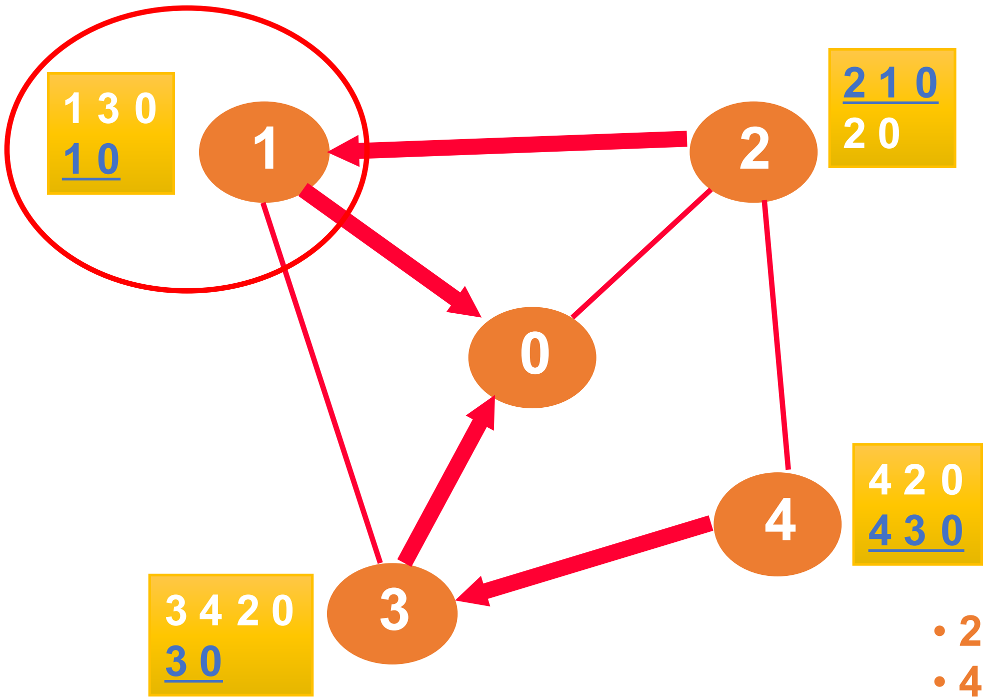
Example: Oscillation



… 3 chooses (3 0) …

- 2 chooses (2 0)
- 4 chooses (4 2 0)
- 3 chooses (3 4 2 0)
- 1 chooses (1 0)

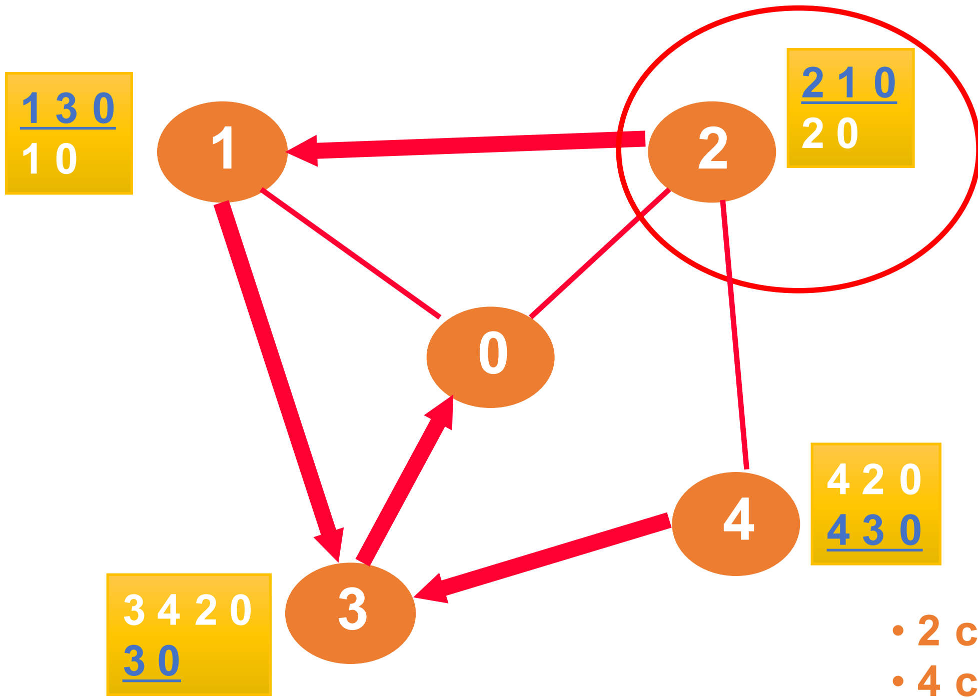
Example: Oscillation



… 4 chooses (4 3 0) …

- 2 chooses (2 0)
- 4 chooses (4 2 0)
- 3 chooses (3 4 2 0)
- 1 chooses (1 0)

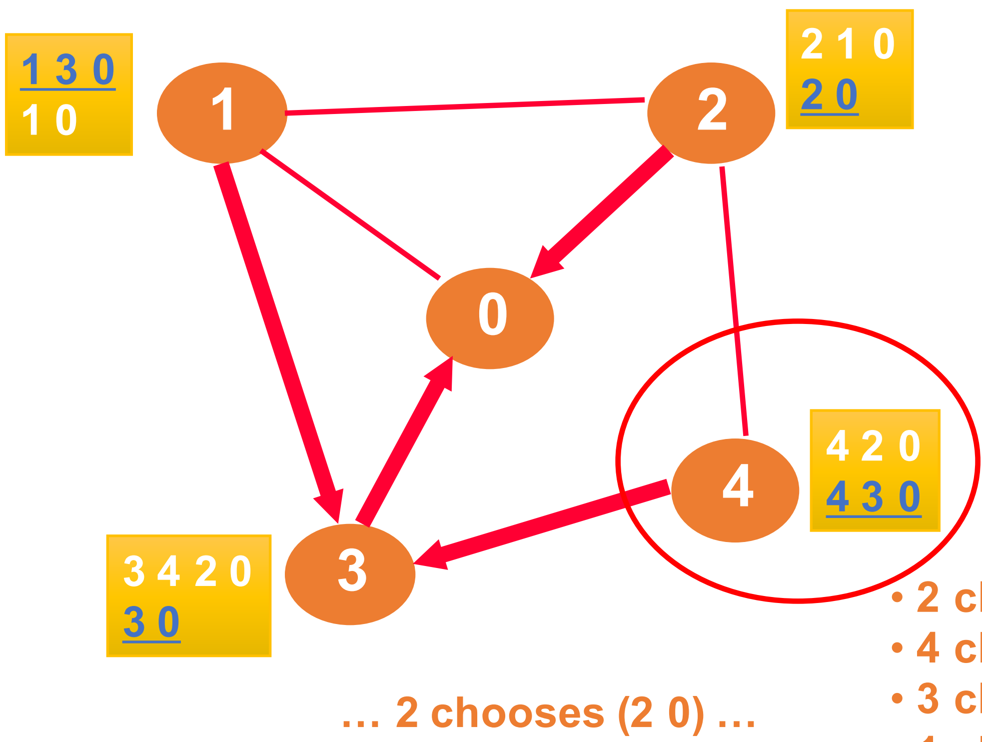
Example: Oscillation



… 1 chooses (1 3 0) …

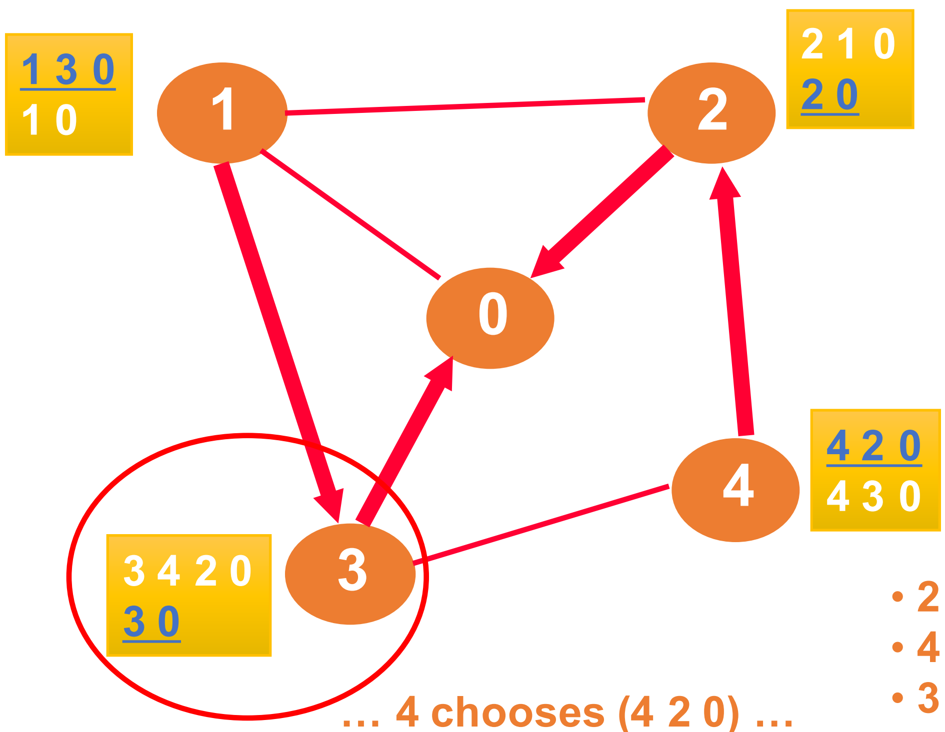
- 2 chooses (2 0)
- 4 chooses (4 2 0)
- 3 chooses (3 4 2 0)
- 1 chooses (1 0)

Example: Oscillation



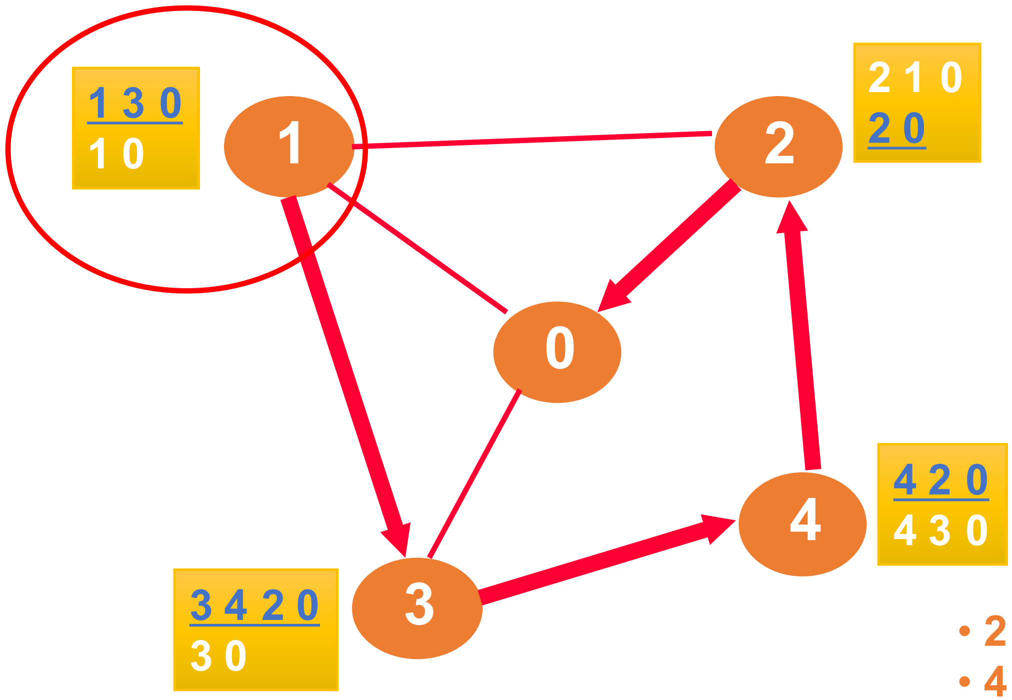
- 2 chooses (2 0)
- 4 chooses (4 2 0)
- 3 chooses (3 4 2 0)
- 1 chooses (1 0)

Example: Oscillation



- 2 chooses (2 0)
- 4 chooses (4 2 0)
- 3 chooses (3 4 2 0)
- 1 chooses (1 0)

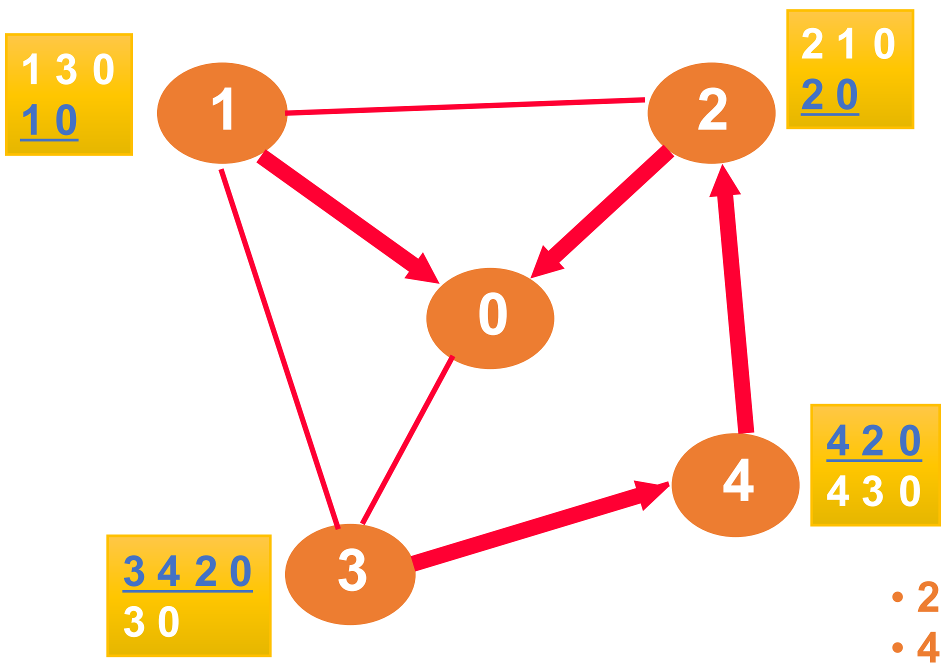
Example: Oscillation



… 3 chooses (3 4 2 0) …

- 2 chooses (2 0)
- 4 chooses (4 2 0)
- 3 chooses (3 4 2 0)
- 1 chooses (1 0)

Example: Oscillation



... 1 chooses (1 0) ...

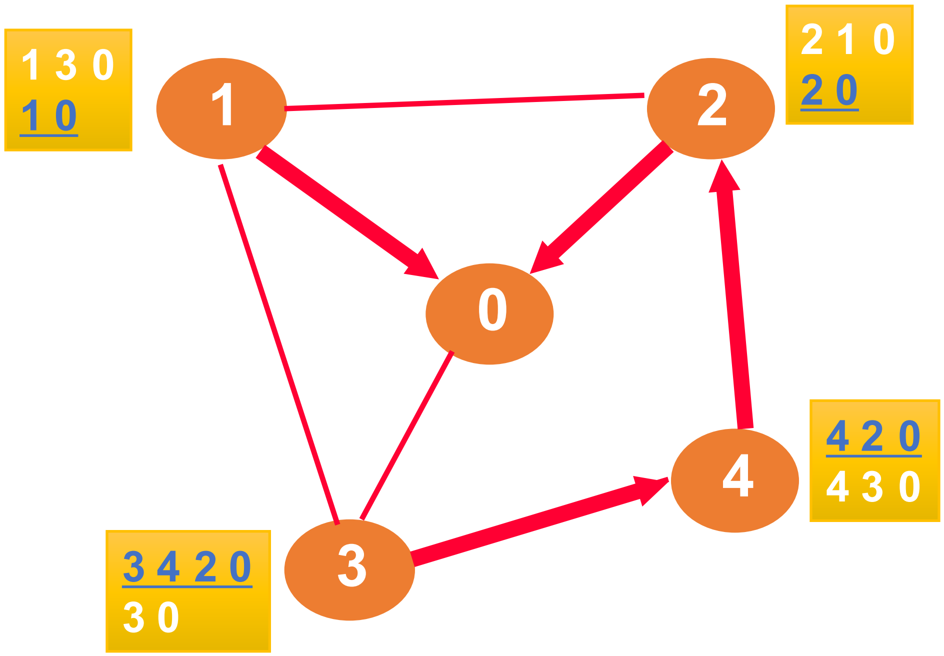
- 2 chooses (2 0)
- 4 chooses (4 2 0)
- 3 chooses (3 4 2 0)
- 1 chooses (1 0)

That was one round of oscillation!

BAD Configuration: No Solution

In BGP-like protocol
- Each node makes local decisions
- At least one node can always improve its path

Result:
- persistent oscillation



Convergence

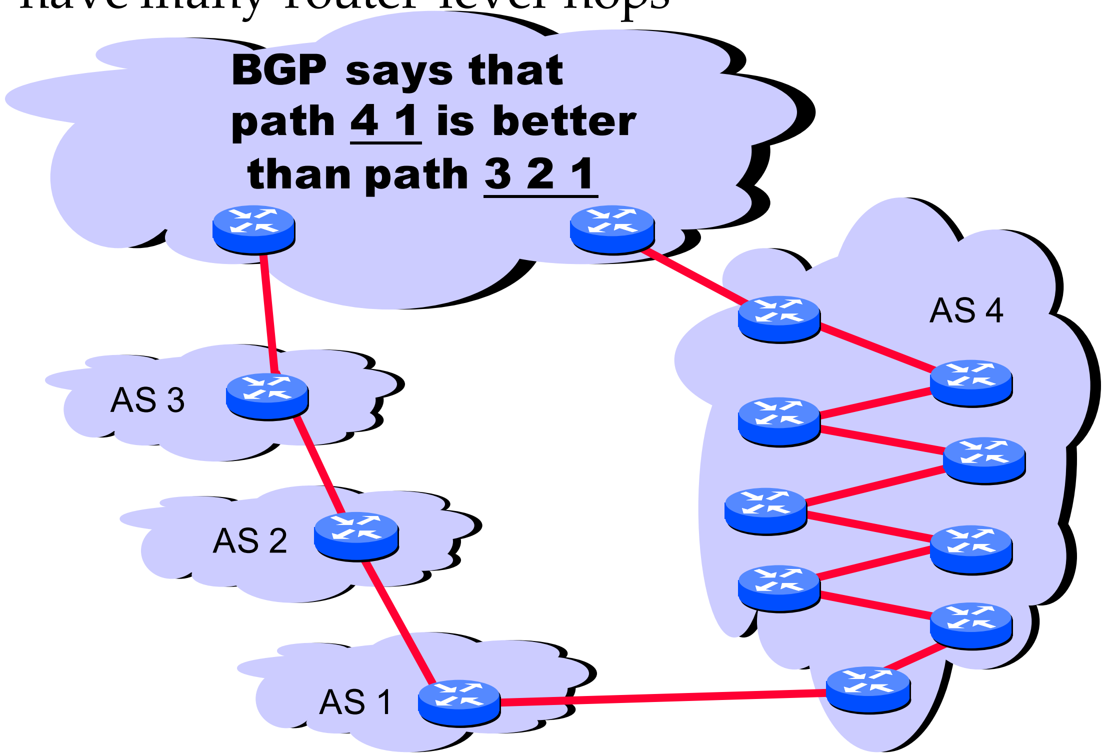
- Result: If all AS policies follow "Gao-Rexford" rules, BGP is guaranteed to converge (safety)
- For arbitrary policies, BGP may fail to converge!
- Why should this trouble us?

Performance Nonissues

- Internal routing (non)
  - Domains typically use “hot potato” routing
  - Not always optimal, but economically expedient
- Policy not about performance (non)
  - So policy-chosen paths aren’t shortest
- AS path length can be misleading (non)
  - 20% of paths inflated by at least 5 router hops

Performance (example)

- AS path length can be misleading
  - An AS may have many router-level hops



Real Performance Issue: Slow convergence

- BGP outages are biggest source of Internet problems

- Labovitz et al. SIGCOMM'97
  - 10% of routes available less than 95% of time
  - Less than 35% of routes available 99.99% of the time

- Labovitz et al. SIGCOMM 2000
  - 40% of path outages take 30+ minutes to repair

- But most popular paths are very stable

BGP Misconfigurations

- BGP protocol is both bloated and underspecified
  - lots of attributes
  - lots of leeway in how to set and interpret attributes
  - necessary to allow autonomy, diverse policies
  - but also gives operators plenty of rope

- Much of this configuration is manual and *ad hoc*
- Core abstraction is fundamentally flawed
  - disjoint per-router configuration to effect AS-wide policy
  - now strong industry interest in changing this!

BGP: How did we get here?

- BGP was designed for a different time
  - before commercial ISPs and their needs
  - before address aggregation
  - before multi-homing

- 1989 : BGP-1 [RFC 1105]
  - Replacement for EGP (1984, RFC 904)
- 1990 : BGP-2 [RFC 1163]
- 1991 : BGP-3 [RFC 1267]
- 1995 : BGP-4 [RFC 1771]
  - Support for Classless Interdomain Routing (CIDR)

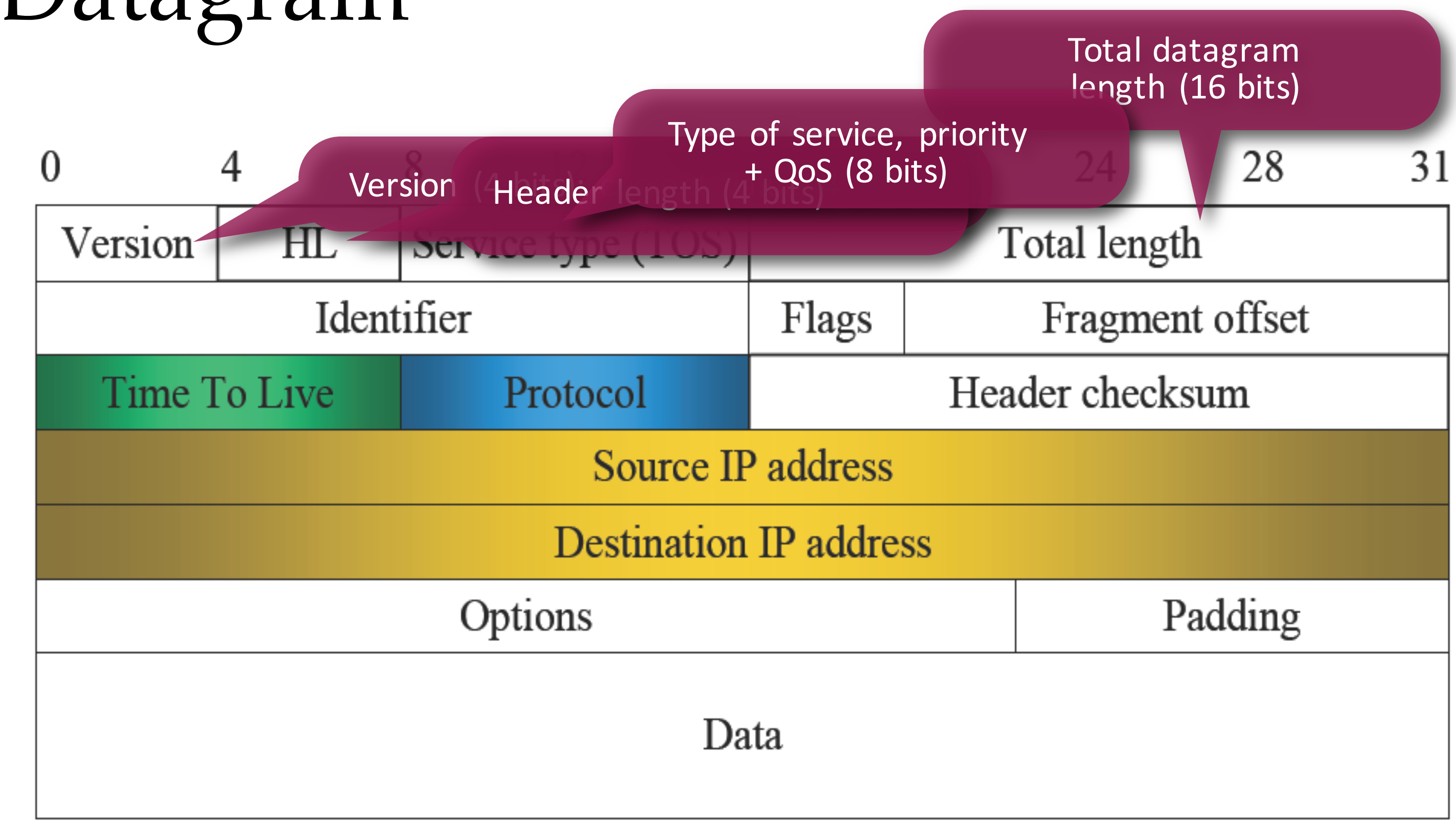
- We don't get a second chance: `clean slate' designs virtually impossible to deploy

- Thought experiment: how would you design a policy-driven interdomain routing solution? How would you deploy it?

Internet Protocol (IP)

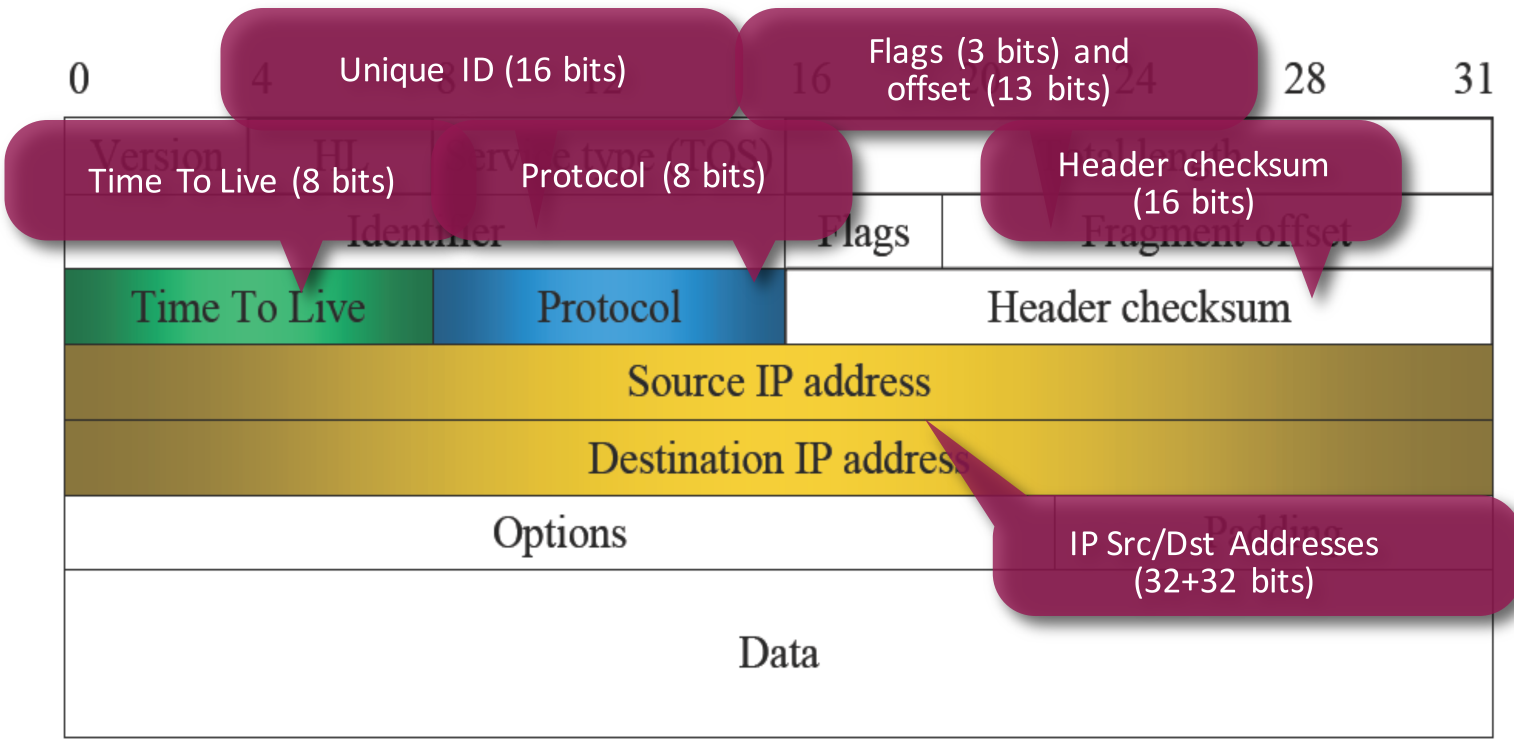
- IP protocol represents the “glue” of the Internet
- Provides a connectionless, unreliable, best-effort datagram delivery service (delivery, integrity, ordering, non-duplication, and bandwidth is not guaranteed)
- IP datagrams can be exchanged between any two nodes (provided they both have an IP address)
- For direct communication IP relies on a number of different lower-level protocols, e.g., Ethernet, Token Ring, FDDI, RS-232

IP Datagram



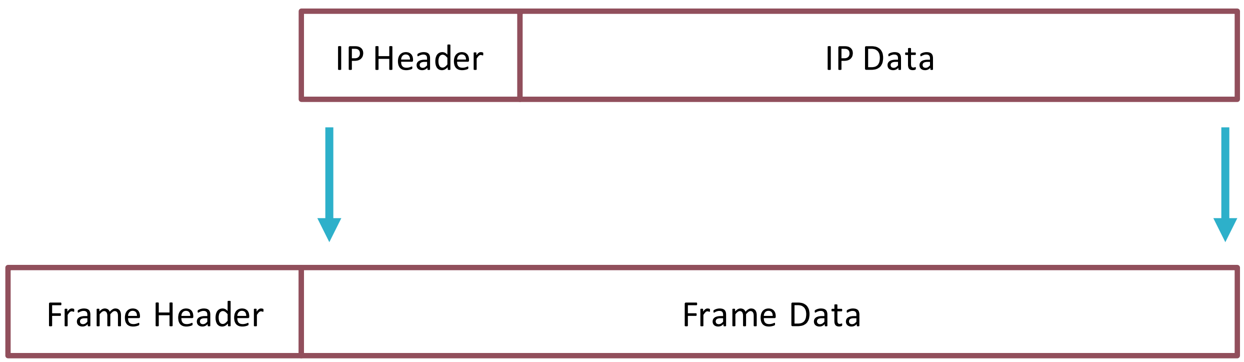
Normal size: 20 bytes

IP Datagram



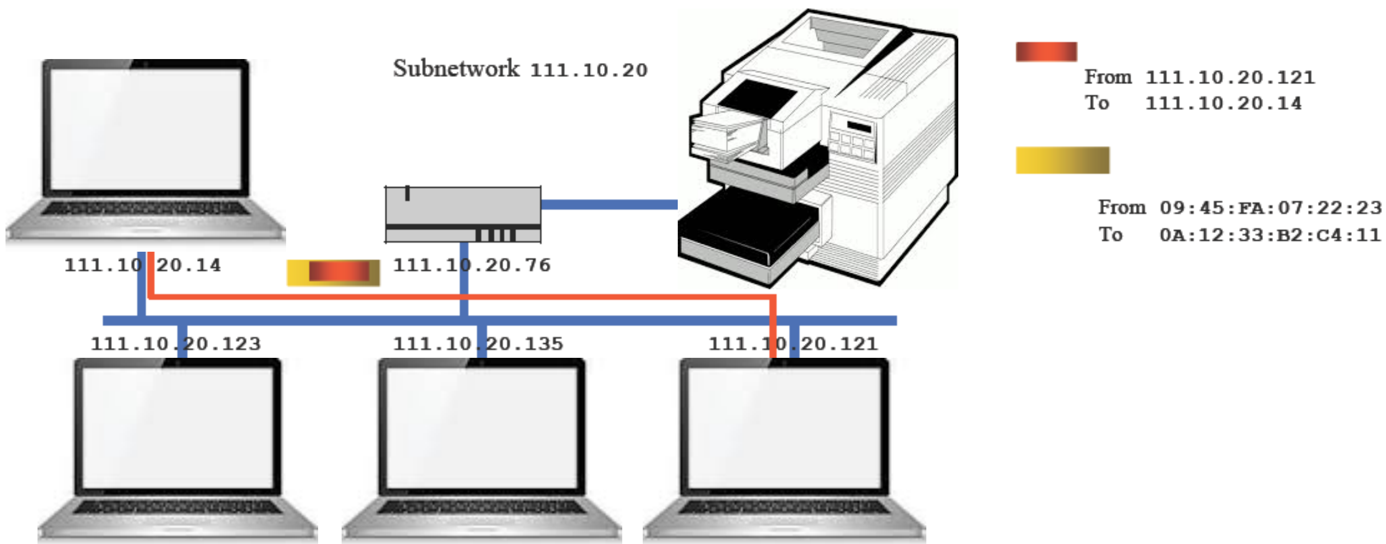
Normal size: 20 bytes

IP Encapsulation



IP: Direct Delivery

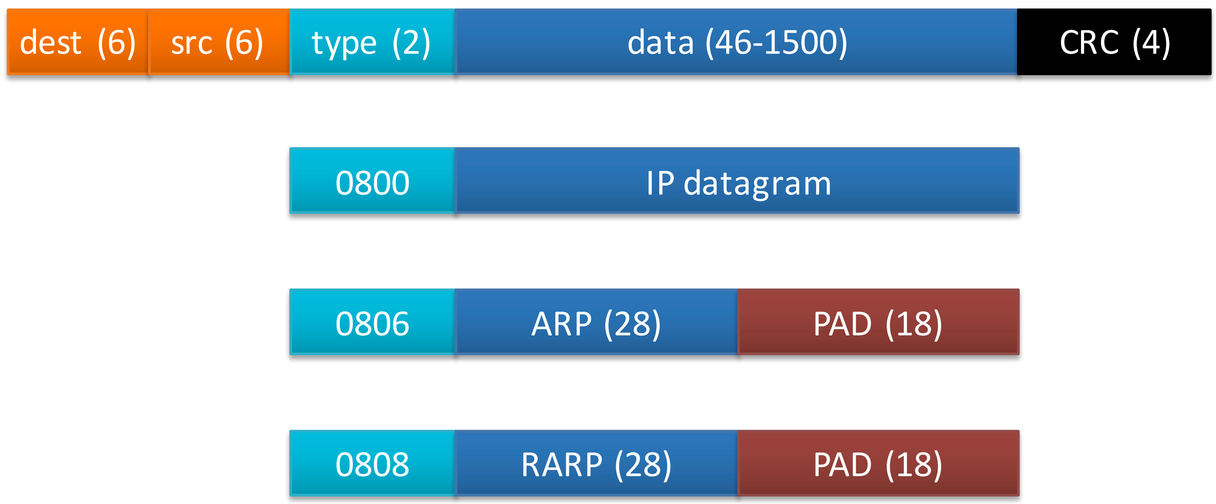
- If two hosts are in the same physical network the IP datagram is encapsulated in a lower level protocol and delivered directly



Ethernet

- Widely-used link-layer protocol
- Uses CSMA/CD
  - (Carrier Sense, Multiple Access with Collision Detection)
- Destination address
  - 48 bits (e.g., 09:45:FA:07:22:23)
- Source address: 48 bits
- Type: 2 bytes (IP, ARP, RARP)
- Data:
  - Min 46 bytes (padding may be needed)
  - Max 1500 bytes
- CRC: Cyclic Redundancy Check, 4 bytes

Ethernet Frame



Address Resolution Protocol

- ARP maps IP addresses to link-level addresses associated with the peer's hardware interface (e.g., Ethernet) to be used in direct delivery

- ARP messages encapsulated in the underlying link level protocol

Address Resolution Protocol

- Host A wants to know hardware address associated with IP address of host B
  - Broadcasts special message to all hosts on the same physical link
  - B answers with message containing its own link-level address
  - A keeps the answer in its cache
- To optimize traffic, when A sends its request, A includes its own IP address
  - The receiver of the ARP request will cache the requester mapping

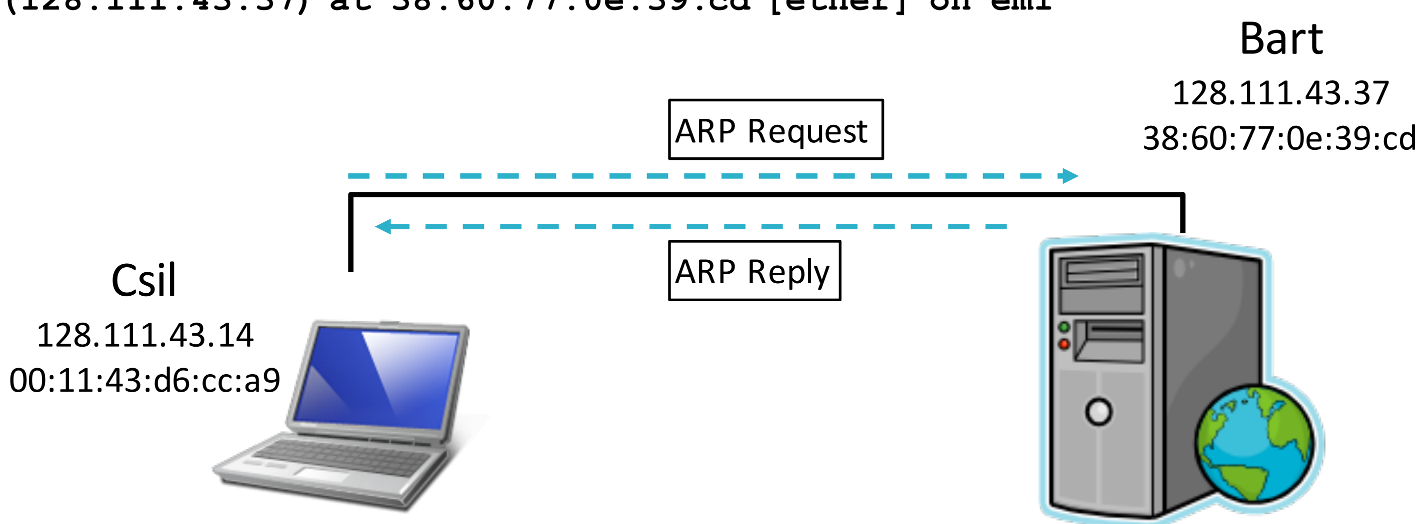
ARP Messages

Hw type	Prot type	Hw size	Prot size	Op	SendEther	SendIP	TargEther	TargIP

- Hardware (2 bytes), protocol (2 bytes), hardware size (1 byte), and protocol size (1 byte) specify the link and network addresses to be mapped (usually Ethernet and IP, respectively) [0x0001, 0x0800, 6, 4]
- OP field specifies if this is an ARP request or an ARP reply (1= ARP request, 2=ARP reply)
- Sender Ethernet/IP: data of the requester
- Target Ethernet: empty in a request
- Target IP: requested IP address

ARP Request

```
csil Wed Jan 13(12:48am)[~]:-> arp -a
csworld43 (128.111.43.1) at 00:d0:2b:fb:3d:00 [ether] on em1
csil Wed Jan 1(12:48am)[~]:-> ping bart
PING bart (128.111.43.37) 56(84) bytes of data.
64 bytes from bart (128.111.43.37): icmp_seq=1 ttl=64 time=0.697
ms
...
csil Wed Jan 13(12:48am)[~]:-> arp -a
csworld43 (128.111.43.1) at 00:d0:2b:fb:3d:00 [ether] on em1
bart (128.111.43.37) at 38:60:77:0e:39:cd [ether] on em1
```



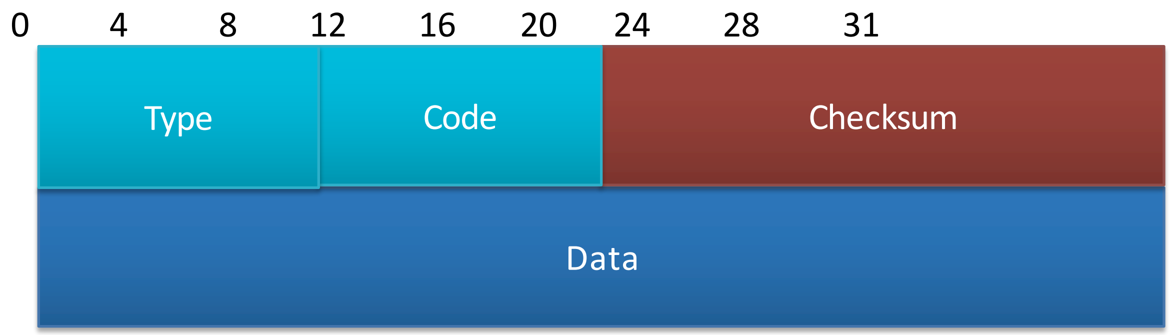
Interesting use of ARP

- Circle: 3rd party device for per-device Internet access control
  - e.g. disconnect dad's iPhone at 11pm
  - e.g. block any connections from kid's iPad to Amazon.com
- Mechanism? ARP jamming/spoofing

Internet Control Message Protocol

- ICMP used to exchange control/error messages about delivery of IP datagrams
- ICMP messages encapsulated inside IP datagrams
- ICMP messages can be:
  - Requests
  - Responses
  - Error messages
    - An ICMP error message includes the header and a portion of the payload (usually the first 8 bytes) of the offending IP datagram

Message Format



ICMP Messages

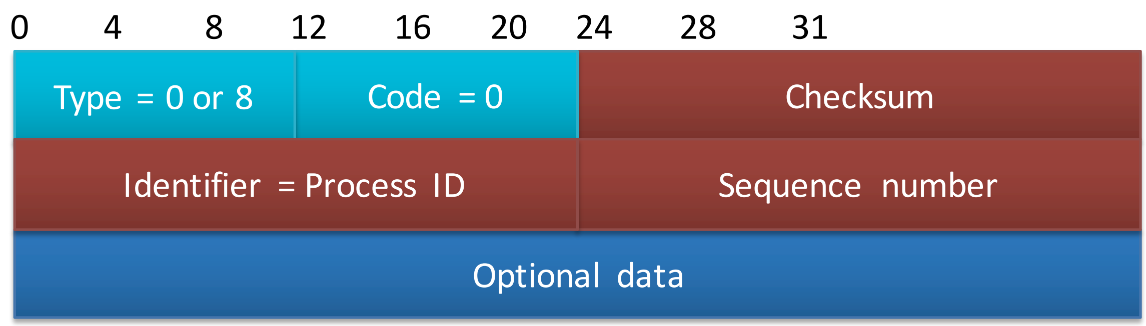
- Address mask request/reply: used by diskless systems to obtain the network mask at boot time
- Timestamp request/reply: used to synchronize clocks
- Source quench: used to inform about traffic overloads
- Parameter problem: used to inform about errors in the IP datagram fields

ICMP Messages

- Echo request/reply: used to test connectivity (ping)
- Time exceeded: used to report expired datagrams (TTL = 0)
- Redirect: used to inform hosts about better routes (gateways)
- Destination unreachable: used to inform a host of the impossibility to deliver traffic to a specific destination

ICMP Echo Request/Reply

- Used by the ping program

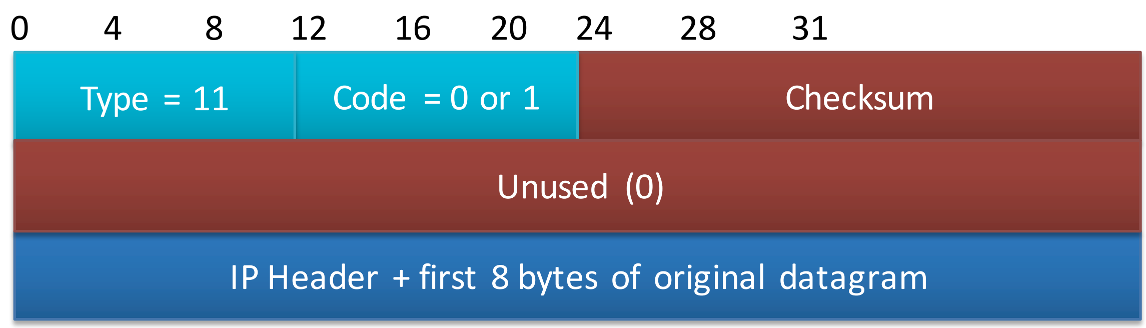


Ping

```
htzheng@groot:~$  ping www.linkedin.com
PING pop-ech2-alpha.www.linkedin.com  (108.174.11.1) 56(84) bytes of data.
64 bytes from 108-174-11-1.fwd.linkedin.com  (108.174.11.1):  icmp_seq=1  ttl=57  time=1.47  ms
64 bytes from 108-174-11-1.fwd.linkedin.com  (108.174.11.1):  icmp_seq=2  ttl=57  time=1.83  ms
64 bytes from 108-174-11-1.fwd.linkedin.com  (108.174.11.1):  icmp_seq=3  ttl=57 time=1.45  ms
64 bytes from 108-174-11-1.fwd.linkedin.com  (108.174.11.1):  icmp_seq=4  ttl=57 time=1.51  ms
64 bytes from 108-174-11-1.fwd.linkedin.com  (108.174.11.1):  icmp_seq=5  ttl=57 time=1.50  ms
64 bytes from 108-174-11-1.fwd.linkedin.com  (108.174.11.1):  icmp_seq=6  ttl=57 time=1.46  ms
^C
--- pop-ech2-alpha.www.linkedin.com  ping statistics  ---
6 packets transmitted,  6 received,  0% packet loss, time 5007ms
rtt min/avg/max/mdev  = 1.452/1.540/1.836/0.143  ms
htzheng@groot:~$
```

ICMP Time Exceeded

- Used when
  - TTL becomes zero (code = 0)
  - The reassembling of a fragmented datagram times out (code =1)



Traceroute

- ICMP Time Exceeded messages used by traceroute to determine the path used to deliver a datagram
  - A series of IP datagrams are sent to the destination node
  - Each datagram has an increasing TTL field (starting at 1)
  - From the ICMP Time exceeded messages returned by the intermediate gateways, can reconstruct route from source to destination
- Note: traceroute allows one to specify loose source routing via -g option (not very useful now, most routers disabled src routing)
- Tools immensely useful (topology mapping)

Traceroute

htzheng@groot:~$ traceroute www.facebook.com

traceroute to www.facebook.com (157.240.18.35), 30 hops max, 60 byte packets

1 d19-07-200-v250.uchicago.edu (128.135.250.3) 0.618 ms 0.806 ms 0.947 ms

2 d19-07-200-to-b65-ll129-300.p2p.uchicago.net (10.5.1.12) 1.812 ms d19-07-200-to-h01-391-300.p2p.uchicago.net (10.5.1.46) 13.785 ms d19-07-200-to-b65-ll129-300.p2p.uchicago.net (10.5.1.12) 1.807 ms

3 h01-391-300-to-borderfw.p2p.uchicago.net (192.170.192.34) 0.575 ms 0.561 ms 0.617 ms

4 borderfw-to-b65-ll129-500.p2p.uchicago.net (192.170.192.36) 0.815 ms 0.806 ms 0.754 ms

5 r-equinix-isp-ae0-2234.wiscnet.net (216.56.50.65) 18.900 ms 18.905 ms 18.898 ms

6 205.213.119.42 (205.213.119.42) 1.625 ms 1.659 ms 1.718 ms

7 po111.asw01.ord3.tfbnw.net (31.13.24.24) 1.704 ms po111.asw03.ord3.tfbnw.net (31.13.24.88) 1.629 ms po121.asw02.ord3.tfbnw.net (31.13.30.134) 1.614 ms

8 po222.psw01.ord2.tfbnw.net (173.252.64.91) 2.407 ms po232.psw04.ord2.tfbnw.net (157.240.40.43) 1.679 ms po241.psw03.ord2.tfbnw.net (157.240.33.213) 1.672 ms

9 157.240.36.83 (157.240.36.83) 1.665 ms 173.252.67.213 (173.252.67.213) 1.751 ms 157.240.36.87 (157.240.36.87) 1.976 ms

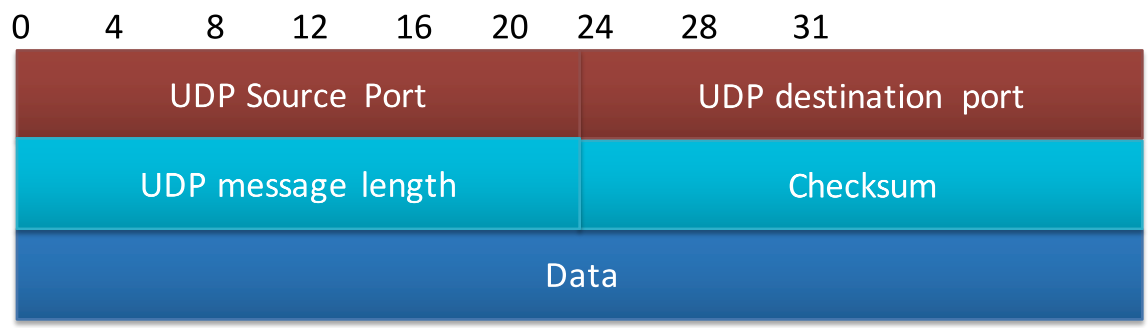
10 edge-star-mini-shv-02-ort2.facebook.com (157.240.18.35) 1.687 ms 2.834 ms 1.782 ms

htzheng@groot:~$

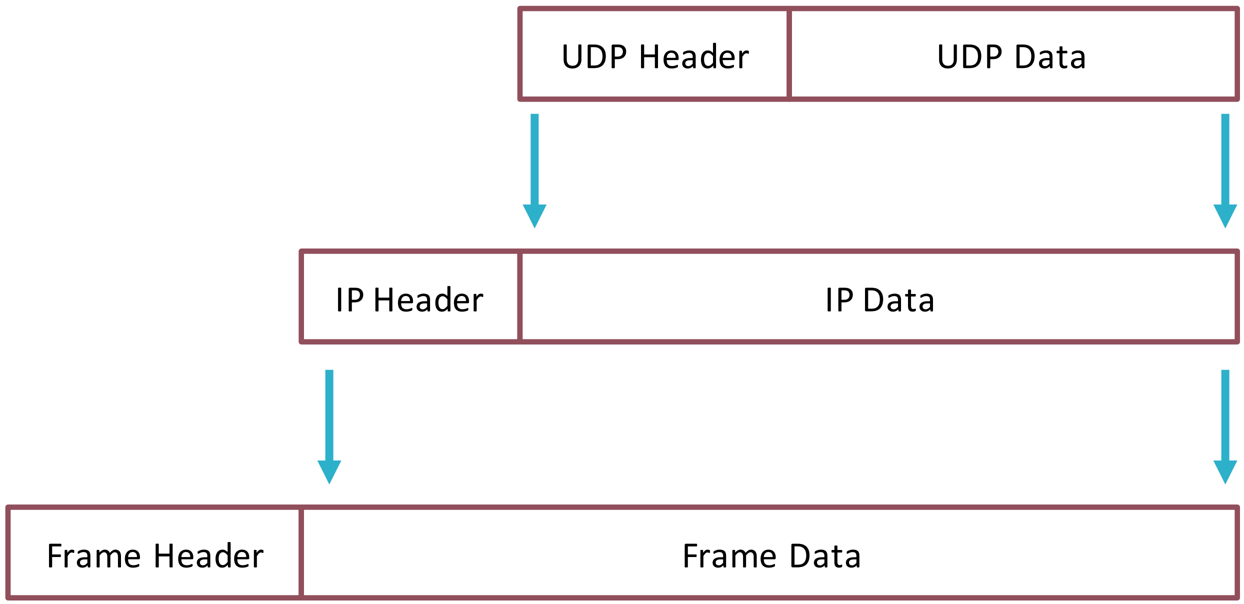
User Datagram Protocol (UDP)

- UDP relies on IP to provide a connectionless, unreliable, best-effort datagram delivery service
  - delivery, integrity, non-duplication, ordering, and bandwidth all not guaranteed
- Introduces the port abstraction that allows one to address different message destinations for the same IP address
- Often used for multimedia (more efficient than TCP) and for services based on request/reply schema (DNS, NIS, NFS, RPC)

UDP Message


0 4 8 12 16 20 24 28 31
UDP Source Port
UDP destination port
UDP message length
Checksum
Data

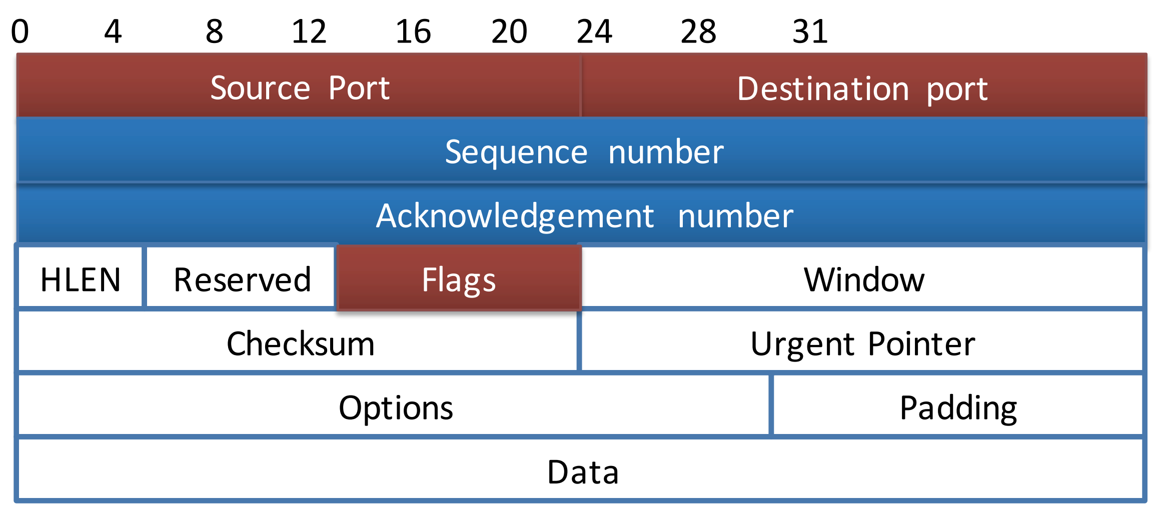
UDP Encapsulation



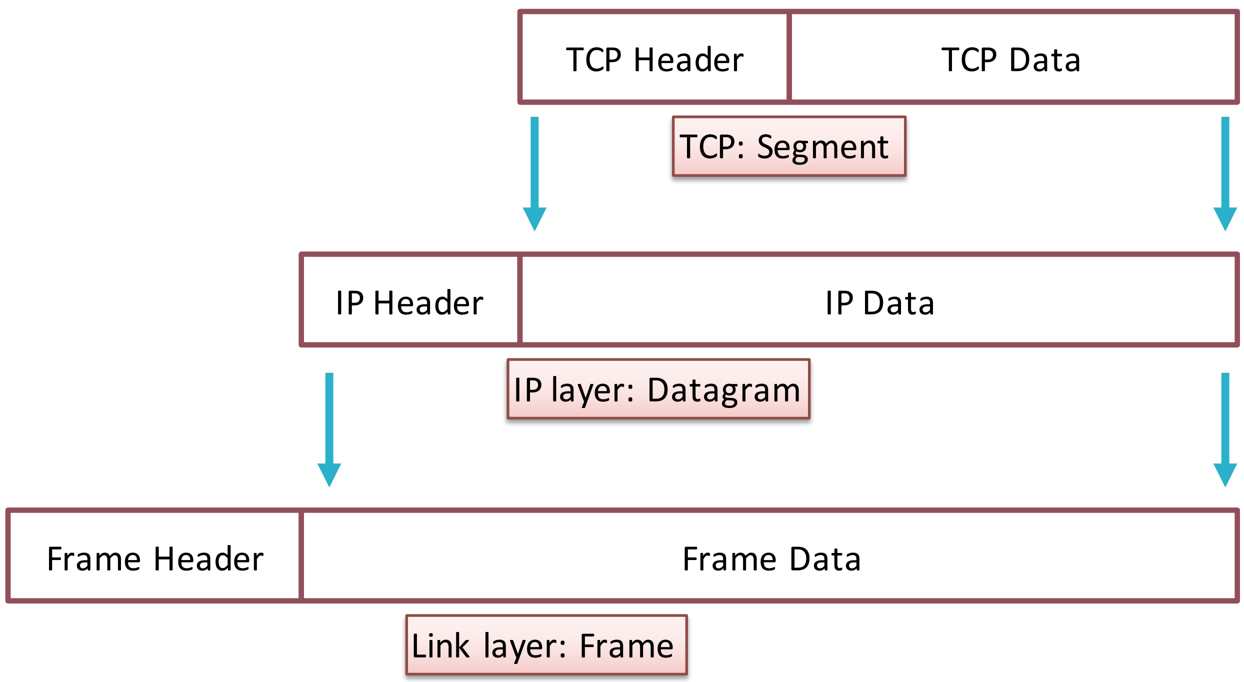
Transmission Control Protocol

- TCP protocol relies on IP to provide a connection-oriented, reliable stream delivery service
  - no loss, no duplication, no transmission errors, correct ordering
- TCP, as UDP, provides the port abstraction
- Allows two nodes to establish a virtual circuit
  - Identified by source IP, destination IP, source port, destination port
  - Virtual circuit composed of two streams
    - full-duplex connection
- IP address/port # pair is sometimes called a socket (and the two streams are called a socket pair)

TCP Segment


0
4
8
12
16
20
24
28
31
Source Port
Destination port
Sequence number
Acknowledgement number
HLEN
Reserved
Flags
Window
Checksum
Urgent Pointer
Options
Padding
Data

TCP Encapsulation



TCP Seq/Ack Numbers

- Sequence # specifies position of the segment data in communication stream
  - SYN=13423: segment payload contains data from byte 13423 to byte 13458
- Acknowledgment # specifies the position of next byte expected from communication partner
  - ACK = 16754: I have received correctly up to byte 16753 in the stream, I expect the next byte to be 16754
- These # used to manage retransmission of lost segments, duplication, flow control

TCP Gives You…

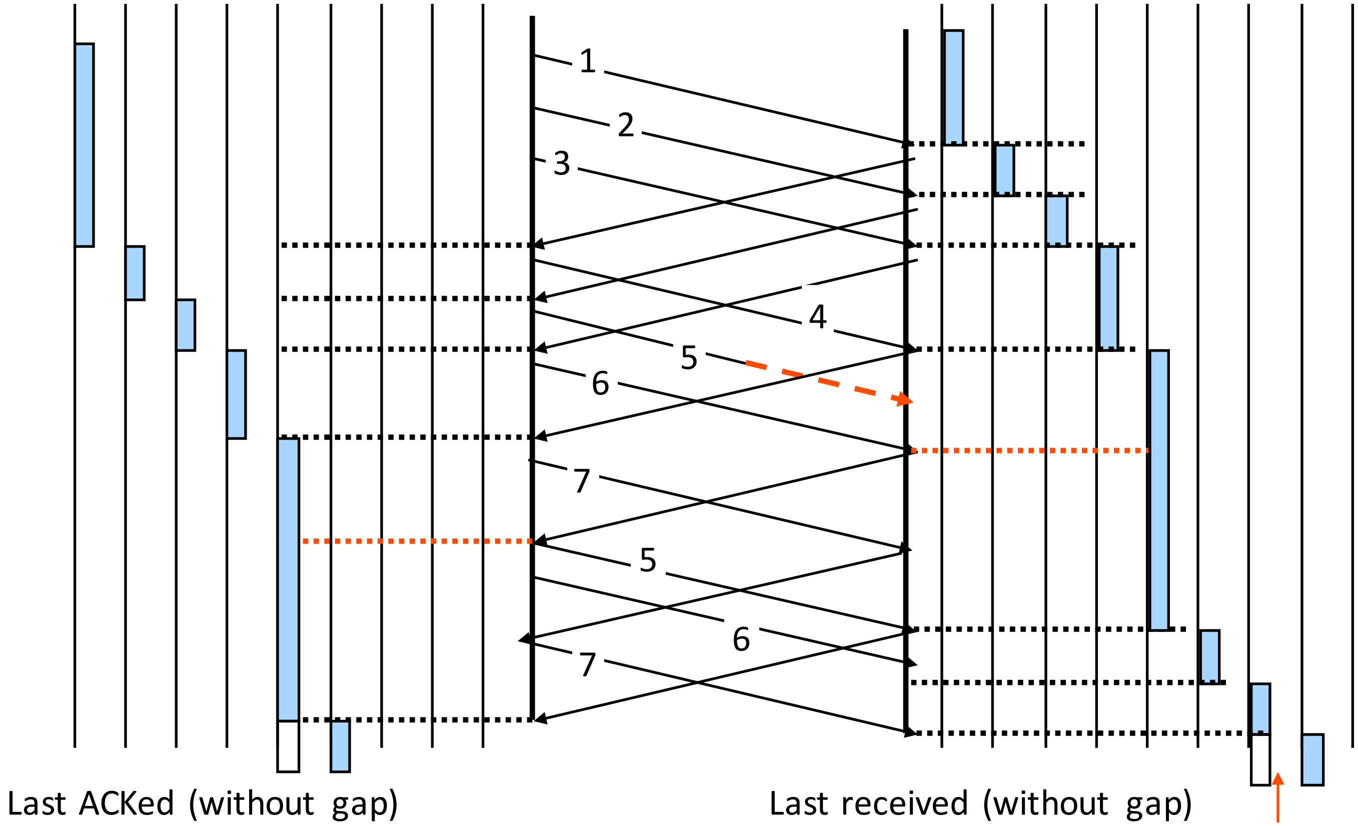
- Flow control
  - Make sure sender sends at rate receiver can handle
- Reliable data transfer
  - Via packet retransmissions
- Congestion control
  - Detects network congestion via lost packets
  - (Initially not included in TCP)

TCP Flow Control Window

- Make sure receiving end can handle data
  - TCP window used to perform flow control
  - Negotiated end-to-end, with no regard to network
- Ends must ensure that no more than W packets are in flight
  - Receiver ACKs packets
  - When sender gets an ACK, it knows packet has arrived
- Segment accepted IFF sequence # is inside window:

  ```
  ack # < sequence # < (ack # + window)
  ```
- Window size can change dynamically to adjust the amount of information sent by the sender

Sliding Window



Observations

- Throughput is ~ (w/RTT)
  - Longer RTT, longer pipe
- Sender has to buffer all unacknowledged packets, because they may require retransmission
- Receiver may be able to accept out-of-order packets, but only up to its buffer limits

TCP Reliability/Error Recovery

- Must retransmit packets that were dropped
- To do this efficiently
  - Keep transmitting whenever possible
  - Detect dropped packets and retransmit quickly
- Requires:
  - Timeouts (with good timers)
  - Other hints that packet were dropped