CIS 457 - Data Communications

Nathan Bowman

Images taken from Kurose and Ross book

TCP Congestion Control -- Basics

Critical that we keep network congestion down to maintain acceptable performance

One natural question is how we determine there is congestion

Congestion control can either be:

- network-assisted, or
- end-to-end

By default, IP does not provide support for congestion control

TCP must determine some way for end systems to infer congestion is occurring

Newer modifications to TCP/IP allow network layer to assist, and we will briefly see how later, but for the most part we stick to basic protocol

To implement congestion control, TCP sender will voluntarily lower its own sending rate when congestion is detected

Three major questions from this:

- how does sender limit its rate? (mechanism)
- how does sender detect congestion?
- what algorithm does sender use to determine rate? (policy)

We deal with these questions in order

Recall variables related to flow control

`LastByteRead`, `rwnd`, and so on

Sender has another variable, the **congestion window** (`cwnd`), to limit amount of unACK'd data it can send at once

Formula becomes

```
LastByteSent - LastByteAcked <= min(cwnd, rwnd)
```

To focus on congestion control, we will assume receiver has unlimited buffer, so `rwnd` is infinite and therefore irrelevant

Also assume sender always has data it wishes to send

Sender can send cwnd bytes of data at once, and it takes roughly RTT for them to be ACK'd

(We are ignoring transmission delay and other delays for sake of simplification)

Send rate thus roughly cwnd/RTT bytes/second

Adjusting cwnd up or down increases or decreases sending rate

`cwnd` gives us mechanism for controlling rate at which data sent

Next, need to know when to apply mechanism -- how to tell when congestion occurs

TCP hosts know only what they send to and receive from network, so all decisions must be based on this local information

TCP detects congestion via "loss events":

- timeouts
- triple duplicate ACKs

Either of these could be caused by overtaxed router dropping packet, which indicates congestion

In general, successful ACK indicates congestion is not an issue, so `cwnd` increased

Loss event indicates congestion may be present, so decrease `cwnd`

On fast networks (low latency, high bandwidth) ACKs arrive quickly, so rate changes quickly

On slower networks, rate changes more slowly

Because TCP uses ACKs to control transmission rate, said to be "self-clocking"

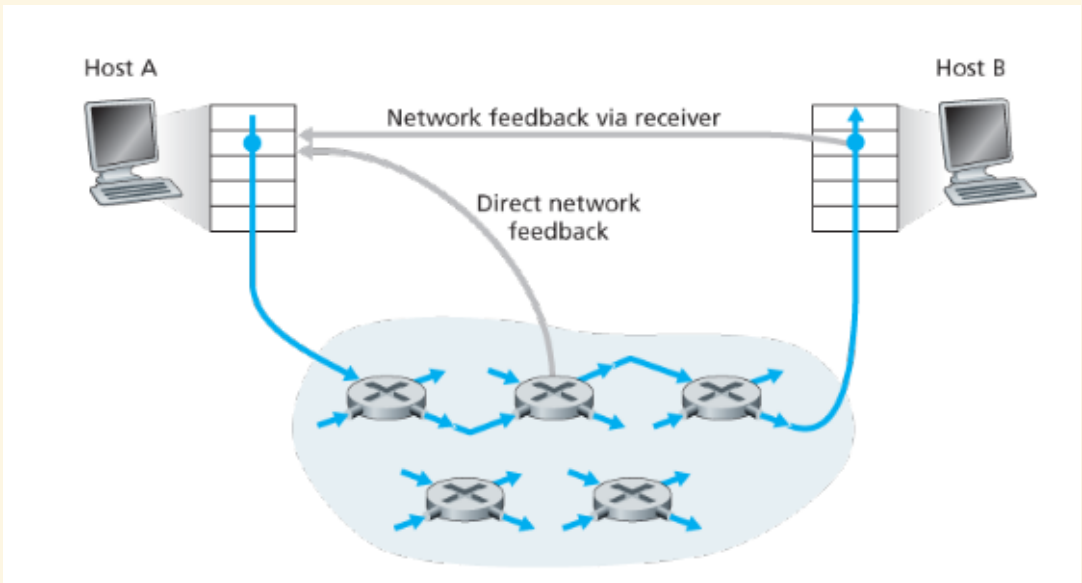
Final question is policy: how much do we lower or raise rate based on ACKs and losses?

Consider this in future lecture

Briefly look at network-assisted congestion control

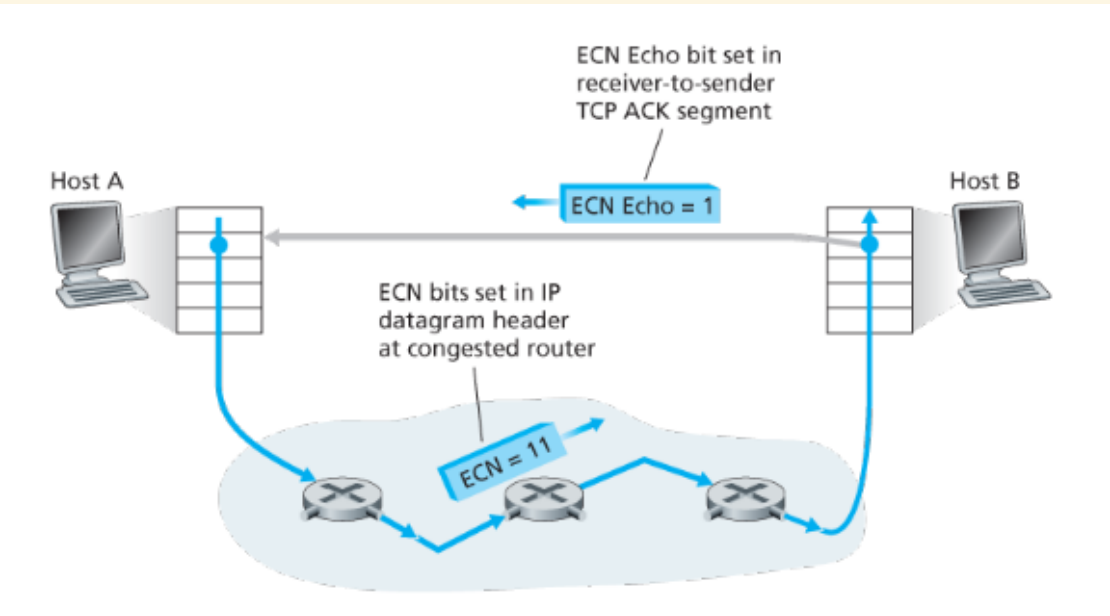
Network provides explicit feedback on congestion to hosts, rather than hosts needing to infer for themselves

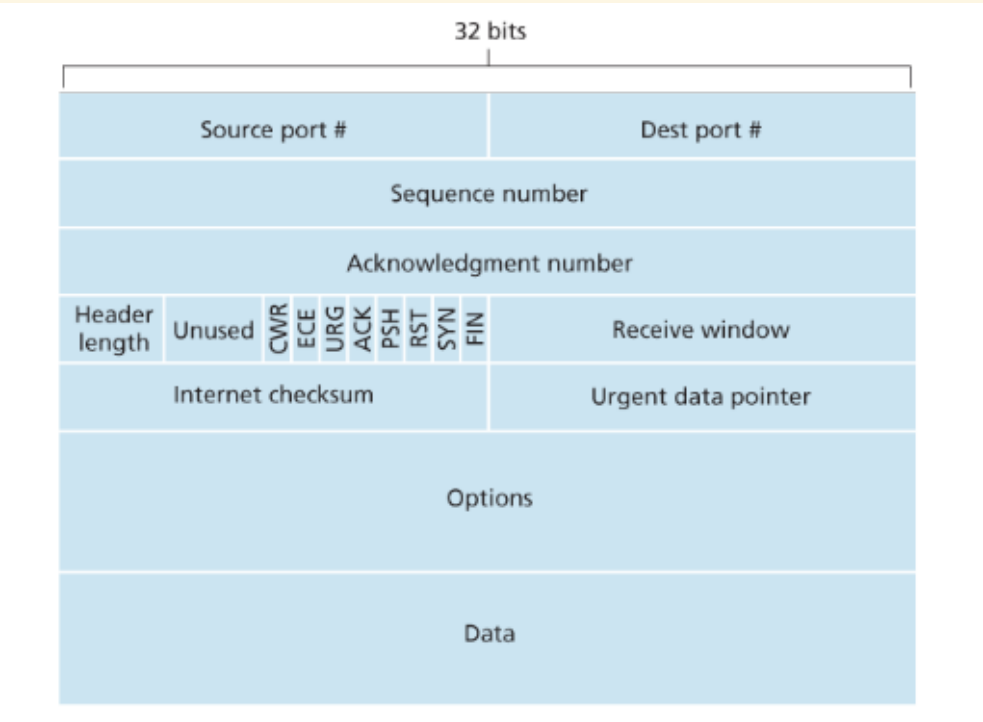
Feedback can be sent directly from network to sending host, or can be done indirectly via receiving host


Host A
Host B
Network feedback via receiver
Direct network
feedback

Explicit congestion notification (ECN) allows IP to notify TCP of congestion

ECN works through indirect feedback from receiver


ECN Echo bit set in
receiver-to-sender
TCP ACK segment
Host A
ECN Echo = 1
Host B
ECN bits set in IP
datagram header
at congested router
ECN = 11


32 bits
Source port #
Dest port #
Sequence number
Acknowledgment number
Header length
Unused
CWR
ECE
URG
ACK
PSH
RST
SYN
FIN
Receive window
Internet checksum
Urgent data pointer
Options
Data

Using ECN gives TCP hosts additional way to detect congestion, but does not change how TCP host reacts to congestion