Congestion Control

Phenomenon: when too much traffic enters into system, performance degrades

⟶ excessive traffic can cause congestion

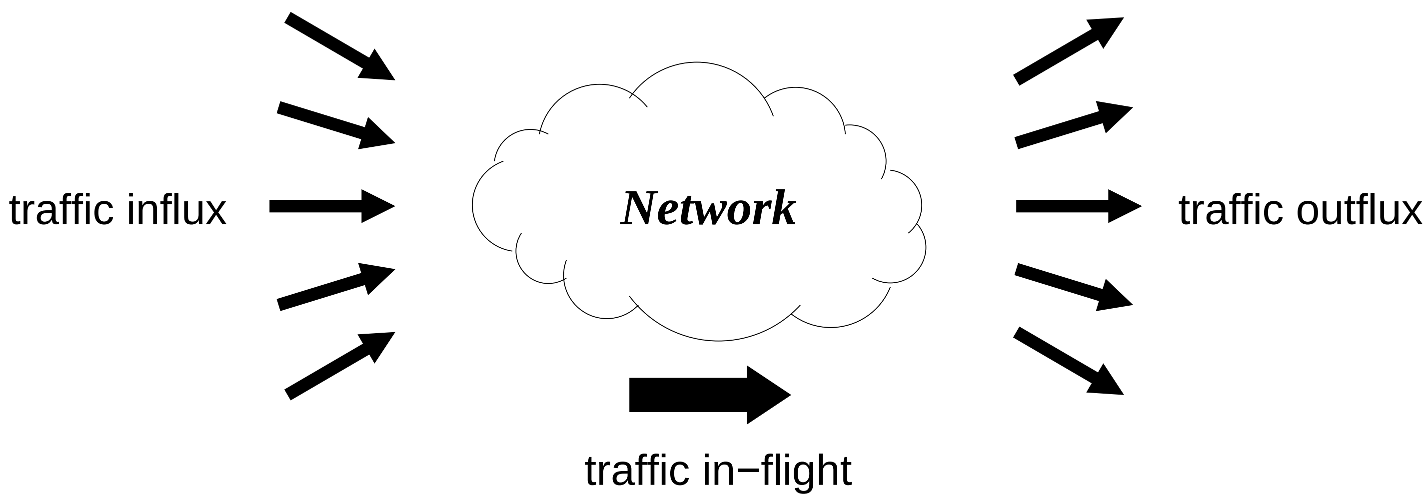
Problem: regulate traffic influx such that congestion does not occur

⟶ not too fast, not too slow

⟶ congestion control

⟶ first question: what is congestion?

Set-up: "traffic influx vs. outflux viewpoint"



- traffic influx: $\lambda(t)$ "offered load"

  $\rightarrow$ rate: bps (or pps) at time t

- traffic outflux: $\gamma(t)$ "throughput"

  $\rightarrow$ rate: bps (or pps) at time t

- traffic in-flight: $Q(t)$ "load"

  $\rightarrow$ volume: total packets in transit at time t

Examples:

Highway system:

- traffic influx: no. of cars entering highway per second
- traffic outflux: no. of cars exiting highway per second
- traffic in-flight: no. of cars traveling on highway

$\longrightarrow$ at time instance t

California Dept. of Transportation (Caltrans)

Water faucet and sink:

- traffic influx: water influx per second
- traffic outflux: water outflux per second
- traffic in-flight: water level in sink

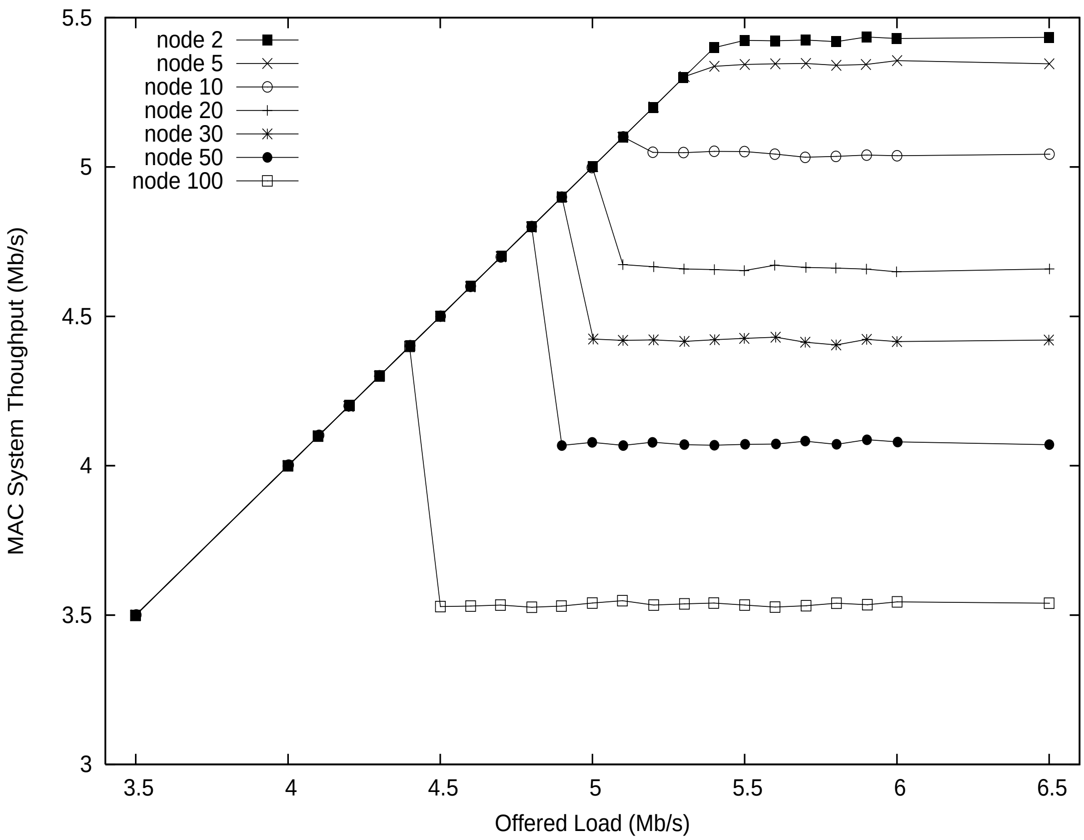
$\longrightarrow$ "congestion?"

faucet.com

Thermostat . . .

802.11b WLAN:

- Throughput



⟶ unimodal or bell-shaped

⟶ recall: less pronounced in real systems

What we can regulate or control:

$\longrightarrow$ traffic influx rate $\lambda(t)$

$\longrightarrow$ only decision variable under our control

Ex.:

- Faucet knob in water sink
- Temperature needle in thermostat
- Cars entering onto highway
- Traffic sent by UDP or TCP

What we cannot control: the rest

$\longrightarrow$ except in the long run: bandwidth planning

$\longrightarrow$ does scheduling (e.g., FIFO, round robin) help?

$\longrightarrow$ Kleinrock's conservation law: "zero-sum pie"

How does in-flight traffic or load $Q(t)$ vary?

Consider two time instances t and $t+1$.

At time $t+1$:

$$Q(t+1) = Q(t) + \lambda(t) - \gamma(t)$$

- $Q(t)$: what was there to begin with
- $\lambda(t)$: what newly arrived
- $\gamma(t)$: what newly exited (delivered to applications)
- $\lambda(t) - \gamma(t)$: net influx
- $Q(t)$ cannot be negative

  $\rightarrow Q(t+1) = \max\{0, Q(t) + \lambda(t) - \gamma(t)\}$
- missing item?

Pseudo Real-Time Multimedia Streaming

$\longrightarrow$ e.g., RealPlayer, iTunes, Internet radio

$\longrightarrow$ "pseudo" because of prefetching trick

$\longrightarrow$ application is given headstart: few seconds

$\longrightarrow$ fill buffer & prevent from becoming empty

Steps involved:

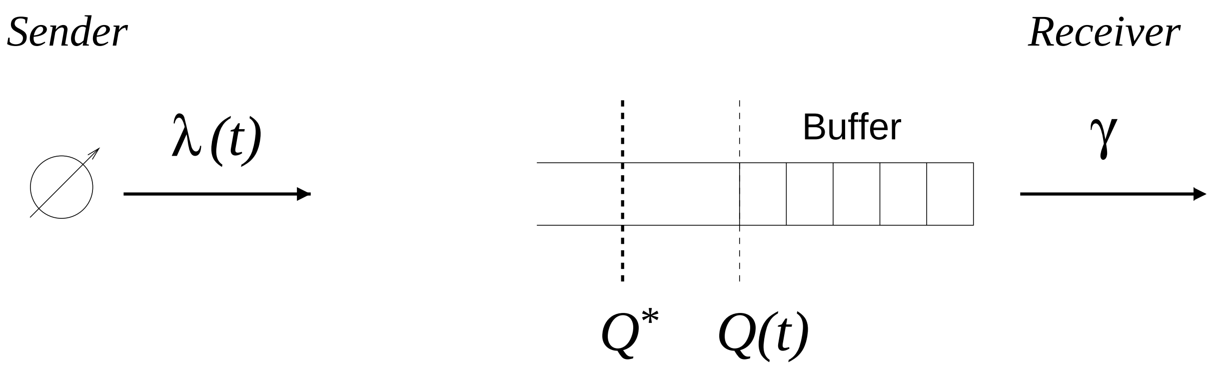
- prefetch X seconds worth of audio/video data
- causes initial delayed playback

  $\rightarrow$ e.g., a couple of seconds delay after click
- keep fetching audio/video data such that X seconds worth of future data resides in receiver's buffer

  $\rightarrow$ hides spurious congestion from user

  $\rightarrow$ continuous playback experience

Pseudo real-time application architecture:



- $Q(t)$: current buffer level
- Q^*: desired buffer level
- γ: throughput, i.e., playback rate

  $\rightarrow$ e.g., for video 24 frames-per-second (fps)

Goal: vary $\lambda(t)$ such that $Q(t) \approx Q^*$

$\longrightarrow$ don't buffer too much: memory cost

$\longrightarrow$ don't buffer too little: cannot hide congestion

Other applications:

⟶ pseudo real-time set-up is highly versatile

⟶ captures many scenarios

Ex. 1: Router congestion control

⟶ active queue management (AQM)

- receiver is a router
- Q^* is desired buffer occupancy/delay at router
- router throttles sender(s) to maintain Q^*

⟶ send control messages to senders

⟶ slow down, go faster, stay put

Ex. 2: Slightly modified Internet standard

$\longrightarrow$ ECN (explicit congestion notification)

- two bits in IPv4 TOS field

  → ECT: ECN capable transport (bit 6)

  → CE: congestion experienced (bit 7)

- congested router marks ECT
- supported in most routers, default not turned on
- requires TCP sender/receiver changes

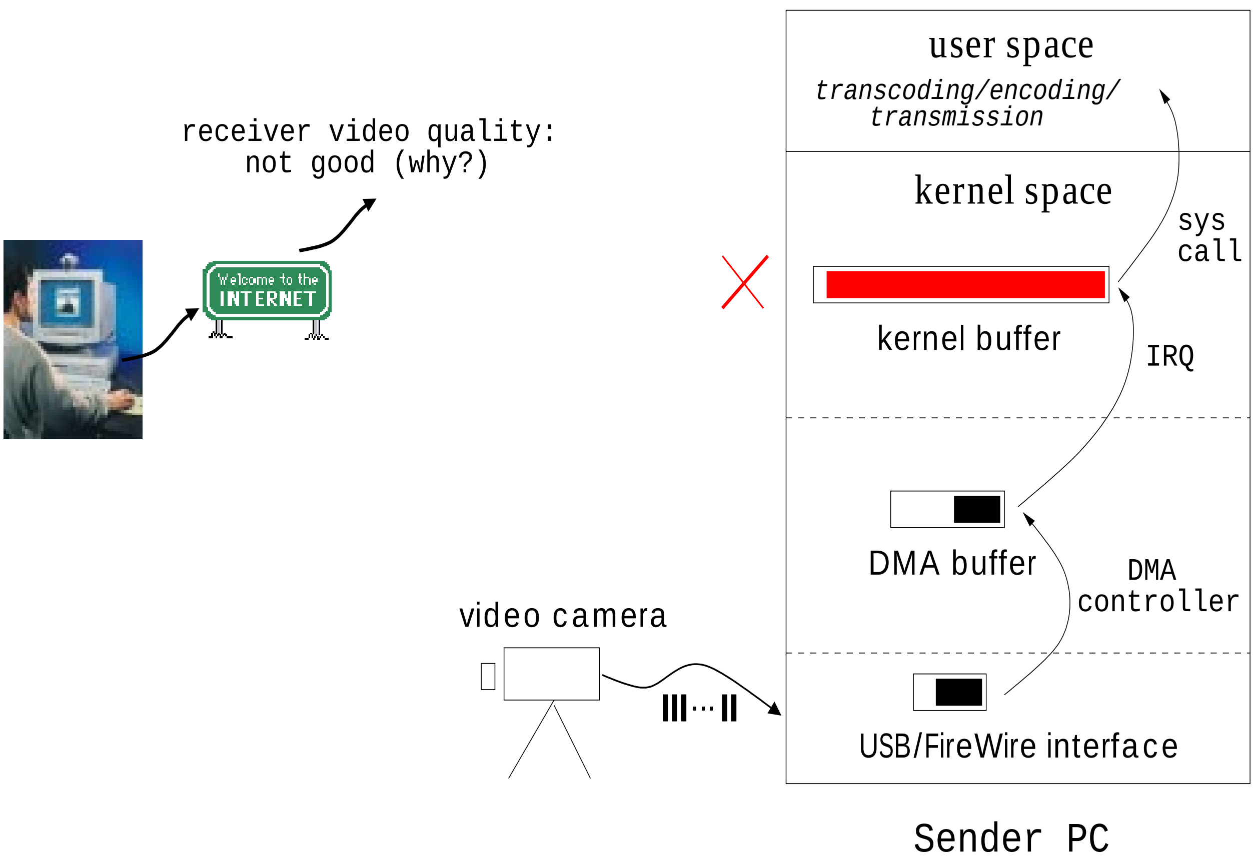
  → sender slows down if CE bit turned on in ACK

Ex. 3: Desktop videoconferencing

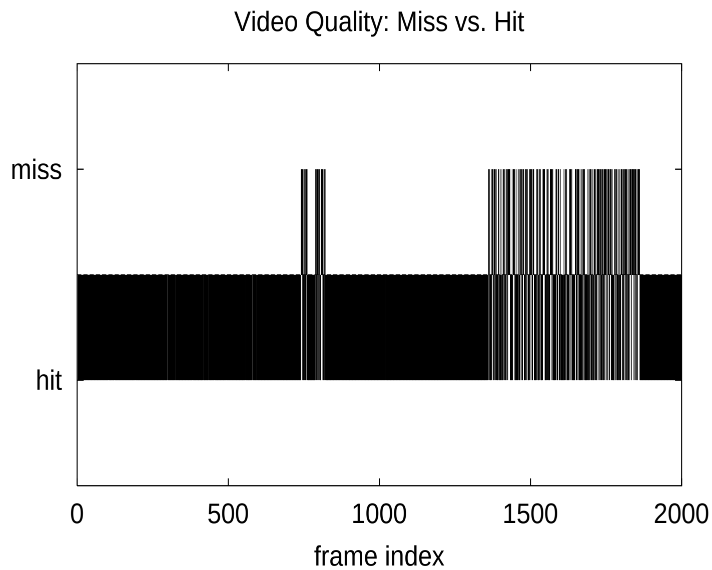
⟶ e.g., AOL, MSN, Skype, Yahoo

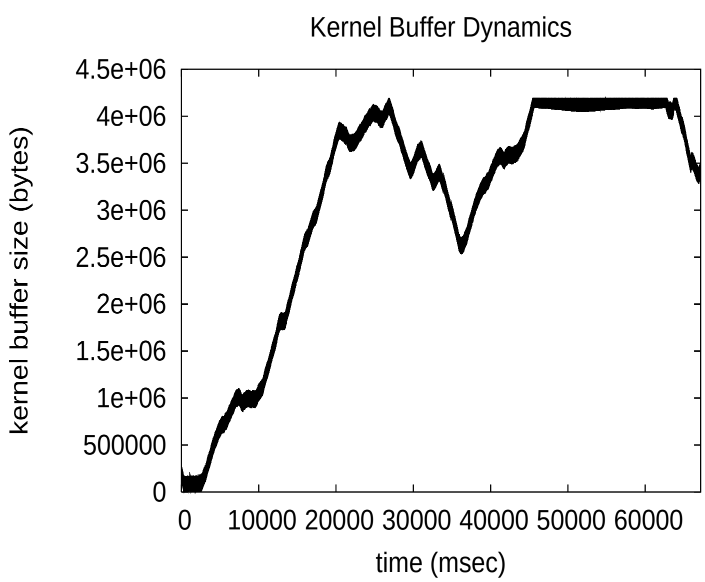
⟶ video quality is not good: why?

⟶ misconception: network



Performance consequences:





Thus: pseudo real-time multimedia streaming application of congestion control

- $\longrightarrow$ producer/consumer rate mismatch problem
- $\longrightarrow$ called "flow control"
- $\longrightarrow$ origin of "congestion control"
- $\longrightarrow$ sender-receiver point-to-point link

Note: in OS

- $\longrightarrow$ focus on orderly access of shared data structure
- $\longrightarrow$ i.e., kernel buffer
- $\longrightarrow$ e.g., use of counting semaphores
- $\longrightarrow$ necessary but insufficient

What to do to achieve goal (i.e., $Q(t) = Q^*$)?

Basic idea:

- if $Q(t) = Q^*$ do nothing
- if $Q(t) < Q^*$ increase $\lambda(t)$
- if $Q(t) > Q^*$ decrease $\lambda(t)$

  $\longrightarrow$ "control law"

Protocol implementation:

- control action undertaken at sender

  $\rightarrow$ smart sender/dump receiver

  $\rightarrow$ when might the opposite be better?

- receiver informs sender of Q^* and $Q(t)$

  $\rightarrow$ feedback packet ("control signaling")

  $\rightarrow$ or up/down (binary)

  $\rightarrow$ or $Q^* - Q(t)$

Key question in feedback congestion control: **how much** to increase or decrease $\lambda(t)$

$\longrightarrow$ we already know which direction

Desired state of the system:

$\longrightarrow$ i.e., target operating point

$$\text{want:} \quad Q(t) = Q^* \text{ and } \lambda(t) = \gamma$$

$\longrightarrow$ why can it not be anything else?

Start from:

$\longrightarrow$ empty buffer and no sending rate at start

$$\text{i.e.,} \quad Q(t) = 0 \text{ and } \lambda(t) = 0$$

Time evolution (or dynamics): track $Q(t)$ and $\lambda(t)$