Congestion Control

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

\[ \rightarrow \text{excessive traffic can cause congestion} \]

Problem: regulate influx of traffic such that congestion does not occur

\[ \rightarrow \text{congestion control} \]

Need to understand:

- What is congestion?
- How do we prevent or manage it?
Traffic influx/outflux picture:

- traffic influx: $\lambda(t)$
  - rate: packets-per-second at time $t$
- traffic outflux: $\gamma(t)$
  - rate: packets-per-second at time $t$
- traffic in-flight: $Q(t)$
  - 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

\[\rightarrow\text{ all at time instance } t\]
Water faucet and sink:

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

→ “congestion?”

Thermostat ...
What we can regulate or control:

\[ \rightarrow \text{traffic influx rate } \lambda(t) \]

\[ \rightarrow \text{e.g., faucet knob in water sink} \]

How does in-flight traffic \( Q(t) \) vary?
Discrete time case:

\[ Q(t + 1) = Q(t) + \lambda(t) - \gamma(t) \]

- \( \lambda(t) - \gamma(t) \): net influx

- \( Q(t) \) cannot be negative
  \[ \rightarrow Q(t + 1) = \max\{Q(t) + \lambda(t) - \gamma(t), 0\} \]

- ignores packet loss
  \[ \rightarrow Q(t + 1) = [Q(t) + \lambda(t) - \gamma(t) - \ell(t)]^+ \]

- apply water sink water level example
Continuous time case:

\[ \frac{dQ(t)}{dt} = \lambda(t) - \gamma(t) \]

\[ \rightarrow \text{ no essential different from discrete time case} \]

\[ \rightarrow \text{ more compact and elegant} \]

Note: if \( \lambda(t) > \gamma(t) \) for all \( t \) then

\[ Q(t) \rightarrow \infty \quad \text{as} \quad t \rightarrow \infty \]

\[ \rightarrow \text{ i.e., water level in sink grows and grows} \]

\[ \rightarrow \text{ of course, water sink has finite “buffer” capacity} \]

\[ \rightarrow \text{ want to keep water level stable; how?} \]
How is $\gamma(t)$ related to $Q(t)$?

- $\gamma(t)$: viewed as “throughput”
- $Q(t)$: viewed as system load

Two cases: monotone and unimodal
Examples:

Monotone load-throughput curve:

- packets departing on router’s output port
  → “raw throughput”
- water draining from sink

Unimodal load-throughput curve:

- ARQ’s receiver throughput experienced by application
  → “reliable throughput”
- CSMA throughput
  → load: no. of users or sending rate
Congestion control for monotone curve:

→ simple case

\[ \lambda(t) \]

\[ Q(t) \]

\[ Q^* \]

**Sender** & **Receiver**

- \( \gamma \): constant (e.g., link bandwidth 100 Mbps)
- \( \lambda(t) \): under our control
- \( Q(t) \): current buffer level
- \( Q^* \): desired buffer level

→ goal: vary \( \lambda(t) \) such that \( Q(t) \approx Q^* \)
Basic idea:

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

$\rightarrow$ control law

$\rightarrow$ thermostat control (same as water faucet)

Protocol implementation:

- Control action undertaken at sender
- Receiver needs to inform sender of $Q^*$ and $Q(t)$
  $\rightarrow$ feedback packet

Key question: how much to increase or decrease $\lambda(t)$?
Applications:

Router congestion control

→ active queue management (AQM)

• Receiver is viewed as router

• $Q^*$ is viewed as desired buffer occupancy and delay

• Router throttles sender(s) to maintain $Q^*$

→ feedback: e.g., binary (ECN bit)

→ ECN: explicit congestion notification

→ two bits in IPv4 TOS field

→ piggybacking trick
Pseudo real-time streaming applications

→ e.g., Internet radio, video streaming

→ e.g., RealPlayer, MediaPlayer

Method:

- Prefetch $X$ seconds worth of data (e.g., 20 seconds)
  → audio/video frames

- Initial delayed playback
  → penalty: pseudo real-time

- Keep fetching audio/video data such that around $X$ seconds of future data resides in receiver’s buffer
  → buffering allows hiding of spurious congestion

  → if properly done, user experiences continuous playback