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

→ congestion control

Need to understand:

• What is congestion?

• How do we prevent or manage it?
Traffic influx/outflux picture:

- 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: $Q(t)$
  $\rightarrow$ 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

→ 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 . . .
802.11b WLAN:

- Throughput

\[\text{MAC System Throughput (Mb/s)}\]

\[\text{Offered Load (Mb/s)}\]

\[\text{unimodal or bell-shaped}\]
802.11b WLAN:

- Collision

\[ \text{underlying cause of unimodal throughput} \]
What we can regulate or control:

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

Ex.:

- Faucet knob in water sink
- Temperature needle in thermostat
- Cars entering onto highway
- Traffic sent by UDP or TCP
How does in-flight traffic or load $Q(t)$ vary?

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
- Missing item
Goal: Want to keep system in “good” state

Ex.: If $\lambda(t) > \gamma(t)$ for all time then

$$Q(t) \to \infty \text{ as } t \to \infty$$

$\Rightarrow$ water level in sink grows and grows

$\Rightarrow$ water sink has finite “buffer” capacity, overflows

$\Rightarrow$ want to keep water level stable; how?

Control actions:

- If water level is too high, close faucet
- If water level is too low, open faucet

$\Rightarrow$ feedback control
Pseudo Real-Time Multimedia Streaming:

→ e.g., RealPlayer, Rhapsody, Internet radio
→ “pseudo” because prefetching trick
→ name of the game: prevent empty buffer

Method:

• Prefetch $X$ seconds worth of data (e.g., 5 seconds)
  → audio/video frames

• Initial delayed playback
  → penalty: pseudo real-time

• Keep fetching audio/video data such that $X$ seconds of future data resides in receiver’s buffer
  → buffering allows hiding of spurious congestion
  → continuous playback experience
Pseudo real-time traffic control:

- $Q(t)$: current buffer level
- $Q^*$: desired buffer level
- $\gamma$: e.g., for video 24 frames-per-second (fps)

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

$\rightarrow$ don’t buffer too much
$\rightarrow$ don’t buffer too little
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 (“control signaling”)
Other 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: ECN (explicit congestion notification)
→ two bits in IPv4 TOS field
→ supported in most routers, not turned on

Also proposed to throttle denial-of-service attack traffic

→ called push-back
→ good guy vs. bad guy problem
Key question in feedback traffic control: how much to increase or decrease $\lambda(t)$?

\[ \rightarrow \text{ “control problem”} \]

\[ \rightarrow \text{ job of traffic protocols (including TCP)} \]

What is the desired state of the system?

\[ \rightarrow \text{ operating point} \]

\[ Q(t) = Q^* \text{ and } \lambda(t) = \gamma \]

Where do we start from?

\[ \rightarrow \text{ empty buffer and no sending rate at start} \]

\[ Q(t) = 0 \text{ and } \lambda(t) = 0 \]
Time evolution (or dynamics):

\[ Q(t) \]

\[ \lambda(t) \]

\[ \gamma \]
Method A:

- if $Q(t) = Q^*$ then $\lambda(t + 1) \leftarrow \lambda(t)$
- if $Q(t) < Q^*$ then $\lambda(t + 1) \leftarrow \lambda(t) + a$
- if $Q(t) > Q^*$ then $\lambda(t + 1) \leftarrow \lambda(t) - a$

where $a > 0$ is a fixed parameter

Question: does it work?

Example:

- $Q^* = 100$
- $\gamma = 10$
- $Q(0) = 0$
- $\lambda(0) = 0$
- $a = 1$
With $a = 0.5$:
With $a = 3$: 
With $a = 6$:
Remarks:

• Method A isn’t that great no matter what a value is used

• Actually: would lead to unbounded oscillation if not for physical restriction $\lambda(t) \geq 0$ and $Q(t) \geq 0$

$\longrightarrow$ easily seen: start from non-zero buffer
With $a = 1$, $Q(0) = 110$, $\lambda(0) = 11$: 
Method B:

- if $Q(t) = Q^*$ then $\lambda(t + 1) \leftarrow \lambda(t)$
- if $Q(t) < Q^*$ then $\lambda(t + 1) \leftarrow \lambda(t) + a$
- if $Q(t) > Q^*$ then $\lambda(t + 1) \leftarrow \delta \cdot \lambda(t)$

where $a > 0$ and $0 < \delta < 1$ are fixed parameters

$\rightarrow$ linear increase with slope $a$

$\rightarrow$ exponential decrease with backoff factor $\delta$

$\rightarrow$ e.g., binary backoff in case $\delta = 1/2$
With $a = 1, \delta = 1/2$: 

![Graph of Load Evolution and Target](image1)

![Graph of Lambda Evolution and Gamma](image2)
With $a = 3$, $\delta = 1/2$: 
With $a = 1$, $\delta = 1/4$:
With $a = 1$, $\delta = 3/4$: 

![Graph of Load Evolution and Target](image)

![Graph of Lambda Evolution and Gamma](image)
Note:

• Method B isn’t that great either

• One advantage over Method A: doesn’t lead to unbounded oscillation
  → due to asymmetry in increase vs. decrease policy
  → “sawtooth” pattern

• Method B is used by TCP
  → linear increase/exponential decrease
  → additive increase/multiplicative decrease (AIMD)

Question: Can we do better?
Method C:

$$\lambda(t + 1) \leftarrow \lambda(t) + \varepsilon(Q^* - Q(t))$$

where $\varepsilon > 0$ is a fixed parameter

Tries to adjust magnitude of change as a function of gap $Q^* - Q(t)$

- if $Q^* - Q(t) > 0$, increase $\lambda(t)$ proportional to gap
- if $Q^* - Q(t) < 0$, decrease $\lambda(t)$ proportional to gap

Trying to be more clever…

$$\rightarrow$$ bottom line: is it any good?
With $\varepsilon = 0.1$: 

![Graph of Load Evolution and Target](image1.png)

![Graph of Lambda Evolution and Gamma](image2.png)
With $\varepsilon = 0.5$: 

\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{load_evolution.png}
\caption{Load Evolution}
\end{figure}

\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{lambda_evolution.png}
\caption{Lambda Evolution}
\end{figure}
Answer: No.

Time to try something strange.

Method D:

$$\lambda(t + 1) \leftarrow \lambda(t) + \varepsilon(Q^* - Q(t)) - \beta(\lambda(t) - \gamma)$$

where $\varepsilon > 0$ and $\beta > 0$ are fixed parameters

$\longrightarrow$ kind of like Method C

$\longrightarrow$ what’s going on?

Sanity check: at desired operating point $Q(t) = Q^*$ and $\lambda(t) = \gamma$, what will happen under Method D?
With $\varepsilon = 0.2$ and $\beta = 0.5$: 
With $\varepsilon = 0.5$ and $\beta = 1.1$: 

![Diagram 1](#)

![Diagram 2](#)
With $\varepsilon = 0.1$ and $\beta = 1.0$: 

![Diagram of Load Evolution and Lambda Evolution](image-url)
Remarks:

- Method D has desired behavior
- Is superior to Methods A, B, and C
- No unbounded oscillation
- In fact, dampening and convergence to desired operating point
  → asymptotic stability
  → oscillations die down
Why does it work?

What is the role of the $-\beta(\lambda(t) - \gamma)$ term in the control law

$$\lambda(t + 1) \leftarrow \lambda(t) + \varepsilon(Q^* - Q(t)) - \beta(\lambda(t) - \gamma)$$

Need to look beneath the hood . . .
Visualize action in 2-D \((Q(t), \lambda(t))\)-space:

\[ \rightarrow \text{ phase space} \]
Convergent trajectory:

\[ \rightarrow \text{ asymptotically stable \& optimal} \]
Divergent trajectory:

\[ \rightarrow \text{ unstable} \]
Stable (but not asymptotically so) trajectory:

\[ \lim_{t \to \infty} \lambda(t) = \lambda \]

\[ Q(t) \]
Which case arises depends on the specifics of protocol actions.

For example:

- Methods A and C: divergent
- Method B: stable (but not asymptotically)
  → TCP
- Method D: asymptotically stable & optimal
  → “optimal control”

Why does Method D work?!
  → ’cause it does

For longer explanation, come to graduate school...
  → not just any graduate school