What is the goal?

$\longrightarrow$ achieve $Q(t) = Q^*$

$\longrightarrow$ or close to it: $|Q(t) - Q^*| < \varepsilon$

How to: basic idea

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

  $\rightarrow$ too little in the buffer
- if $Q(t) > Q^*$ decrease $\lambda(t)$

  $\rightarrow$ too much in the buffer

$\longrightarrow$ a rule of thumb: called control law

Since state of receiver buffer must be conveyed to sender who adjusts $\lambda(t)$:

$\longrightarrow$ called feedback control

$\longrightarrow$ also closed-loop control

Network protocol implementation:

$\longrightarrow$ some design options available

- control action undertaken at sender

  $\rightarrow$ smart sender/dump receiver

  $\rightarrow$ preferred mode of many Internet protocols

  $\rightarrow$ when might the opposite be better?

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

  $\rightarrow$ feedback packet (control signaling/messaging)

  $\rightarrow$ feedback could just be gap $Q^* - Q(t)$

  $\rightarrow$ or simply up/down binary indication

Key question in feedback congestion control:

$\longrightarrow$ **how much** to increase/decrease $\lambda(t)$

$\longrightarrow$ already know in which direction

Desired state of the system:

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

$\longrightarrow$ why is "$\lambda(t) = \gamma$" needed?

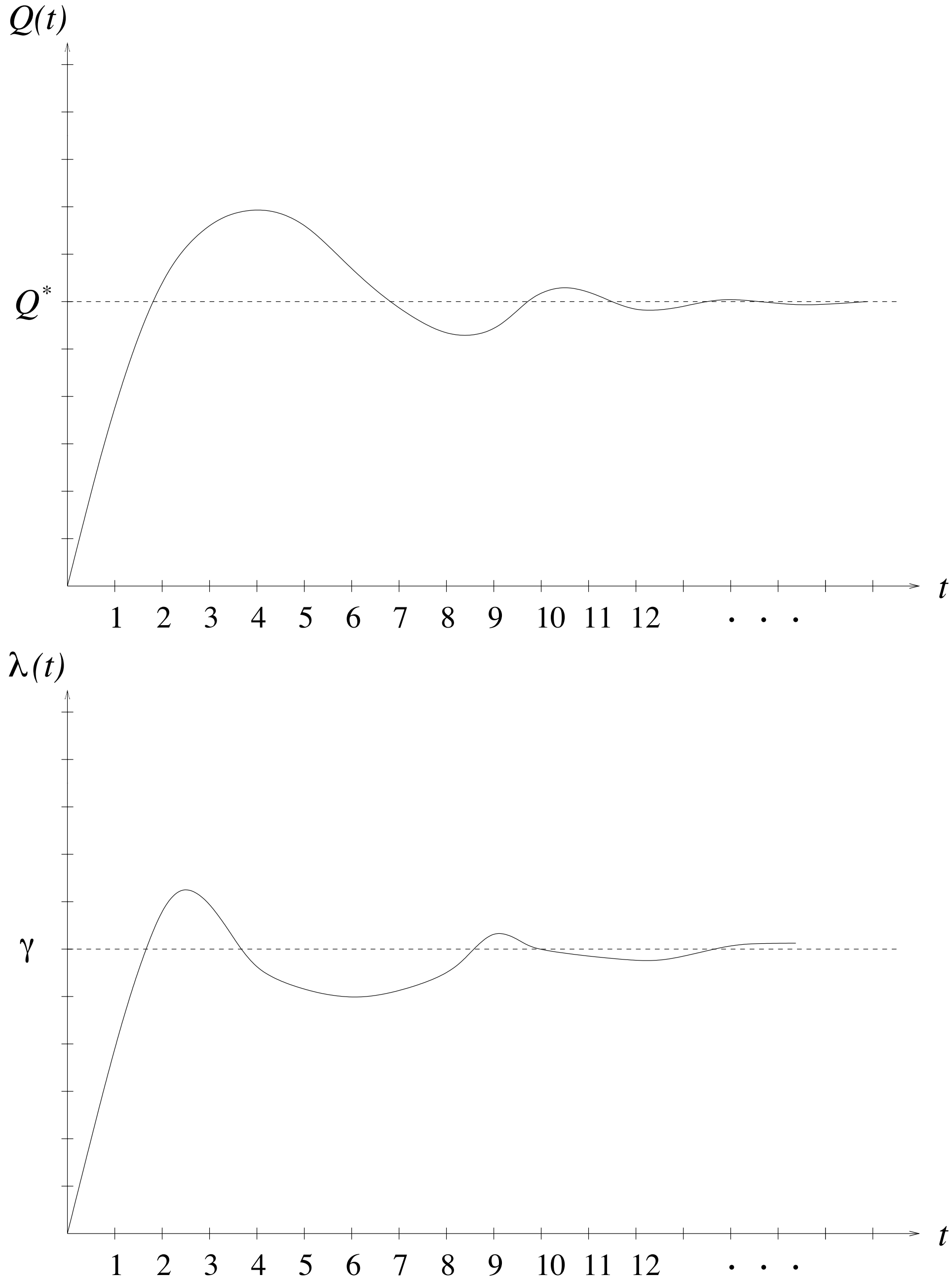
Starting state:

$\longrightarrow$ empty buffer and nothing is being sent

$\longrightarrow$ think of iTunes, Rhapsody, etc.

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

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



Congestion control methods: A, B, C and D

`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 $\lambda(t+1) \leftarrow \lambda(t) - a$

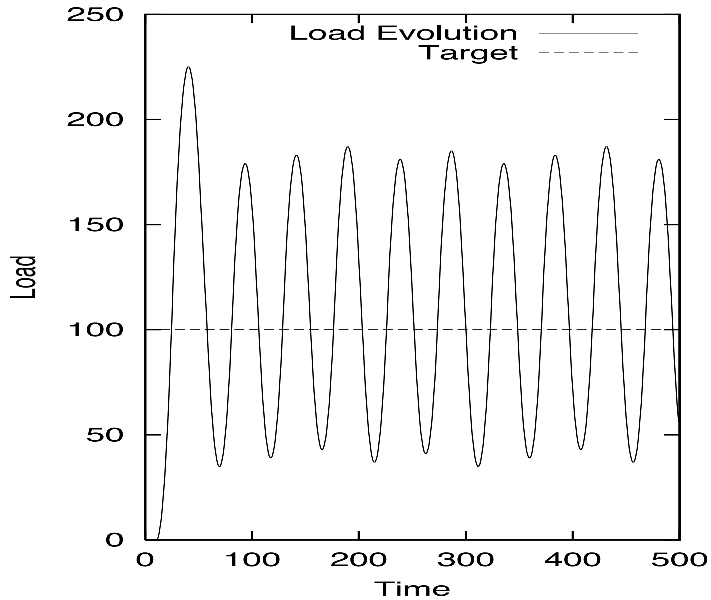
where $a > 0$ is a fixed parameter

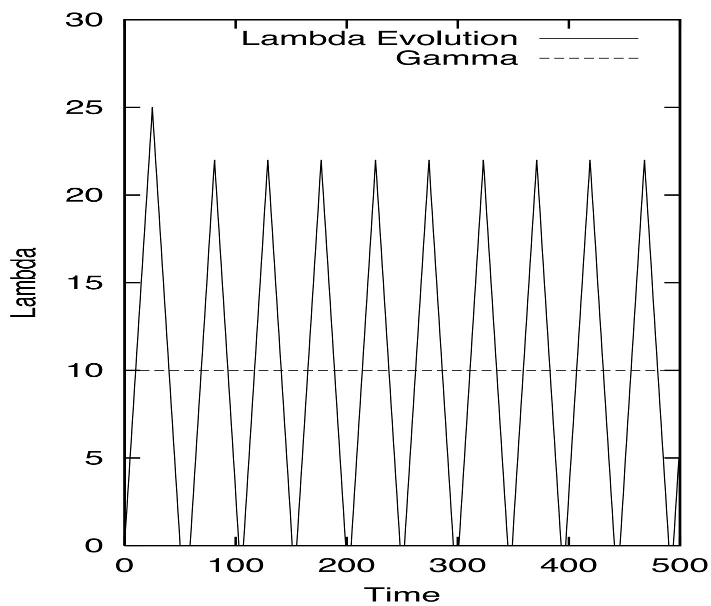
$\longrightarrow$ called linear increase/linear decrease

Question: how well does it work?

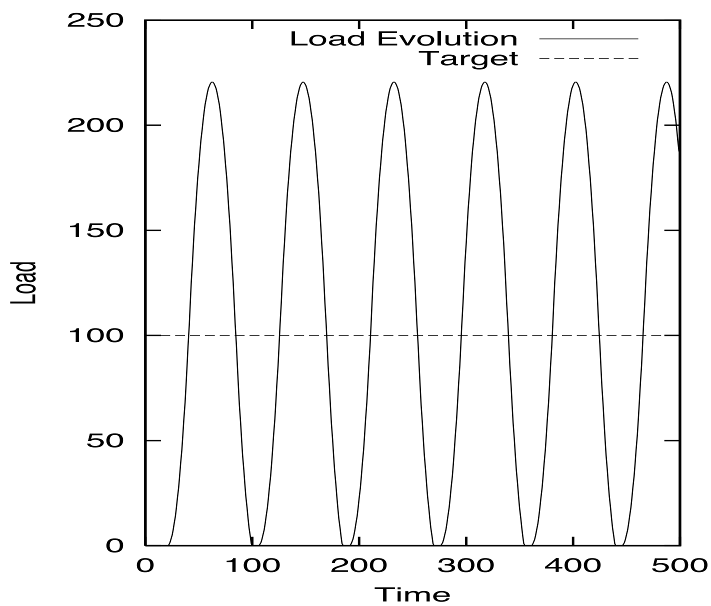
Example:

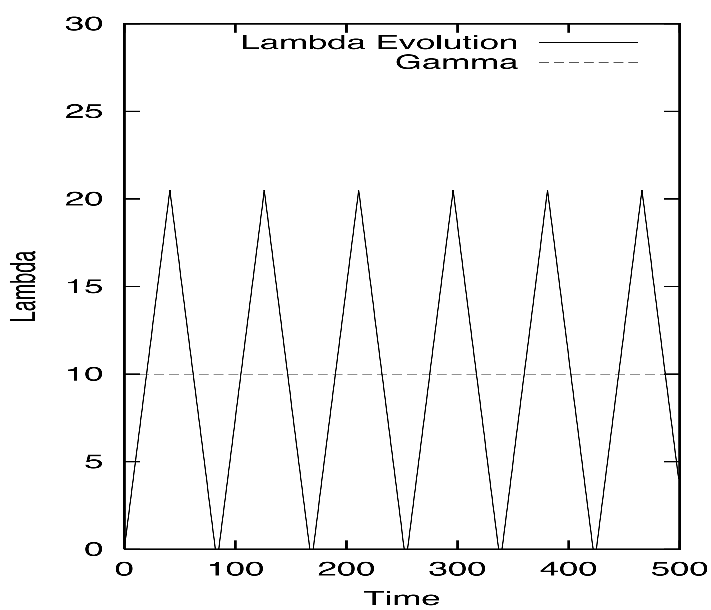
- $Q(0) = 0$
- $\lambda(0) = 0$
- $Q^* = 100$
- $\gamma = 10$
- $a = 1$


Load Evolution
Target
Load
Time
0
50
100
150
200
250
0
100
200
300
400
500

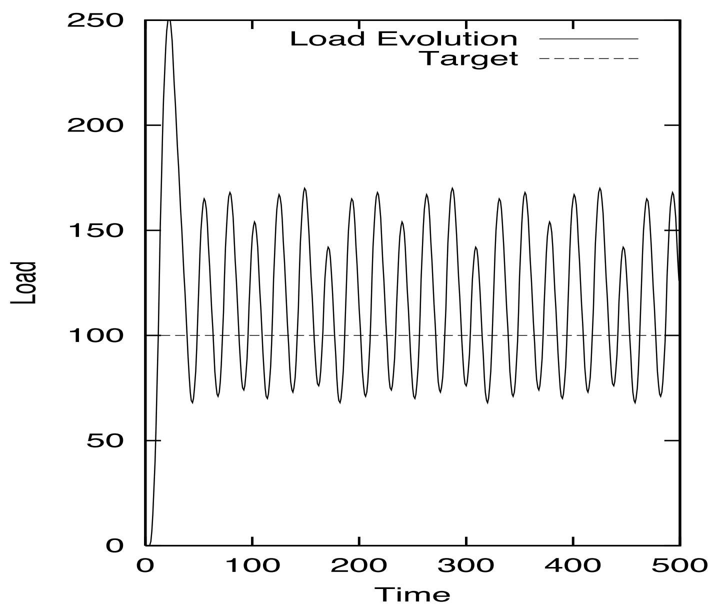

Lambda Evolution
Gamma
Lambda
Time
0
5
10
15
20
25
30
0
100
200
300
400
500

With $a = 0.5$:

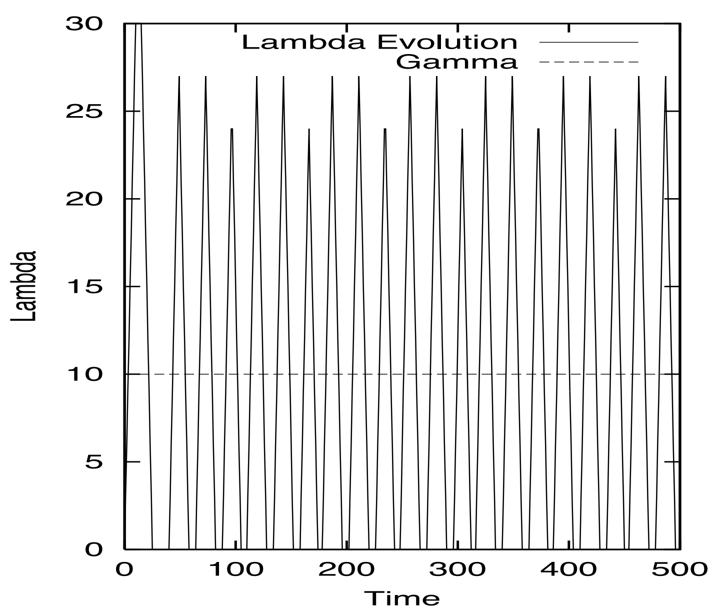

Load Evolution
Target
250
200
150
100
50
0
Load
0
100
200
300
400
500
Time


Lambda Evolution
Gamma
30
25
20
15
10
5
0
Lambda
0
100
200
300
400
500
Time

With $a = 3$:

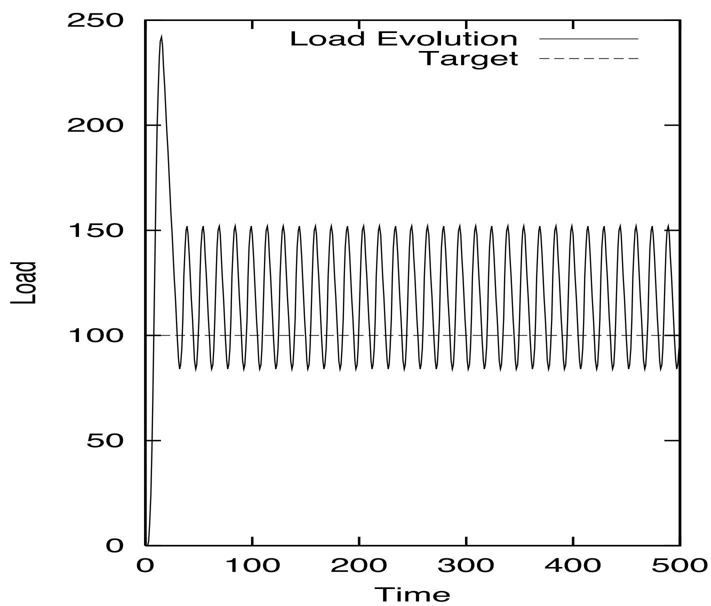


Load Evolution
Target
Load
Time
250
200
150
100
50
0
0
100
200
300
400
500

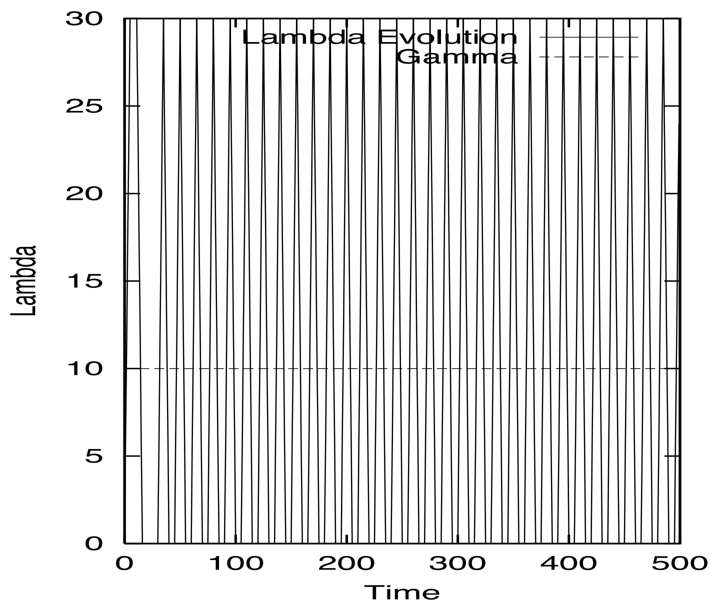


Lambda Evolution
Gamma
Lambda
Time
30
25
20
15
10
5
0
0
100
200
300
400
500

With $a = 6$:



Load Evolution
Target
Load
Time



Lambda Evolution
Gamma
Lambda
Time

Remarks:

- Method A isn't that great no matter what a value is used

  $\rightarrow$ keeps oscillating

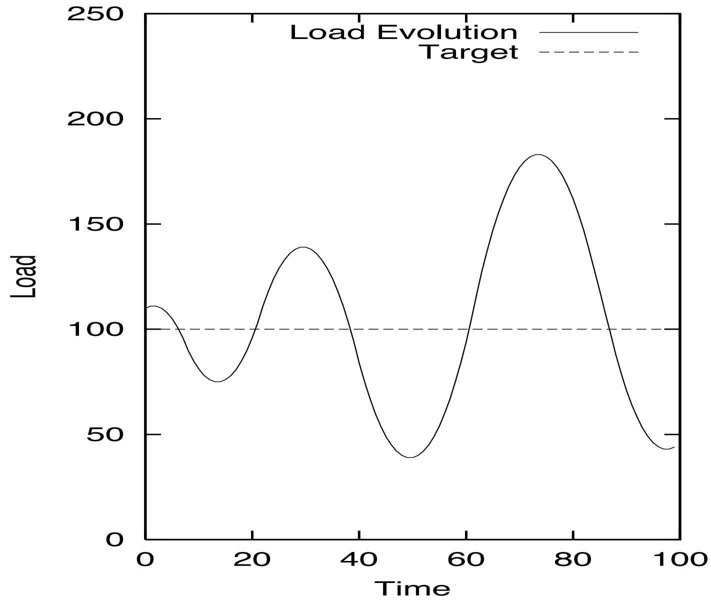
- Actually: would lead to unbounded oscillation if not for physical restriction $\lambda(t) \geq 0$ and $Q(t) \geq 0$

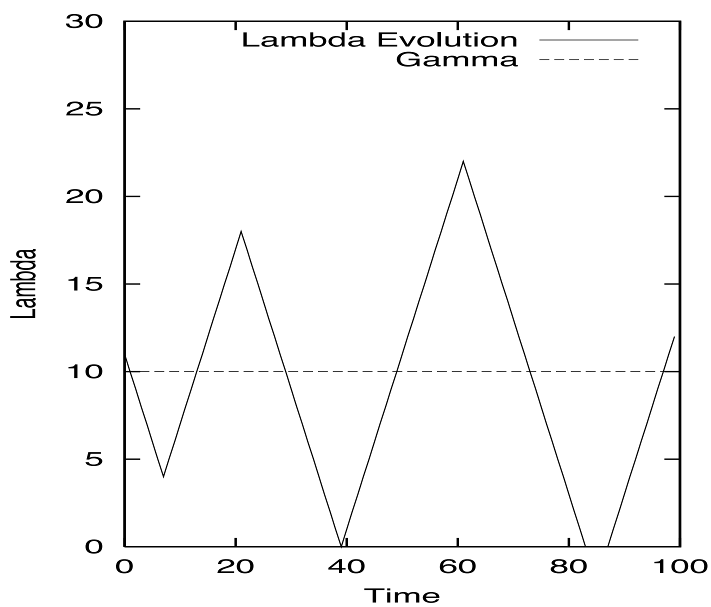
  $\longrightarrow$ i.e., bottoms out

  $\longrightarrow$ easily seen: start from non-zero buffer

  $\longrightarrow$ e.g., $Q(0) = 110$

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

Note: only decrease part differs from `Method A`.

$\longrightarrow$ linear increase with slope a

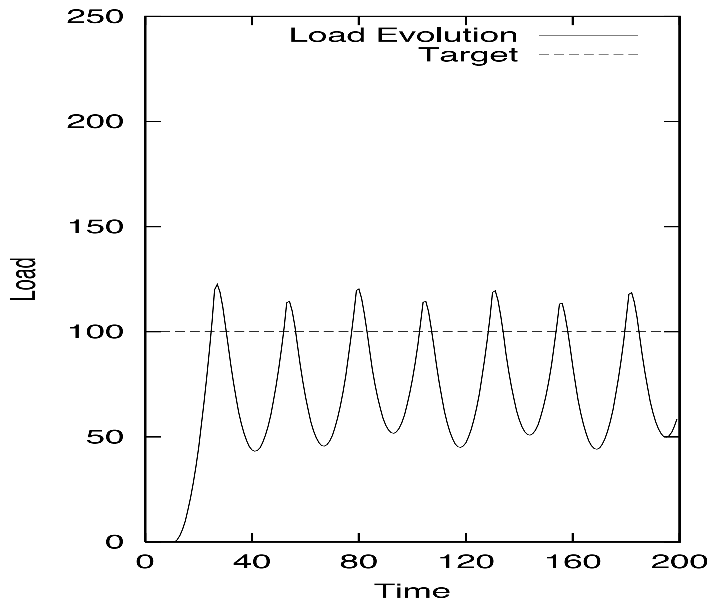
$\longrightarrow$ exponential decrease with backoff factor δ

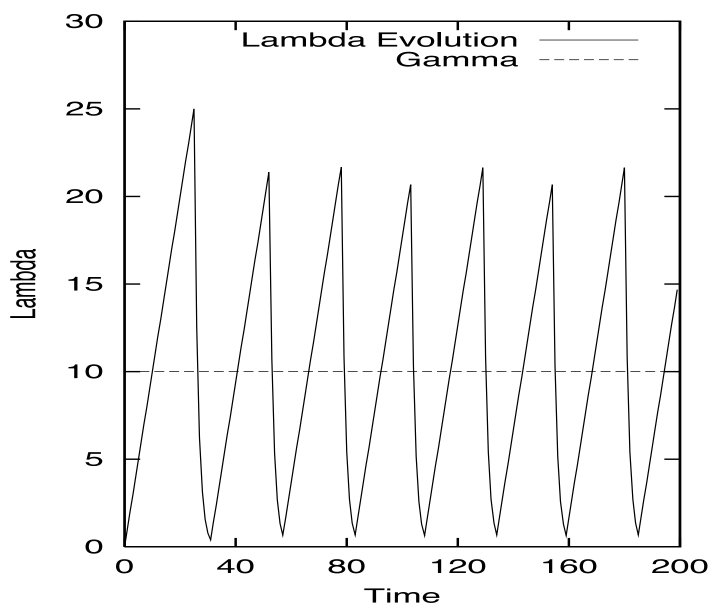
$\longrightarrow$ e.g., binary backoff in case $\delta = 1/2$

Similar to Ethernet and WLAN backoff

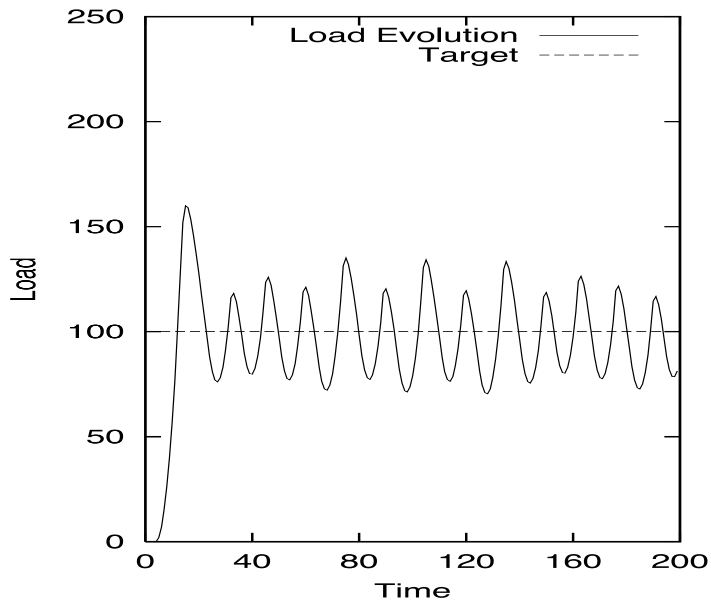
$\longrightarrow$ question: does it work?

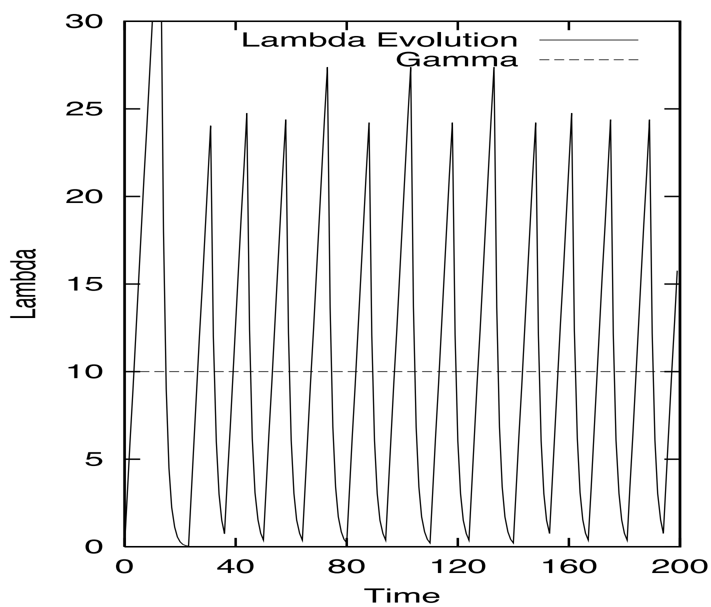
With $a = 1$, $\delta = 1/2$:



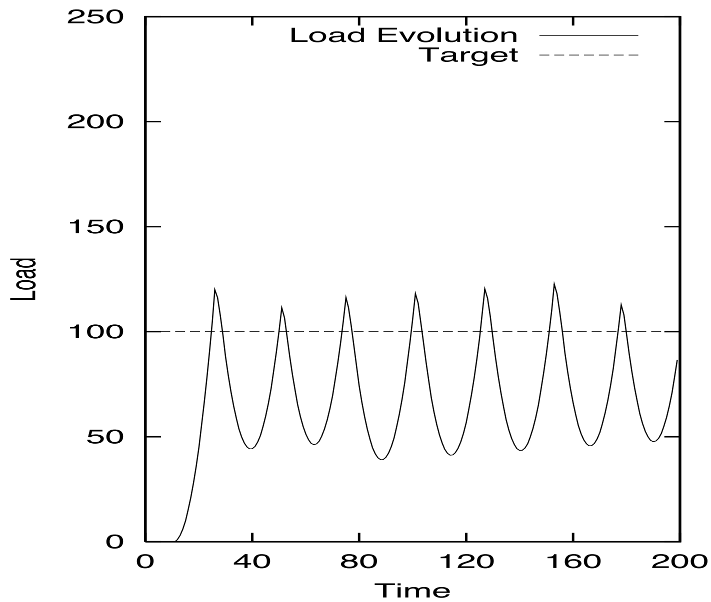


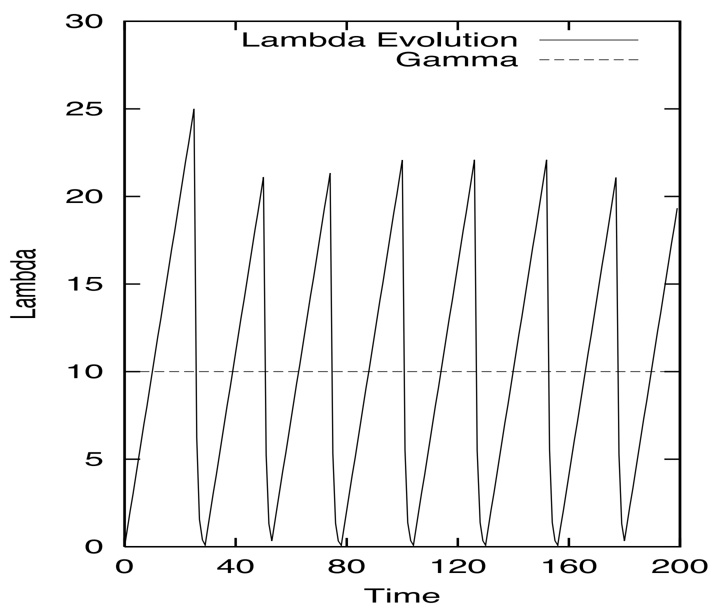
With $a = 3$, $\delta = 1/2$:



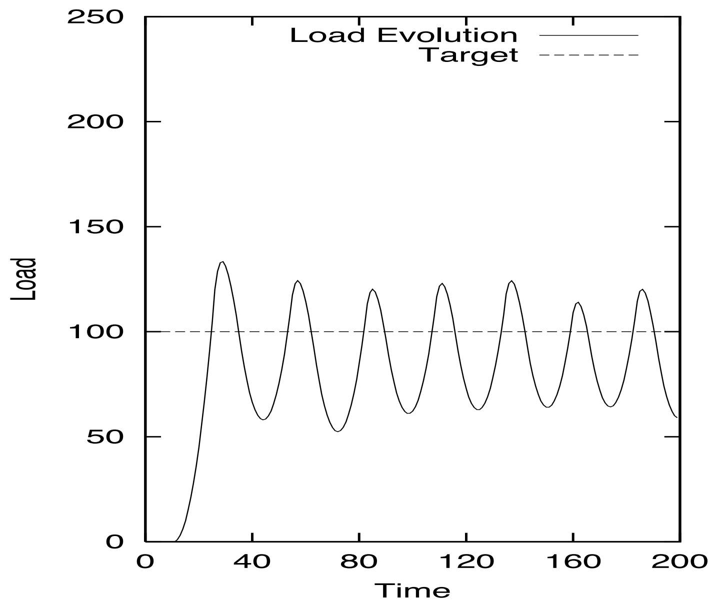


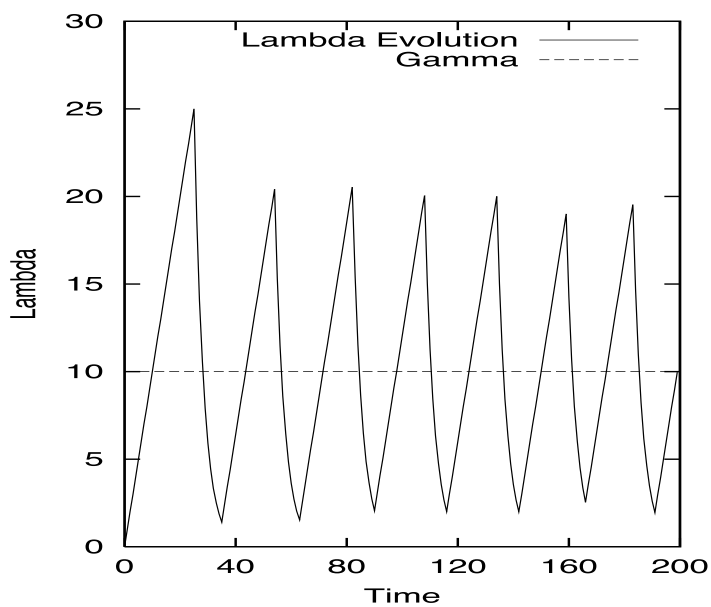
With $a = 1$, $\delta = 1/4$:





With $a = 1$, $\delta = 3/4$:





Note:

- Method B isn't that great either
- One advantage over Method A: doesn't lead to unbounded oscillation

  $\rightarrow$ note: doesn't hit "rock bottom"

  $\rightarrow$ due to asymmetry in increase vs. decrease policy

  $\rightarrow$ we observe "sawtooth" pattern
- Method B is used by TCP

  $\rightarrow$ linear increase/exponential decrease

  $\rightarrow$ additive increase/multiplicative decrease (AIMD)

Question: can we do better?

$\longrightarrow$ what "freebie" have we not made use of?

`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 the gap $Q^* - Q(t)$

$\longrightarrow$ incorporate distance from target Q^*

$\longrightarrow$ before: just the sign (above/below)

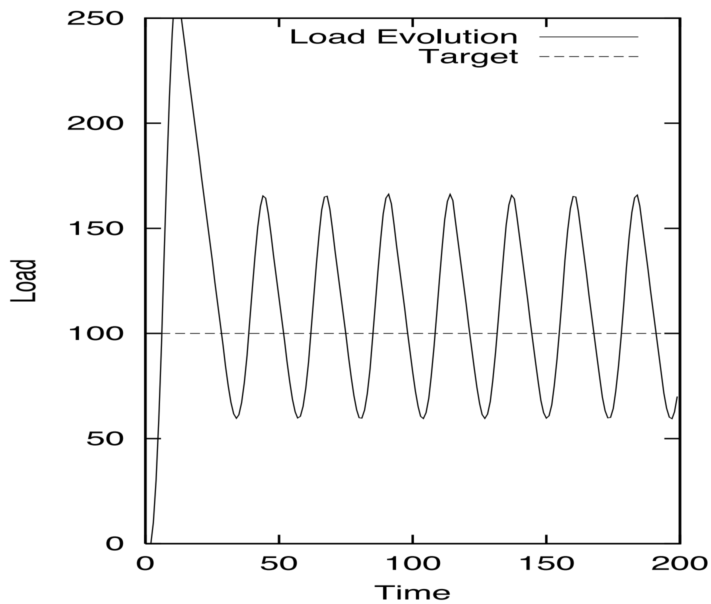
Thus:

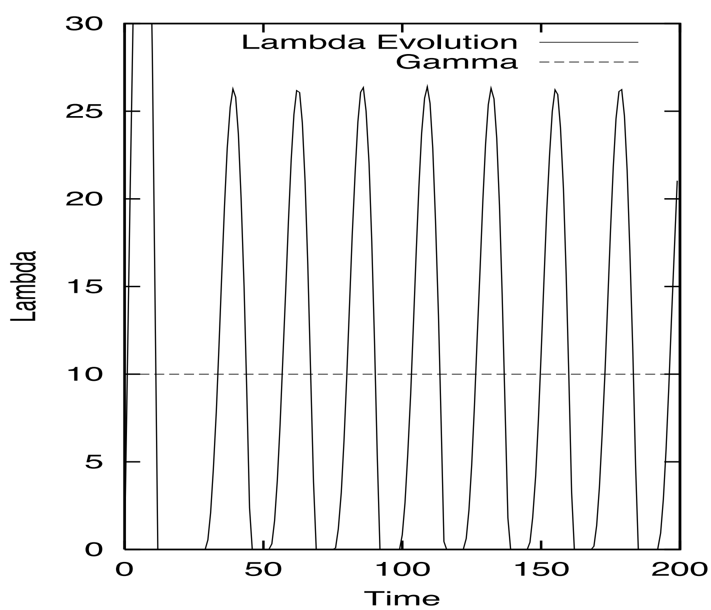
- if $Q^* - Q(t) > 0$, increase $\lambda(t)$ proportional to gap
- if $Q^* - Q(t) < 0$, decrease $\lambda(t)$ proportional to gap

Trying to be more clever...

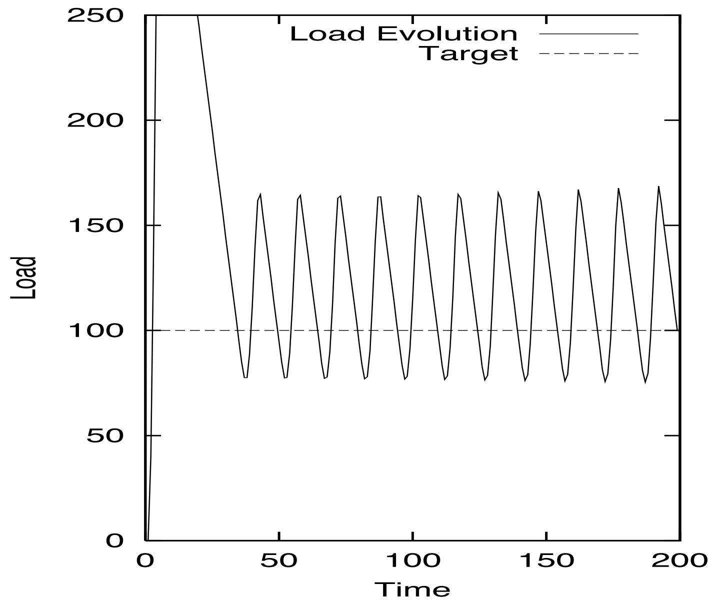
$\longrightarrow$ bottom line: is it any good?

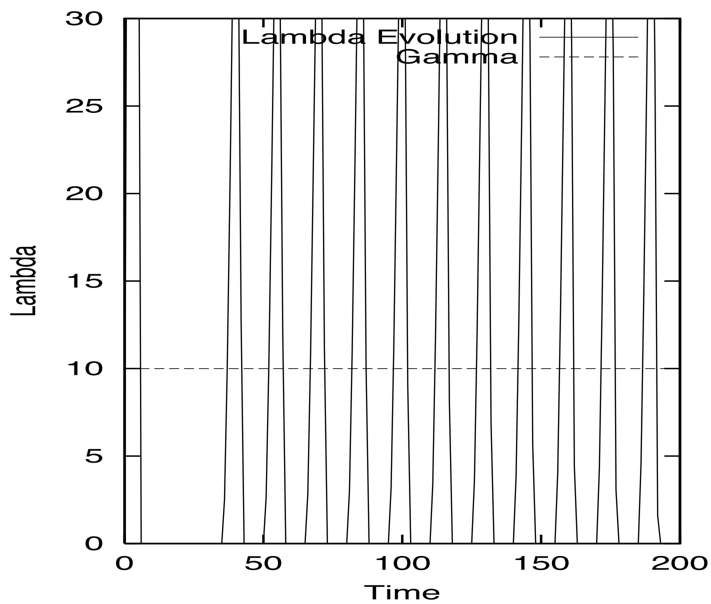
With $\varepsilon = 0.1$:





With $\varepsilon = 0.5$:





Answer: no

$\longrightarrow$ control law looks good on the surface

$\longrightarrow$ but looks can be deceiving

Time to try something strange

$\longrightarrow$ any (crazy) ideas?

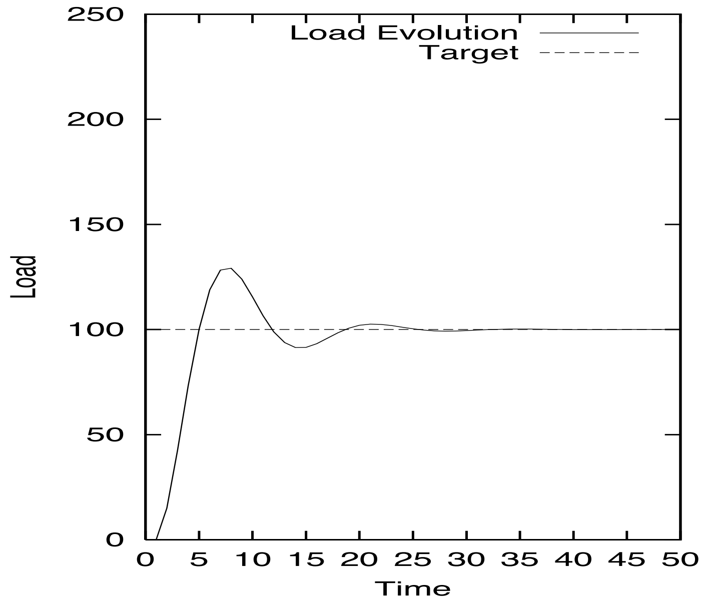
`Method D`:

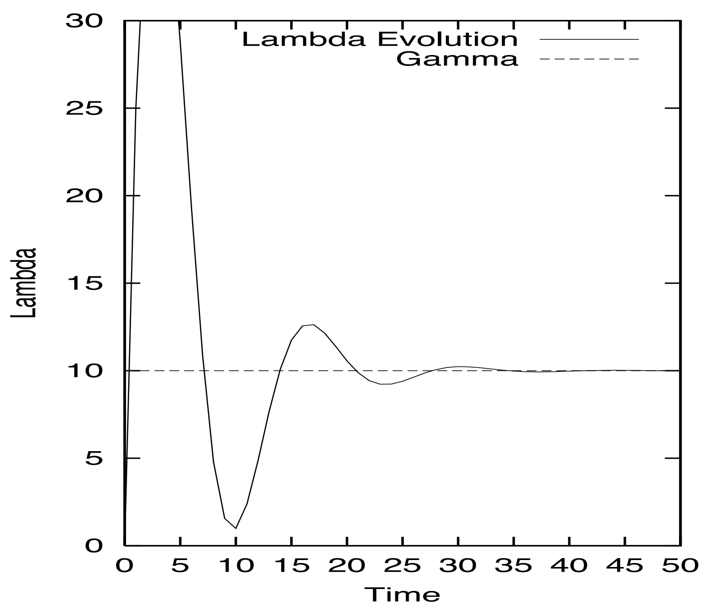
$$\lambda(t+1) \leftarrow \lambda(t) + \varepsilon(Q^* - Q(t)) - \beta(\lambda(t) - \gamma)$$

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

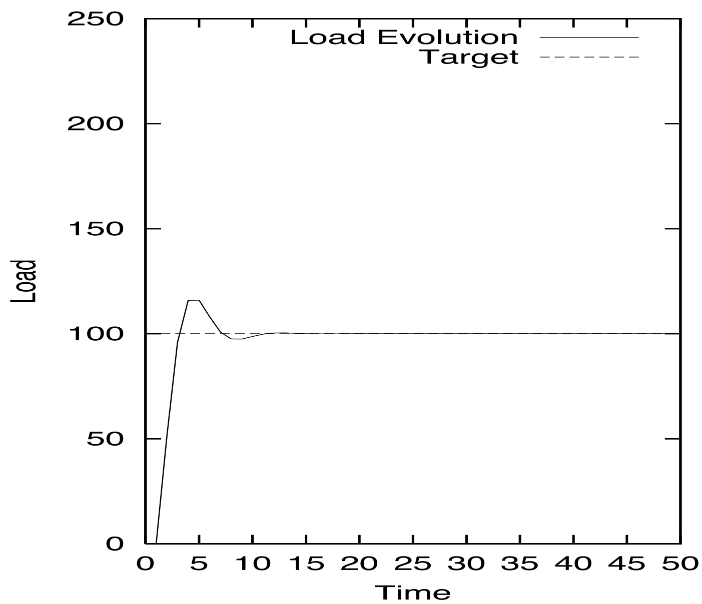
- $\longrightarrow$ odd looking modification to `Method C`
- $\longrightarrow$ additional term $-\beta(\lambda(t) - \gamma)$
- $\longrightarrow$ what's going on?
- $\longrightarrow$ does it work?

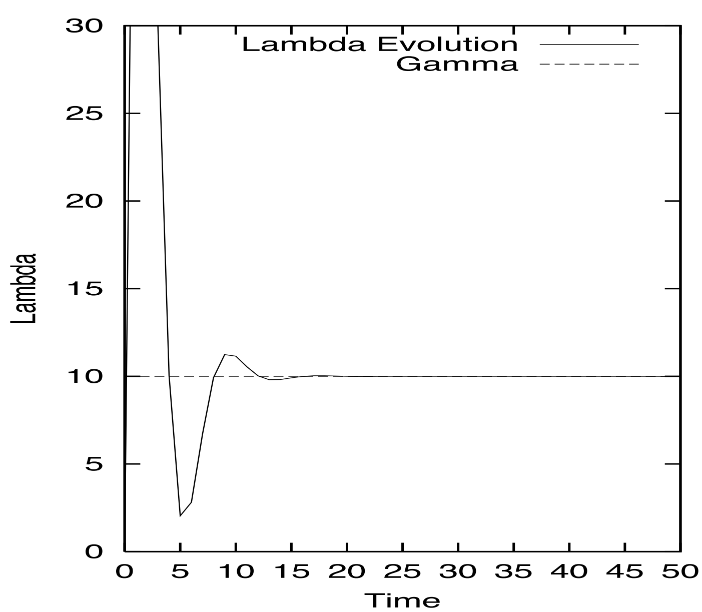
With $\varepsilon = 0.2$ and $\beta = 0.5$:





With $\varepsilon = 0.5$ and $\beta = 1.1$:





With $\varepsilon = 0.1$ and $\beta = 1.0$: