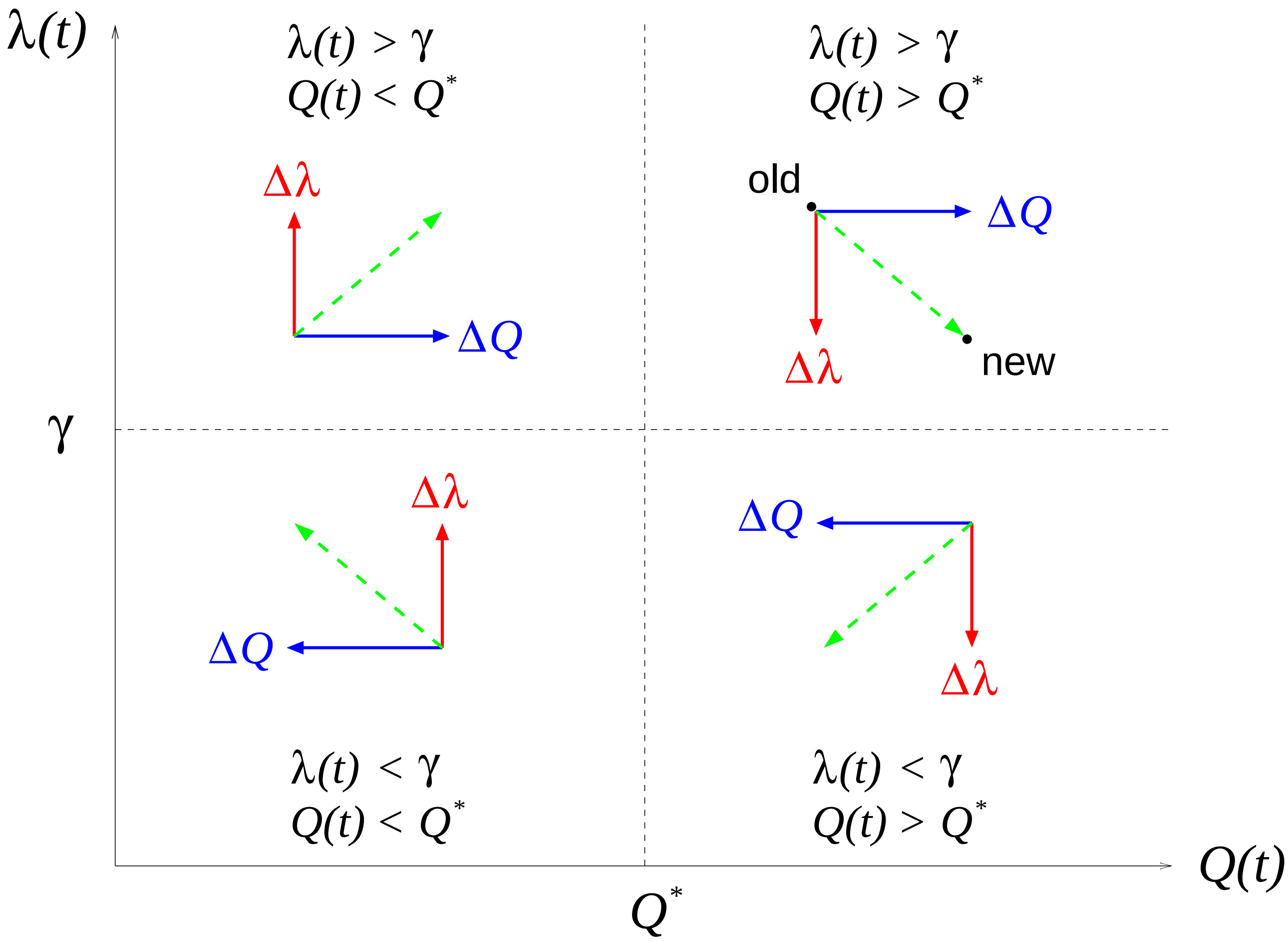
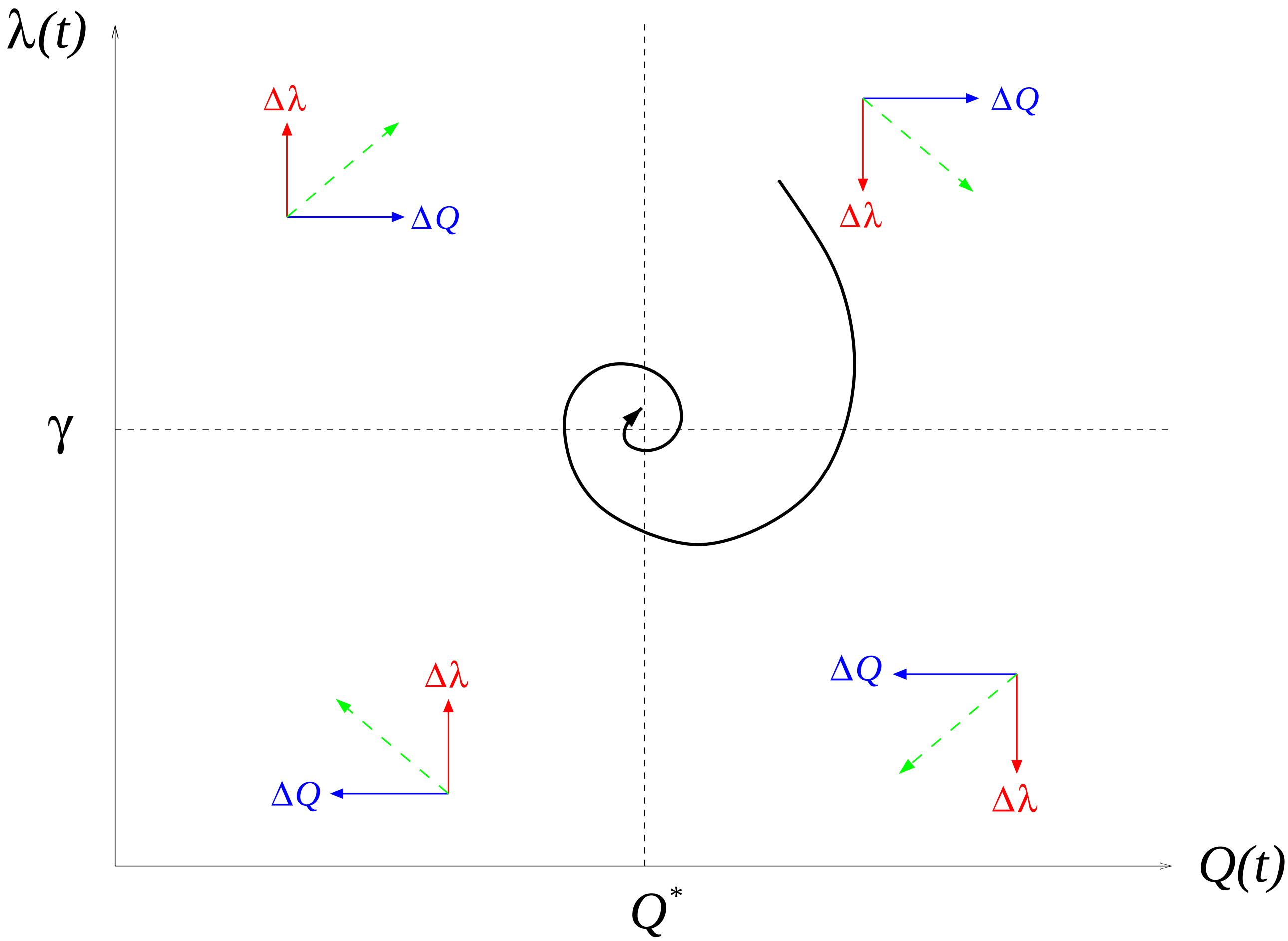
Visualize action in 2-D $(Q(t), \lambda(t))$-space:

$\longrightarrow$ phase space



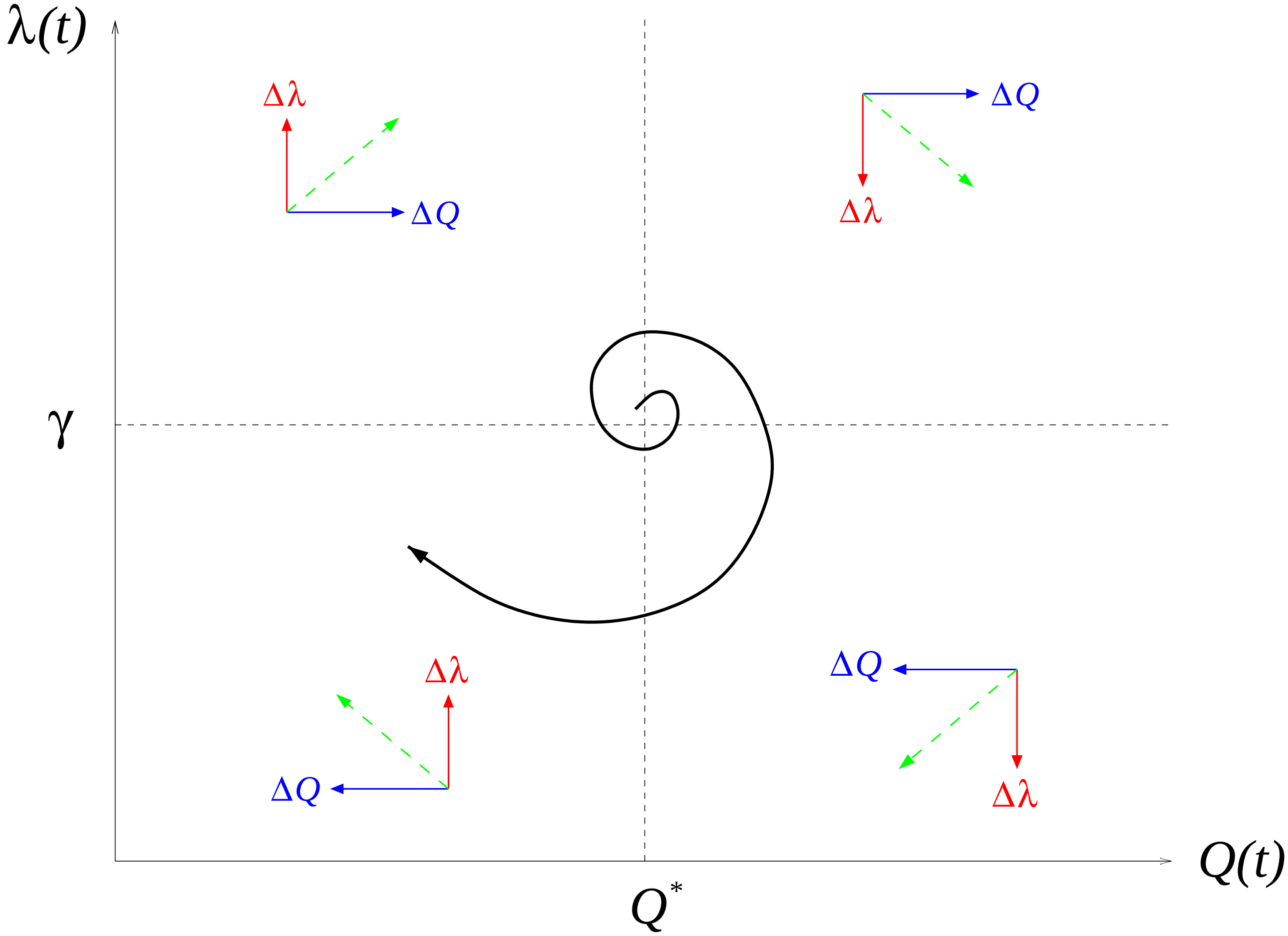
Convergent trajectory:

$\longrightarrow$ asymptotically stable & optimal



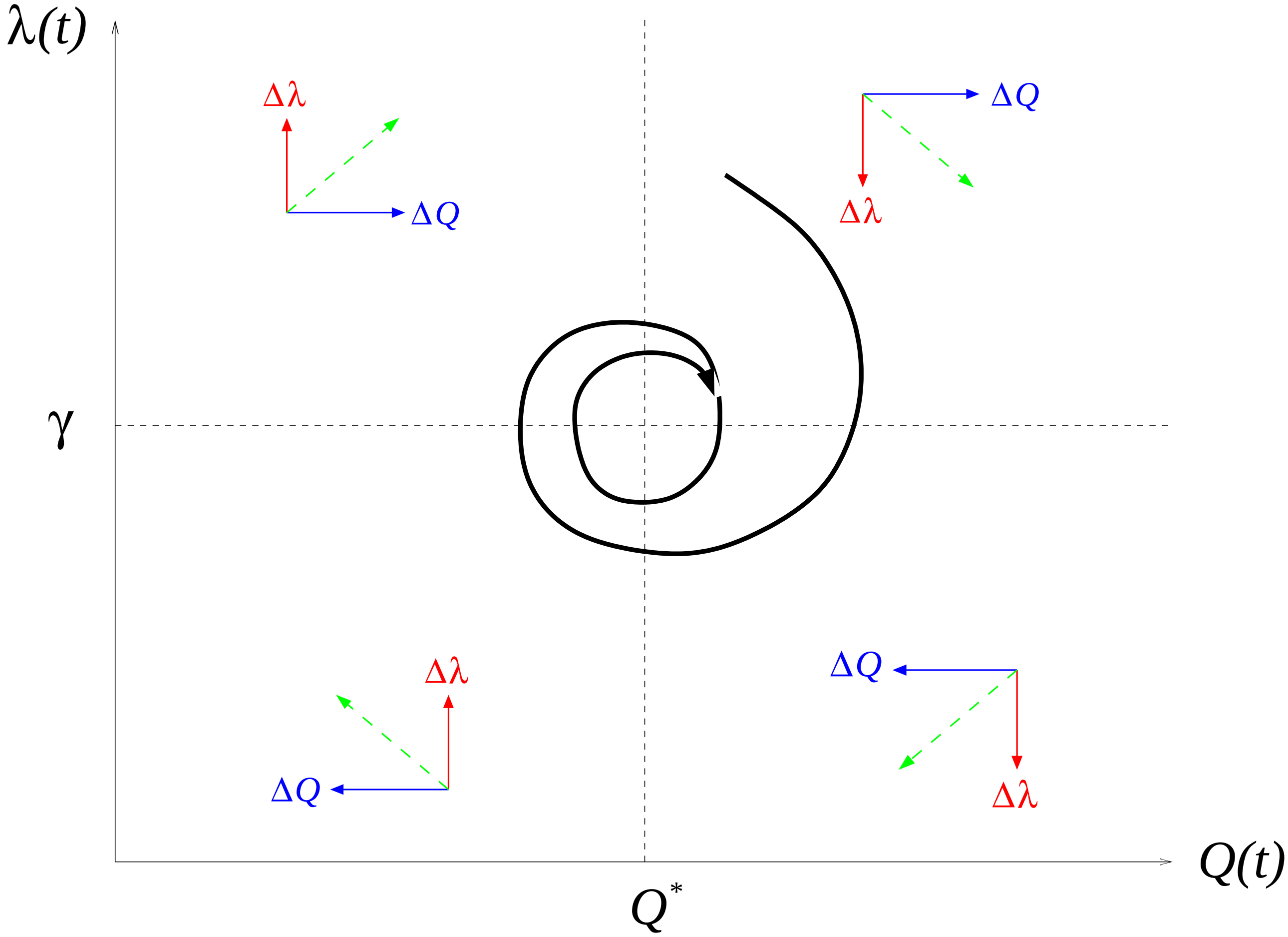
Divergent trajectory:

$\longrightarrow$ unstable



Stable (but not asymptotically so) trajectory:

$\longrightarrow$ limit cycle



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:

$\longrightarrow$ overview of underlying mathematics

First, represent in continuous form:

$\longrightarrow$ easier to manipulate

$\longrightarrow$ more elegant

$$\begin{aligned} \frac{dQ(t)}{dt} &= \lambda(t) - \gamma \\ \frac{d\lambda(t)}{dt} &= \varepsilon(Q^* - Q(t)) - \beta(\lambda(t) - \gamma) \end{aligned}$$

In vector form:

$$\frac{d}{dt}\begin{pmatrix} Q(t) \\ \lambda(t) \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ -\varepsilon & -\beta \end{pmatrix} \begin{pmatrix} Q(t) \\ \lambda(t) \end{pmatrix} + \begin{pmatrix} -\gamma \\ \varepsilon Q^* + \beta\gamma \end{pmatrix}$$

View as

$$\frac{d\boldsymbol{x}(t)}{dt} = A\,\boldsymbol{x}(t) + \boldsymbol{b}$$

where $\boldsymbol{x}(t) = (Q(t), \lambda(t))$.

Note: at desired/optimal operating point (Q^*, γ),

$$\frac{dQ(t)}{dt} = 0 \quad \text{and} \quad \frac{d\lambda(t)}{dt} = 0$$

$\longrightarrow$ (Q^*, γ) is a rest point or equilibrium

$\longrightarrow$ "when in heaven/nirvana, stay put"

Rest point (or fixed-point) condition is necessary but not sufficient:

$\longrightarrow$ external factors & noise makes $\boldsymbol{x}$ stray

$\longrightarrow$ can it return to (Q^*, γ)

$\longrightarrow$ question of stability

Ex.: ball on a hill vs. ball in a valley

$\longrightarrow$ both are fixed-points

$\longrightarrow$ only one is stable w.r.t. perturbation

How do we determine stability of

$$d\boldsymbol{x}(t)/dt = A\,\boldsymbol{x}(t) + \boldsymbol{b}\ ?$$

Idea: consider $dz/dt = az$

$\longrightarrow$ a: positive or negative constant

$\longrightarrow$ what's the rest point?

$\longrightarrow$ what happens to z over time?

$\longrightarrow$ a: eigenvalue of the (1-D) system

Same idea applies to multi-dimensional linear systems

$\longrightarrow$ in n dimensions: n eigenvalues

$\longrightarrow$ asymptotic stability: all are negative

$\longrightarrow$ stability: some may be 0

$\longrightarrow$ unstable: one or more positive eigenvalues

What are eigenvalues, eigenvectors...

$\longrightarrow$ given matrix A: $A\boldsymbol{u} = a\boldsymbol{u}$

$\longrightarrow$ $\boldsymbol{u}$: eigenvector of A

$\longrightarrow$ a: $\boldsymbol{u}$'s eigenvalue

$\longrightarrow$ "operator" A stretches or pulls

Thus: for our 2-D congestion control system, stability depends on the eigenvalues of

$$A = \begin{pmatrix} 0 & 1 \\ -\varepsilon & -\beta \end{pmatrix}$$

Congestion control system is asymptotically stable if the real parts of the eigenvalues are strictly negative

$\longrightarrow$ eigenvalues can be complex

What remains... let's calculate and check!

Eigenvalues are obtained from characteristic equation

$$\begin{aligned}\det\begin{pmatrix}-\nu & 1 \\ -\varepsilon & -\beta-\nu\end{pmatrix} &= \nu(\beta+\nu)+\varepsilon \\ &= \nu^2+\beta\nu+\varepsilon = 0\end{aligned}$$

This yields:

$$\nu = \frac{-\beta}{2} \pm \frac{\sqrt{\beta^2-4\varepsilon}}{2}$$

If $\varepsilon > 0$, then $Re(\nu) < 0$. If, in addition, $0 < \varepsilon < \beta/2$ then the eigenvalues are real.

Check: without the $-\beta(\lambda(t)-\gamma)$ term in the control law, the characteristic equation becomes

$$\nu^2 + \varepsilon = 0$$

Thus, $\nu = \pm\sqrt{\varepsilon}$ and the real part of ν is zero, violating the asymptotic stability condition.

$\longrightarrow$ mathematical requirement

$\longrightarrow$ intuitively: damping effect

What about when throughput γ experiences congestion: under excessive load goes down

$\longrightarrow$ note: we assumed constant γ in analysis

TCP congestion control

Recall:

$$\texttt{EffectiveWindow} = \texttt{MaxWindow} - (\texttt{LastByteSent} - \texttt{LastByteAcked})$$

where

$$\texttt{MaxWindow} = \min\{\,\texttt{AdvertisedWindow}, \texttt{CongestionWindow}\,\}$$

Key question: how to set `CongestionWindow` which, in turn, affects ARQ's sending rate?

$\longrightarrow$ linear increase/exponential decrease

$\longrightarrow$ AIMD

TCP congestion control components:

(i) Congestion avoidance

$\longrightarrow$ linear increase/exponential decrease

$\longrightarrow$ additive increase/exponential decrease (AIMD)

As in Method B, increase `CongestionWindow` linearly, but decrease exponentially

Upon receiving ACK:

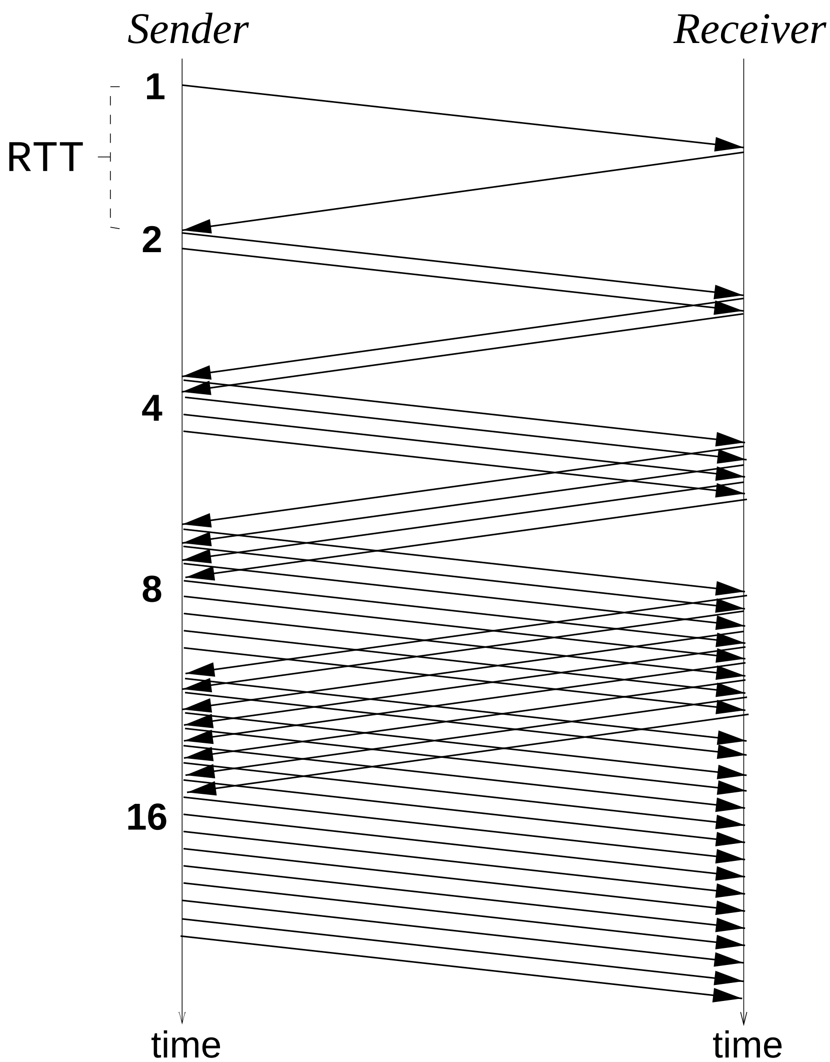
$$\texttt{CongestionWindow} \leftarrow \texttt{CongestionWindow} + 1$$

Upon timeout:

$$\texttt{CongestionWindow} \leftarrow \texttt{CongestionWindow}\,/\,2$$

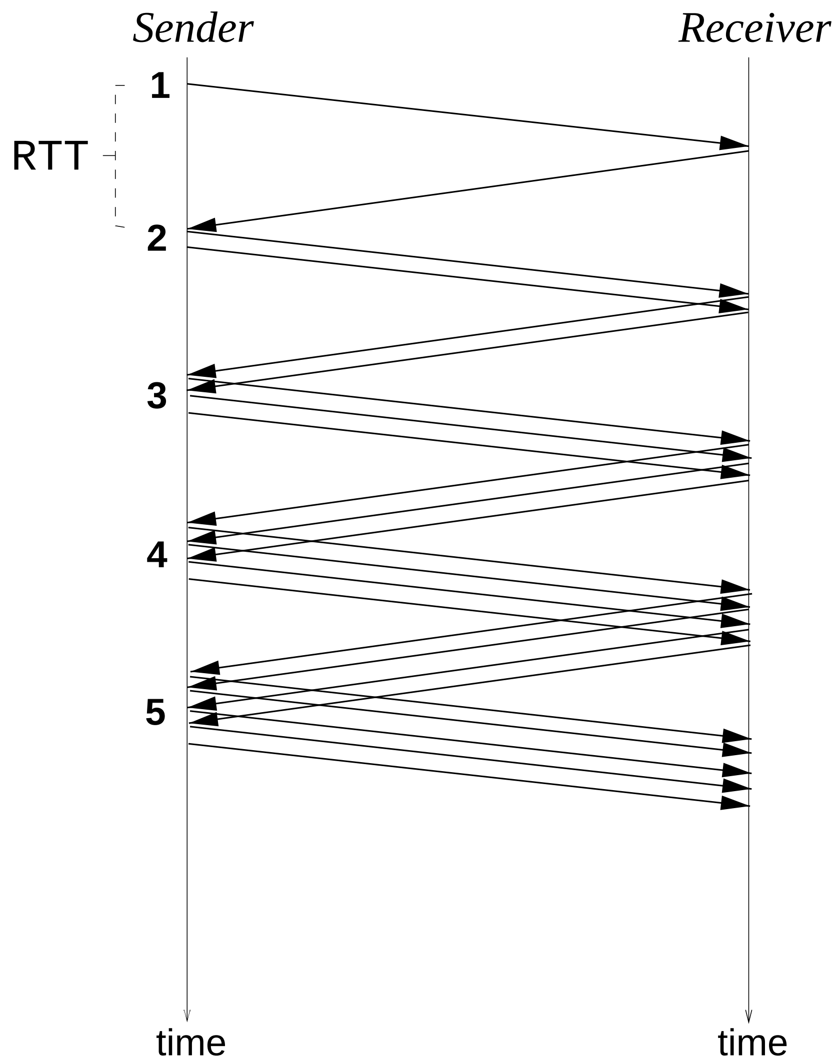
But is it correct...

"Linear increase" time diagram:



$\longrightarrow$ results in exponential increase

What we want:



$\longrightarrow$ increase by 1 every window

Thus, linear increase update:

$$\texttt{CongestionWindow} \leftarrow \texttt{CongestionWindow} + (1\,/\,\texttt{CongestionWindow})$$

Upon timeout and exponential backoff,

$$\texttt{SlowStartThreshold} \leftarrow \texttt{CongestionWindow}\,/\,2$$

(ii) Slow Start

Reset `CongestionWindow` to 1

Perform exponential increase

$$\texttt{CongestionWindow} \leftarrow \texttt{CongestionWindow} + 1$$

- Until timeout at start of connection

  $\rightarrow$ rapidly probe for available bandwidth

- Until `CongestionWindow` hits `SlowStartThreshold` following Congestion Avoidance

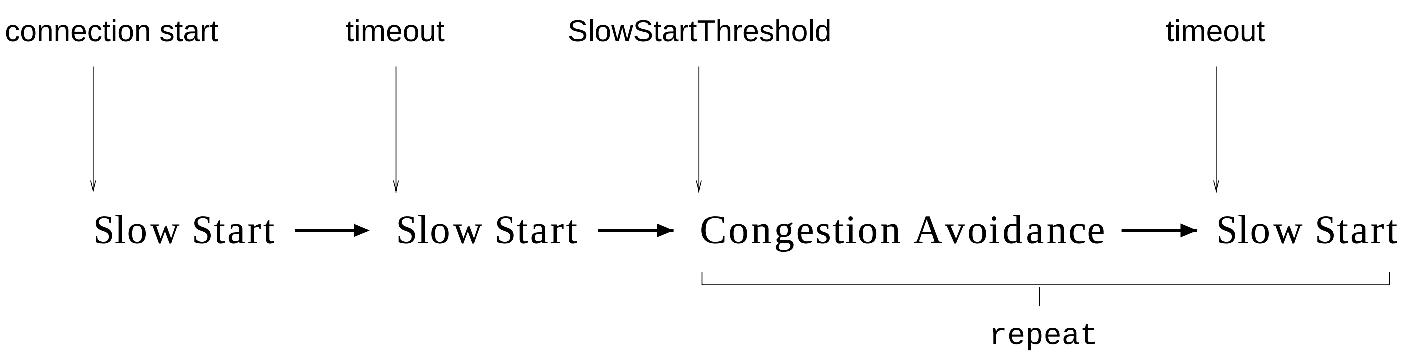
  $\rightarrow$ rapidly climb to safe level

$\longrightarrow$ "slow" is a misnomer

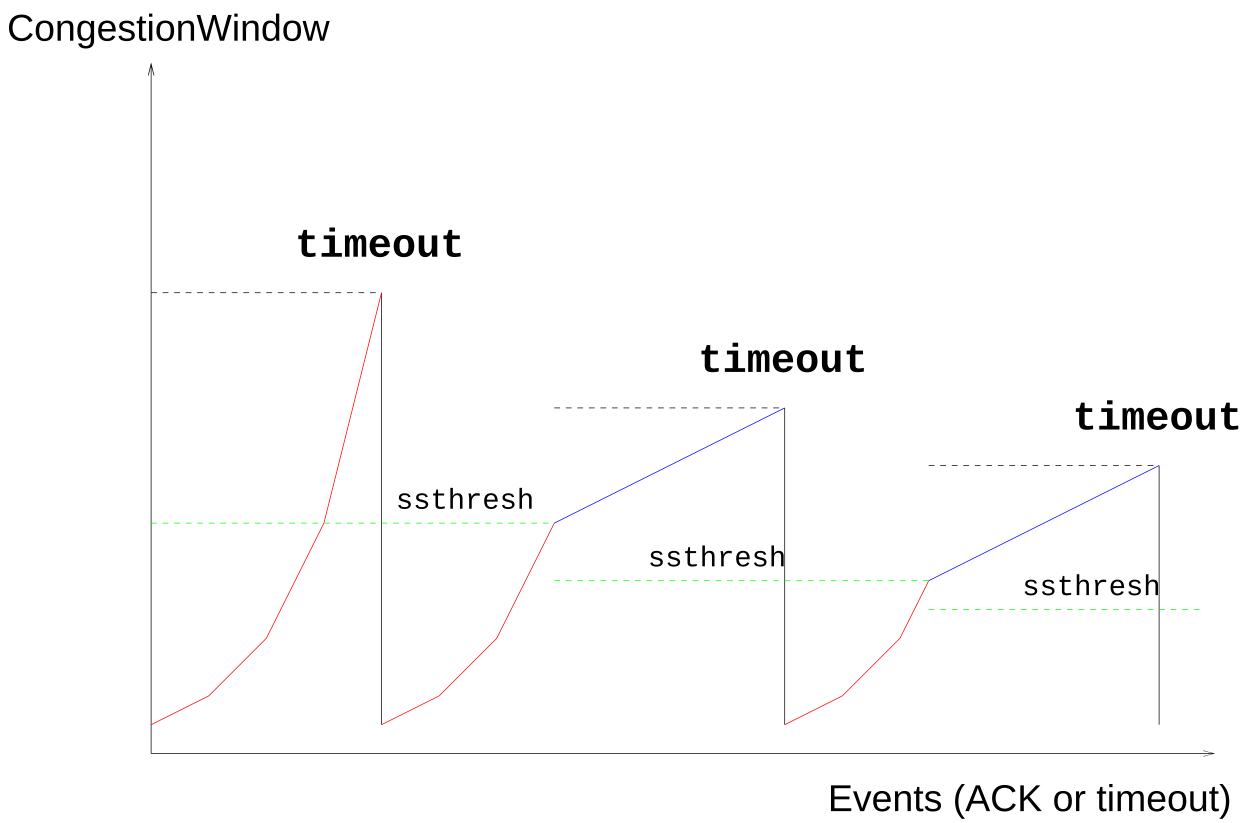
$\longrightarrow$ exponential increase is super-fast

Basic dynamics:

- $\longrightarrow$ after connection set-up
- $\longrightarrow$ before connection tear-down



`CongestionWindow` evolution:



$\longrightarrow$ what happens if receiver window size hits max?

$\longrightarrow$ DOE, supercomputing centers, etc.

(iii) Exponential timer backoff

$$\texttt{TimeOut} \leftarrow 2 \cdot \texttt{TimeOut} \qquad \text{if retransmit}$$

(iv) Fast Retransmit

Upon receiving three duplicate ACKs:

- Transmit next expected segment

  $\rightarrow$ segment indicated by ACK value

- Perform exponential backoff and commence Slow Start

  $\longrightarrow$ three duplicate ACKs: likely segment is lost

  $\longrightarrow$ react before timeout occurs

TCP Tahoe: features (i)-(iv)

(v) Fast Recovery

Upon Fast Retransmit:

- Skip Slow Start and commence Congestion Avoidance

  $\longrightarrow$ dup ACKs: likely spurious loss

- Insert "inflationary" phase just before Congestion Avoidance

Inflationary phase:

- `SlowStartThreshold` $\leftarrow$ `CongestionWindow` / 2
- `CongestionWindow` $\leftarrow$ `SlowStartThreshold` + 3
- On each additional duplicate ACK, increment `CongestionWindow`
- On first non-dup ACK, commence Congestion Avoidance

  `CongestionWindow` $\leftarrow$ `SlowStartThreshold`

TCP Reno: features (i)-(v)

$\longrightarrow$ pre-dominant form

Many more versions of TCP:

$\longrightarrow$ NewReno w/ SACK, w/o SACK, Vegas, etc.

$\longrightarrow$ wireless, ECN, multiple time scale

$\longrightarrow$ mixed verdict; pros/cons

Given sawtooth behavior of TCP's linear increase/exponential backoff:

Why use exponential backoff and not Method D?

- For multimedia streaming (e.g., pseudo real-time), AIMD (Method B) is not appropriate

  $\longrightarrow$ use Method D

- For unimodal case—throughput decreases when system load is excessive—story is more complicated

  $\longrightarrow$ asymmetry in control law needed for stability