Comparison of TCP Reno and TCP Vegas via Fluid Approximation

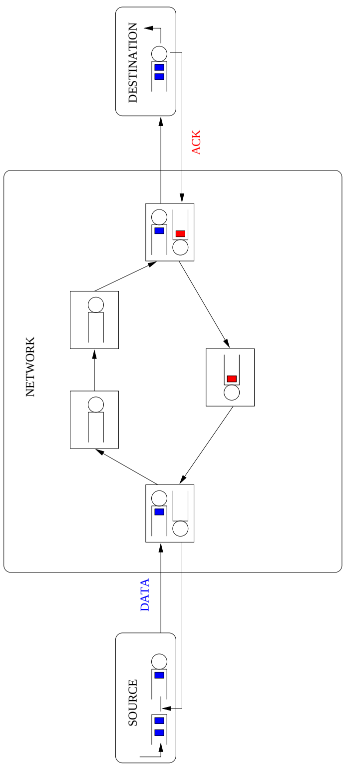
Thomas Bonald
Thomas.Bonald@inria.fr

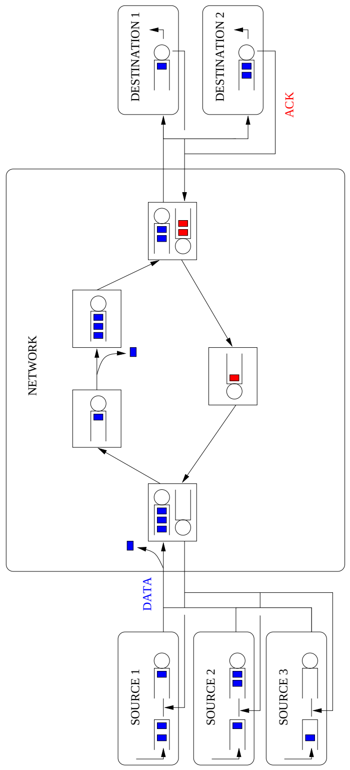
INRIA SOPHIA ANTIPOLIS

Workshop on the modeling of TCP

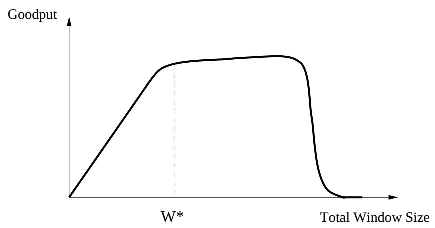
December 7-8, 1998

Outline





Static *vs* Dynamic Window



Two reasons for using a **dynamic** window:

1. The "optimal" window size W^* depends on the characteristics of the link (speed of the routers, propagation delay,...);
2. The number of connections sharing this link changes with time.

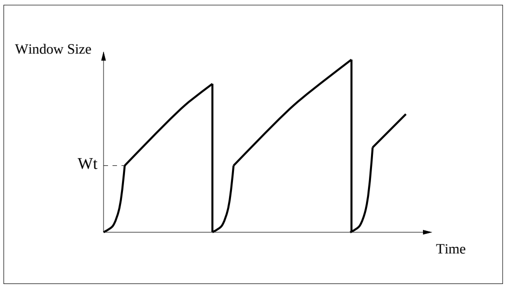
TCP Reno

Additive Increase:

`every` ACK `do`
`if` $W < W_t$ `then` $W := W + 1$
`else` $W := W + 1/W$

Multiplicative Decrease:

`every` LOSS `do`
$W_t := \gamma\ W$
$W := 1$



TCP Vegas

Additive Increase-Decrease:

`every` RTT `do`
`if` Diff $< \alpha/\text{RTT}_{\text{min}}$ `then` $W := W + 1$
`else if` Diff $> \beta/\text{RTT}_{\text{min}}$ `then` $W := W - 1$

Measurement: RTT, min(RTT).

Computation: Diff $= \lambda_{\text{exp}} - \lambda_{\text{act}}$.

Expected throughput:

$$\lambda_{\text{exp}} = \frac{W}{\text{RTT}_{\text{min}}}$$

Actual throughput:

$$\lambda_{\text{act}} = \frac{W}{\text{RTT}}$$

Question

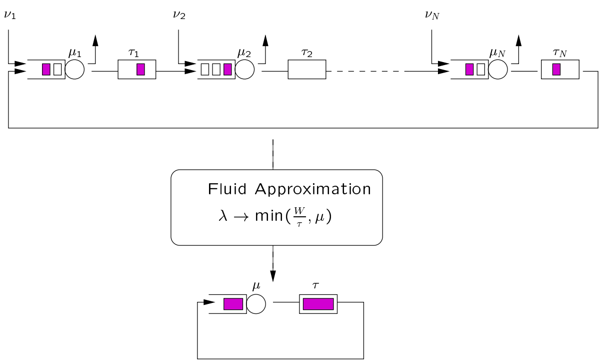
Does Vegas improve the performance of the *flow control* of TCP ?

Issues which are not addressed:

- **Error control**
  Loss detection, retransmission, ...
- **Transient behavior**
  Transfer of short files, slow-start phase,...

1 - Network Model

Consider the case of a **single** connection, with **saturated** input and **static** window size W.

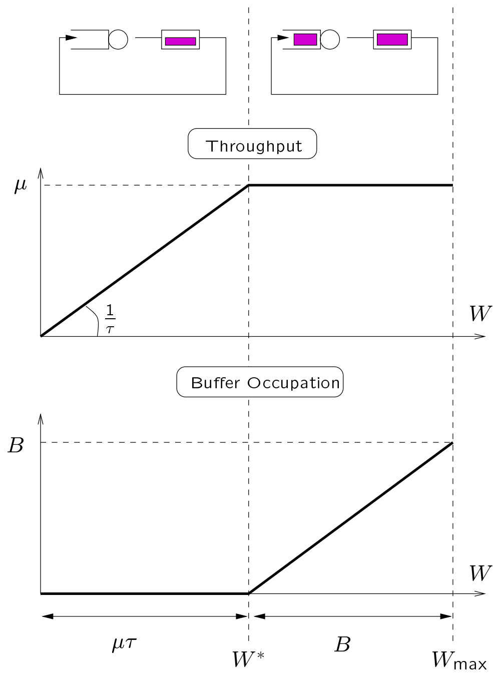


The model reduces to a single *bottleneck*, with

$$\mu = \min_i(\mu_i - \nu_i)$$

$$\tau = \tau_1 + \ldots + \tau_N$$

$\lambda = \min\left(\frac{W}{\tau}, \mu\right)$



Fluid Approximation

In the case of a **static** window size W, the throughput of the connection is given by

$$\lambda = \min\left(\frac{W}{\tau}, \mu\right). \qquad (1)$$

By Little's law, we obtain the round-trip time of the packets:

$$\text{RTT} = \frac{W}{\lambda}. \qquad (2)$$

Assumption 1: When the window is **dynamic**, the throughput and the round-trip time are still given by (1) and (2) at any time t.

2 - Window Dynamics

TCP Reno:

$$\frac{dW}{dt} = \frac{1}{\text{RTT}}.$$

TCP Vegas:

$$\frac{dW}{dt} = \frac{\varepsilon}{\text{RTT}},$$

with

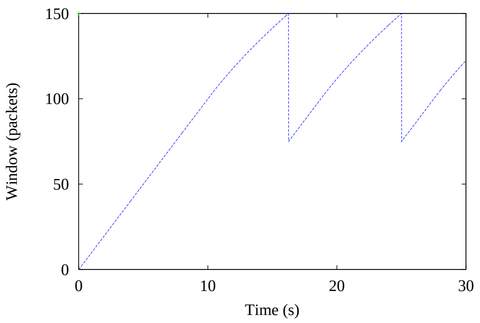
$$\varepsilon = \begin{cases} 1 & \text{if} \quad \text{Diff} \times \text{RTT}_{\min} < \alpha, \\ -1 & \text{if} \quad \text{Diff} \times \text{RTT}_{\min} > \beta, \\ 0 & \text{otherwise.} \end{cases}$$

Loss Detection:

$$W(t) = W_{\max} \;\Rightarrow\; W(t^+) = \gamma\, W(t).$$

Example 1 (TCP Reno)

$\mu = 1000$ packets/s
$\tau = 100$ ms
$B = 50$ packets
$\gamma = 1/2$



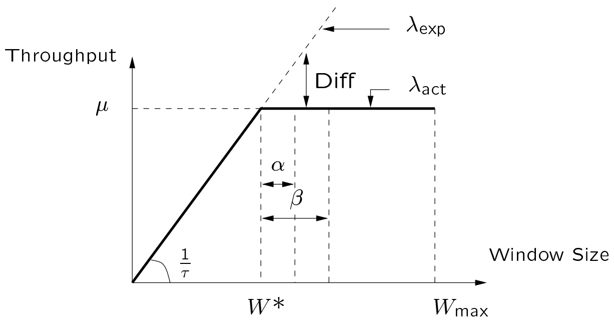
TCP Vegas

Computation : $\text{Diff} = \lambda_{\text{exp}} - \lambda_{\text{act}}$.
Expected throughput:

$$\lambda_{\text{exp}} = \frac{W}{\text{RTT}_{\text{min}}}$$

Actual throughput:

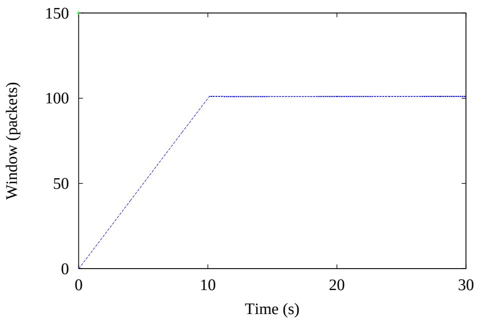
$$\lambda_{\text{act}} = \frac{W}{\text{RTT}}$$



Stabilization :
$\text{Diff} \times \text{RTT}_{\text{min}}$ between α and β.

Example 1 (TCP Vegas)

$\mu = 1000$ packets/s
$\tau = 100$ ms
$B = 50$ packets
$\alpha = 1$, $\beta = 3$.



Multiple Connections

First consider the case of a **fixed** number of K **homogeneous** connections.

TCP Reno:

$$\frac{dW_k}{dt} = \frac{1}{\text{RTT}}.$$

TCP Vegas:

$$\frac{dW_k}{dt} = \frac{\varepsilon_k}{\text{RTT}},$$

with

$$\varepsilon_k = \begin{cases} 1 & \text{if} \quad \text{Diff}_k \times \text{RTT}_{\text{min}} < \alpha, \\ -1 & \text{if} \quad \text{Diff}_k \times \text{RTT}_{\text{min}} > \beta, \\ 0 & \text{otherwise.} \end{cases}$$

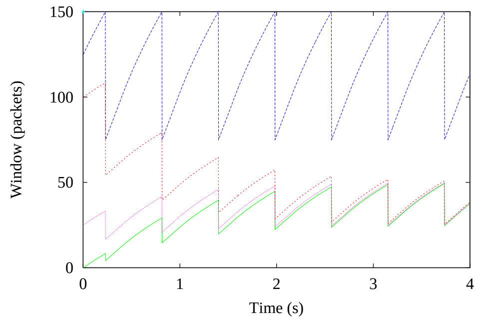
Synchronized Losses:

$$W(t) = W_{\text{max}} \;\Rightarrow\; W_k(t^+) = \gamma\, W_k(t).$$

Example 2 (TCP Reno)

$\mu = 1000$ packets/s
$\tau = 100$ ms
$B = 50$ packets
$\gamma = 1/2$
$K = 15$ connections

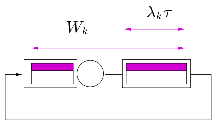
Initial State:
$W_{1,\dots,5} = 0,\ W_{6,\dots,10} = 5,\ W_{11,\dots,15} = 20.$



TCP Vegas

Computation :

$$\text{Diff}_k = \frac{W_k}{\text{RTT}_{\text{min}}} - \lambda_k.$$



If the windows stabilize, the buffer occupation of **each** connection lies between α and β in the steady state.

Stabilization: If $K\alpha < B$, the window sizes stabilize in finite time. In the steady state,

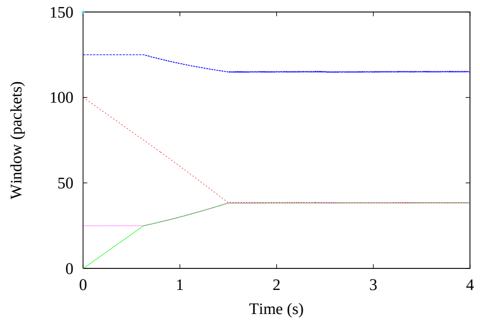
$$\frac{\max(\lambda_k)}{\min(\lambda_k)} \leq \frac{\beta}{\alpha}.$$

If $K\alpha \geq B$, then after a transient period, *each* of the K TCP Vegas connections behaves like a TCP Reno connection.

Example 2 (TCP Vegas)

$\mu = 1000$ packets/s
$\tau = 100$ ms
$B = 50$ packets
$\alpha = \beta = 1$
$K = 15$ connections

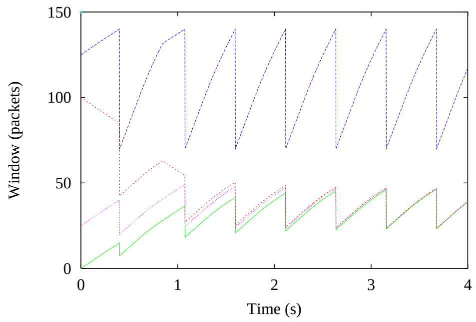
Initial State:
$W_{1,\dots,5} = 0,\ W_{6,\dots,10} = 5,\ W_{11,\dots,15} = 20.$



Example 3 (TCP Vegas)

$\mu = 1000$ packets/s
$\tau = 100$ ms
$B = 40$ packets
$\alpha = \beta = 3$
$K = 15$ connections

Initial State:
$W_{1,\dots,5} = 0,\ W_{6,\dots,10} = 5,\ W_{11,\dots,15} = 20.$



3 - Efficiency

TCP Reno:

$\surd$ Sensitive to the *bandwidth-delay* product (with respect to the buffer size)

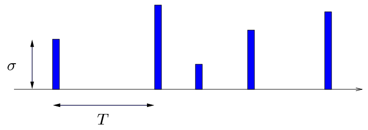
Insensitive to the number of connections

TCP Vegas:

Insensitive to the bandwidth-delay product

$\surd$ Sensitive to the *number of connections* (with respect to the buffer size)

Dynamic Model



$$\text{Load} = \frac{\sigma/T}{\mu}.$$

Assumption 2: The connections are in the steady state described above at any time t.

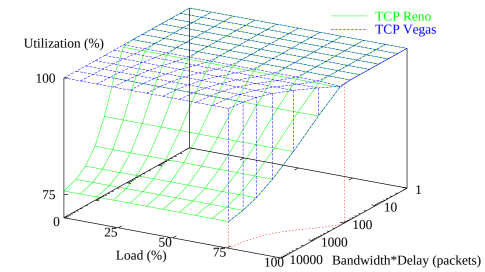
Average performance criteria:

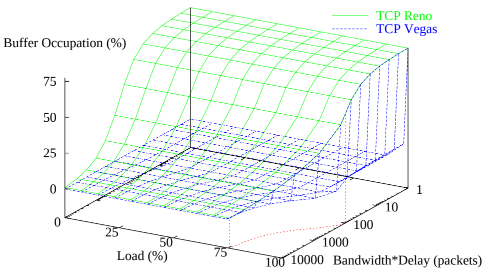
$$\text{Utilization} = \frac{1}{\mu}\,\mathbb{E}(\lambda \mid K > 0)$$

$$\text{Buffer Occupation} = \frac{1}{B}\,\mathbb{E}(W - \lambda\tau \mid K > 0)$$

Stability Condition:

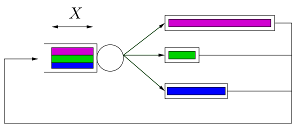
$$\boxed{\text{Load} < \text{Utilization(Reno)}}$$





4 - Fairness

Case of a **fixed** number of K **heterogeneous** connections, with propagation delays $\tau_1, \ldots, \tau_K$.



The **round-trip times** are given by

$$\text{RTT}_k = X/\mu + \tau_k.$$

In particular,

$$X = 0 \quad \Longleftrightarrow \quad \sum_{k=1}^{K} \frac{W_k}{\tau_k} \le \mu,$$

and when $X > 0$, we have

$$\sum_{k=1}^{K} \frac{W_k}{X/\mu + \tau_k} = \mu.$$

TCP Reno

Small propagation delays:

$$\frac{dW_k}{dt} \approx \frac{\mu}{X} \qquad \Rightarrow \qquad \frac{W_k}{W_l} \approx 1$$

$$\boxed{\frac{\lambda_k}{\lambda_l} \approx 1}$$

Large propagation delays:

$$\frac{dW_k}{dt} \approx \frac{1}{\tau_k} \qquad \Rightarrow \qquad \frac{W_k}{W_l} \approx \frac{\tau_l}{\tau_k}$$

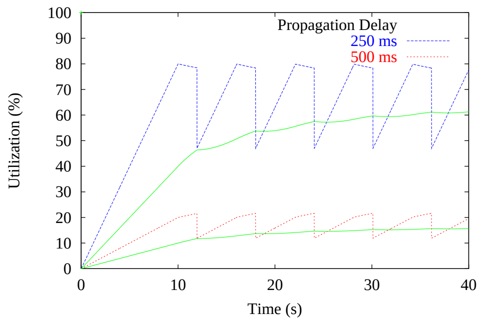
$$\boxed{\frac{\lambda_k}{\lambda_l} \approx \left(\frac{\tau_l}{\tau_k}\right)^2}$$

$\Rightarrow$ TCP Reno significantly discriminates against connections with **large** propagation delays.

Example (TCP Reno)

$\mu = 1000$ packets/s
$B = 50$ packets
$\gamma = 1/2$
$K = 10$ connections

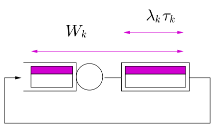
Propagation delays:
$\tau_{1,\dots,5} = 250$ ms, $\tau_{6,\dots,10} = 500$ ms



TCP Vegas

Computation :

$$\text{Diff}_k = \frac{W_k}{\tau_k} - \lambda_k.$$



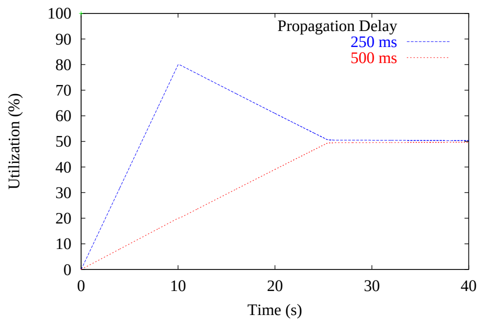
If the windows stabilize, the buffer occupation of **each** connection lies between α and β in the steady state.

$\Rightarrow$ In the case $\alpha = \beta$, TCP Vegas is **fair**.

Example (TCP Vegas)

$\mu = 1000$ packets/s
$B = 50$ packets
$\alpha = \beta = 1$
$K = 10$ connections

Propagation delays:
$\tau_{1,\dots,5} = 250$ ms, $\tau_{6,\dots,10} = 500$ ms



Conclusion

Main benefits of TCP Vegas:

- **Conservative** behavior
  ⇒ Should improve the overall performance of the Internet;

- **Insensitive** to the bandwidth-delay product
  ⇒ Suitable for satellite links;

- **Fair** sharing of the bandwidth
  (provided $\alpha = \beta$)

Open Questions

- **Measurement** of the minimum RTT

  Varying propagation delay
  (e.g. due to route changes)
  Effect of queueing delays
  $\sim K \log(K)\alpha$ instead of $K\alpha$

- How to **deploy** TCP Vegas?

  Vegas $>$ Reno ⇒ natural
  Vegas $<$ Reno ⇒ enforced (pricing,...)