Towards a low-delay Internet

Exploiting ACK-clock stability in congestion control

Niels Möller

2012-08-30

Joint work with Krister Jacobsson and Karl Henrik Johansson.

About this talk

Short bio

- PhD 2008, graduate studies at KTH, automatic control.
- Currently at Conemtech. Network time synchronization.
- Spare time projects include GNU Nettle (crypto library) and GNU GMP (bignum arithmetic).
- Undergraduate studies at Linköping.

Topics of this presentation

- Work done during my PhD time.
- Considering publishing in book format.

Outline

Window based congestion control

Analysis of the ACK-clock

Congestion control for small queues

Simulation results

Objectives and constraints

Objectives

- Avoid network overload.
- Efficient resource utilization.
- “Fair” sharing of resources.
- Small queueing delays.

Objectives and constraints

Objectives

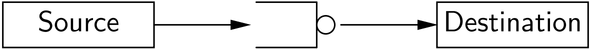
- Avoid network overload.
- Efficient resource utilization.
- "Fair" sharing of resources.
- Small queueing delays.

Design constraints

- End-to-end principle.
- Robustness to uncertainties.
- Tunability.
- Incremental deployability.

Notation

Single link, single flow topology

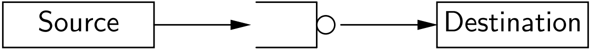


Constant network parameters

c	capacity of bottleneck link [bytes/s]
τ	end-to-end propagation delay [s]
γ	proportion of capacity which is available
m	packet size [bytes]

Notation

Single link, single flow topology



Source

Destination

Constant network parameters

c	capacity of bottleneck link [bytes/s]
τ	end-to-end propagation delay [s]
γ	proportion of capacity which is available
m	packet size [bytes]

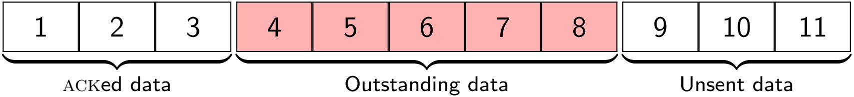
System state

$q(t)$	queue size [bytes]
$w(t)$	window size [bytes]
$r(t)$	sending rate [bytes/s]

$\mathrm{RTT} = \tau + q(t)/c$

ACK-clock

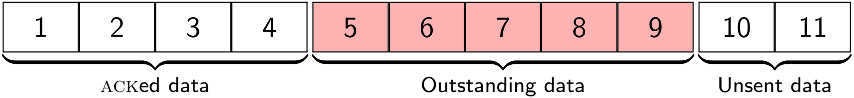
Window size: The amount of outstanding data.



1 2 3 4 5 6 7 8 9 10 11

ACKed data · Outstanding data · Unsent data

ACK for packet 4 received:



1 2 3 4 5 6 7 8 9 10 11

ACKed data · Outstanding data · Unsent data

ACK-clock: One packet sent for each received ACK. Sending rate roughly w/RTT.

Additive increase/multiplicative decrease

TCP Congestion avoidance algorithm

- All goes well: Increase *w* by one packet by RTT.
- When loss is detected: Reduce *w* by half.

Additive increase/multiplicative decrease

TCP Congestion avoidance algorithm

- All goes well: Increase *w* by one packet by RTT.
- When loss is detected: Reduce *w* by half.

TCP square root formula

$$\text{average rate} = \frac{m}{\mathrm{RTT}} \sqrt{\frac{2(1-p)}{p}}$$

Then

$$\begin{aligned} & \text{rate} \approx \frac{w}{\mathrm{RTT}} \\ \implies & p \approx 2/n^2 \qquad n = w/m, \text{ window size in packets} \end{aligned}$$

Additive increase/multiplicative decrease

TCP Congestion avoidance algorithm

- All goes well: Increase w by one packet by RTT.
- When loss is detected: Reduce w by half.

TCP square root formula

$$\text{average rate} = \frac{m}{\mathrm{RTT}}\sqrt{\frac{2(1-p)}{p}}$$

Then

$$\begin{aligned} \text{rate} &\approx \frac{w}{\mathrm{RTT}} \\ \implies p &\approx 2/n^2 \qquad n = w/m, \text{ window size in packets} \end{aligned}$$

Hand waving

Assume p is a function of network state, and also uniform $U(0, 4/n^2)$. Compute mutual information over one RTT:

$$\mathrm{I}(\text{ACKs}, \text{network state}) \approx 0.56/n \text{ bits/RTT}$$

Does TCP achieve the objectives?

Yes Avoid network overload.

Yes Efficient resource utilization.

Yes “Fair” sharing of resources.

No Small queueing delays.

Does TCP achieve the objectives?

Yes Avoid network overload.

Yes Efficient resource utilization.

Yes “Fair” sharing of resources.

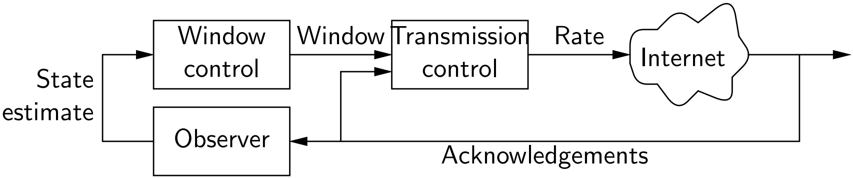
No Small queueing delays.

Popular (as of 2008. . .) TCP research

- Efficiency over wireless.
- Efficiency over fat pipes ($c\tau$ large).
- Delay-related: Buffer sizing, “flow aware” networking, QoS.

Analysis of the ACK-clock

A cascaded control system



Inner loop:	ACK-clock.
Outer loop:	Adaptation of the window size.
Measured signals:	ACK-packets.

- Two distinct control loops.
- Window size is a crucial state variable.

Queue dynamics

Standard fluid-flow model

$$\dot{q}(t) = \begin{cases} r(t) - \gamma c & q(t) > 0 \\ \max(0, r(t) - \gamma c) & q(t) = 0 \end{cases}$$

Queue dynamics

Standard fluid-flow model

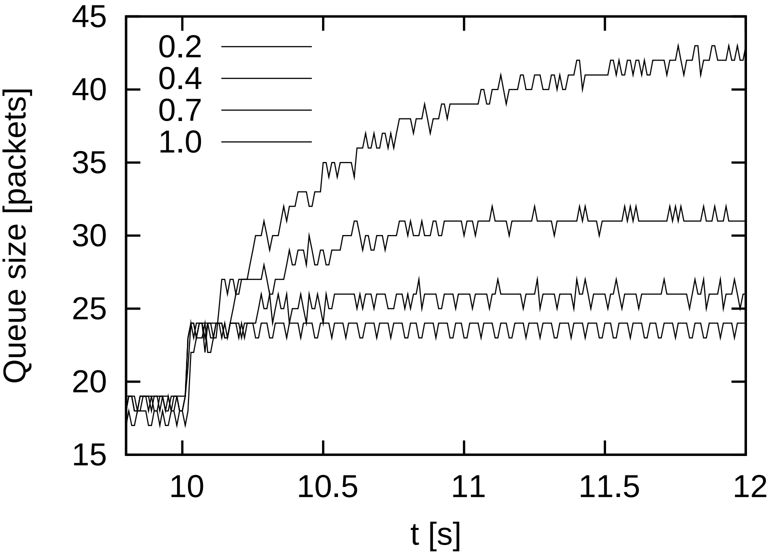
$$\dot{q}(t) = \begin{cases} r(t) - \gamma c & q(t) > 0 \\ \max(0, r(t) - \gamma c) & q(t) = 0 \end{cases}$$

How to get from w to r?

$r(t) = \dfrac{w(t)}{\tau}$ “Integrator”” model. Hollot et al. Infocom 2001

$q(t) = w(t) - c\tau$ “Static model. Wang, et al. Infocom 2005

Step responses



$q(t)$ response to a $w(t)$-step, for several values of γ.

A better inner loop model

$$\begin{aligned} r(t) &= \underbrace{\frac{w(t-\tau)}{\tau + q(t-\tau)/c}}_{\text{Rate of received \textsc{ack}s}} + \underbrace{\dot{w}(t)}_{\text{Direct term}} \\ \dot{q}(t) &= r(t) - \gamma c \\ &= \frac{w(t-\tau)}{\tau + q(t-\tau)/c} + \dot{w}(t) - \gamma c \end{aligned}$$

A better inner loop model

$$
\begin{aligned}
r(t) &= \underbrace{\frac{w(t-\tau)}{\tau + q(t-\tau)/c}}_{\text{Rate of received ACKs}} + \underbrace{\dot{w}(t)}_{\text{Direct term}} \\
\dot{q}(t) &= r(t) - \gamma c \\
&= \frac{w(t-\tau)}{\tau + q(t-\tau)/c} + \dot{w}(t) - \gamma c
\end{aligned}
$$

Time constant

- $\gamma \geq 0.3$: Time constant < 3.4 RTT.
- $\gamma < 0.3$: Time constant $\approx \text{RTT}/\gamma$.

Is the ACK-clock stable for an arbitrary network?

The model extends nicely to general networks.

Stability results

Fix the window size at each source. What happens at the queues?

- Total number of packets is bounded (trivial).
- Single link, single flow topology, arbitrary delay: Globally asymptotically stable.
- Single link, multiple flow topology, heterogeneous delays: Locally asymptotically stable.
- General network, simplified model without signalling delays: Globally asymptotically stable.

Large queue fluctuations seem unlikely.

Taking advantage of the stable inner loop

Lessons for outer loop design

- Inner loop stabilizes the system.
- For high cross traffic, dynamics of inner loop must not be ignored.
- Design the window update law for the other objectives.

Outer loop responsibilities

- Fair sharing between flows.
- Keep equilibrium queues small.
- And don't create instability.

Congestion control for small queues

New congestion control scheme

Rationale:

- Keep ACK-clock inner-loop.
- Additive increase implies a “pressure” on the queues.
- Need balancing back-pressure to stop queue growth.

New congestion control scheme

Rationale:

- Keep ACK-clock inner-loop.
- Additive increase implies a “pressure” on the queues.
- Need balancing back-pressure to stop queue growth.

Control laws:

- Usual additive increase.
- Packet marking probability $p(t) = q(t)/(q(t) + q_0)$
- Additive decrease for each ACK carrying a mark.

Comparison to traditional AQM + ECN

Source's point of view

- More frequent packet marks (average one mark / RTT).
- Cancels the additive increase on RTT time scale.

Comparison to traditional AQM + ECN

Source's point of view

- More frequent packet marks (average one mark / RTT).
- Cancels the additive increase on RTT time scale.

Router's point of view

- Each marked packet implies one less packet arriving an RTT later.
- Contrast standard ECN, where response is amplified by the unknown window size.
- Single tuning knob (q_0). Bad tuning cause reduced utilization or a large queue, but not large oscillations.

Stability

Model: Single link, single flow + cross traffic.

$$\dot{w}(t) = \frac{1}{\tau + q(t-\tau)/c}\left(m - \frac{q(t-\tau)w(t-\tau)}{q_0 + q(t-\tau)}\right)$$

$$\dot{q}(t) = \begin{cases} \dfrac{w(t-\tau)}{\tau + q(t-\tau)/c} + \dot{w}(t) - \gamma c & q(t) > 0 \\ \max\left(0, \frac{w(t-\tau)}{\tau+q(t-\tau)/c} + \dot{w}(t) - \gamma c\right) & q(t) = 0 \end{cases}$$

Theorem: Locally asymptotically stable if $q_0 \geq c\tau$ and $\gamma < 1$.

Further properties

Equilibrium

With $q_0 = c\tau$:

$$\begin{aligned} q^* &= m/\gamma \\ w^* &= \gamma c\tau + m \end{aligned}$$

Further properties

Equilibrium

With $q_0 = c\tau$:

$$q^* = m/\gamma$$
$$w^* = \gamma c\tau + m$$

Hand waving

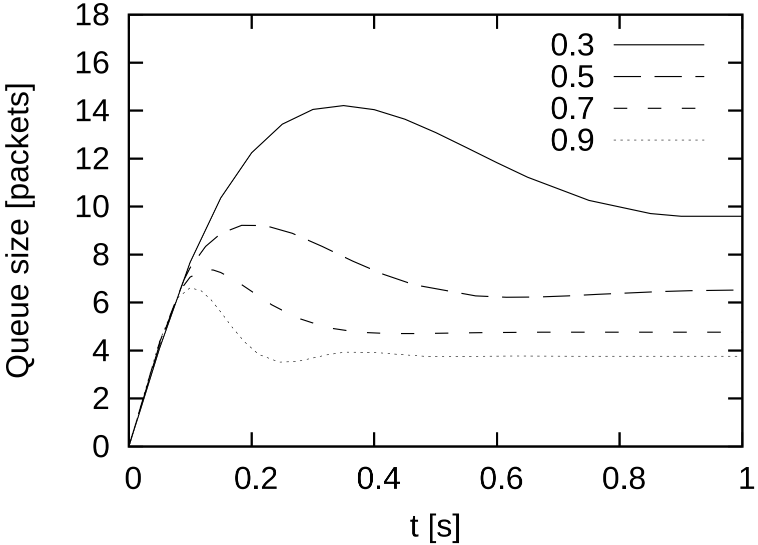
On average, $p = 1/n$. Mutual information received in one RTT:

$$\mathrm{I}(\textsc{acks}, \text{network state}) \approx 0.28 \text{ bits/RTT}$$

independent of the window size.

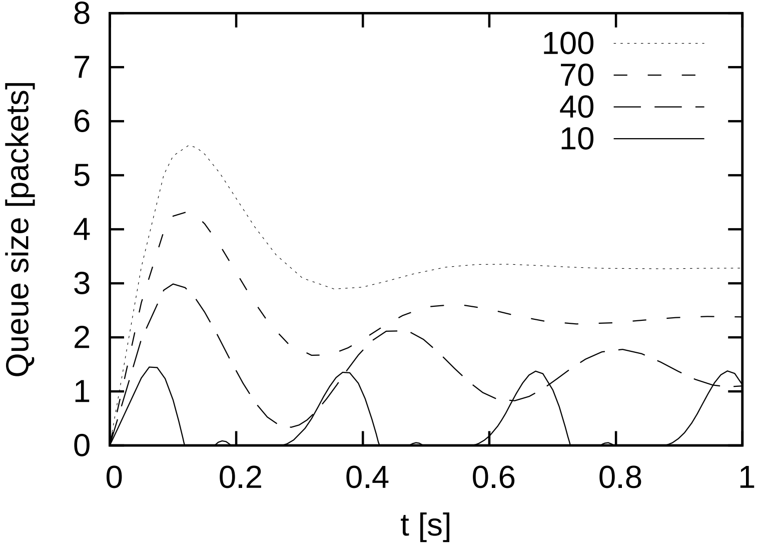
Simulation results

Fluid-flow simulation (1)



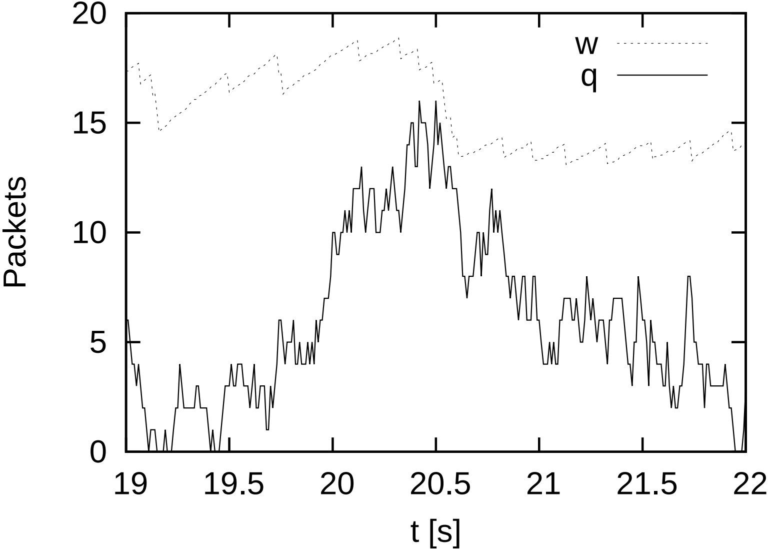
Varying γ, fraction of capacity available.

Fluid-flow simulation (2)



Varying the tuning parameter q_0.

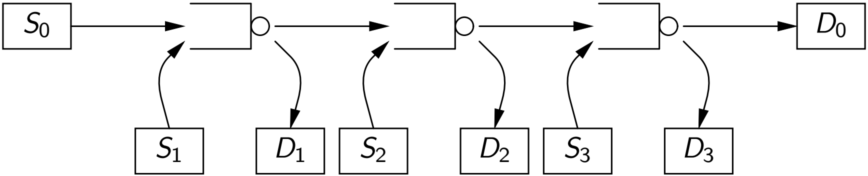
Packet simulation (1)



Step response: At $t = 20$, cross traffic increased from 20% to 40%.

Packet simulation (2)

“Parking lot” topology



Values for the leftmost link:

	New Reno	Vegas	New protocol
Loss	1.96	0.25	0.00
Util.	2.00	1.99	1.99
Queue	11.98	8.78	4.37
dev.	4.32	4.02	2.15

Packet simulation (3)

Long and short flow

	New Reno		Vegas		New protocol	
Flow	long	short	long	short	long	short
Throughput	0.11	1.49	0.88	0.71	0.12	1.48
Loss rate	6.55	1.69	0.37	0.31	0.00	0.00
Window	4.49	13.05	25.48	4.40	2.94	7.06
dev.	2.01	3.57	10.08	0.75	0.75	1.26
Delay	323.05	91.29	270.83	62.56	181.30	44.63
dev.	46.30	26.29	51.95	24.14	23.59	12.49

Conclusions and further work

Nice properties

- Small, stable queues.
- Easy to tune.
- Robust to uncertainties (RTT, cross traffic, # of flows, . . .).
- Fairness properties close to New Reno.

Conclusions and further work

Nice properties

- Small, stable queues.
- Easy to tune.
- Robust to uncertainties (RTT, cross traffic, # of flows, ...).
- Fairness properties close to New Reno.

Why?

- Router's point of view: Response to packet mark predictable.
- Source's point of view: More frequent feedback.
- Accurate model for inner-loop.

Conclusions and further work

Nice properties

- Small, stable queues.
- Easy to tune.
- Robust to uncertainties (RTT, cross traffic, # of flows, . . .).
- Fairness properties close to New Reno.

Why?

- Router's point of view: Response to packet mark predictable.
- Source's point of view: More frequent feedback.
- Accurate model for inner-loop.

Further work

- Behaviour with large aggregation.
- Using the mark bits also for switching from slow-astart to congestion avoidance.
- Rigorous analysis of feedback information contents.