

# ECN

the identifier of a new service model

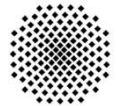
**Bob Briscoe, BT**

Mirja Kuhlewind, David Wagner, Stuttgart Uni

Koen De Schepper, Alcatel-Lucent

IETF-89 TSVAREA

Mar 2014



Universität  
Stuttgart



REDUCING INTERNET TRANSPORT LATENCY

Bob Briscoe & Koen de Schepper are part-funded by the European Community under its Seventh Framework Programme through the Reducing Internet Transport Latency (RITE) project ICT-317700

# outline

- classic ECN wasn't useful enough to deploy
- opportunity to identify a new service model using ECN
  - consistent low delay service *for all*
- incremental deployment path
- take-home message for AQM implementers:
  - please allow ECN & loss parameters to be independent

# classic ECN

just avoiding loss has not been good enough

- classic ECN = “drop equivalence”
  - networks & hosts treat ECN & drop identically [RFC3168]
- cannot justify the deployment pain\*
  - ECN largely off-by-default in Internet clients
- for a small performance gain
  - loss levels are low, and falling
  - applications can find other ways to mask losses
- ECN can do so much better...



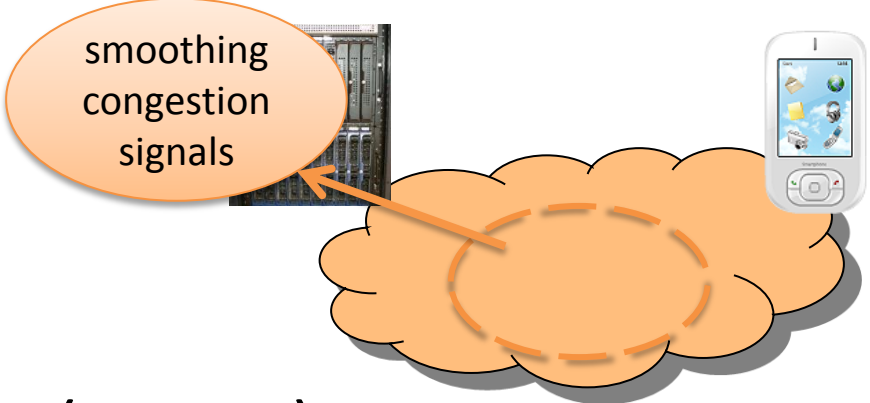
---

\* 2001: arbitrary blocking by firewalls & NATs – largely fixed by 2003

2002-2007: IP/ECN packets triggered **1 bugged home gateway model to crash** and 4 to increase drop

2009: some border gateways started zeroing ECN – fixes were rapidly deployed

# Immediate ECN + Smooth TCP response



smoothing  
congestion  
signals

- inspired by Data Centre TCP (DCTCP)
  1. AQM: Immediate ECN – remove sluggish signal smoothing
  2. TCP decrease: proportionate to ECN feedback over an RTT
  - deployed extensively in ECN data centres
- translate this approach to the public Internet?
  - more gain
    - promise of *consistent* low delay for *all*
  - less pain
    - removes sensitivity of AQM to sluggish RTT setting
    - can start with *existing* hardware, firmware or software

# consistent low delay for all

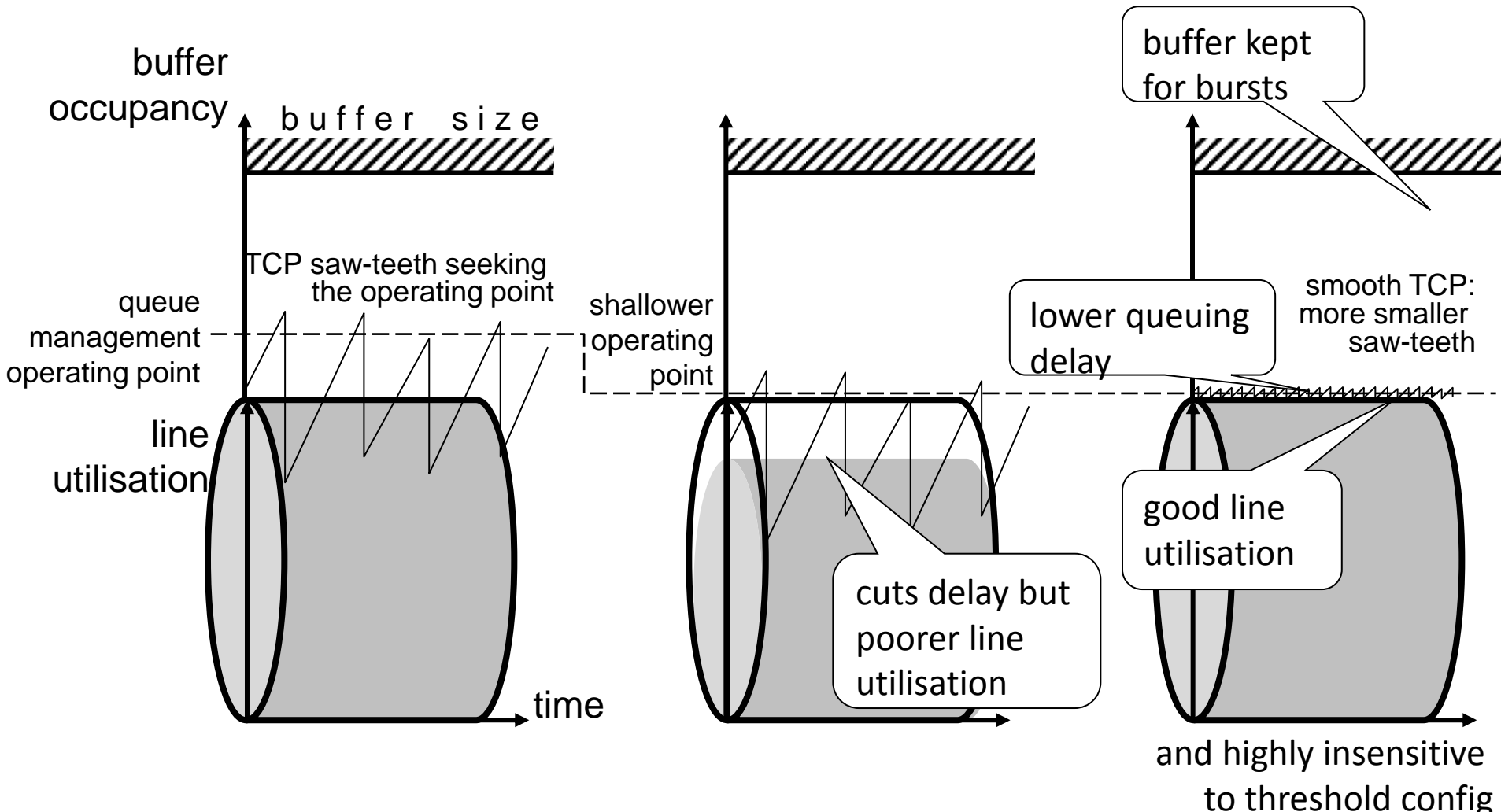
Today (at best)

TCP on end-systems

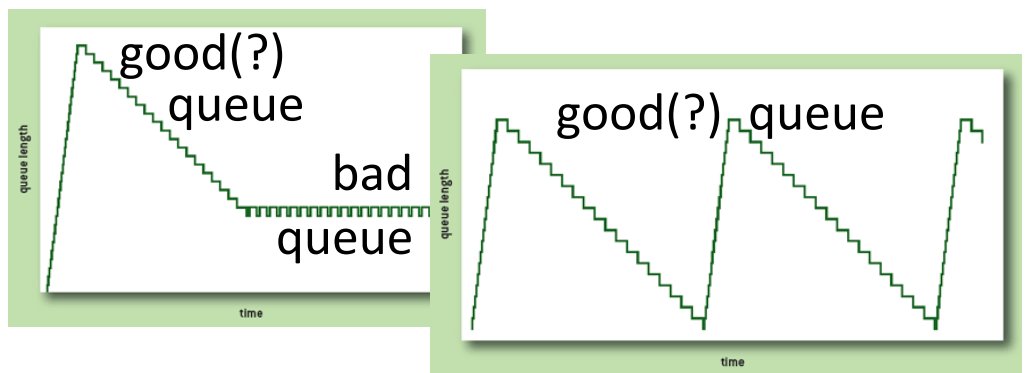
AQM at bottlenecks

if change bottlenecks alone

change bottlenecks  
and TCP



# RTT auto-tuning



## Drop

- new AQMs defer dropping for  $\sim 100\text{ms}$  (*worst-case* RTT)
  - no response for 5 CDN RTTs, or 25 media server RTTs

## ECN

- AQM can signal ECN immediately – no risk of impairment
- the transport can smooth out ECN bursts if it chooses
  - it knows if it's RTP or TCP in slow-start or cong-avoidance
- then a transport's smoothing delay is only its *own* RTT
  - short RTT flows can fill troughs and absorb peaks
  - no need to make all flows as sluggish as the worst-case RTT

- continues progress towards zero config

– line-rate auto-tuning: ARED ✓ CoDel ✓ PIE ✓

– **round-trip auto-tuning: Shift smoothing to hosts ✓**

100ms  
origin



20ms  
CDN



4ms

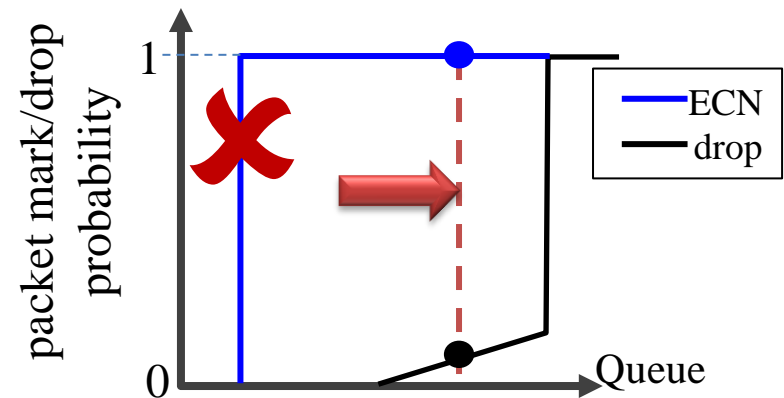


home  
media  
server



# deployment problem #1: co-existence of smoothed traffic and existing traffic

- data centre TCP was so-called only because it couldn't co-exist with Internet traffic
- for Internet: can't have a low delay threshold for ECN and a deep threshold for drop in one FIFO queue
- drop traffic would push the queue to its own balance point
- causing 100% marking of ECN packets
- then ECN traffic would starve itself



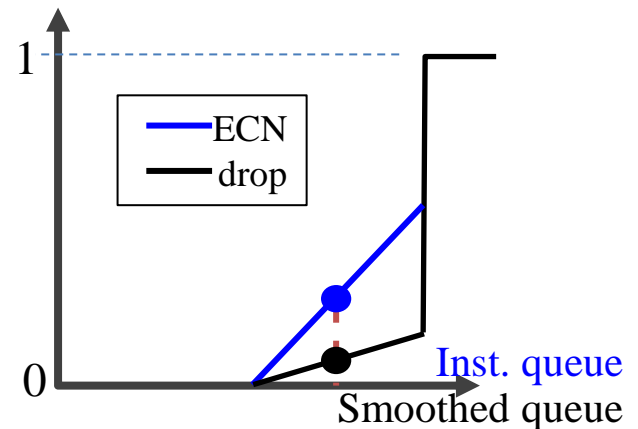
# problem #1 co-existence: solution

## initially use existing network hardware

- use weighted RED (WRED) implementation
- in an unusual configuration
  - one FIFO queue with two instances of RED algo
    - smoothed queue for drop (EWMA-constant = 9 say)\*
    - instantaneous queue for ECN (EWMA-constant = 0)
- may require new firmware / software

- similar approach possible for ECN in (fq\_)CoDel & (fq\_)PIE

**– please ensure parameters for ECN & drop are independent**



\* if exponential-weighting-constant = B,  
then RED smooths the queue over  $2^B$  packets  
if B = 9, RED smooths over  $2^9 = 512$  packets  
if B = 0, RED smooths over  $2^0 = 1$  packet (i.e. it doesn't smooth)

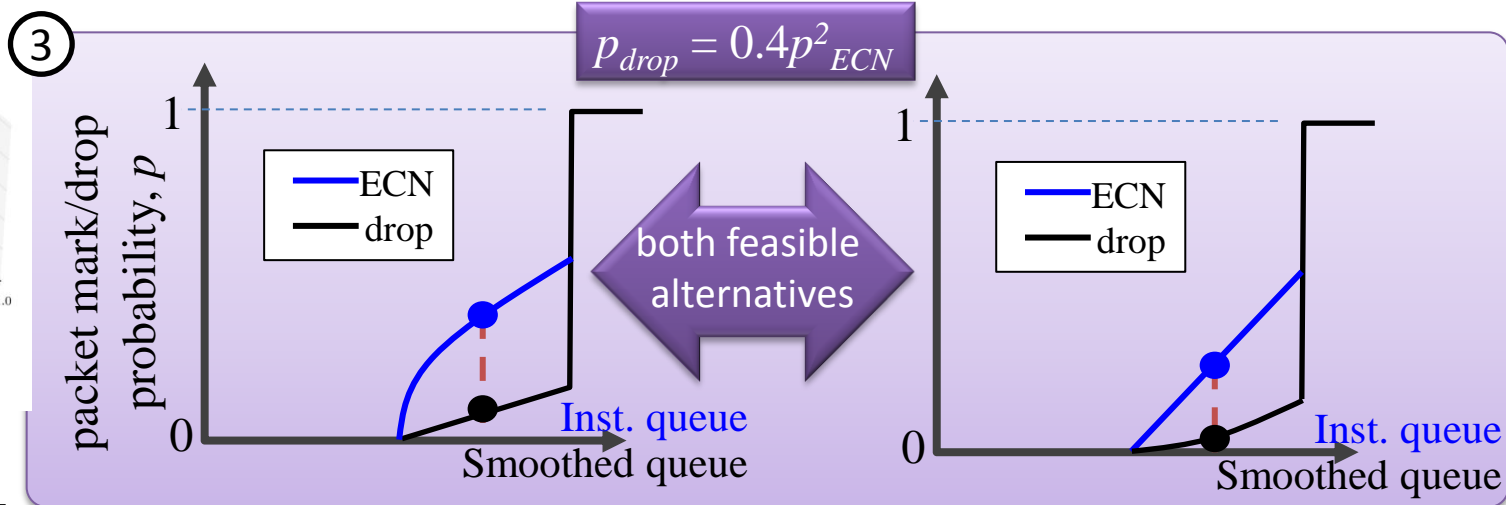
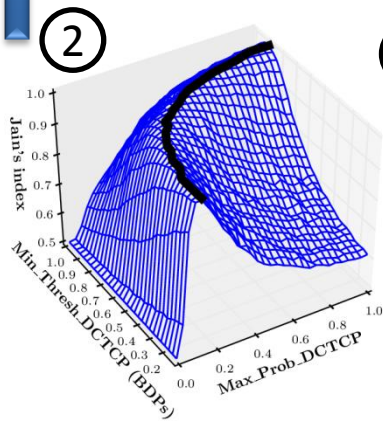
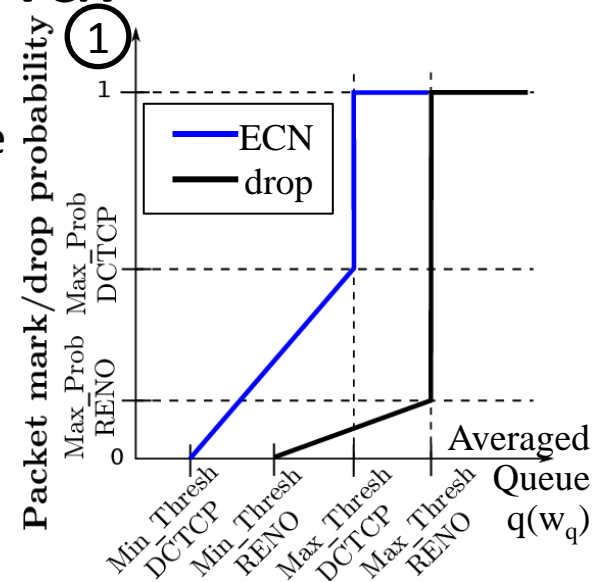


# problem #1 co-existence: solution

## results of testing so far

goals: a) robust against starvation; b) rough rate equality

1. simulated large part of the much larger parameter space
  - paper under submission, available on request
2. curve fit implied an analytical solution would exist
3. derived relation between ECN & drop curves
  - such that  $n$  ECN-DCTCP flows and  $m$  drop-Reno flows would all have equal rates for any  $n$  &  $m$
4. next steps:
  - test theory by simulation
  - test whether an approximation to the curve is good enough

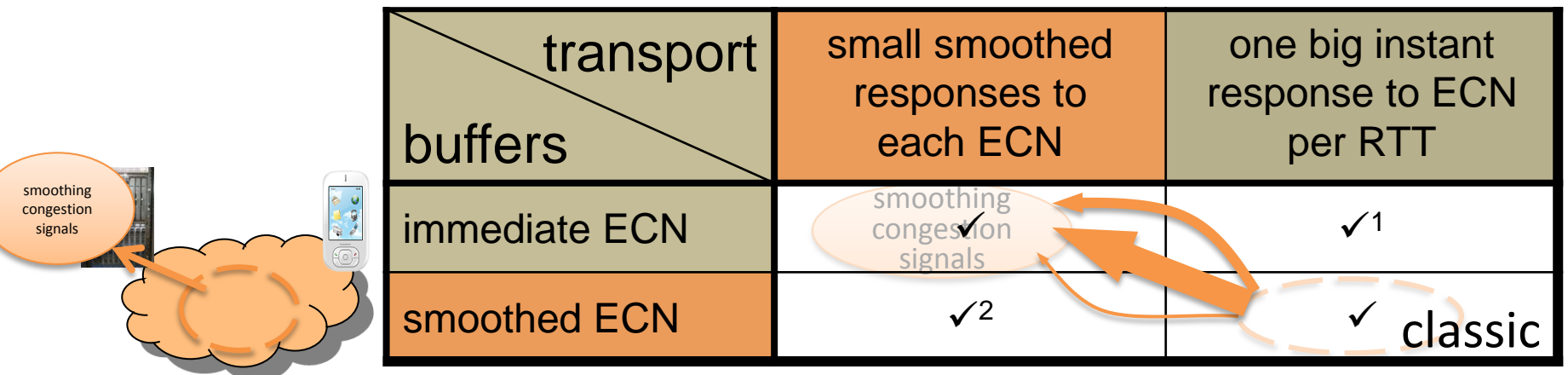


# deployment problem #2: solution

## incremental deployment

interop between classic and immediate ECN

- smoothed TCP (sndr) requires ‘accurate ECN feedback’ (rcvr)
  - tied together by accurate feedback negotiation during 3WHS
- server: ECN on-by-default in majority of servers
- client: turn ECN on-by-default in client when deploying:
  - accurate ECN feedback + ECN fall-back



<sup>1</sup> more marks from network, but one response per RTT at sender  
don't get smoother latency until host upgrades as well

(see [Tuning ECN for data center networks](#), Microsoft Research Asia, In Proc CoNEXT'12)

<sup>2</sup> doubly smoothed response to congestion

these two ticks are based on conjecture, not experimental evidence (yet)

# cross-layer / cross-wg impact on IETF

	problem	IETF wg formed
1	real-time media congestion avoidance	rmcat
2	prevent TCP bloating queues	aqm
3	prevent TCP's swings in delay	-



	component	IETF wg	document
3.1	redefine meaning of ECN CE	tsvwg	Expt update to RFC3168
3.2	specify ECN behaviour in AQM algos	aqm	CoDel, PIE, ARED
3.3	specify change to TCP feedback	tcpm	draft-ietf-tcpm-accecn-reqs draft-kuehlewind-accurate-ecn
3.4	specify change to TCP sender algo	tcpm	draft-bensen-tcpm-dctcp

1. RFC 3168 may not need to be updated (see spare slide)
2. urgent, given pace of AQM development
3. wire protocol: the main standards track change
4. algorithm experimentation expected

# Immediate ECN concluding messages

- promise of *consistent* low delay *for all*
  - host uses smoother sawteeth
  - immediate signals during dynamics  
not delayed for a worst-case round trip
- removes hard-coded round-trip-time from AQMs
- can use existing network hardware
- AQM implementers note well:
  - research in progress
  - but please allow ECN & loss parameters to be independent

# Immediate ECN

## Q&A

Q1: if subsequent experiments are as promising as these, would there be an appetite in the transport area to tweak the meaning of ECN?

spare slides

# flow separation (FQ\_AQM)

isolates me from other TCP sawteeth?

- it doesn't isolate me from my own sawteeth
  - queue varying between 0-1 RTT unavoidable in access links
  - video delivery over TCP needs larger play-out buffer
- flow separation
  - implies you have to choose the scheduling policy
    - IPSec VPNs – FQ gives them 1 flow's share
    - variable rate video wants to vary its share
    - LEDBAT wants to take less than its share
  - end-to-end principle
    - look very hard for an alternative before you break it



# which codepoint for immediate ECN?

- To use CE for immediate ECN,  
may not need to update RFC3168 (Addition of ECN to IP):

...if the ECT codepoint is set in that packet's IP header  
... then instead of dropping the packet, the router MAY  
instead set the CE codepoint in the IP header.

An environment where all end nodes were ECN-Capable could  
allow new criteria to be developed for setting the CE  
codepoint, and new congestion control mechanisms for end-  
node reaction to CE packets. However, this is a research  
issue, and as such is not addressed in this document.

- Could use ECT(1) for immediate ECN
  - but this unnecessarily wastes the CE codepoint  
(who would want 'sluggish ECN'?)

# a similar coexistence approach should be applicable to other AQMs

- ultimately, want to auto-tune against line-rate *and* RTT
  - use a modern AQM that uses queuing delay as its metric
  - *and* separate drop and ECN algos

AQM	smoothing parameter	non-ECN packets	ECN packets
ARED	ewma-const	9	0
PIE	max_burst	100ms	0
CoDel	interval	100ms	0

- message for implementers (esp. in silicon)
  - ensure parameters can be configured independently for ECN



# immediate *and* shallow ECN?

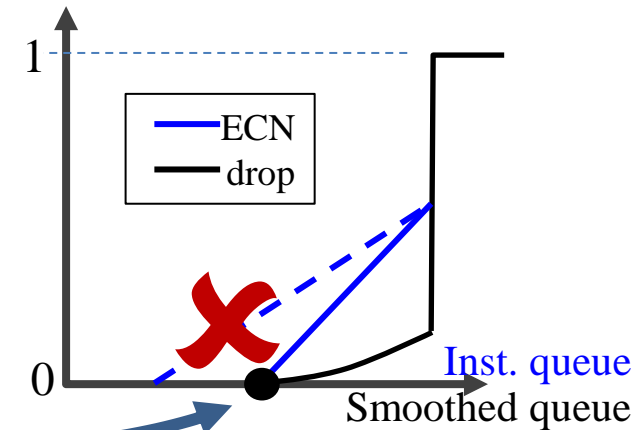
- to avoid starvation in any scenario
  - if FIFO, min thresholds for ECN & drop need to be the same
  - (in theory – to be tested)

## 1. first immediate ECN only

- incentivises initial deployment
- performance gain only from *immediate* ECN
- not from a *shallow* threshold
- cannot risk lower utilisation for prevalent non-ECN traffic

## 2. later, immediate and shallow ECN with shallow drop

- must shift both thresholds together
- as ECN traffic becomes prevalent
  - smoother ECN traffic will maintain high utilisation
  - also more insensitive to config



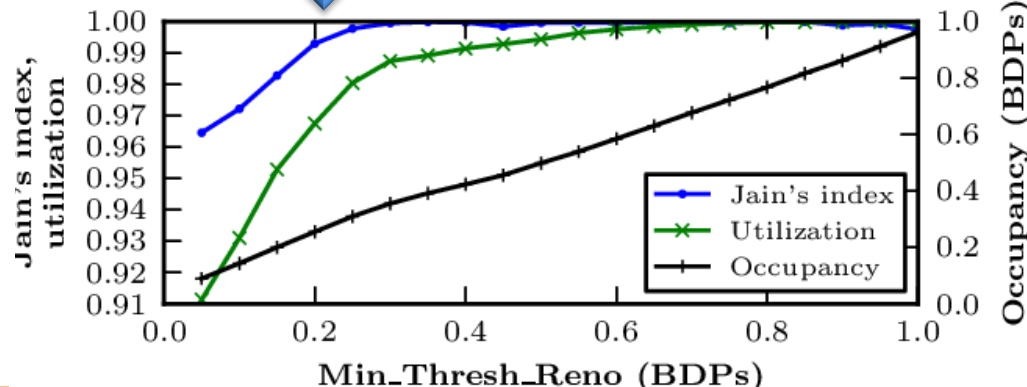
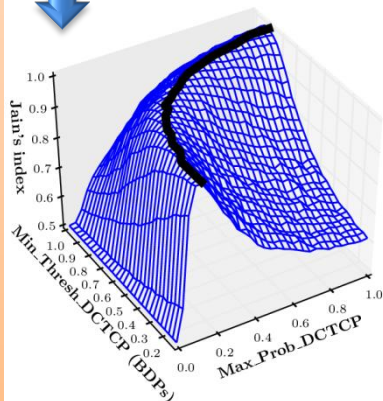
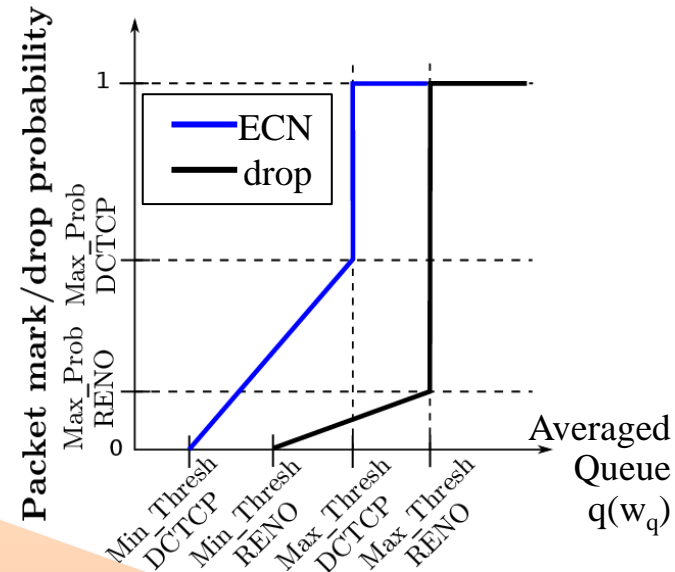
# co-existence

## results of 'gating tests'

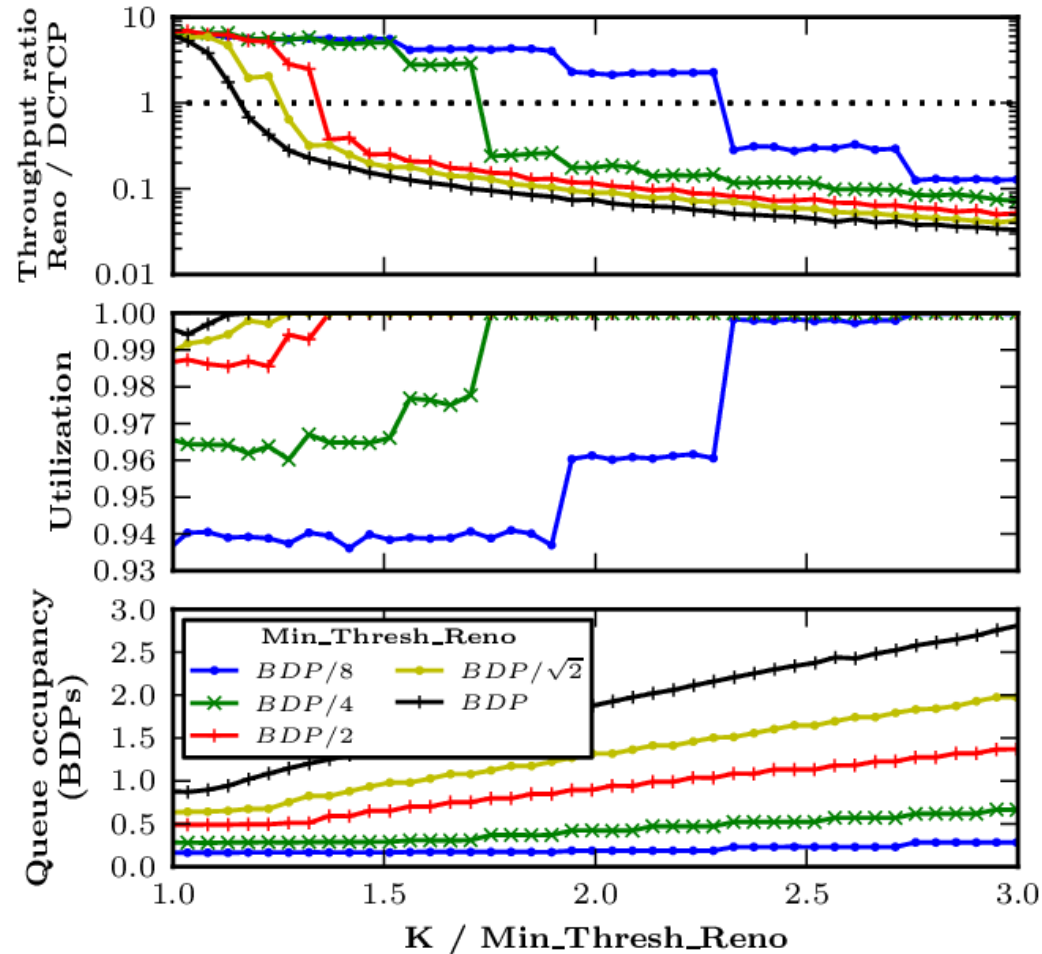
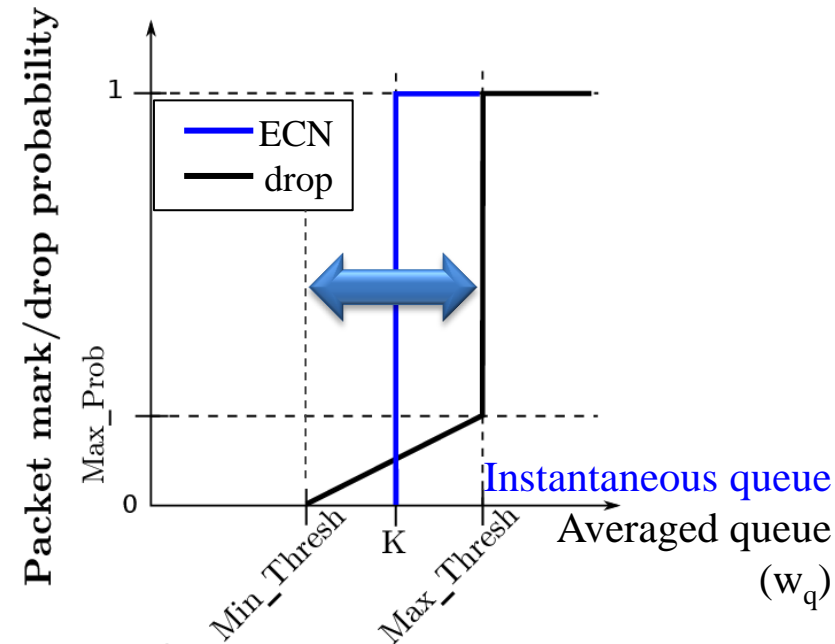
- explored large part of the much larger parameter space
  - implemented in Linux 3.2.18; simulated in IKR simlib
  - 'gating tests': long-running flows only
  - paper under submission, available on request

robust against starvation

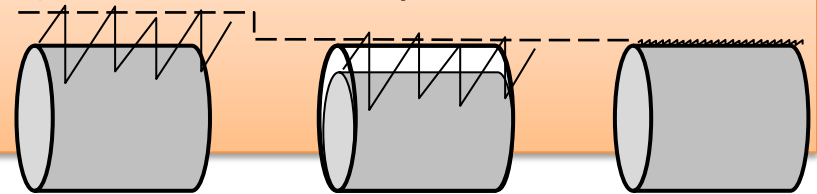
- formula to derive ECN config from drop config to maintain rate fairness
- can then find sweet spot for the drop config



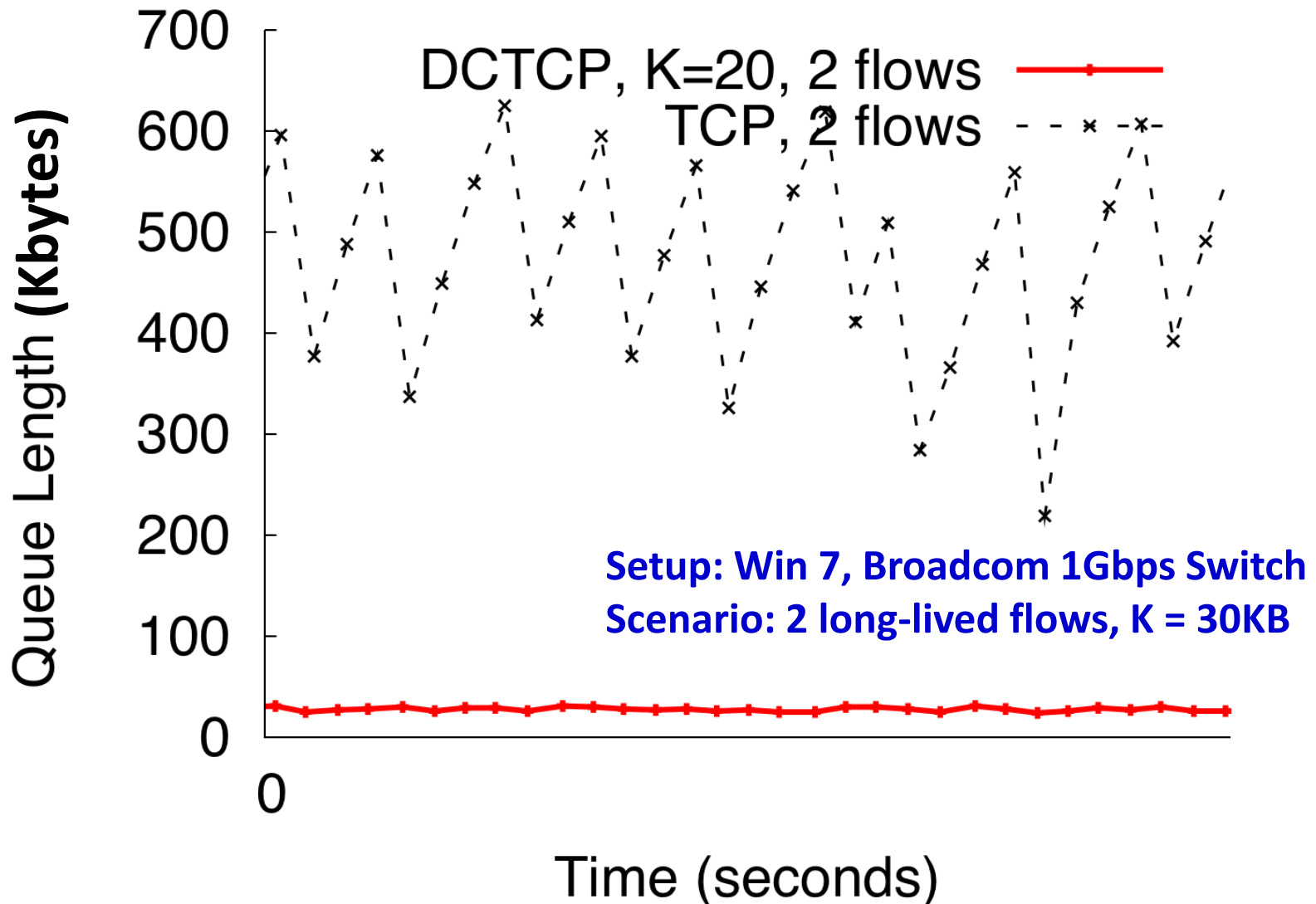
a sample of the results so far



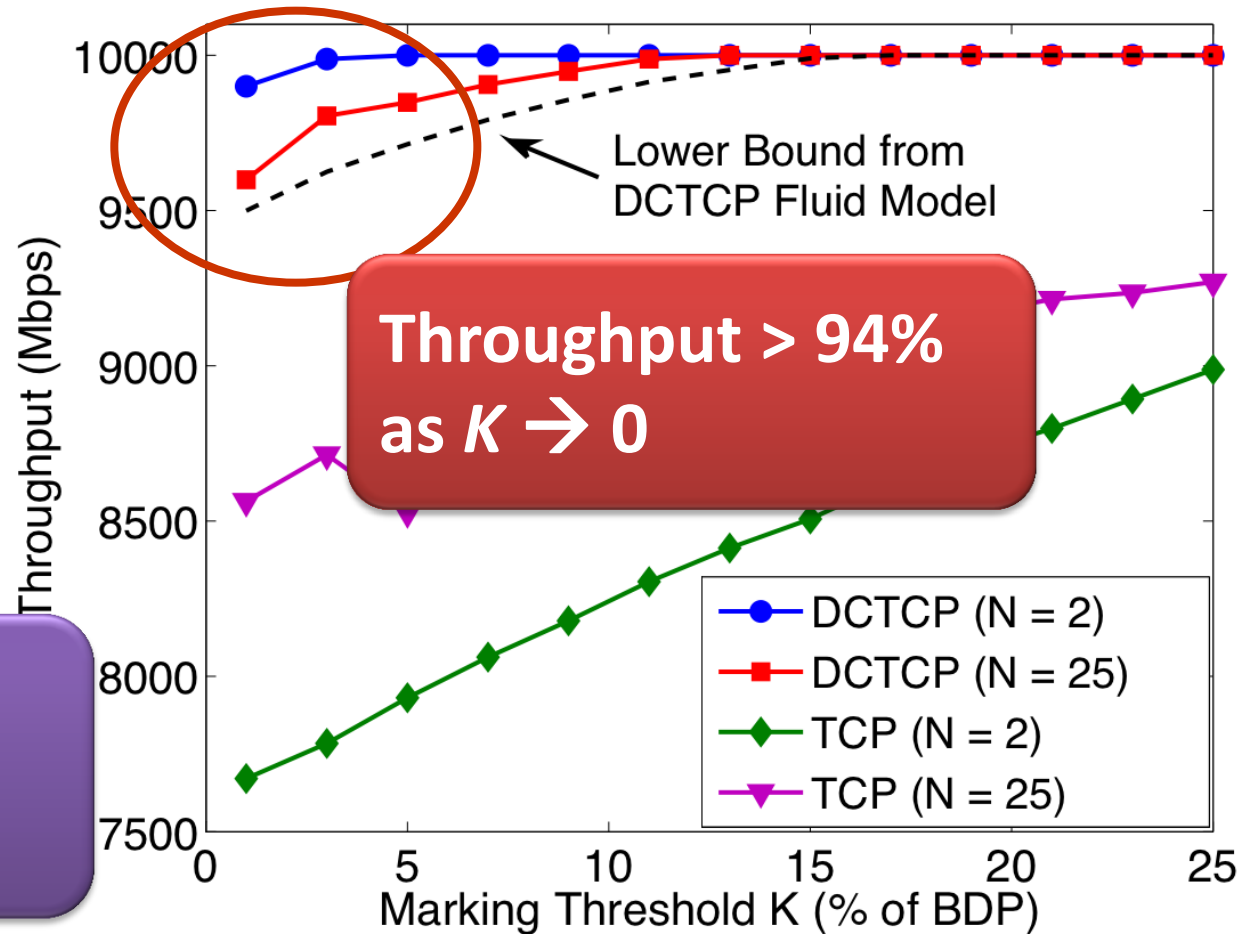
- early deployment, when traffic mostly drop-based have to set drop (and therefore ECN) threshold deep
- as more flows shift to DCTCP, can set both thresholds shallower



# DCTCP in Action



# Throughput-Latency Tradeoff

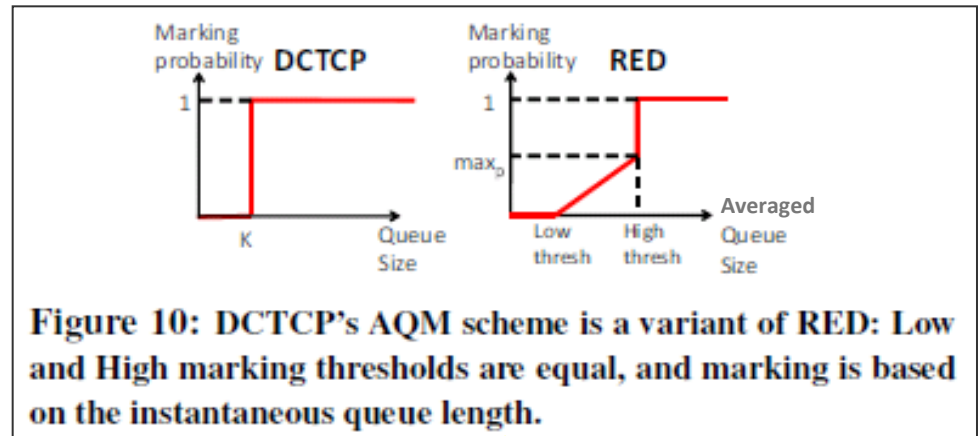


For TCP:  
Throughput  $\rightarrow$  75%

Throughput  $>$  94%  
as  $K \rightarrow 0$

Parameters:  
link capacity = 10Gbps  
RTT = 480 $\mu$ s  
smoothing constant (at source),  $g = 0.05$ .

# DCTCP activity



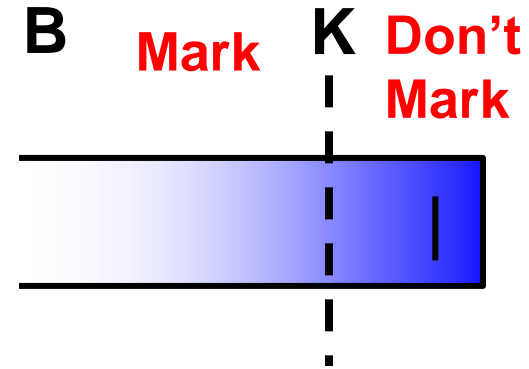
Figures courtesy of Alizadeh et al

- E2e Transport
  - In Windows 8 Server data center template
  - I-D for DCTCP feedback (intended EXP) [draft-kuehlewind-tcpm-accurate-ecn-01]
- AQM
  - Existing kit: Just a degenerate config of RED
  - Can be implemented as just a step at K packets (single 'if' command)
  - For zero-delay can use a virtual queue [RC5670]
    - hardware implementations [[“How to Build a Virtual Queue from Two Leaky Buckets”](#)]
    - see [HULL](#) for specifics with DCTCP
- Analysis, papers, Linux & ns2 implementation, etc
  - <http://www.stanford.edu/~alizade/Site/DCTCP.htm>
  - SIGCOMM paper gives entry point

# Data Center TCP Algorithm

## Switch side:

- Mark packets when **Queue Length > K**



## Sender side:

- Maintain **moving average** of **fraction** of marked packets ( $\alpha$ )

$$\text{each RTT : } F = \frac{\# \text{ of marked ACKs}}{\text{Total \# of ACKs}} \Rightarrow \alpha \leftarrow (1 - g)\alpha + gF$$

- Adaptive congestion window decrease:  $W \leftarrow (1 - \frac{\alpha}{2})W$