EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

FACULTY OF SCIENCE
Communication Networks

Alternative Best Effort (ABE) for Service Differentiation: Trading Loss versus Delay

Steffen Lindner, Gabriel Paradzik, Michael Menth

Published in IEEE/ACM Transactions on Networking
https://doi.org/10.1109/TNET.2022.3221553

http://kn.inf.uni-tuebingen.de

EBERHARD KARLS UNIVERSITÄT TÜBINGEN

Agenda

- Motivation
- Overview
- Credit Devaluation
- Implementation in Linux
- Results
- Conclusion

Motivation (1)

- Bufferbloat leads to temporarily long end-to-end delays
  - Real-time applications, e.g., VoIP, may be affected

- DiffServ may be used to offer various per-hop behaviors for service differentiation
  - Typically degrades best effort service for non-priority traffic
  - Controversial in the context of network neutrality

- **Idea:** Provide low-delay forwarding at the expense of additional packet loss
  - Without degrading best effort service
  - Users may choose between low-delay and best effort service

- ISPs may therefore leave the choice of PHB to end users

Motivation (2)

TABLE I
REALTIME APPLICATIONS WITH REQUIREMENTS REGARDING END-TO-END DELAY AND PACKET LOSS.

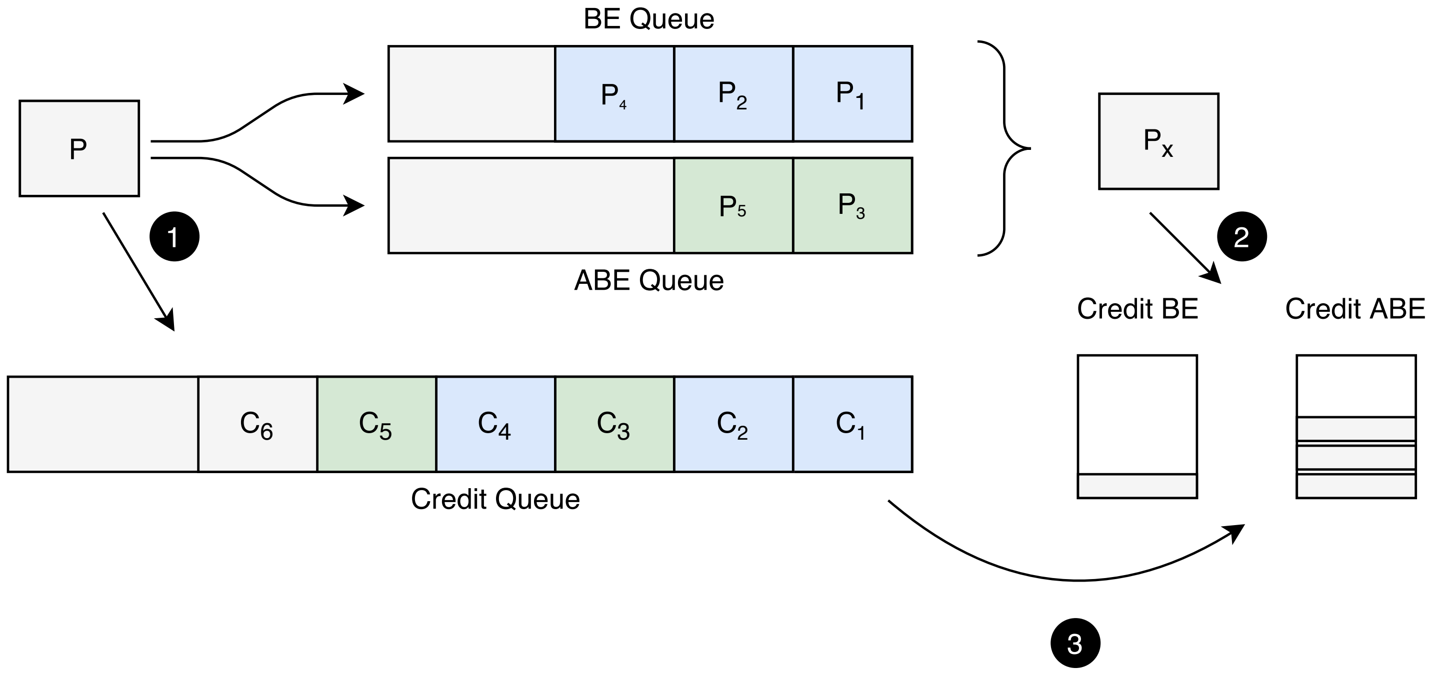
Application	E2E delay	Packet loss	Source
Online gaming (FPS)	20 ms – 80 ms	$\lesssim$ 5%	[2] [3]
Cloud gaming	$\lesssim$ 50 ms	$\lesssim$ 5%	[4] [5] [6]
Voice over IP (VoIP)	$\lesssim$ 150 ms	1% – 3%	[7]

- Has been already discussed in IETF
  - draft-hurley-alternative-best-effort (2000)
  - draft-you-tsvwg-latency-loss-tradeoff (2016)

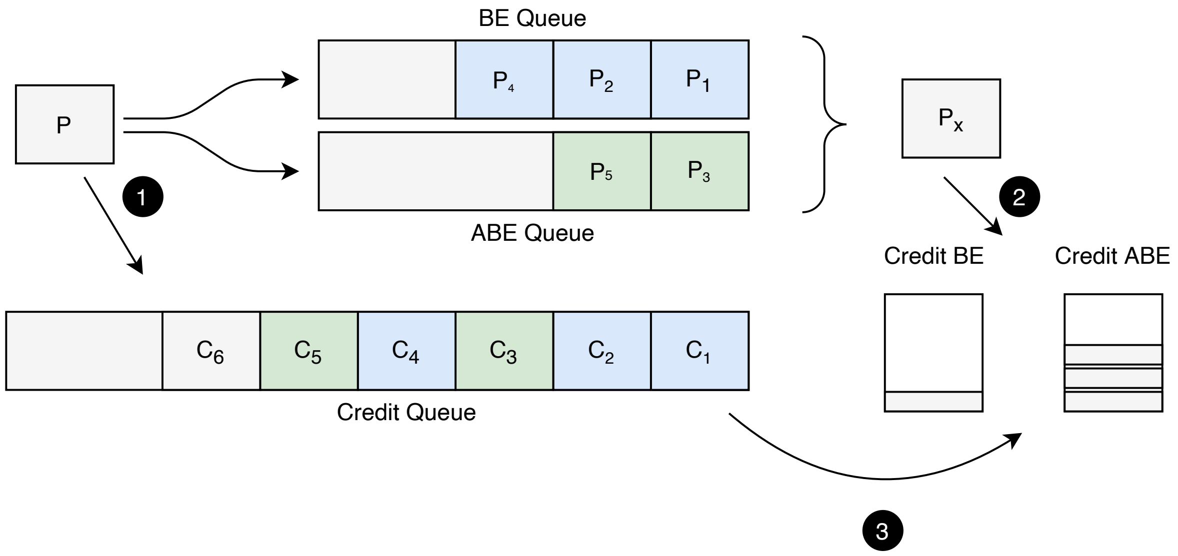
- Existing approaches
  - Are complex, e.g., require per-flow state
  - Only evaluated in simulations

Overview (1)

- **D**eadlines, **S**aved **C**redits, and **D**ecay (DSCD)
  - Novel scheduler for best effort (BE) and alternative best effort (ABE)
  - Runs locally at a bottleneck node
  - Objective: BE traffic is not degraded by ABE traffic

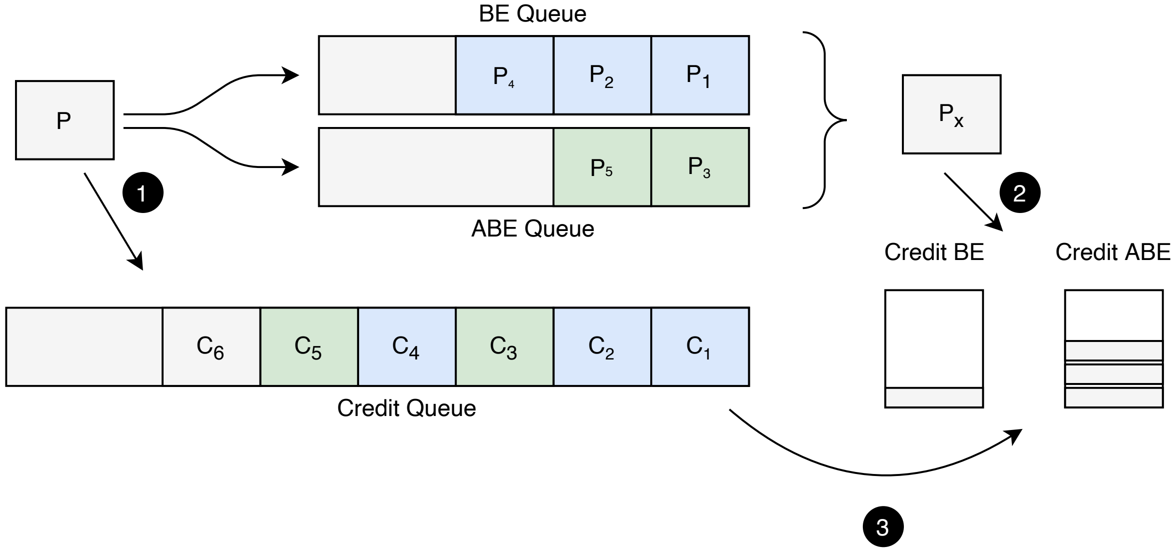


Overview (2)



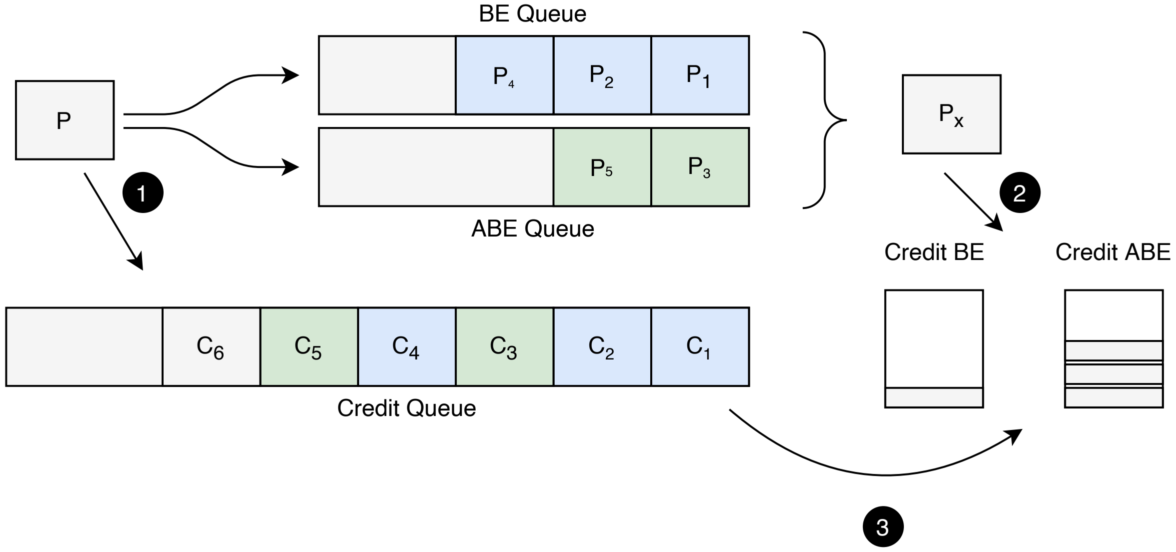
- A DSCD-enabled node consists of
  - Two FIFO queues; one for BE, one for ABE
  - A FIFO-based credit queue
  - Two class-specific counters (BE, ABE)

Overview (3)



- Arriving packets are enqueued in their class-specific FIFO queue 1
  - Credit element with packet’s size and class is added to credit queue
  - ABE packets are equipped with deadline T_d
- Packets are dequeued if the class-specific counter has enough credit 2
  - ABE packets that exceed their deadline T_d are dropped
  - Credit is consumed when packets are dequeued

Overview (4)



- Credit elements are dequeued if no class has enough credit 3
  - Credit of dropped ABE packets remains in the system
  - Can be used to serve subsequent ABE packets faster
  - FIFO order ensures service for BE packets

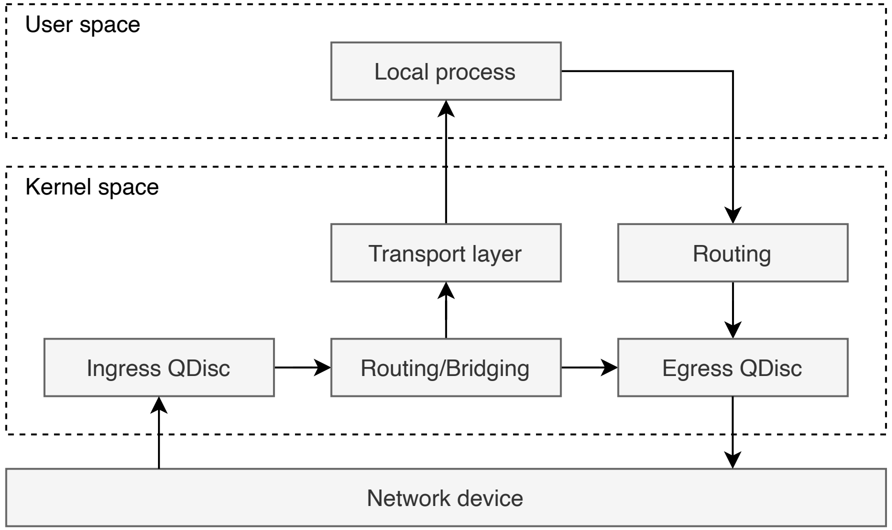
→ ABE packets receive low-delay service with higher packet loss

Credit Devaluation

- Credit should not remain in the system for infinite time
  - Users may send unnecessary data to accumulate credit
  - Credit should vanish after congestion ends

- Credit is devaluated
  - Exponentially with rate λ between two dequeue operations → $t_h = \frac{\ln(2)}{\lambda}$
  - Linearly with link bandwidth C if both queues are empty

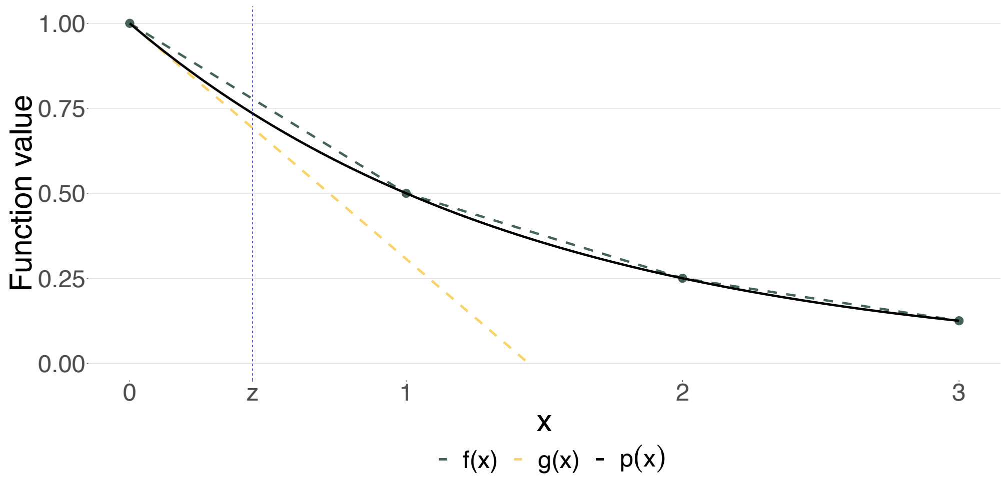
Implementation in Linux (1)

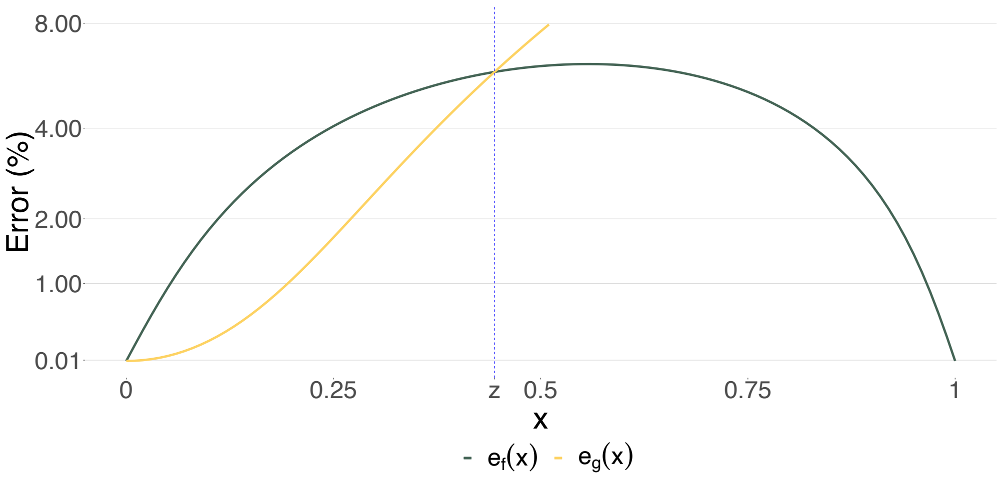


- DSCD is implemented in the Linux Network Stack
  - Implemented as egress QDisc
  - Efficient approximation of the exponential function
  - Precise bandwidth estimation algorithm

Implementation in Linux (2)

- Approximate e-function through piecewise linear function of 2^{-x}
  - $f(x) = \frac{\lfloor x \rfloor - x + 2}{2^{\lfloor x \rfloor + 1}}$, for $x > z$
  - $g(x) = 1 - \ln(2) \times x$, for small values of $0 \leq x \leq z$





Implementation in Linux (3)

Bandwidth Estimation Algorithm

- Required for linear credit devaluation after congestion period
- Leverages the moving average UTEMA[1] to weight sent bytes and transmission times
  - Considers intervals when queue is backlogged
  - Works even with low link utilization

Algorithm 4: EstimateBandwidth

Input : *Packet p*

if *backlogged* **then**
- $\Delta = t_{now} - lastRateUpdate$
- $S_B = S_B \cdot exp(-\mu \cdot \Delta) + lastPktSize$
- $S_T = S_T \cdot exp(-\mu \cdot \Delta) + (t_{now} - lastDequeue)$
- $C = S_B / S_T$
- $lastRateUpdate = t_{now}$

if $Q[ABE].len + Q[BE].len > 0$ **then**
- $backlogged = true$

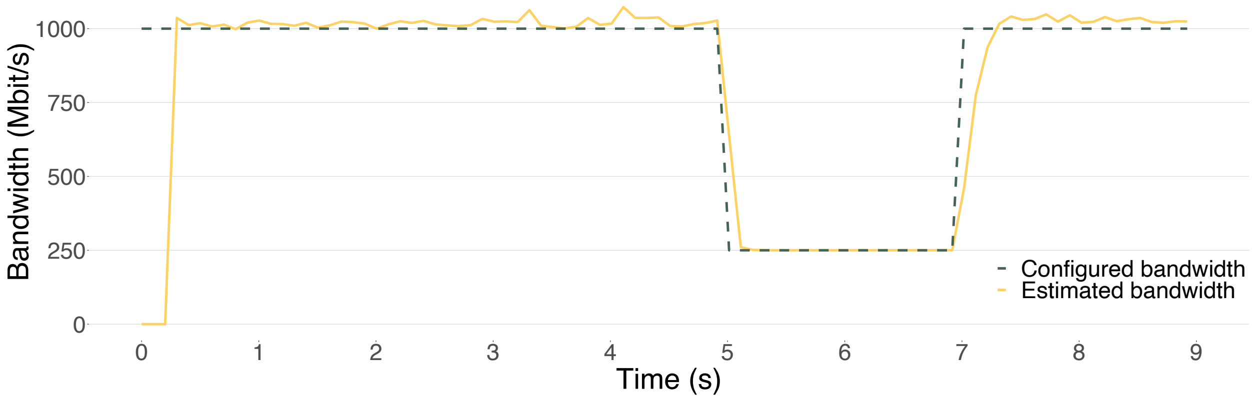
else
- $backlogged = false$

$lastDequeue = t_{now}$

$lastPktSize = p.len$

Example

- Bursty traffic with offered load $\rho = 0.5$
- Bandwidth of the link changes over time



1. M. Menth, F. Hauser, "On Moving Averages, Histograms and Time-Dependent Rates for Online Measurement", https://doi.org/10.1145/3030207.3030212

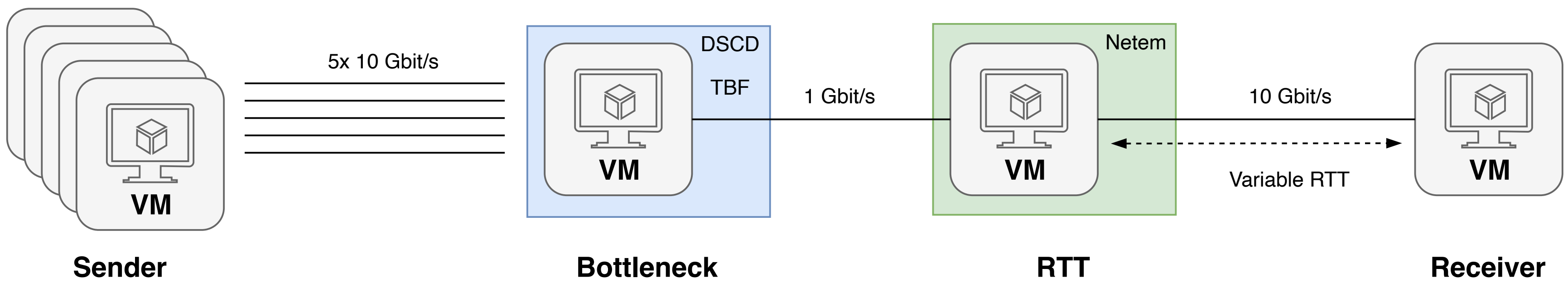
Results (1)

- Efficiency of the DSCD Implementation
  - Comparison of DSCD with standard QDiscs (FQ-Codel, FQ-Pie, SFQ, …) on a 100 Gbit/s bottleneck
  - 32 TCP flows, 50/50 BE/ABE in case of DSCD

TABLE III
TCP GOODPUT AND CPU LOAD OF VARIOUS LINUX QDISCS.

QDisc	TCP goodput (Gbit/s)	CPU load (%)
DSCD	89.08	36.27
FQ-CoDel	89.02	38.99
FQ-PIE	89.00	44.21
SFQ	89.03	38.72
pfifo	89.06	35.41

Results (2)



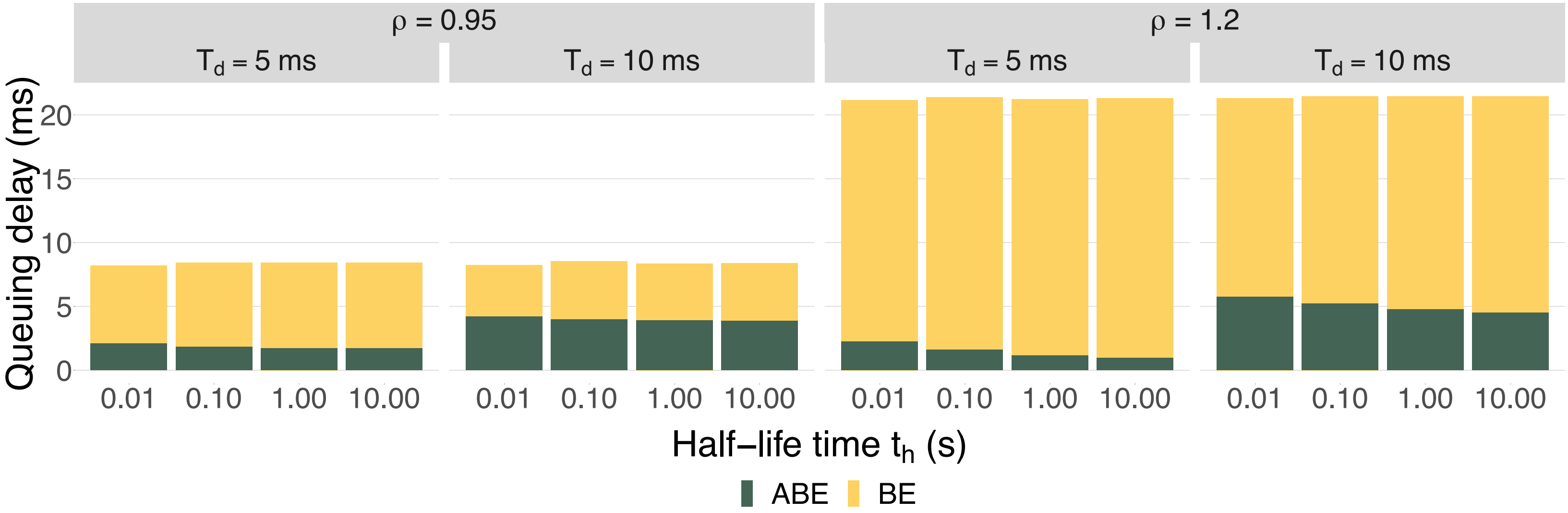
- Semi-virtualized testbed with dedicated 10 Gbit/s NICs
  - Up to five VMs send traffic through a 1 Gbit/s DSCD-based bottleneck
  - RTT VM delays packets to a configured RTT

TABLE IV
DEFAULT CONFIGURATION OF TESTBED AND DSCD ALGORITHM.

Parameter	C	RTT	B_{max}	T_d	t_h	T_q
Value	1 Gbit/s	100 ms	25 ms	10 ms	100 ms	1

Results (3)

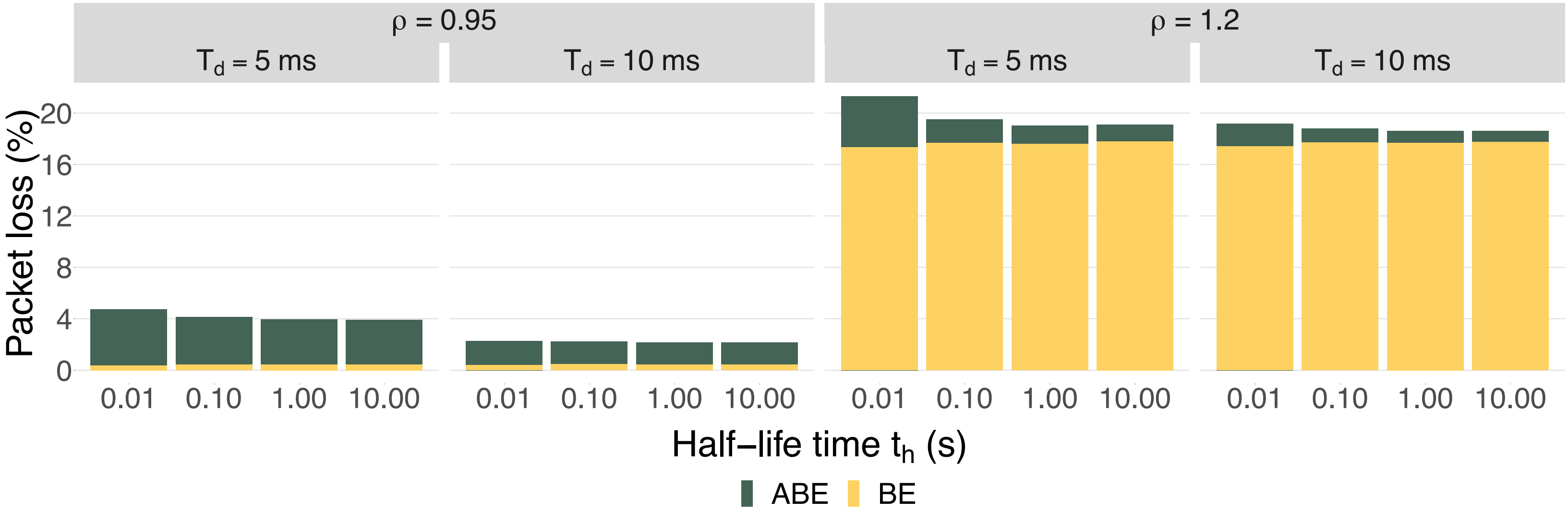
- Performance of DSCD with Non-Adaptive Traffic with Bursts
  - UDP-based traffic pattern with bursts with an offered load $\rho \in \{0.95, 1.2\}$
  - 90% of traffic is randomly labeled as BE, the other 10% as ABE



Queuing Delay

Results (4)

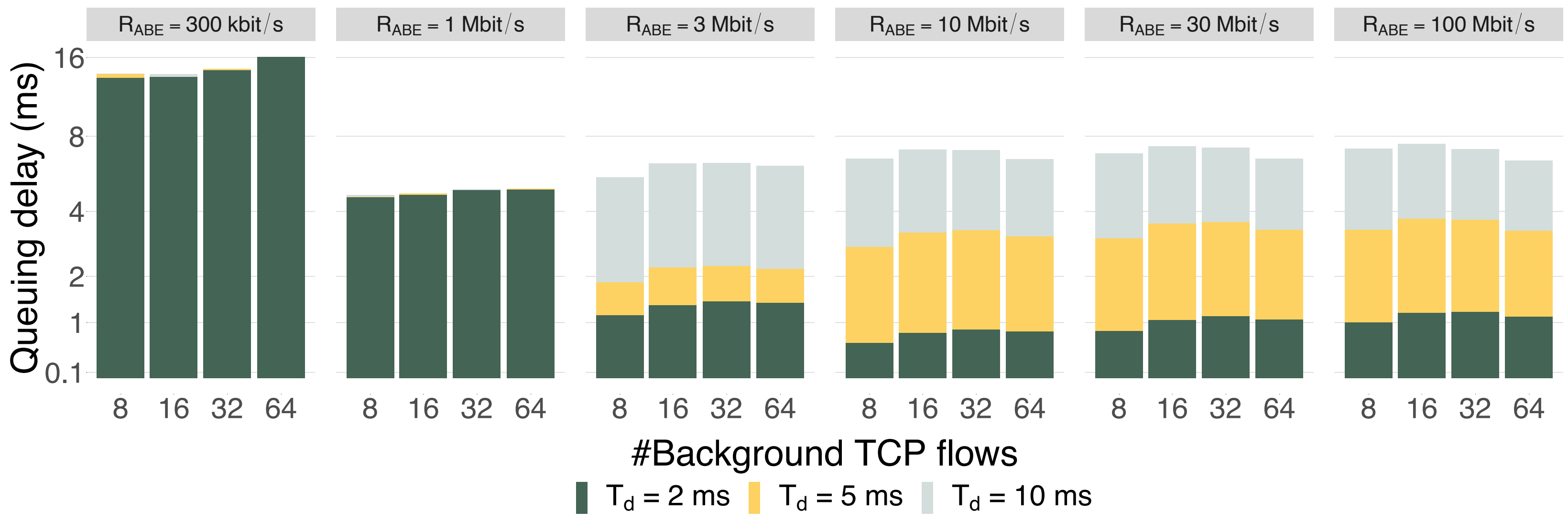
- Performance of DSCD with Non-Adaptive Traffic with Bursts
  - UDP-based traffic pattern with bursts with an offered load $p \in \{0.95, 1.2\}$
  - 90% of traffic is randomly labeled as BE, the other 10% as ABE



Packet loss

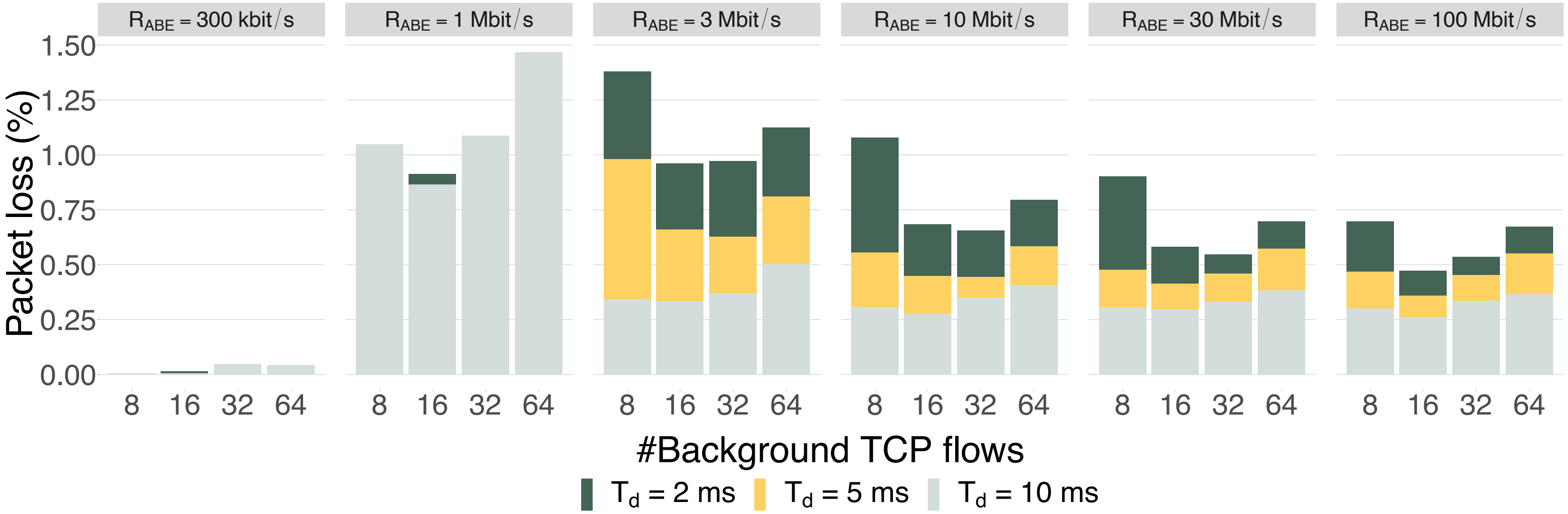
Results (5)

- Performance of DSCD with Periodic Traffic and TCP Traffic
  - Periodic UDP-based ABE traffic
  - Different number of background BE TCP flows



Queuing Delay of ABE

- Performance of DSCD with Periodic Traffic and TCP Traffic
  - Periodic UDP-based ABE traffic
  - Different number of background BE TCP flows



Packet Loss of ABE

Conclusion

- We presented DSCD
  - High performance QDisc that supports ABE and BE scheduling
  - Implemented in the Linux Network stack and achieves 100 Gbit/s
- Presented an efficient approximation of the exponential function
  - Is used for credit devaluation & bandwidth estimation
- Bandwidth estimation
  - Works even at moderate link utilizations
- Experiments show that DSCD
  - Offers low-delay service for ABE without degrading BE traffic
  - ABE traffic turns packet loss into delay advantage

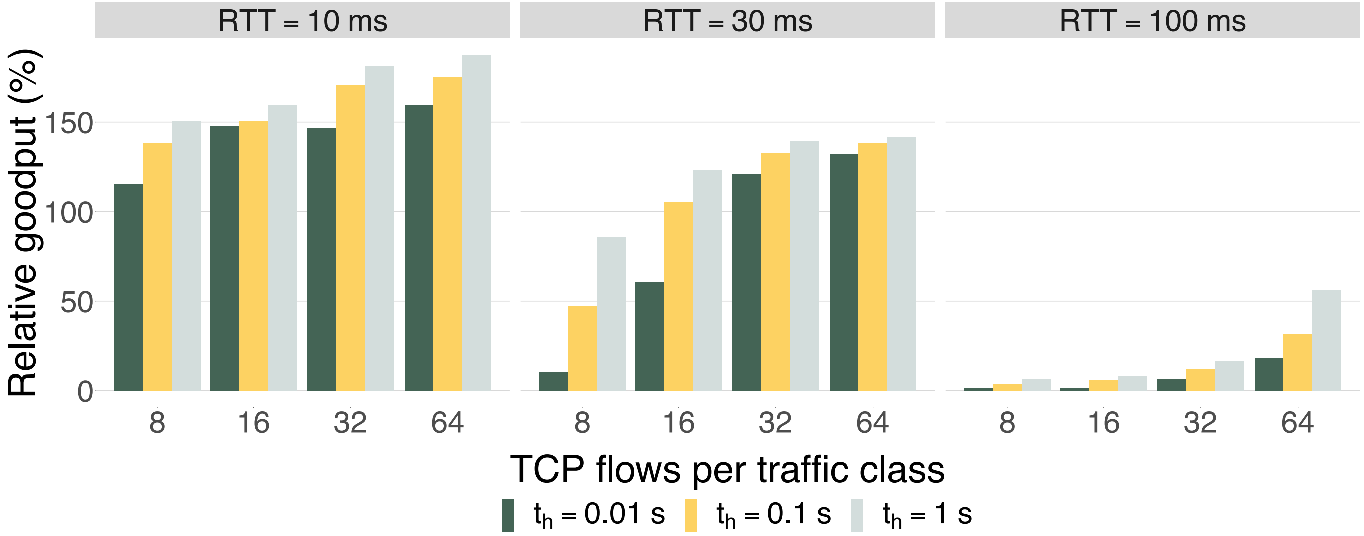
Any Questions?

Alternative Best Effort (ABE) for Service Differentiation: Trading Loss versus Delay

Published in IEEE/ACM Transactions on Networking
https://doi.org/10.1109/TNET.2022.3221553

TCP Unfairness (1)

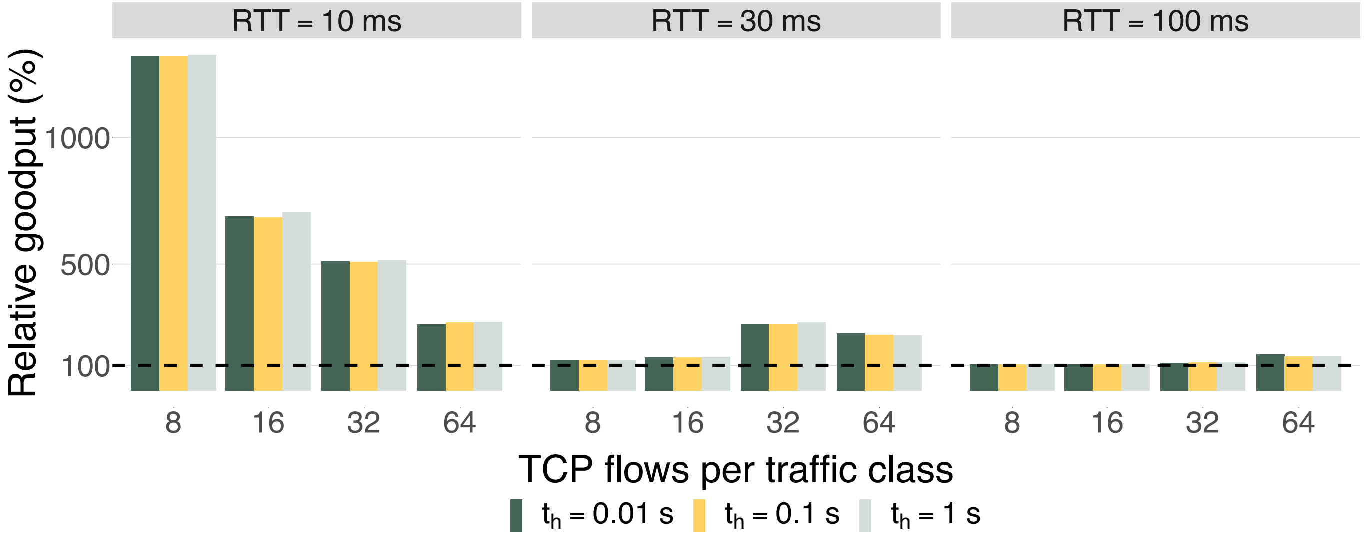
- Inter-Class Unfairness with TCP Cubic
  - Smaller queuing delay of ABE traffic may introduce unfairness with TCP
  - Half-life time can control unfairness through higher packet loss



TCP Goodput

TCP Unfairness (2)

- Inter-Class Unfairness with TCP BBR
  - Smaller queuing delay of ABE traffic may introduce unfairness with TCP
  - BBR does not react to higher packet loss



TCP Goodput