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
Agenda

► Motivation
► Overview
► Credit Devaluation
► Implementation in Linux
► Results
► Conclusion
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.**

<table>
<thead>
<tr>
<th>Application</th>
<th>E2E delay</th>
<th>Packet loss</th>
<th>Source</th>
</tr>
</thead>
<tbody>
<tr>
<td>Online gaming (FPS)</td>
<td>20 ms – 80 ms</td>
<td>≤ 5%</td>
<td>[2] [3]</td>
</tr>
<tr>
<td>Cloud gaming</td>
<td>≤ 50 ms</td>
<td>≤ 5%</td>
<td>[4] [5] [6]</td>
</tr>
<tr>
<td>Voice over IP (VoIP)</td>
<td>≤ 150 ms</td>
<td>1% – 3%</td>
<td>[7]</td>
</tr>
</tbody>
</table>

- Has been already discussed in IETF
  - draft-you-tsvwg-latency-loss-tradeoff (2016)

- Existing approaches
  - Are complex, e.g., require per-flow state
  - Only evaluated in simulations
**Deadlines, Saved Credits, and Decay (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
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)
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
Credit elements are dequeued if no class has enough credit
- 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 $\lambda$ between two dequeue operations $\rightarrow t_h = \frac{\ln(2)}{\lambda}$
  - Linearly with link bandwidth $C$ if both queues are empty
DSCD is implemented in the Linux Network Stack
- Implemented as egress QDisc
- Efficient approximation of the exponential function
- Precise bandwidth estimation algorithm
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 $0 \leq x \leq z$

![Graph showing function values and error percentages for $f(x)$, $g(x)$, $p(x)$, $e_f(x)$, and $e_g(x)$ over a range of $x$ values from 0 to 3.](image-url)
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

Example

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

### Algorithm 4: EstimateBandwidth

**Input**: Packet \( p \)

1. if backlogged then
2. \( \Delta = t_{\text{now}} - \text{lastRateUpdate} \)
3. \( S_B = S_B \cdot \exp(-\mu \cdot \Delta) + \text{lastPktSize} \)
4. \( S_T = S_T \cdot \exp(-\mu \cdot \Delta) + (t_{\text{now}} - \text{lastDequeue}) \)
5. \( C = S_B / S_T \)
6. \( \text{lastRateUpdate} = t_{\text{now}} \)
7. if \( Q[ABE].\text{len} + Q[BE].\text{len} > 0 \) then
8. \( \text{backlogged} = \text{true} \)
9. else
10. \( \text{backlogged} = \text{false} \)
11. \( \text{lastDequeue} = t_{\text{now}} \)
12. \( \text{lastPktSize} = p.\text{len} \)

---

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>
<thead>
<tr>
<th>QDisc</th>
<th>TCP goodput (Gbit/s)</th>
<th>CPU load (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>DSCD</td>
<td>89.08</td>
<td>36.27</td>
</tr>
<tr>
<td>FQ-CoDel</td>
<td>89.02</td>
<td>38.99</td>
</tr>
<tr>
<td>FQ-PIE</td>
<td>89.00</td>
<td>44.21</td>
</tr>
<tr>
<td>SFQ</td>
<td>89.03</td>
<td>38.72</td>
</tr>
<tr>
<td>pfifo</td>
<td>89.06</td>
<td>35.41</td>
</tr>
</tbody>
</table>

Alternative Best Effort (ABE) for Service Differentiation: Trading Loss versus Delay | IETF 116
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>
<thead>
<tr>
<th>Parameter</th>
<th>$C$</th>
<th>RTT</th>
<th>$B_{max}$</th>
<th>$T_d$</th>
<th>$t_h$</th>
<th>$T_q$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Value</td>
<td>1 Gbit/s</td>
<td>100 ms</td>
<td>25 ms</td>
<td>10 ms</td>
<td>100 ms</td>
<td>1</td>
</tr>
</tbody>
</table>
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
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

![Performance of DSCD with Non-Adaptive Traffic with Bursts](image)
Performance of DSCD with Periodic Traffic and TCP Traffic

- Periodic UDP-based ABE traffic
- Different number of background BE TCP flows

![Diagram showing the queuing delay of ABE with varying background TCP flows and ABE rates.](image)

Queuing delay (ms)

- **$R_{ABE} = 300$ kbit/s**
- **$R_{ABE} = 1$ Mbit/s**
- **$R_{ABE} = 3$ Mbit/s**
- **$R_{ABE} = 10$ Mbit/s**
- **$R_{ABE} = 30$ Mbit/s**
- **$R_{ABE} = 100$ Mbit/s**

---

**Results (5)**
Performance of DSCD with Periodic Traffic and TCP Traffic
- Periodic UDP-based ABE traffic
- Different number of background BE TCP flows
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
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

![Graph showing relative goodput (%) vs. TCP flows per traffic class with different RTTs and half-life times.](image)
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 flows per traffic class:
- 8
- 16
- 32
- 64

TCP Goodput:
- \( t_h = 0.01 \) s
- \( t_h = 0.1 \) s
- \( t_h = 1 \) s