Policy-oriented AQM Steering

Roland Bless, Mario Hock, Martina Zitterbart
published @Networking 2018, Zürich, Switzerland, May 14th-16th
Newer AQMs: CoDel, PIE, GSP
- Try to be parameter-less (in normal operation)
- Work reasonably well over wide range of traffic situations

One crucial parameter remaining: *Target “delay setpoint”*
- Typically set to default value, e.g., 5 ms

But: Achievable performance depends on traffic situation
- Especially: number of flows and their RTTs

Possible outcome: Unnecessarily large delay or underutilization
Motivation

- Newer AQMs: CoDel, PIE, GSP
  - Try to be parameter-less (in normal operation)
  - Work reasonably well over wide range of traffic situations
- One crucial parameter remaining: Target “delay setpoint”
  - typically set to default value, e.g., 5 ms
- But: Achievable performance depends on traffic situation
  - especially: number of flows and their RTTs
- Possible outcome: Unnecessarily large delay or underutilization
Motivation

- Newer AQMs: CoDel, PIE, GSP
  - Try to be parameter-less (in normal operation)
  - Work reasonably well over wide range of traffic situations

- One crucial parameter remaining: Target “delay setpoint”
  → typically set to default value, e.g., 5 ms

- But: Achievable performance depends on traffic situation
  → especially: number of flows and their RTTs

- Possible outcome: Unnecessarily large delay or underutilization
Motivation

- Newer AQMs: CoDel, PIE, GSP
  - Try to be parameter-less (in normal operation)
  - Work reasonably well over wide range of traffic situations

- One crucial parameter remaining: Target "delay setpoint"
  → typically set to default value, e.g., 5 ms

- But: Achievable performance depends on traffic situation
  → especially: number of flows and their RTTs

- Possible outcome: Unnecessarily large delay or underutilization
AQM Steering: Overview

- **Goal:** Improve AQM performance

- **AQM Steering:** External control loop around existing AQM
  → Adjust "target delay setpoint" to current traffic situation
AQM Steering: Overview

- **Goal:** Improve AQM performance
- **AQM Steering:** External control loop around existing AQM
  → Adjust “target delay setpoint” to current traffic situation
Target Delay Setpoints

- What is a “target delay setpoint”? 
  - Newer AQMs try to keep queuing delay around specific target, e.g., 5 ms.
  - Visualization at bottleneck buffer:

Example: “Global Synchronization Protection” (GSP)  
  ➔ draft-lauten-aqm-gsp
  - Drop packet(s) if target delay setpoint is exceeded
  - Dynamically find suitable dropping rate
Target Delay Setpoints

- What is a “target delay setpoint”?
  - Newer AQMs try to keep queuing delay around specific target, e.g., 5 ms.
  - Visualization at bottleneck buffer:

  ![Diagram showing target delay setpoint]

  - Example: “Global Synchronization Protection” (GSP)
    → draft-lauten-aqm-gsp
    - Drop packet(s) if target delay setpoint is exceeded
    - Dynamically find suitable dropping rate
AQM Steering: Basic Idea

- AQM Steering *adjusts setpoint* to current traffic characteristics

---

**Default setpoint**

→ Underutilization

**Optimal setpoint**

- Increase setpoint if necessary to achieve *desired* throughput
  - **Trade-off**: Higher throughput for higher delay
  - Limitations caused by used *congestion control*, e.g., *CUBIC TCP*
AQM Steering: Basic Idea

- AQM Steering adjusts setpoint to current traffic characteristics

Default setpoint
→ Too high

- Decrease setpoint if possible without sacrificing throughput
  - Default setpoint: Throughput achieved at a too high “price”
  → Adapted setpoint: Same throughput, lower delay!
AQM Steering: Policies

- Trade-off: Throughput vs. delay
  - What is your priority?
  - With fixed setpoint: Not much control!

- Easy to grasp policies
  - \((u_{\text{low}}, \text{target}_{\text{max}}), \) optionally: \(u_{\text{target}}\)
    - Queuing Delay \(\leq \text{target}_{\text{max}}\) (Upper delay bound)
    - Link utilization \(\geq u_{\text{low}}\) (Lower utilization target)
    - Link utilization \(\leq u_{\text{target}}\) (Upper utilization target)

- Meaningful parameters
  - \(\text{target}_{\text{max}}\): "How much delay am I willing to trade for high throughput?"
  - \(u_{\text{low}}\): "At which throughput am I not willing to trade delay anymore?"
  - \(u_{\text{target}}\): "How much throughput am I willing to trade for ultra low delay?"

- Find best throughput vs. delay trade-off within these policy bounds
AQM Steering: Policies

- Trade-off: Throughput vs. delay
  - What is your priority?
  - With fixed setpoint: Not much control!

- Easy to grasp policies
  - \( \langle u_{low}, \text{target}_{max} \rangle \), optionally: \( u_{target} \)
    - Queuing Delay \( \leq \text{target}_{max} \) (Upper delay bound)
    - Link utilization \( \geq u_{low} \) (Lower utilization target)
    - Link utilization \( \leq u_{target} \) (Upper utilization target)

- Meaningful parameters
  - \( \text{target}_{max} \): “How much delay am I willing to trade for high throughput?”
  - \( u_{low} \): “At which throughput am I not willing to trade delay anymore?”
  - \( u_{target} \): “How much throughput am I willing to trade for ultra low delay?”

- Find best throughput vs. delay trade-off within these policy bounds
AQM Steering: Challenges

1. When should the setpoint be assessed / adjusted?
2. To which value should the setpoint be changed?
3. How to achieve ultra low latencies with existing AQMs?
Challenge 1: When to Assess / Adjust Setpoint?
Challenge 1: When to Assess / Adjust Setpoint?

- AQM Steering observes interplay: AQM ↔ congestion control

No bottleneck

- No packets dropped: Link is no bottleneck
  → Increasing setpoint will not increase throughput!

Not converged, yet

- Queuing delay persistently above setpoint: AQM is still adjusting!
Challenge 1: When to Assess / Adjust Setpoint?

- AQM Steering observes interplay: AQM ↔ congestion control

No bottleneck

- No packets dropped: Link is no bottleneck
  → Increasing setpoint will not increase throughput!

Not converged, yet

- Queuing delay persistently above setpoint: AQM is still adjusting!
Challenge 1: When to Assess / Adjust Setpoint?

- AQM Steering observes interplay: AQM ↔ congestion control

No bottleneck

No packets dropped: Link is no bottleneck
   → Increasing setpoint will not increase throughput!

Not converged, yet

Queuing delay persistently above setpoint: AQM is still adjusting!
Challenge 1: When to Assess / Adjust Setpoint?

- AQM Steering observes interplay: AQM ↔ congestion control

<table>
<thead>
<tr>
<th>Amount of Inflight Data</th>
<th>Time</th>
<th>Buffer Link</th>
</tr>
</thead>
<tbody>
<tr>
<td>No bottleneck</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- No packets dropped: Link is no bottleneck
  → Increasing setpoint will not increase throughput!

- Queuing delay persistently above setpoint: AQM is still adjusting!

Converged, finally
Challenge 2: Determine New Setpoint
Challenge 2: Determine New Setpoint

- Different strategies required for increase and decrease

**Decrease**

- **Reduction** by $\Delta d$ maintains high throughput
  - $\Delta d$ can be measured within the AQM, but is noisy
    → Smoothing, averaging, variance

**Increase**

- **Increase** depends on $bdp/RTT$ (not known by AQM)
  → Appropriate amount cannot be determined directly → probing required!
Evaluation
Evaluation

- Implementation based on the AQM “GSP”
  - GSP-AS (“GSP with AQM Steering”)
    - DPDK-based prototyping switch
    - Intel’s “Data Plane Development Kit” (for high speed network functions)

- Testbed:

- Comparison
  - GSP-AS \( u_{low} = 99\% \), \( \text{target}_{max} = 30\ ms \)
  - CoDel, GSP, (setpoint = 2.5 ms)
  - Taildrop (small buffer (2.5 ms) / large buffer (30 ms))
Evaluation
Selected Experiments

- Experiment 1: Proof general idea: Steady state, long lived flows
  - GSP-AS is able to trade off throughput vs. delay, according to the policy
  - Regular AQMs: Fixed delay target, performance depends on traffic situation
  - Tail drop: High throughput or low delay — depends on buffer size

- Experiment 2: Transition behavior
Evaluation

- Experiment 1: Steady state, long lived flows

- Lower number of flows → low loss de-synchronization
  - GSP-AS: High throughput, increased delay
  - AQMs: Underutilization (fixed setpoint too low)
  - Tail drop (small / large): Low throughput / high delay
Evaluation

- Experiment 1: Steady state, long lived flows

- Lower number of flows → low loss de-synchronization
  - GSP-AS: High throughput, increased delay
  - AQMs: Underutilization (fixed setpoint too low)
  - Tail drop (small / large): Low throughput / high delay
### Evaluation

- **Experiment 1: Steady state, long lived flows**

<table>
<thead>
<tr>
<th>Number of Flows</th>
<th>TD 2.5ms</th>
<th>CoDel</th>
<th>GSP</th>
<th>GSP-AS</th>
<th>TD 30ms</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>36</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Throughput [Gbit/s]**

<table>
<thead>
<tr>
<th>Queuing Delay [ms]</th>
</tr>
</thead>
<tbody>
<tr>
<td>TD 2.5ms</td>
</tr>
<tr>
<td>2</td>
</tr>
<tr>
<td>9</td>
</tr>
<tr>
<td>36</td>
</tr>
</tbody>
</table>

**Queuing Delay [ms]**

- "Right" number of flows → **reasonable loss de-synchronization**
  - AQMs: Given setpoint suitable for this traffic situation
  - GSP-AS: adjusts to similar values (setpoint, delay, throughput)
  - Tail drop (small): Still *low* loss de-synchronization!
Evaluation

- **Experiment 1: Steady state, long lived flows**

  ![Graph showing throughput and queuing delay for different number of flows and AQM types](image)

  - **Higher number of flows → high loss de-synchronization**
    - GSP-AS: High throughput, very low delay
    - AQMs: Unnecessarily large delay (fixed setpoint too high)
    - Tail drop (small): Still underutilization
Evaluation

- Experiment 1: Steady state, long lived flows

Higher number of flows → high loss de-synchronization

- GSP-AS: High throughput, very low delay
- AQMs: Unnecessarily large delay (fixed setpoint too high)
- Tail drop (small): Still underutilization
Evaluation

- Experiment 2: Transition behavior
  - # Flows changed: 2 → 36 → 2

- Two flows:
  - Setpoint increased to ≈ 11 ms
  - Necessary to keep throughput policy
Evaluation

- Experiment 2: Transition behavior
  - # Flows changed: 2 → 36 → 2

- Sudden increase in # flows (2 → 36):
  - Setpoint smoothly adjusts to new traffic situation
  - Smoothing prevents overreactions
Evaluation

- Experiment 2: Transition behavior
  - # Flows changed: 2 → 36 → 2

36 flows:
- Low setpoint sufficient to keep throughput policy
  → Notice: Only small fluctuations of in-flight data
Evaluation

- Experiment 2: Transition behavior
  - # Flows changed: 2 → 36 → 2

- 34 flows suddenly stop (36 → 2):
  - Temporary underutilization!
  - Congestion control needs some time to reclaim free bandwidth
Evaluation

- Experiment 2: Transition behavior
  - \# Flows changed: 2 → 36 → 2

- Utilization target not fulfilled (36 → 2):
  - AQM Steering adjusts to new traffic situation
  - Step-wise increment of setpoint
Evaluation

- Experiment 2: Transition behavior
  - 
  - # Flows changed: 2 → 36 → 2

![Graph showing queuing delay over time]

- Two flows:
  - Necessary setpoint regained ≈ 10 ms
  - Notice: In-flight data fluctuates between empty buffer and setpoint
Conclusion

- AQM Steering: Improvement of (existing) AQMs
  - Avoid unnecessarily large delays
  - Achieve high utilization
  → Find best trade-off: Throughput vs. delay under given policy

- External control loop around existing AQM

- Evaluation in physical high speed testbed (10 Gbit/s bottleneck)
  - GSP-AS is able to trade off throughput vs. delay, according to the policy
  - Adapts to changing traffic
  → Improves performance of existing AQMs

- Paper:
Thank you very much for your attention!

Questions?

Additional Slides
Challenge 3: Ultra low latencies

- Approach: Keep link utilization below 100% \( U_{\text{target}} \)

- Challenge
  - Targeted AQMs work on queuing delay
  - Cannot react before a queue builds up

- Solution: Virtual Queues
  - Simulate virtual egress rate \( r_{\text{virt}} < r_{\text{phy}} \)
  - Calculate queue size / delay that \emph{would} build up

\[
\dot{q}_{\text{real}} = \begin{cases} 
    r_{\text{in}} - r_{\text{phy}} & \text{if } q > 0, \\
    (r_{\text{in}} - r_{\text{phy}})^+ & \text{if } q = 0
\end{cases} \quad \dot{q}_{\text{virtual}} = \begin{cases} 
    r_{\text{in}} - r_{\text{virt}} & \text{if } q > 0, \\
    (r_{\text{in}} - r_{\text{virt}})^+ & \text{if } q = 0
\end{cases}
\]

\[ \rightarrow \] Do \emph{not} shape traffic (since actual queues would build up)!

- AQM is seamlessly switched between virtual queue and physical queue
Figure: Steady state, long lived flows + short lived flows

- Much higher delay required to keep high throughput
  - Reduced throughput for fixed AQMs
  - GSP-AS can adapt
- Small tail-drop buffer problematic!
Figure: Interplay of physical and virtual queue ($u_{target} = 95\%$)

- **Low number of flows**
  - Physical queue required to fulfill policy ($u_{low} = 0.94$)

- **Higher number of flows**
  - No physical queue necessary to fulfill lower delay bound ($u_{low}$)
  - Virtual queue required to fulfill upper delay bound ($u_{target}$)

→ (Sufficiently) high throughput, *no queuing delay* at all.
Additional Slides
Evaluation: Further Experiments

Figure: Transition behavior

- Adjusting to sudden changes in traffic
- Traffic changed during adaptation
  - GSP-AS control loop does not destabilize
  - Quick increase, slow decrease
Discussion

- AQM Steering cannot always achieve $u_{low}$!
  - If link is no bottleneck $u_{low}$ is irrelevant
  - $target_{max}$ can be lower than necessary

- When converged, throughput comparable with large tail-drop buffer
  - Throughput $\geq \min\left(u_{low}, thr_{taildrop}(target_{max})\right)$
    - $thr_{taildrop}(\ldots)$: throughput with tail-drop buffer of given size

- Reclaim of free bandwidth is different!
  - When a flow disappears, total in-flight data is suddenly reduced (by its $CWnd$)
  - Sudden drop in delay (large queues) or link utilization (small queues)
    - Can conceptually not be compensated by AQM Steering