Resource Constraints

Can you mute noisy neighbors?

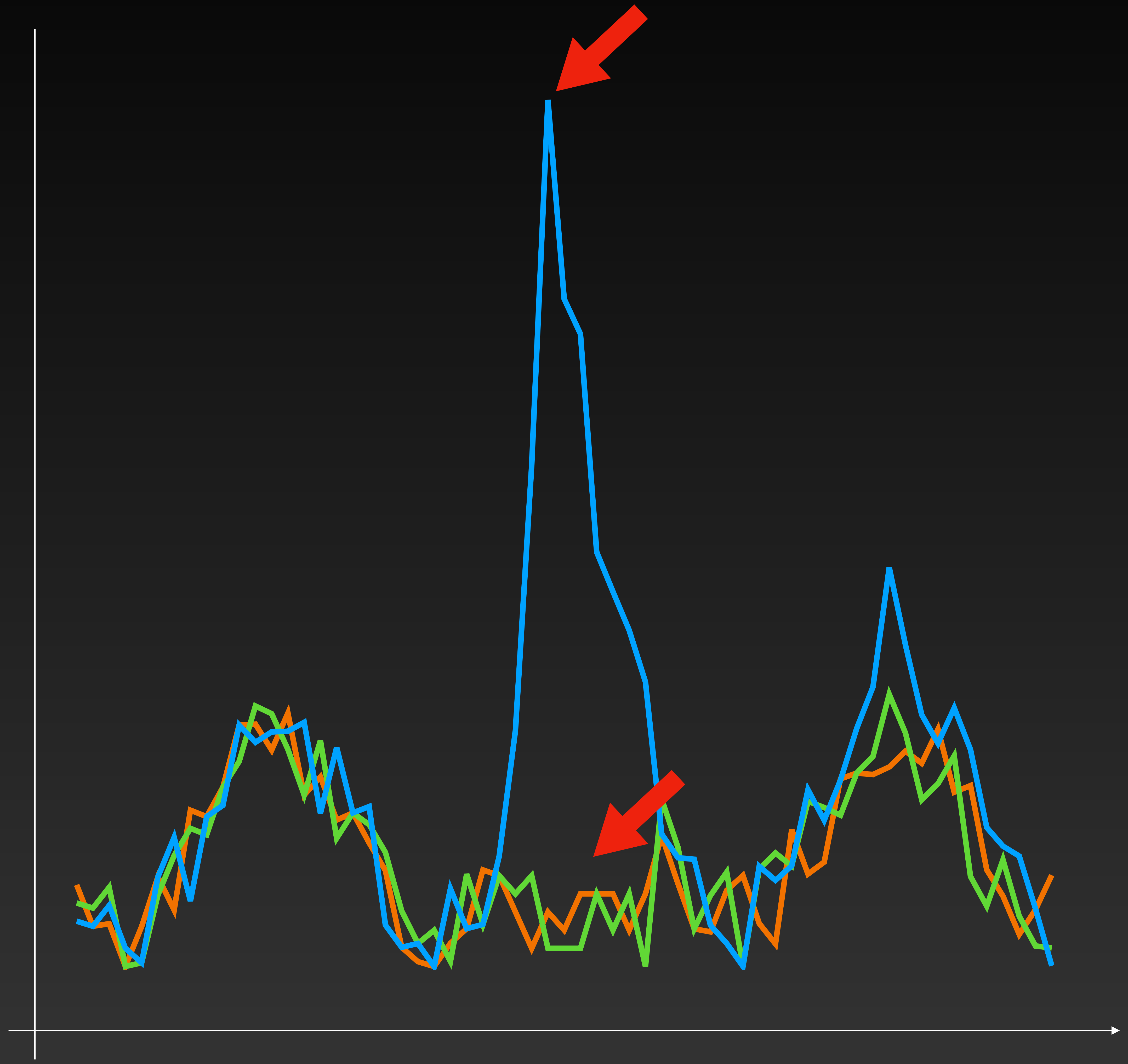
ATS Fall 2021 Summit

Masaori Koshiba (masaori@apache.org)

Noisy Neighbours

Spike!

- Request Per Second
- TLS Handshakes
- Active Connections
- Event System Usage
- Memory Usage
- Disk IO
- Network IO

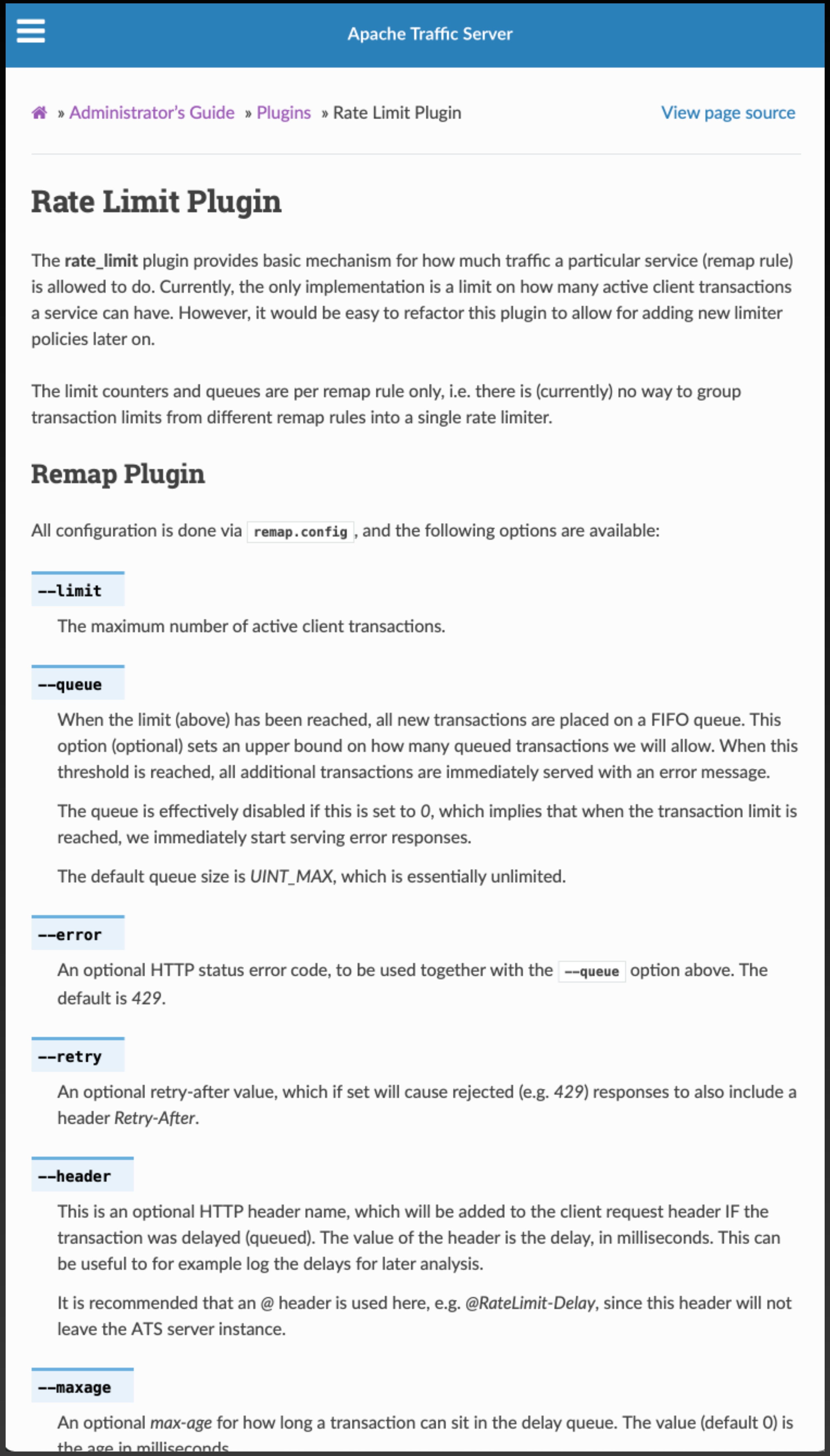


Rate Limit Plugin (#7623, #8021)

- Token Bucket

- Per remap or SNI

- #7623 New rate_limit plugin for simple resource limitations
- #8021 rate_limit: Add a global hook to rate limit concurrent connections based on SNI



Apache Traffic Server

» Administrator's Guide » Plugins » Rate Limit Plugin   View page source

Rate Limit Plugin

The **rate_limit** plugin provides basic mechanism for how much traffic a particular service (remap rule) is allowed to do. Currently, the only implementation is a limit on how many active client transactions a service can have. However, it would be easy to refactor this plugin to allow for adding new limiter policies later on.

The limit counters and queues are per remap rule only, i.e. there is (currently) no way to group transaction limits from different remap rules into a single rate limiter.

Remap Plugin

All configuration is done via `remap.config`, and the following options are available:

`--limit`

The maximum number of active client transactions.

`--queue`

When the limit (above) has been reached, all new transactions are placed on a FIFO queue. This option (optional) sets an upper bound on how many queued transactions we will allow. When this threshold is reached, all additional transactions are immediately served with an error message.

The queue is effectively disabled if this is set to *0*, which implies that when the transaction limit is reached, we immediately start serving error responses.

The default queue size is *UINT_MAX*, which is essentially unlimited.

`--error`

An optional HTTP status error code, to be used together with the `--queue` option above. The default is *429*.

`--retry`

An optional retry-after value, which if set will cause rejected (e.g. *429*) responses to also include a header *Retry-After*.

`--header`

This is an optional HTTP header name, which will be added to the client request header IF the transaction was delayed (queued). The value of the header is the delay, in milliseconds. This can be useful to for example log the delays for later analysis.

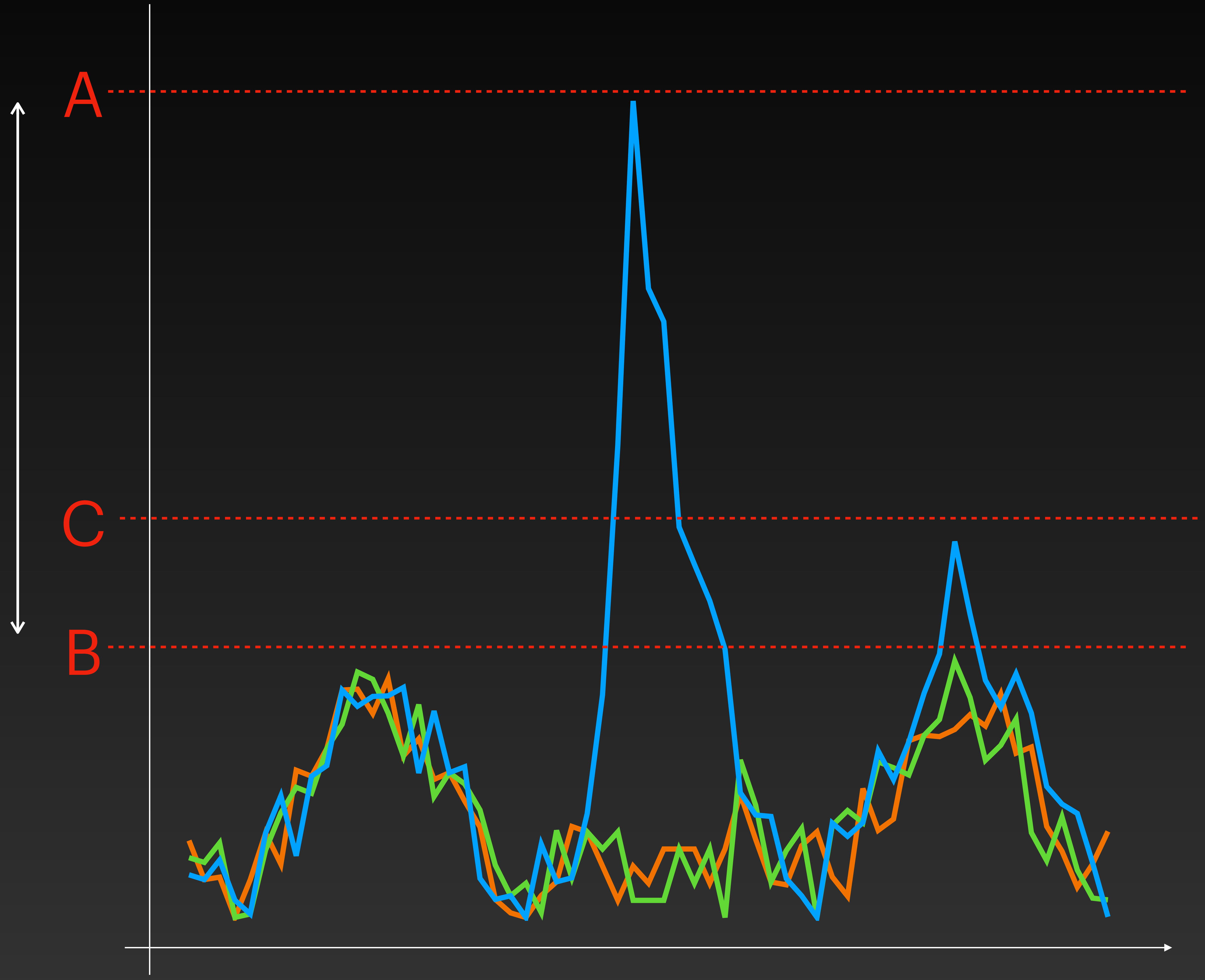
It is recommended that an @ header is used here, e.g. *@RateLimit-Delay*, since this header will not leave the ATS server instance.

`--maxage`

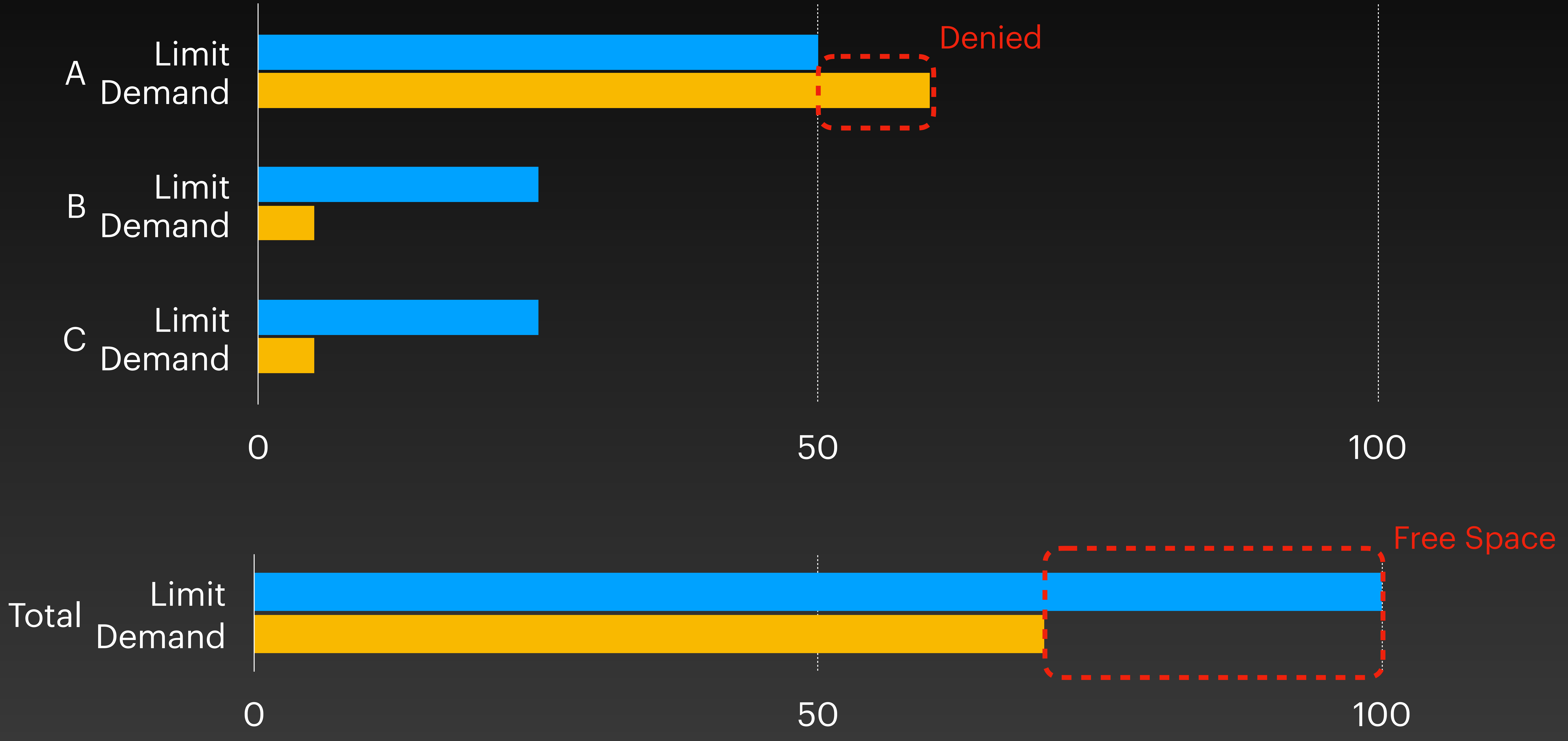
An optional *max-age* for how long a transaction can sit in the delay queue. The value (default 0) is the age in milliseconds.

Tune in the dark

What is "good" value?



Total Resource Usage

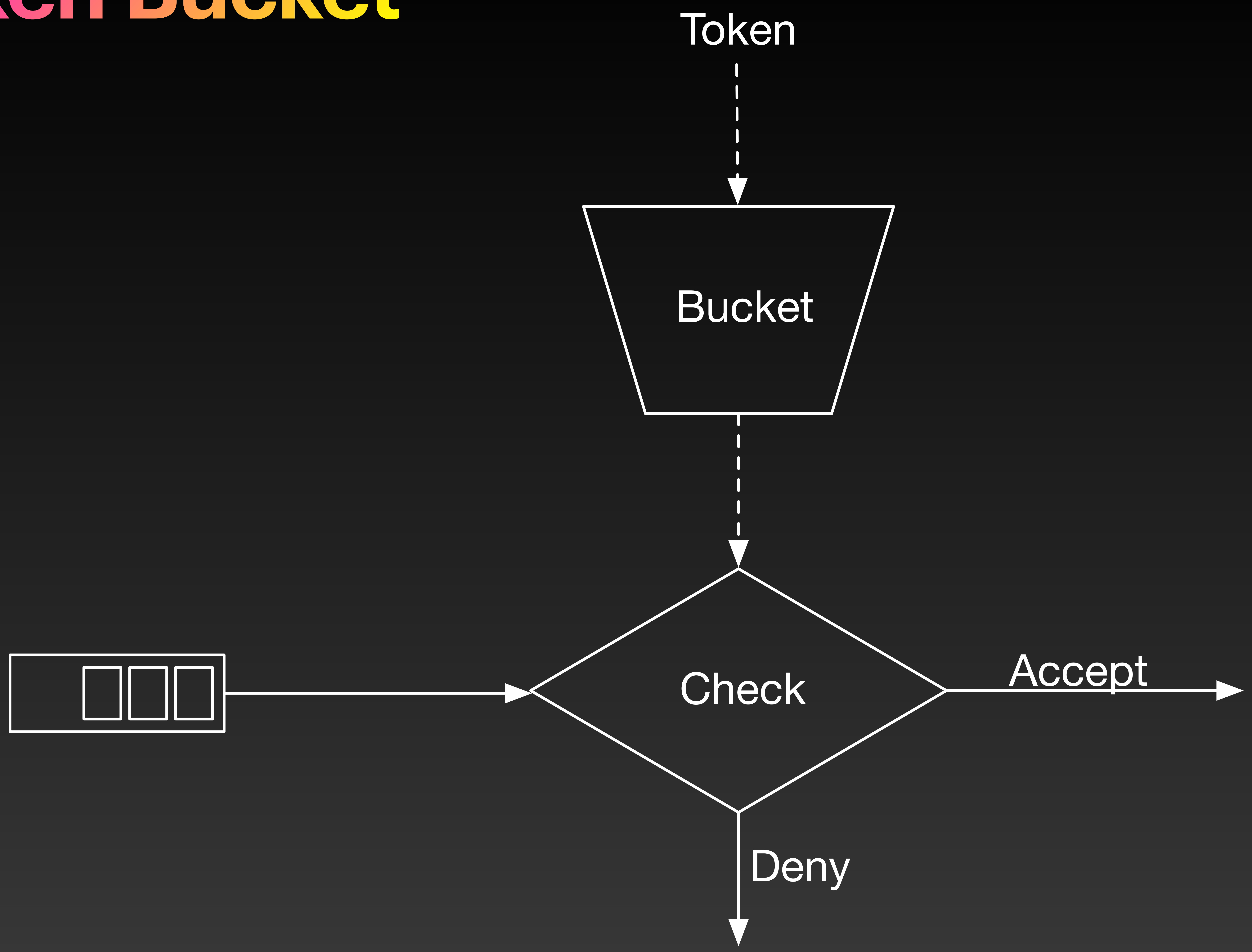


Goal of this project

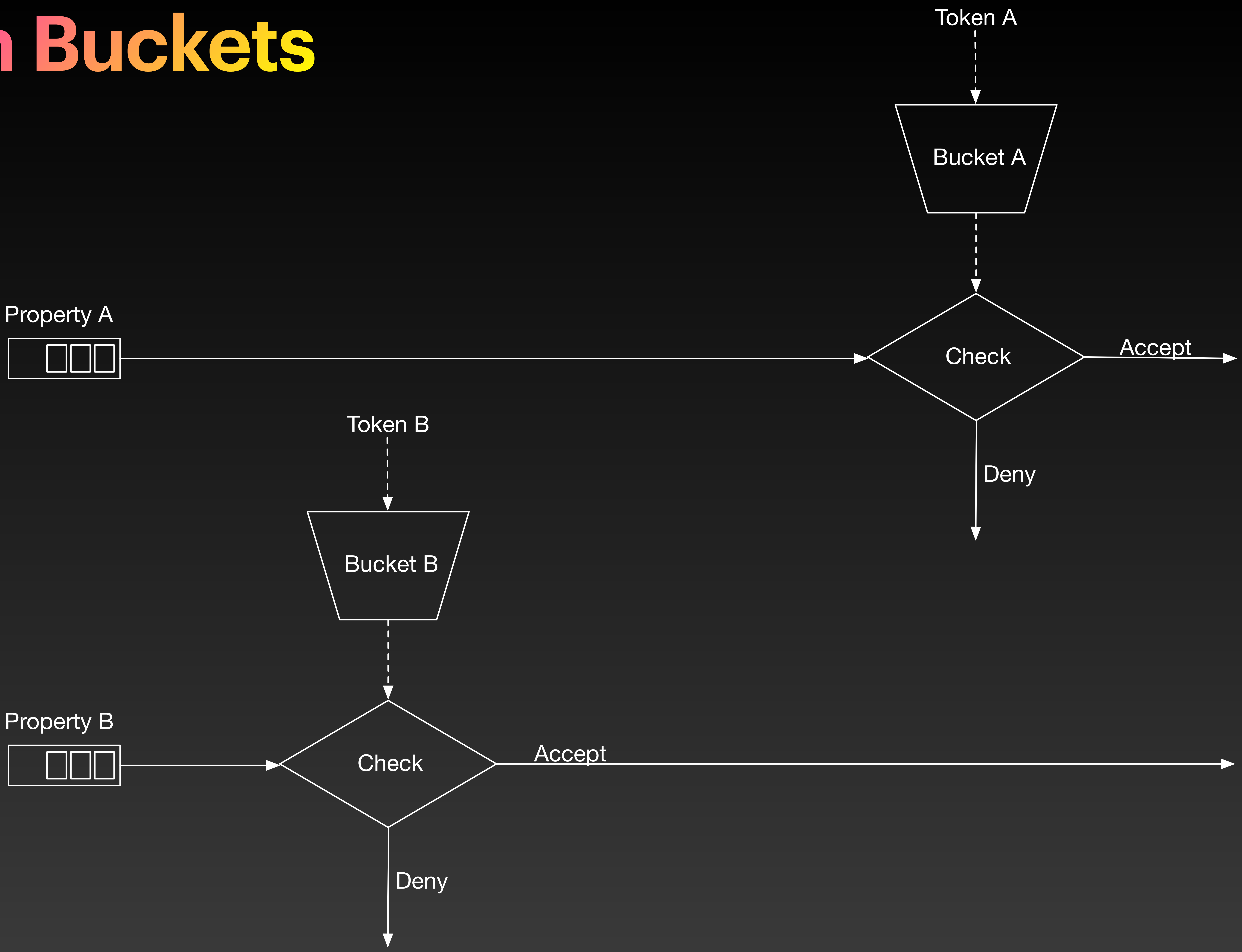
1. Operator friendly
2. Efficiency of Resource Usage
3. Support many metrics

Algorithm Overview

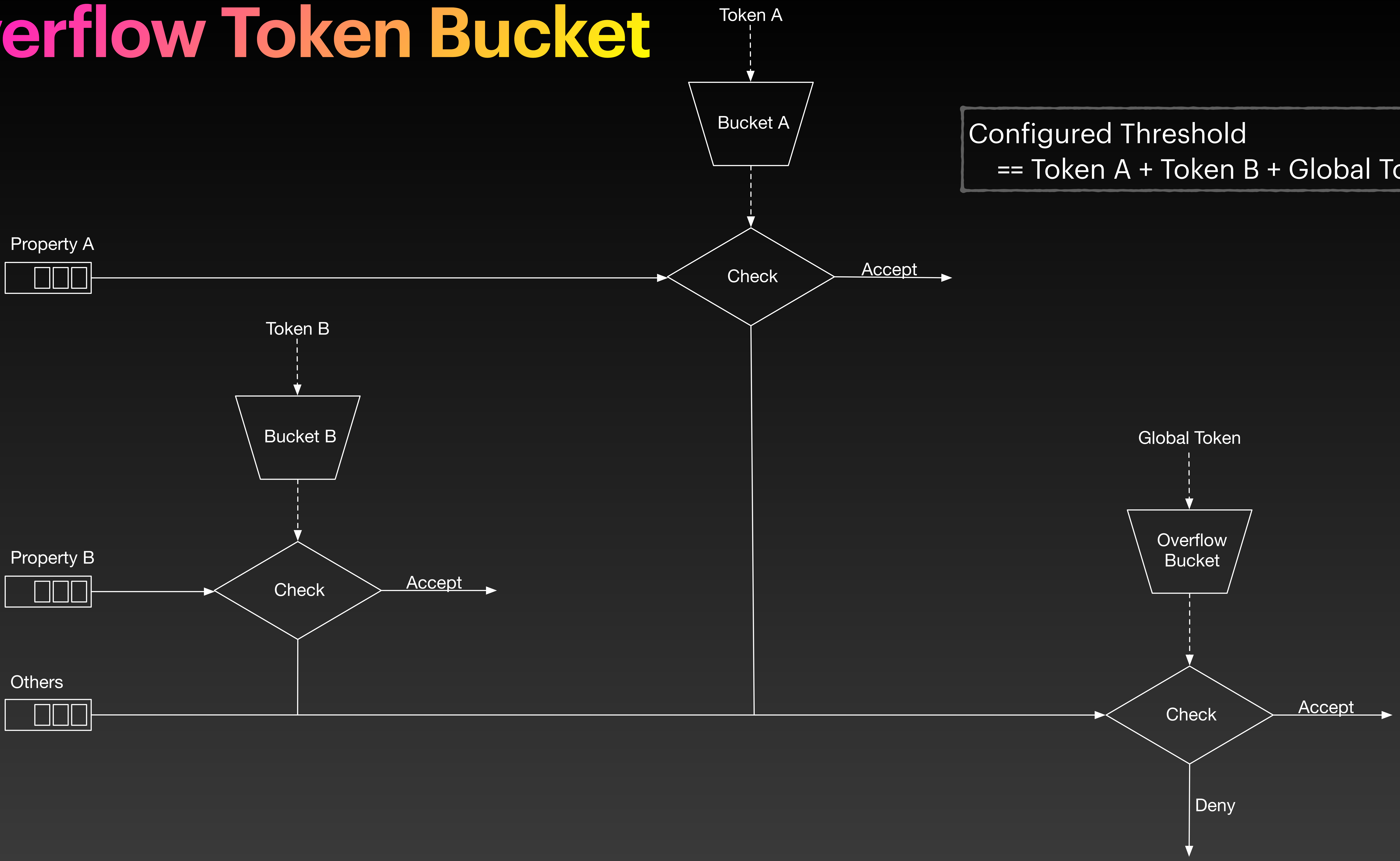
Token Bucket



Token Buckets



Overflow Token Bucket

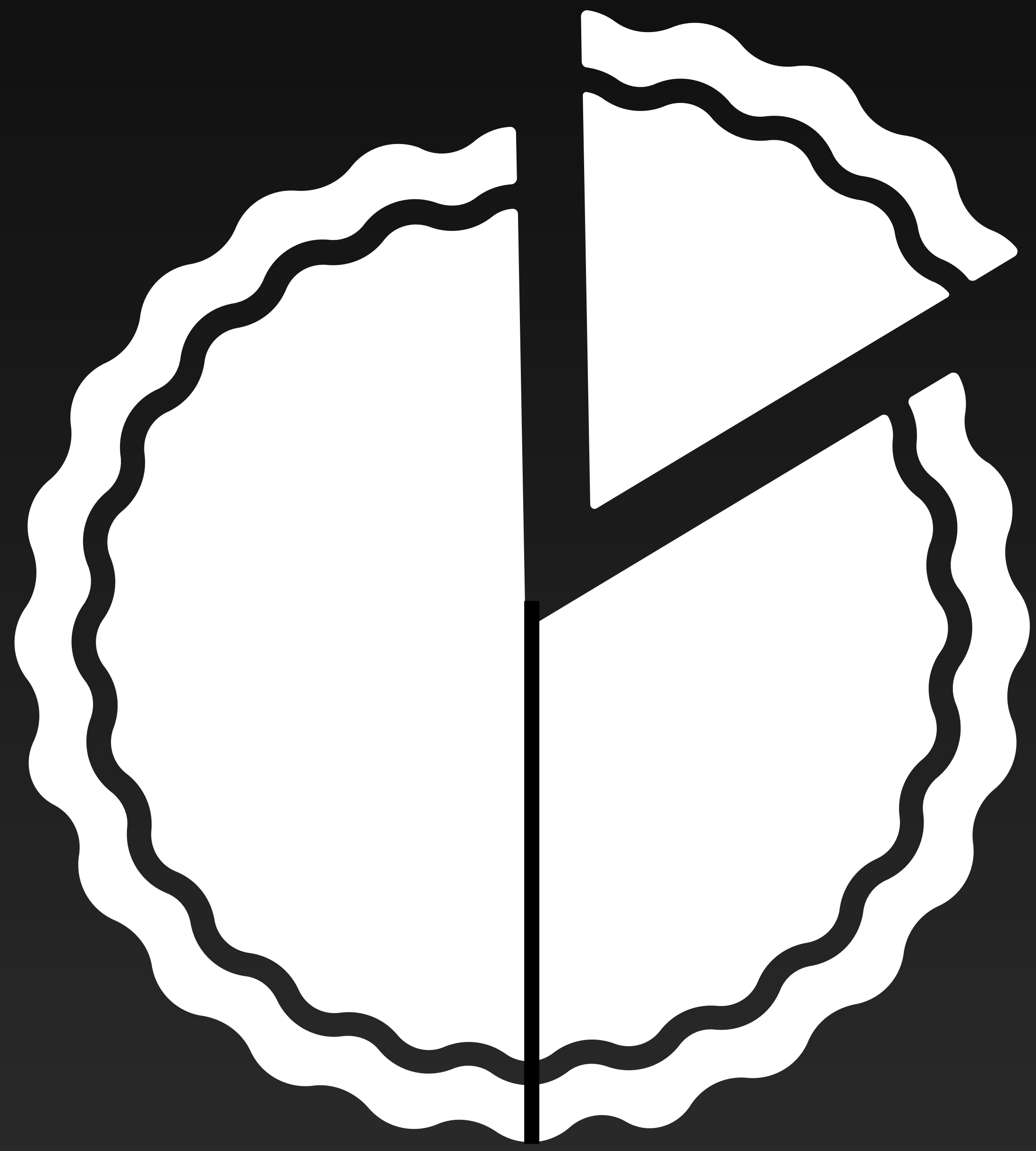


Configured Threshold
== Token A + Token B + Global Token

Strategy of dividing an apple-pie

1. Adjust the size to the demands

2. Follow the trend



Implementation Overview

Extend Continuation

1. Continuation Tag

2. Set tag on SNI Action / Remap

```
  #/iocore/eventsystem/I_Continuation.h
  class Continuation : private force_VFPT_to_top
  {
  ...
+  /**
+    Tag for Property
+   */
+  uint64_t tid = 0;
```

Target Metrics

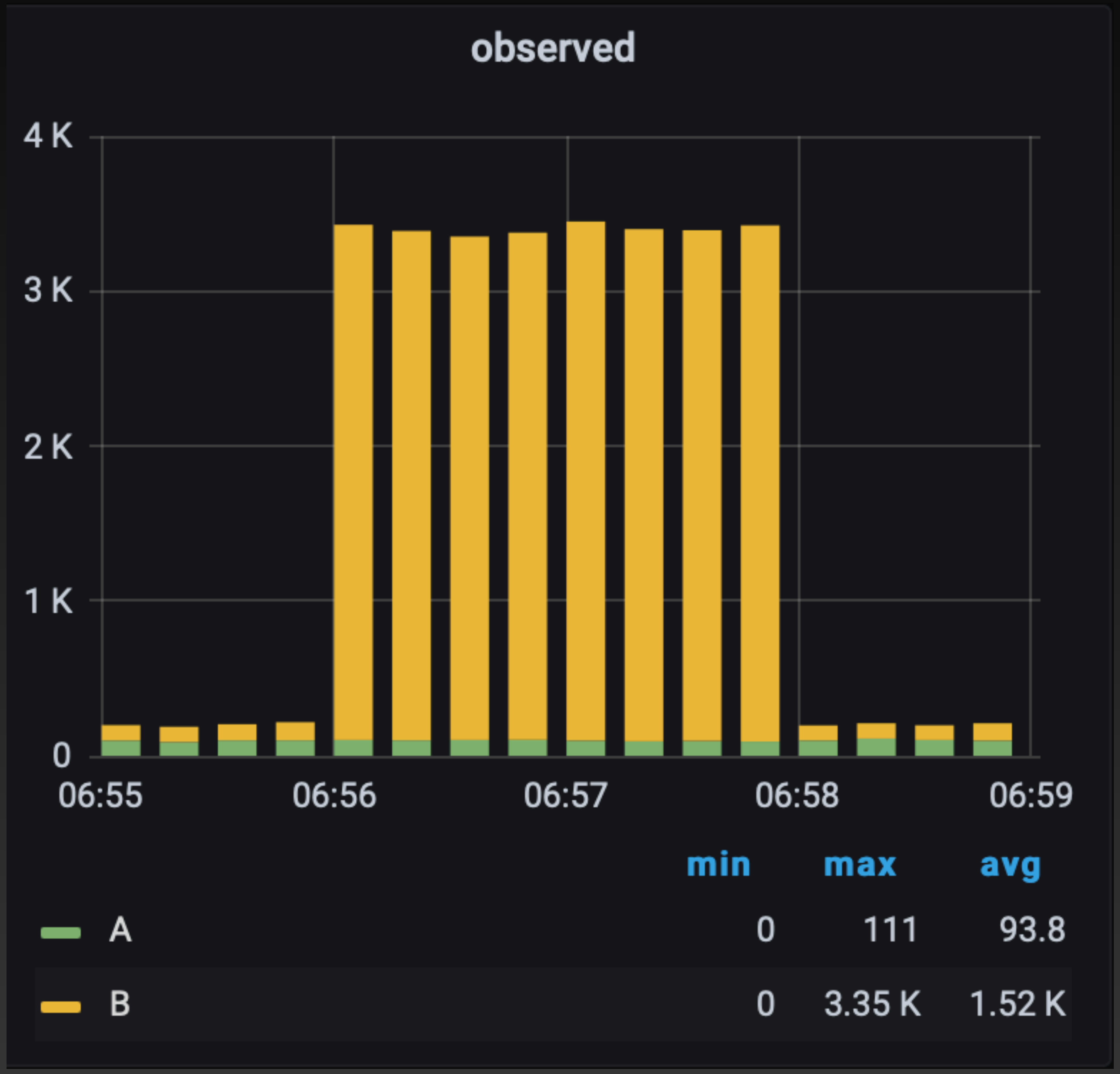
Metrics	Type	Source
TLS Handshake	Counter	SNI Callback
Active Connection	Gauge	NetHandler::add_to_active_queue NetHandler::remove_from_active_queue
Network IO	Counter	read_from_net write_to_net
Disk IO	Counter	ink_aio_read ink_aio_write
Event System Usage	Gauge	EThread::execute_regular(?)
Memory Usage	Gauge	MIOBuffer(?)

EXP.

Condition

- Client (wrk2)
    - Control: (A: 100 rps, B: 100 rps)
    - Noise: (B: 3200 rps)

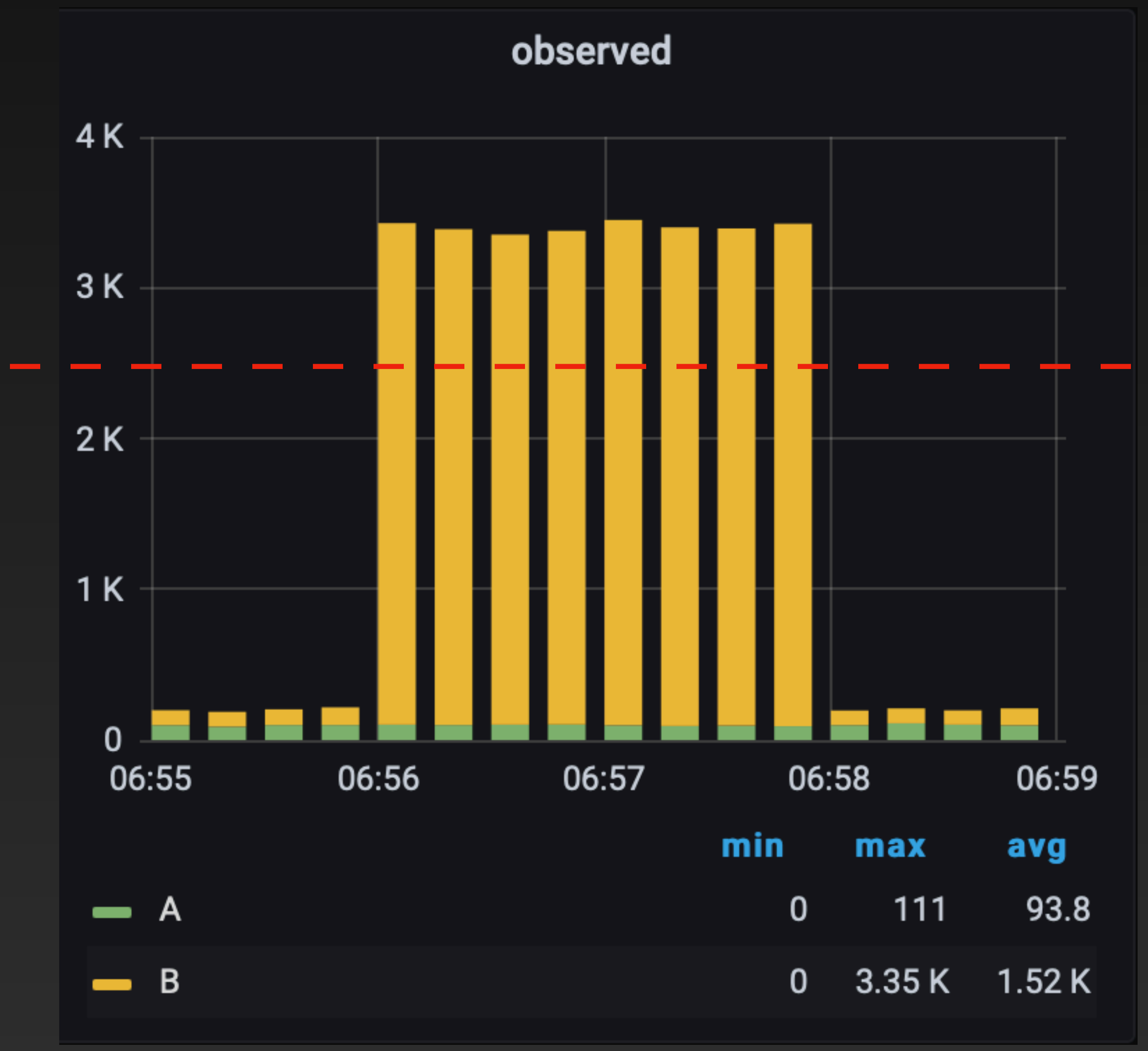
- ATS
    - 32 threads



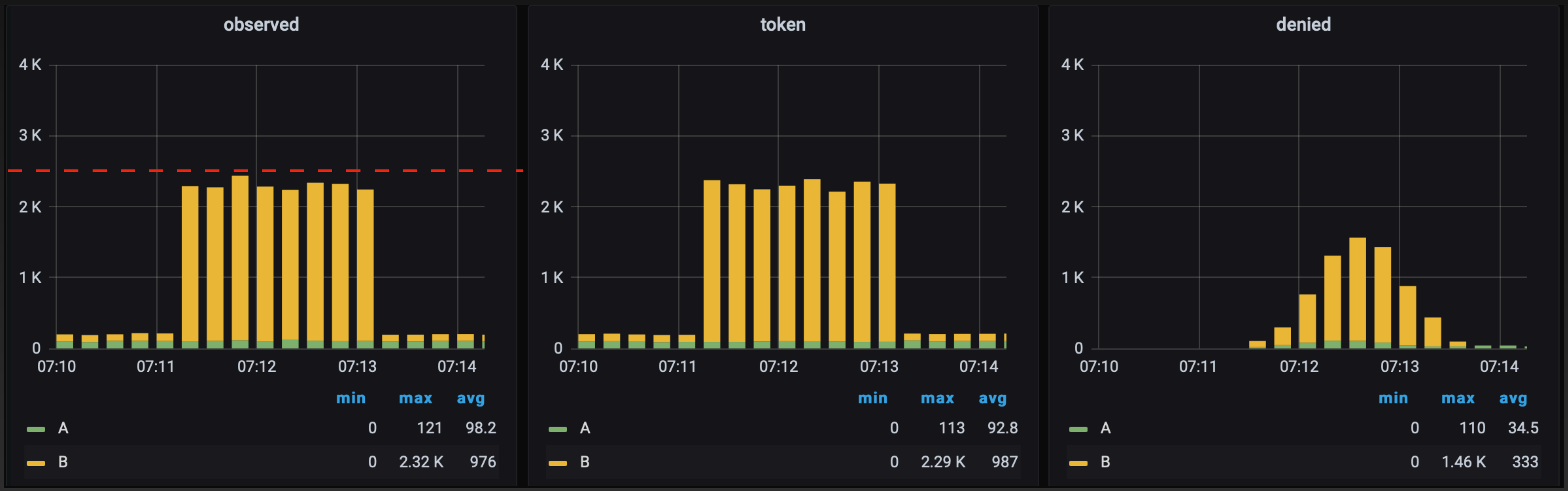
	min	max	avg
🟩 A	0	111	93.8
🟨 B	0	3.35 K	1.52 K

TLS Handshake

Before/After

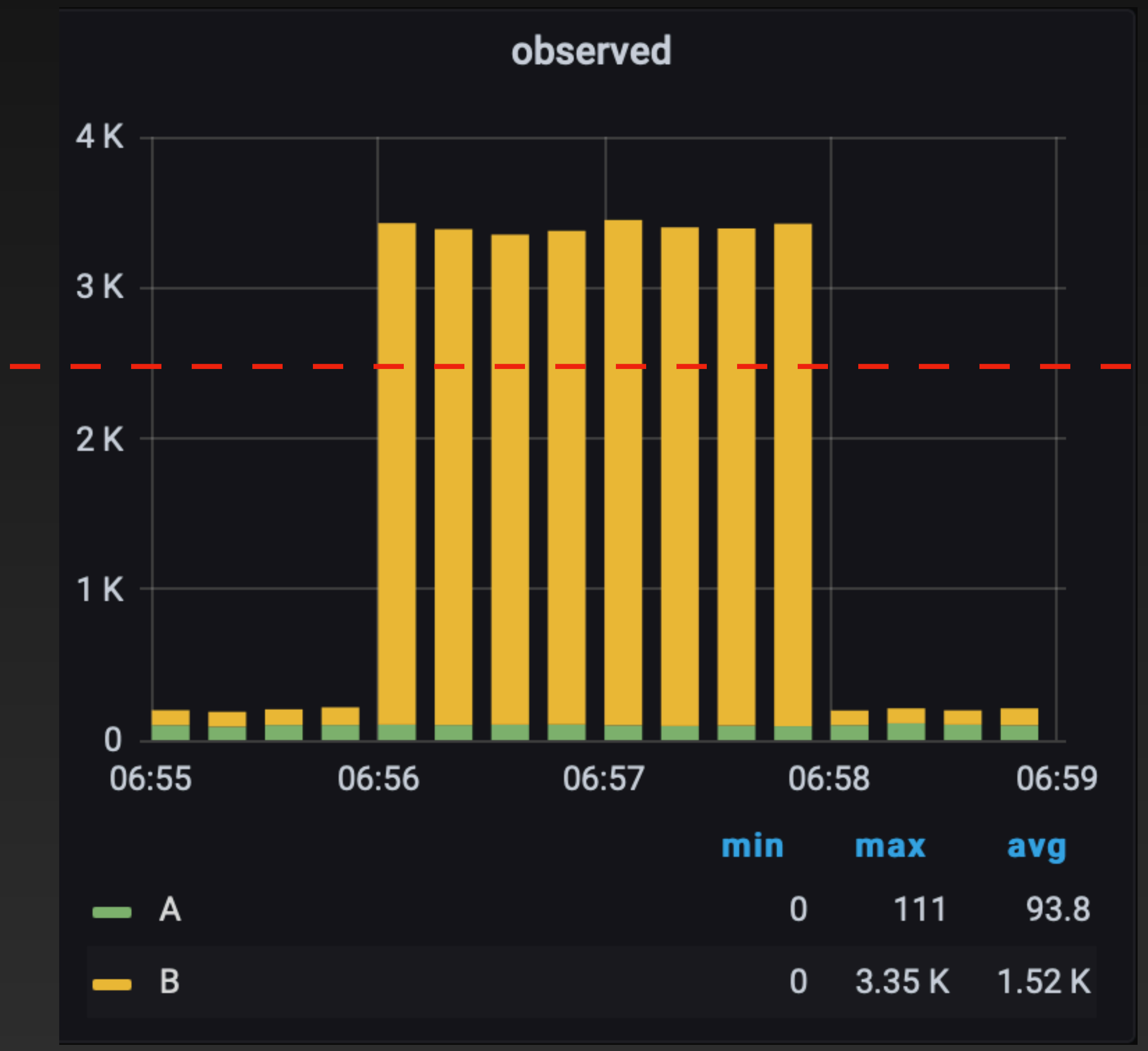


No Limit



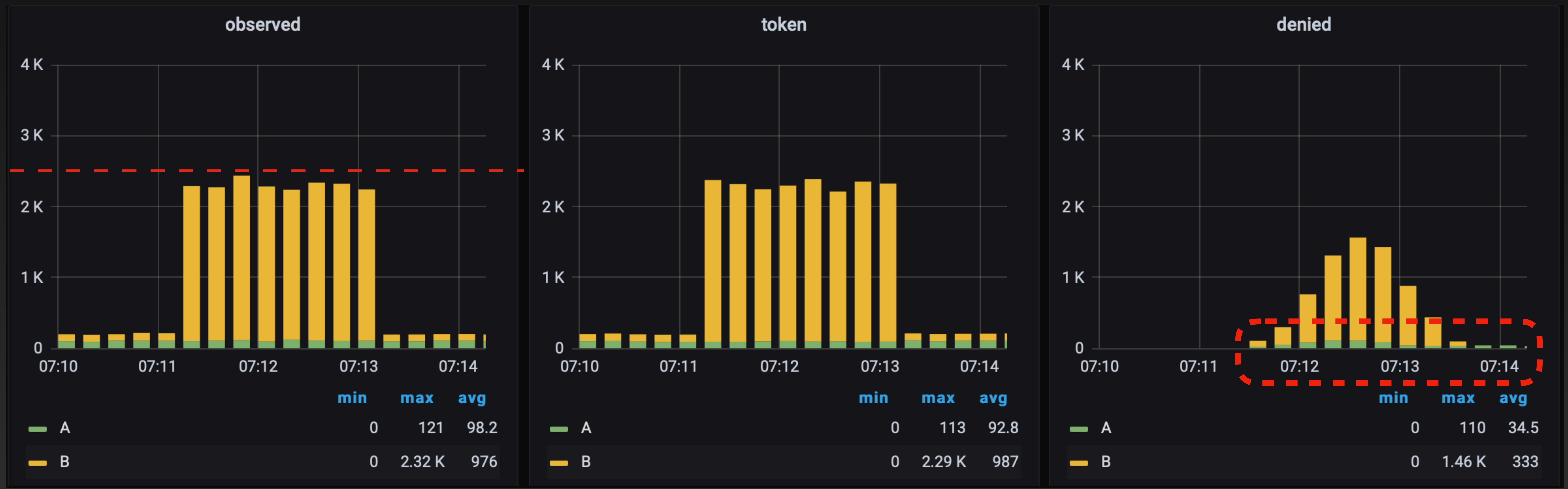
Limit == 2560

Before/After



	min	max	avg
A	0	111	93.8
B	0	3.35 K	1.52 K

No Limit



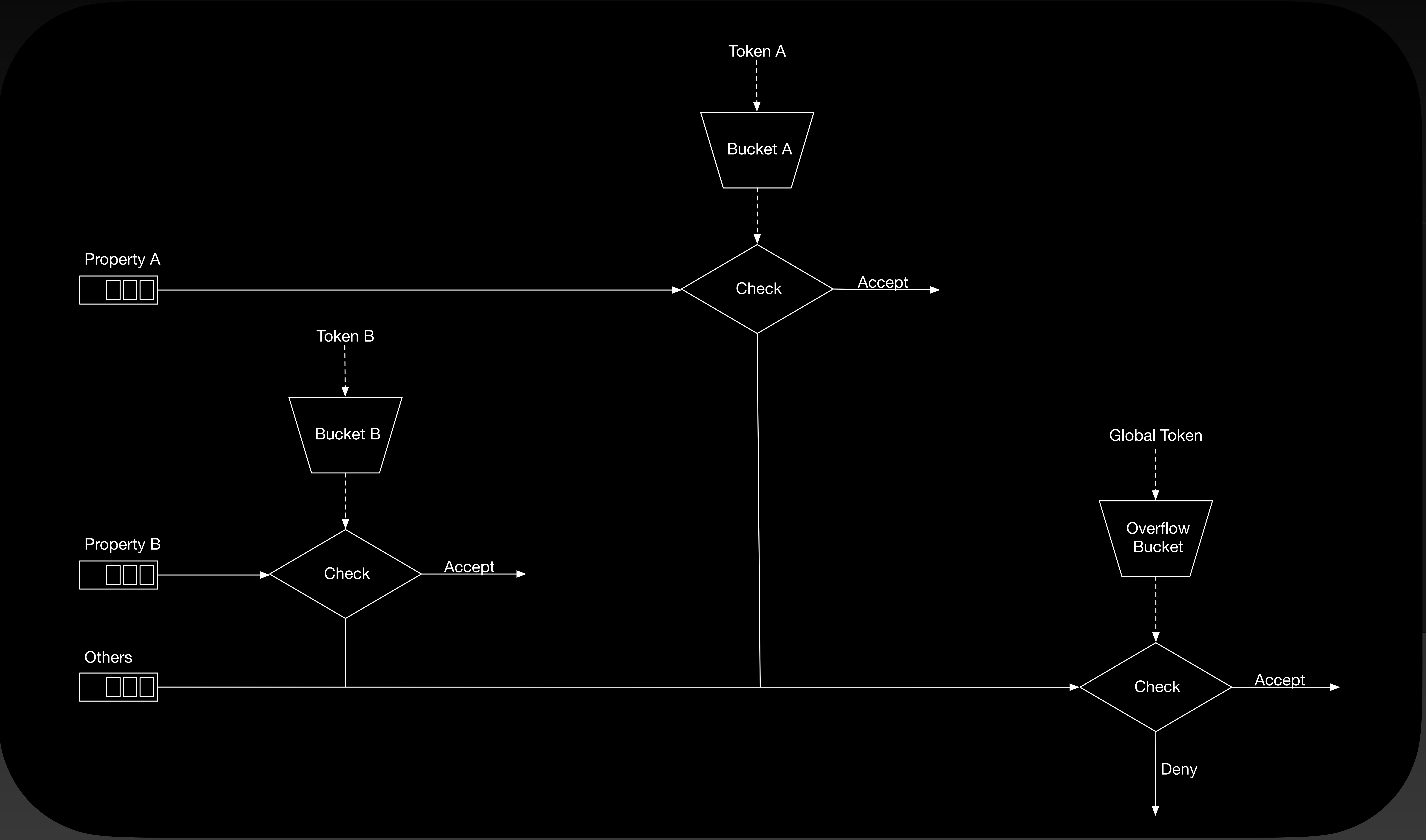
	min	max	avg
A	0	121	98.2
B	0	2.32 K	976

	min	max	avg
A	0	113	92.8
B	0	2.29 K	987

	min	max	avg
A	0	110	34.5
B	0	1.46 K	333

Limit == 2560

Before/After



Token A

Bucket A

Property A

Check

Accept

Token B

Bucket B

Property B

Check

Accept

Others

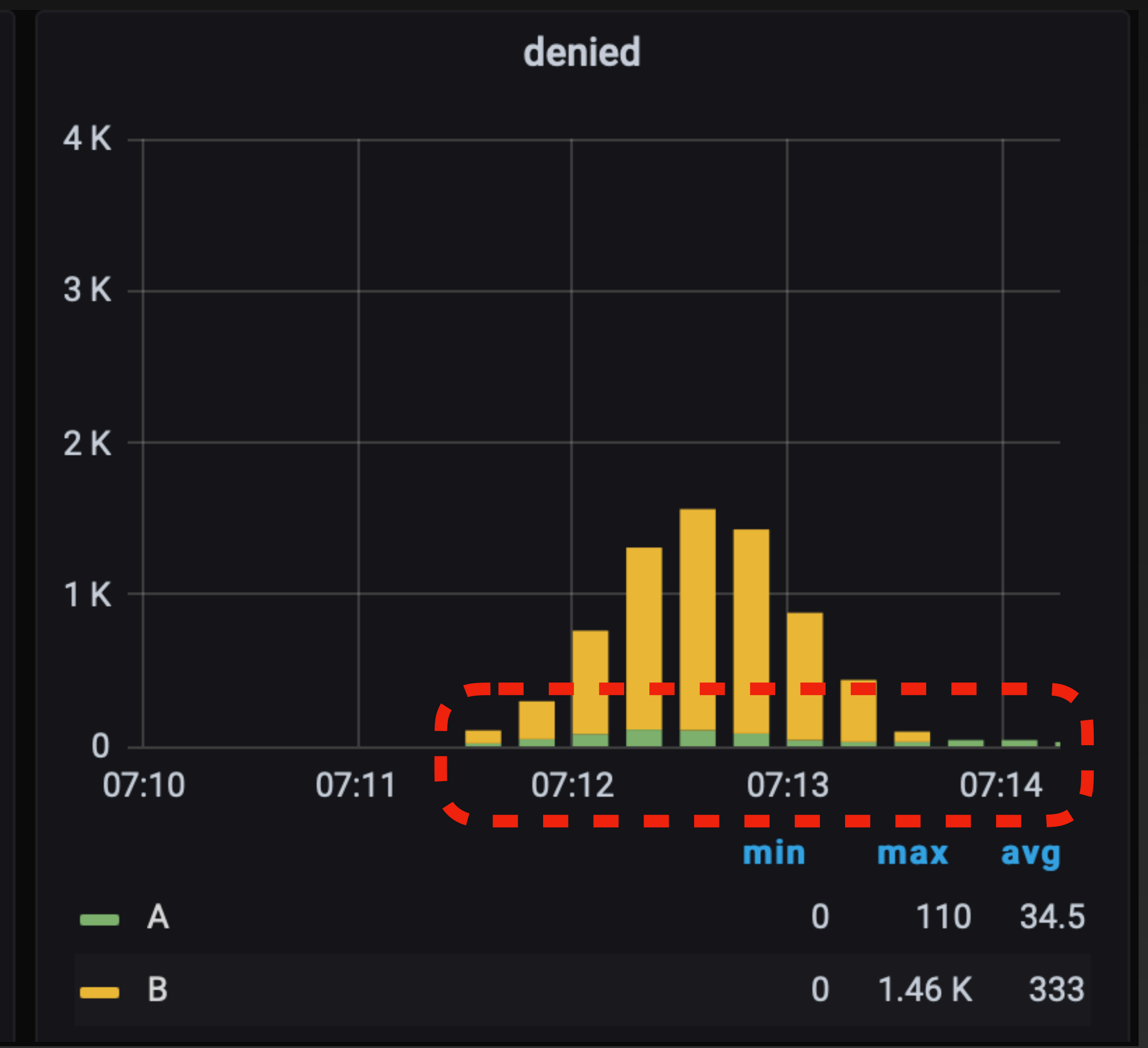
Global Token

Overflow Bucket

Check

Accept

Deny



	min	max	avg
🟩 A	0	110	34.5
🟨 B	0	1.46 K	333

Roadmap

1. More Tests at Lab
2. Run Prototype on production
3. Revisit Algorithm and Implementation
   - weights, rolling average, fair allocation…
4. Support more Metrics

References

1. Zhe Du, X Dai, and J Yu. Dynamic Token Allocation Strategy Based on Weight in TCSN System. CNCI, 2020.

2. J. Kidambi, D. Ghosal, and B. Mukherjee. Dynamic Token Bucket (DTB): A Fair Bandwidth Allocation Algorithm for High-Speed Networks. *Journal of High-Speed Networks*, 2001.

traffic server