Internet Engineering Task Force
Internet-Draft
Intended status: Informational
Expires: 4 September 2025

B. Zhang
Tsinghua SIGS
Z. Meng
HKUST
3 March 2025

Adapting Home Routers to Congestion Control's Reactions for Consistent Low Latency
draft-zhang-iccrg-confucius-00

Abstract

This document describes Confucius -- a practical queue management scheme designed for offering real-time flows with consistently low latency regardless of competition. Confucius employs an age-aware weight adjustment mechanism to slows down bandwidth adjustment to match the reaction of congestion control, so that the end host can reduce the sending rate without overshooting the network. Confucius is deployed on home routers and requires no configuration in normal Internet deployments.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on 4 September 2025.

Copyright Notice

Table of Contents

1. Introduction

Emerging high-quality real-time video streaming applications require consistently low latency, which is often disrupted by latency spikes caused by competing flows, especially those from web traffic.

The root cause of latency spikes in real-time flow is the mismatch in reaction time between the bandwidth reallocation on routers and congestion control mechanisms on endpoints. When loading a website, with 9 flows, in the same time with an existing real-time flow, the available bandwidth of the real-time flow is immediately reduced to 1/10 of what it was.

Congestion Control Algorithms (CCAs) reduce the real-time flow' s congestion window or sending rate only after end hosts observe latency increases or packet loss, by which time the reaction is already too late. It takes several RTTs for CCAs to converge to the new available bandwidth, while the excessive packets during this period will result in a bufferbloat and lead to an increase in the end-to-end delay.

Several approaches to AQM have been developed over the past two decades, but none have been able to practically control latency spikes in scenarios where real-time flows compete with eb traffic. With fair queueing (FQ and FqCoDel[RFC8289]), the latency spikes is

worsen since the per-flow fair queueing shifts large bandwidth away from real-time flows. By labeling flows in advance, AQMs like CBQ[CBQ1995], DiffServ[RFC4594] and L4S[RFC9330] schedule real-time flows and web traffic differently on the router using ’ StrictPriority ‘ or weighted class. However, this is impractical in home networks where the application and router belong to different entities since applications will be incentivized to fake their labels for better performance.

Learning from this history, the Confucius approach is designed to meet the following goals:

(a) Provide latency consistency to real-time flows independently of the number, rate, or CCA of the competing flows.

(b) Achieve eventual fairness between real-time flows and competing flows within a bounded convergence time.

(c) Operate transparently without endpoint labeling, ensuring practical deployment.

Confucius introduces, for the first time, an age-aware weight adjustment mechanism capable of distinguishing real-time flows from web traffic while smoothly regulating bandwidth allocation between them.

2. Conventions Used in This Document

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

3. Overview of Confucius

The Confucius algorithm described in the rest of this document uses one key variable: LAMBDA, which controls the speed of the weight adjustment.

3.1. Exponential Bandwidth Re-allocation

The scheduling of Confucius among flows is implemented with a per-flow Deficit Weighted Round Robin (DWRR) mechanism, where a flow’s bandwidth share equals its weight divided by the sum of all weights under DWRR scheduling. Unlike the uniform weights in FQ, Confucius carefully calculates the weight for each flow.

When a flow arrives at the home router, it is classified as a 'new flow' with an initial weight based on the number of existing new flows.

For each new flow, Confucius doubles its weight every 1/LAMBDA milliseconds. This shows how Confucius exponentially increases the weights of new flows. After a time of t the weight of the new flow will be multiplied by 2 LAMBDA * t.

Confucius uses a flow weight threshold of 1 to 'age out' new flows. Once the weight of a flow reaches 1, the flow is no longer considered new and will be marked as old flow.

3.2. Setting LAMBDA

A large LAMBDA leads to abrupt reductions in available bandwidth and causes latency spike, while a small LAMBDA results in unfairness for new flows.

The principle of setting LAMBDA SHOULD be to limit the performance degradation of competing flows to an acceptable level while maximizing 1/LAMBDA.

We configure LAMBDA = 0.004 (ms^-1) to have a doubling interval of 1/LAMBDA = 250 ms based on the reduction speed metrics of the real-time flows. For typical websites with 10-20 flows, the Page Load Time (PLT) degradation is less than 100 ms, which is acceptable compared to the second-level PLT.

With no changes to parameters, Confucius is expected to work across a wide range of conditions, with varying links and all major delay-controlling CCAs.

4. Annotated Pseudocode for Confucius

What follows is the pseudocode of the Confucius algorithm, illustrating the specific process of a new flow arriving, completing, and reweighting. We denote the set of new flows and all flows as F_new and F_all, respectively.

4.1. Arrival and completion of new flows

```
NewFlowArrival(f):
    for i in F_new:           /* Scale down weights */
        w_i = (|F_new| * w_i) / (|F_new| + 1)
    w_f = 1 / (|F_new| + 1)   /* Initialize the weight for f */
    F_new = F_new + f         /* Add flow f to set 'F_new' */
    F_all = F_all + f         /* Add flow f to set 'F_all' */
```

When a flow f just arrives at the queue, Confucius will adjust the weights of the flows that are already in F_new, rebalance the weights to the weight w_f for flow f.

* The sum of weights of new flows is upper-bounded.

  Rebalancing the weights of the new flows in F_new to the just arrived flow f ensures that the sum of weights of new flows will not surge. In general, the weights of new flows will increase only when they are doubled. In this case, the existing flows will not be affected by the arrival of flow f.

* The ratio between the weights of flows in F_new is maintained.

  During the rebalancing, the flows that arrived earlier will still proportionally have larger weights. For two flows i and j, if flow i arrives T units of time earlier than flow j and before the rebalancing, w_i / w_j = 2T, the ratio still holds after rebalancing. This will help flow with a longer age to exit from F_new earlier, and help to maintain the long-term fairness.

```
NewFlowCompletion(f):
    for i in F_new:           /* Scale up weights */
        w_i = (|F_new| * w_i) / (|F_new| - 1)
    F_new = F_new - f         /* Remove flow f from set 'F_new' */
    F_new = F_all - f         /* Remove flow f from set 'F_all' */
```

When one flow completes before its weight reaches the threshold, we will reallocate the bandwidth share to all flows in F_new. This ensures that the new flows can converge fast without affecting the existing flows.

4.2. Age-aware Weight Adjustment Mechanism

As described in Section 3.1, Confucius reweights every 1/LAMBDA ms as follows:

```
Reweight(f):
    for i in F_new:
        w_f = w_f * 2         /* The weight doubles every 1/LAMBDA ms */
    ration = sum_weight_of(F_new) / max(1, sum_weight_of(F_old))
    if ration > 1:            /* New flows are the minority */
        for f in F_new:
            w_f = w_f / ration
    for f in F_new:
        if w_f >= 1:          /* The new flow is old enough */
            w_f = 1
            F_new = F_new - f
```

* When there are more old flows than new flows.

  Confucius addresses the issue where a burst of new flows impacts a few existing flows. However, there may be more existing flows and only a few new flows. In this case, the new flow will not drastically grab the available bandwidth from the existing flows. We allocate resources to new flows as long as the bandwidth reduction of existing flows is bounded by half and the fair share.

* When flows are no longer new, i.e., their weight reaches 1, Confucius removes the flow from F_new. After that, the flow's weight will remain 1 all the time in the scheduling.

Consider the example in Section 1:

There is a real-time flow existing in the Confucius queue that is marked as an 'old flow'. Then, a web page initiates multiple flows, leading to a burst of new flows. The sum weight of the weights of the new flows equals the weight of the real-time flow, causing the bandwidth of the real-time flow to be reduced to half of its original value instead of 1/10. Subsequently, the weights of the web flows grow exponentially and quickly converge towards fair allocation within a few RTTs, which is much shorter than the completion time of web traffic. During this process, the bandwidth is smoothly allocated between the real-time flow and the web traffic.

Web traffic tends to start with multiple concurrently active flows, while real-time traffic usually generates only a few flows. Despite the absence of labels from the endpoints, Confucius is still able to distinguish web traffic and implement a different reaction compared to real-time flows.

5. IANA Considerations

This memo includes no request to IANA.

6. Security Considerations

This document should not affect the security of the Internet.

7. References

7.1. Normative References

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

7.2. Informative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

[RFC8289] Nichols, K., "Controlled Delay Active Queue Management", January 2018, <https://www.rfc-editor.org/info/rfc8289>.

[CBQ1995] Floyd, S., "Link-sharing and resource management models for packet networks", 1995.

[RFC4594] Babiarz, J., "Configuration Guidelines for DiffServ Service Classes", August 2006, <https://www.rfc-editor.org/info/rfc4594>.

[RFC9330] Briscoe, B., "Low Latency, Low Loss, and Scalable Throughput (L4S) Internet Service: Architecture", January 2023, <https://www.rfc-editor.org/info/rfc9330>.

Contributors

Thanks to all of the contributors.

Zili Meng
Hong Kong University of Science and Technology (HKUST)
Email: zilim@ust.hk

Bochun Zhang
Tsinghua Shenzhen International Graduate School (SIGS)
Email: zbc22@mails.tsinghua.edu.cn

Authors' Addresses

Bochun Zhang
Tsinghua SIGS
China
Email: zbc22@mails.tsinghua.edu.cn

Zili Meng
HKUST
HongKong
China
Email: zilim@ust.hk

URI: https://zilimeng.com/