CLOUDFLARE®

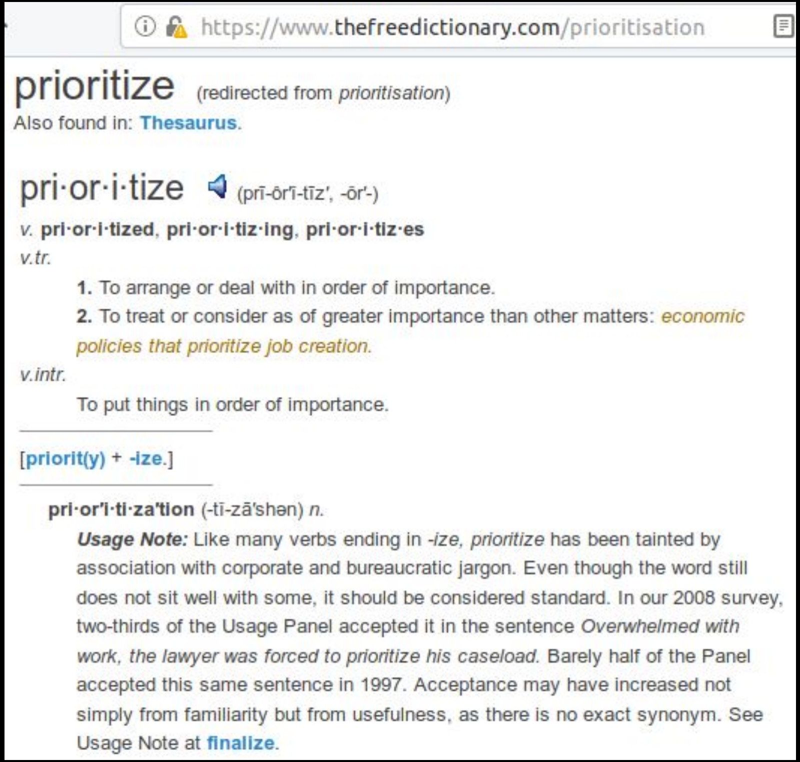
H TTP PRIOR ITIZATION

Lucas Pardue & Nick Jones, with contribution from Pat Meenan

H TTP PRIOR ITIZATION

CLOUDFLARE

H
TTP
PRIOR
ITISATION



https://www.thefreedictionary.com/prioritisation

prioritize (redirected from *prioritisation*)

Also found in: **Thesaurus**.

pri·or·i·tize (prī-ôr′ĭ-tīz′, -ŏr′-)

v. **pri·or·i·tized**, **pri·or·i·tiz·ing**, **pri·or·i·tiz·es**

v.tr.

1. To arrange or deal with in order of importance.

2. To treat or consider as of greater importance than other matters: *economic policies that prioritize job creation.*

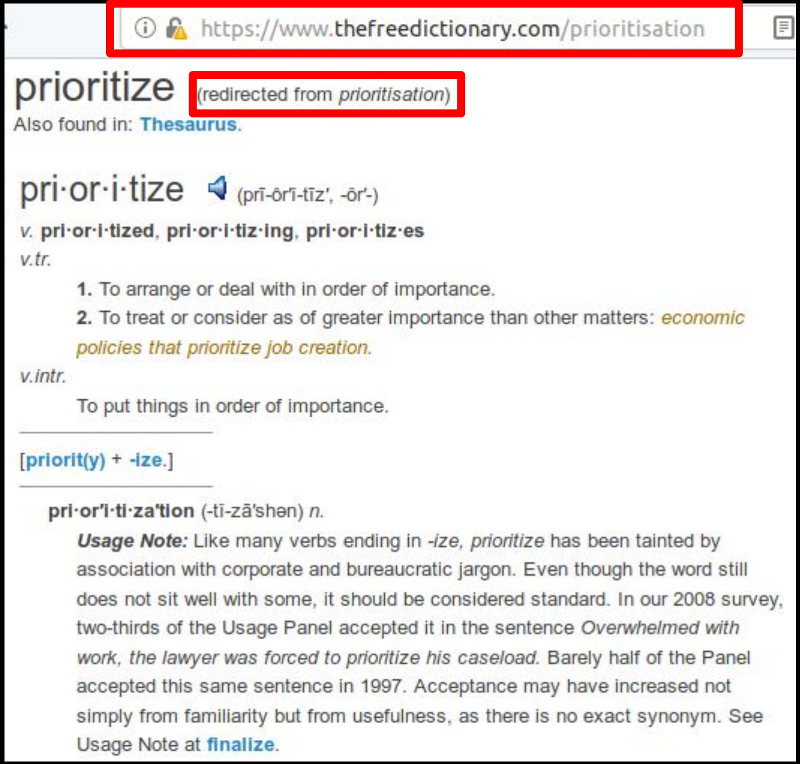
v.intr.

To put things in order of importance.

[**priorit(y)** + **-ize**.]

pri·or′i·ti·za′tion (-tĭ-zā′shən) *n.*

Usage Note: Like many verbs ending in *-ize, prioritize* has been tainted by association with corporate and bureaucratic jargon. Even though the word still does not sit well with some, it should be considered standard. In our 2008 survey, two-thirds of the Usage Panel accepted it in the sentence *Overwhelmed with work, the lawyer was forced to prioritize his caseload.* Barely half of the Panel accepted this same sentence in 1997. Acceptance may have increased not simply from familiarity but from usefulness, as there is no exact synonym. See Usage Note at **finalize**.



https://www.thefreedictionary.com/prioritisation

prioritize (redirected from *prioritisation*)

Also found in: Thesaurus.

pri·or·i·tize (prī-ôr′ĭ-tīz′, -ŏr′-)

v. **pri·or·i·tized**, **pri·or·i·tiz·ing**, **pri·or·i·tiz·es**

v.tr.

1. To arrange or deal with in order of importance.

2. To treat or consider as of greater importance than other matters: *economic policies that prioritize job creation.*

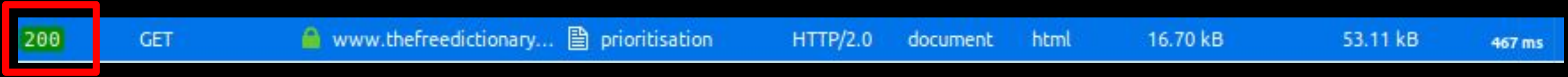
v.intr.

To put things in order of importance.

[priorit(y) + -ize.]

pri·or′i·ti·za′tion (-tĭ-zā′shən) *n.*

Usage Note: Like many verbs ending in *-ize, prioritize* has been tainted by association with corporate and bureaucratic jargon. Even though the word still does not sit well with some, it should be considered standard. In our 2008 survey, two-thirds of the Usage Panel accepted it in the sentence *Overwhelmed with work, the lawyer was forced to prioritize his caseload.* Barely half of the Panel accepted this same sentence in 1997. Acceptance may have increased not simply from familiarity but from usefulness, as there is no exact synonym. See Usage Note at finalize.



```
200    GET    www.thefreedictionary...    prioritisation    HTTP/2.0    document    html    16.70 kB    53.11 kB    467 ms
```

"Even though the word [Prioritization] does not sit well with some, it should be considered standard."

-The Free Dictionary

"Lies, damned lies, and statistics."

-Someone

Prioritization in not merely an HTTP/2 thing.

Effective use of available resources to achieve the best* user experience.

* For some definition of best

CLOUDFLARE

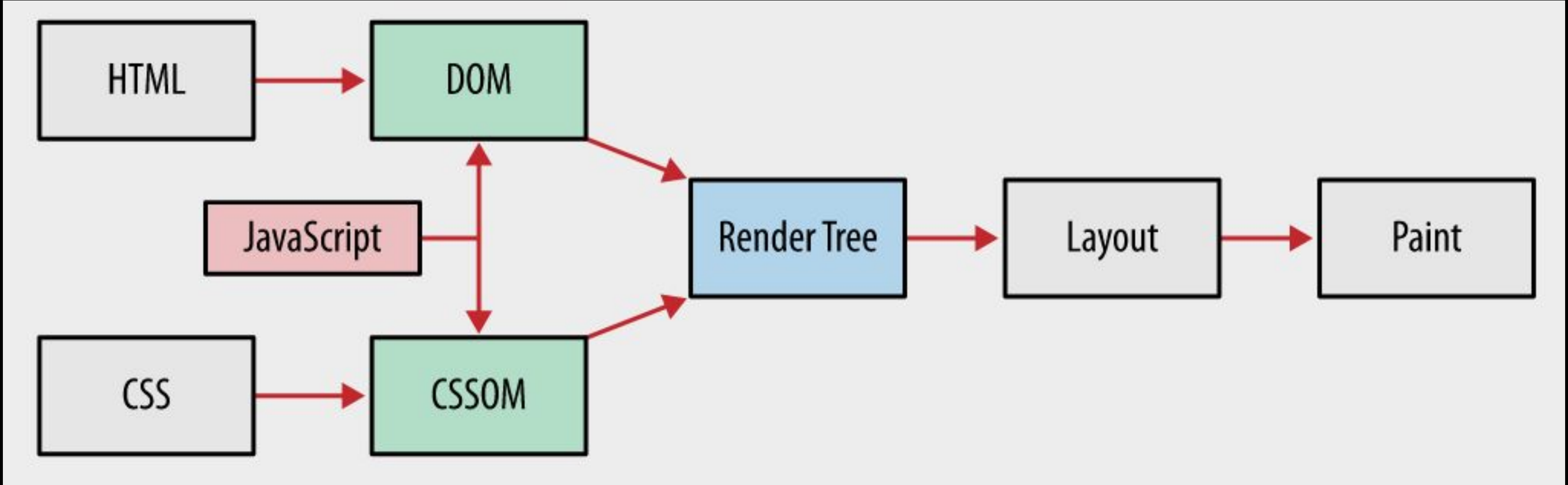


Figure 10-1. Browser processing pipeline: HTML, CSS, and JavaScript

https://hpbn.co/primer-on-web-performance/

User-centric metrics

Time to Interactive **(TTI)** - The point in time when the main content has painted and the user can expect it to respond quickly to input.

First Contentful Paint **(FCP)** - The time when the first text or image is drawn to the screen after navigation (i.e. not a background page color).

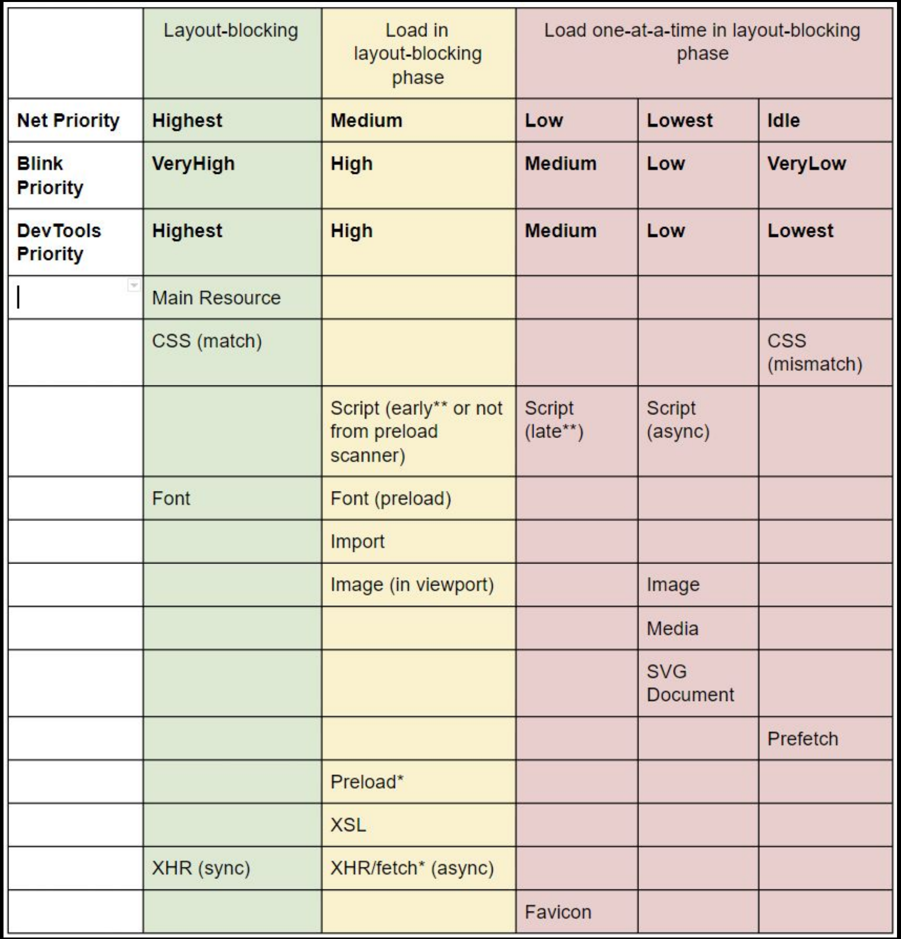
DOM Content Loaded **(DCL)** - Basically when the main HTML parser has made it to the end of the document.

Speed Index **(SI)** - The average time to get content onto the screen.

CLOUDFLARE

https://developers.google.com/web/fundamentals/performance/user-centric-performance-metrics

Resource fetch prioritization



	Layout-blocking	Load in layout-blocking phase	Load one-at-a-time in layout-blocking phase		
Net Priority	**Highest**	**Medium**	**Low**	**Lowest**	**Idle**
Blink Priority	**VeryHigh**	**High**	**Medium**	**Low**	**VeryLow**
DevTools Priority	**Highest**	**High**	**Medium**	**Low**	**Lowest**
	Main Resource				
	CSS (match)				CSS (mismatch)
		Script (early** or not from preload scanner)	Script (late**)	Script (async)	
	Font	Font (preload)			
		Import			
		Image (in viewport)		Image	
				Media	
				SVG Document	
					Prefetch
		Preload*			
		XSL			
	XHR (sync)	XHR/fetch* (async)			
			Favicon		

https://docs.google.com/document/d/1bCDuq9H1ih9iNjgzyAL0gpwNFiEP4TZS-YLRp_RuMlc/edit?usp=sharing

https://www.southwalesargus.co.uk/news/17301626.updated-tolls-on-the-severn-bridges-abolished-today-after-52-years/

1st parties, 3rd parties and damned connections

C - TCP connections per host

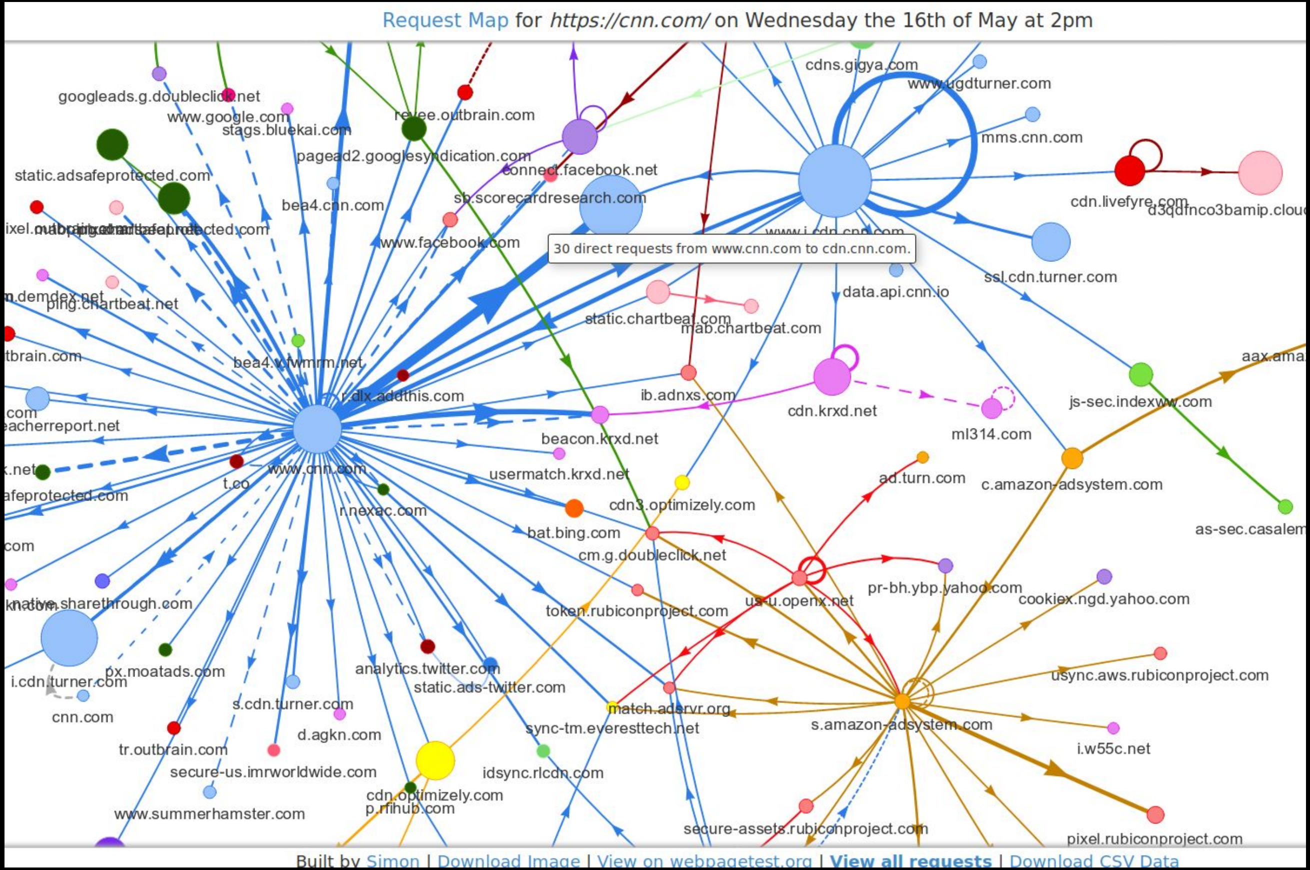
S - domain shards per site

T - 3rd parties per site

$$Total = 1C + SC + \int_0^T S_T C$$

- SPDY, HTTP/2, HTTP/3 multiplexing reduces C to 1.
- Coalescing, ORIGIN, CERTIFICATE help reduce S to 0.

$$Total = 1 + \lim_{S \to 0}(S) + \int_0^T S_T$$



https://csswizardry.com/2018/05/identifying-auditing-discussing-third-parties/
http://requestmap.webperf.tools/render/180516_9B_f8f13e248bf901309ace50687acc070b

Weight-based priority

SPDY 3.1

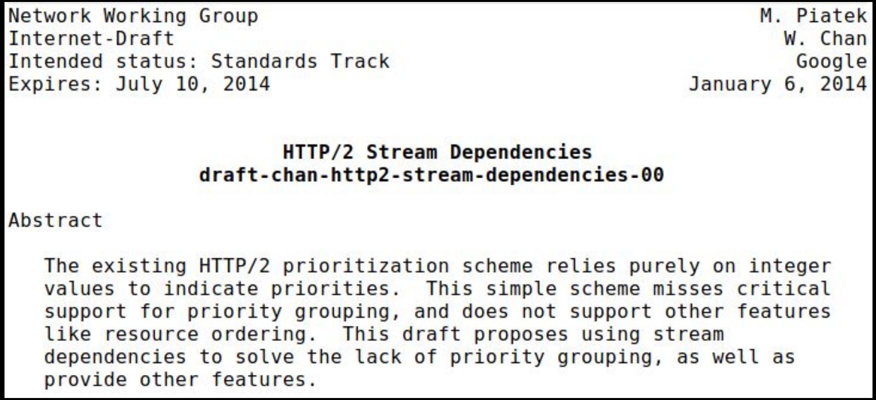
The creator of a stream assigns a priority for that stream. **Priority is represented as an integer from 0 to 7. 0 represents the highest priority and 7 represents the lowest priority.**

The sender and recipient SHOULD use best-effort to process streams in the order of highest priority to lowest priority.

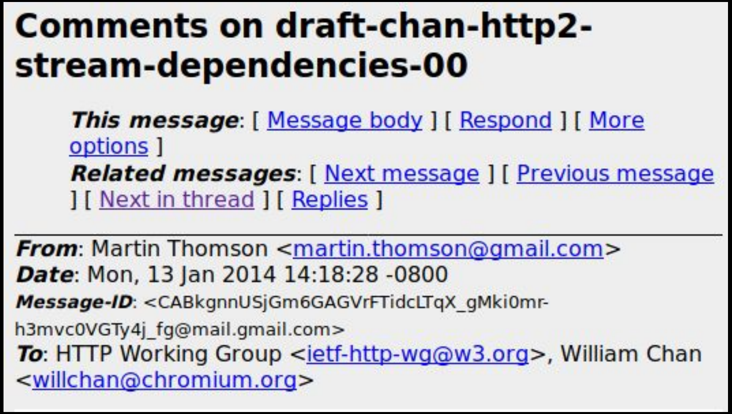
HTTP/2 until draft 11

The endpoint establishing a new stream can assign a priority for the stream. **Priority is represented as an unsigned 31-bit integer. 0 represents the highest priority and 2^31-1 represents the lowest priority.**

Dependency & weight-based priority



```
Network Working Group                                          M. Piatek
Internet-Draft                                                   W. Chan
Intended status: Standards Track                                  Google
Expires: July 10, 2014                                   January 6, 2014


                        HTTP/2 Stream Dependencies
                 draft-chan-http2-stream-dependencies-00

Abstract

   The existing HTTP/2 prioritization scheme relies purely on integer
   values to indicate priorities.  This simple scheme misses critical
   support for priority grouping, and does not support other features
   like resource ordering.  This draft proposes using stream
   dependencies to solve the lack of priority grouping, as well as
   provide other features.
```



Comments on draft-chan-http2-stream-dependencies-00

This message: [Message body] [Respond] [More options]
Related messages: [Next message] [Previous message] [Next in thread] [Replies]

From: Martin Thomson <martin.thomson@gmail.com>
Date: Mon, 13 Jan 2014 14:18:28 -0800
Message-ID: <CABkgnnUSjGm6GAGVrFTidcLTqX_gMki0mr-h3mvc0VGTy4j_fg@mail.gmail.com>
To: HTTP Working Group <ietf-http-wg@w3.org>, William Chan <willchan@chromium.org>

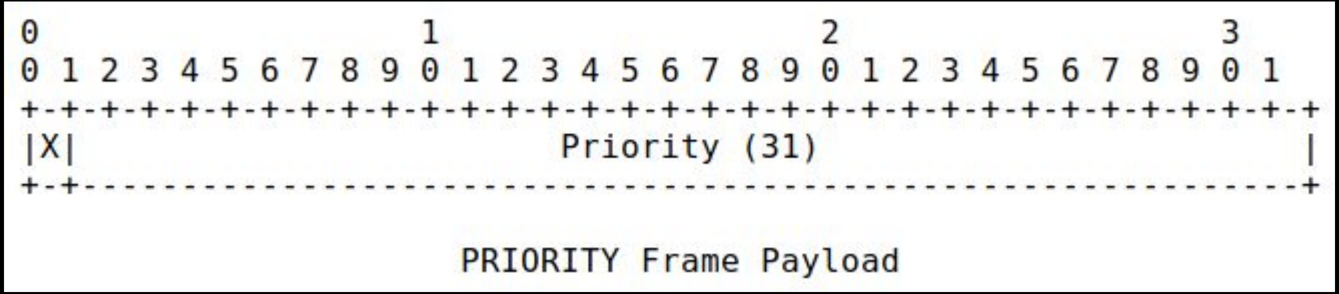
https://tools.ietf.org/html/draft-chan-http2-stream-dependencies-00

https://lists.w3.org/Archives/Public/ietf-http-wg/2014JanMar/0090.html

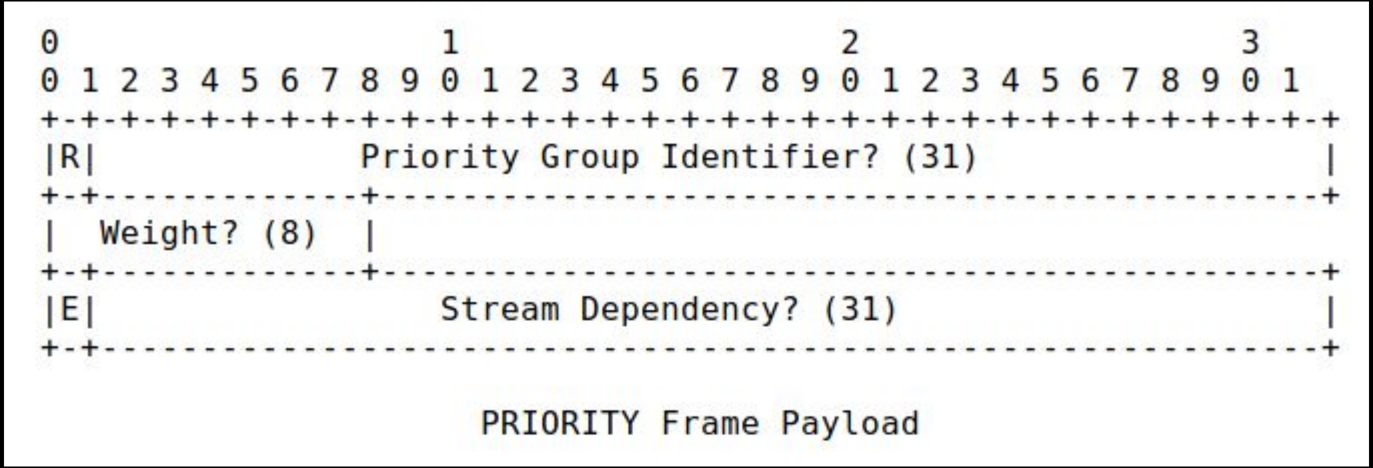
"Section 5 is a wee bit long ... that said, I'm surprised at how little mention there is of the intermediary bug that first caused this to arise."

-MT

Add dependencies to HTTP/2



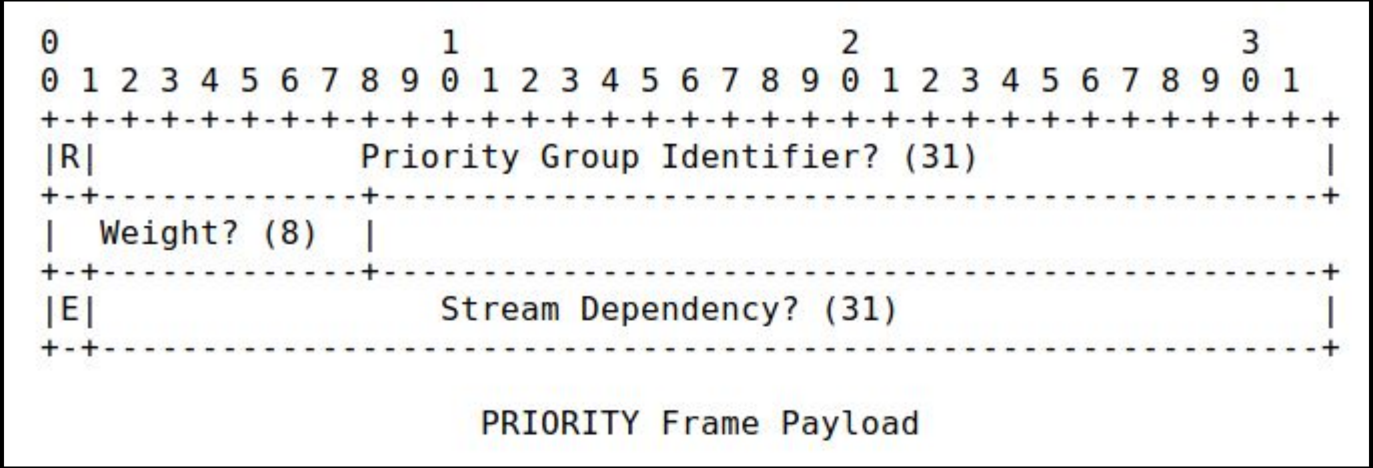
```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|X|                        Priority (31)                        |
+-+-------------------------------------------------------------+

                     PRIORITY Frame Payload
```



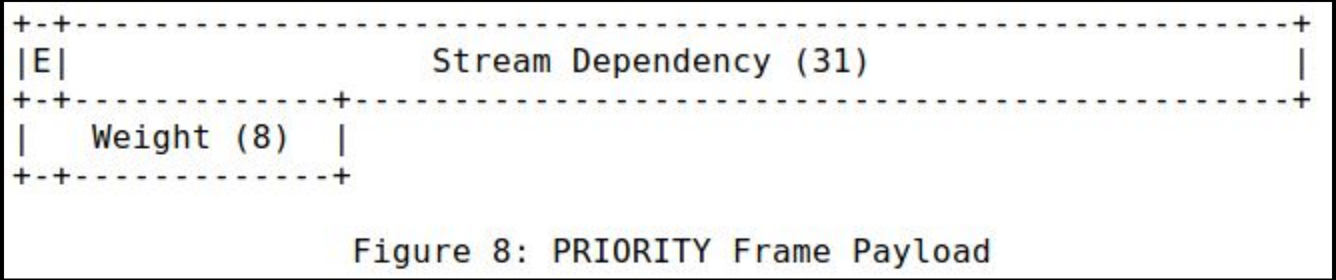
```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|R|              Priority Group Identifier? (31)                |
+-+-------------+-----------------------------------------------+
|  Weight? (8)  |
+-+-------------+-----------------------------------------------+
|E|                  Stream Dependency? (31)                    |
+-+-------------------------------------------------------------+

                     PRIORITY Frame Payload
```

https://tools.ietf.org/rfcdiff?url2=draft-ietf-httpbis-http2-11.txt

Refactor Prioritization in HTTP/2

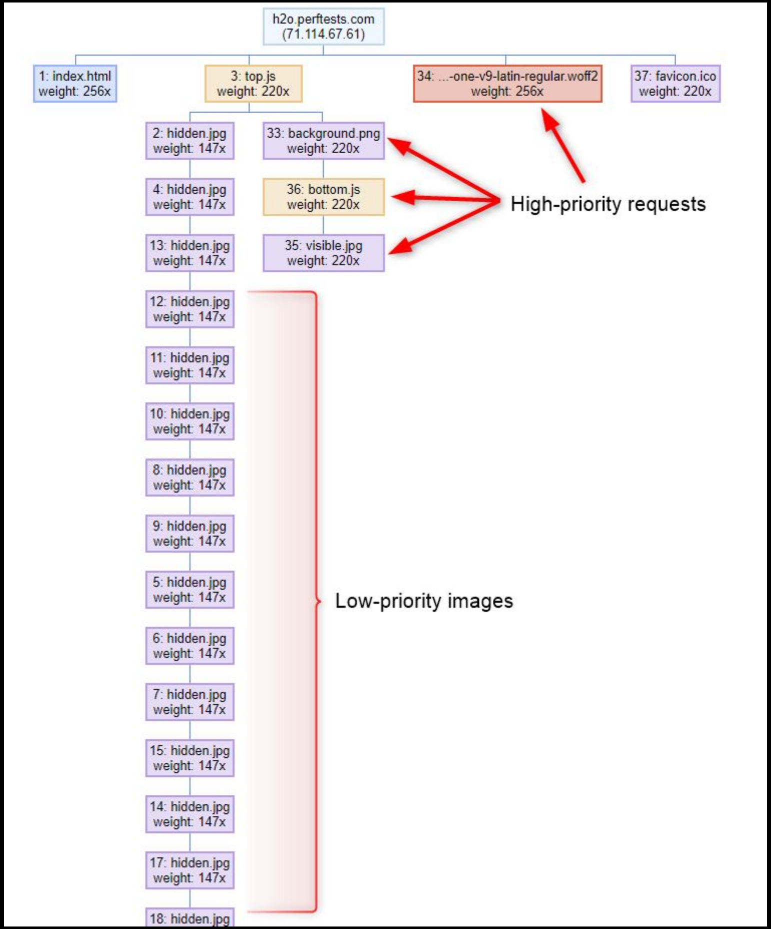


```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|R|             Priority Group Identifier? (31)                 |
+-+-------------+-----------------------------------------------+
|  Weight? (8)  |
+-+-------------+-----------------------------------------------+
|E|                 Stream Dependency? (31)                     |
+-+-------------------------------------------------------------+

                  PRIORITY Frame Payload
```



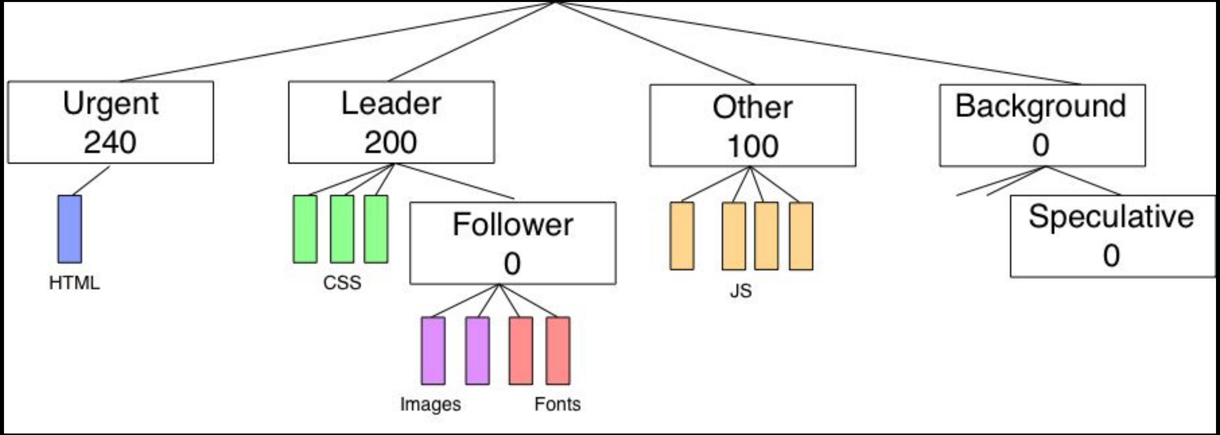
```
+-+-------------------------------------------------------------+
|E|                  Stream Dependency (31)                     |
+-+-------------+-----------------------------------------------+
|   Weight (8)  |
+-+-------------+

               Figure 8: PRIORITY Frame Payload
```

Chrome


h2o.perftests.com (71.114.67.61)
1: index.html weight: 256x
3: top.js weight: 220x
34: ...-one-v9-latin-regular.woff2 weight: 256x
37: favicon.ico weight: 220x
2: hidden.jpg weight: 147x
33: background.png weight: 220x
4: hidden.jpg weight: 147x
36: bottom.js weight: 220x
13: hidden.jpg weight: 147x
35: visible.jpg weight: 220x
12: hidden.jpg weight: 147x
11: hidden.jpg weight: 147x
10: hidden.jpg weight: 147x
8: hidden.jpg weight: 147x
9: hidden.jpg weight: 147x
5: hidden.jpg weight: 147x
6: hidden.jpg weight: 147x
7: hidden.jpg weight: 147x
15: hidden.jpg weight: 147x
14: hidden.jpg weight: 147x
17: hidden.jpg weight: 147x
18: hidden.jpg
High-priority requests
Low-priority images

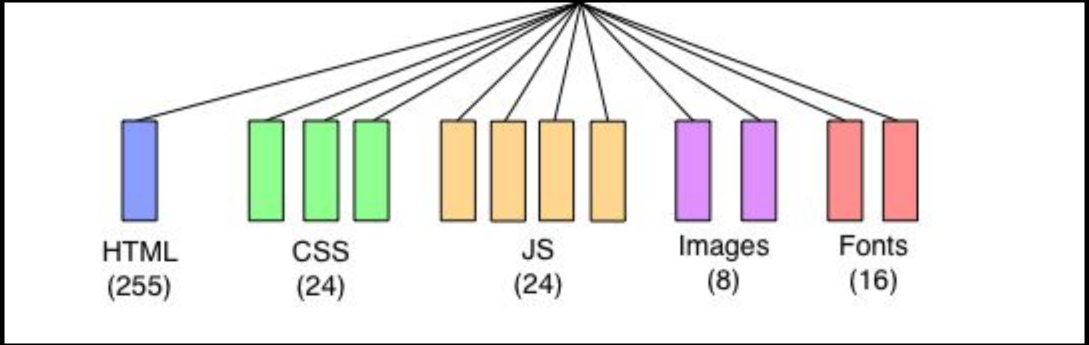
https://calendar.perfplanet.com/2018/http2-prioritization/

Firefox



https://calendar.perfplanet.com/2018/http2-prioritization/

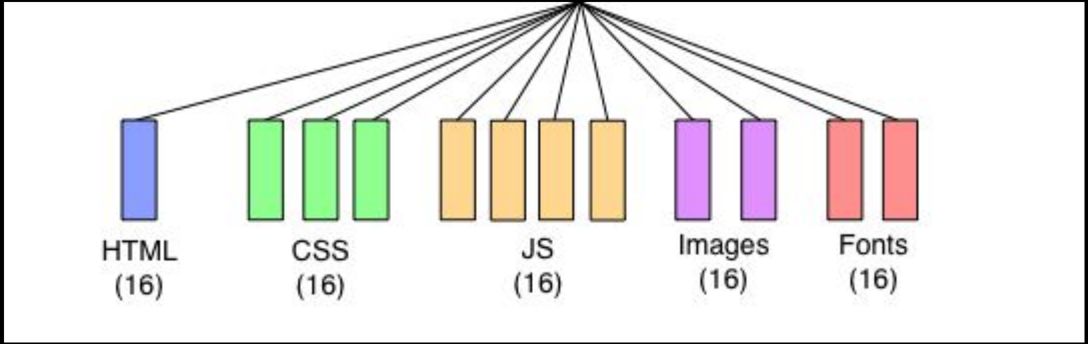
Safari



https://calendar.perfplanet.com/2018/http2-prioritization/

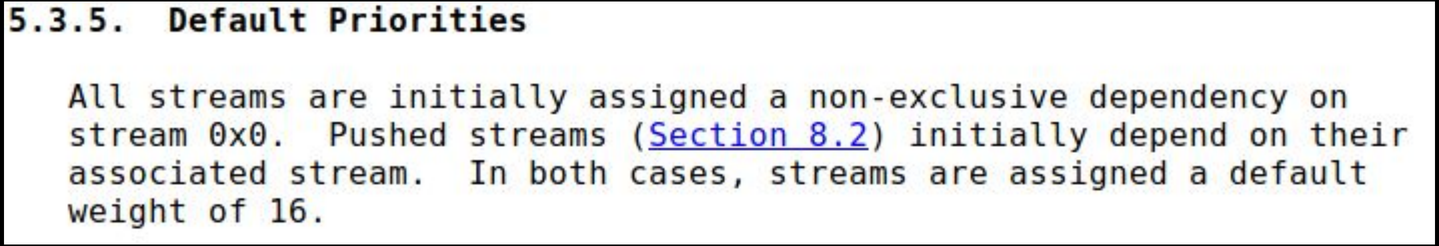
Edge

¯_(ツ)_/¯



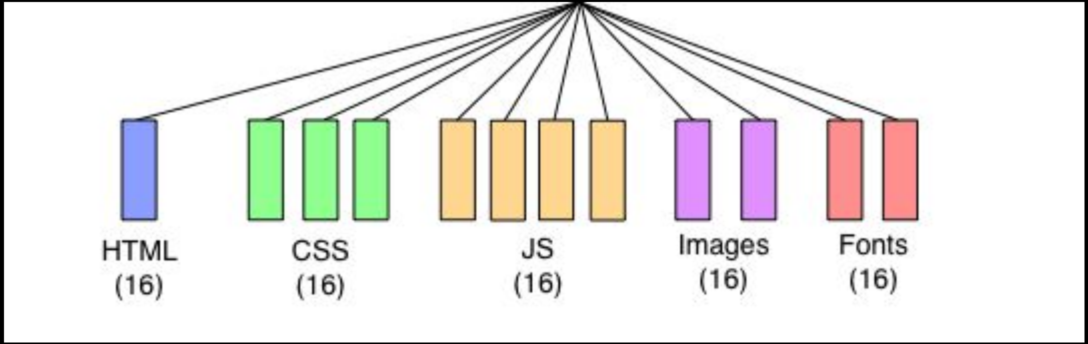
https://calendar.perfplanet.com/2018/http2-prioritization/

libcurl



5.3.5. Default Priorities

All streams are initially assigned a non-exclusive dependency on stream 0x0. Pushed streams (Section 8.2) initially depend on their associated stream. In both cases, streams are assigned a default weight of 16.



Non-browsers / proxies

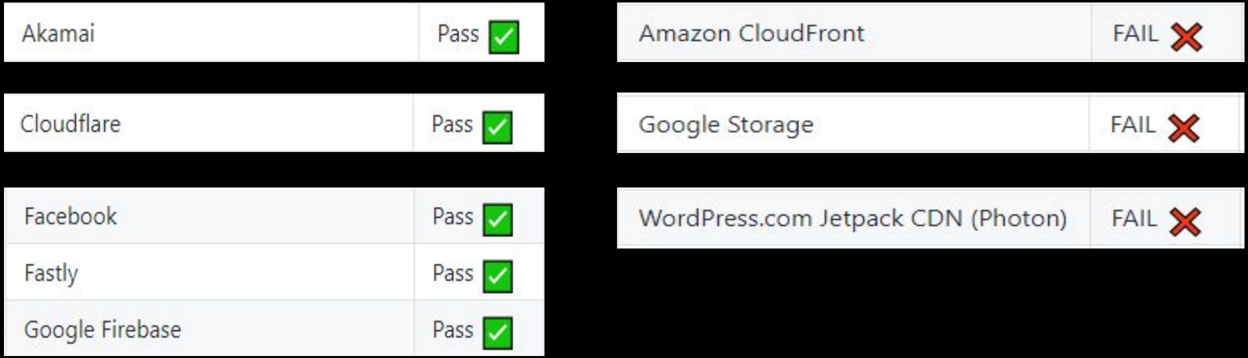
- nghttp2 - nghttp client based on FF model. Server algorithm based on h2o (slides).
- Node.js - defaults? Per-stream overridable.
- Go - defaults? Stream settings and priority write strategies.
- Python - ? Priority model for servers, based on h2o.
- Proxy software????

https://nghttp2.org/documentation/nghttp.1.html#dependency-based-priority
https://www.slideshare.net/kazuho/h2o-making-http-better
https://nodejs.org/api/http2.html#http2_http2stream_priority_options
https://godoc.org/golang.org/x/net/http2
https://python-hyper.org/projects/priority/en/latest/

CLOUDFLARE

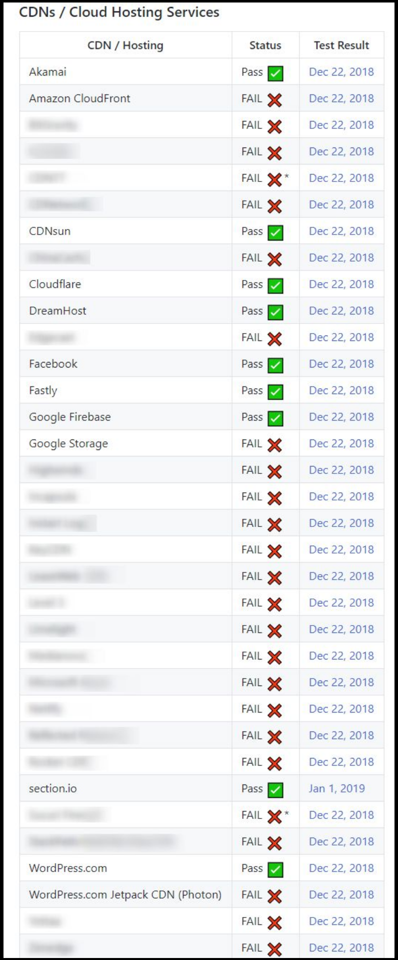
Servers

"Unfortunately not all servers are equal – some don't appear to implement prioritization and so serve responses on a 'first come, first served' basis."



Akamai	Pass ✓
Cloudflare	Pass ✓
Facebook	Pass ✓
Fastly	Pass ✓
Google Firebase	Pass ✓

Amazon CloudFront	FAIL ✗
Google Storage	FAIL ✗
WordPress.com Jetpack CDN (Photon)	FAIL ✗

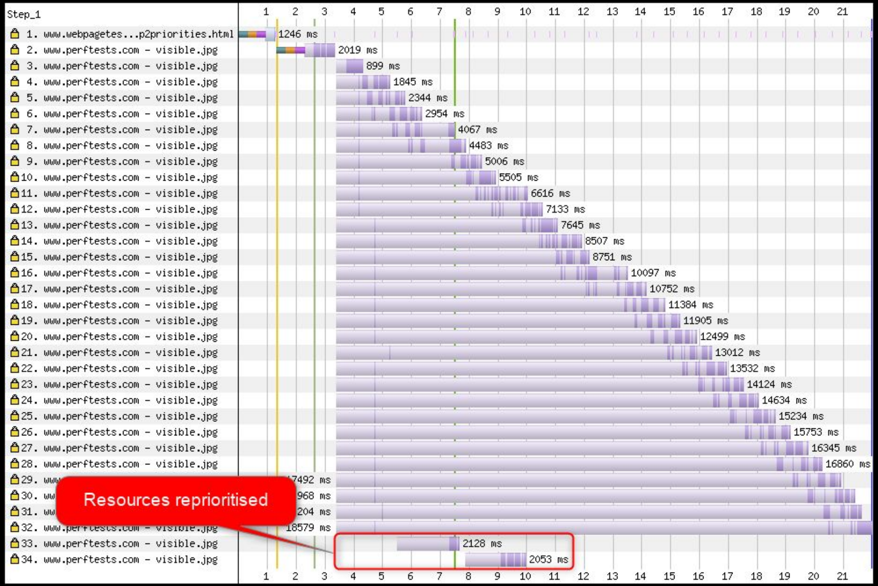


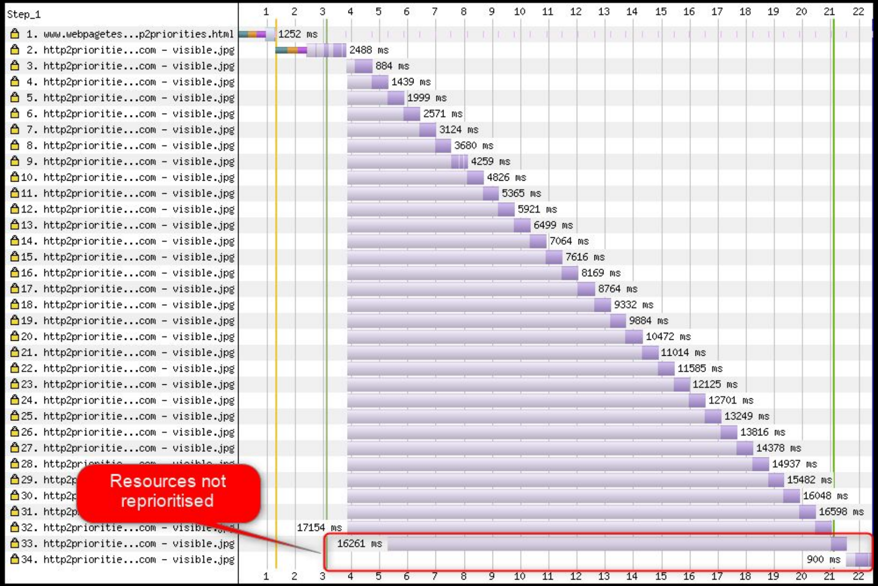
CDNs / Cloud Hosting Services

CDN / Hosting	Status	Test Result
Akamai	Pass ✓	Dec 22, 2018
Amazon CloudFront	FAIL ✗	Dec 22, 2018
[redacted]	FAIL ✗	Dec 22, 2018
[redacted]	FAIL ✗	Dec 22, 2018
[redacted]	FAIL ✗ *	Dec 22, 2018
[redacted]	FAIL ✗	Dec 22, 2018
CDNsun	Pass ✓	Dec 22, 2018
[redacted]	FAIL ✗	Dec 22, 2018
Cloudflare	Pass ✓	Dec 22, 2018
DreamHost	Pass ✓	Dec 22, 2018
[redacted]	FAIL ✗	Dec 22, 2018
Facebook	Pass ✓	Dec 22, 2018
Fastly	Pass ✓	Dec 22, 2018
Google Firebase	Pass ✓	Dec 22, 2018
Google Storage	FAIL ✗	Dec 22, 2018
[redacted]	FAIL ✗	Dec 22, 2018
[redacted]	FAIL ✗	Dec 22, 2018
[redacted]	FAIL ✗	Dec 22, 2018
[redacted]	FAIL ✗	Dec 22, 2018
[redacted]	FAIL ✗	Dec 22, 2018
[redacted]	FAIL ✗	Dec 22, 2018
[redacted]	FAIL ✗	Dec 22, 2018
[redacted]	FAIL ✗	Dec 22, 2018
[redacted]	FAIL ✗	Dec 22, 2018
[redacted]	FAIL ✗	Dec 22, 2018
[redacted]	FAIL ✗	Dec 22, 2018
[redacted]	FAIL ✗	Dec 22, 2018
section.io	Pass ✓	Jan 1, 2019
[redacted]	FAIL ✗ *	Dec 22, 2018
[redacted]	FAIL ✗	Dec 22, 2018
WordPress.com	Pass ✓	Dec 22, 2018
WordPress.com Jetpack CDN (Photon)	FAIL ✗	Dec 22, 2018
[redacted]	FAIL ✗	Dec 22, 2018
[redacted]	FAIL ✗	Dec 22, 2018

CLOUDFLARE

https://Ishttp2fastyet.com



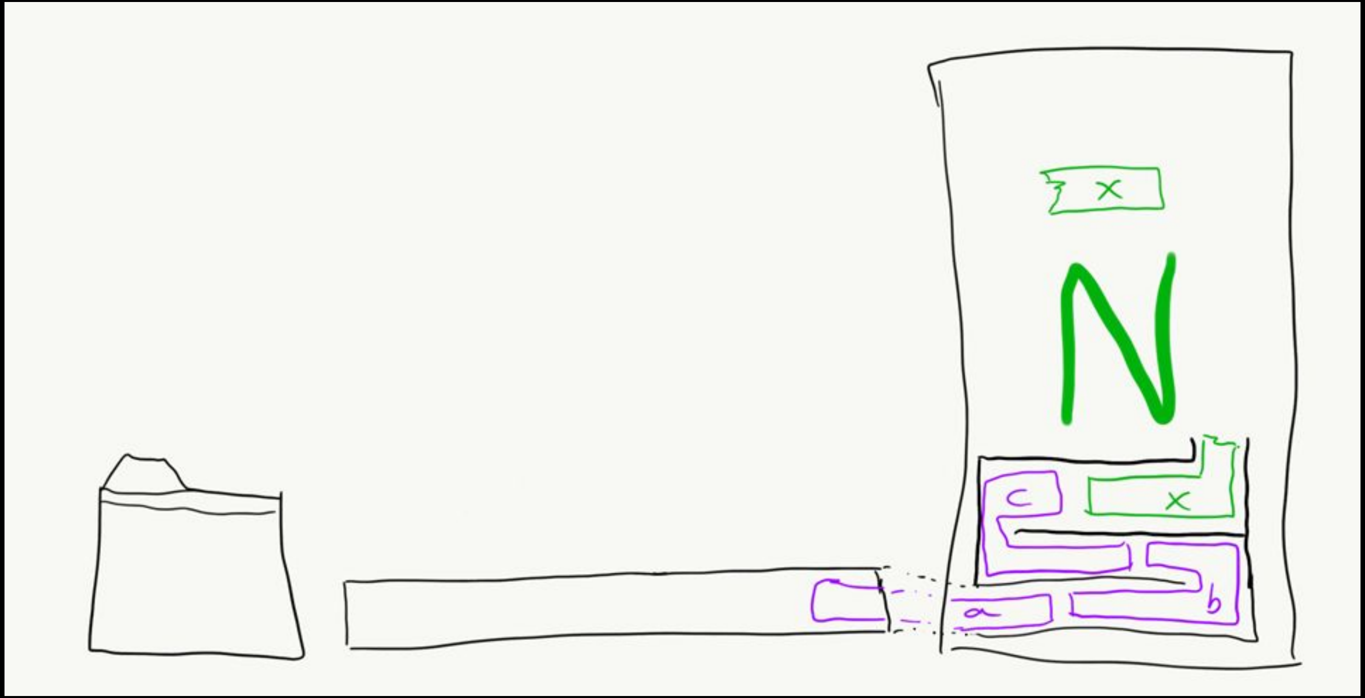


https://Ishttp2fastyet.com

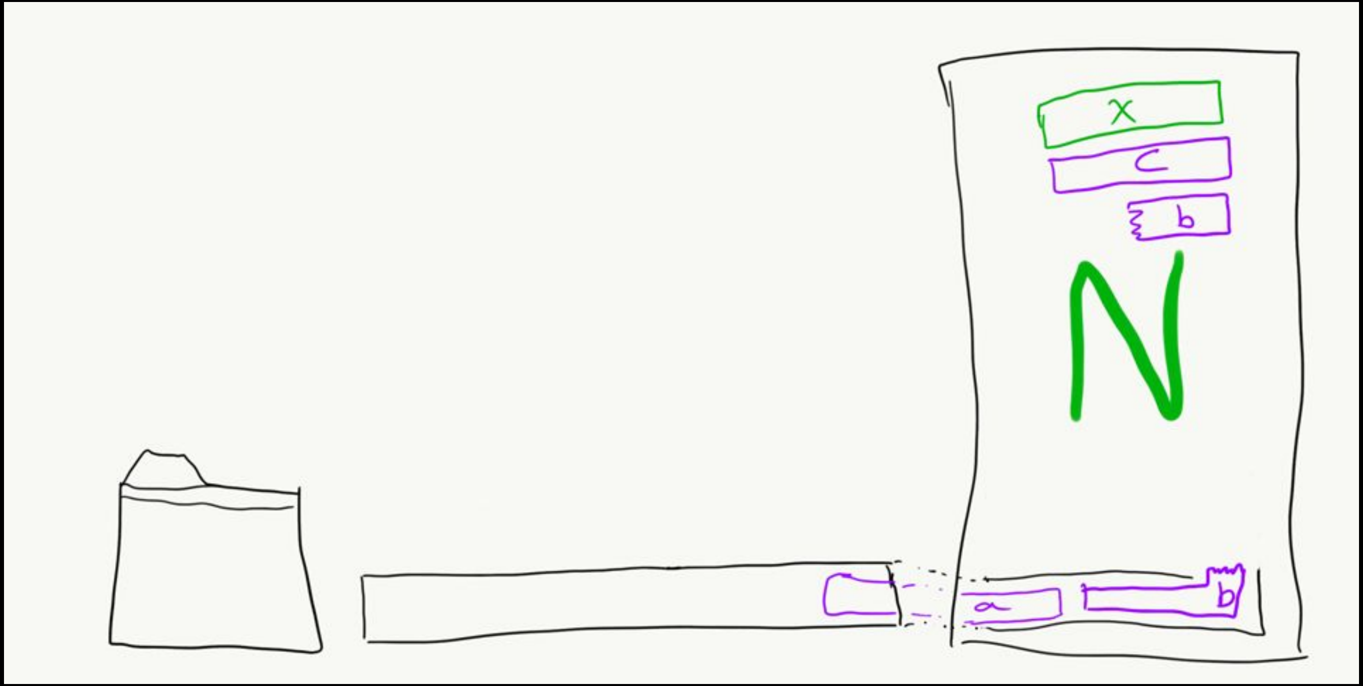
Reprioritization

https://blog.cloudflare.com/http-2-prioritization-with-nginx/

Buffering breaks reprioritization (BBR)

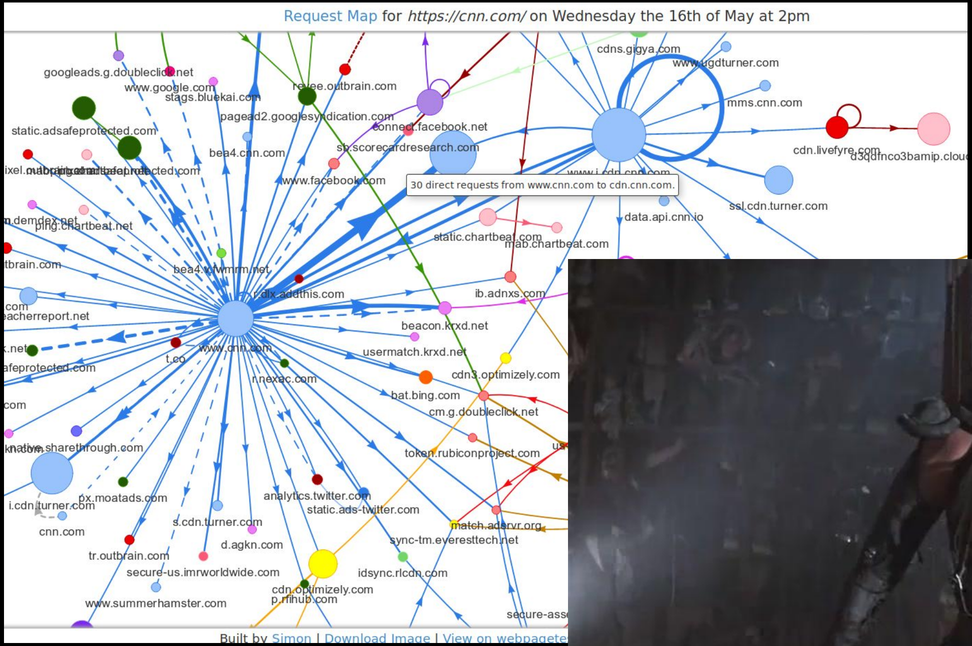


https://blog.cloudflare.com/http-2-prioritization-with-nginx/

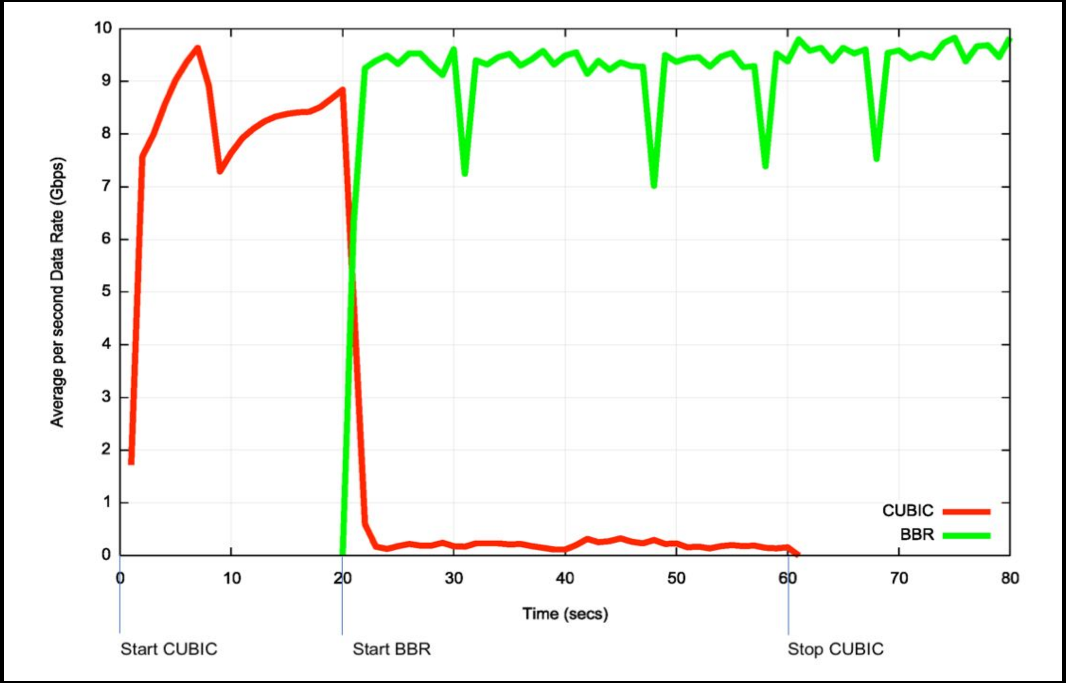


https://blog.cloudflare.com/http-2-prioritization-with-nginx/

Prioritizing across connections



Prioritizing across congestion control algorithms



https://labs.ripe.net/Members/gih/bbr-tcp

Shiny HTTP

Shiny HTTP

CLOUDFLARE

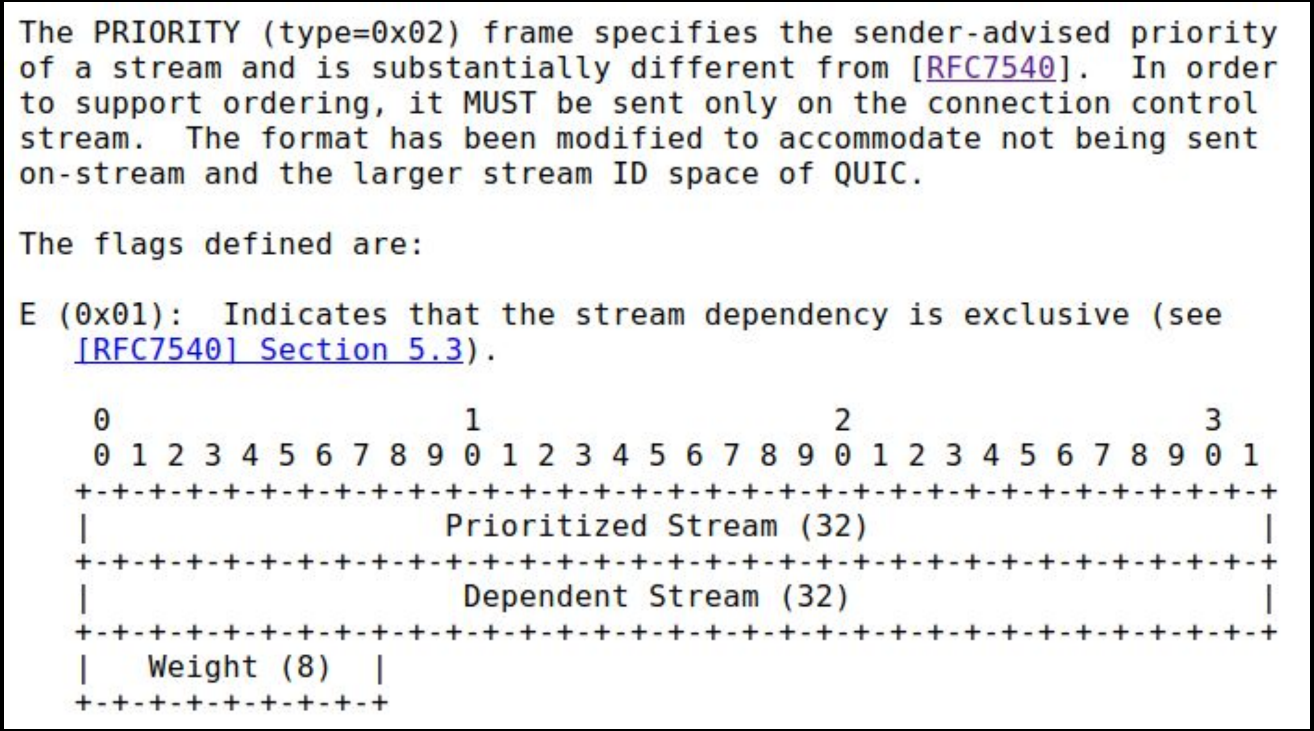
Priority in gQUIC and early IETF QUIC

HTTP/2-over-QUIC uses the HTTP/2 priority scheme described in RFC7540 Section 5.3.

ALTERNATIVE DESIGN: if the core QUIC protocol implements priorities, then this document should map the HTTP/2 priorities scheme to that provided by the core protocol. This would likely involve prohibiting the sending of HTTP/2 PRIORITY frames and setting of the PRIORITY flag in HTTP/2 HEADERS frames, to avoid conflicting directives.

https://tools.ietf.org/html/draft-shade-quic-http2-mapping-00#section-7

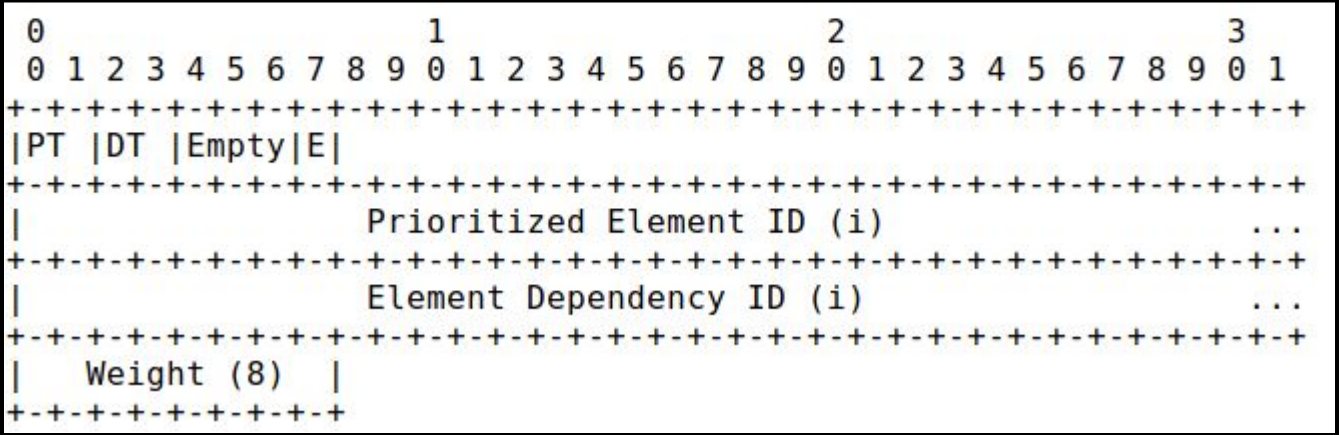
Priority in HTTP/3 -01



```
The PRIORITY (type=0x02) frame specifies the sender-advised priority
of a stream and is substantially different from [RFC7540].  In order
to support ordering, it MUST be sent only on the connection control
stream.  The format has been modified to accommodate not being sent
on-stream and the larger stream ID space of QUIC.

The flags defined are:

E (0x01):  Indicates that the stream dependency is exclusive (see
   [RFC7540] Section 5.3).

    0                   1                   2                   3
    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                   Prioritized Stream (32)                     |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                    Dependent Stream (32)                      |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |   Weight (8)  |
   +-+-+-+-+-+-+-+-+
```

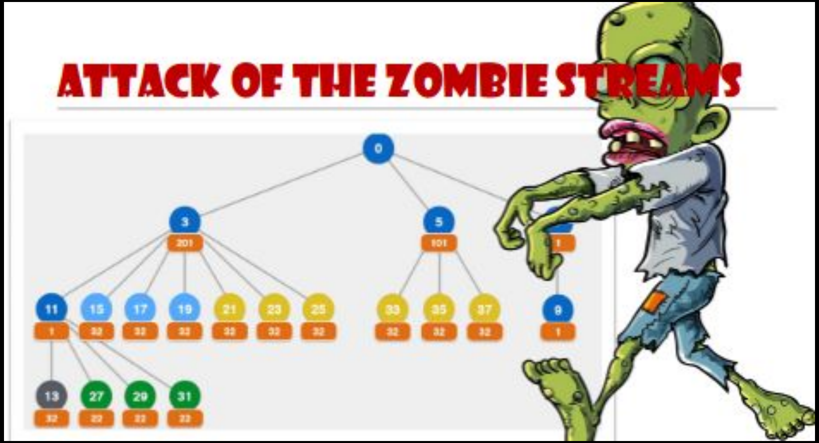
https://tools.ietf.org/html/draft-ietf-quic-http-01#section-5.2.3

Priority in HTTP/3 -13

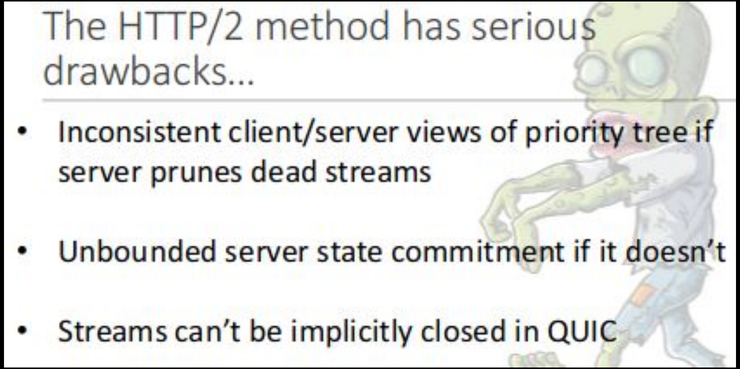


```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|PT |DT |Empty|E|
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                 Prioritized Element ID (i)                 ...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                 Element Dependency ID (i)                  ...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   Weight (8)  |
+-+-+-+-+-+-+-+-+
```

https://tools.ietf.org/html/draft-ietf-quic-http-13#section-4.2.4

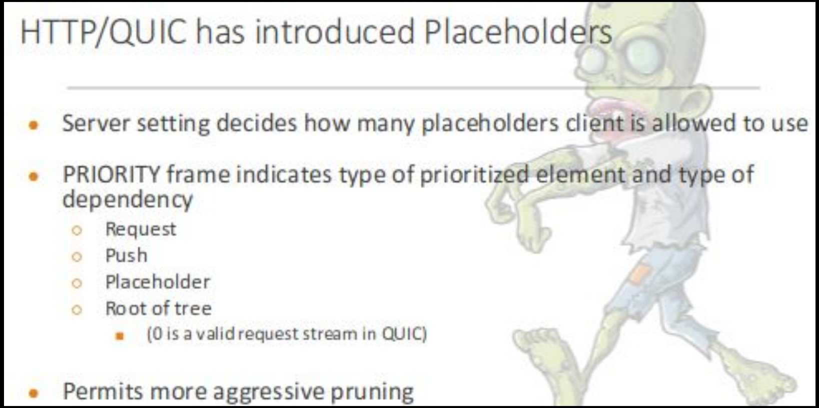


ATTACK OF THE ZOMBIE STREAMS



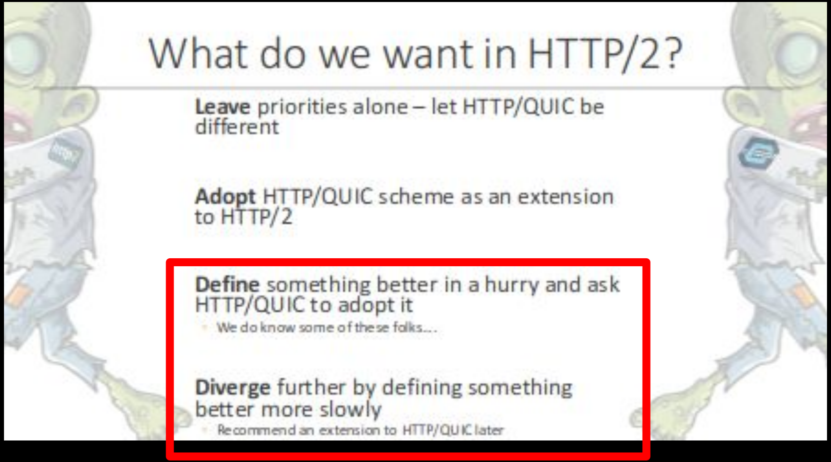
The HTTP/2 method has serious drawbacks...

- Inconsistent client/server views of priority tree if server prunes dead streams
- Unbounded server state commitment if it doesn't
- Streams can't be implicitly closed in QUIC



HTTP/QUIC has introduced Placeholders

- Server setting decides how many placeholders client is allowed to use
- PRIORITY frame indicates type of prioritized element and type of dependency
  - Request
  - Push
  - Placeholder
  - Root of tree
    - (0 is a valid request stream in QUIC)
- Permits more aggressive pruning



What do we want in HTTP/2?

Leave priorities alone – let HTTP/QUIC be different

Adopt HTTP/QUIC scheme as an extension to HTTP/2

Define something better in a hurry and ask HTTP/QUIC to adopt it
- We do know some of these folks...

Diverge further by defining something better more slowly
- Recommend an extension to HTTP/QUIC later

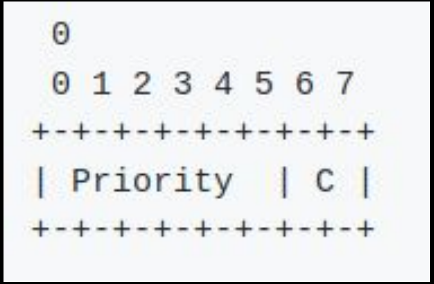
https://datatracker.ietf.org/meeting/102/materials/slides-102-httpbis-httpquic-cross-pollination-00

Alternative HTTP/3 scheme

- Proposal by Pat Meenan
  - https://github.com/pmeenan/http3-prioritization-proposal/blob/master/README.md
- List discussion
  - https://lists.w3.org/Archives/Public/ietf-http-wg/2019JanMar/0073.html

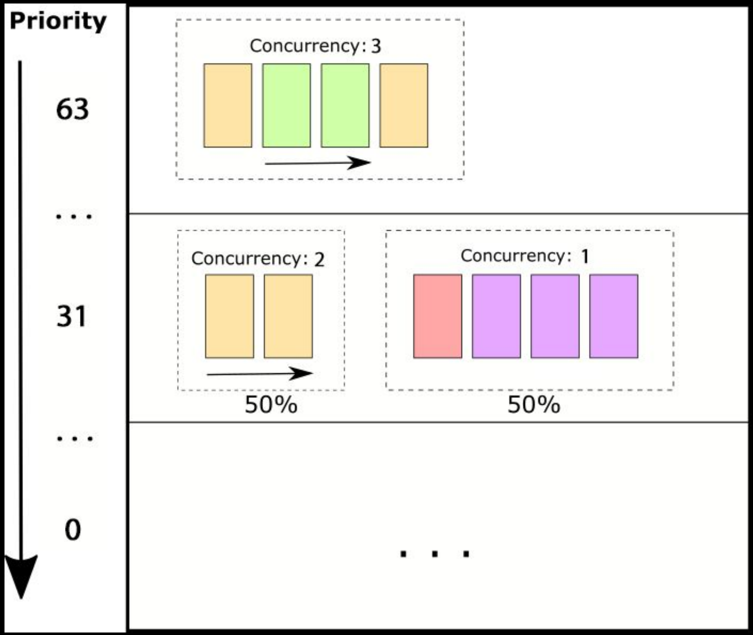
Goals

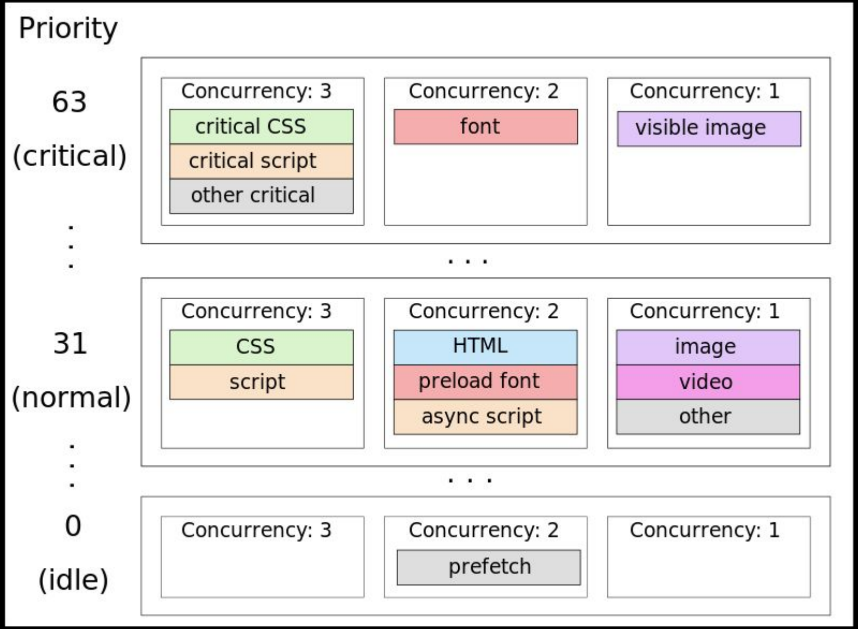
- A priority scheme that can provide the appropriate scheduling without needing the full context of other streams.
- Ordering of streams (more important delivered before lower importance).
- Specifying concurrency of download for requests to allow for both sequential and concurrent transfer of streams.
- Simple to implement for both clients and servers.



```
 0
 0 1 2 3 4 5 6 7
+-+-+-+-+-+-+-+-+
| Priority    | C |
+-+-+-+-+-+-+-+-+
```

Alternative HTTP/3 scheme





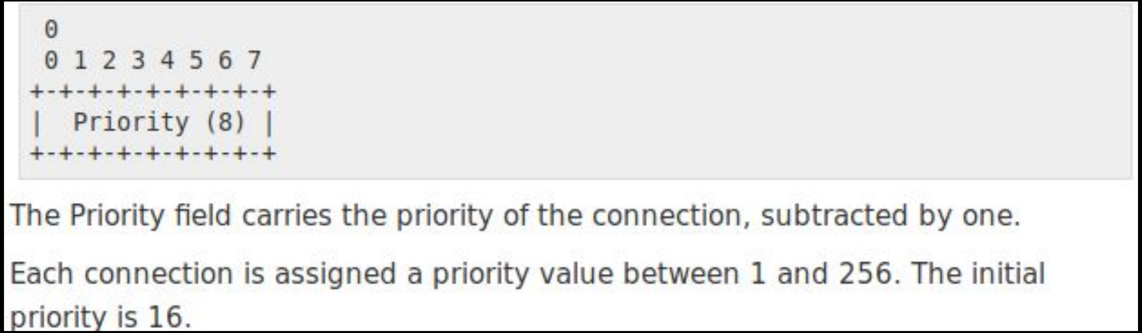
QUIC shared connection congestion control

Proposal by Kazuho Oku - https://kazuho.github.io/draft-kazuho-quic-shared-cc/draft-kazuho-quic-shared-cc.html

Based on work by Erik Sy - https://mailarchive.ietf.org/arch/msg/quic/0oew7O36giudo4EiXPct5f7-Bts

Share resources across connections between two endpoints

Multiple HTTP/3 connections or different QUIC application mappings - e.g. HTTP/3 & DNS over QUIC



```
 0
 0 1 2 3 4 5 6 7
+-+-+-+-+-+-+-+-+
|  Priority (8) |
+-+-+-+-+-+-+-+-+
```

The Priority field carries the priority of the connection, subtracted by one.

Each connection is assigned a priority value between 1 and 256. The initial priority is 16.

CLOUDFLARE project: *Edge Driven HTTP2 Prioritisation*

CLOUDFLARE project: *Edge Driven HTTP2 Prioritisation*

1. Allow site owners to influence priority

CLOUDFLARE project: *Edge Driven HTTP2 Prioritisation*

1. Allow site owners to influence priority

2. Experimental strategies for response data ordering

CLOUDFLARE project: *Edge Driven HTTP2 Prioritisation*

1. Allow site owners to influence priority

2. Experimental strategies for response data ordering

3. Future support for dynamic re-prioritisations

1. Allow site owners to influence priority
    - Forwarding browser priority hints to an Edge Worker
    - Edge Worker API for overriding priority of the response
2. Experimental strategies for response data ordering

3. Future support for dynamic re-prioritisations

CLOUDFLARE

1. Allow site owners to influence priority
   - Forwarding browser priority hints to an Edge Worker
   - Edge Worker API for overriding priority of the response
2. Experimental strategies for response data ordering
   - Define prioritisation options as a choice of concurrency strategies
   - Traversal of available data to apply chosen strategy
3. Future support for dynamic re-prioritisations

CLOUDFLARE

1. Allow site owners to influence priority
    - Forwarding browser priority hints to an Edge Worker
    - Edge Worker API for overriding priority of the response
2. Experimental strategies for response data ordering
    - Define prioritisation options as a choice of concurrency strategies
    - Traversal of available data to apply chosen strategy
3. Future support for dynamic re-prioritisations
    - Origin driven: re-prioritisation byte stream offsets: progressive quality images
    - Eyeball driven: update priority frame for in-viewport images

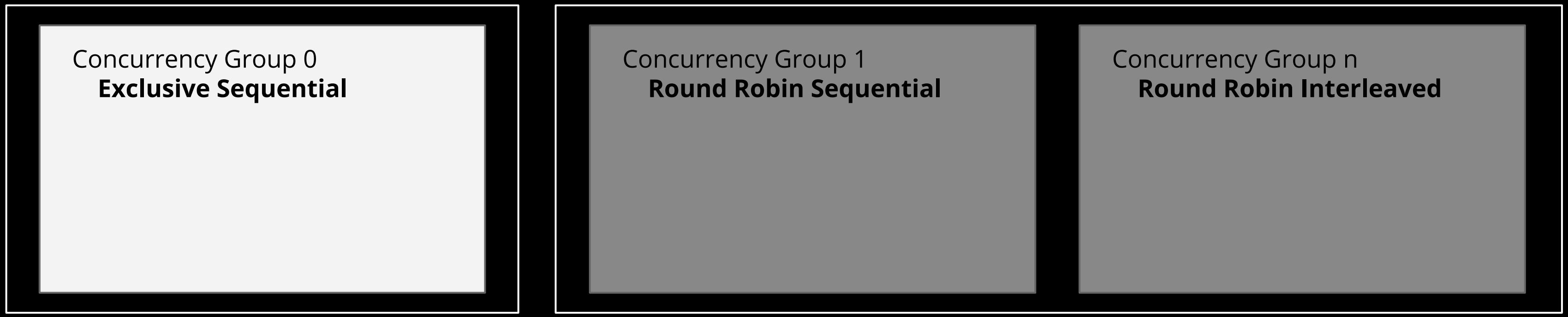
CLOUDFLARE

1. Allow site owners to influence priority
    - Forwarding browser priority hints to an Edge Worker
    - Edge Worker API for overriding priority of the response
2. Experimental strategies for response data ordering
    - Define prioritisation options as a choice of concurrency strategies
    - Traversal of available data to apply chosen strategy
3. Future support for dynamic re-prioritisations
    - Origin driven: re-prioritisation byte stream offsets: progressive quality images
    - Eyeball driven: update priority frame for in-viewport images

Describe prioritisation as concepts instead of simply using numbers

CLOUDFLARE

Prioritisation Concurrency Strategies



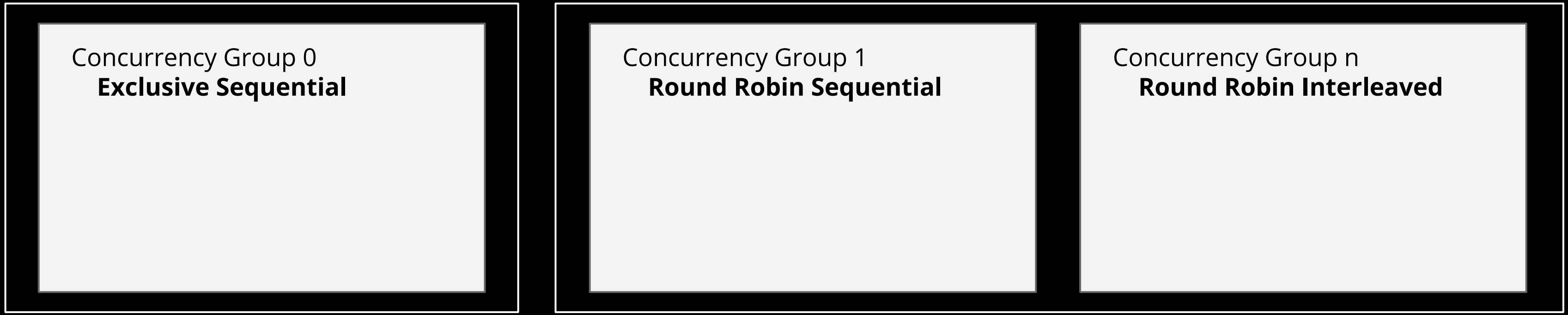
Exclusive

Will use all bandwidth for **available** frames

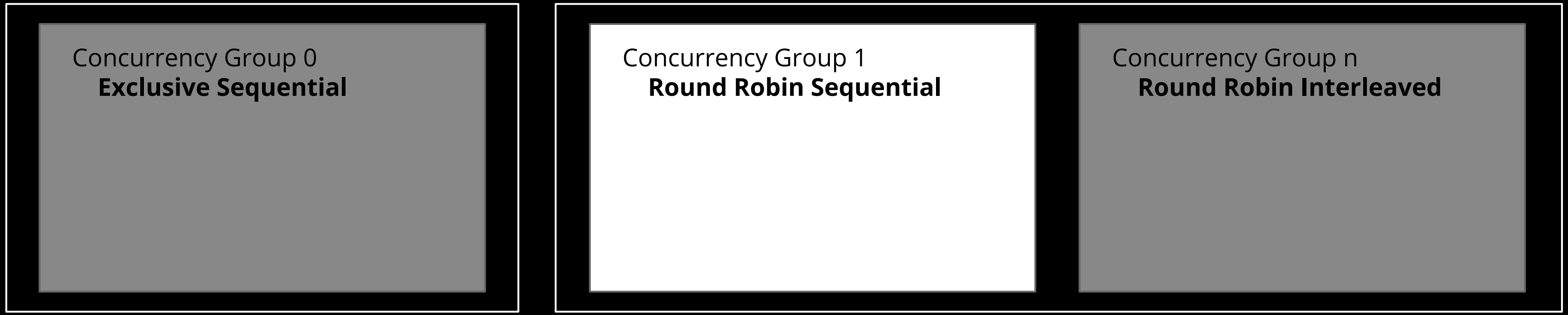
Sequential

Will have all **available** frames written before moving on to next stream in the same group

Prioritisation Concurrency Strategies



Prioritisation Concurrency Strategies



Round Robin

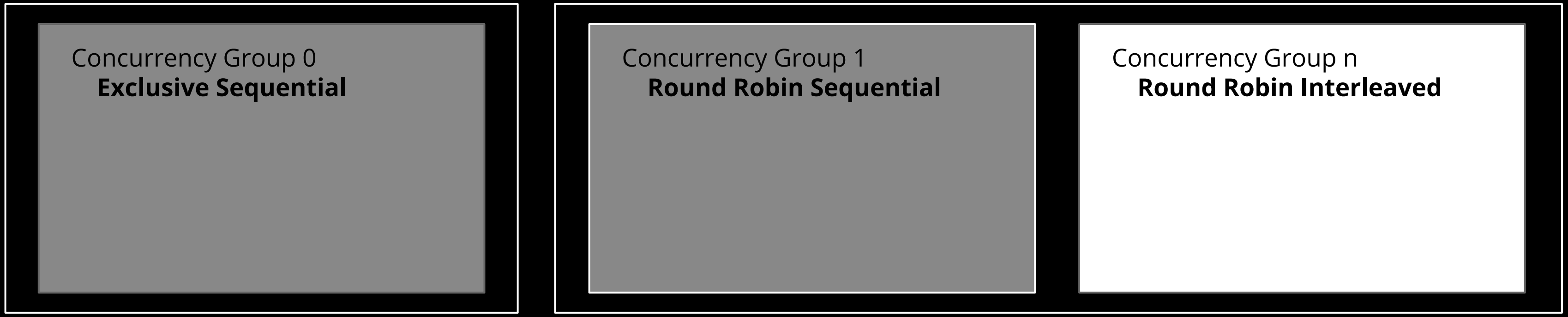
Will use 50% of non-exclusive bandwidth

Sequential

Will have all **available** frames written before moving on to next stream in the same group

CLOUDFLARE

Prioritisation Concurrency Strategies



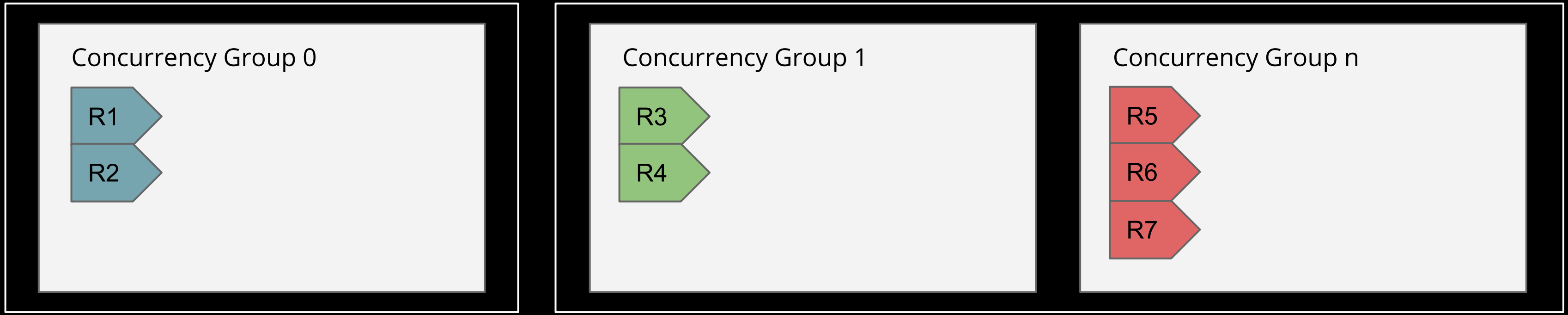
Round Robin

Will use 50% of non-exclusive bandwidth

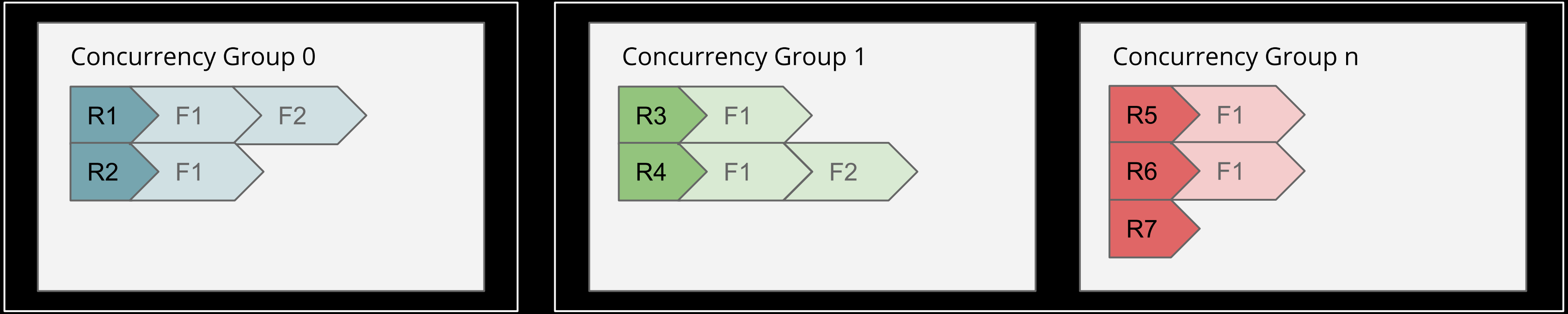
Interleaved

Iterate through all streams in the group, taking one **available** frame from each

Prioritisation Concurrency Strategy **Demonstration**

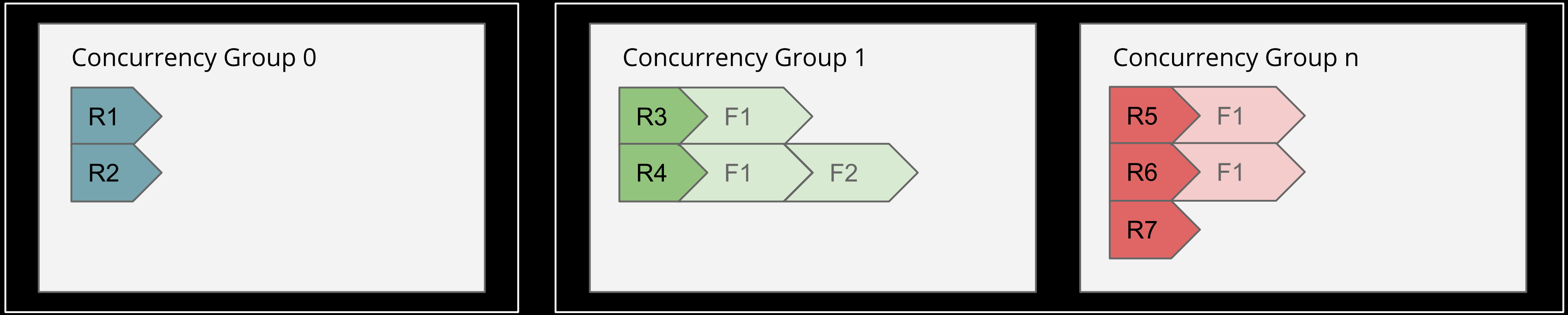


Prioritisation Concurrency Strategy **Demonstration**

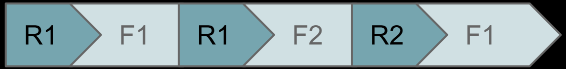


Cycle 1

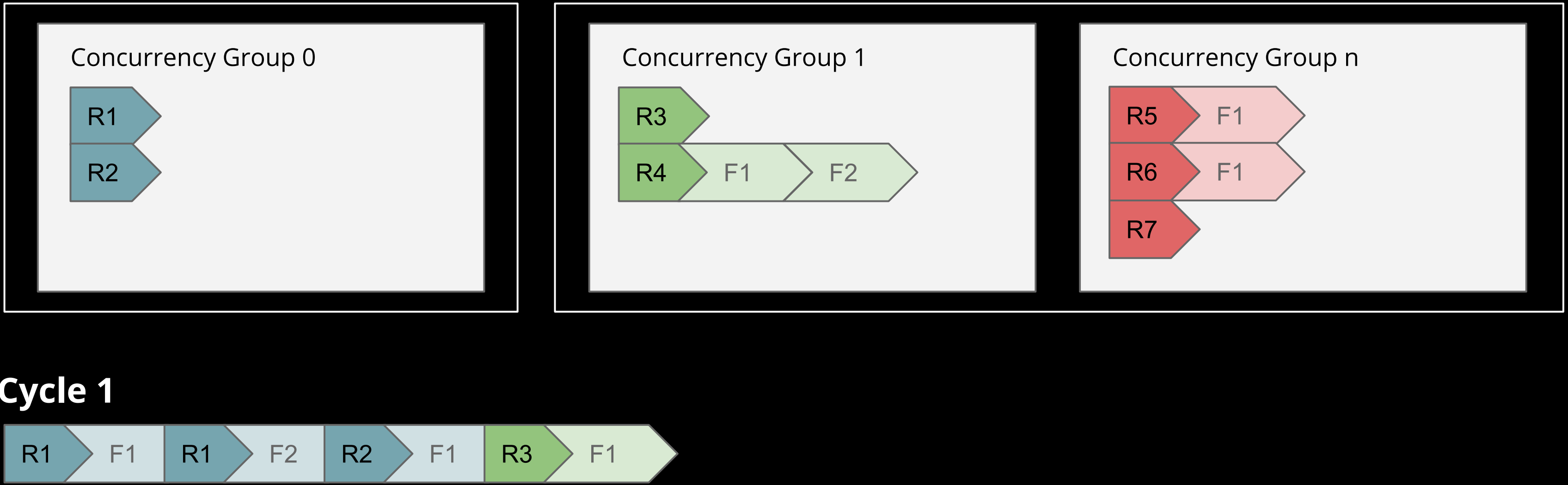
Prioritisation Concurrency Strategy **Demonstration**



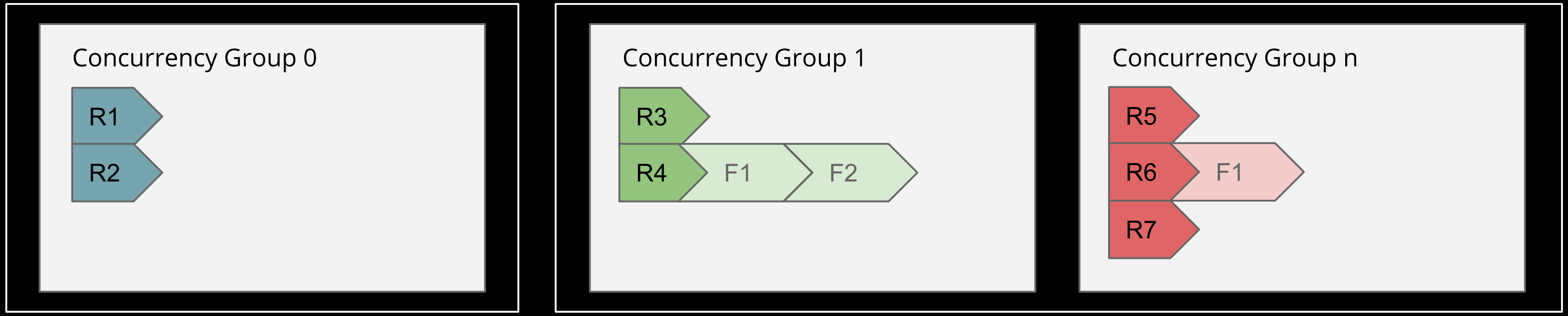
Cycle 1



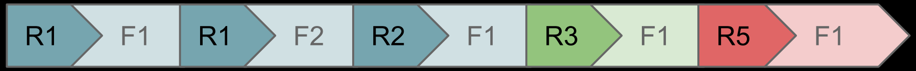
Prioritisation Concurrency Strategy **Demonstration**



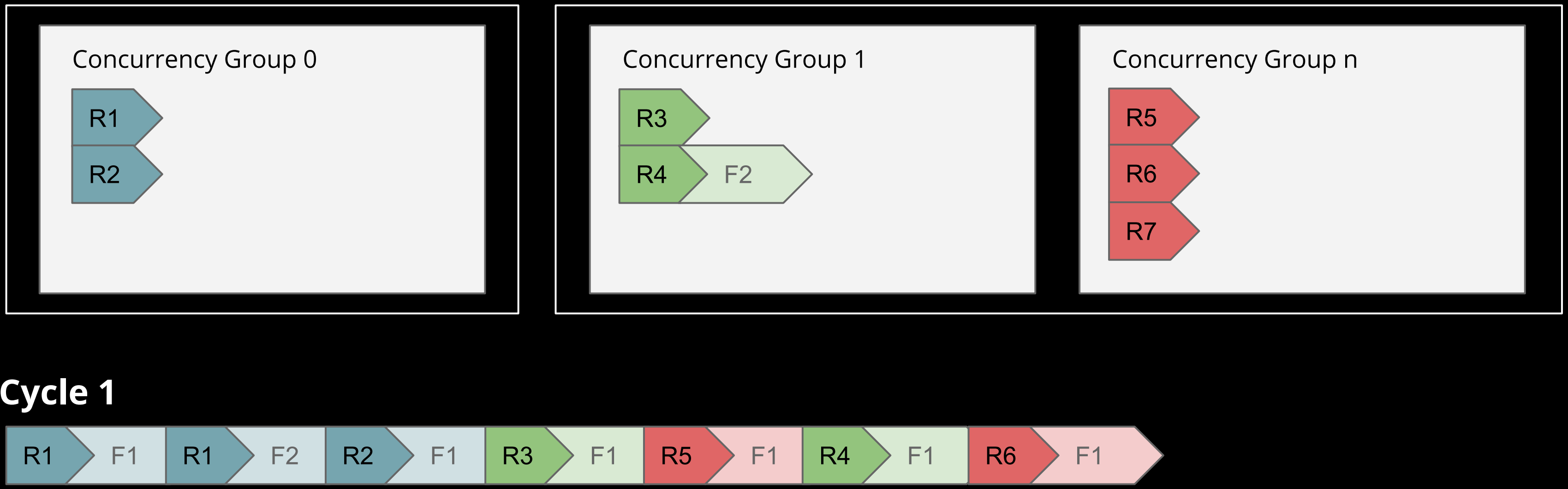
Prioritisation Concurrency Strategy **Demonstration**



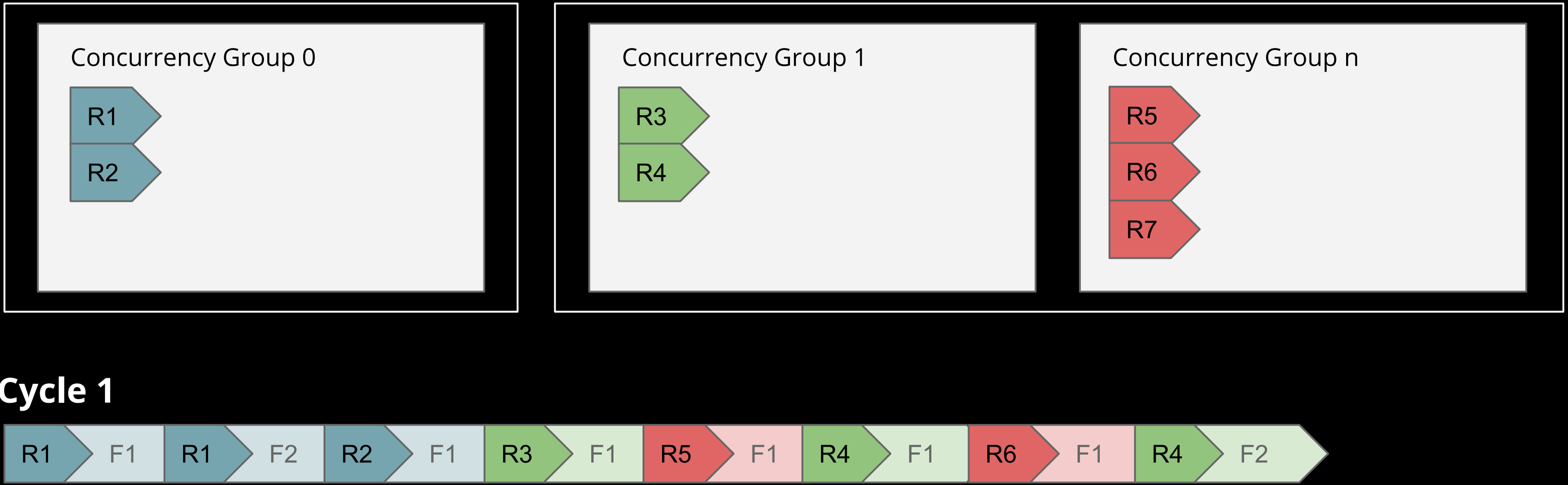
Cycle 1



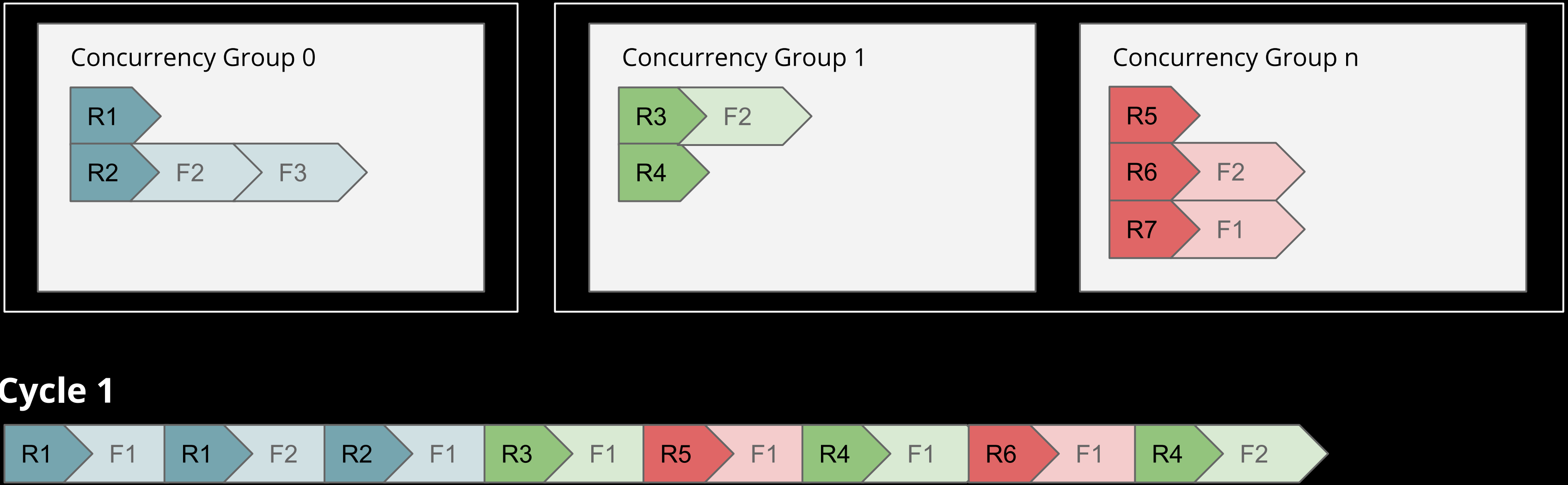
Prioritisation Concurrency Strategy **Demonstration**



Prioritisation Concurrency Strategy **Demonstration**



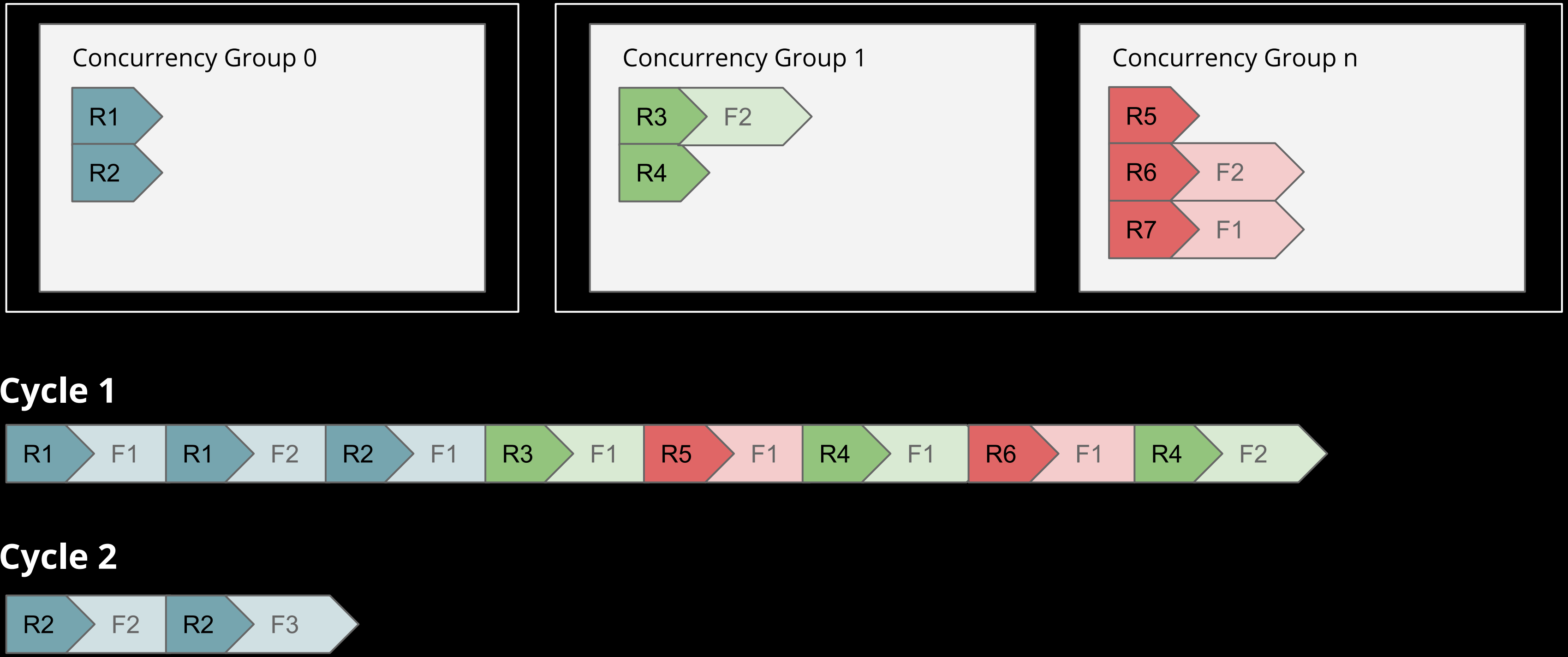
Prioritisation Concurrency Strategy **Demonstration**



Cycle 2

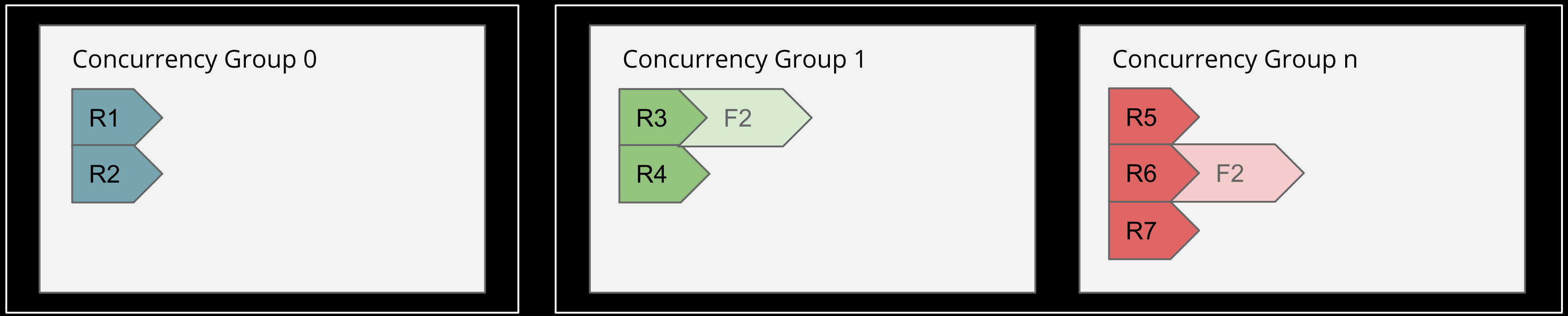
CLOUDFLARE

Prioritisation Concurrency Strategy **Demonstration**

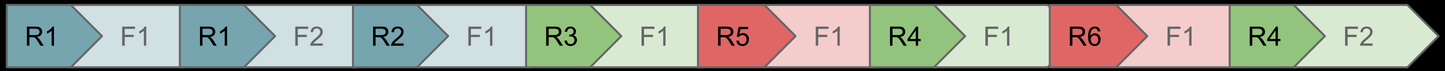


CLOUDFLARE

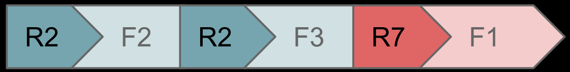
Prioritisation Concurrency Strategy **Demonstration**



Cycle 1

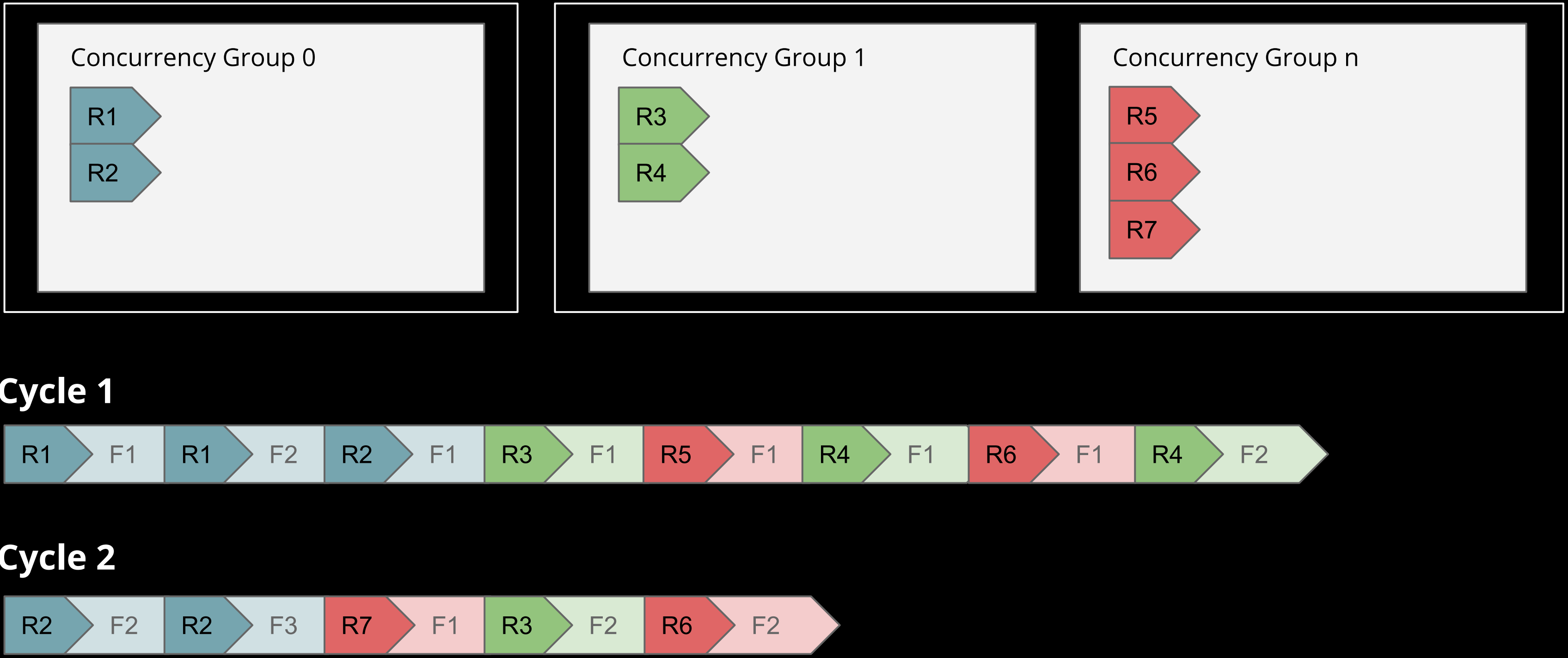


Cycle 2



CLOUDFLARE

Prioritisation Concurrency Strategy **Demonstration**

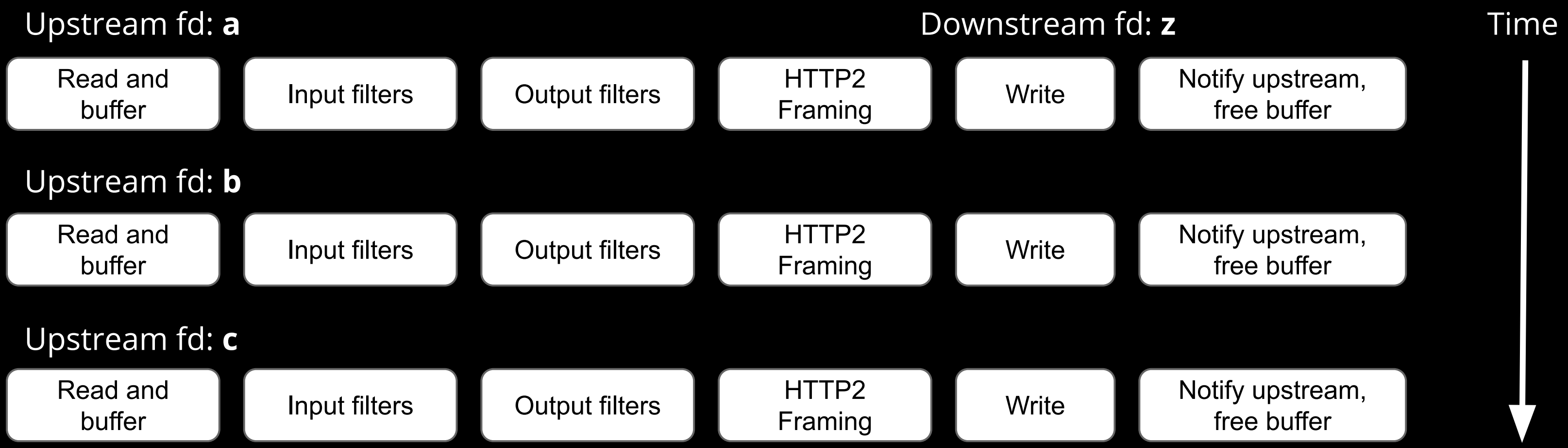


CLOUDFLARE

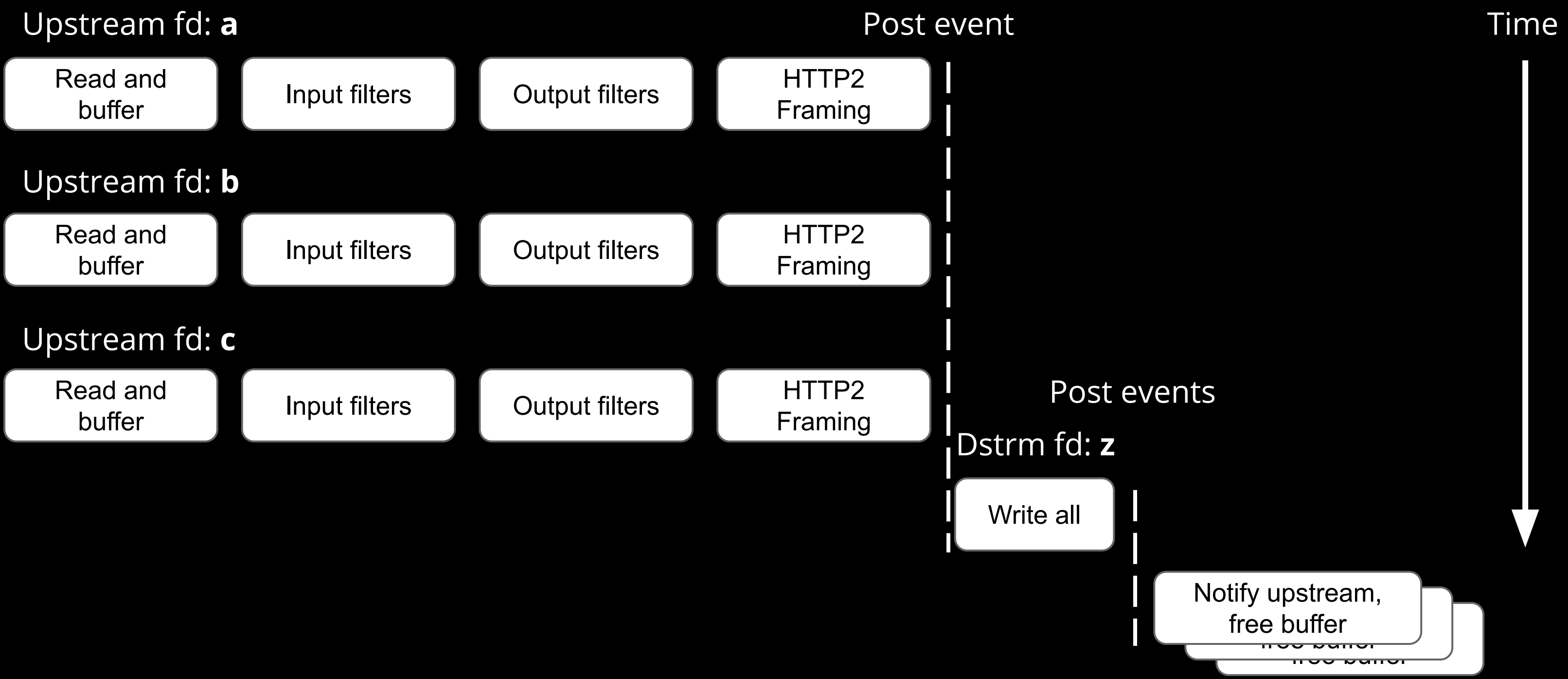
*"All is **not** as it seems."*

-Nick Jones - HTTP Workshop - Amsterdam 2019

Upstream read event processing in NGINX **(CURRENT)**



Upstream read event processing in NGINX **(REORDERED)**



Other structural aspects to consider

- Frame size policy (different for Exclusive vs Round Robin)
- Partial write ‘reclaim’
- Upstream request
    - Order of request commencement
    - Resources devoted to requests (buffer sizes, socket options)
    - Order of processing socket reads/writes
- Priority could influence many aspects of request processing, not just order of writes

Closing

Priorities are hard.

Did we get it right in design?

Did anyone bother implementing it?

Is it time to repaint the bike shed?