# Scheduling HTTP streams

**Evgeny Mekhanik, Alexander Krizhanovsky, Konstantin Tatar**

Tempesta Technologies, Inc.

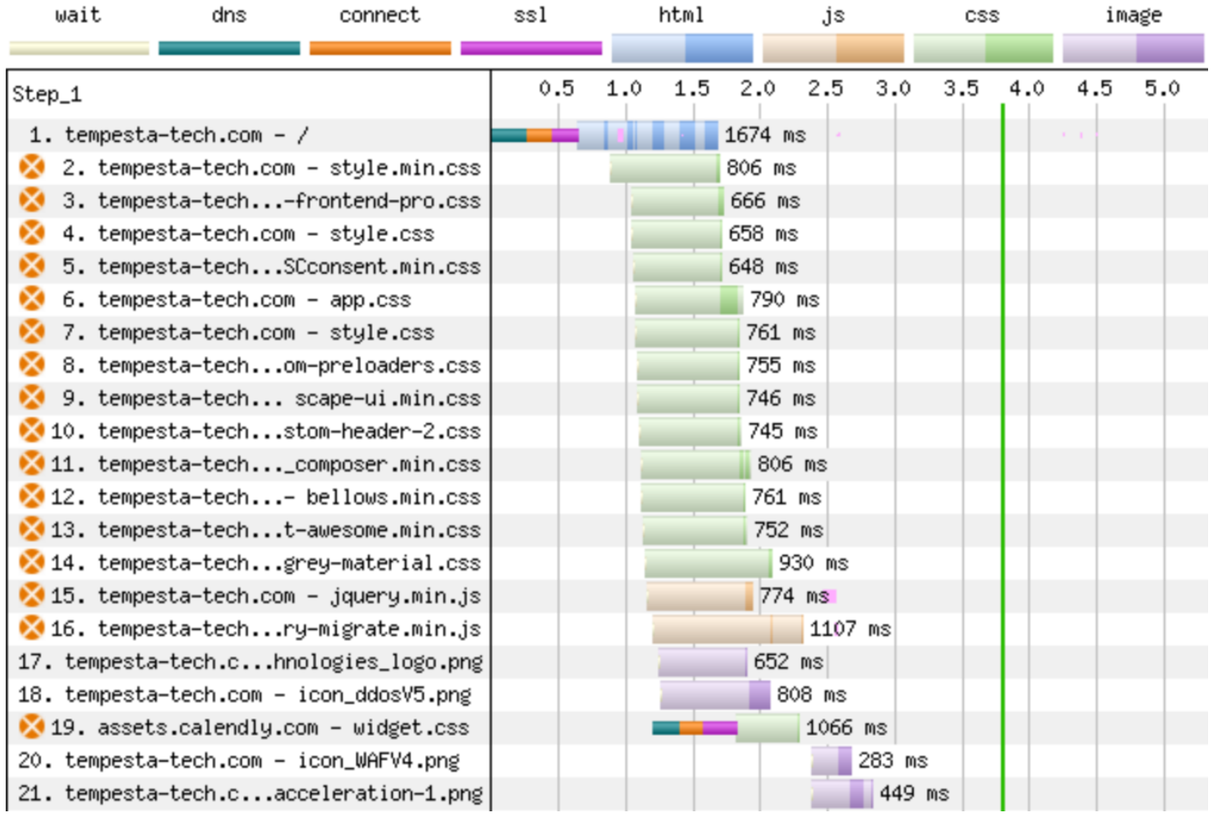
*{em,ak,kt}@tempesta-tech.com*

# HTTP streams

- *https://www.webpagetest.org/*

| TTFB | Start Render | FCP | Speed Index | LCP | CLS | TBT | DC Time | DC Requests | DC Bytes | Total Time |
|---|---|---|---|---|---|---|---|---|---|---|
| .846s | 3.800s | 10.929s | 11.021s | 10.929s | .002 | 3.716s | 10.312s | 85 | 3,969KB | 10.669s |

- One TCP connection (HTTP/2)
- Shared by tens HTTP streams
- CSS stream **depends** on an HTML stream
- CSS stream has higher **priority** than a JPG stream

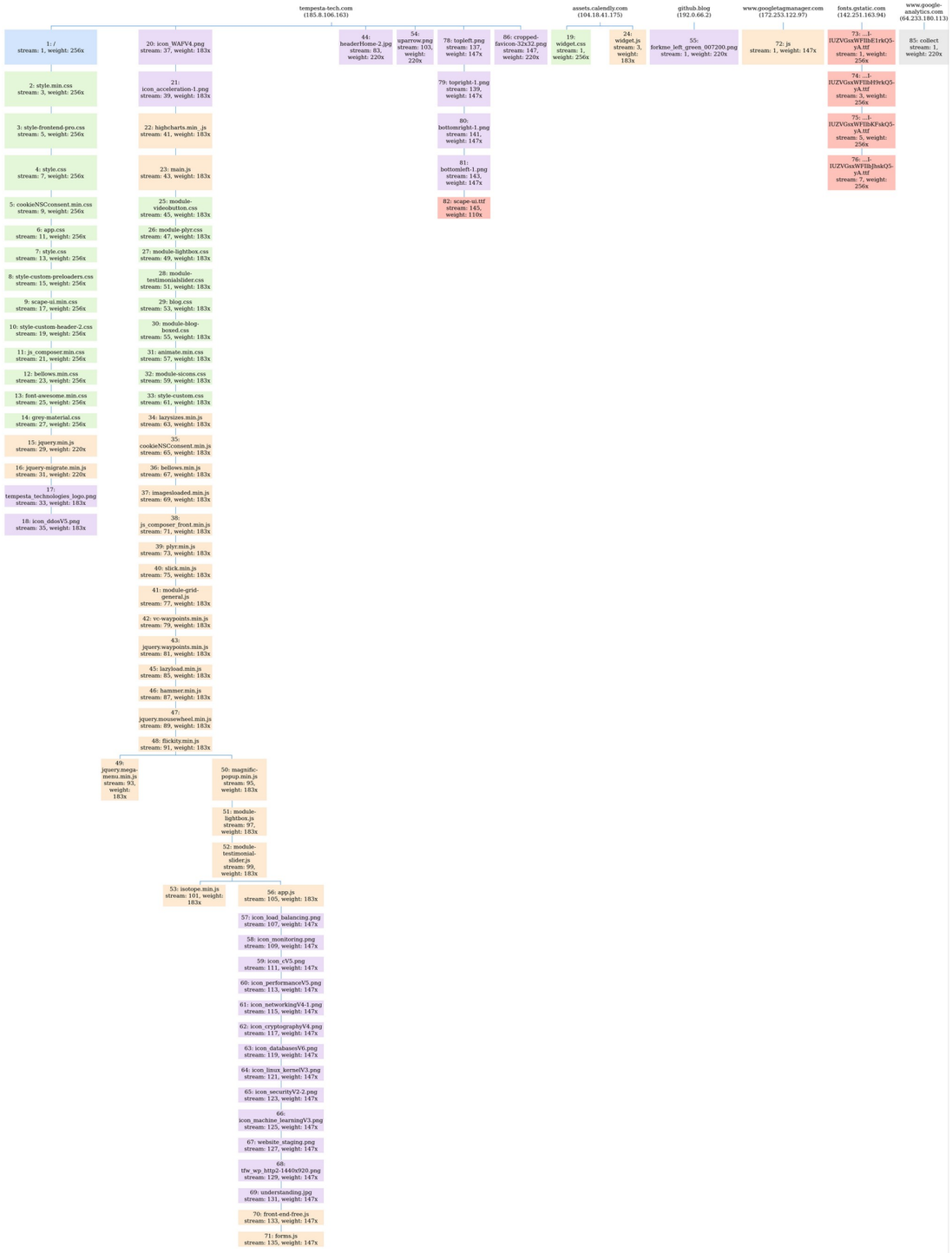


# HTTP streams prioritization in late ‘23

- Well studied
  - “Of the Utmost Importance: Resource Prioritization in HTTP/3 over QUIC”, R.Marx et all, 2019
  - “Resource Multiplexing and Prioritization in HTTP/2 over TCP versus HTTP/3 over QUIC”, R.Marx et al, 2020
- RFC 7540 is supported by all major web clients and servers
- RFC 9218 (June ‘22) is also supported by all web clients and servers
- All HTTP/3 – RFC 9218
- All HTTP/2 – RFC 7540

# HTTP streams trees

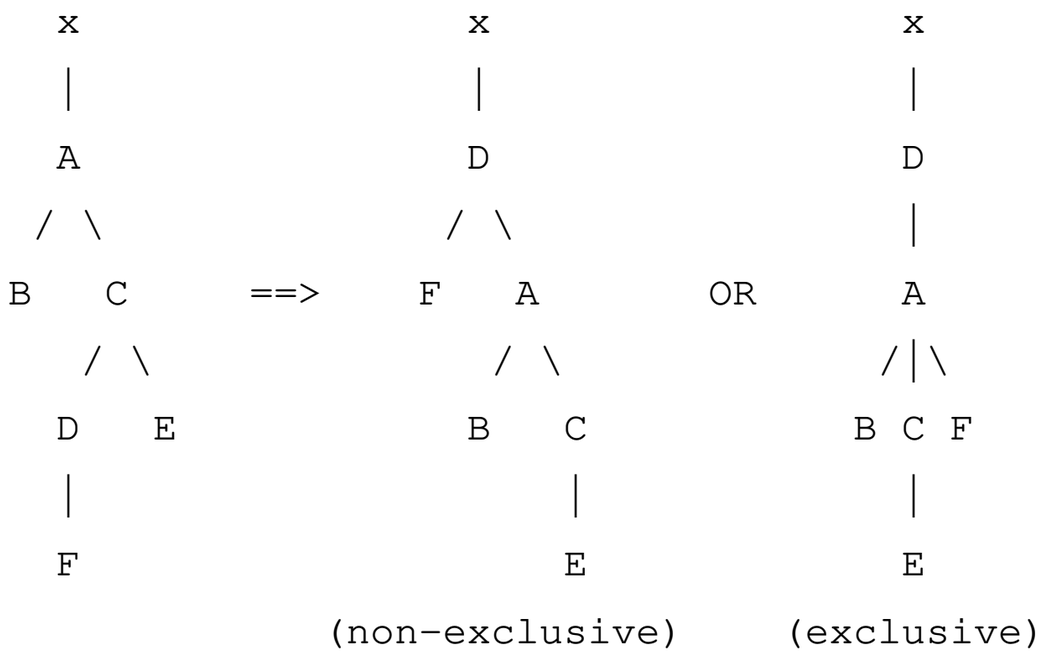
- RFC 7540: dependency tree
  - control (dummy) stream 0 is the root
- Priority queue on the same tree level
  - streams have [1, 256] **weight** or [1, 7] **urgency**
  - **re**prioritization – dependency and weight recomputation & reinsertion
  - **idle** (new) streams are in the tree, but can be removed at once
  - **out-of-priority**: upstream responses may arrive on their own order



# Reprioritization

- `PRIORITY` frame
- Weight change
- Dependency tree can be reconstructed on any state
  - browsers change priorities of streams
- Sharing bandwidth:
  - RFC 7540: **exclusive** streams
  - RFC 9218: and **incremental** flag

- Stream `A` is made dependent on stream `D`



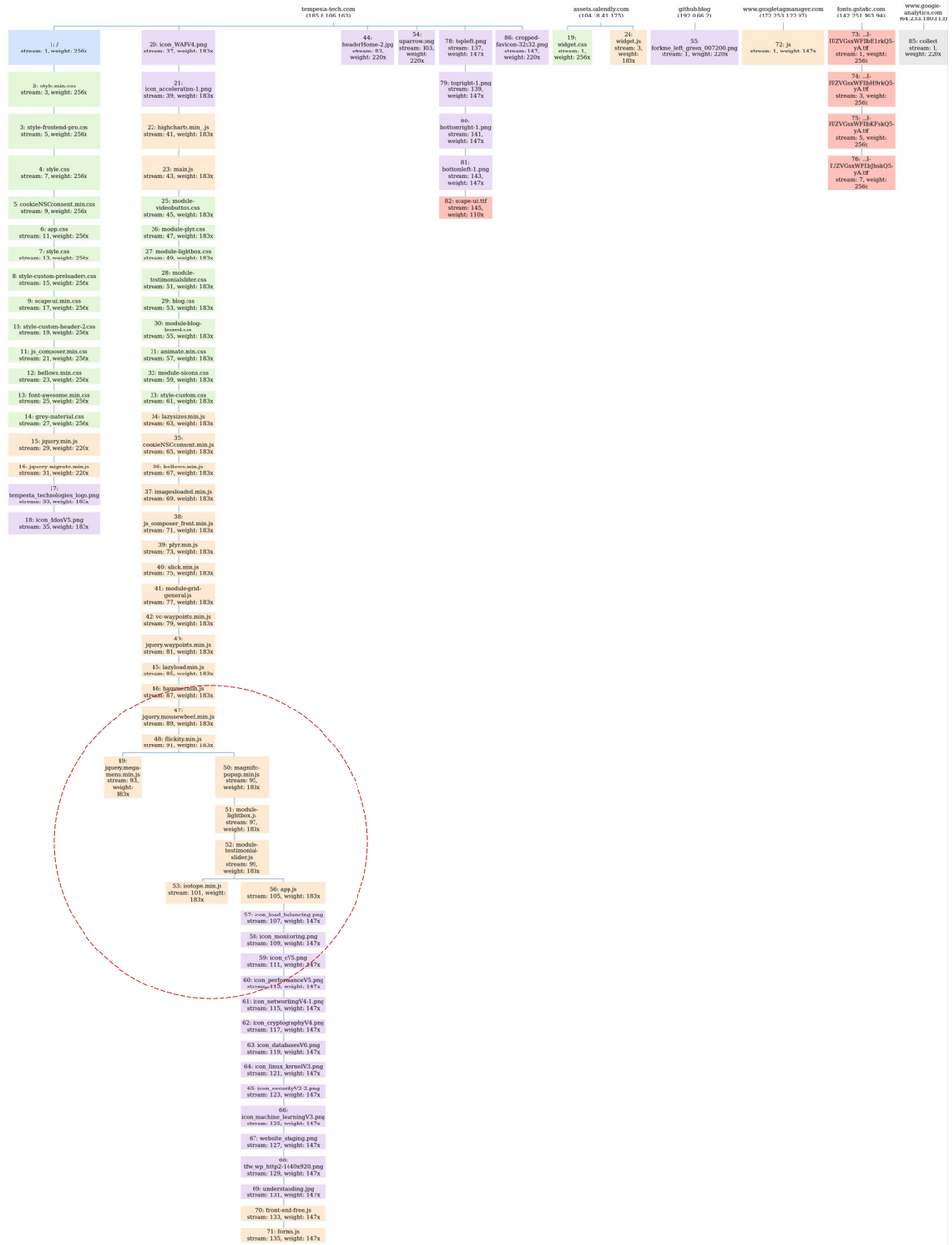

Tempesta Technologies

# Firefox 125: RFC 7540 - dependency tree

```
Create new stream id 3, weight 0, excl 0, depends on 0
Change stream 3 dependency: prev depends on 0, now depends on 0, new weight 201, excl 0

Create new stream id 5, weight 0, excl 0, depends on 0
Change stream 3 dependency: prev depends on 0, now depends on 0, new weight 101, excl 0

Create new stream id 7, weight 0, excl 0, depends on 0
Change stream 7 dependency: prev depends on 0, now depends on 0, new weight 1, excl 0

Create new stream id 9, weight 0, excl 0, depends on 0
Change stream 9 dependency: prev depends on 0, now depends on 7, new weight 1, excl 0

Create new stream id 11, weight 0, excl 0, depends on 0
Change stream 11 dependency: prev depends on 0, now depends on 3, new weight 1, excl 0

Create new stream id 13, weight 0, excl 0, depends on 0
Change stream 13 dependency: prev depends on 0, now depends on 0, new weight 241, excl 0
```

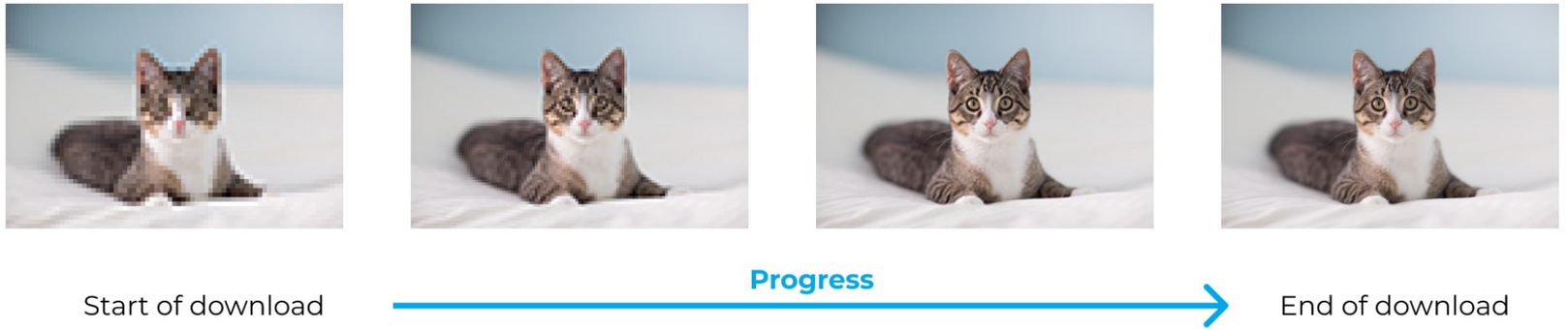
# What priority actually means?

- All the streams are **exclusive**
- Yes, even **siblings** with the same parent
  - **RFC contradiction**
  - seems `webpagetest.org` bug
  - a new exclusive dependent stream evicts previous exclusive dependent one
- **Weights don’t mean anything,** except
  - progressive JPEG
  - Firefox
  - nghttp2/h2load & web API**(?)**




Tempesta Technologies

# Progressive JPEG

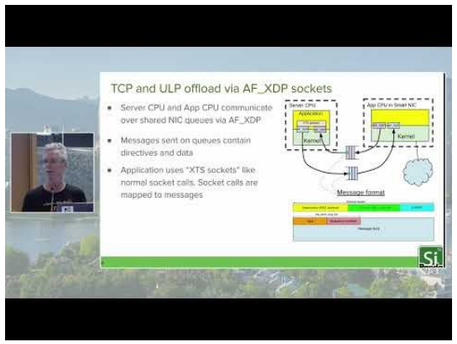



- The popular many-images resources use non-progressive JPG or WEBP

**Facebook**

**images.google.com**

**youtube.com**

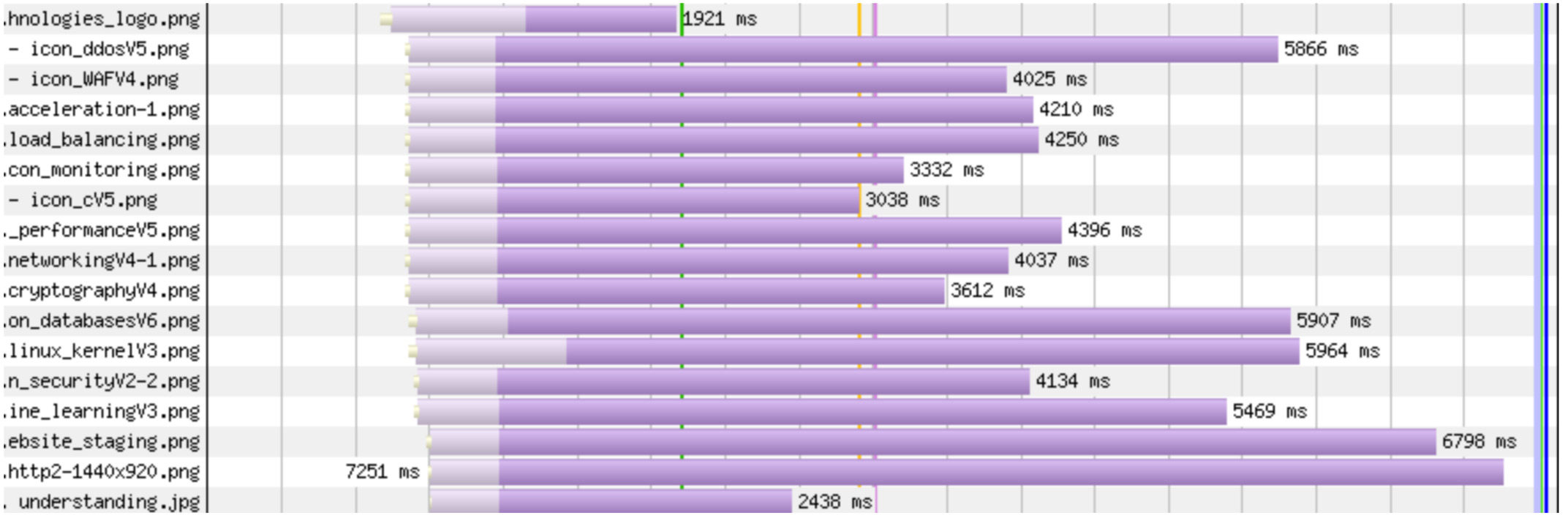



**ebay.com**

Tempesta Technologies

# Firefox streams prioritization

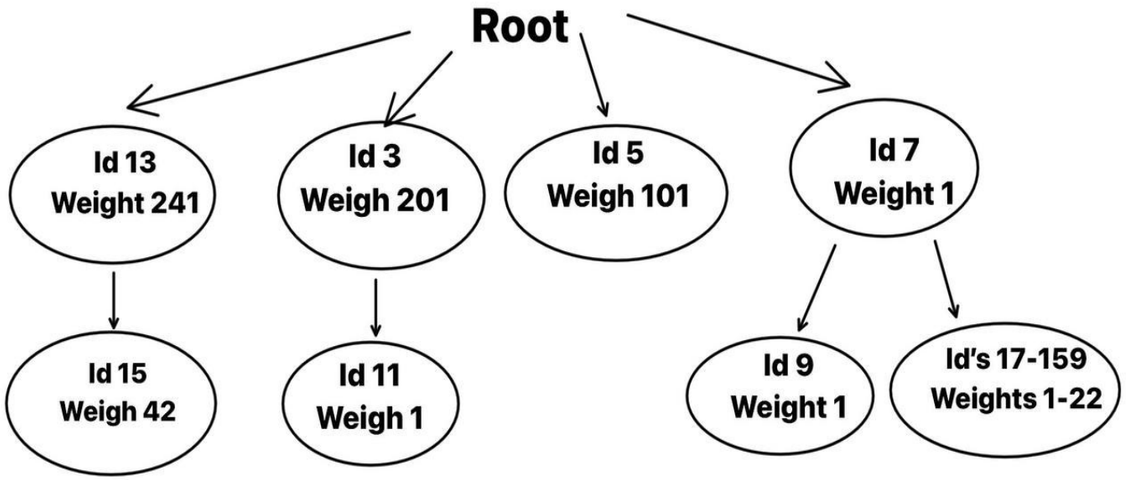
- Doesn’t use `EXCLUSIVE` flag (vs Chrome, Safari, Edge)
- Not so much better than fair round-robin




Tempesta Technologies

# Firefox dependency tree building

- Create idle streams as a nodes in dependency tree
  - to avoid race on adding a new stream and its parent deletion
  - unique for Firefox
  - idle forever
- Use weights to specify streams priority
- Sends more reprioritization frames than in Chrome




Streams 3, 5, 7, 13 are idle.

Tempesta Technologies

# RFC 7540 5.3 vs RFC 9218

- 7540 is outdated and replaced with 9218
  - too sophisticated streams management in 7540
  - nowadays 7540 is only supported for HTTP/2
- 9218 is used only for HTTP/3
  - **(ordering)** `urgency [0,7]` – responses precedence (like weight)
  - **(bandwidth sharing)** `incremental [0, 1]` (like not exclusive)
  - urgency reprioritization, but not dependency tree
  - `10.1` prohibits sending low-urgency early received responses
- Many existing implementations with GPLv2, MIT or BSD licenses...

# RFC 7540: Nginx

- Not a full RFC support:
  - sends frames on dependent streams (ranks comparison)
  - unfair bandwidth sharing between streams on the same level

```
void ngx_http_v2_queue_frame(ngx_http_v2_connection_t *h2c, ngx_http_v2_out_frame_t *frame) {
  ngx_http_v2_out_frame_t  **out;
  for (out = &h2c->last_out; *out; out = &(*out)->next) {
    if ((*out)->blocked || (*out)->stream == NULL)
      break;

    if ((*out)->stream->node->rank < frame->stream->node->rank
        || ((*out)->stream->node->rank == frame->stream->node->rank
            && (*out)->stream->node->rel_weight >= frame->stream->node->rel_weight))
      break;
  }
  // . . .
```

# RFC 7540: H2O

- Fast `O(1)` scheduler
  - DRR-like **deficit** computation
  - 64 deficit groups (anchors)
  - frames granularity
- high memory consumption (1056B for each stream)
- Sharing bandwidth between streams on the same level is not fair *(in theory)*

```
struct st_h2o_http2_scheduler_queue_t {
    uint64_t bits;
    size_t offset;
    h2o_linklist_t anchors[64];
    h2o_linklist_t anchor257;
};
```

```
2 streams sent the same amounts
34 streams sent 1 less frames (3% inaccuracy)
116 streams sent 2 less frames (4% inaccuracy)
52 streams sent 3 less frames (7% inaccuracy)
10 streams sent 4 less frames (17% inaccuracy)
3 streams sent 5 less frames (100% inaccuracy)
4 streams sent 6 less frames (69% inaccuracy)
3 streams sent 7 less frames (100% inaccuracy)
6 streams sent 8 less frames (83% inaccuracy)
5 streams sent 9 less frames (94% inaccuracy)
5 streams sent 10 less frames (76% inaccuracy)
11 streams sent 11 less frames (82% inaccuracy)
5 streams sent 12 less frames (81% inaccuracy)
```

# RFC 7540: nghttp2 (Envoy, Apache HTTPD etc)

- Fair WFQ and good RFC support
- `log(N)` scheduler on binary heap
  - non-intrusive
  - many memory allocations
  - copies

```
static void
bubble_up(nghttp2_pq *pq, size_t index) {
  while (index != 0) {
    parent = (index - 1) / 2;
    if (!pq->less(pq->q[index], pq->q[parent]))
      return;
    swap(pq, parent, index);
    index = parent;
  }
}

int
nghttp2_pq_push(nghttp2_pq *pq,
                nghttp2_pq_entry *item) {
  if (pq->capacity <= pq->length) {
    n = nghttp2_max_size(4, pq->capacity * 2);
    nq = nghttp2_mem_realloc(pq->mem, pq->q,
                             n * sizeof(...));
    ...
  }
  ...
  bubble_up(pq, pq->length - 1);
  return 0;
}
```

# Data structures for WFQ

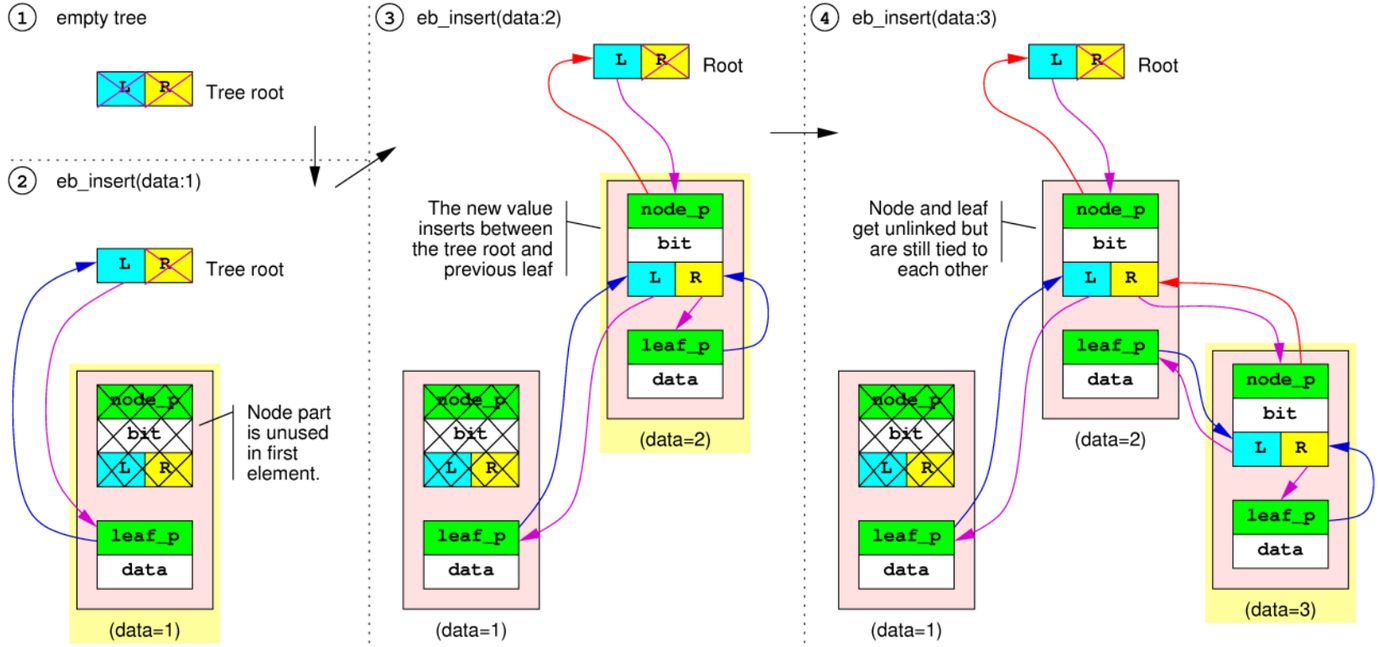
- Dependency tree
  - number of streams is *usually <100*
  - the most frequent operation: **reinsert** a minimum key

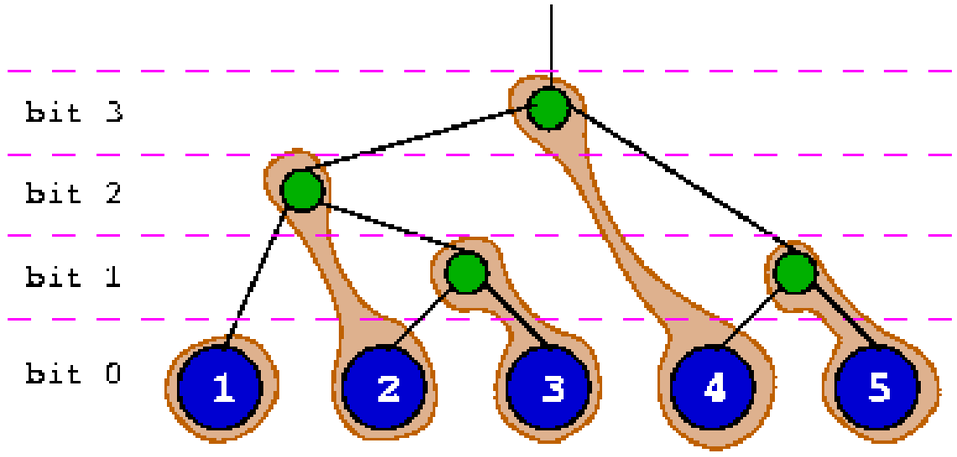
```
struct tfw_stream_sched_entry_t {
        . . .
        struct tfw_stream_sched_entry_t *parent;
        struct eb_root                  active;
        struct eb_root                  blocked;
}
```

- Weighted queue
  - ellastic binary tree is for the weighted queue
  - binary and Fibonacci heaps
  - insertion-sorted array (H2O)

# Ellastic binary tree

- *http://wtarreau.blogspot.com/2011/12/elastic-binary-trees-ebtree.html*
- **Unbalanced binary radix** tree - good on small data
- All nodes keep data (good cache locality)
- Used in HAProxy also for scheduling



# Microbenchmark for streams data structures

- Fibonacci heap – insert **O(1)**, delete O(log(n)), but bad performance
- Insertion-sorted array – the fastest, but memory greedy
- Binary heap – worse than ebtree even w/o account memory reallocations
- Linked lists – bad theoretical complexity.
- *https://github.com/tempesta-tech/blog/tree/master/h2_stream_wfq*

```
Benchmark  100 elem              Time             CPU   Iterations
BM_ebtree_insert_delete       22.6 ns         22.6 ns     31330583
BM_fheap_insert_delete         147 ns          147 ns      4791582
BM_heap_insert_delete         30.0 ns         30.0 ns     25555202
BM_h2o_insert_delete          11.7 ns         11.7 ns     59135484
```

# Naive weighted fair queuing

- From *https://www.mew.org/~kazu/material/2015-http2-priority2.pdf* :
- 3 streams:
  - A for weight 10
  - B for weight 5
  - C for weight 1
- Queuing:
  - A(10), A(9), A(8), A(7), A(6), A(5), B(5), A(4), B(4), ...

# WFQ using Deficit Round Robin

- *https://en.wikipedia.org/wiki/Deficit_round_robin*
  *https://www.mew.org/~kazu/material/2015-http2-priority2.pdf*
- Used by H2O (inverted and aproximated, frame granularity)
- nghttp2
  - `penalty` (like `log` in EEVDF) `= last_send * 256 + pending`
  - `cycle` (min is next) `+= penalty / weight`
  - `pending = penalty % weight`

```
A for weight 10, cycle = pending = penalty = 0
B for weight 5, cycle = pending = penalty = 0
A sends 100:  penalty = 25600, cycle = 2560, pending = 0
B sends 100: penalty = 25600, cycle = 5120, pending = 0      : A=400 B=200
A sends 150: penalty = 38400, cycle = 6400, pending = 0
B sends 100: penalty = 25600, cycle = 10240, pending = 0
A sends 100: penalty = 25600, cycle = 8960, pending = 0
A sends 50: penalty = 12800, cycle = 10240, pending = 0
```

# ...modern RFC 9218 support for HTTP/2

- `SETTINGS_NO_RFC7540_PRIORITIES`
- Firefox since 128 (July 9, ‘24)
  *https://developer.mozilla.org/en-US/docs/Mozilla/Firefox/Releases/128*
- Chrome since 124 (April 16, ‘24)
  *https://developer.chrome.com/release-notes/124*
- nghttp2 supports `SETTINGS_NO_RFC7540_PRIORITIES` and RFC9218 for HTTP2
- H2O doesn’t support `SETTINGS_NO_RFC7540_PRIORITIES`, RFC9218 is only for HTTP3
- Nginx doesn’t support `SETTINGS_NO_RFC7540_PRIORITIES`, seems doesn’t support RFC9218 at all

Tempesta Technologies

# Firefox 128: RFC 9218 – no dependency tree

- No changing priorities
- Sends SETTINGS_NO_RFC7540_PRIORITIES: ignore weights and dependency
  - Firefox 126 & 127 don’t send the setting, but work same as Firefox 128 if server sends it
- Sends `priority` header in request (9218 prioritization hint)

```
Create new stream id 3, weight 43, excl 0, depends on 0, u=0, i
Create new stream id 5, weight 22, excl 0, depends on 0, u=0, i
Create new stream id 7, weight 22, excl 0, depends on 0, u=0, i
Create new stream id 9, weight 22, excl 0, depends on 0, u=0, i
```

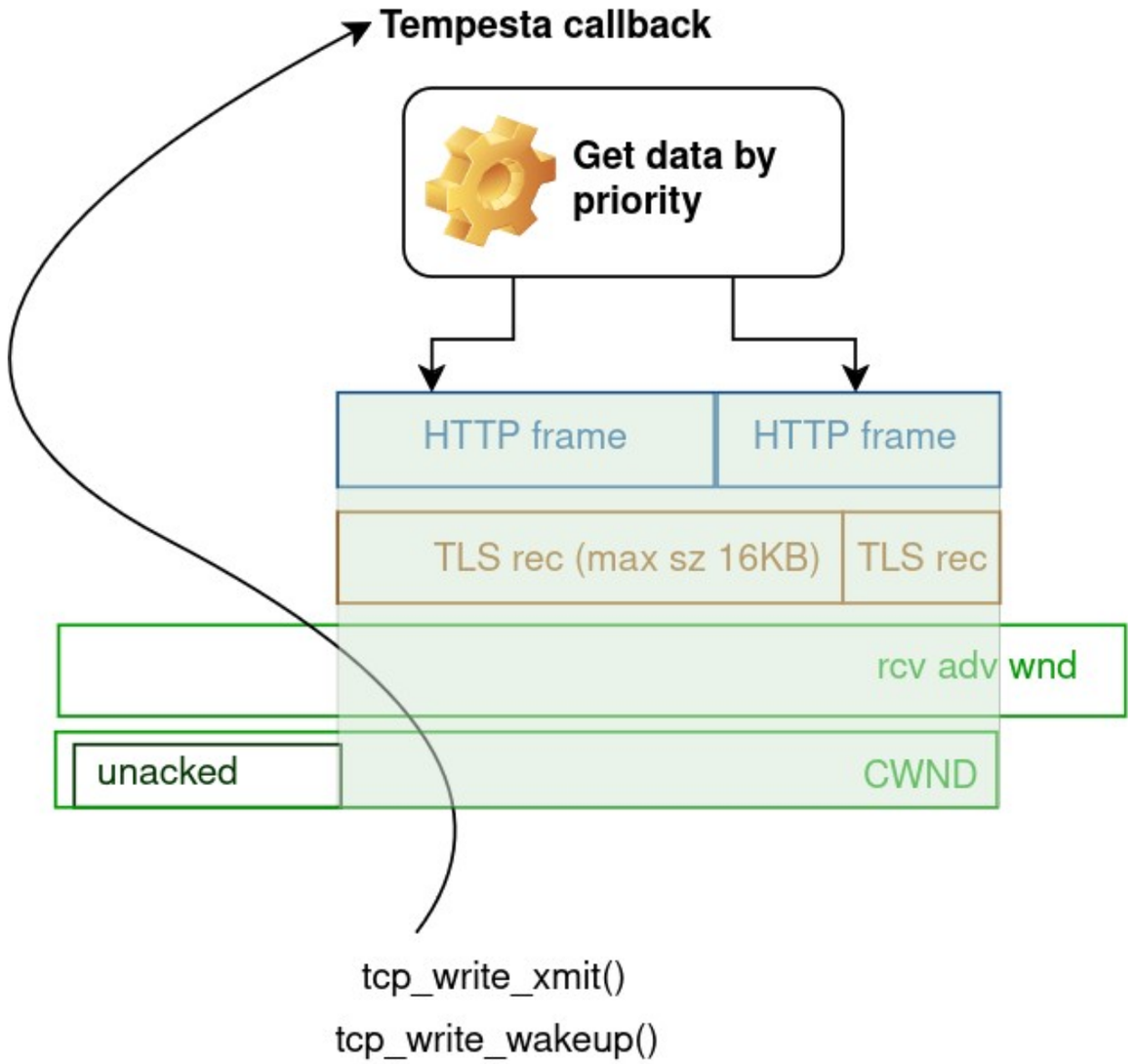
# Tempesta FW HTTP streams scheduling *(in-progress)*

- `O(log(N))` DRR scheduler
- Good memory locality
  - intrusive ebtree nodes in streams
  - streams are preallocated on the same pages
- ebtree on each dependency layer
  - if single node, then no deficit recomputations
- Use weights to **order** responses (like RFC 9218)
  - Tempesta FW uses `weight` as `urgency` (by configuration option)
  - Cloudflare splits resources in groups
    *https://blog.cloudflare.com/better-http-2-prioritization-for-a-faster-web/*

Tempesta Technologies

# Scheduling HTTP frames to TLS & TCP

- We have access to TCP
- Do not send too small TLS records for large HTTP frames



# Security

- Dependency Cycle Attack (CVE-2015-8659): attacker builds the dep tree with cycles which leads to infinite computation of the tree
  - seems only for non-RFC-compliant implementations
- CVE-2019-9511: many `PRIORITY` frames leading to the dep tree reconstruction
  - `max_concurrent_streams` configuration option limits the size of the dep tree (including idle streams)
  - rate-limiting for `PING`, `PRIORITY` and `SETTINGS` frames
- In general, a flood is possible for any control frame
  e.g. `PING` flood, `PRIOIRTY` flood etc.

# Thanks!

- *https://tempesta-tech.com/knowledge-base/HTTP2-streams-prioritization/*
- *https://github.com/tempesta-tech/tempesta*

*em@tempesta-tech.com* *ak@tempesta-tech.com* *kt@tempesta-tech.com*

Tempesta Technologies