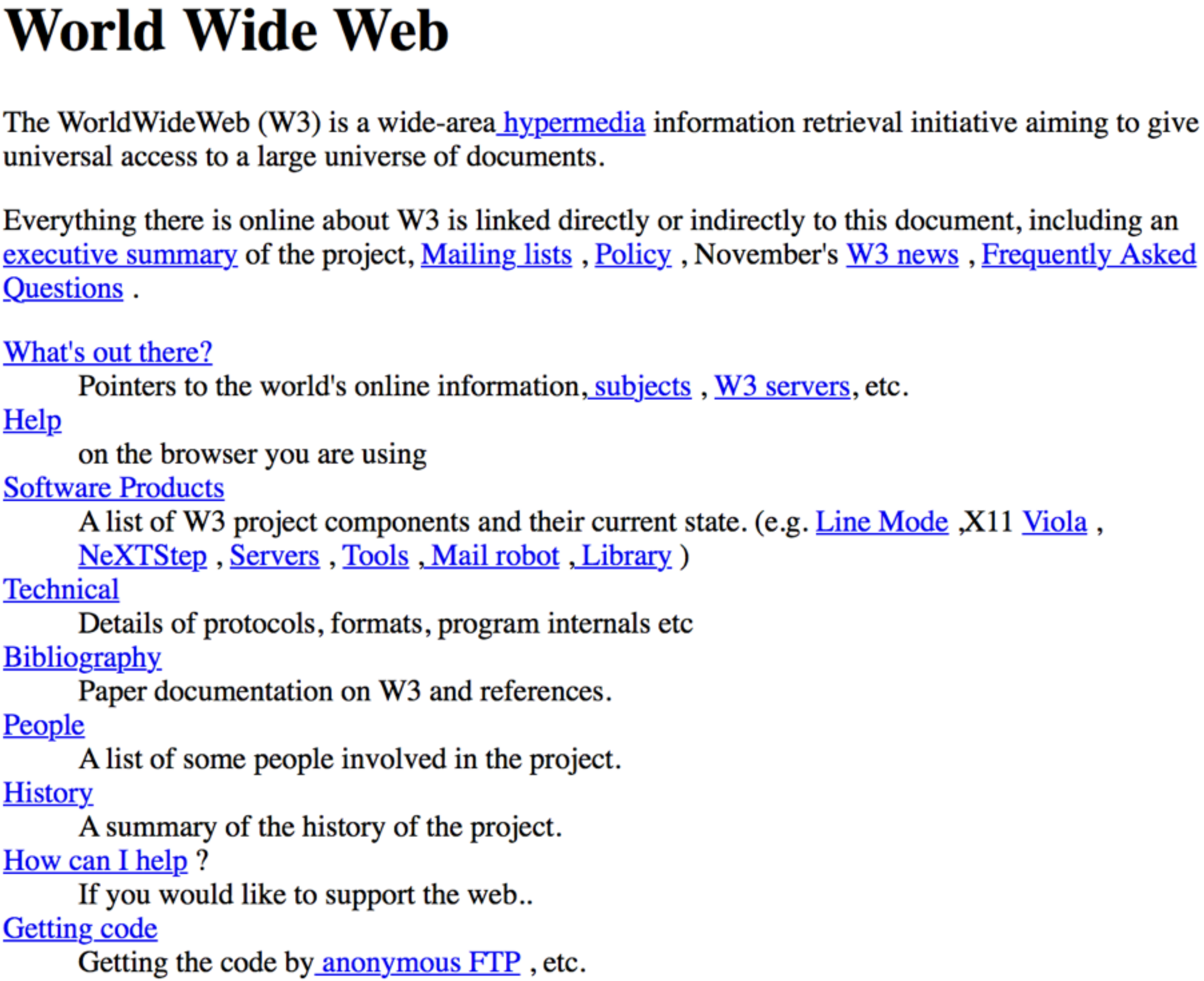
HTTP/2 performance and Non-blocking Architecture

@julienviet



World Wide Web

The WorldWideWeb (W3) is a wide-area hypermedia information retrieval initiative aiming to give universal access to a large universe of documents.

Everything there is online about W3 is linked directly or indirectly to this document, including an executive summary of the project, Mailing lists , Policy , November's W3 news , Frequently Asked Questions .

What's out there?
: Pointers to the world's online information, subjects , W3 servers, etc.

Help
: on the browser you are using

Software Products
: A list of W3 project components and their current state. (e.g. Line Mode ,X11 Viola , NeXTStep , Servers , Tools , Mail robot , Library)

Technical
: Details of protocols, formats, program internals etc

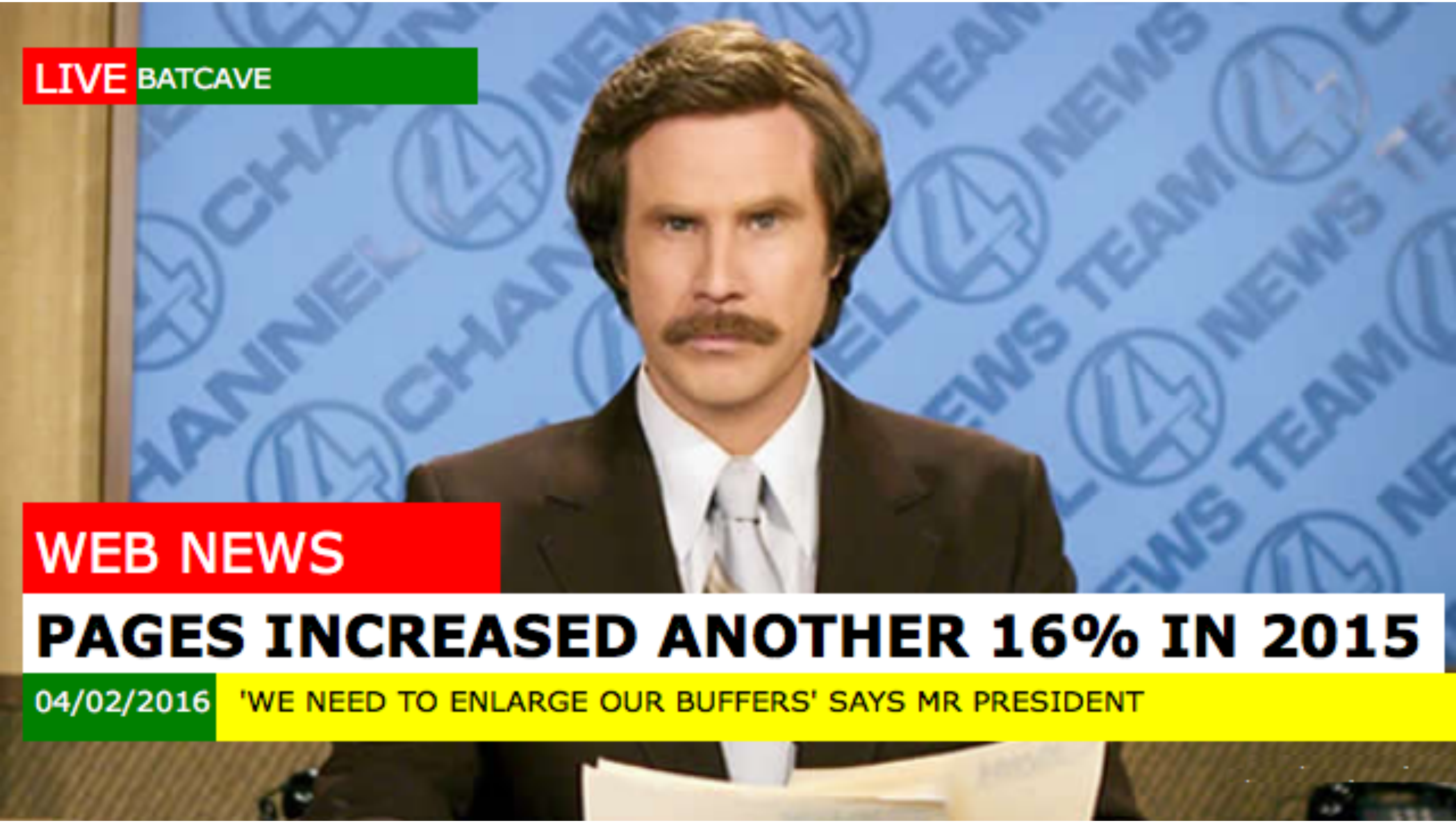
Bibliography
: Paper documentation on W3 and references.

People
: A list of some people involved in the project.

History
: A summary of the history of the project.

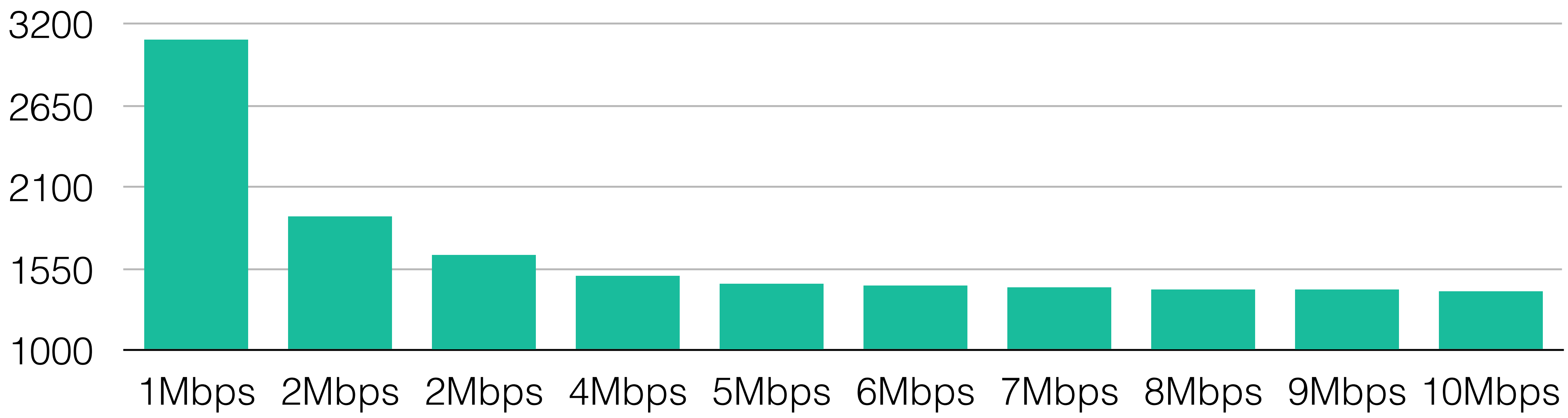
How can I help ?
: If you would like to support the web..

Getting code
: Getting the code by anonymous FTP , etc.

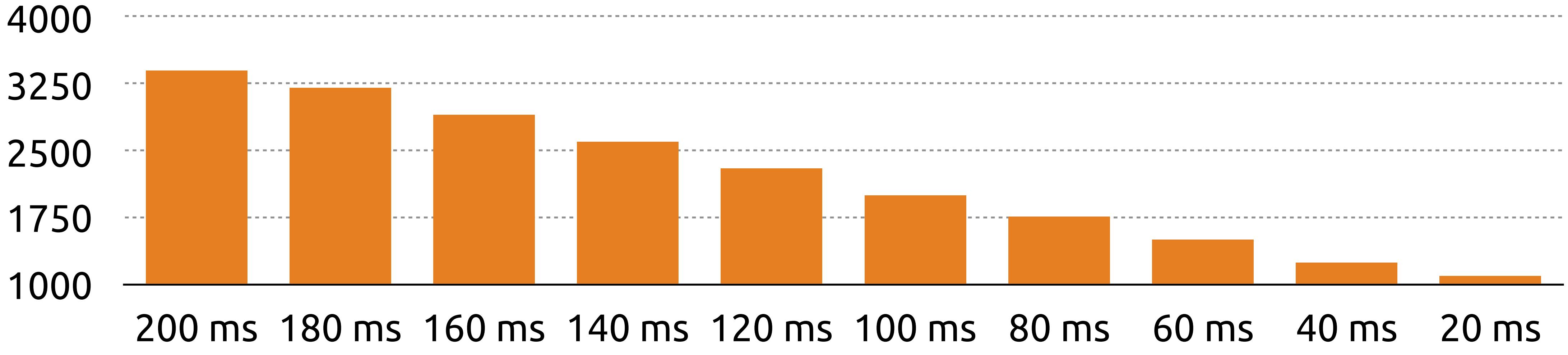

LIVE
BATCAVE
WEB NEWS
PAGES INCREASED ANOTHER 16% IN 2015
04/02/2016
'WE NEED TO ENLARGE OUR BUFFERS' SAYS MR PRESIDENT

Latency vs Bandwidth impact on Page Load Time

Page Load Time as bandwidth increases



Page Load Time as latency decrease



HTTP/1 in the browser

HTTP / TCP impedance mismatch

HTTP/2 in the browser

HTTP/2 intent

Not a new version of the protocol

it’s about how it gets onto the wire

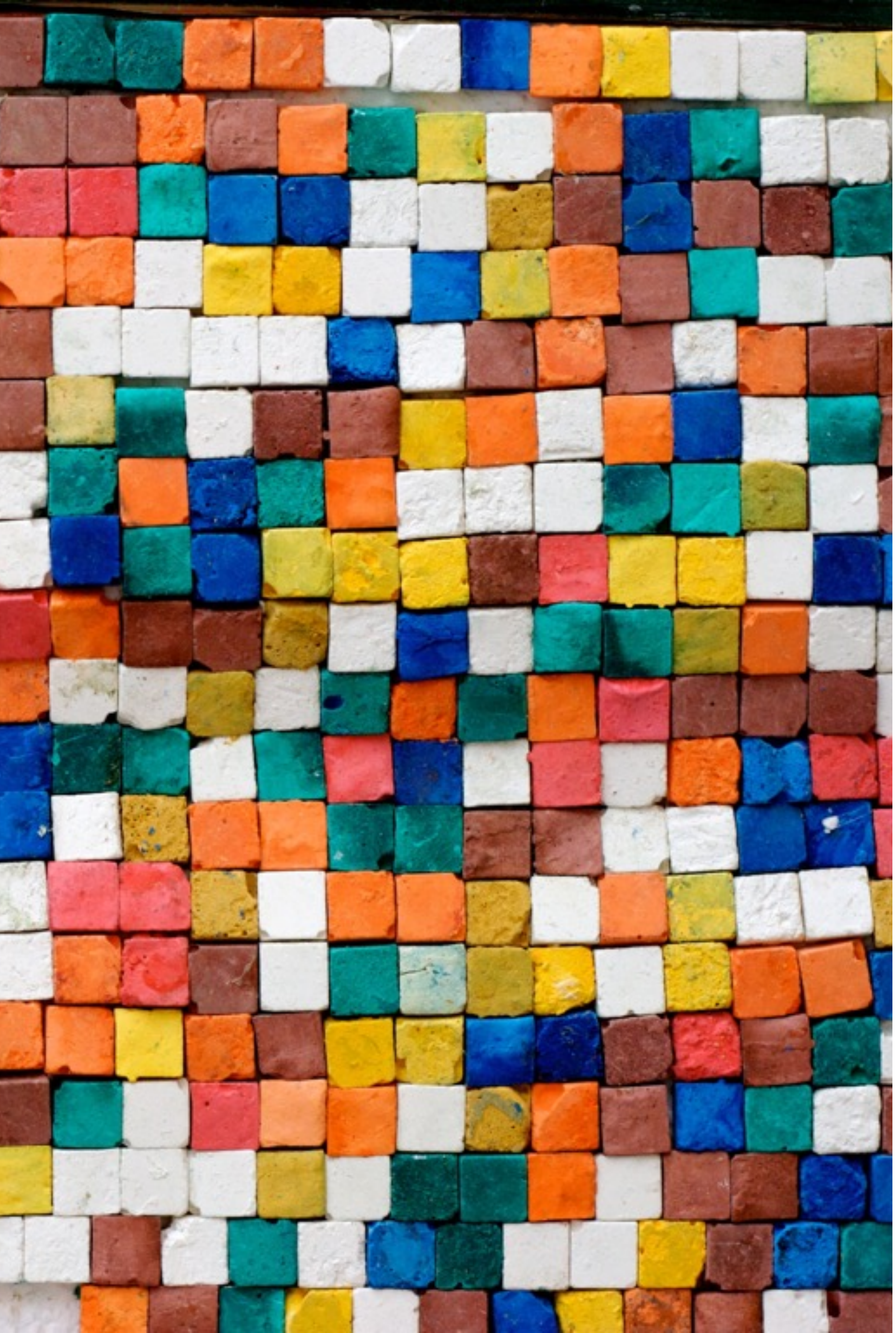
HTTP/2 brings network sympathy

Why HTTP/2 performs better

B1n4ry

COMPR ESS

headers headers headers
headers headers headers
headers headers headers
headers headers headers
headers headers headers
headers headers headers



S-l-i-c-e

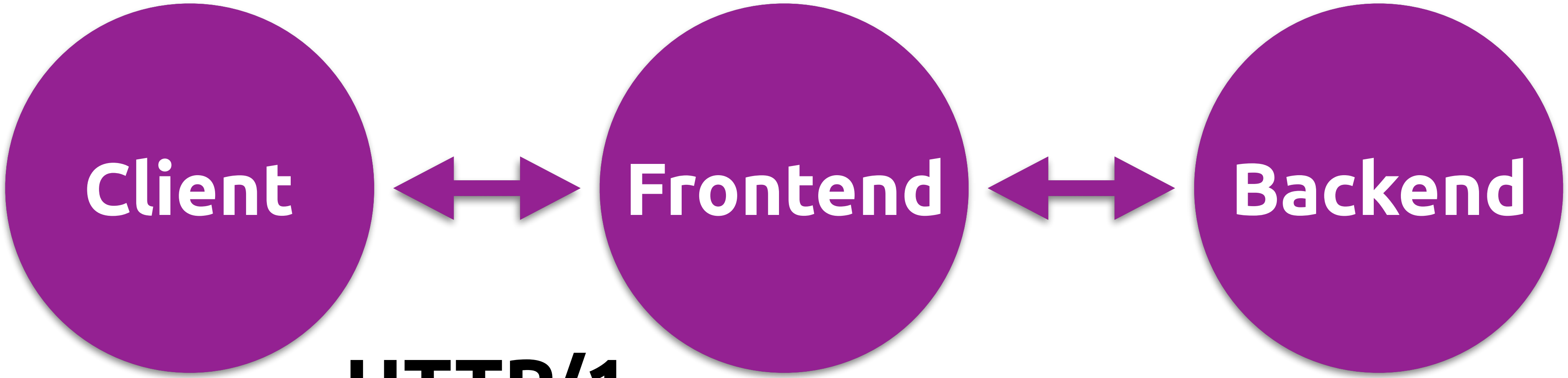
SPEED LIMIT 40
SPEED LIMIT 30
SPEED LIMIT 30
CAUTION
REDUCED SPEED ZONE
HEAVY MERGING TRAFFIC AHEAD
1629

Priorities

PUSH

HTTP/2 on the server

HTTP/1 vs HTTP/2 benchmark



Client

Frontend

Backend

20ms
think time

HTTP/1
vs
HTTP/2

connection
pool

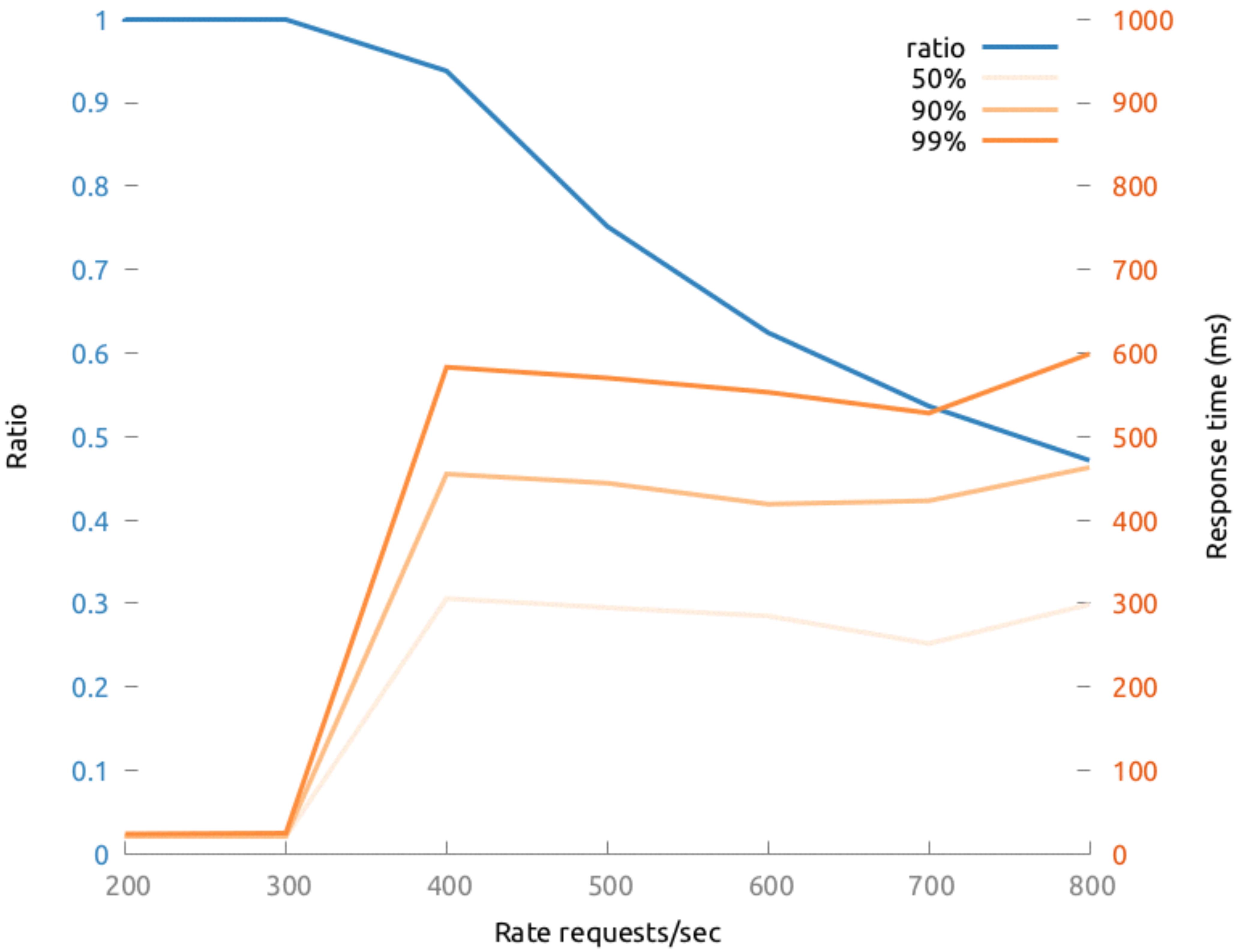
Benchmark

Pace requests at a given rate

Log ratio of requests performed/planned

Log response time percentiles

https://github.com/vietj/http2-bench

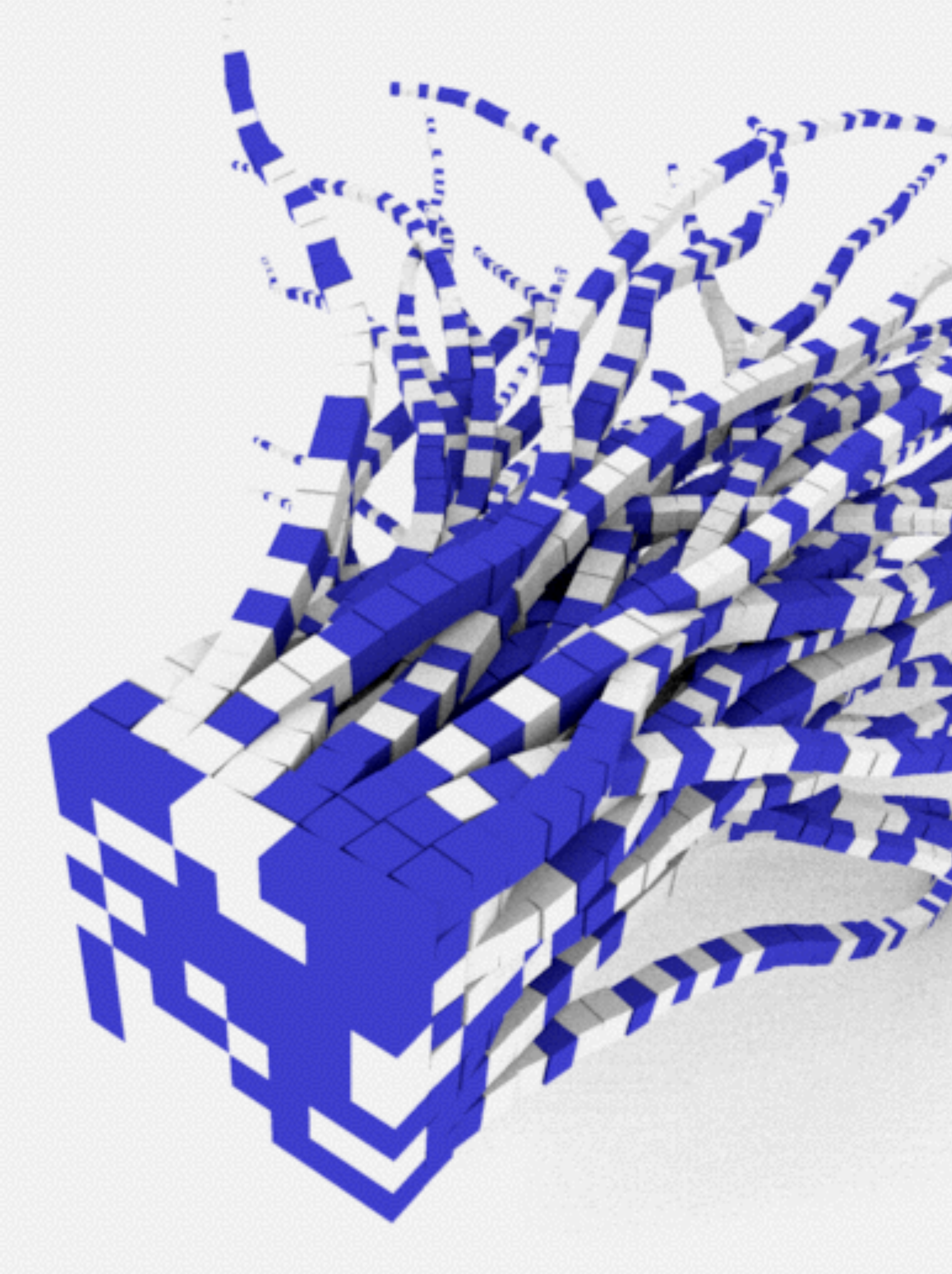


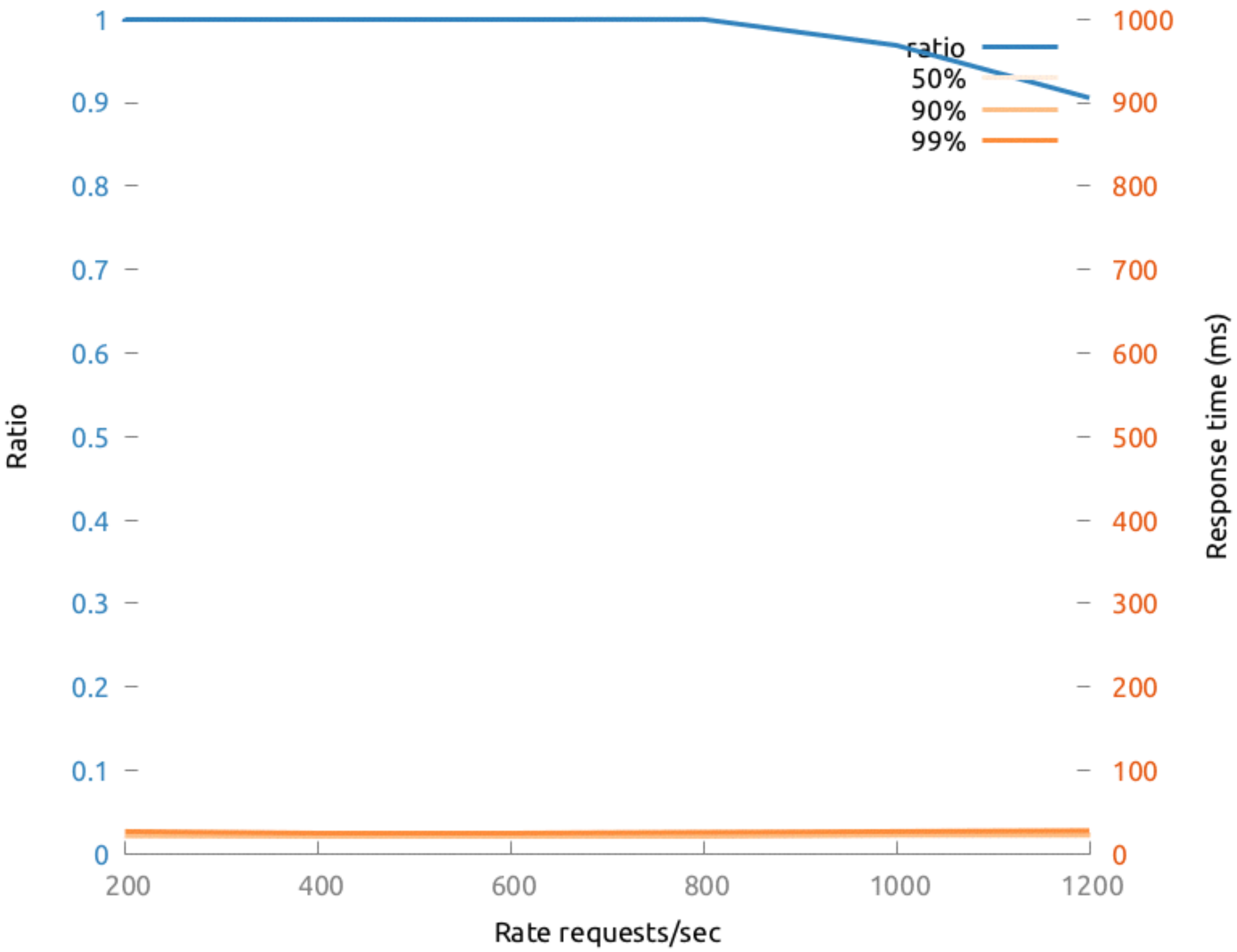
HTTP/1 - 8 connections - pipelined

What is limiting us ?

FIFO
one thing
at a time!

HTTP/2 multiplexing

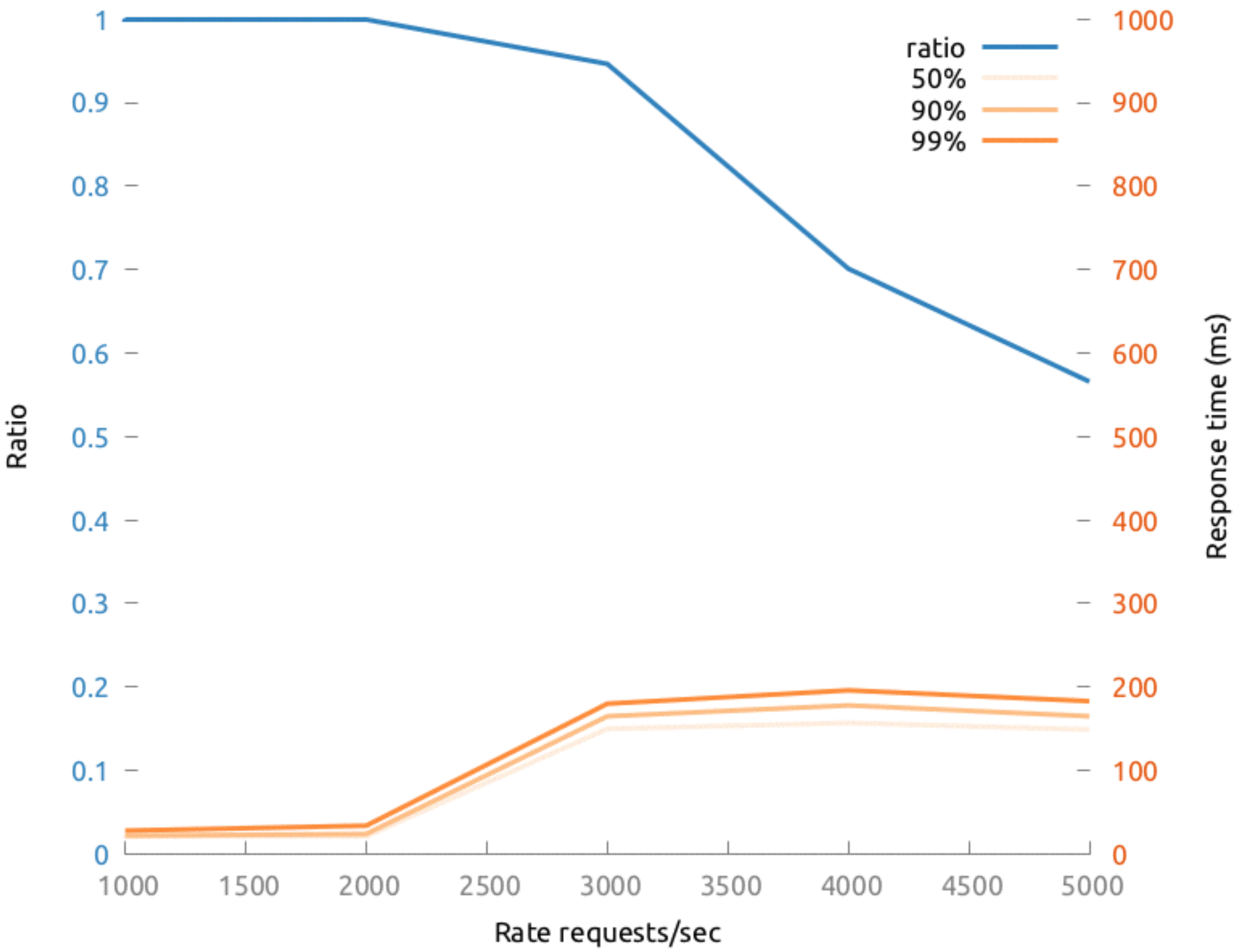




HTTP/2 - 1 connections - concurrency 20

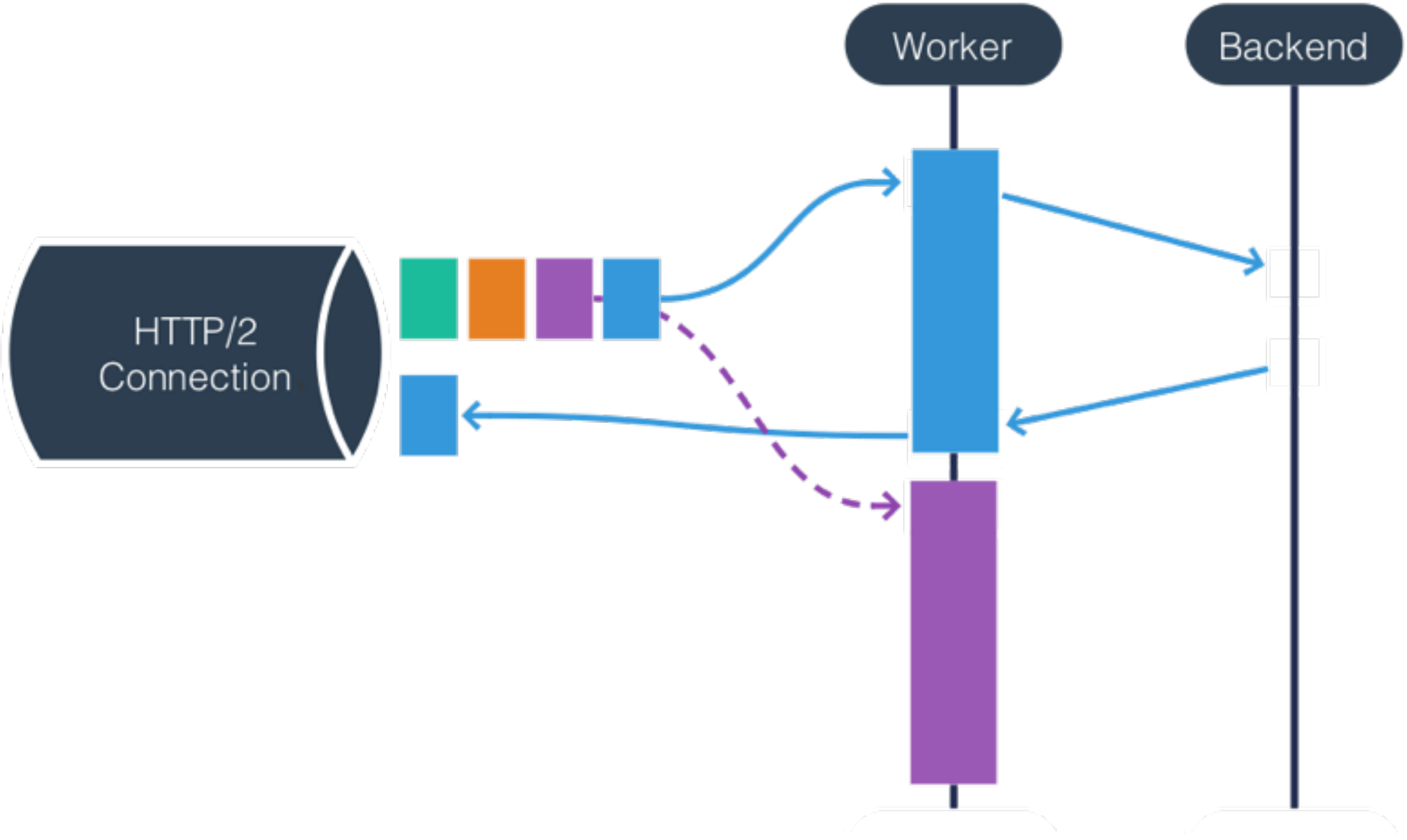
Concurrency increased !

But…



HTTP/2 - 1 connections - concurrency 400

Congestion


Worker
Backend
HTTP/2
Connection

Multithreading is an illusion of parallelism

Input / Output

Streams of ~~byte~~ byte[]

However the reality is different

In reality we have CPU which have cores and we manipulate network packets

We want to make a better usage of our resources

How to write programs that are efficient ?

The real problem is blocking

How to *not* block ?

Concurrency with Vert.x

http://vertx.io

Library for building reactive applications

Eclipse project (ASL2/EPL)

Java 8

Vert.x Core

- core library for building a stack
- embeddable (core 1MB)

Vert.x stack

- coherent set of libraries built on top of Vert.x Core
- data access, messaging, metrics, etc...

What’s Vert.x ?

Inspired from Erlang/OTP and Node

Event driven

Polyglot

Event bus

High performances / easy to scale

Lightweight / embeddable

Clustering / HA

Why Vert.x

Simple concurrency model

Unified async API for IO, filesystem, data access, messaging, ...

Easy to scale

Easy to deploy

Coherent stack

Provides also an RxJava API

Non blocking server

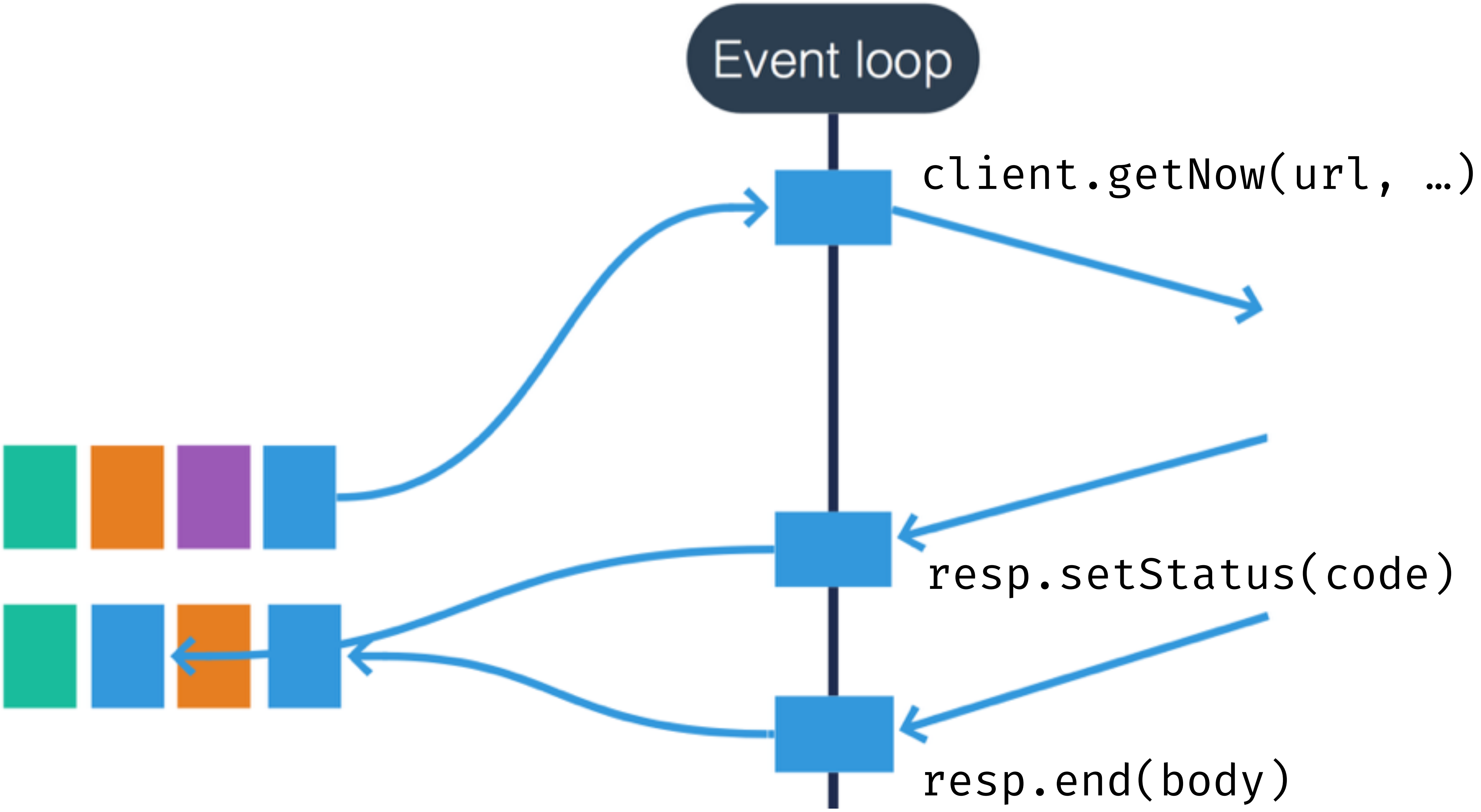
```
public static void main(String□ args) {

  Vertx vertx = Vertx.vertx();
  HttpServer server = vertx.createHttpServer();

  server.requestHandler(req → {
    req.response()
      .putHeader(“Content-Type”, “text/plain”)
      .end(“Hello World”);
  });

  server.listen(8080);
}
```

Non blocking client

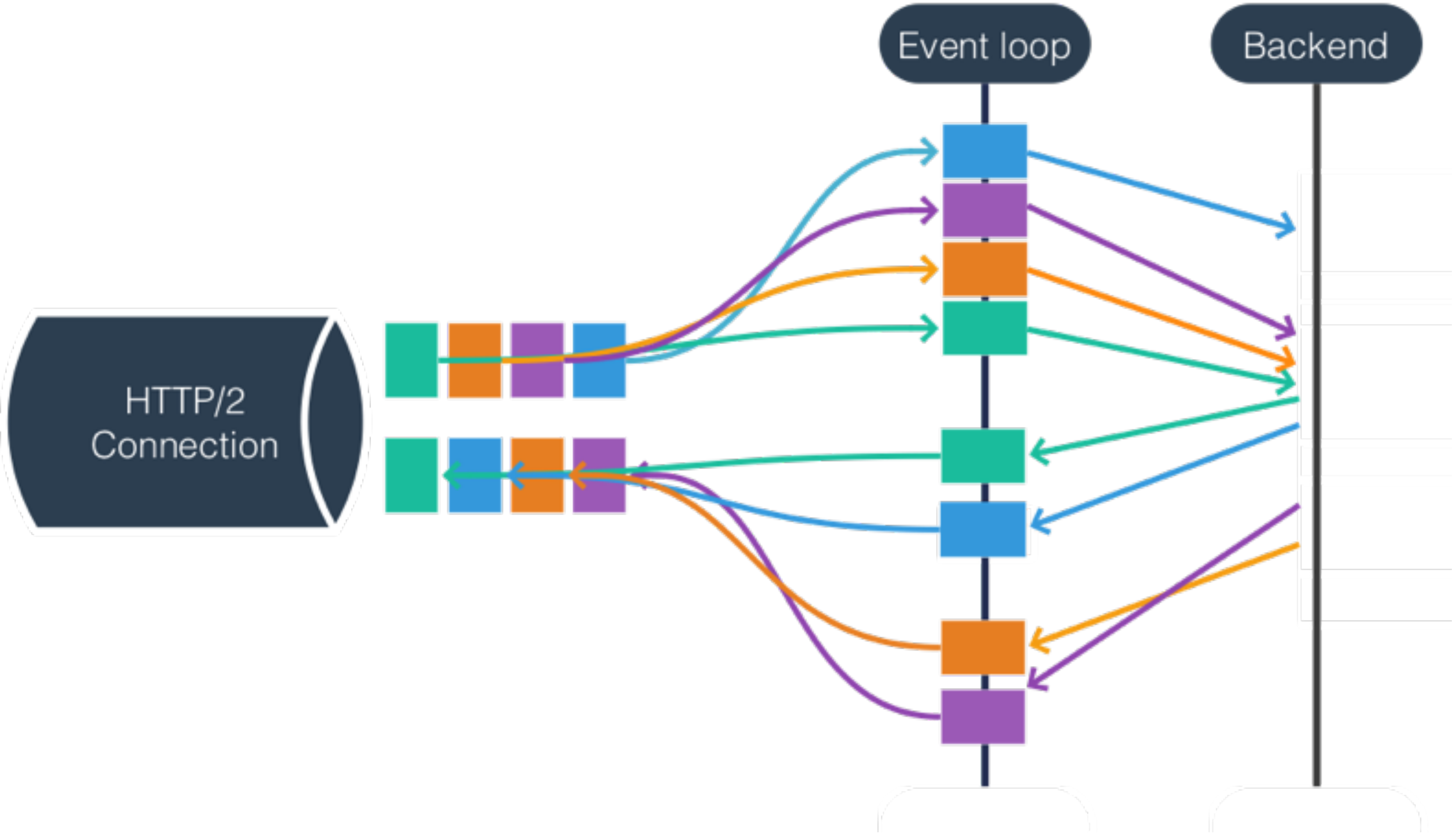
```
public static void main(String□ args) {

  HttpClient client = vertx.createHttpClient();

  client.getNow(“http://backend”, resp → {
    int status = resp.status();

    resp.bodyHandler(body → {
      System.out.println(body.length());

    });
  });
}
```

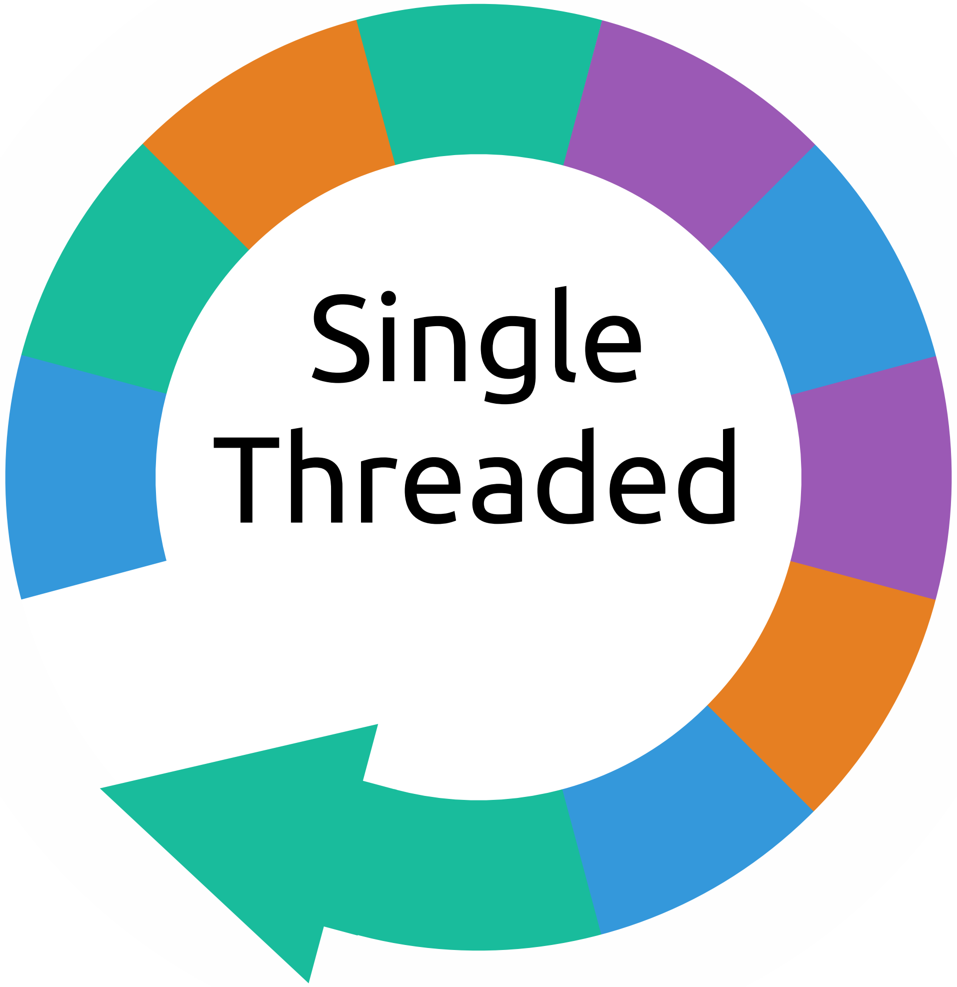
NB server+client

```
server.requestHandler(req -> {
  HttpServerResponse resp = req.response();

 client.getNow(“http://backend”, clientResp -> {
    int code = clientResp.status()
    resp.setStatus(code);

    clientResp.bodyHandler(body -> {
      resp.end(body);
    });
 });
});
```



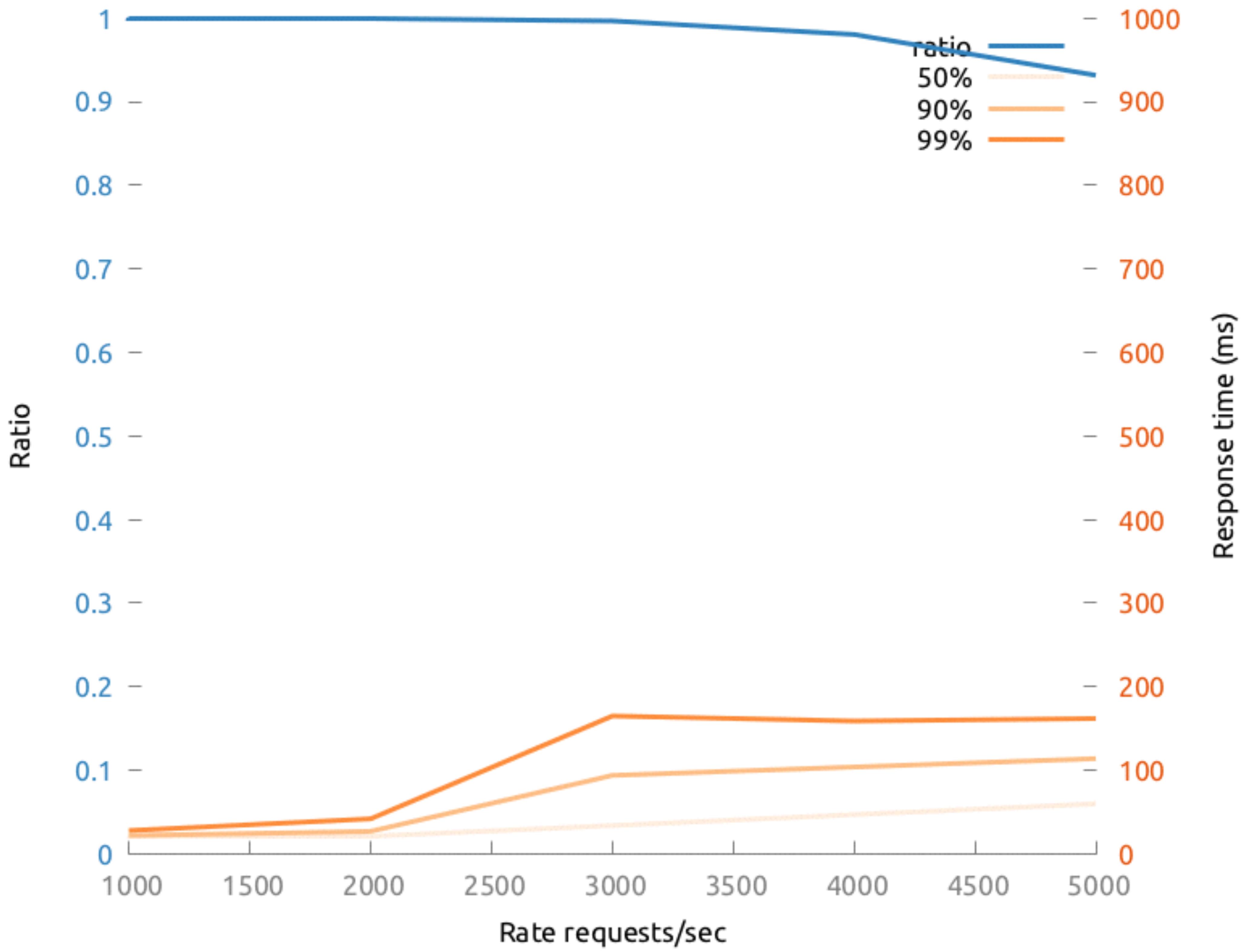
```
client.getNow(url, …)
resp.setStatus(code)
resp.end(body)
```


Event loop
Backend
HTTP/2
Connection



Reactor pattern: Event Loop

C10K
it’s all about concurrency

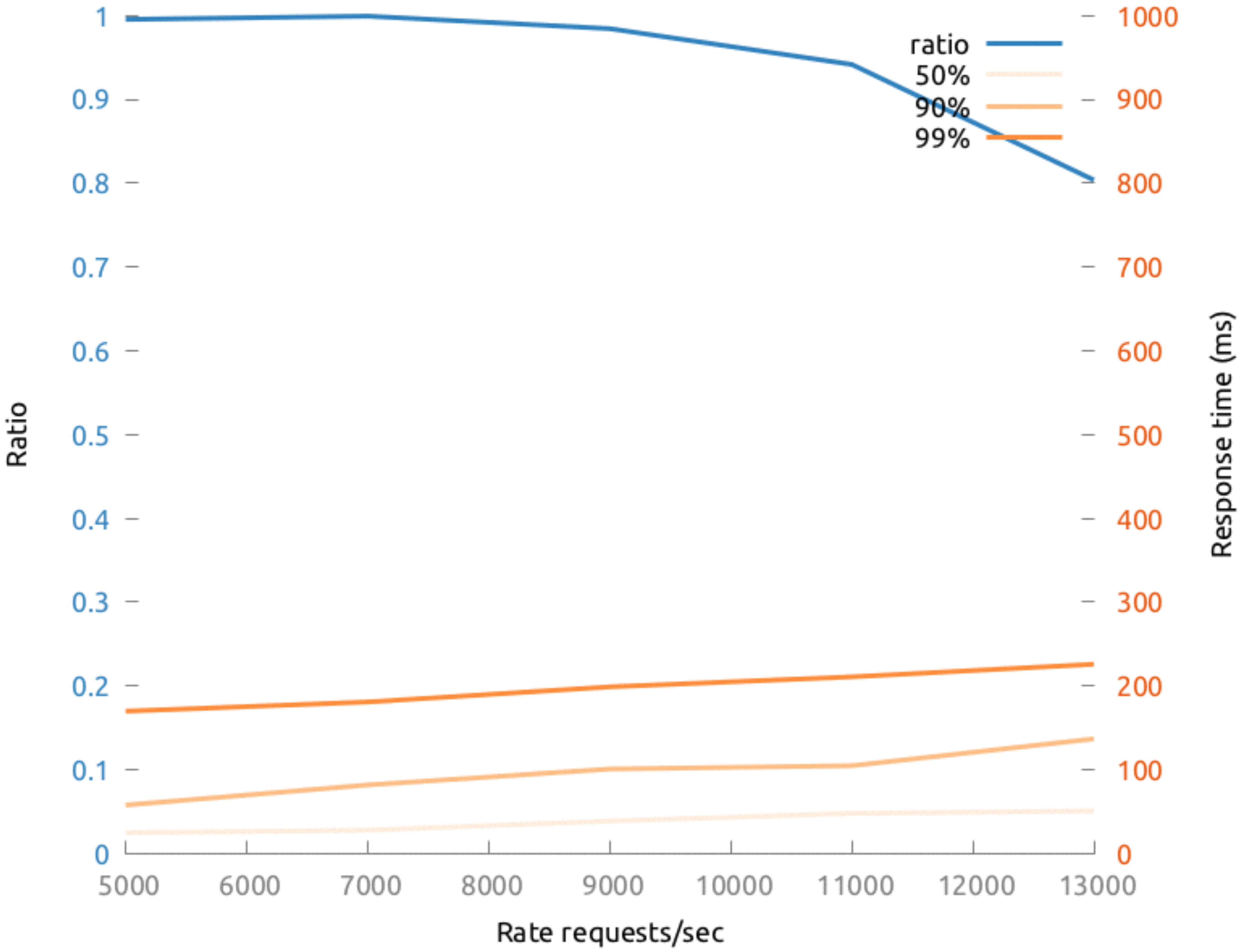


HTTP/2 non blocking - 1 connection - concurrency 400

using…
a single core!

Multi reactor pattern

Load balancing



HTTP/2 non blocking - 4 connections - concurrency 200

Discussion

From network to application

Reactive back pressure

Going distributed