Trusted Browsers for Uncertain Times

David Kohlbrenner and Hovav Shacham

UC San Diego

Building a browser that can provably mitigate timing attacks

Trusted Browsers for Uncertain Times

- Time and web browsers
- Mitigating attacks
- A trusted browser
- A (less) trusted browser

Timing attacks

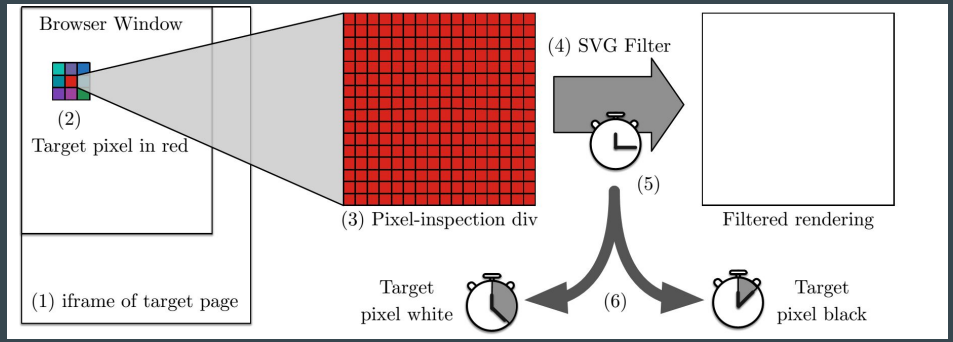
- Time and web browsers
- Mitigating attacks
- A trusted browser
- A (less) trusted browser

Browsers and timing attacks

- Browser has multiple privilege levels
  - User secrets
  - System secrets
  - Origin secrets
- Browsers expose detailed information
  - performance.now()
  - getAnimationFrame()
- Browsers compute and communicate between levels

Timing attacks in web browsers

- SVG Filter cross-origin pixel stealing
- JavaScript cache timing attacks
- Fingerprinting
- History Sniffing



The Spy in the Sandbox – Practical Cache Attacks in Javascript

What is being done about it? - SVG attack

Hm. After reducing the if() in the inner loop to

```
  if (extrema[i] > pixel) {
    extrema[i] = pixel;
  }
```

or

```
  if (extrema[i] < pixel) {
    extrema[i] = pixel;
  }
```

, the problem boils down to: how to implement constant-time min(a, b) and max(a, b) in C++?

volatile? memory barriers? or is this something that should be written in assembly? the problem must have been solved somewhere before...

What is being done about it? - Cache attack

```
// static
double PerformanceBase::clampTimeResolution(double timeSeconds)
{
    const double resolutionSeconds = 0.000005;
    return floor(timeSeconds / resolutionSeconds) * resolutionSeconds;
}

double PerformanceBase::now() const
{
    double nowSeconds = monotonicallyIncreasingTime() - m_timeOrigin;
    return 1000.0 * clampTimeResolution(nowSeconds);
}
```

What is being done about it? - Cache attack

```
diff --git a/dom/base/nsPerformance.cpp b/dom/base/nsPerformance.cpp
index 2cd0aa8..39a213d 100644
--- a/dom/base/nsPerformance.cpp
+++ b/dom/base/nsPerformance.cpp
@@ -424,7 +424,11 @@ nsPerformance::Navigation()
 DOMHighResTimeStamp
 nsPerformance::Now()
 {
-  return GetDOMTiming()->TimeStampToDOMHighRes(mozilla::TimeStamp::Now());
+  // "Implementations that cannot get the required precision (for example, if
+  //  the underlying system doesn't support it) are allowed to only be accurate
+  //  to one millisecond."
+  //    -- https://developer.mozilla.org/en-US/docs/Web/API/DOMHighResTimeStamp
+  return floor(GetDOMTiming()->TimeStampToDOMHighRes(mozilla::TimeStamp::Now())/100.0)*100.0;
 }
```

Unfortunately,
this doesn’t work.

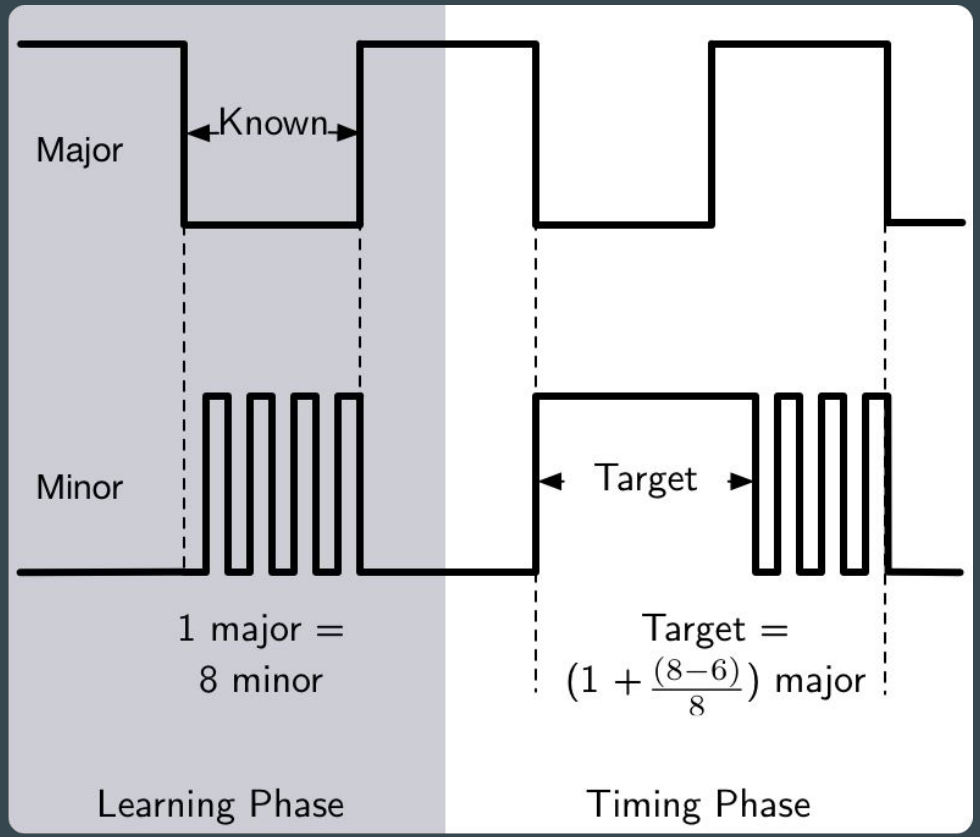
Better clocks with edges

- Time and web browsers
- Mitigating attacks
- A trusted browser
- A (less) trusted browser

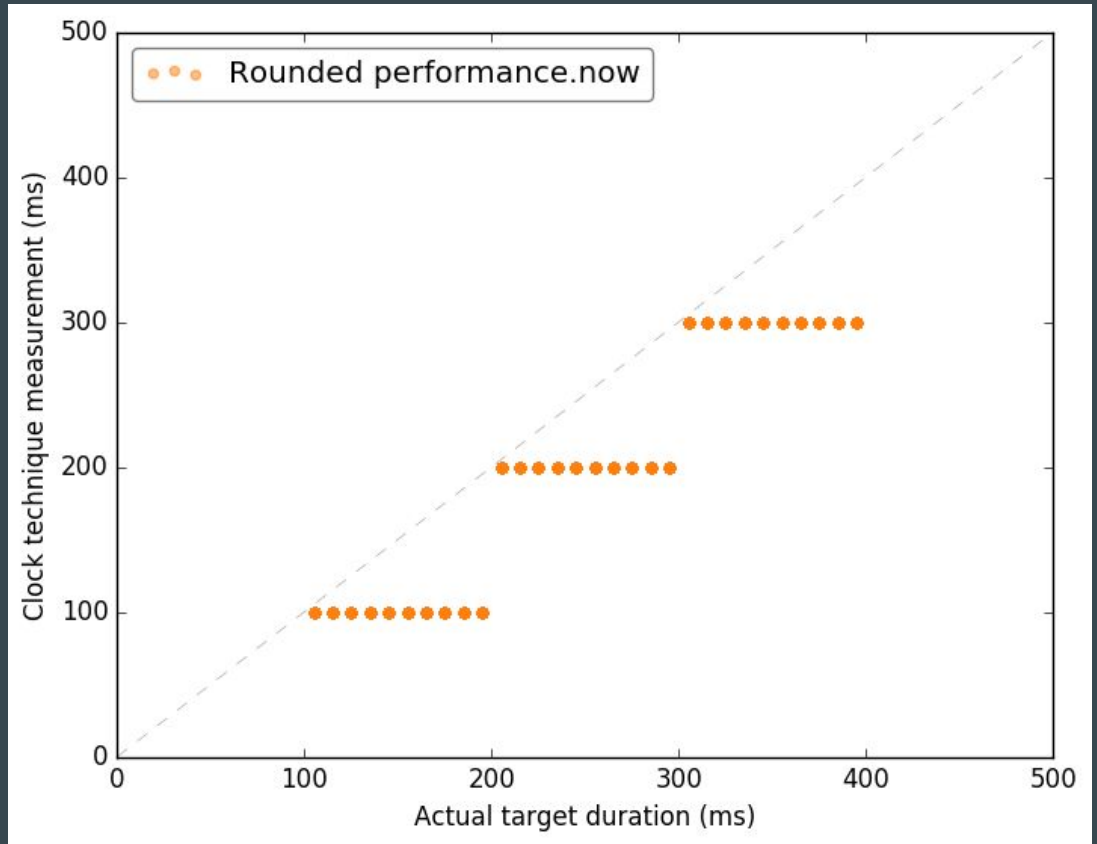
Rounding down the clock

```
diff --git a/dom/base/nsPerformance.cpp b/dom/base/nsPerformance.cpp
index 2cd0aa8..39a213d 100644
--- a/dom/base/nsPerformance.cpp
+++ b/dom/base/nsPerformance.cpp
@@ -424,7 +424,11 @@ nsPerformance::Navigation()
 DOMHighResTimeStamp
 nsPerformance::Now()
 {
-  return GetDOMTiming()->TimeStampToDOMHighRes(mozilla::TimeStamp::Now());
+  // "Implementations that cannot get the required precision (for example, if
+  //  the underlying system doesn't support it) are allowed to only be accurate
+  //  to one millisecond."
+  //    -- https://developer.mozilla.org/en-US/docs/Web/API/DOMHighResTimeStamp
+  return floor(GetDOMTiming()->TimeStampToDOMHighRes(mozilla::TimeStamp::Now())/100.0)*100.0;
 }
```

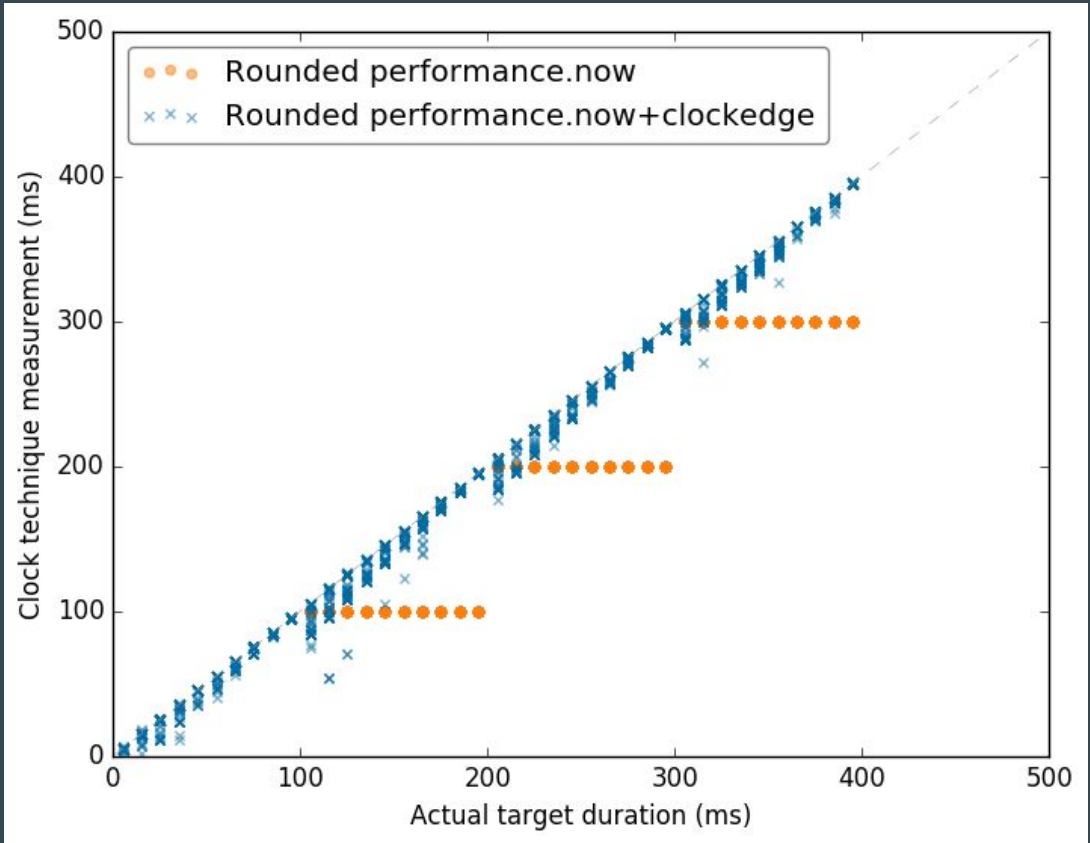
Clock-edge technique



Clock-edge technique - `performance.now()`



Clock-edge technique - `performance.now()`



Implicit clocks in the browser

- Time and web browsers
- Mitigating attacks
- A trusted browser
- A (less) trusted browser

Implicit clocks - Techniques

- `<video>` frames
- Web Speech
- `<video>` played
- `setTimeout()`
- CSS Animations
- WebVTT API
- XHRs with cooperating server

Description	Clock type		
	Firefox	**Chrome**	**Safari**
Explicit clocks	L	L	L
Video frames	L	L	L
Video `played`	X	L	L
WebSpeech API	L	+	—
`setTimeout`	X	X	X
CSS Animations	X	X	X
WebVTT API	X	X	X
Rate-limited server	X	X	X

Table 2: Implicit clock type in different browsers
L Exitless , X Exiting , — Not implemented, + Buggy

Implicit clocks - Techniques

- `<video>` frames
- Web Speech
- `<video>` played
- `setTimeout()`
- CSS Animations
- WebVTT API
- XHRs with cooperating server

Description	Clock type Firefox	Chrome	Safari
Explicit clocks	L	L	L
Video frames	L	L	L
Video `played`	X	L	L
WebSpeech API	L	+	—
`setTimeout`	X	X	X
CSS Animations	X	X	X
WebVTT API	X	X	X
Rate-limited server	X	X	X

Table 2: Implicit clock type in different browsers
L Exitless , X Exiting , — Not implemented, + Buggy

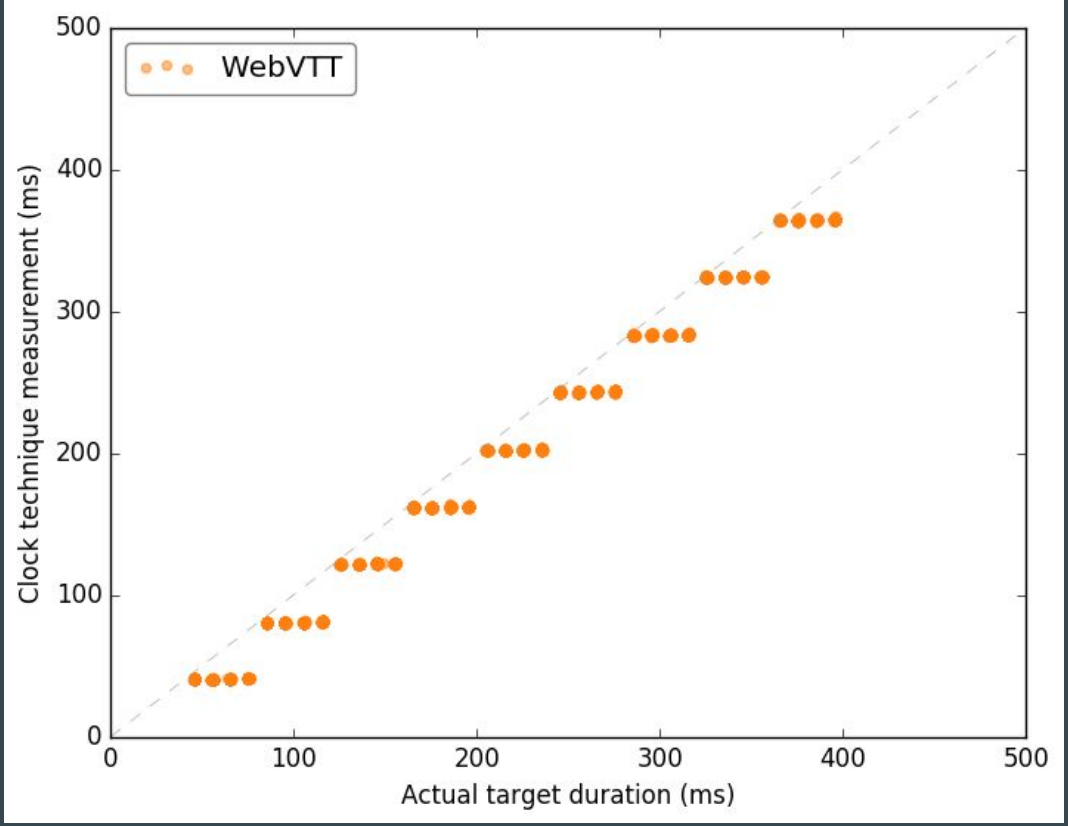
Probably many many more!

Implicit clocks - WebVTT

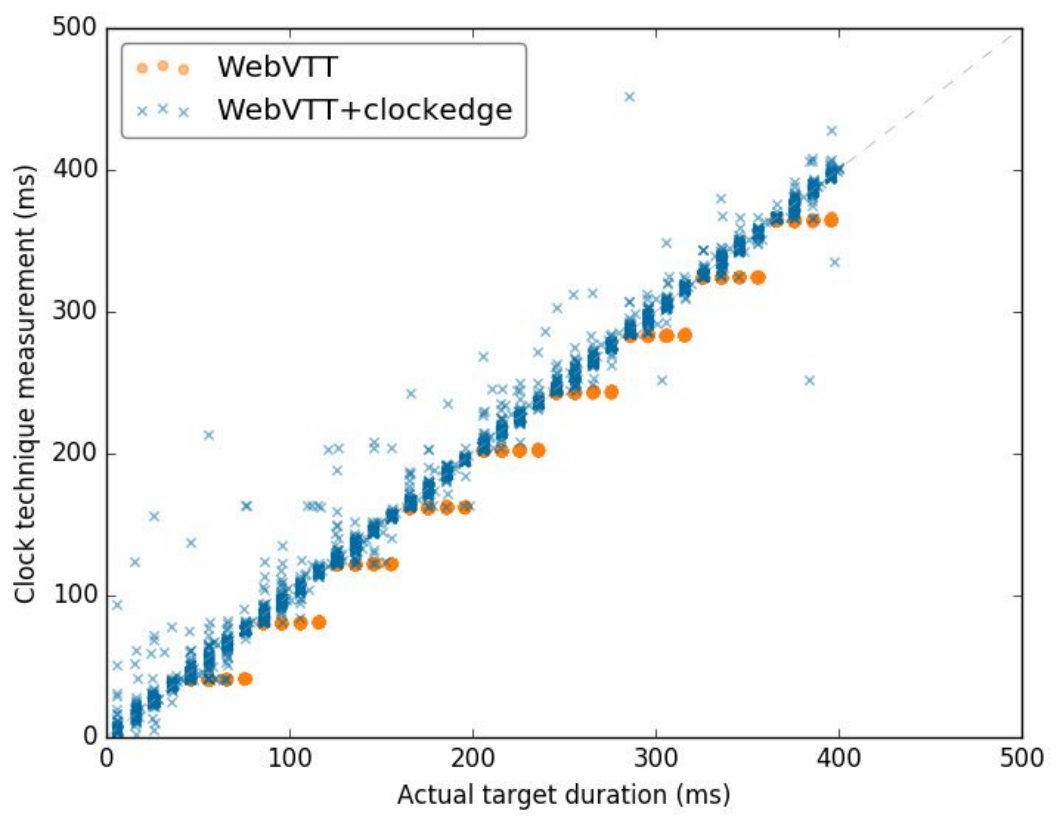
- Subtitles for `<video>` elements
- Specified in a .vtt file
    - WEBVTT
      00:00:00.000 --> 00:00:00.001
      A very short duration subtitle
- Specifies arbitrary subtitles with 1ms granularity
- `track.activeCues` returns all displayed subtitles

```
WEBVTT - cues

0
00:00:00.000 --> 00:00:00.001
0

1
00:00:00.001 --> 00:00:00.002
1

2
00:00:00.002 --> 00:00:00.003
2

3
00:00:00.003 --> 00:00:00.004
3

4
00:00:00.004 --> 00:00:00.005
4
```

Implicit clocks - WebVTT



Implicit clocks - WebVTT and clock-edge



How to mitigate timing attacks

- Time and web browsers
- Mitigating attacks
- A trusted browser
- A (less) trusted browser

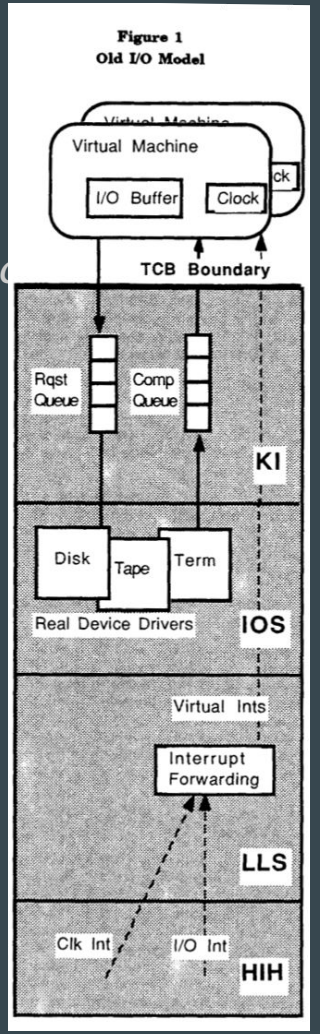
Degrade all clocks
available to the attacker.

Fuzzy time for the VAX security kernel

- "[*A*] *collection of techniques that reduces the bandwidths of c*

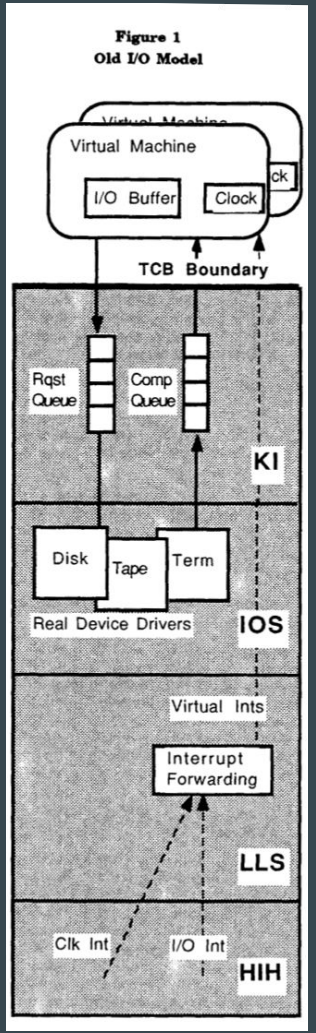
  channels by making all clocks available to a process noisy."

- "Reducing Timing Channels with Fuzzy Time"
  - Hu at Oakland 1991!



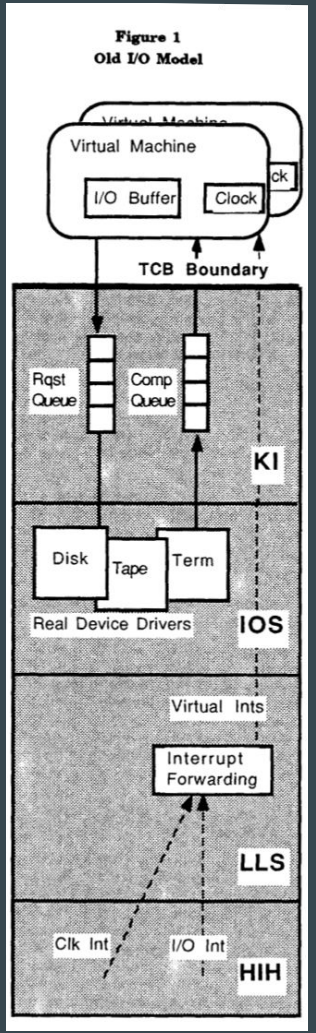
Covert channels

- Two clocks
- Modulated
  - The channel
- Reference
  - Wall clock, etc



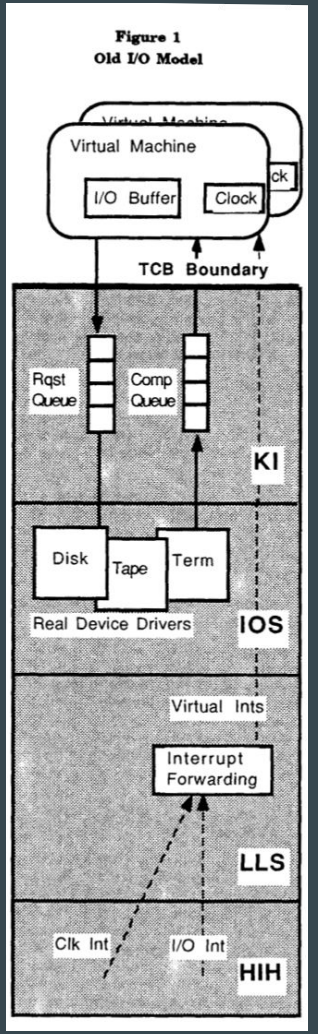
Fuzzy time for the VAX security kernel

- VAX VMM
  - Single thread per VM
  - Clean VM interface
- All I/O is asynchronous



Fuzzy time - Problem

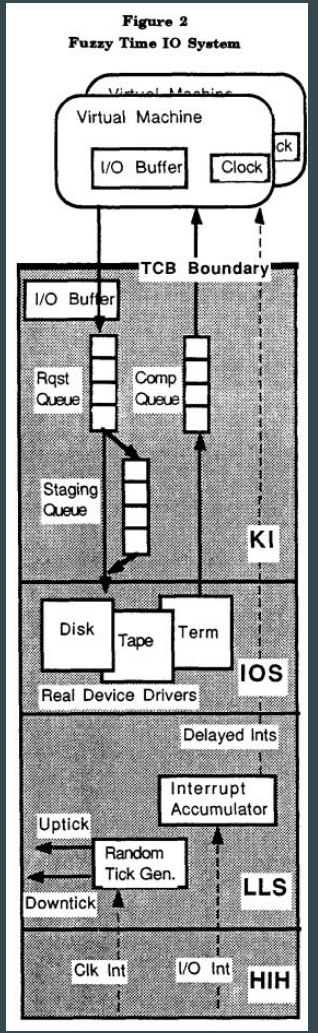
- Ineffective countermeasures to disk covert channel
  - Cannot be closed
  - Not auditable
  - Added noise impractical
  - No hardware solution
- Plenty of other potential ‘shared buses’



Fuzzy time - Solution

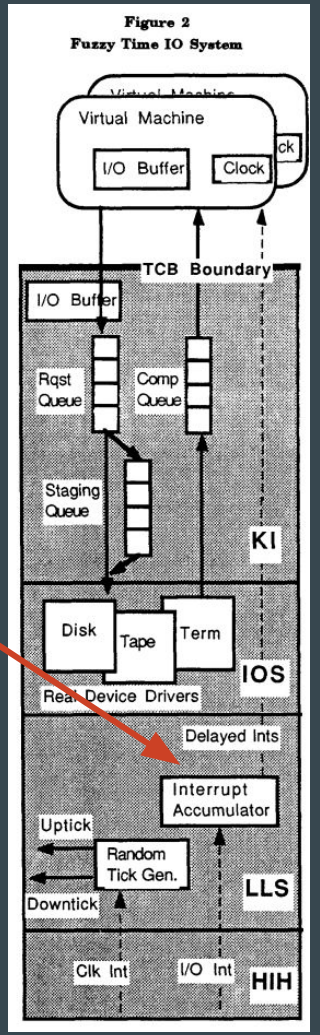
- “reduce the accuracy and precision of system clocks”

- “randomly alter the timings of I/O operations”



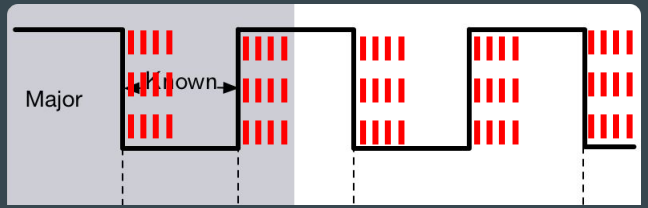
Fuzzy time - Solution

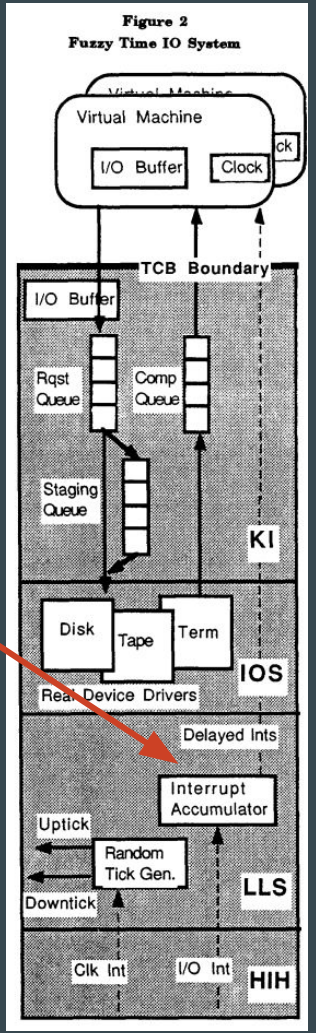
- Explicit clocks
  - “make the interval-timer interrupt random”



Fuzzy time - Solution

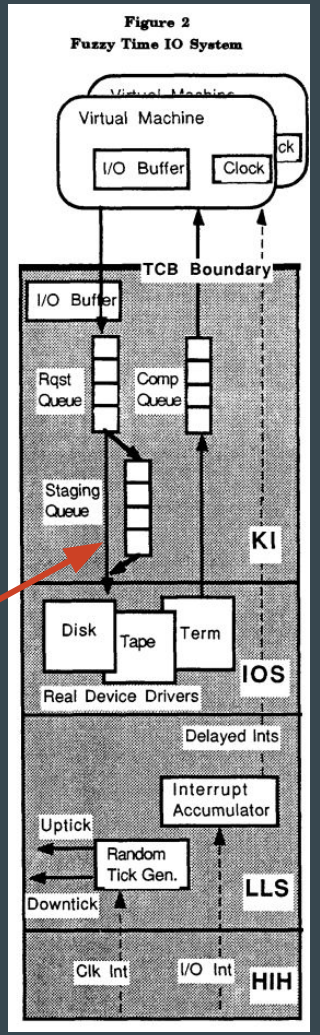
- Explicit clocks
  - “make the interval-timer interrupt random”





Fuzzy time - Solution

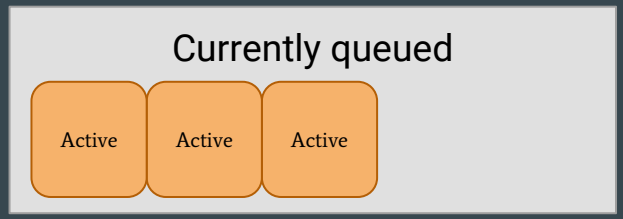
- Explicit clocks
  - “make the interval-timer interrupt random”
- Implicit clocks
  - “[use] random clock ticks ... to make fuzzy the clocks derived from I/O operations”
  - “Add new buffers ... for all I/O operations”

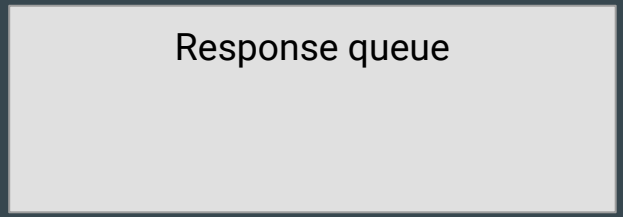


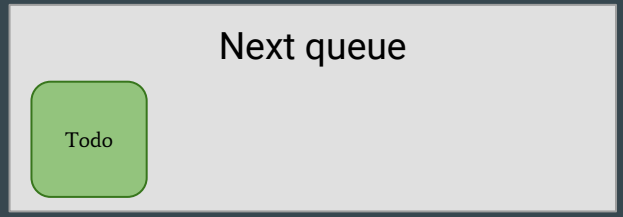
Fuzzy time - Solution guarantees

- Degraded clocks
  - Limit the bandwidth
- Time granularity
  - g
- Bounded channel bandwidth
  - For *any* timing covert channel
  - ~ $\frac{g}{2}$

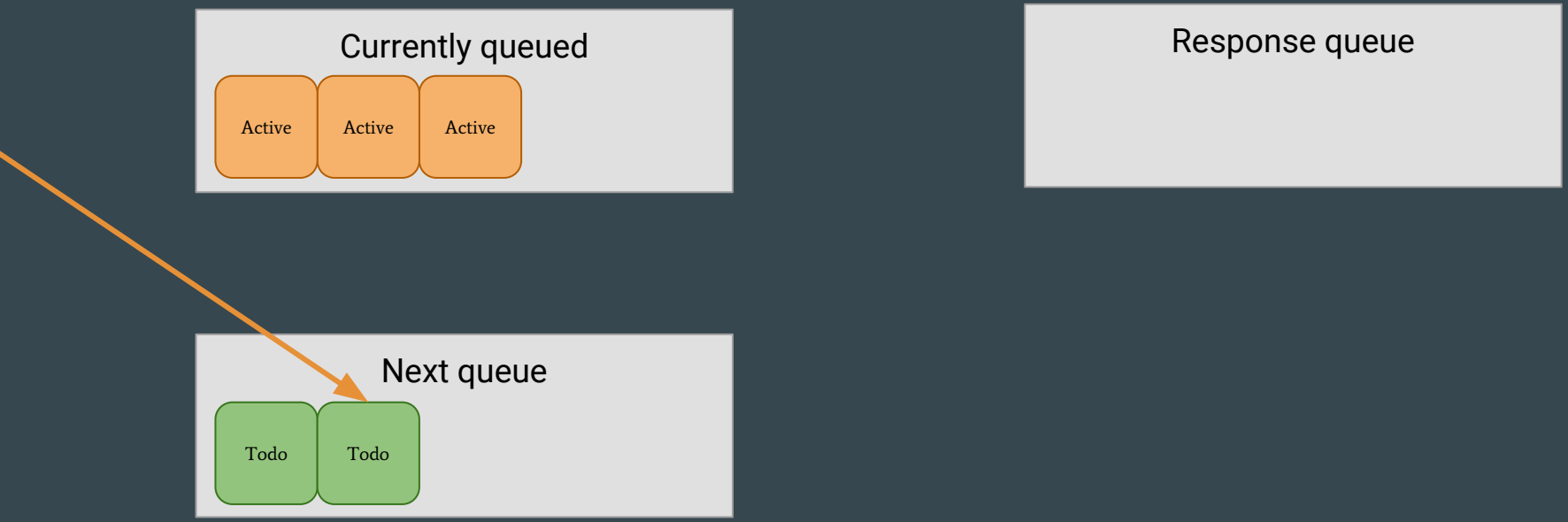
Fuzzy time - I/O queuing



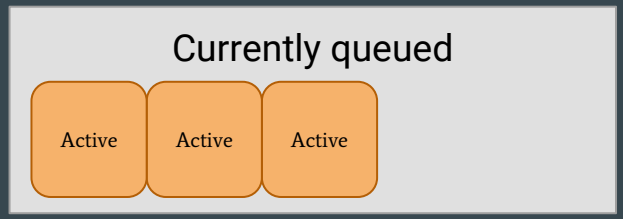


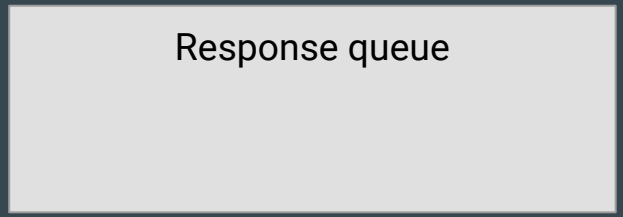


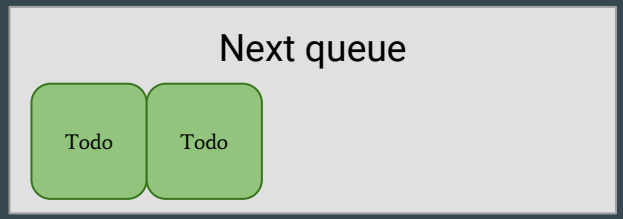
Fuzzy time - I/O queuing



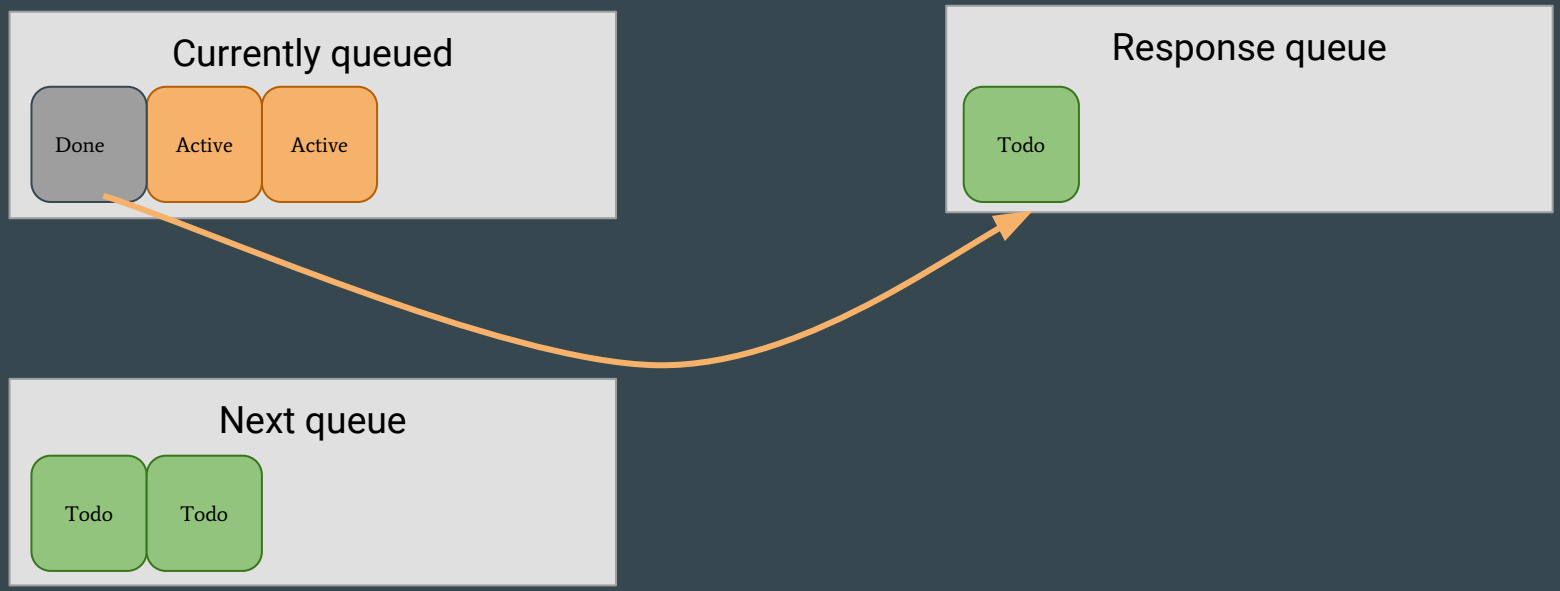
Fuzzy time - I/O queuing



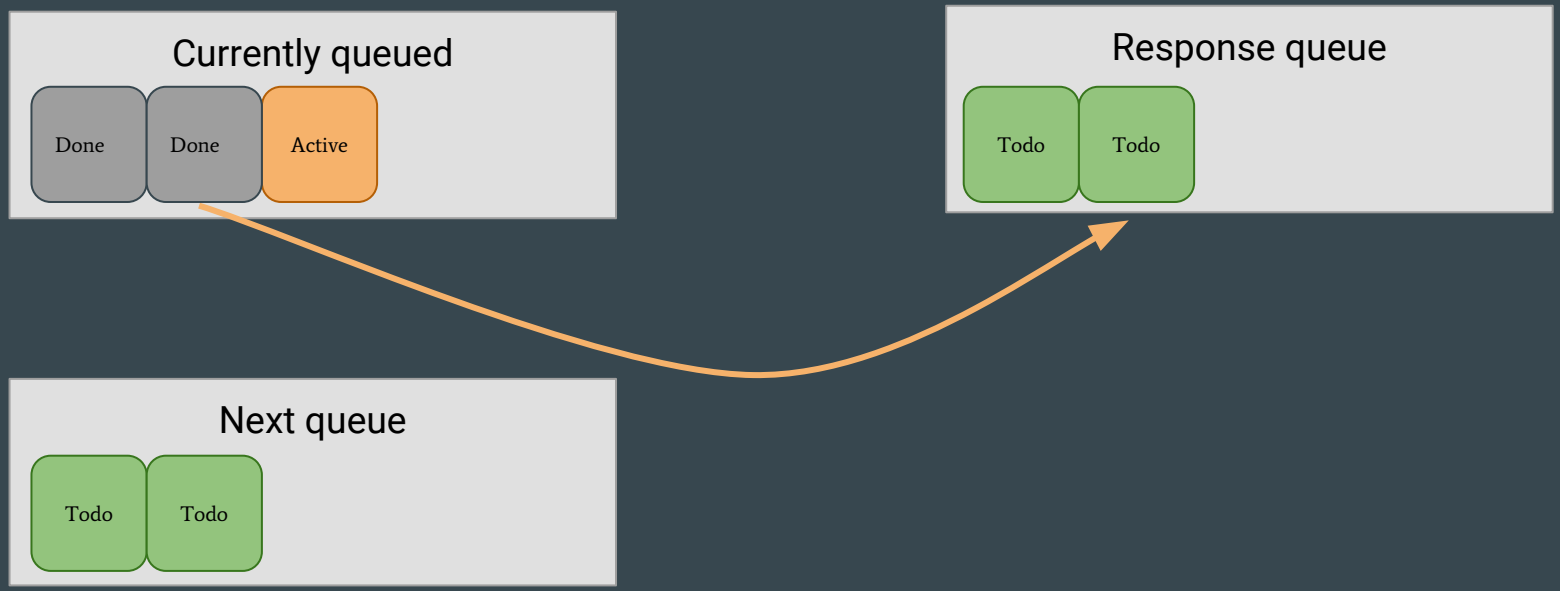




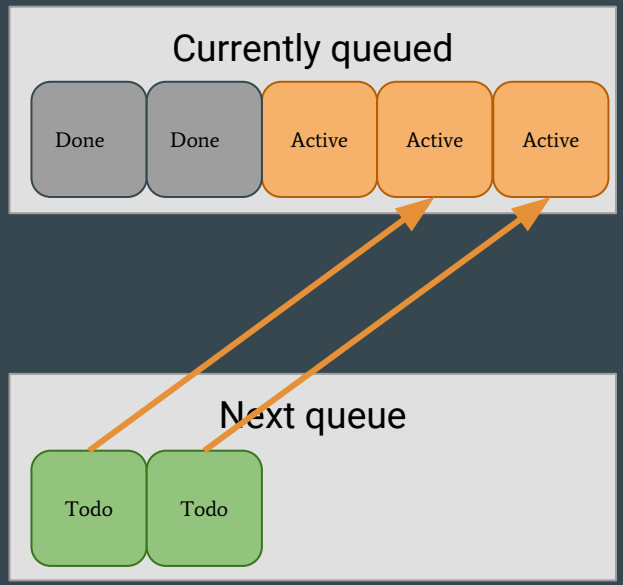
Fuzzy time - I/O queuing

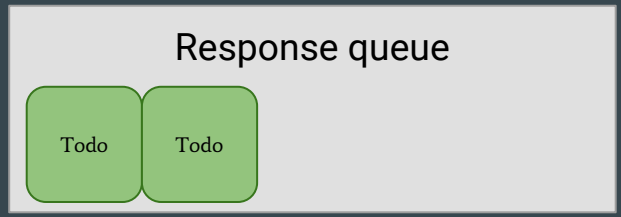


Fuzzy time - I/O queuing

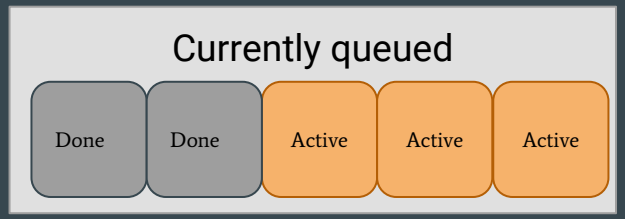


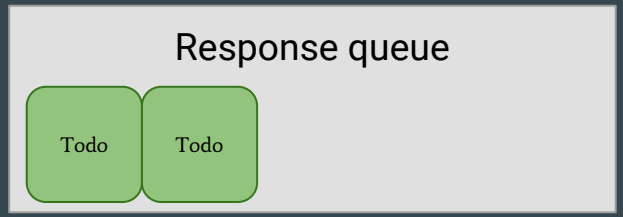
Fuzzy time - I/O queuing

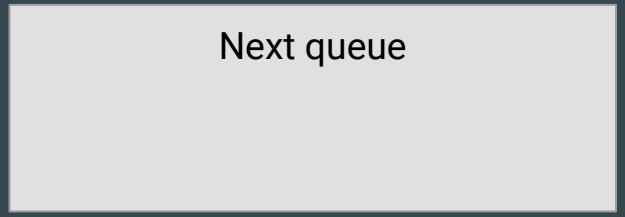




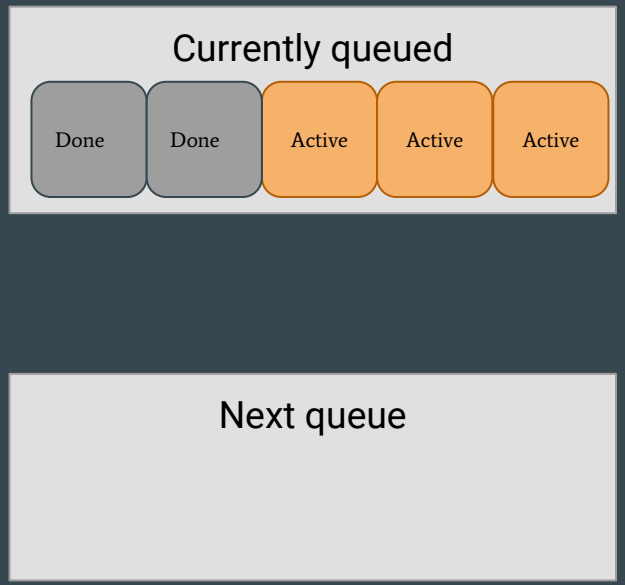
Fuzzy time - I/O queuing



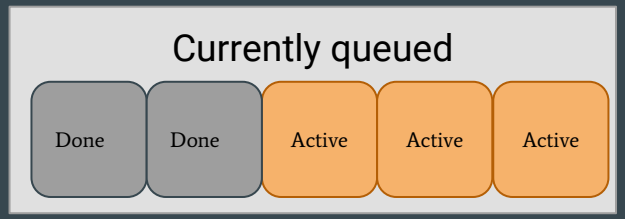


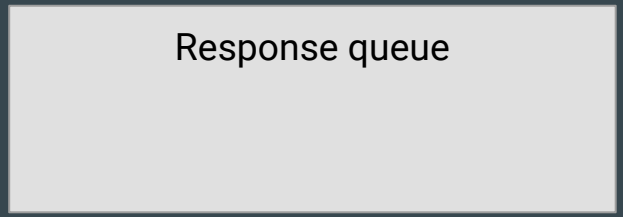


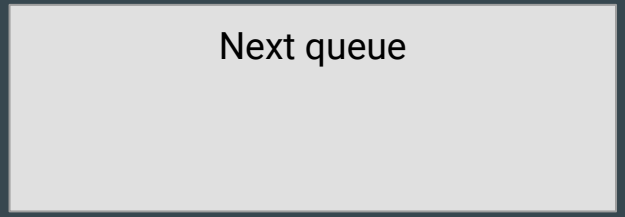
Fuzzy time - I/O queuing



Fuzzy time - I/O queuing







Fermata

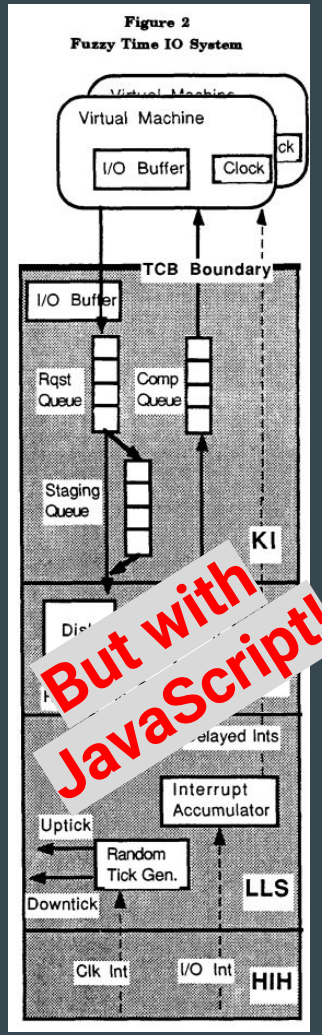
- Time and web browsers
- Mitigating attacks
- A trusted browser
- A (less) trusted browser

Fermata - Why adapt fuzzy time?

- Degrade clocks
  - Slow down attacks
- Verifiability
- Browsers are uniquely well suited

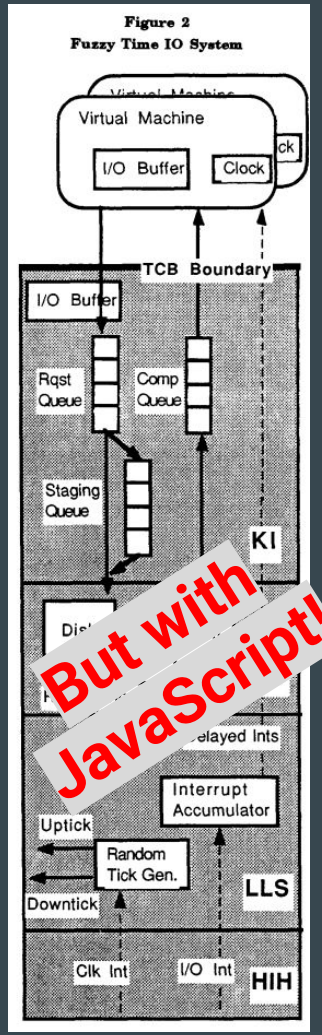
Fermata - Fuzzy time for browsers

- Adapt the VAX fuzzy time model to JS etc!
- Put all I/O operations into queues
- Make all the explicit clocks fuzzy
- Prove everything falls into a fuzzy time defense



Fermata - Fuzzy time for browsers

- Adapt the VAX fuzzy time model to JS etc!
- Put all I/O operations into queues
- Make all the explicit clocks fuzzy
- Prove everything falls into a fuzzy time defense
- Change all DOM accesses to be asynchronous!



Fuzzyfox

Rationale and design

- Time and web browsers
- Mitigating attacks
- A trusted browser
- A (less) trusted browser

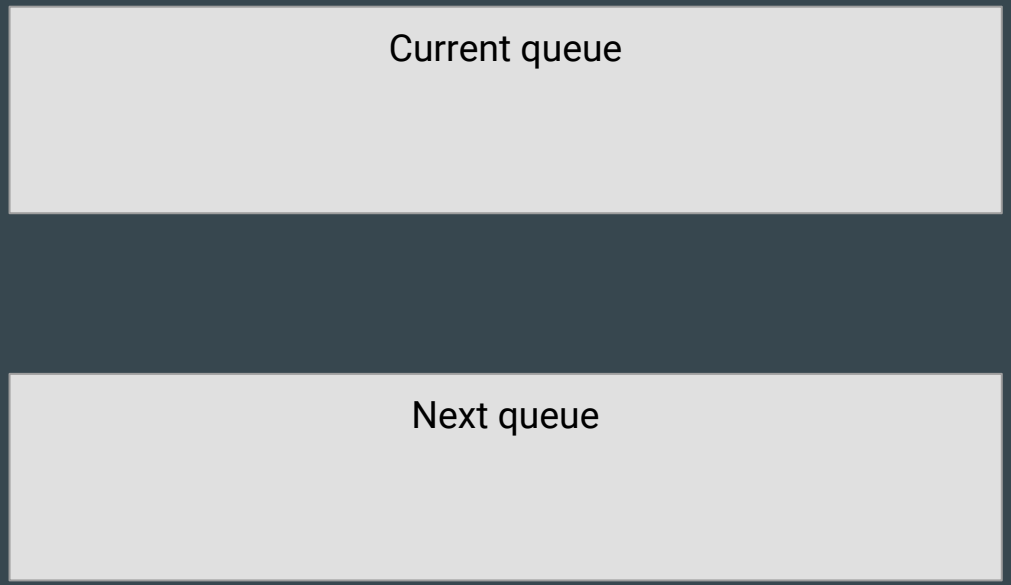
Why we didn’t build Fermata

1. We didn’t know if it would work
2. We didn’t know what to start with
3. We want to push mitigations to real browsers

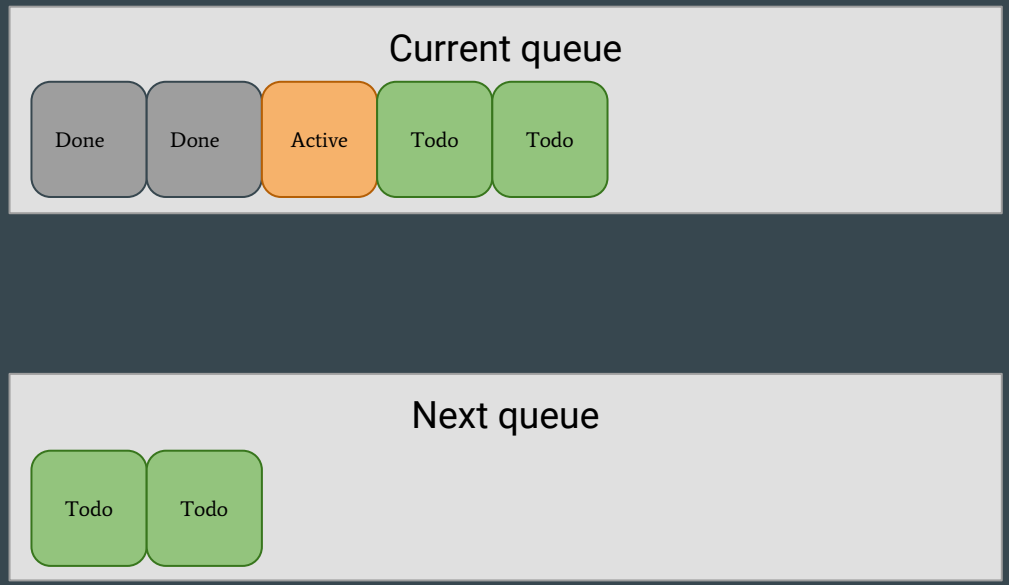
Fuzzyfox

- Patch set on trunk Mozilla Firefox
- Supports multiple clock granularities
  - Tested 0.5ms to 100ms
- Fully fuzzes explicit clocks
- Breaks main thread into ‘ticks’
- Delays outgoing HTTP request start

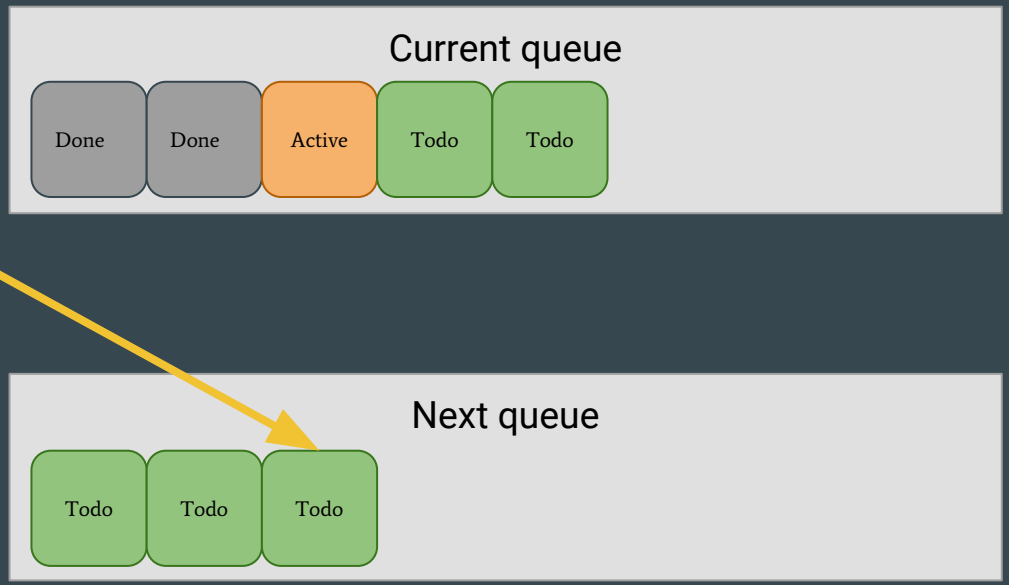
Fuzzyfox - Main thread queuing



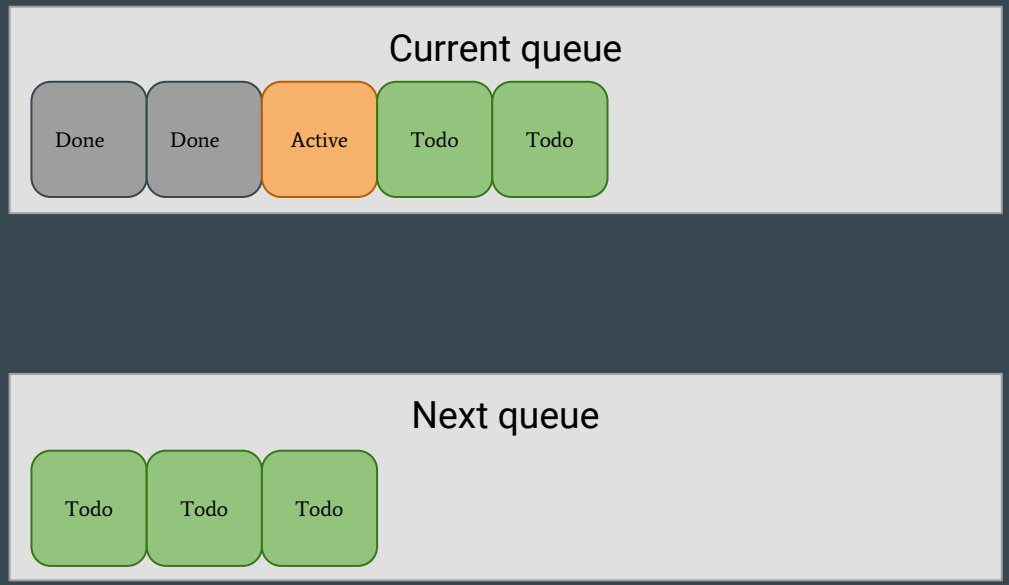
Fuzzyfox - Main thread queuing



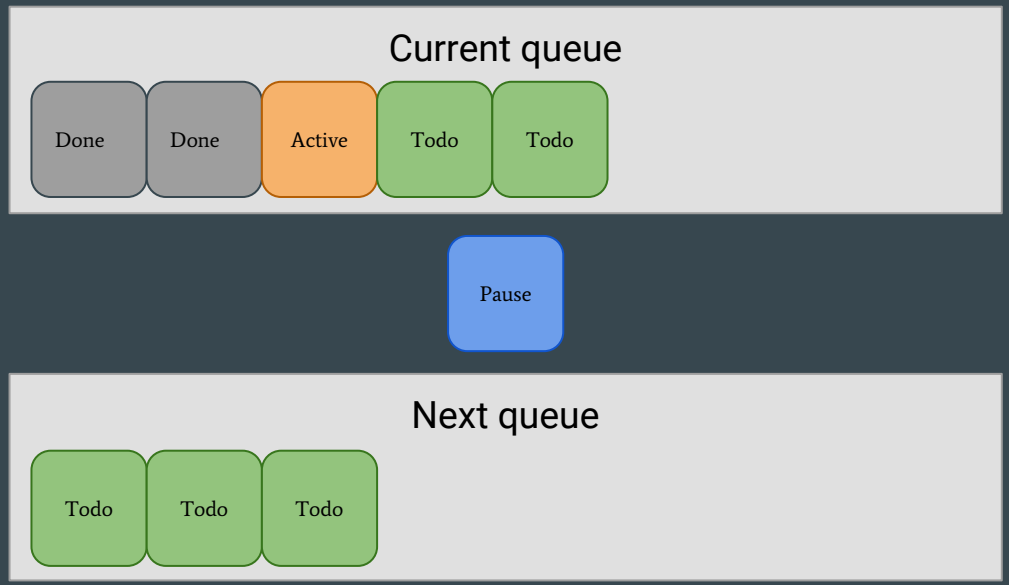
Fuzzyfox - Main thread queuing



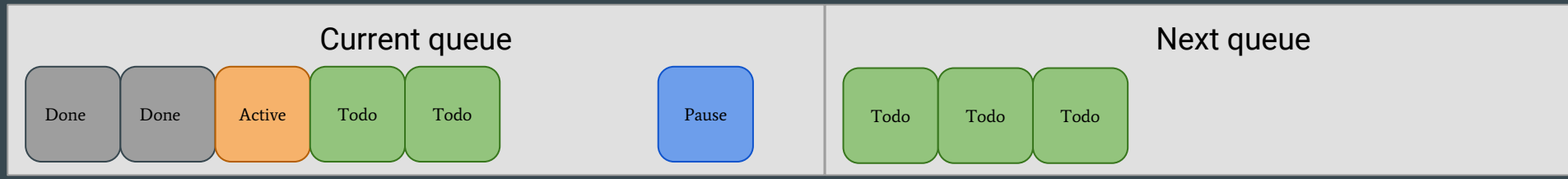
Fuzzyfox - Main thread queuing



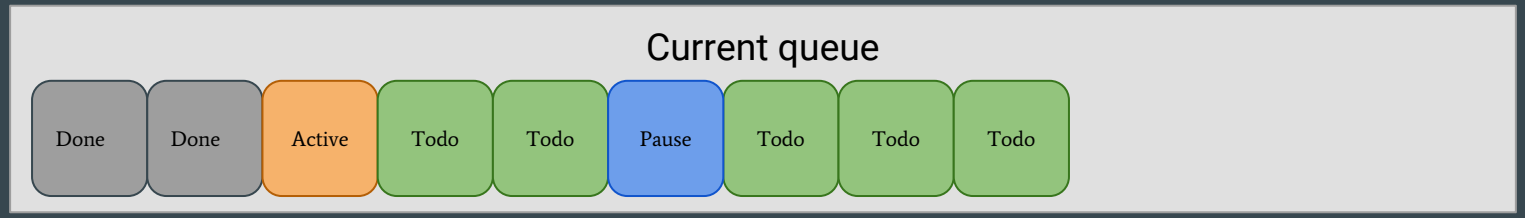
Fuzzyfox - Main thread queuing



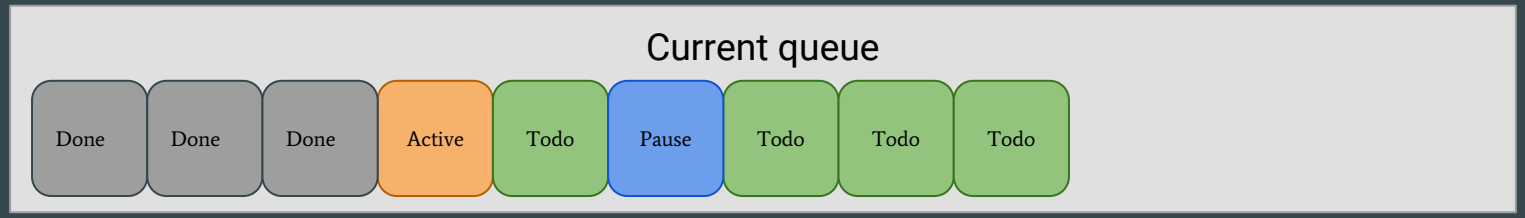
Fuzzyfox - Main thread queuing



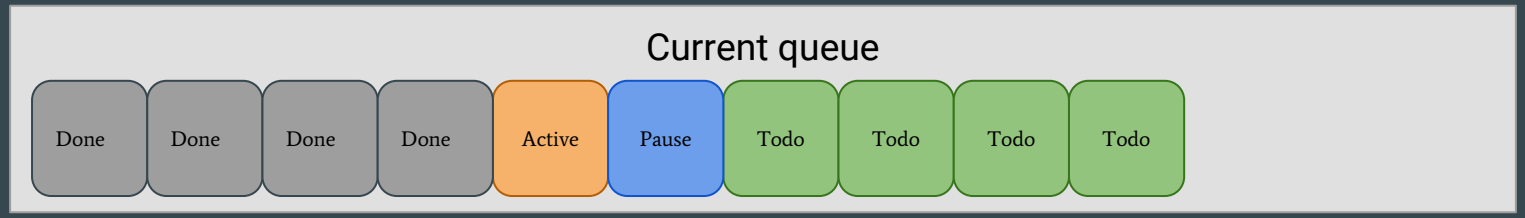
Fuzzyfox - Main thread queuing



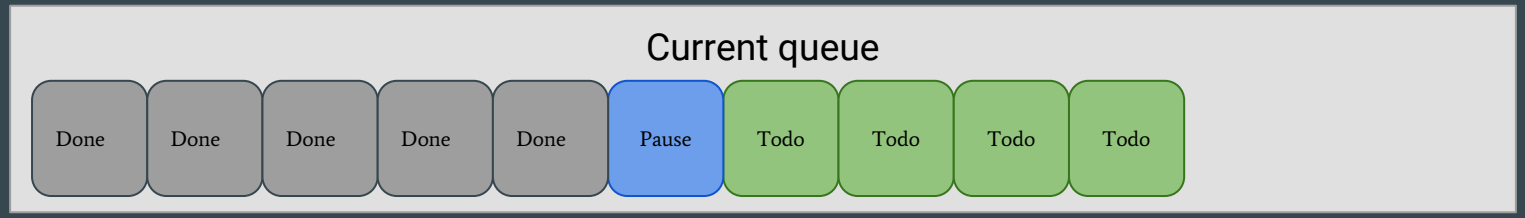
Fuzzyfox - Main thread queuing



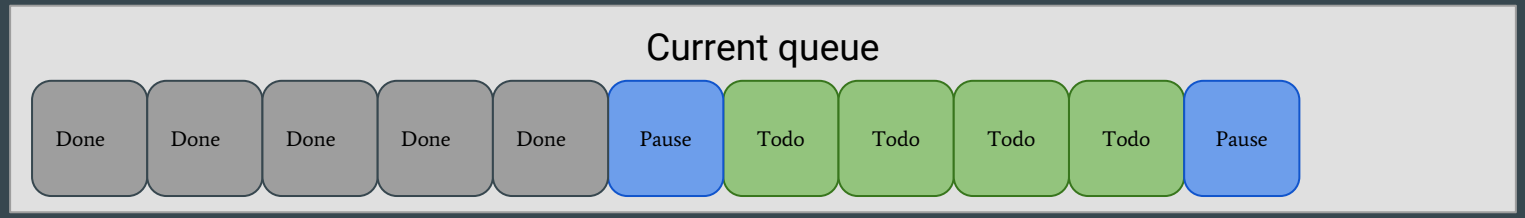
Fuzzyfox - Main thread queuing



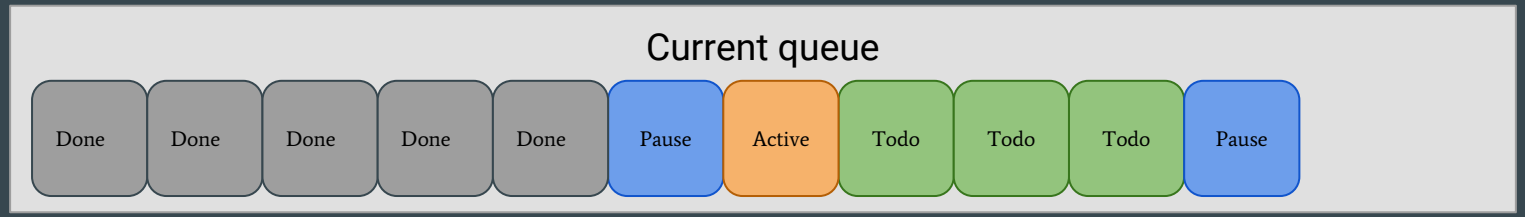
Fuzzyfox - Main thread queuing



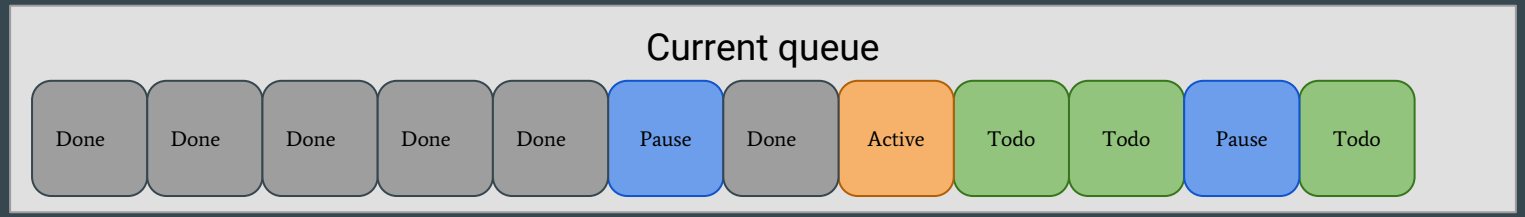
Fuzzyfox - Main thread queuing



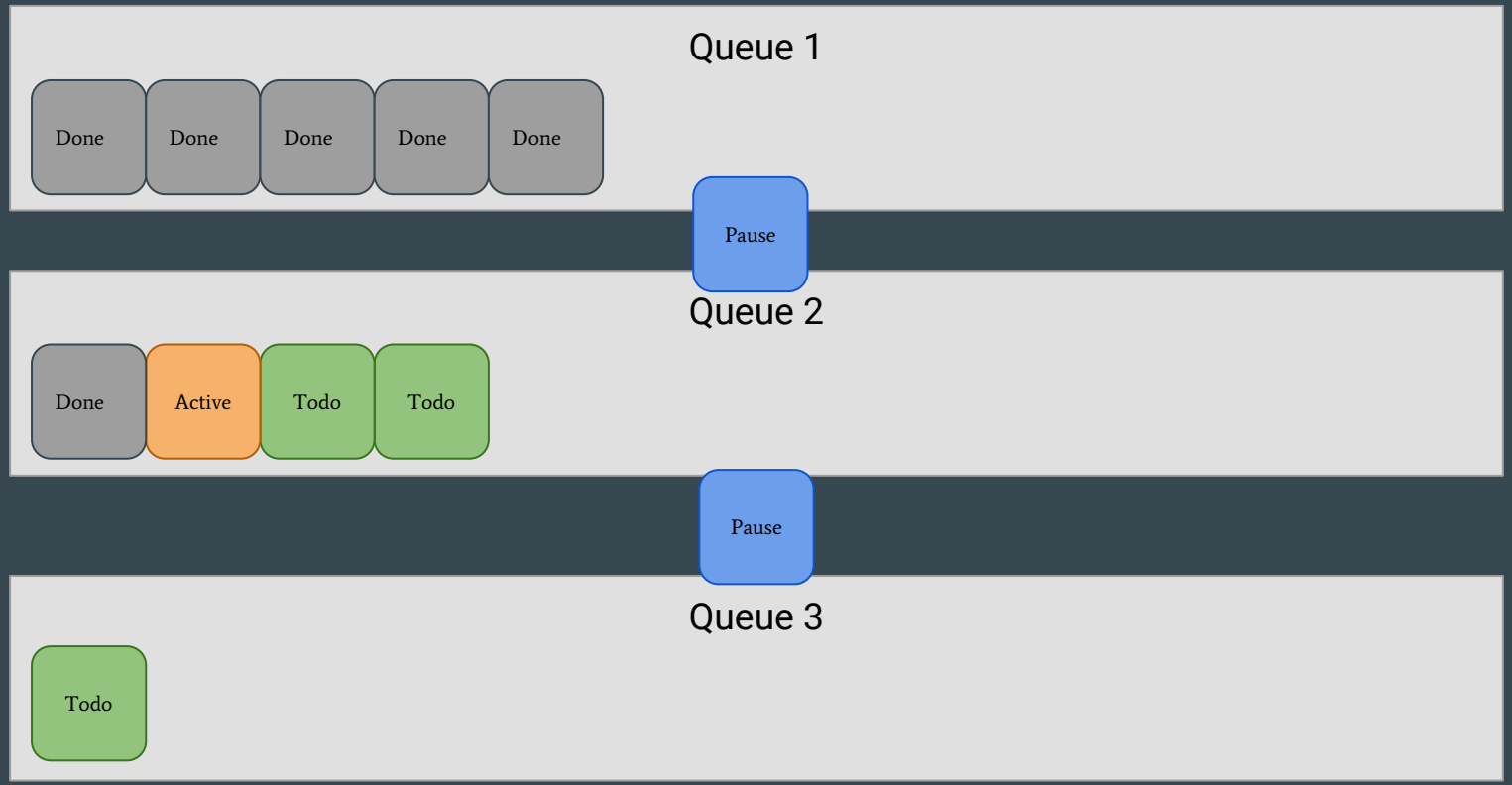
Fuzzyfox - Main thread queuing



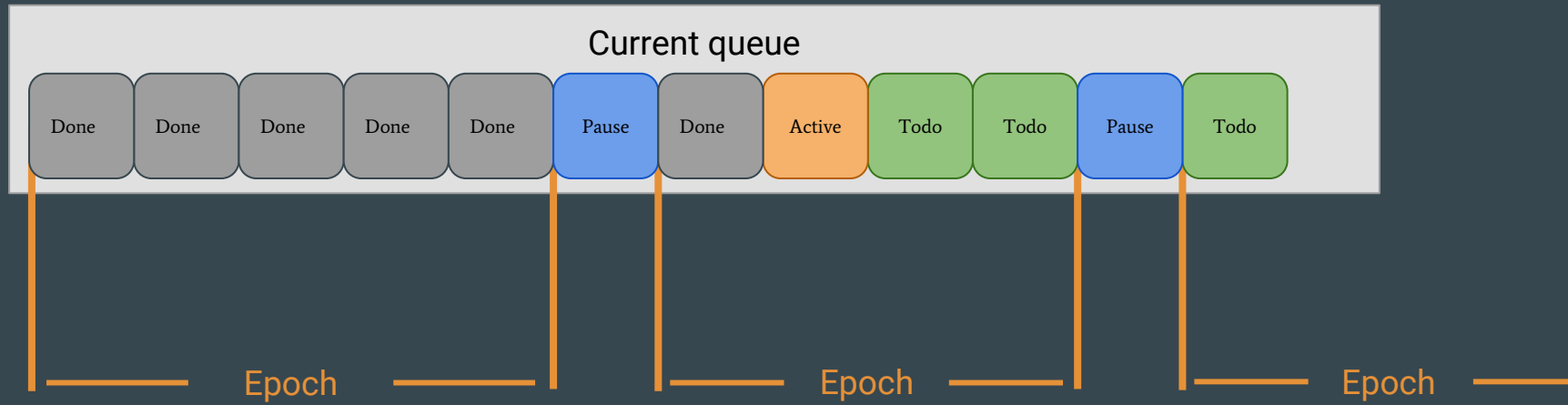
Fuzzyfox - Main thread queuing



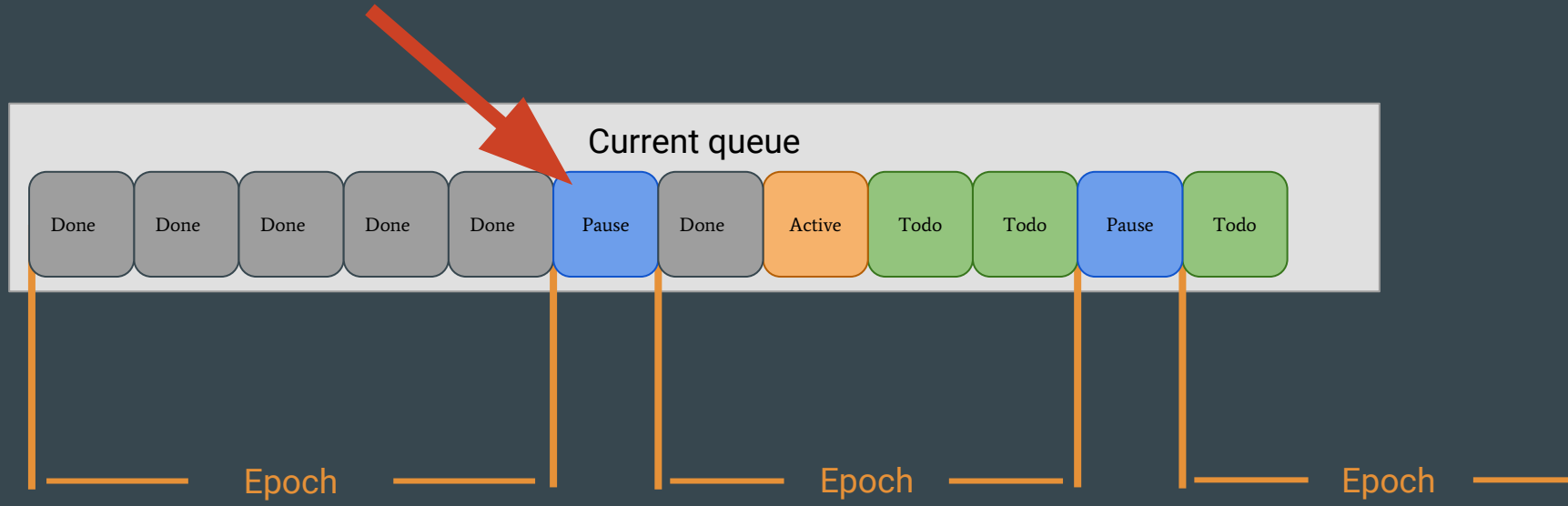
Fuzzyfox - Main thread queuing



Fuzzyfox - Main thread queuing

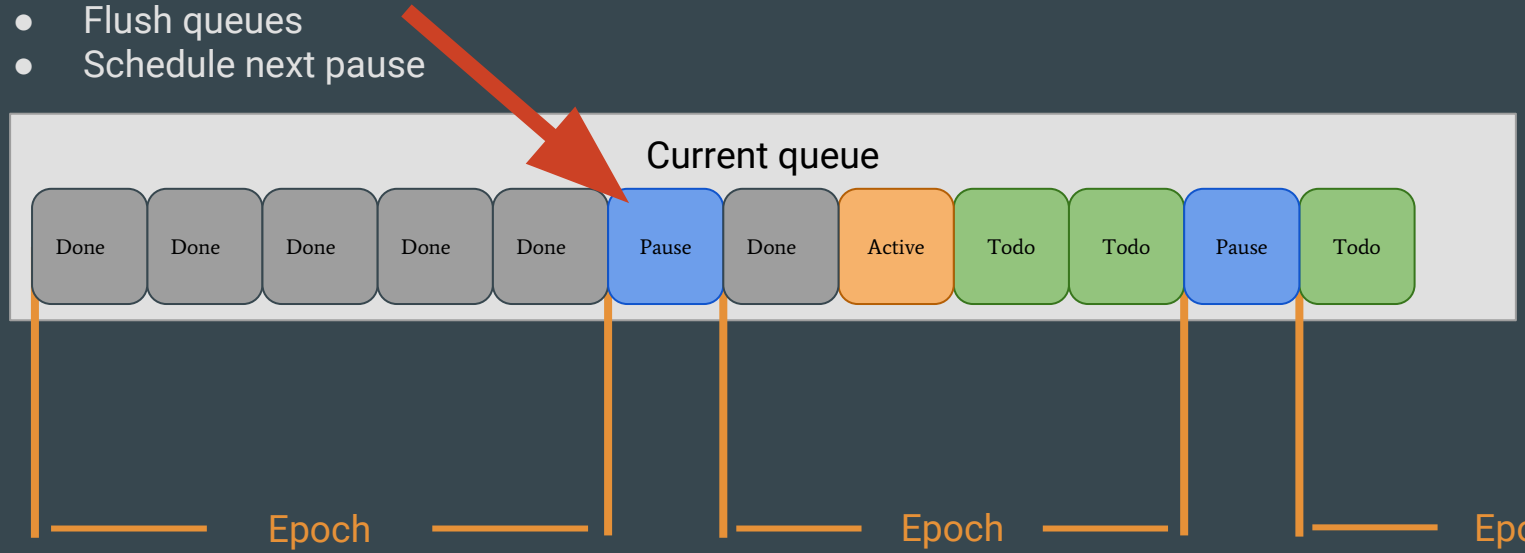


Fuzzyfox - Main thread queuing



Fuzzyfox - Main thread queuing

- Sleep
- Update clocks
- Flush queues
- Schedule next pause



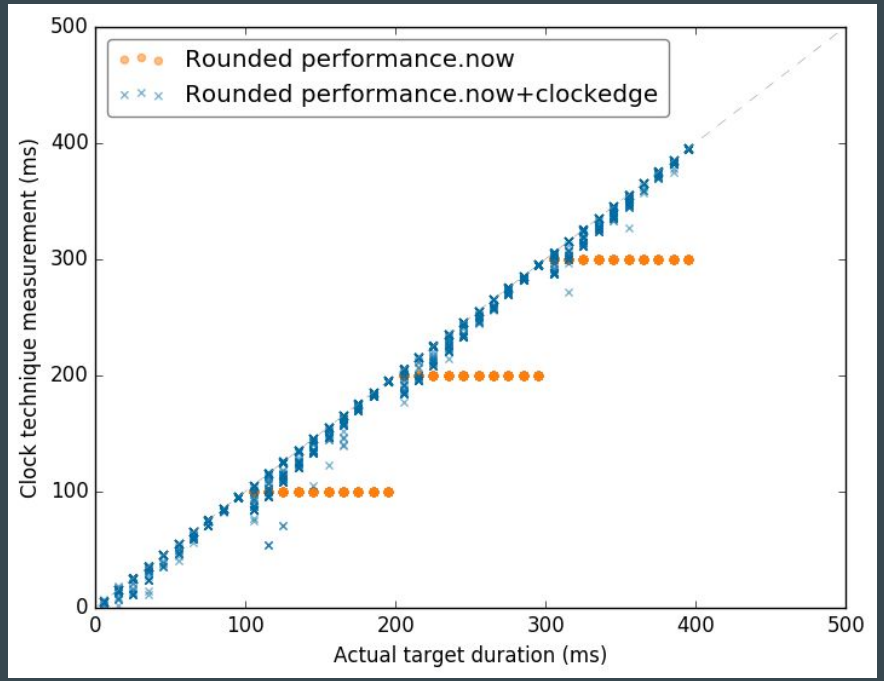
Fuzzyfox

Effectiveness

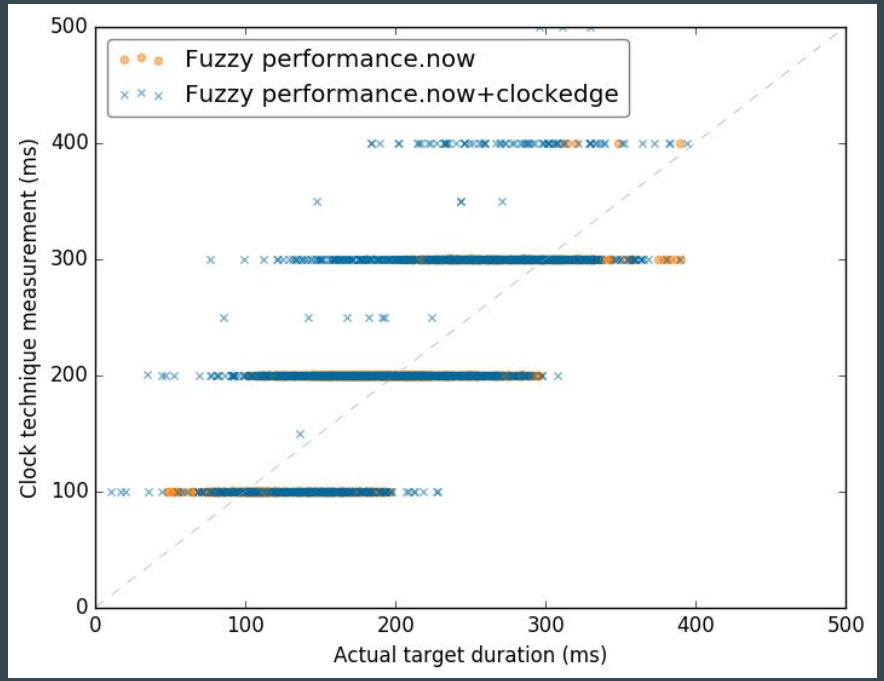
- Time and web browsers
- Mitigating attacks
- A trusted browser
- A (less) trusted browser

Fuzzyfox - Effectiveness - Explicit - performance.now()

Firefox

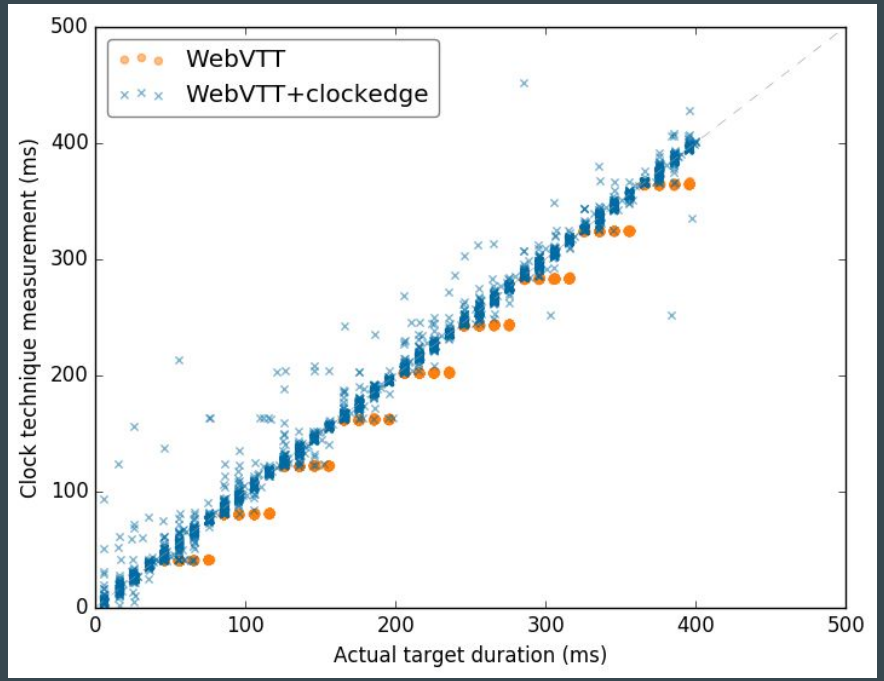


Fuzzyfox

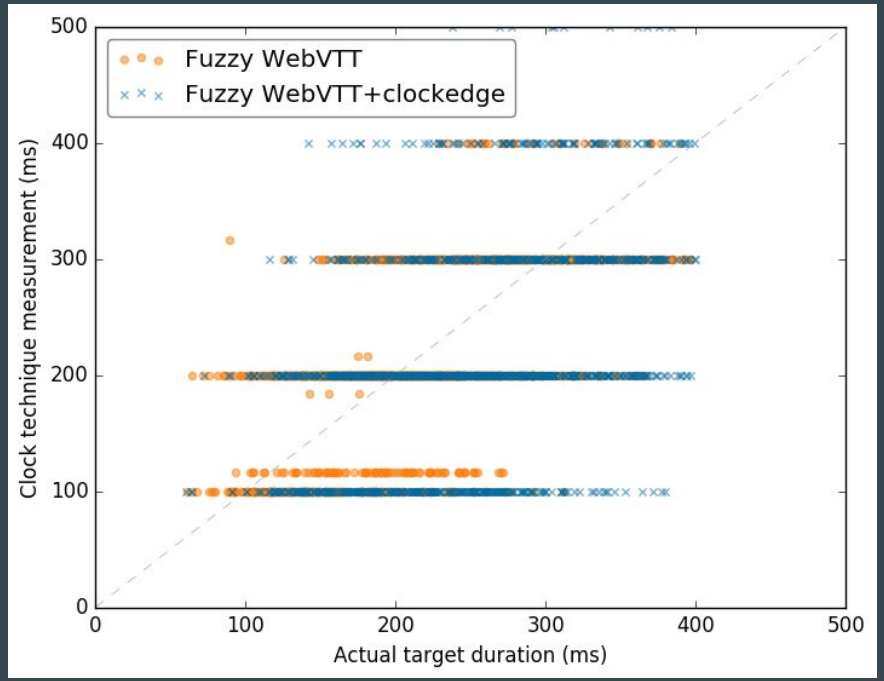


Fuzzyfox - Effectiveness - Implicit - WebVTT clock

Firefox



Fuzzyfox



Fuzzyfox

Performance

- Time and web browsers
- Mitigating attacks
- A trusted browser
- A (less) trusted browser

Fuzzyfox - Performance

- “Micro” performance
  - Synthetic microbenchmark page load times
- “Macro” performance
  - Real website load times
- Interactivity
  - User study

Fuzzyfox - Performance

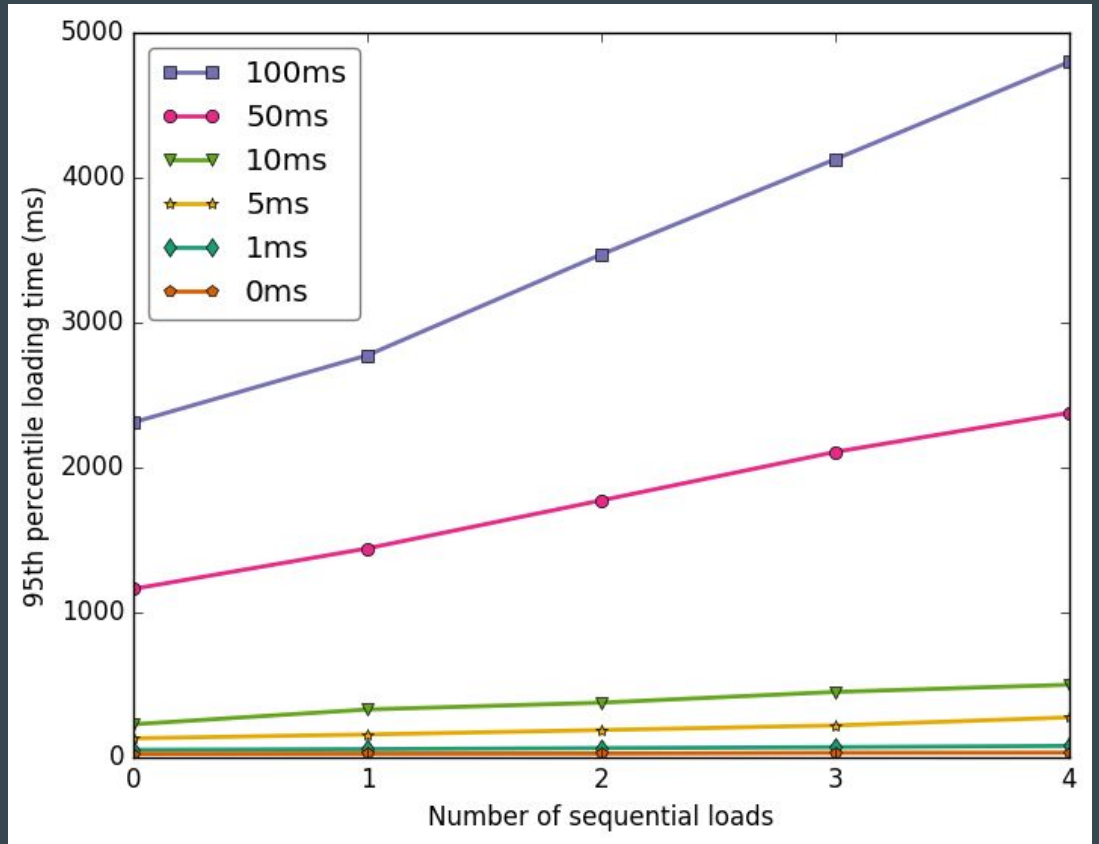
- “Micro” performance
  - Synthetic microbenchmark page load times
- “Macro” performance
  - Real website load times
- Interactivity
  - ~~User study~~

Fuzzyfox - Performance - Micro benchmarks

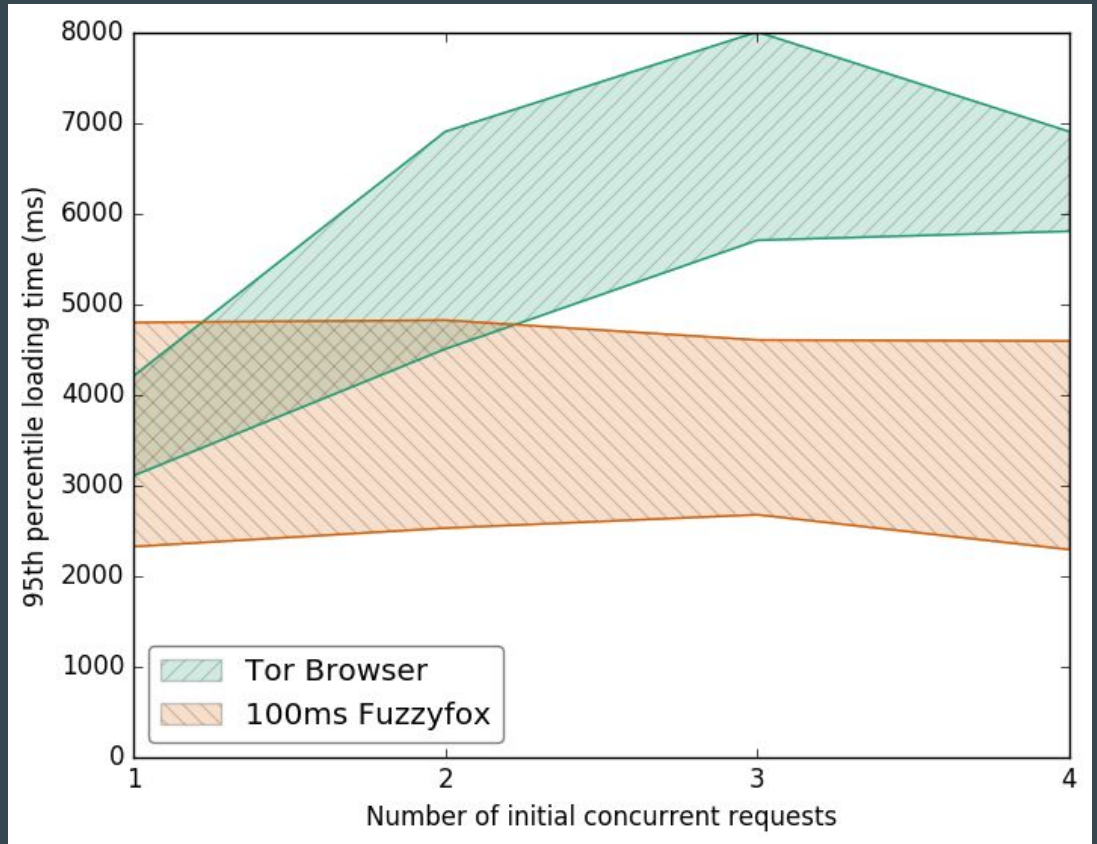
- Page load times
  - As reported by `onload()`

- Measured effects of
  - Sequential resource loads
  - Parallel resource loads

Fuzzyfox - Performance - Sequential loads



Fuzzyfox - Performance vs Tor Browser



Takeaways

- Time and web browsers
- Mitigating attacks
- A trusted browser
- A (less) trusted browser

Timing attacks

Rounding clocks doesn't work

- Time and web browsers
- Mitigating attacks
- A trusted browser
- A (less) trusted browser

Fuzzy time

Secure operating systems tech

- Time and web browsers
- Mitigating attacks
- A trusted browser
- A (less) trusted browser

Fermata

A different design for the browser

- Time and web browsers
- Mitigating attacks
- A trusted browser
- A (less) trusted browser

Fuzzyfox

Defenses that can work and that we can deploy

- Time and web browsers
- Mitigating attacks
- A trusted browser
- A (less) trusted browser

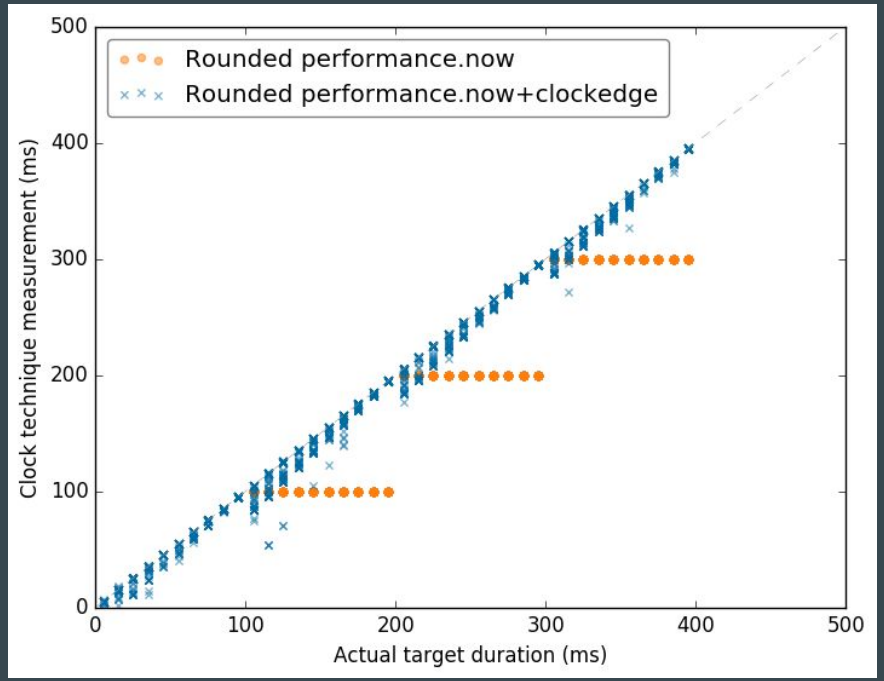
Takeaways

- Time and web browsers
- Mitigating attacks
- A trusted browser
- A (less) trusted browser

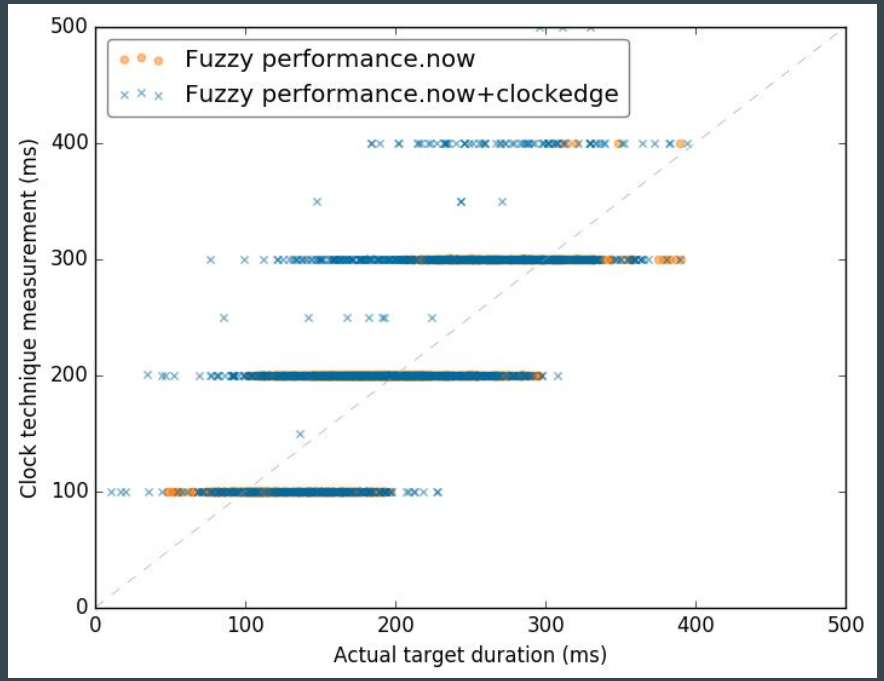
This material is based upon work supported by
the National Science Foundation and by a gift from Mozilla.
We thank Kyle Huey, Patrick McManus, Eric Rescorla,
and Martin Thomson at Mozilla for helpful discussions
about this work, and for sharing their insights with us
about Firefox internals.

Fuzzyfox - Effectiveness - Explicit - performance.now()

Firefox

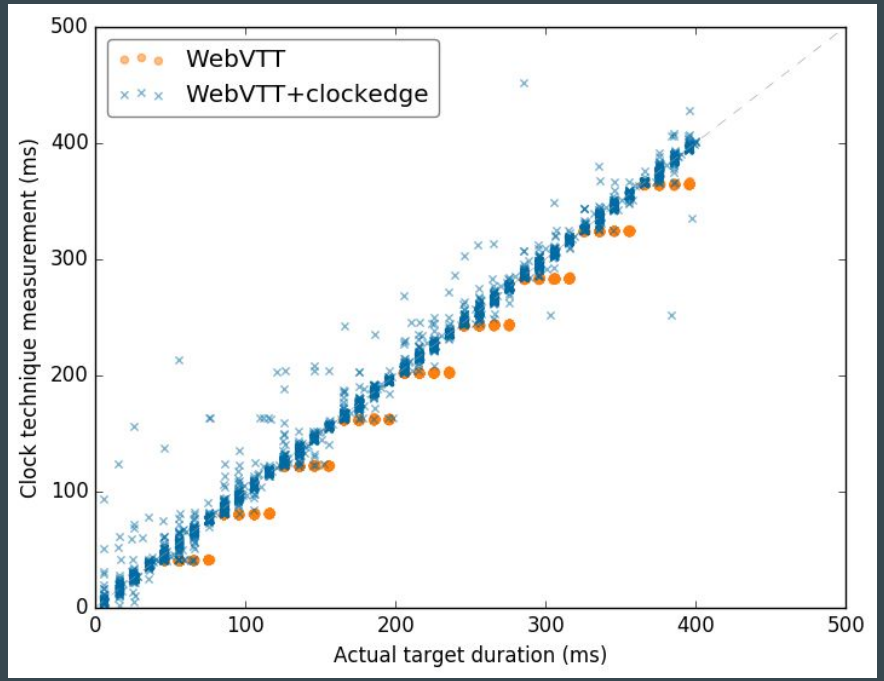


Fuzzyfox

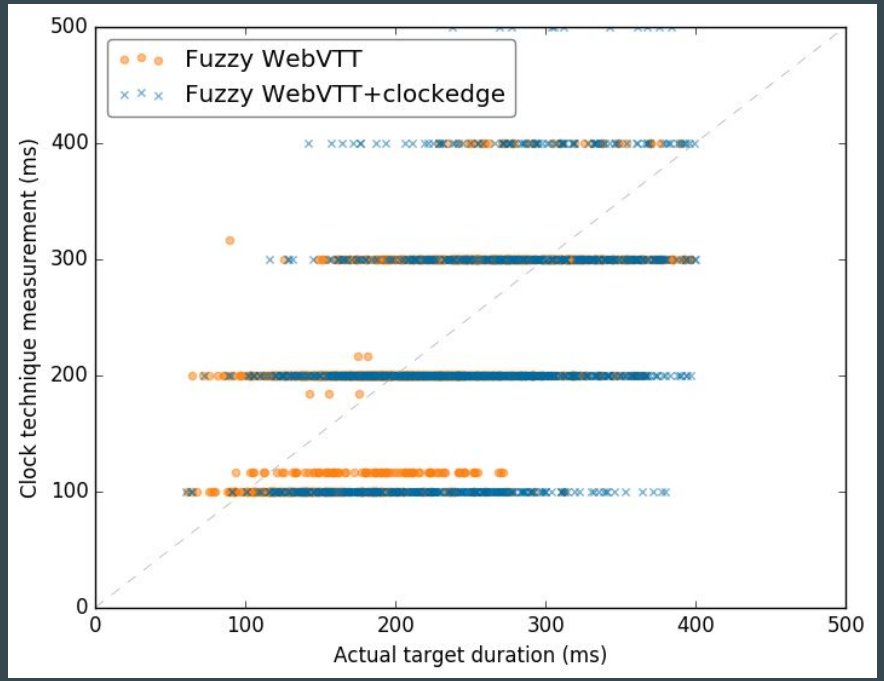


Fuzzyfox - Effectiveness - Implicit - WebVTT clock

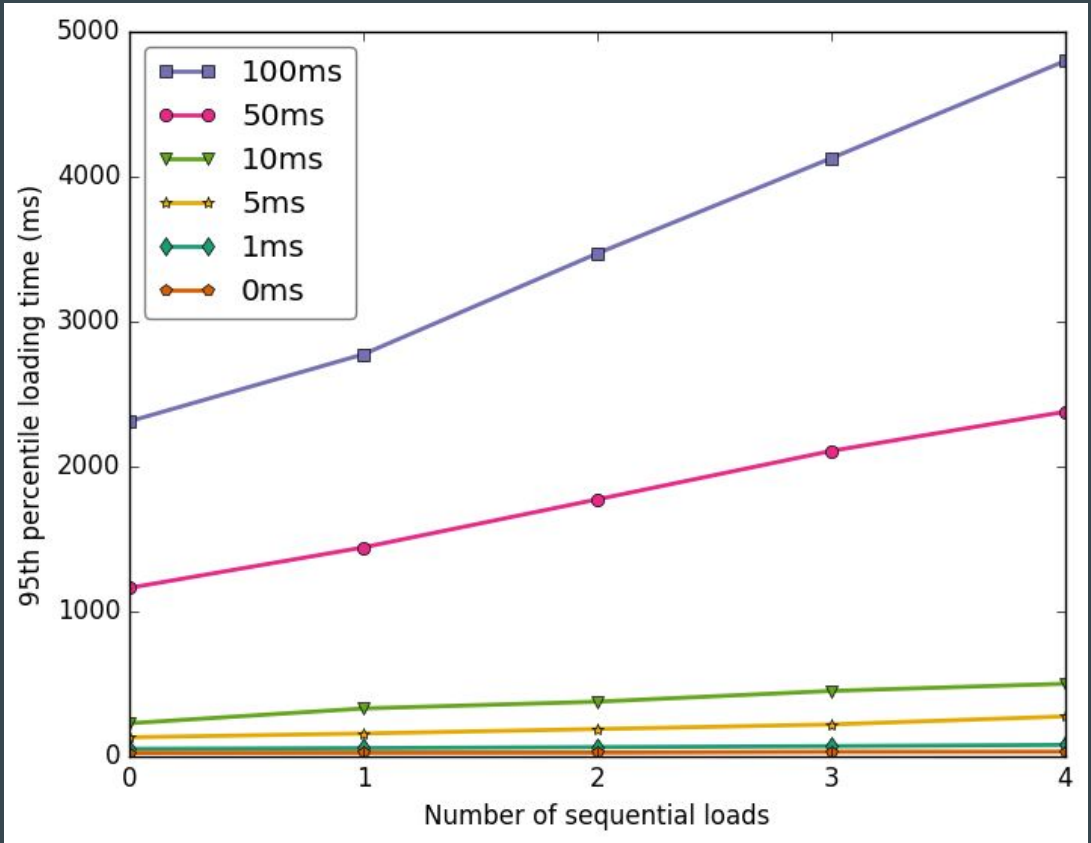
Firefox



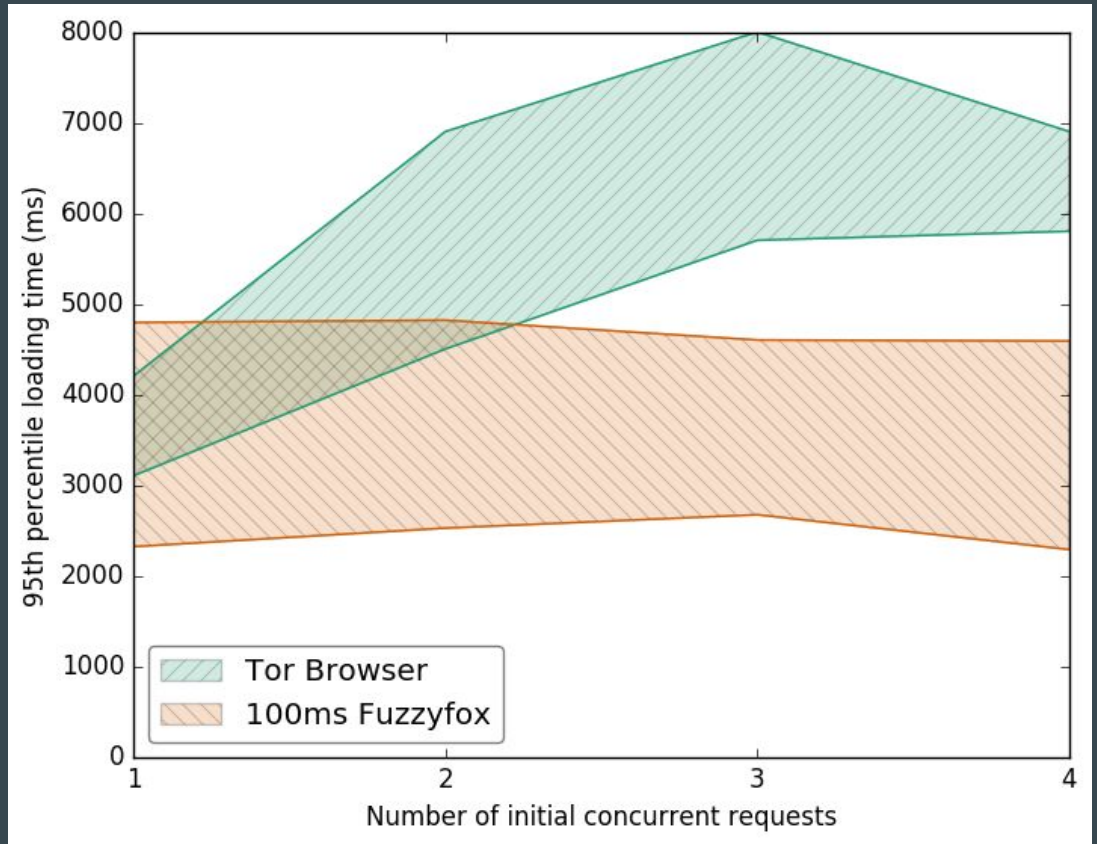
Fuzzyfox



Performance - Micro benchmarks - Sequential loads



Performance - Micro benchmarks - Tor Browser



Performance - Load times* - Google search