Loophole: Timing Attacks on Shared Event Loops in Chrome

Pepe Vila and Boris Köpf

institute imdea software

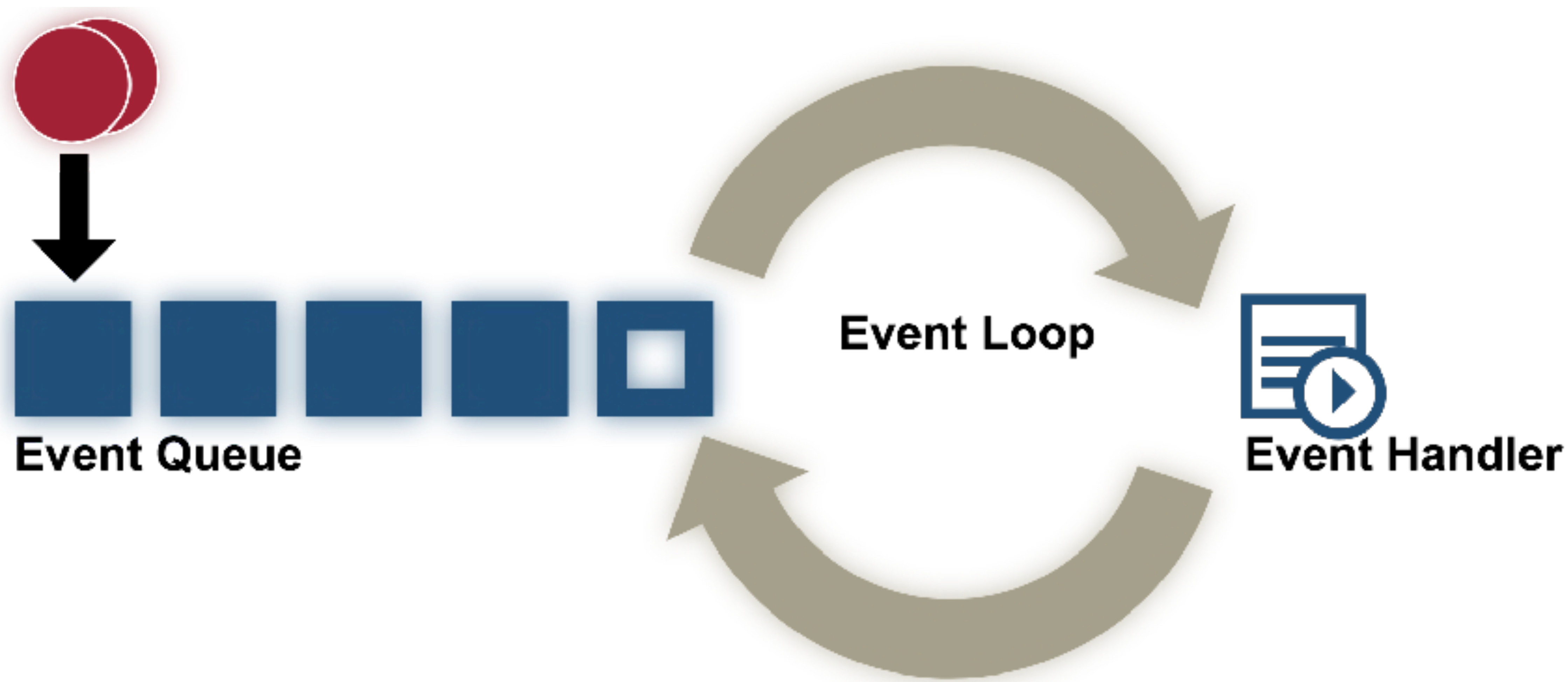
vwzq.net

@cgvwzq

github.com/cgvwzq


EVENT DRIVEN PROGRAMMING
SO HOT RIGHT NOW


EVENT DRIVEN PROGRAMMING
SO HOT RIGHT NOW



Source: http://berb.github.io/diploma-thesis/original/042_serverarch.html

We exploit 2 different shared Event Loops in Chrome:

We exploit 2 different shared Event Loops in Chrome:

I/O's of the **Host Process**

Main thread's of **Renderers**

We exploit 2 different shared Event Loops in Chrome:

I/O’s of the **Host Process**

Main thread’s of **Renderers**

And implement 3 different attacks:

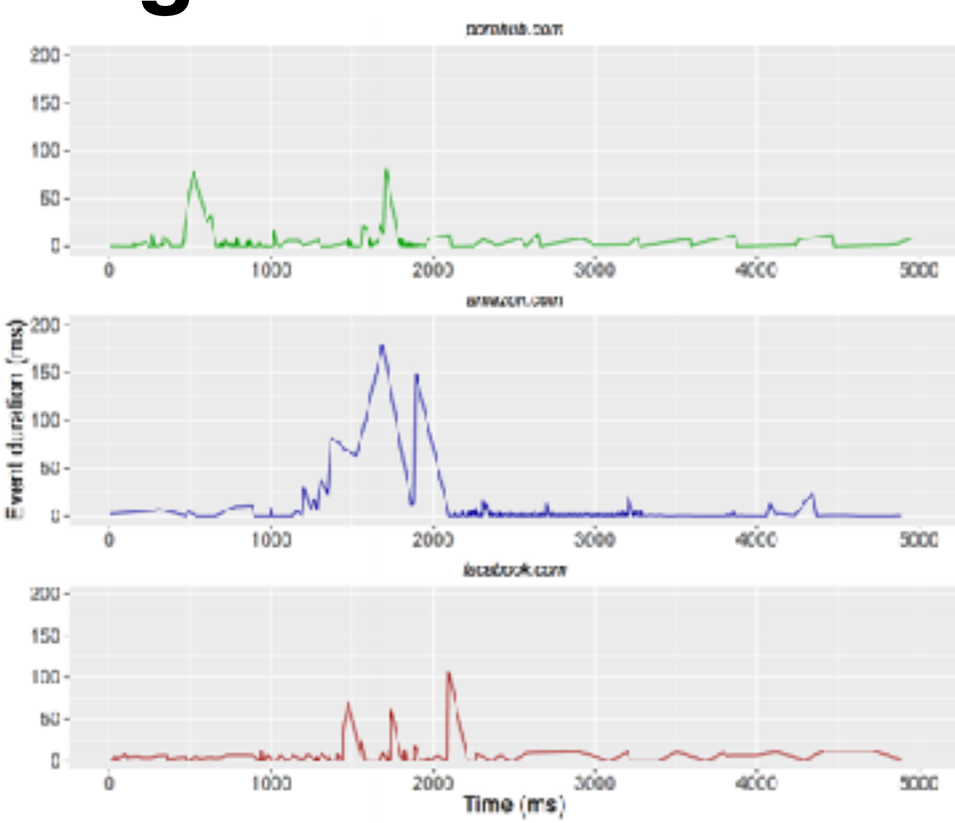
We exploit 2 different shared Event Loops in Chrome:

I/O’s of the **Host Process**

Main thread’s of **Renderers**

And implement 3 different attacks:

Page Identification



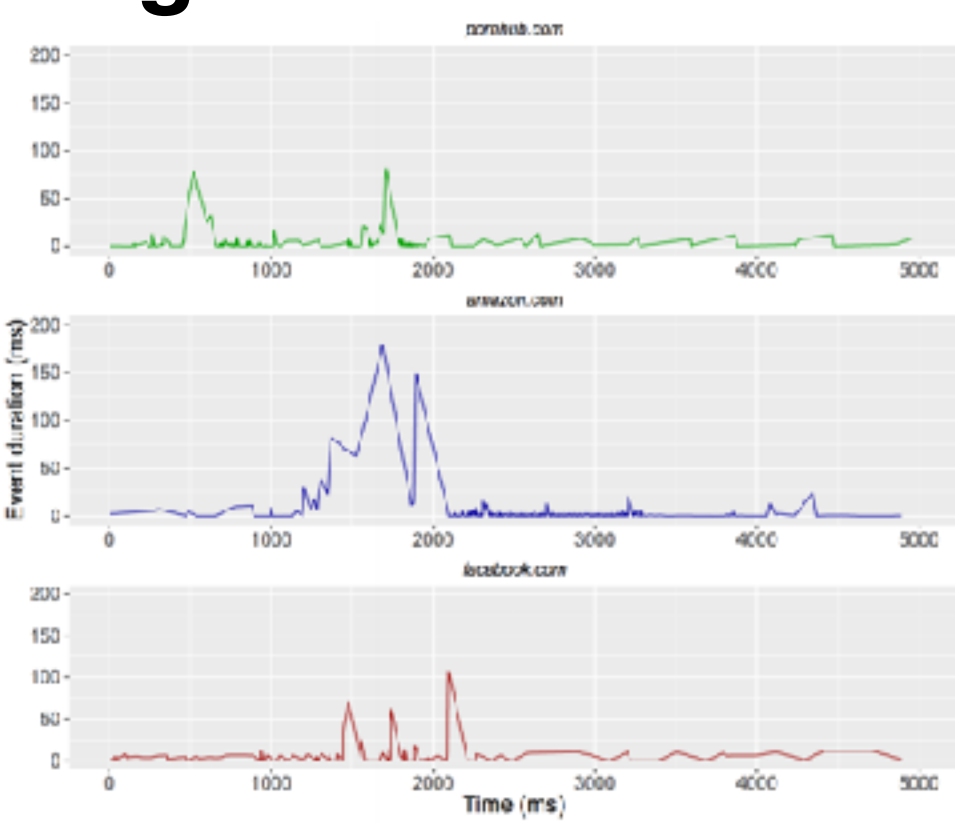
We exploit 2 different shared Event Loops in Chrome:

I/O’s of the **Host Process**

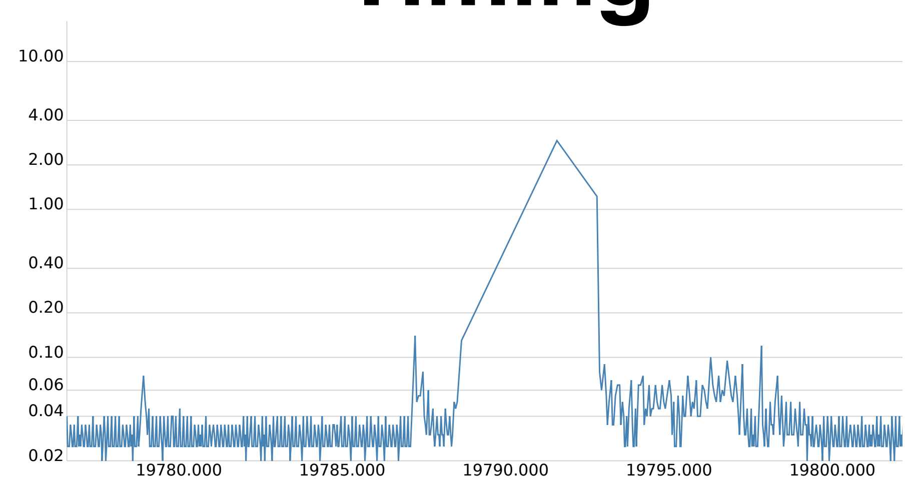
Main thread’s of **Renderers**

And implement 3 different attacks:

Page Identification



Inter-keystroke Timing



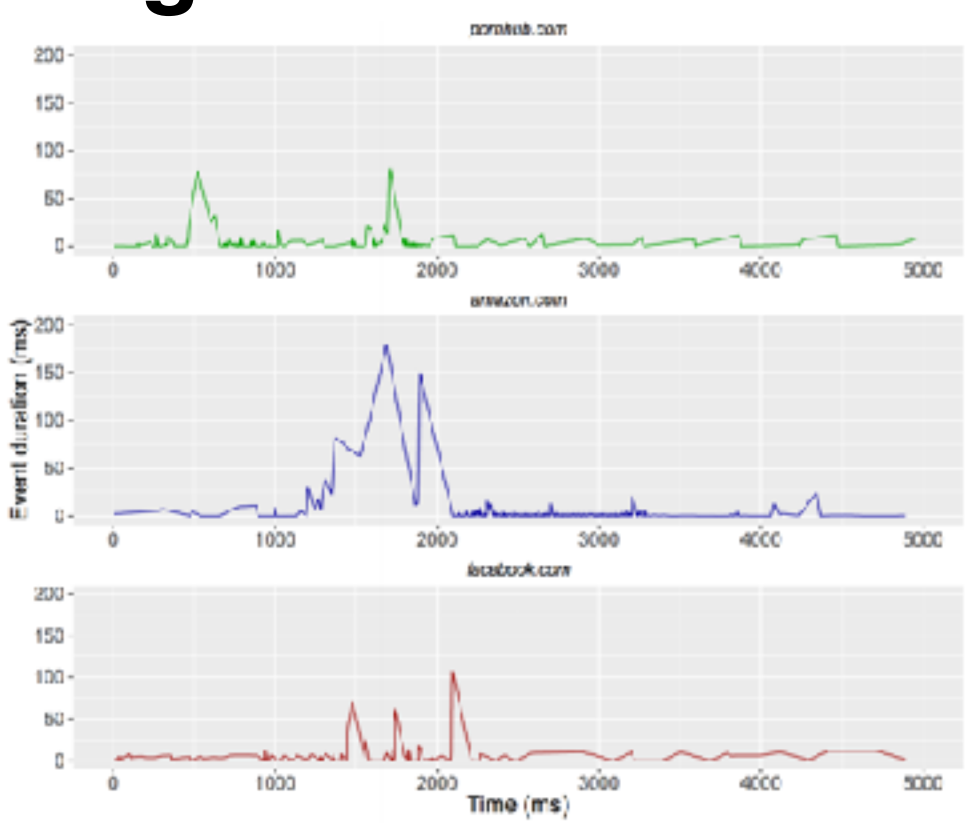
We exploit 2 different shared Event Loops in Chrome:

I/O's of the **Host Process**

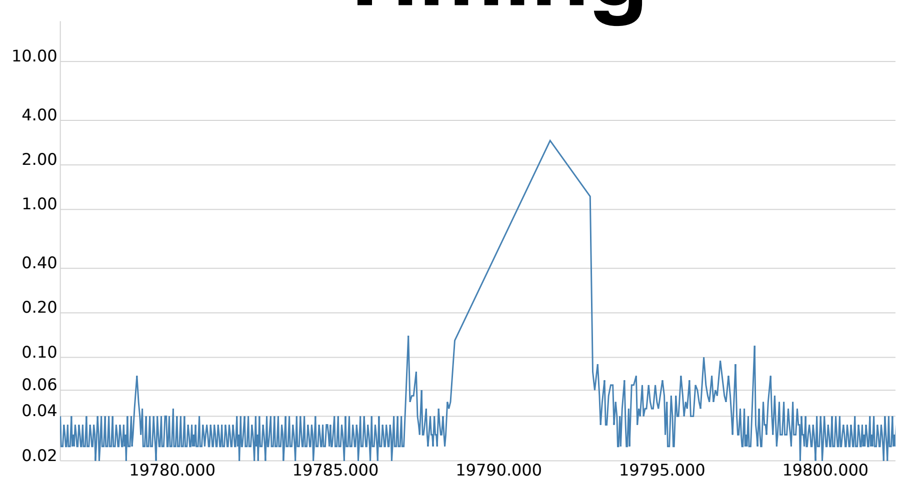
Main thread's of **Renderers**

And implement 3 different attacks:

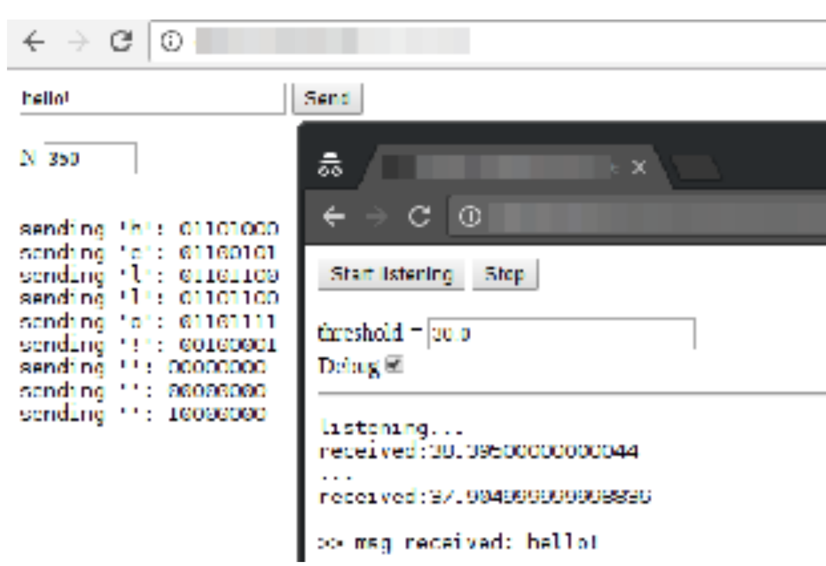
Page Identification

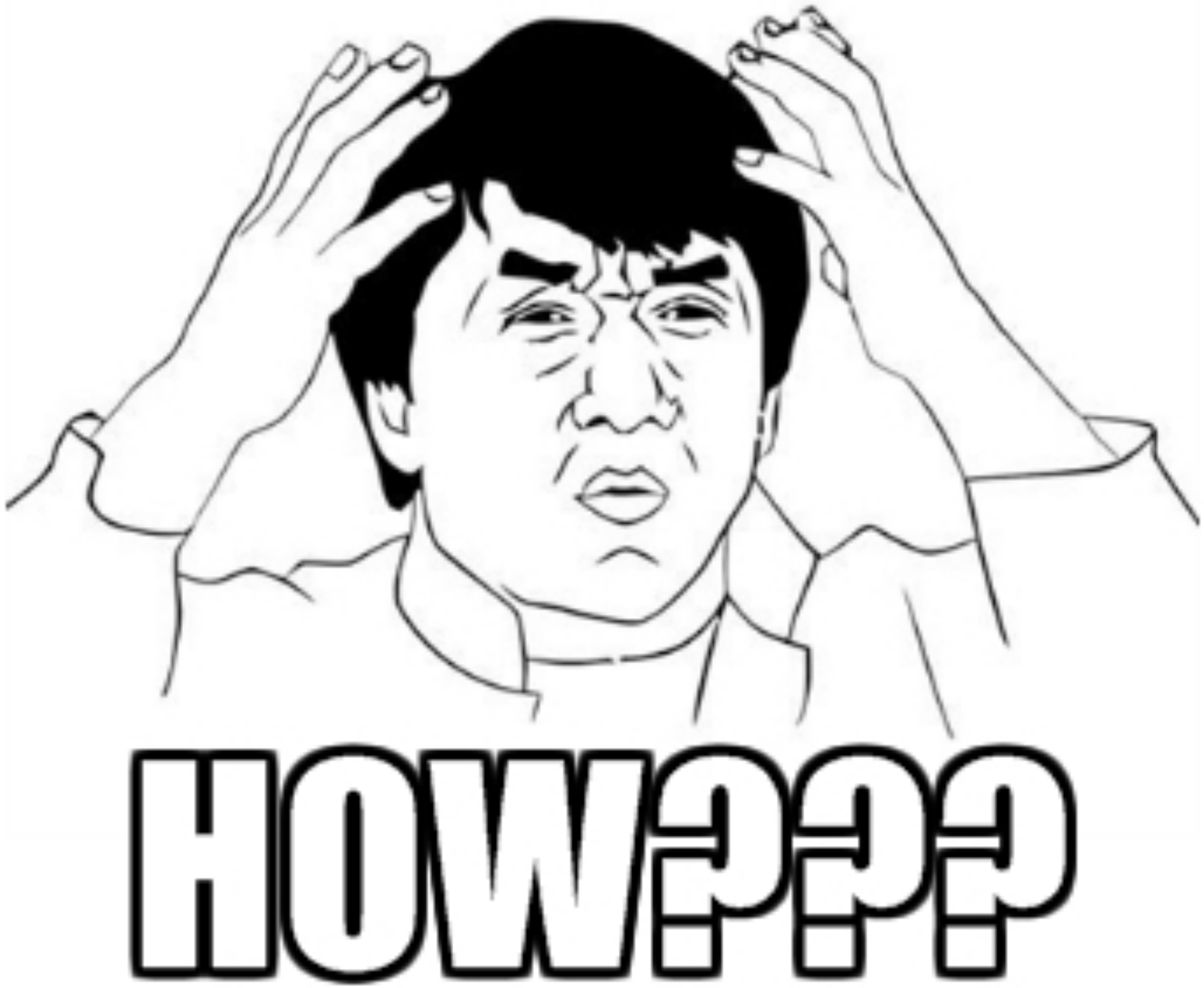


Inter-keystroke Timing



Covert Channel

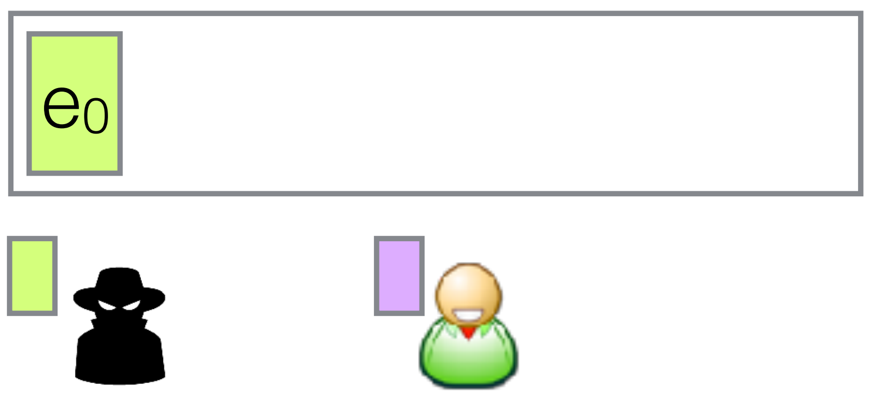



HOW???

Shared Event Loop

Shared Event Loop

FIFO queue



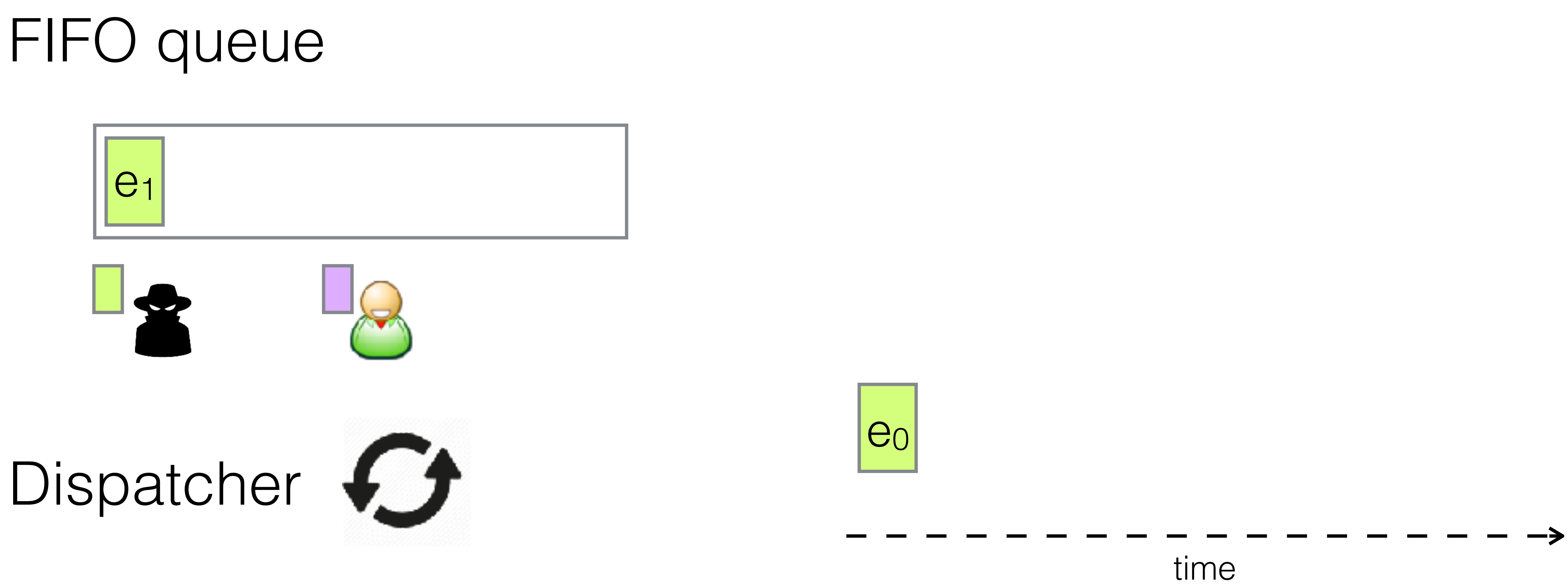
Dispatcher

time

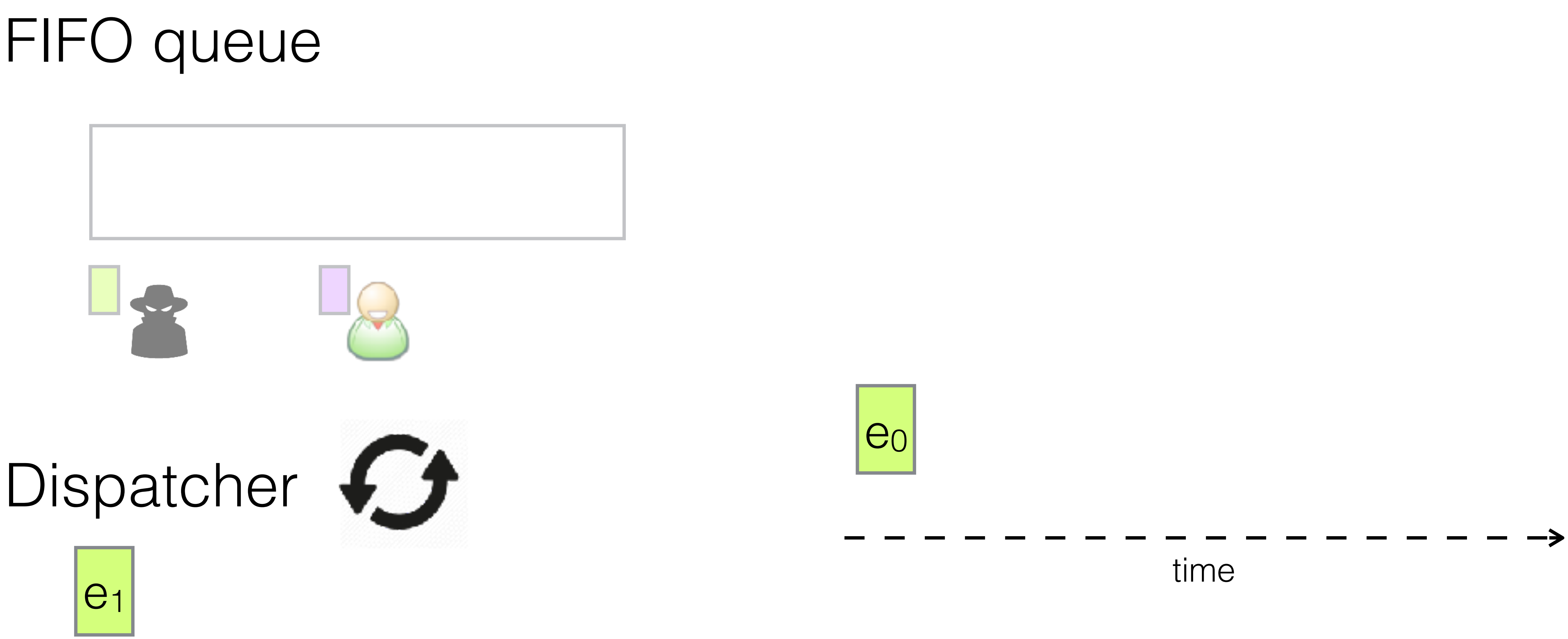
Shared Event Loop

Shared Event Loop

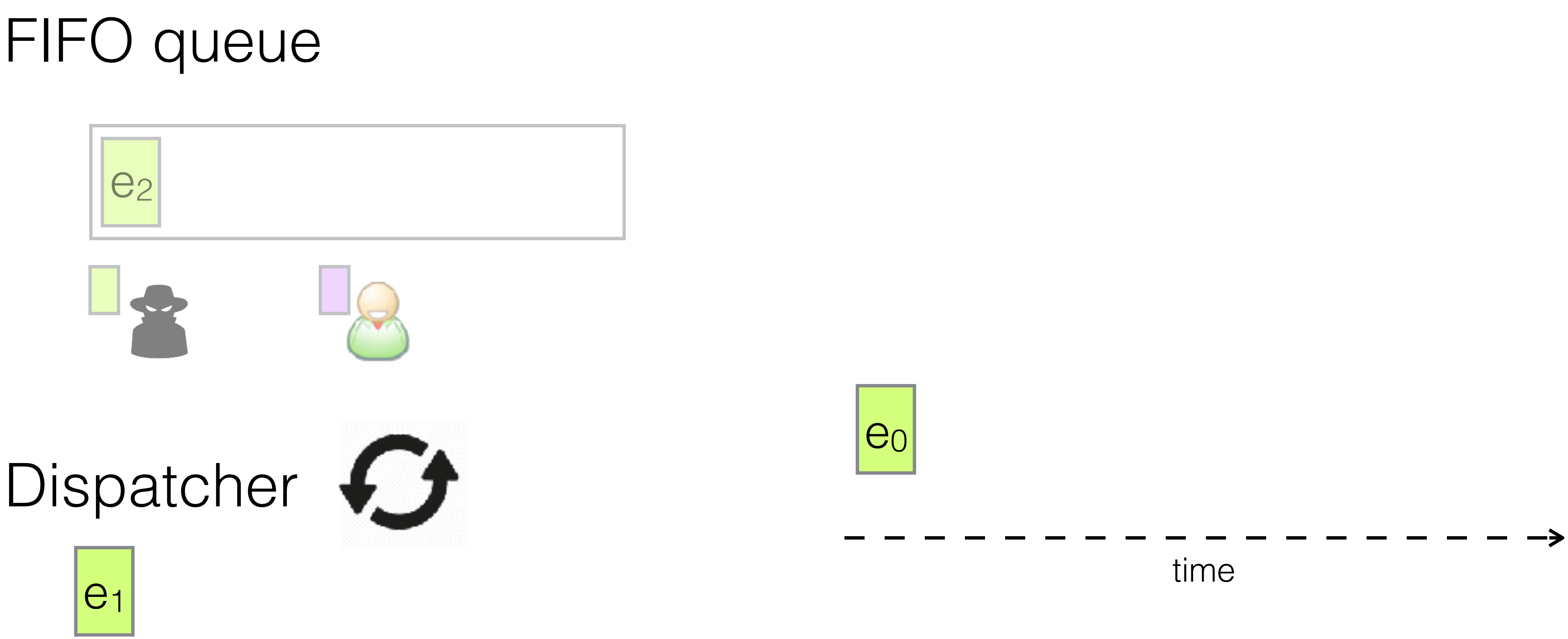
Shared Event Loop



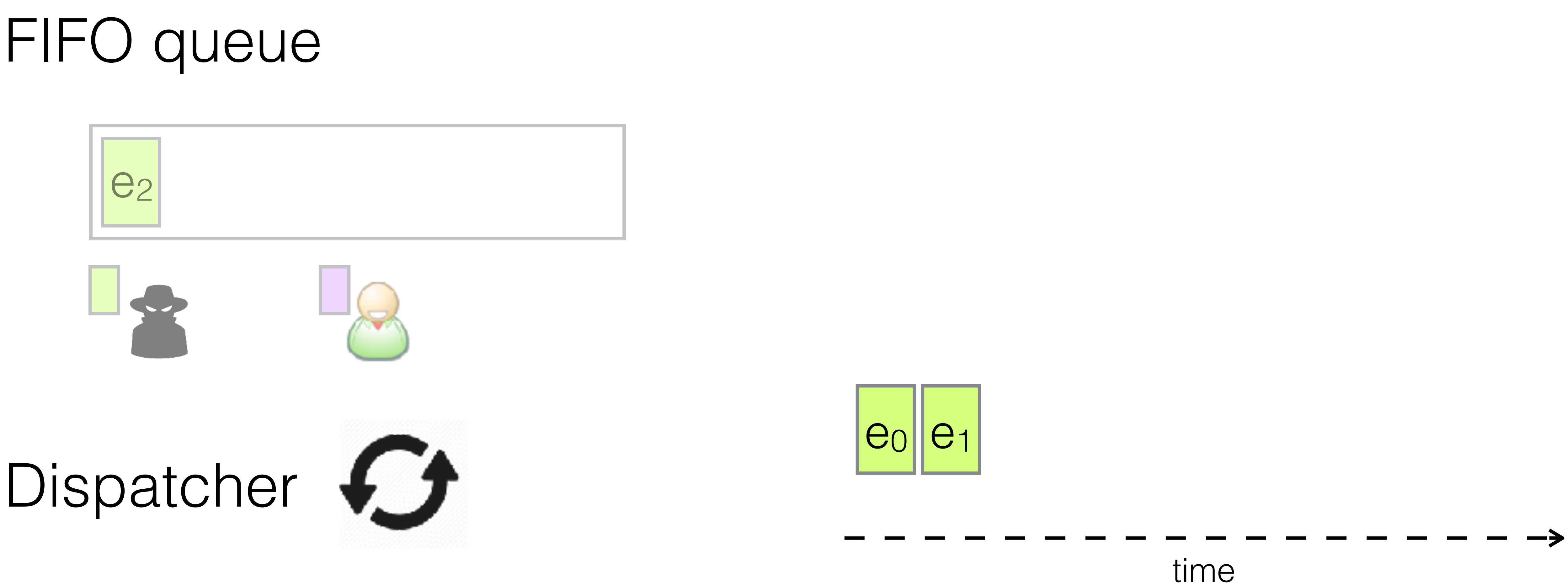
Shared Event Loop



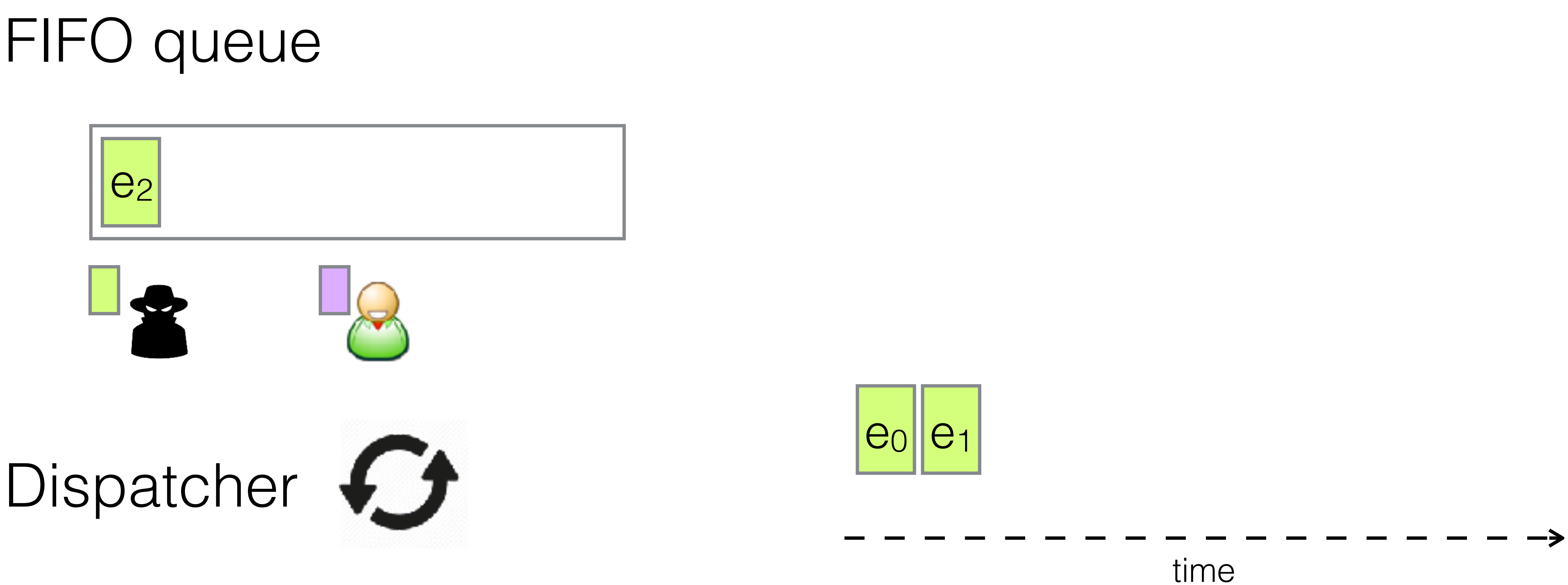
Shared Event Loop



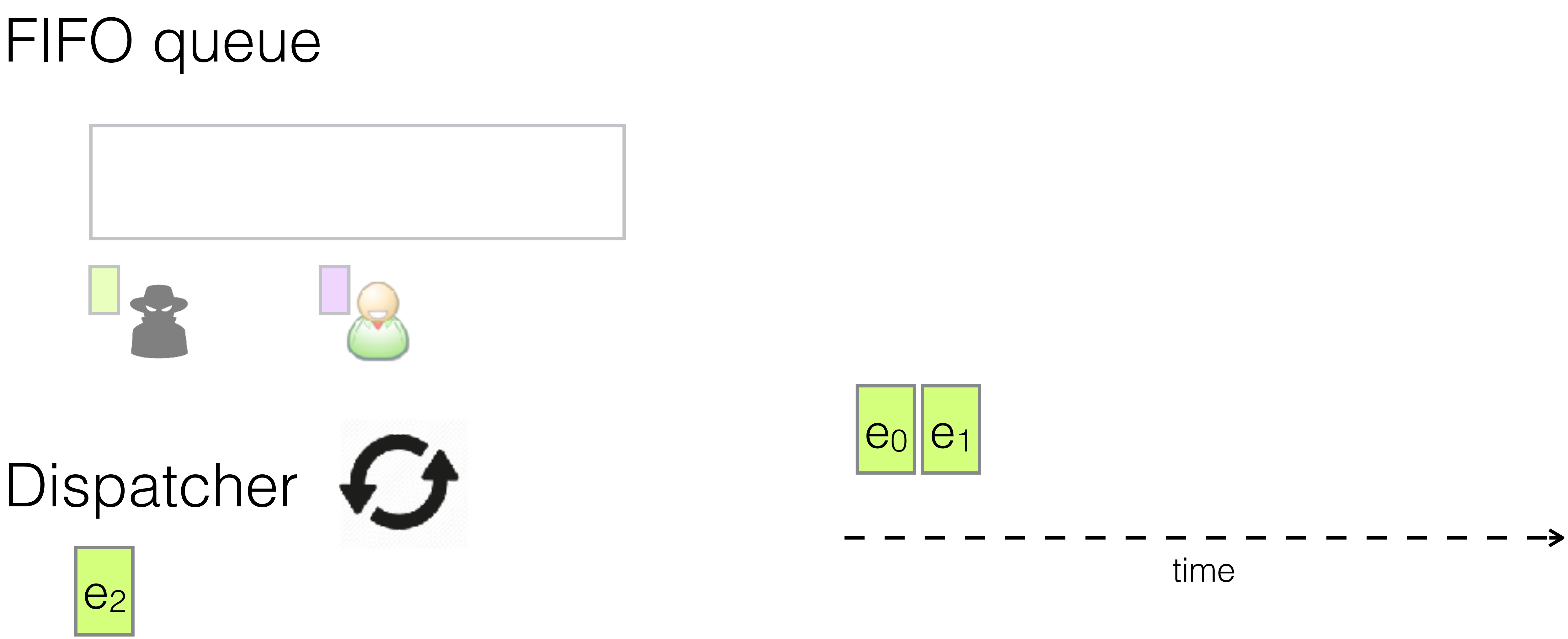
Shared Event Loop



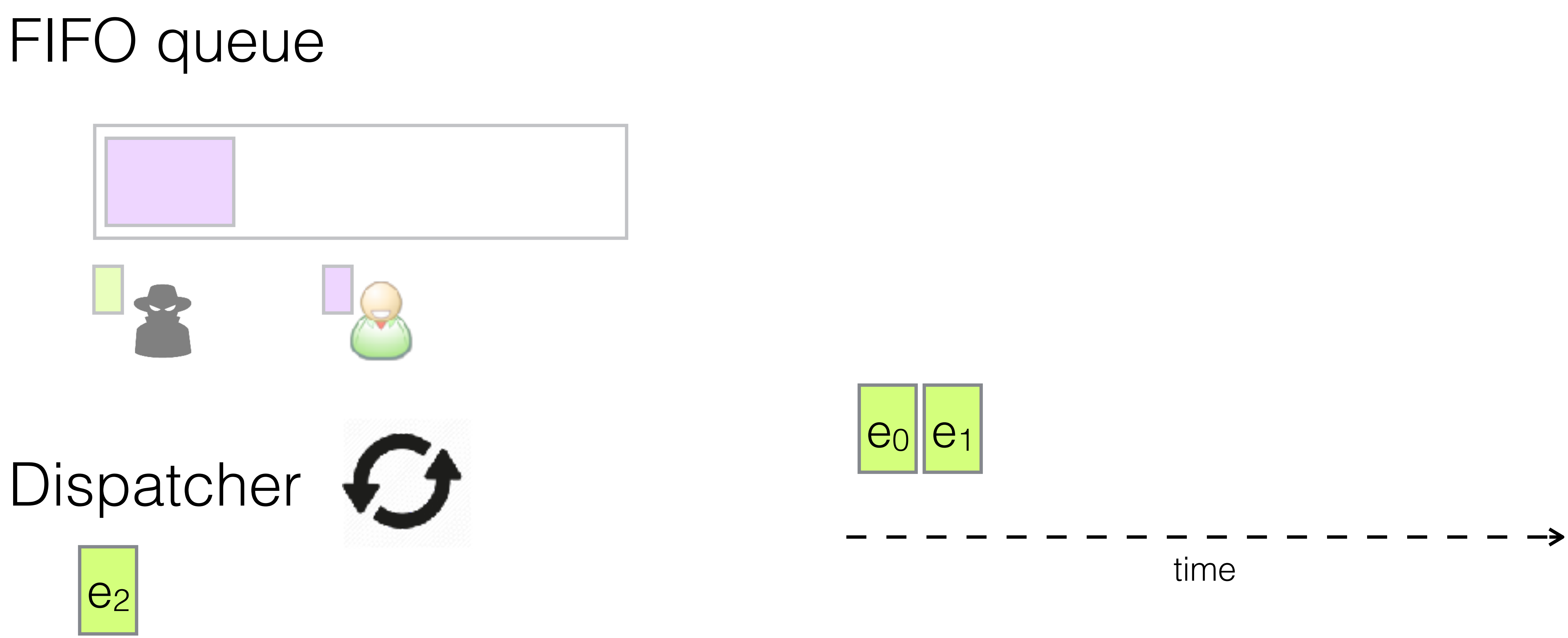
Shared Event Loop



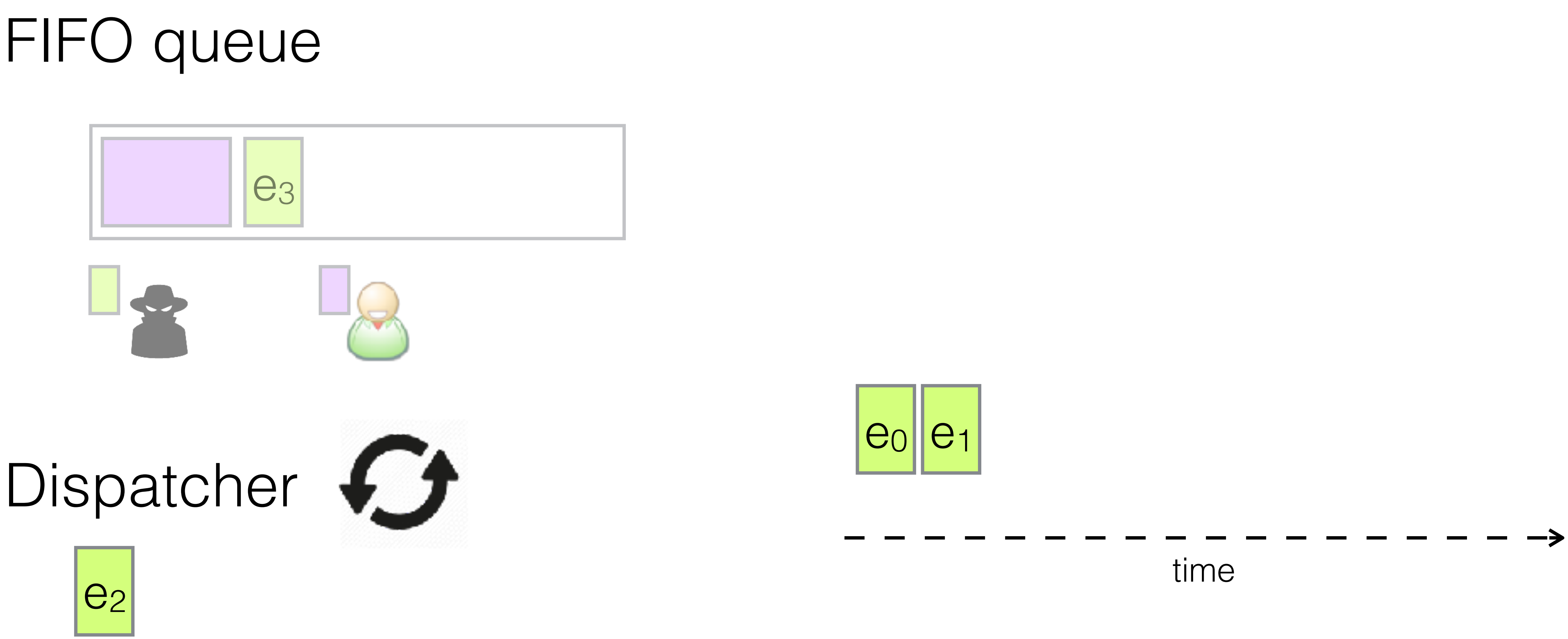
Shared Event Loop



Shared Event Loop

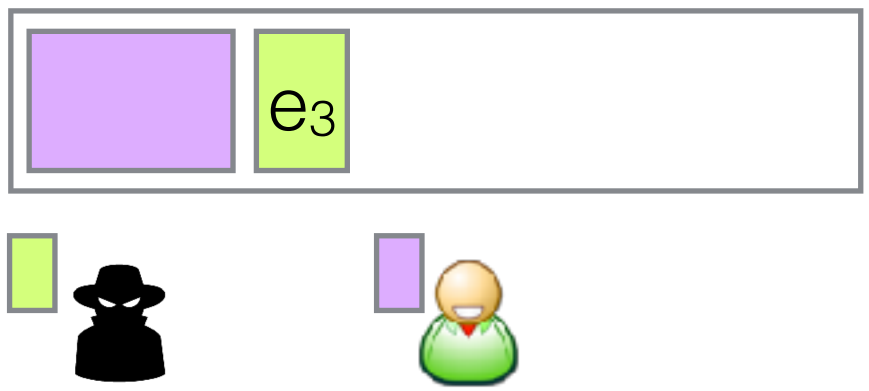


Shared Event Loop

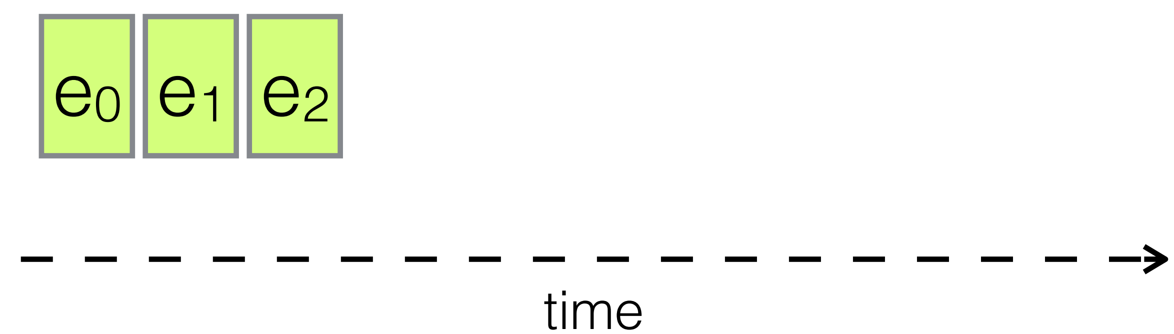


Shared Event Loop

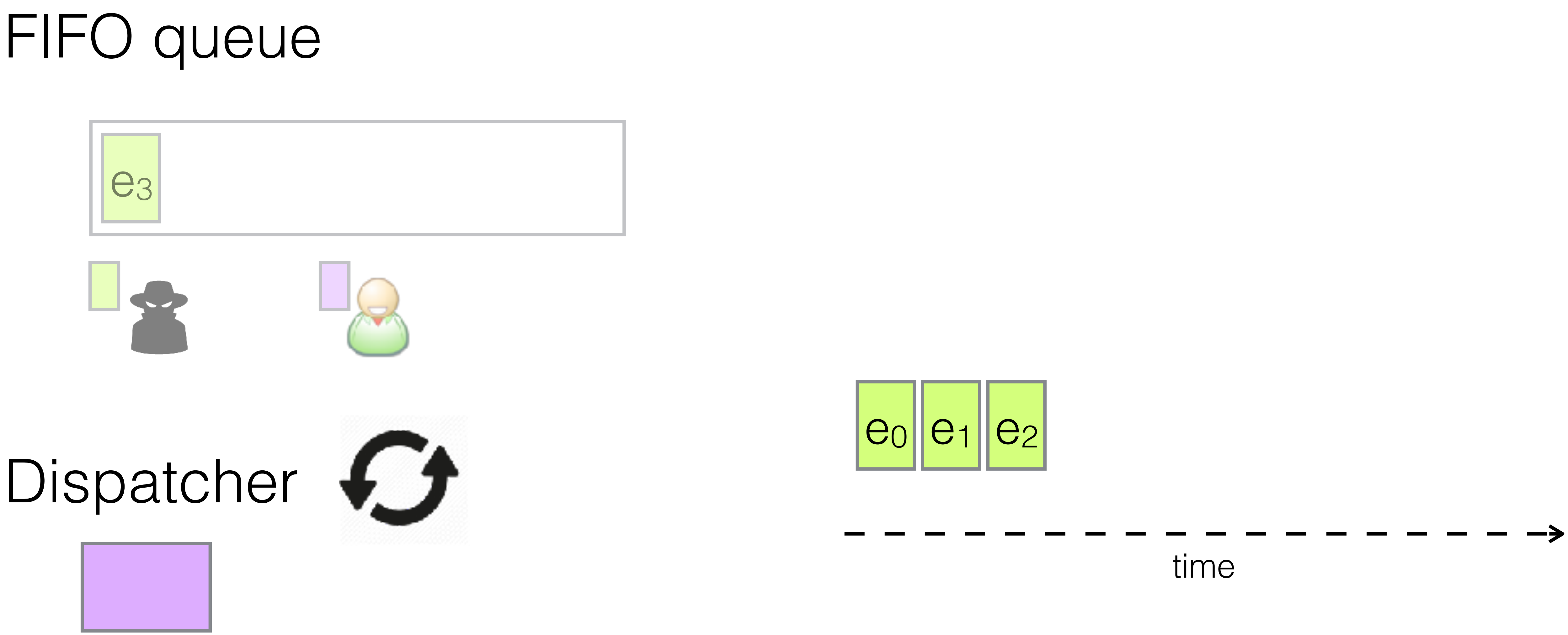
FIFO queue



Dispatcher

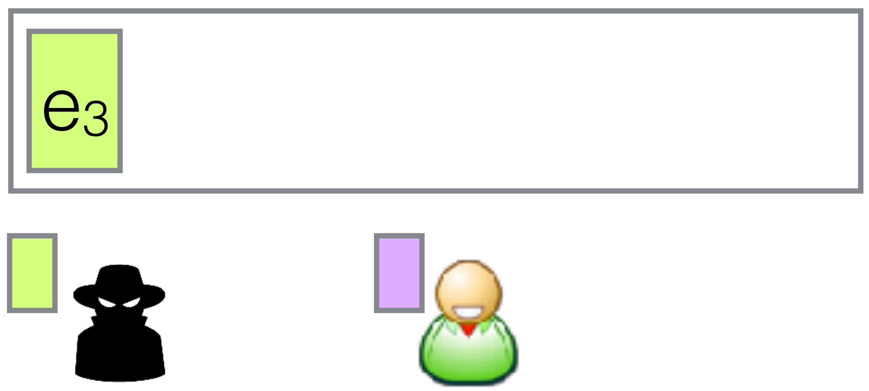


Shared Event Loop

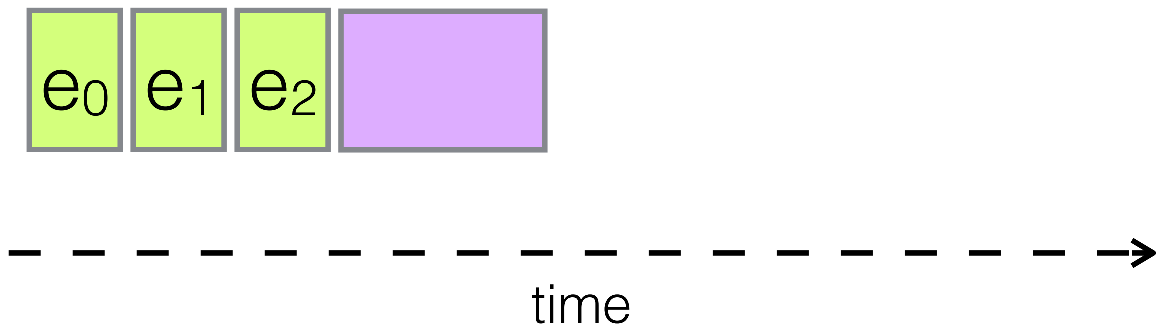


Shared Event Loop

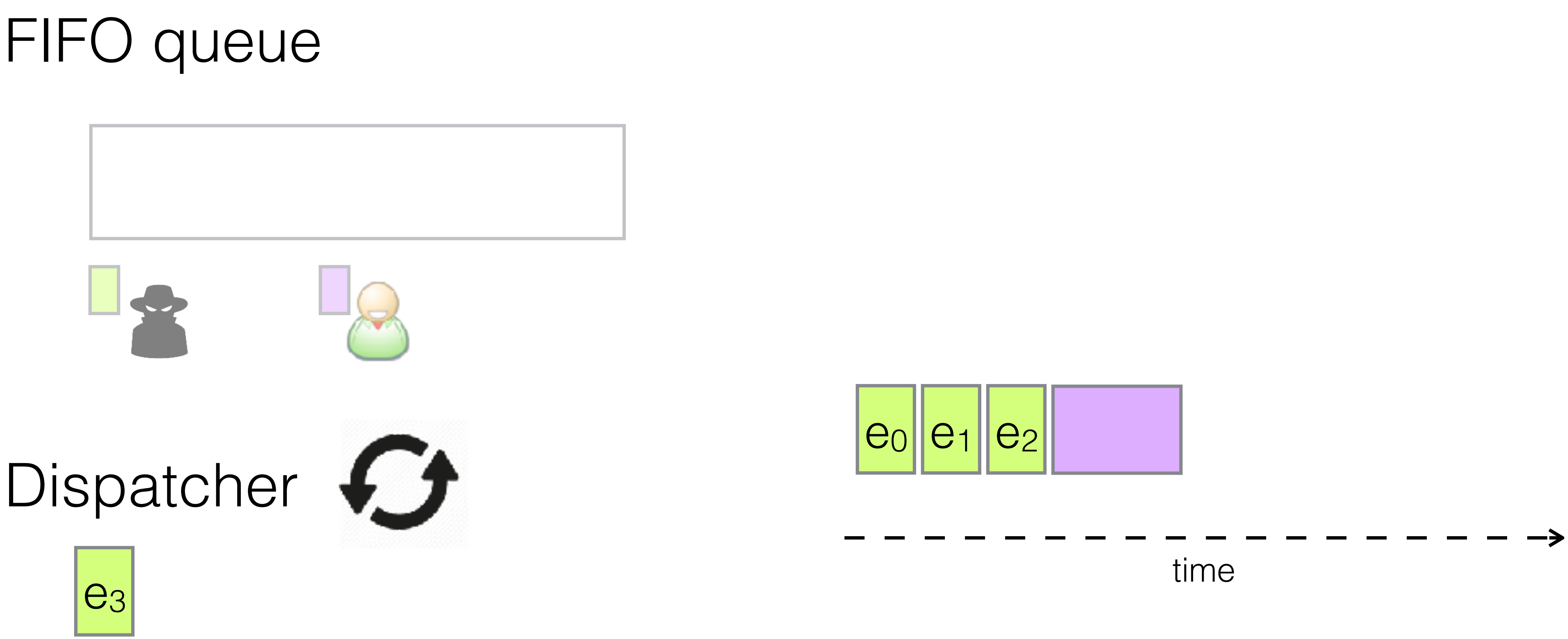
FIFO queue



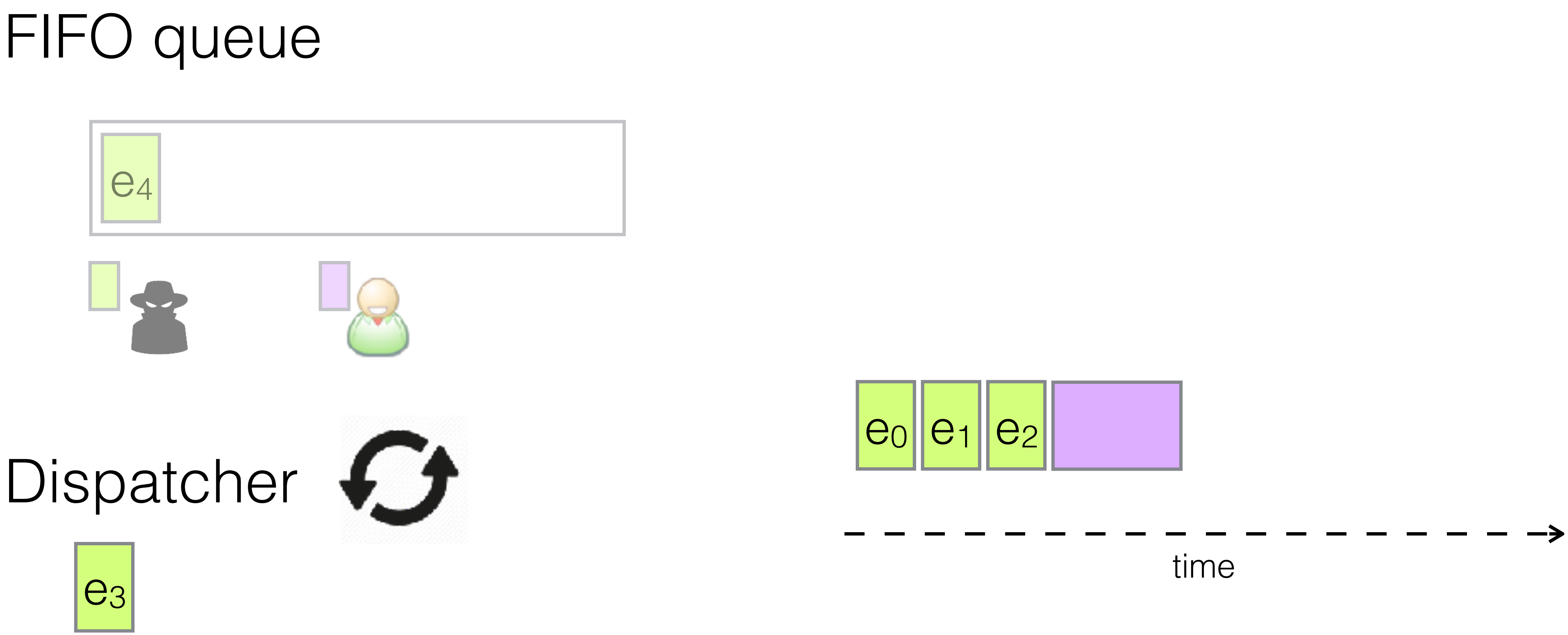
Dispatcher



Shared Event Loop

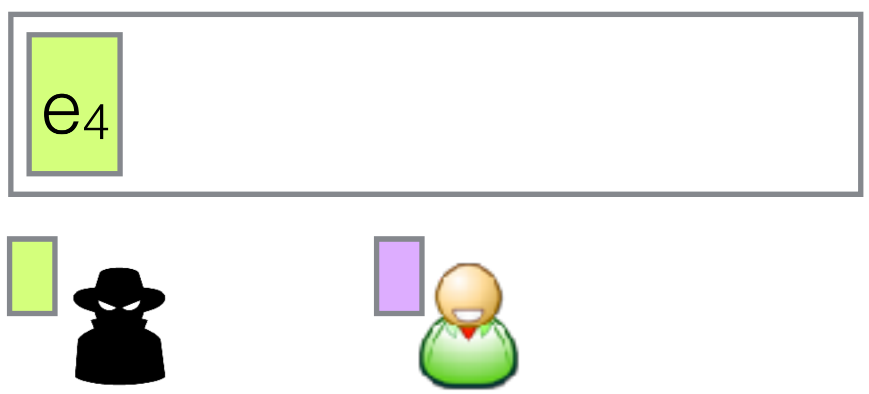


Shared Event Loop

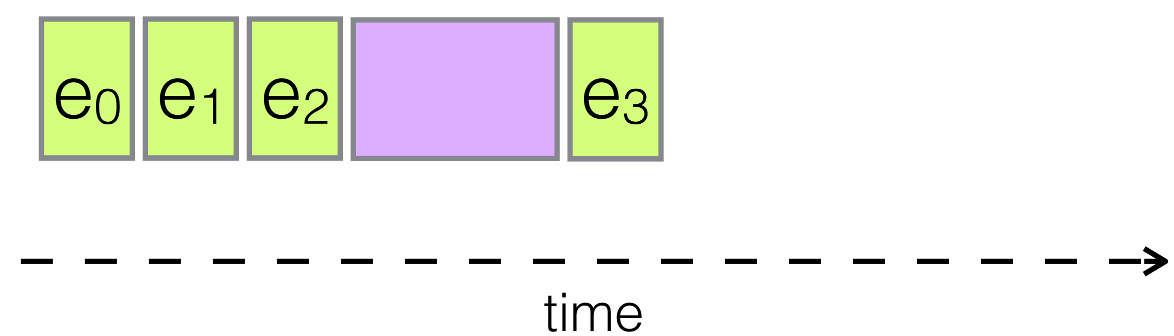


Shared Event Loop

FIFO queue

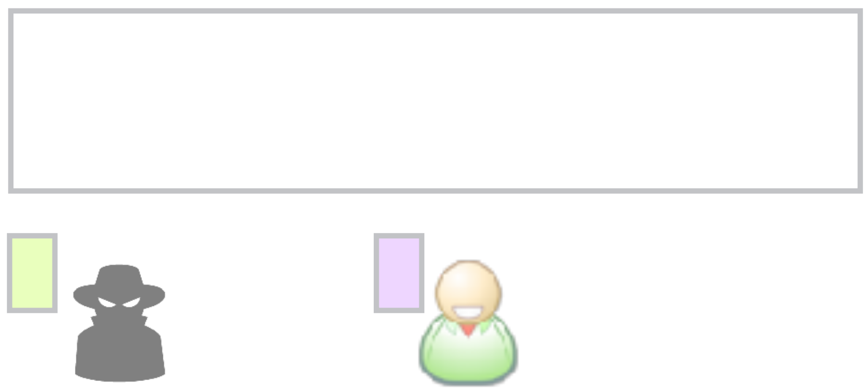


Dispatcher

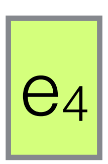


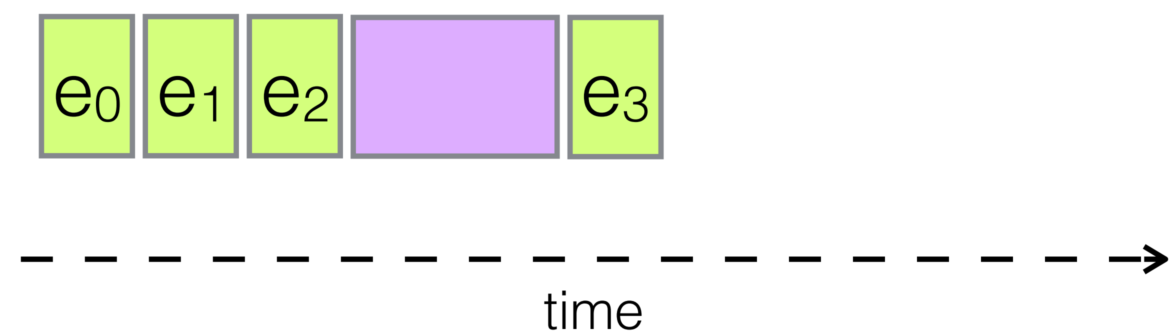
Shared Event Loop

FIFO queue



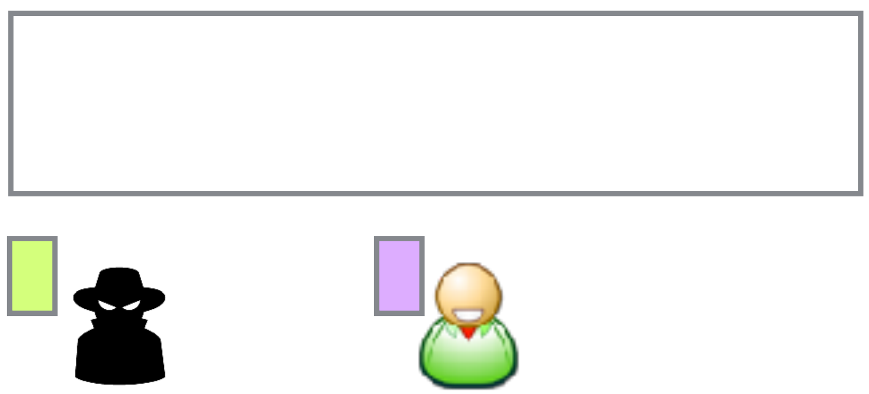
Dispatcher



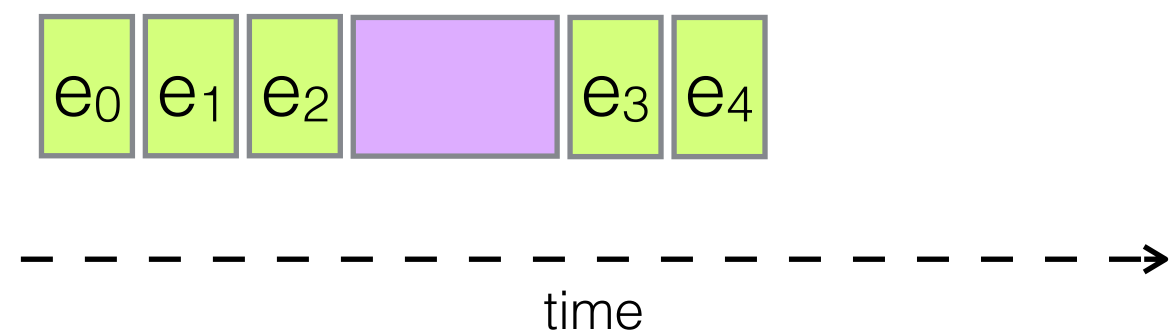


Shared Event Loop

FIFO queue

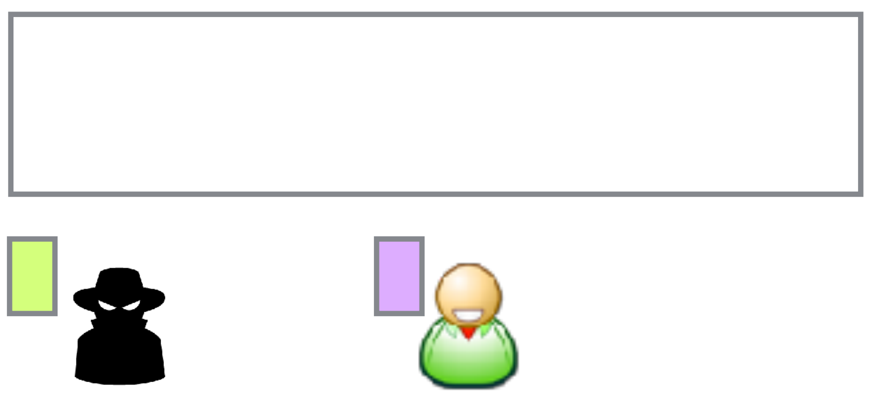


Dispatcher

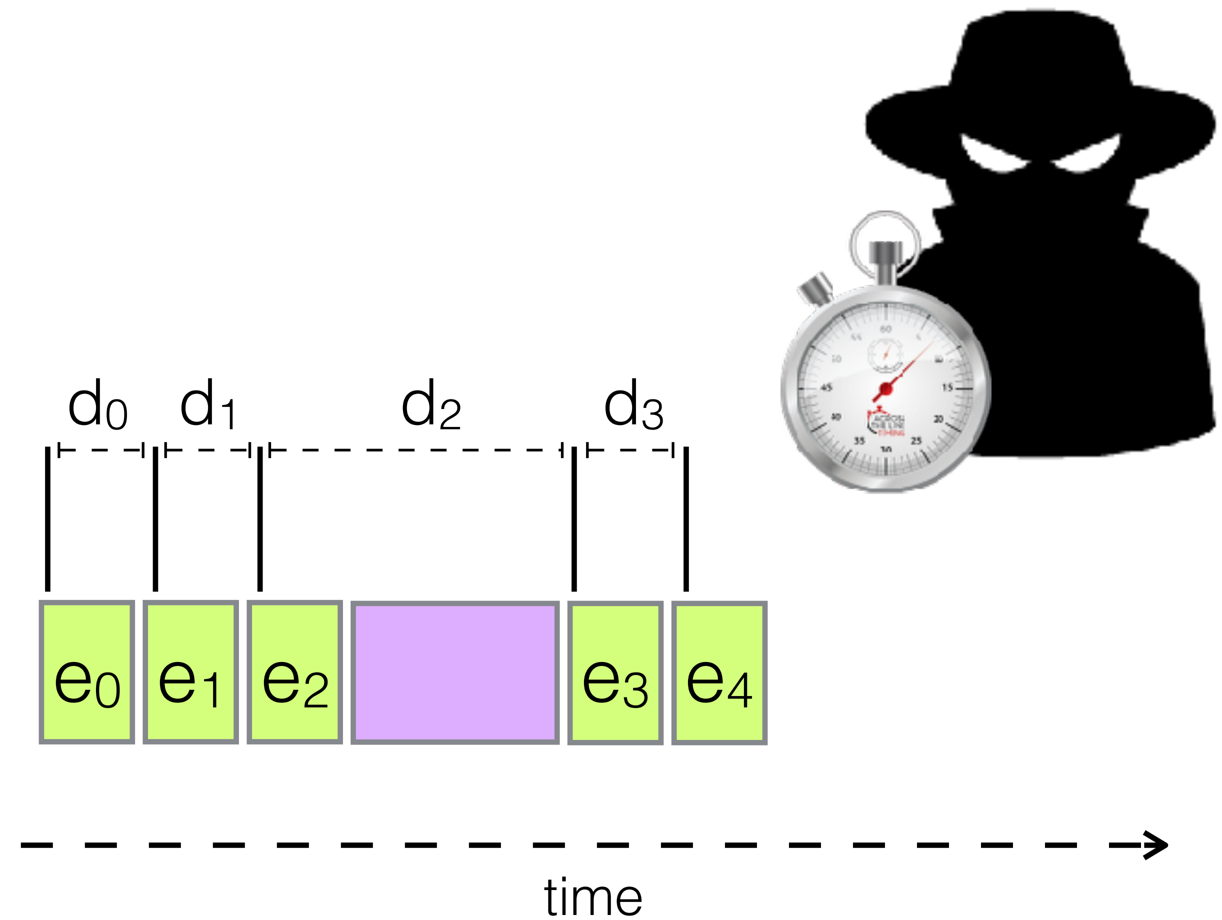


Shared Event Loop

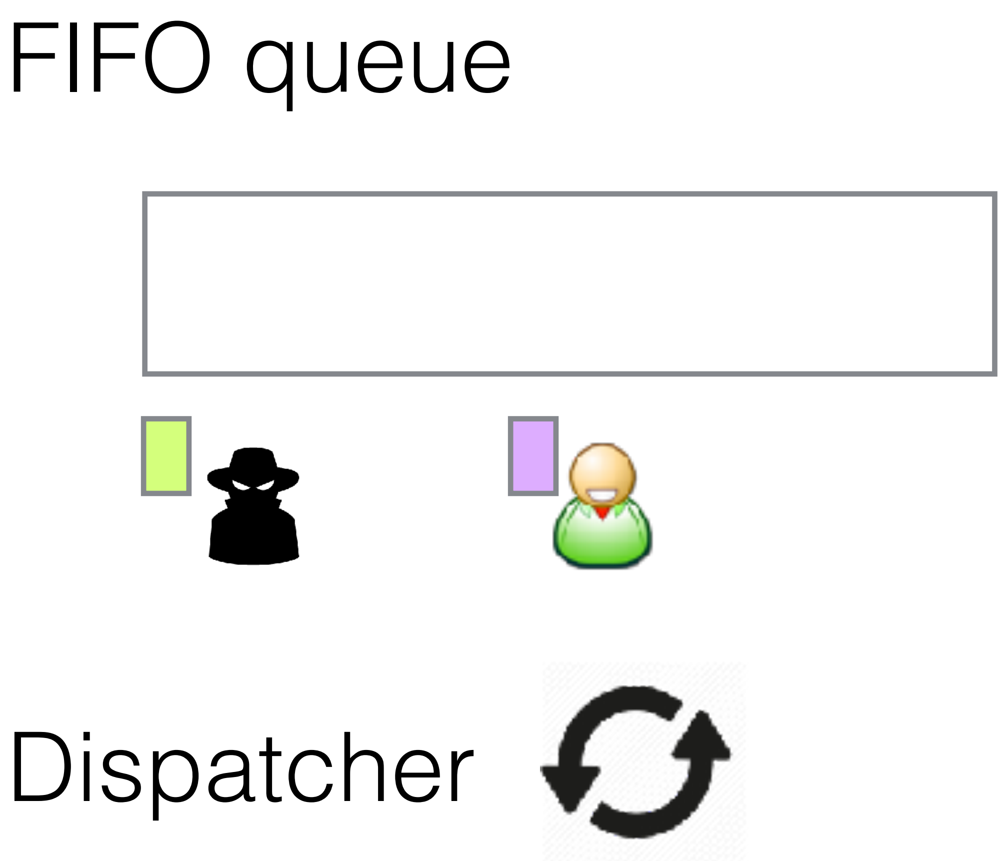
FIFO queue

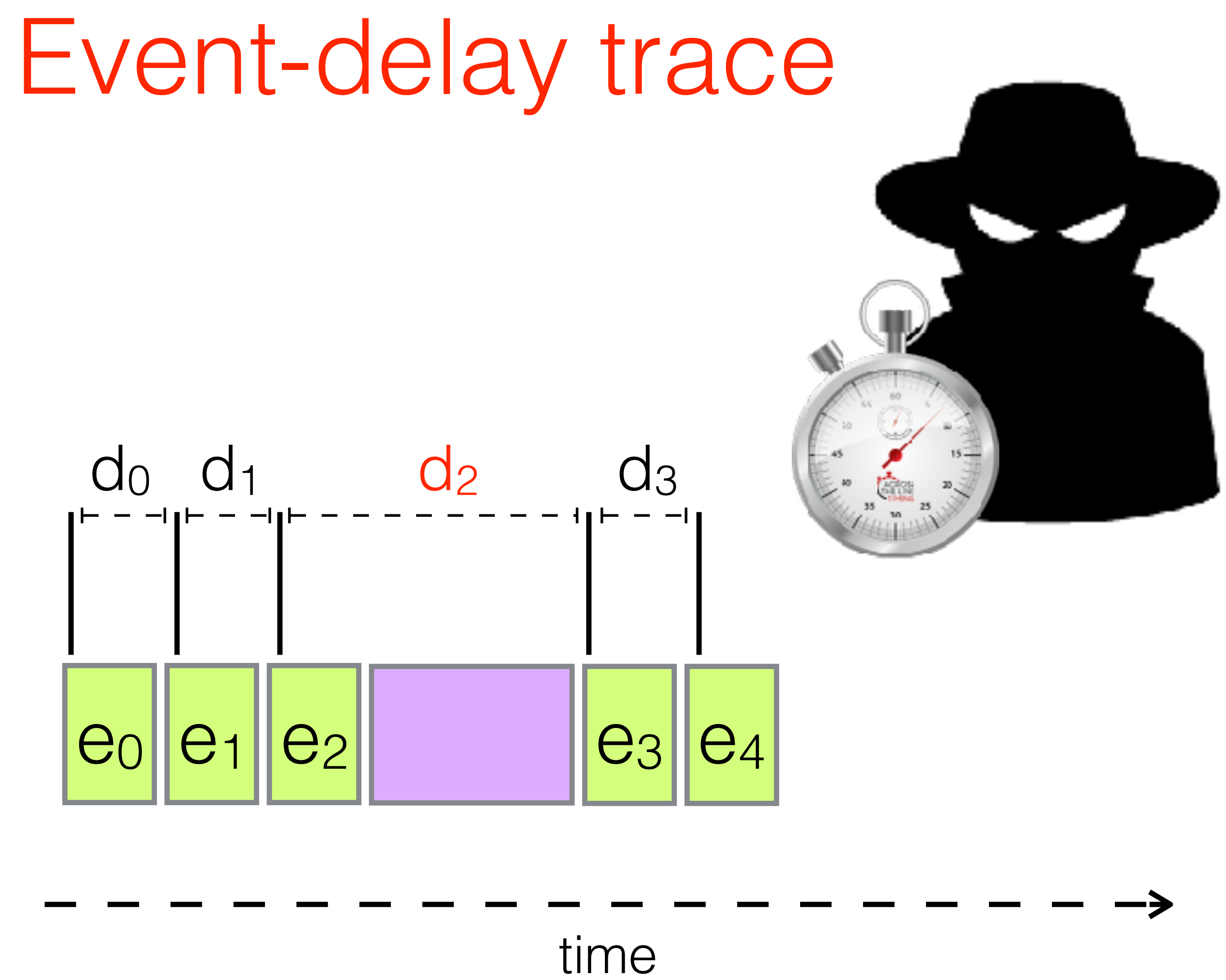


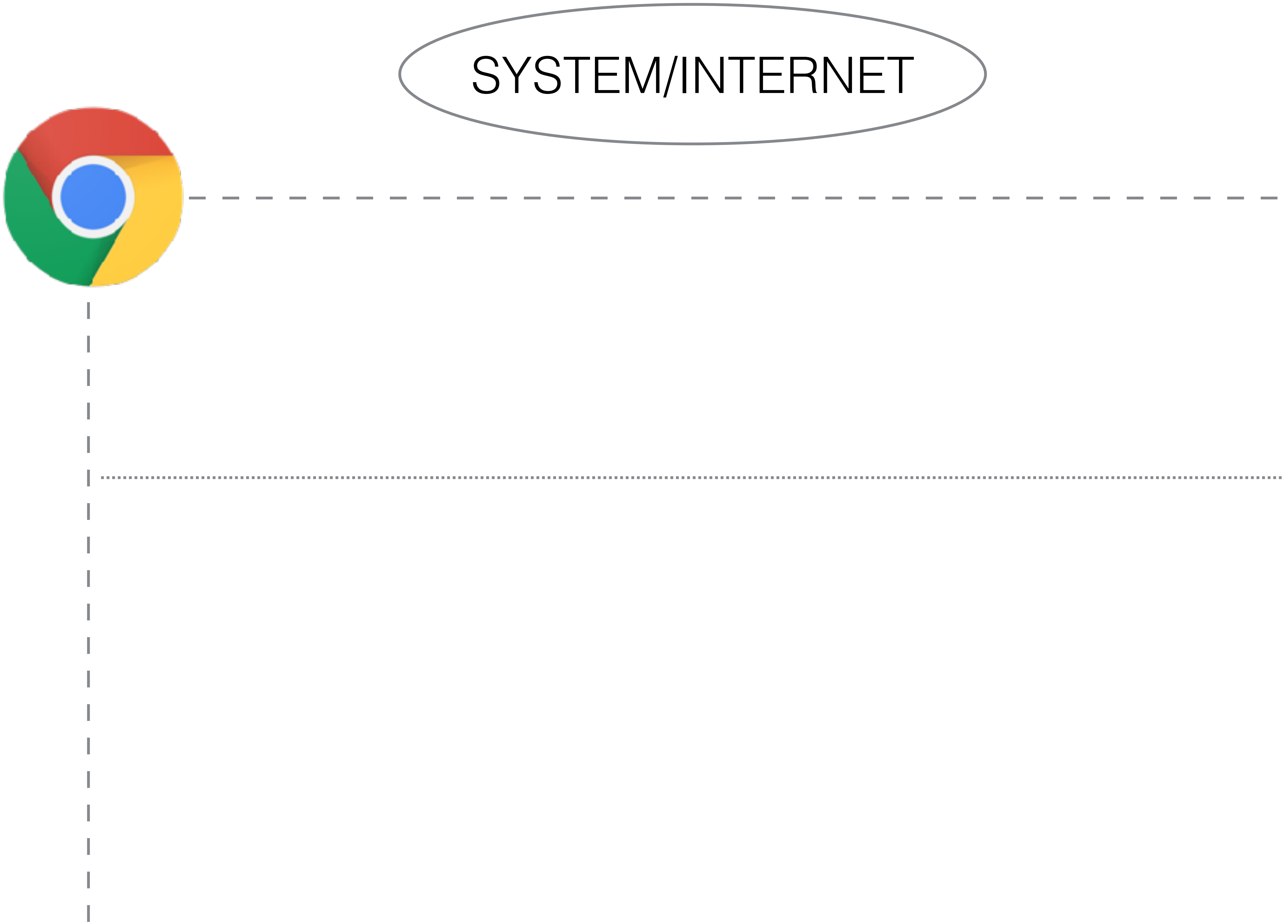
Dispatcher

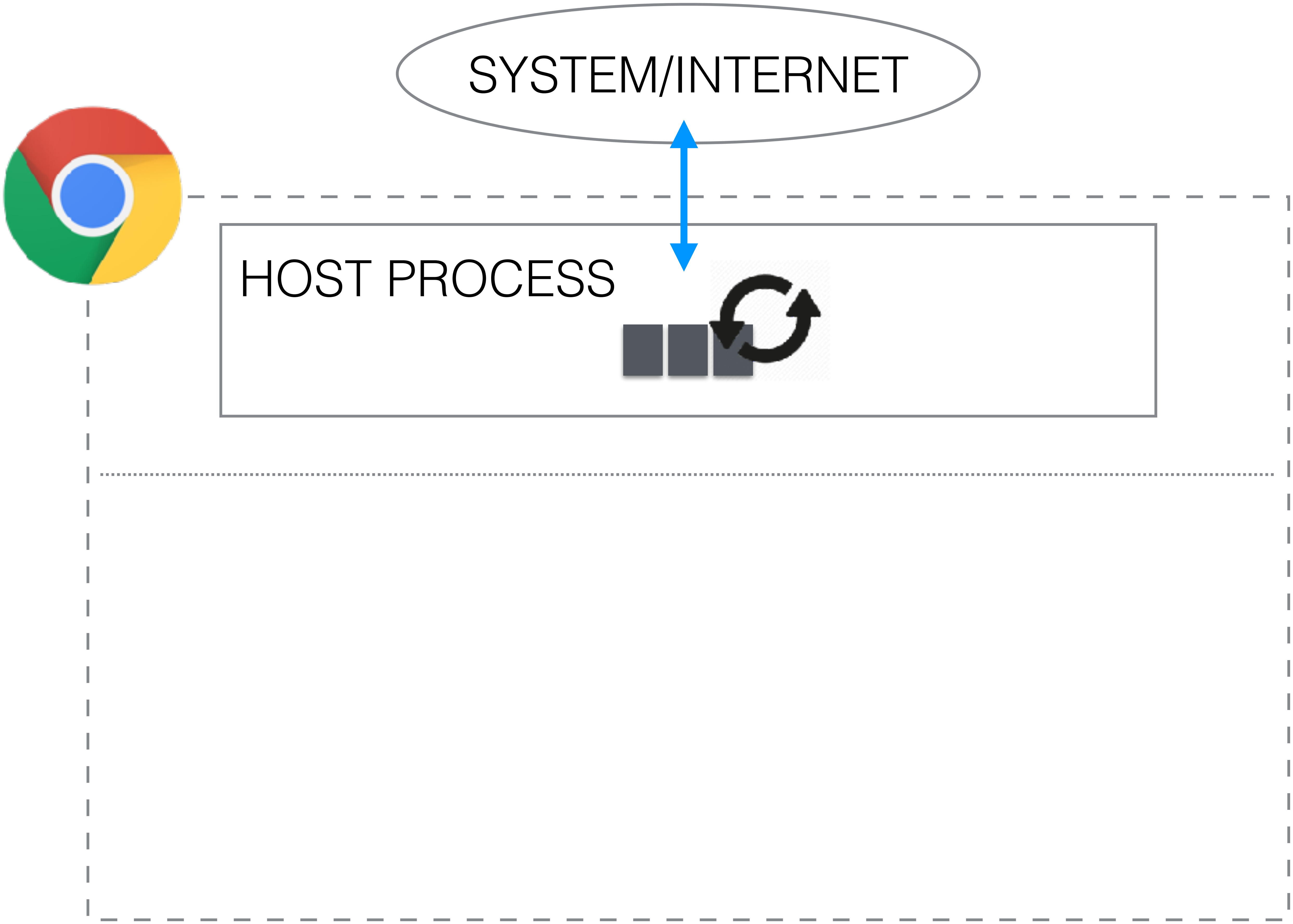


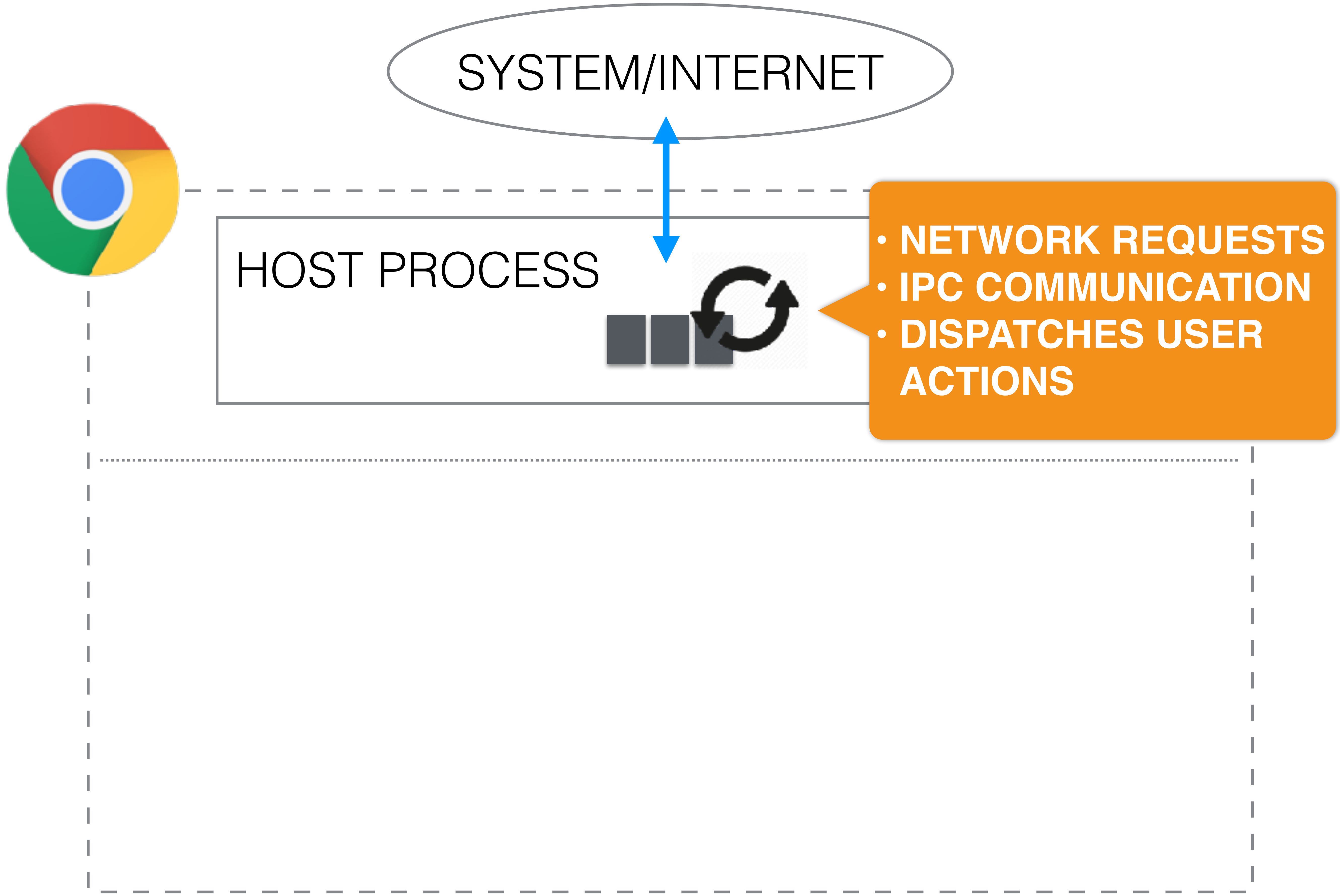
Shared Event Loop

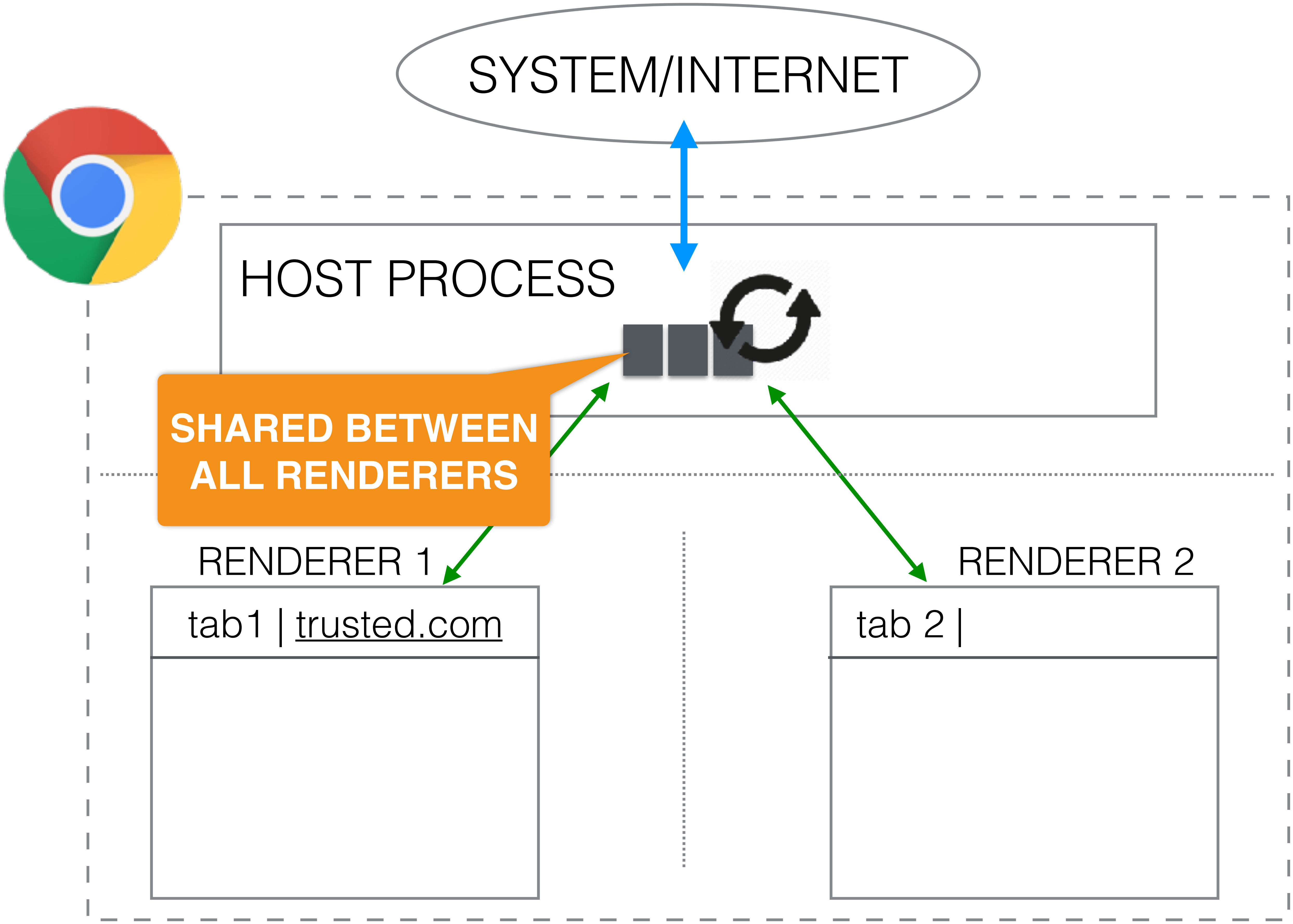


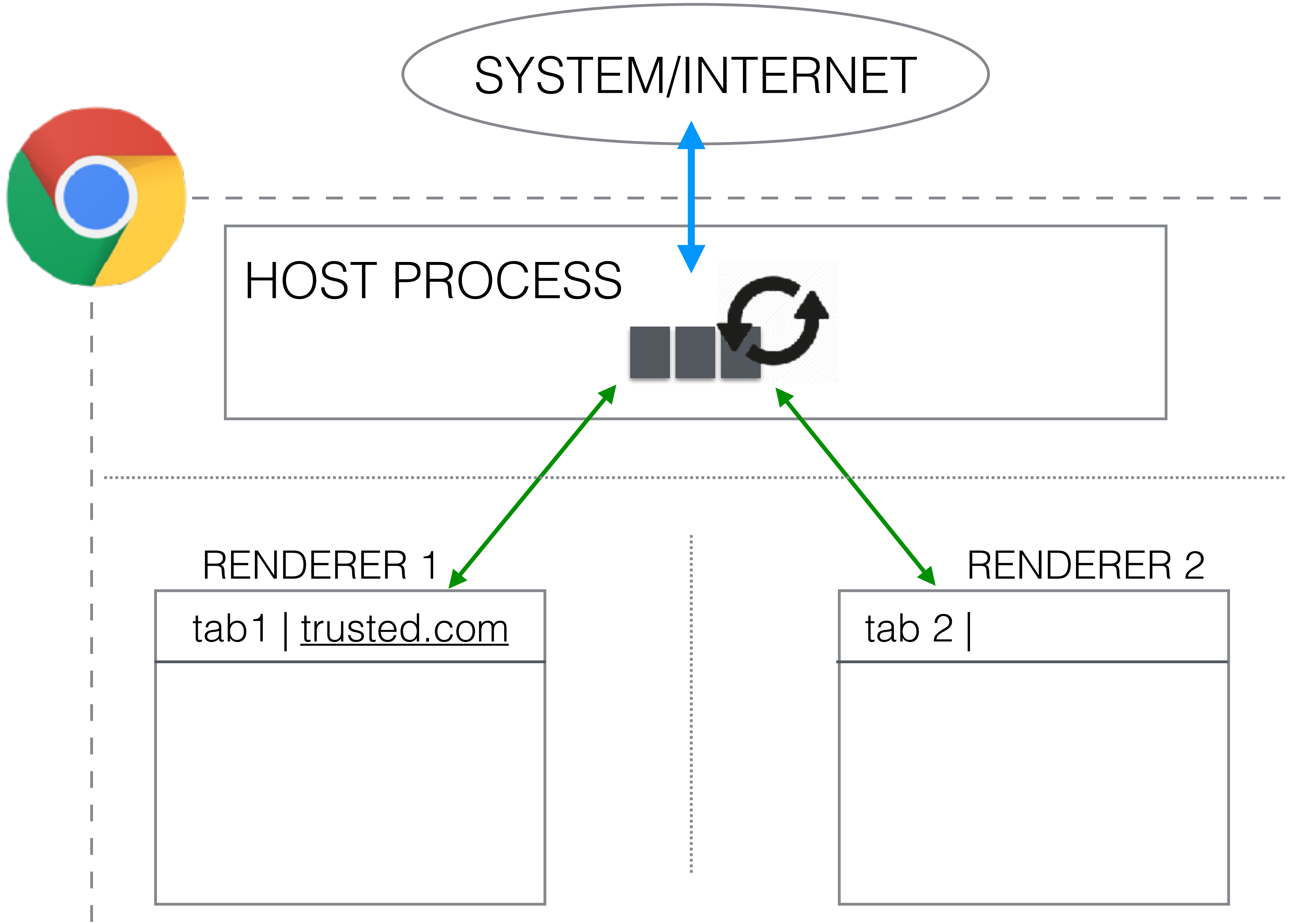



SYSTEM/INTERNET

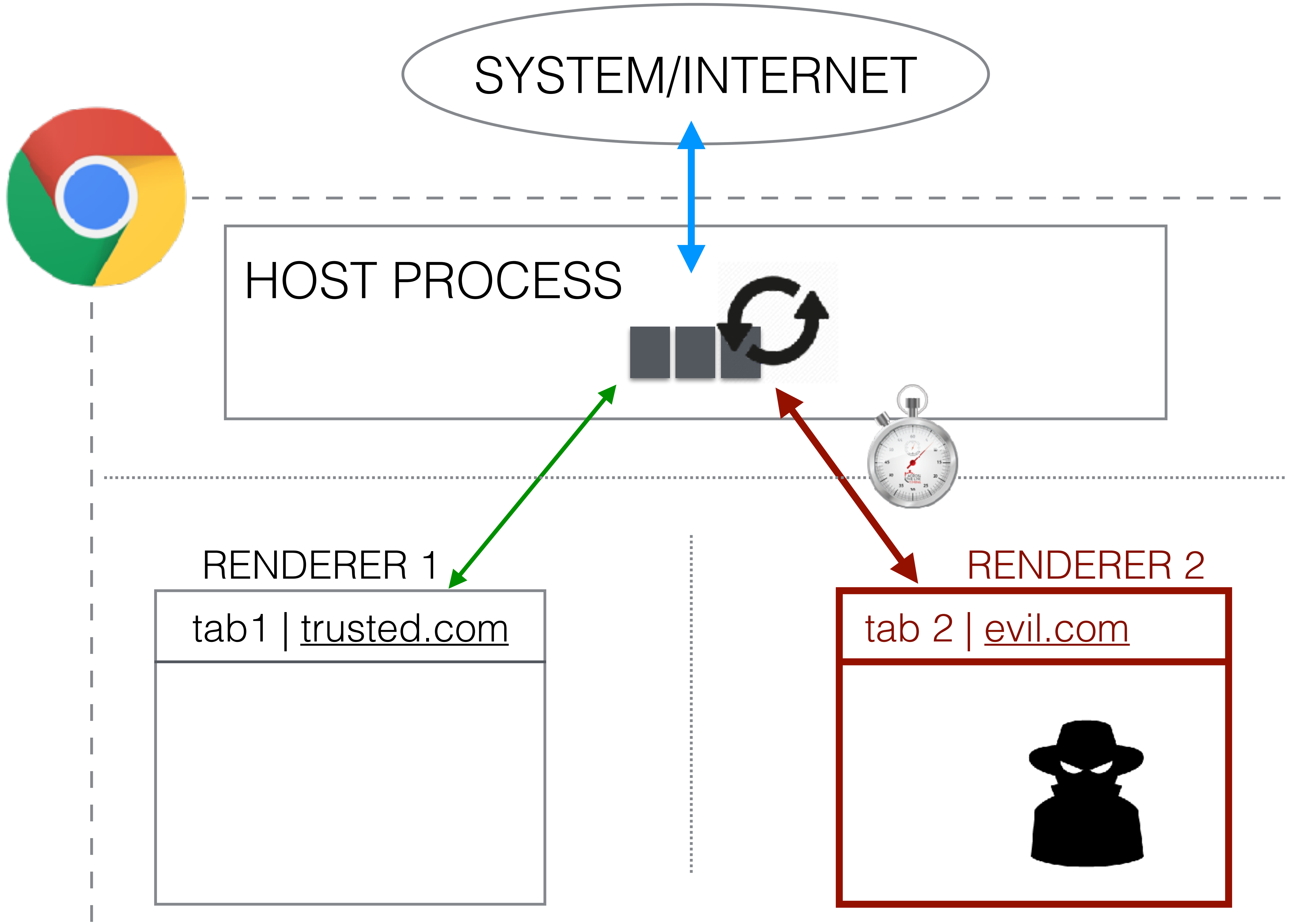

SYSTEM/INTERNET
HOST PROCESS


SYSTEM/INTERNET
HOST PROCESS
• NETWORK REQUESTS
• IPC COMMUNICATION
• DISPATCHES USER ACTIONS


SYSTEM/INTERNET
HOST PROCESS
SHARED BETWEEN ALL RENDERERS
RENDERER 1
tab1 | trusted.com
RENDERER 2
tab 2 |



SYSTEM/INTERNET
HOST PROCESS
RENDERER 1
tab1 | trusted.com
RENDERER 2
tab 2 |


SYSTEM/INTERNET
HOST PROCESS
RENDERER 1
tab1 | trusted.com
RENDERER 2
tab 2 | evil.com

Spying on the Host

```
<script>
function loop () {
    save(performance.now());
    fetch(new Request("http://0/"))
       .catch(loop);
}
loop();
</script>
```

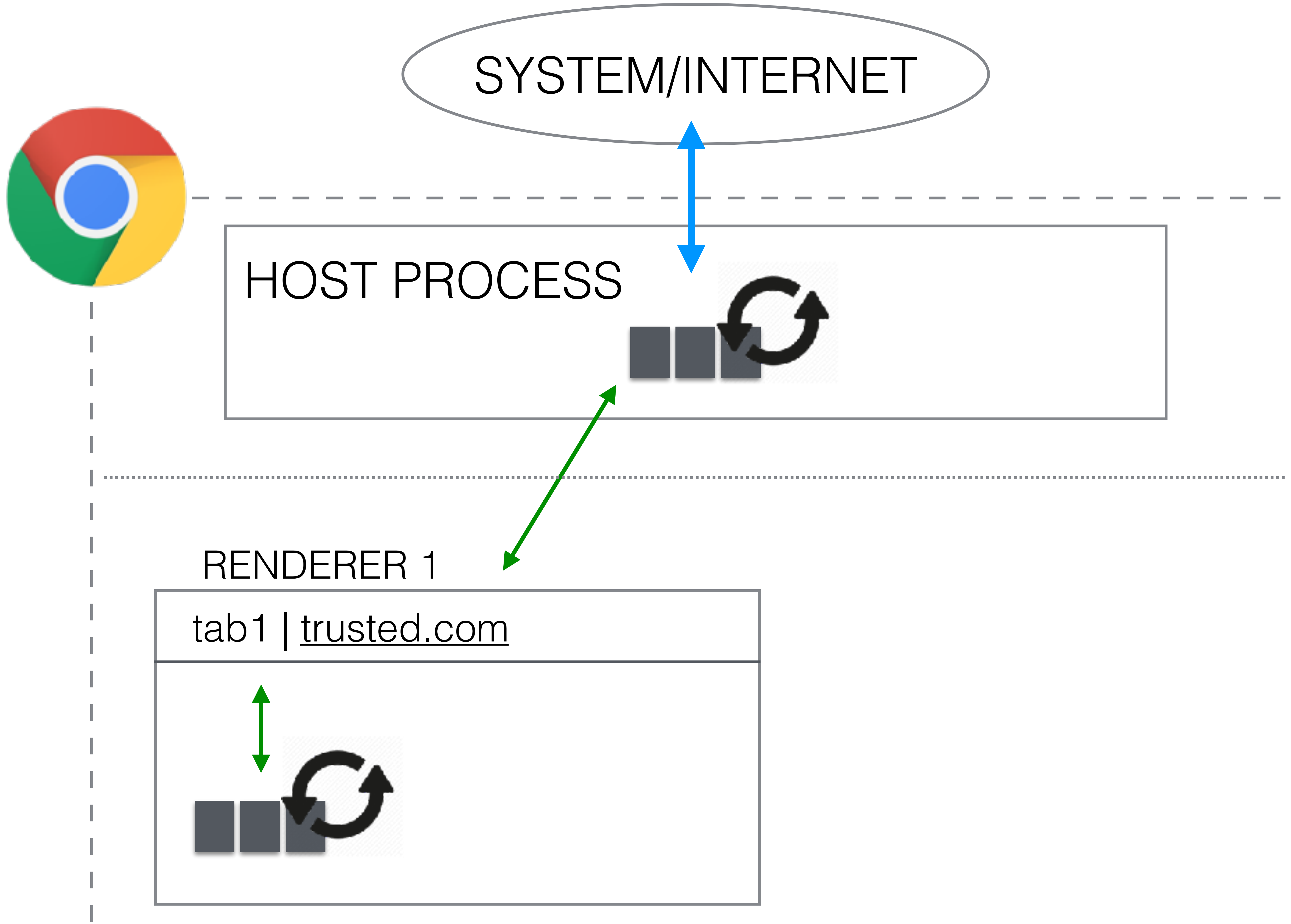
Timing resolution of ~500 μs

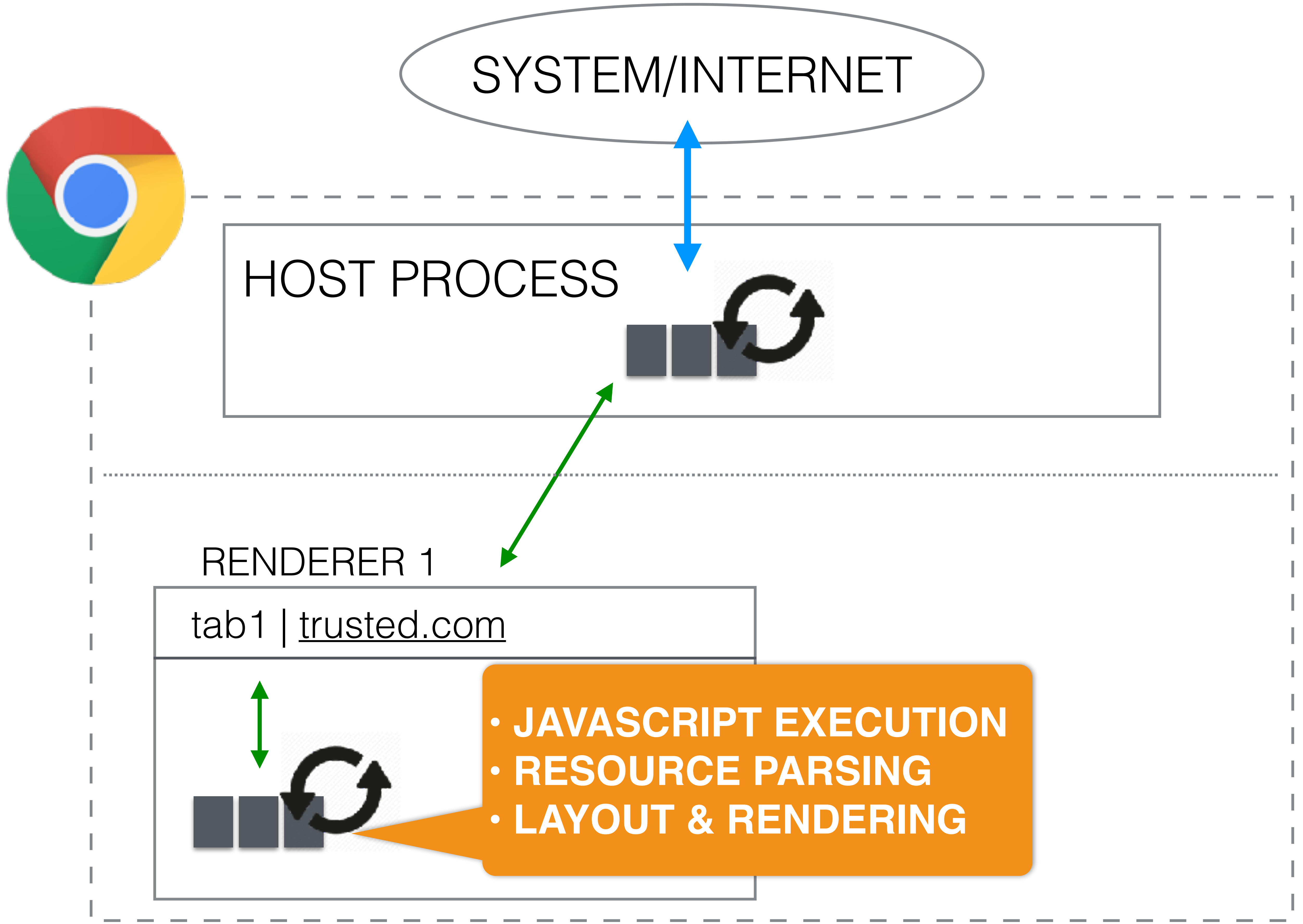
Spying on the Host

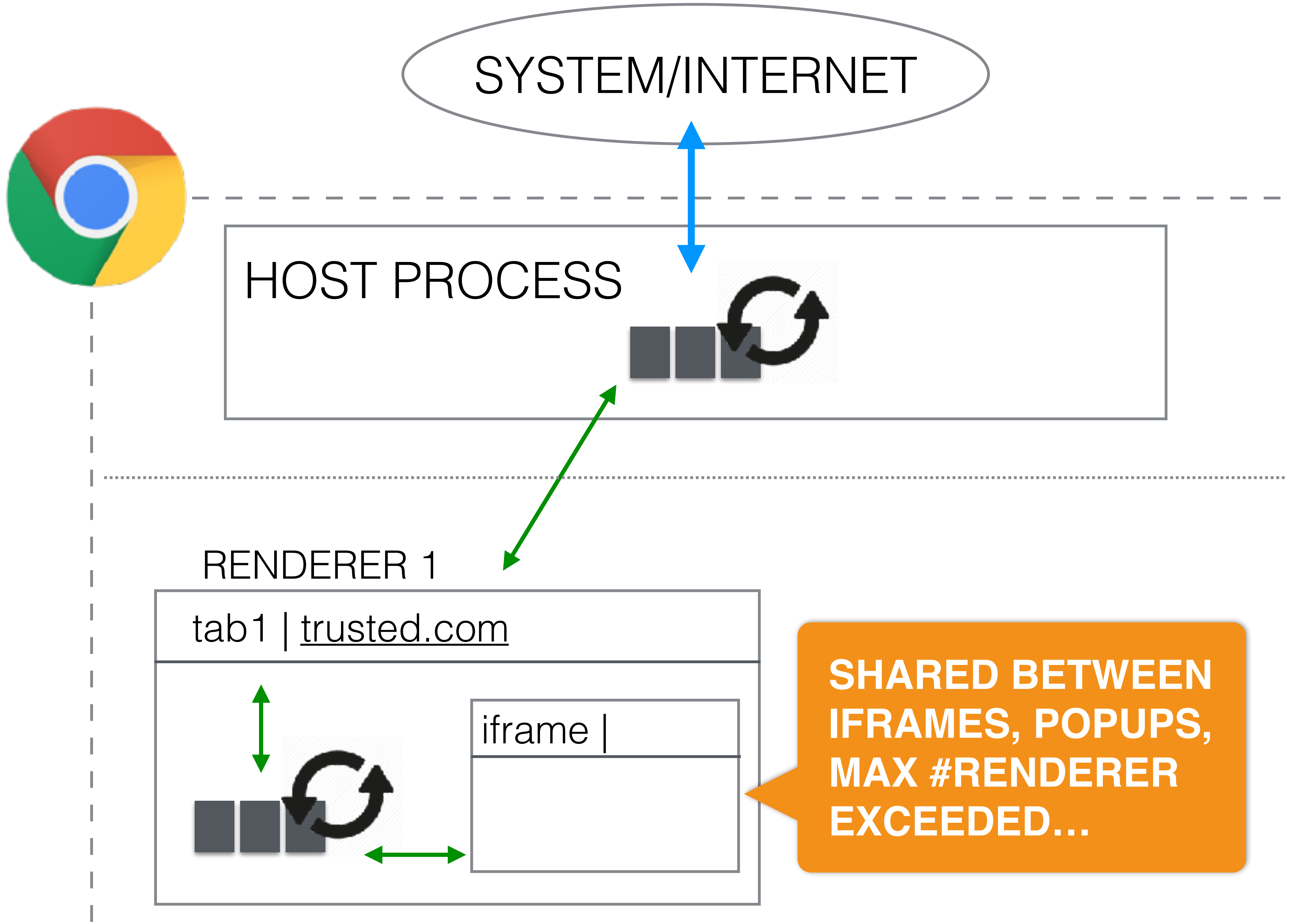
```
<script>
function loop () {
    save(performance.now());
    fetch(new Request("http://0/"))
        .catch(loop);
}
loop();
</script>
```

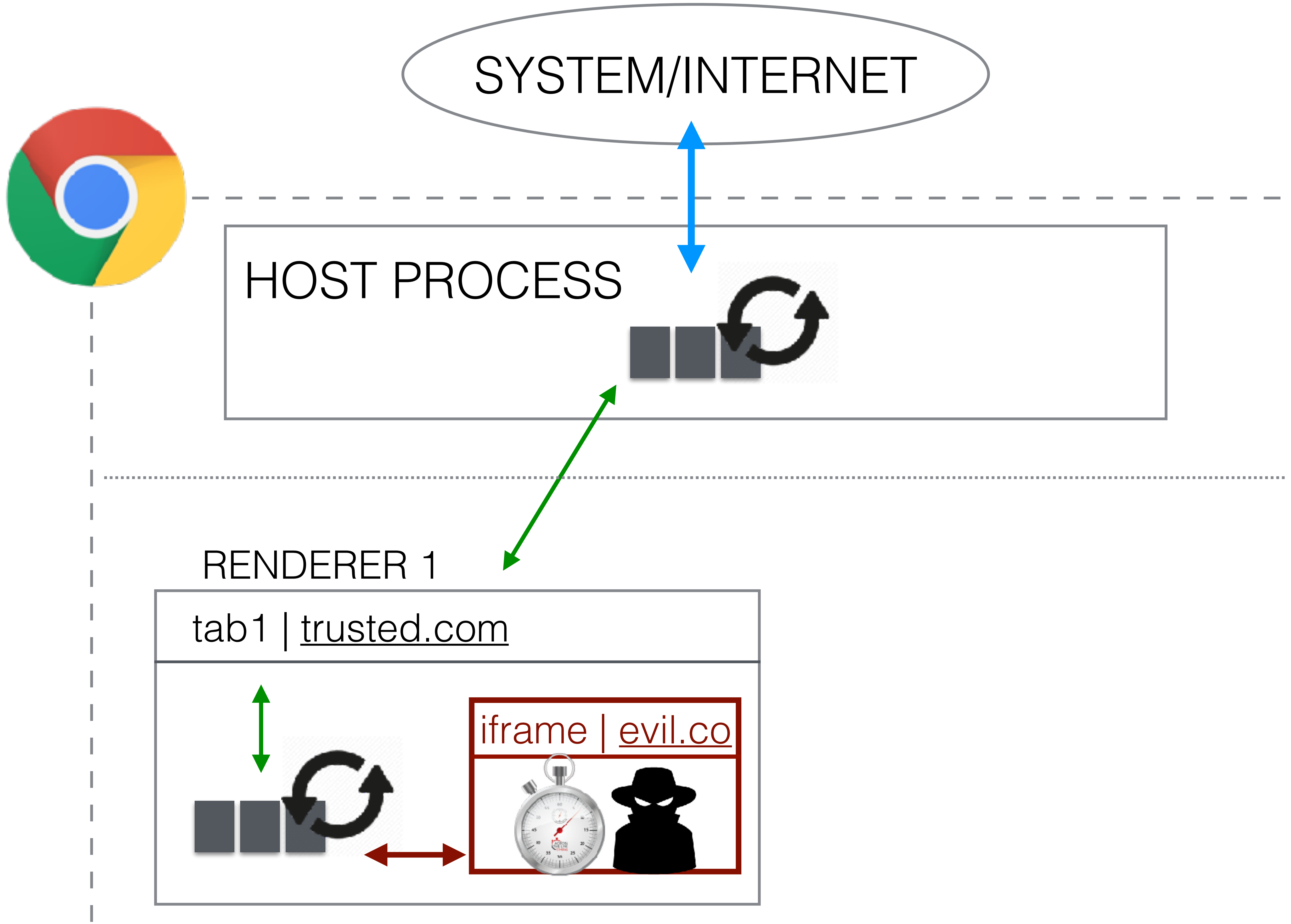
~~Timing resolution of ~500 µs~~

With some smarter techniques we obtain <100 µs
(see the paper)


SYSTEM/INTERNET
HOST PROCESS
RENDERER 1
tab1 | trusted.com


SYSTEM/INTERNET
HOST PROCESS
RENDERER 1
tab1 | trusted.com
• JAVASCRIPT EXECUTION
• RESOURCE PARSING
• LAYOUT & RENDERING


SYSTEM/INTERNET
HOST PROCESS
RENDERER 1
tab1 | trusted.com
iframe |
SHARED BETWEEN IFRAMES, POPUPS, MAX #RENDERER EXCEEDED...



SYSTEM/INTERNET
HOST PROCESS
RENDERER 1
tab1 | trusted.com
iframe | evil.co

Spying on the Renderer

```
<script>
function loop() {
    save(performance.now());
    self.postMessage(0, "*");
}
self.onmessage = loop;
loop();
</script>
```

Timing resolution of <25 μs

Duration of Events in the Renderer

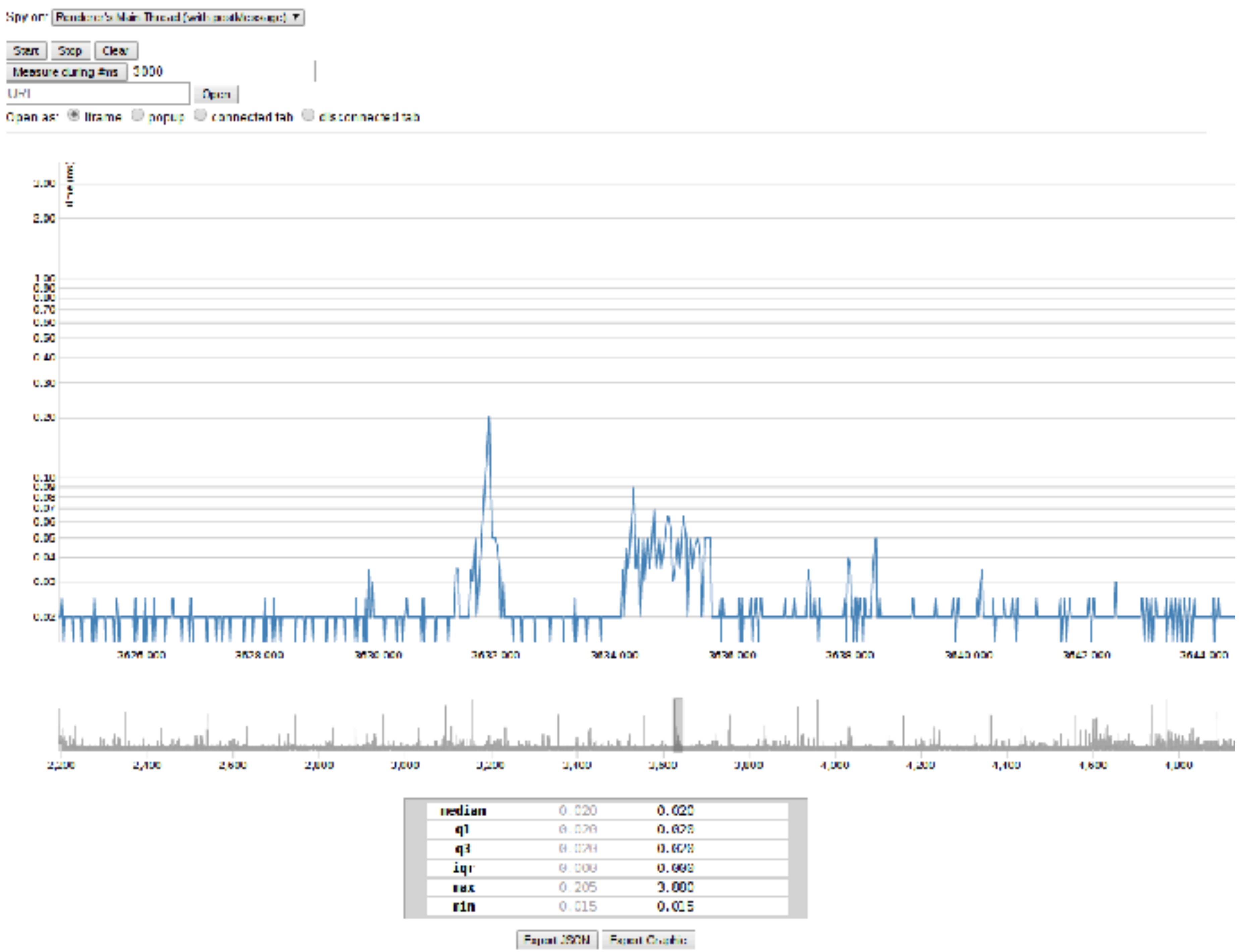
µ-arch events	**loop()**	Mouse movement	GC scavenge	JS event handlers
<5µs	**25 µs**	100 µs	<1 ms	>2 ms

Duration of Events

μ-arch events	**loop()**	Mouse movement	GC scavenge	JS event handlers
<5μs	**25 μs**	100 μs	<1 ms	>2 ms

Responsive code is harder to identify

LoopScan Tool

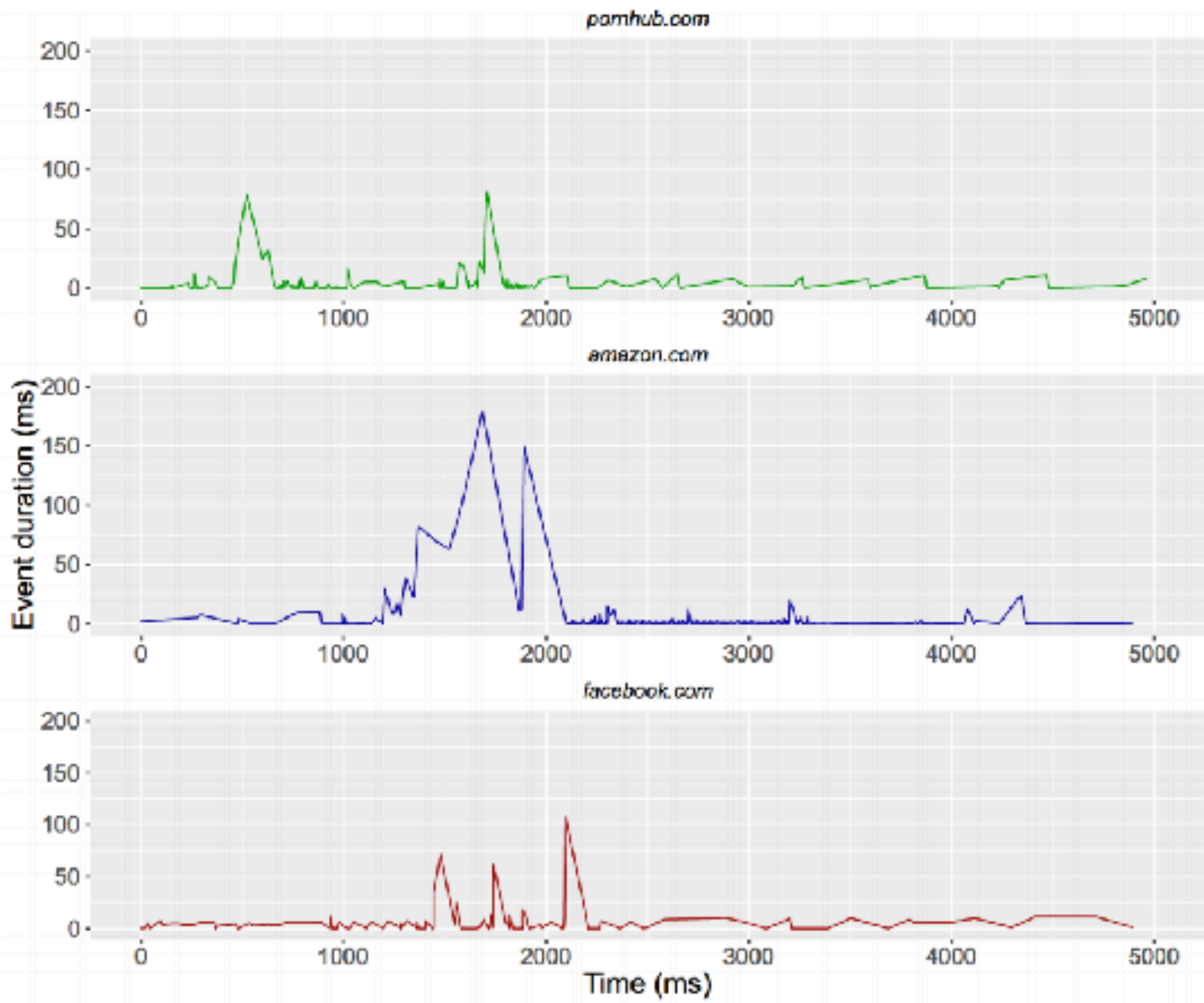


https://github.com/cgvwzq/loopscan

Web Page Identification

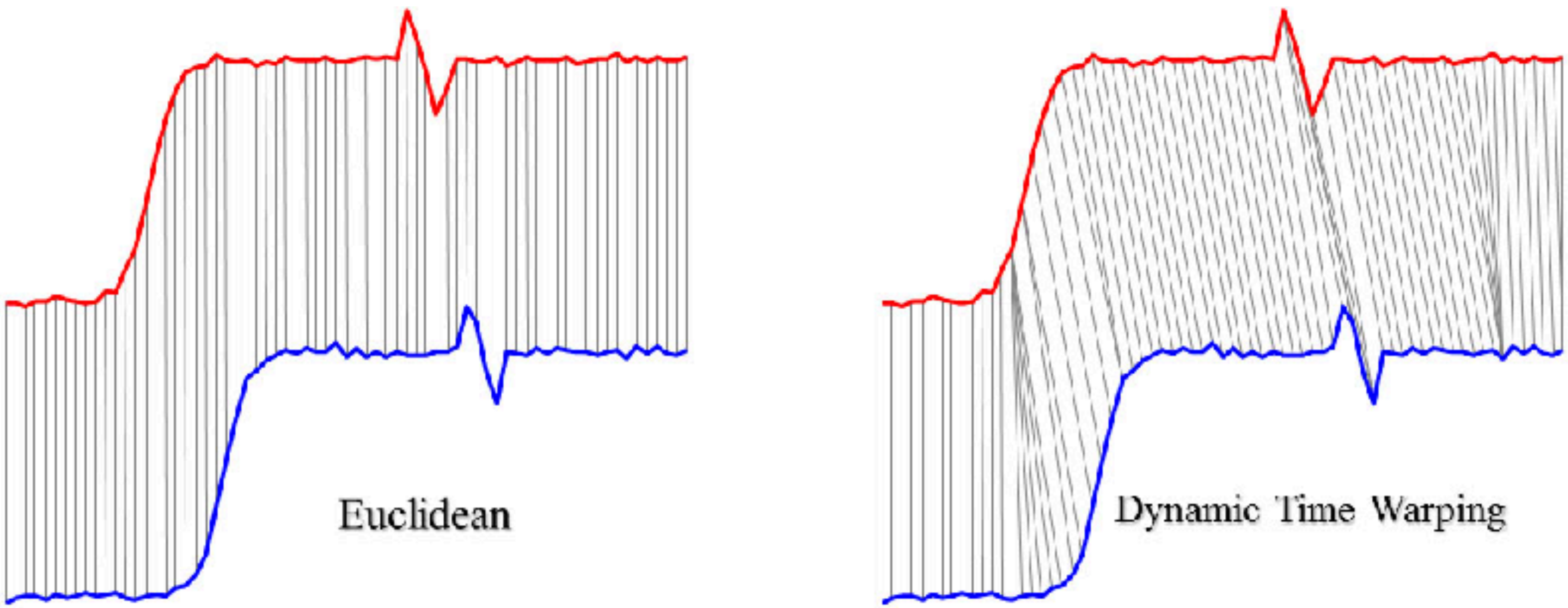
& Inter-keystroke Timing

Web Page Identification



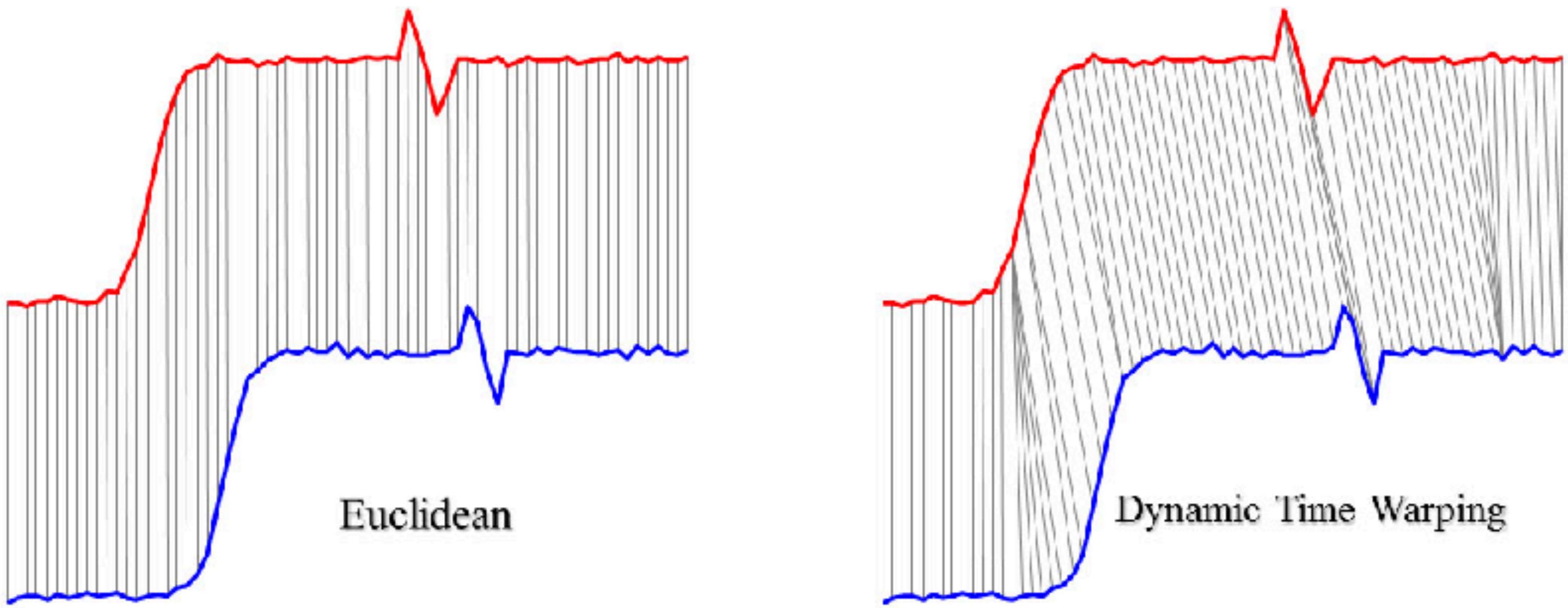
Monitor the EventLoop while page loading

Dynamic Time Warping



DTW is resistant to delays in the occurrence of events

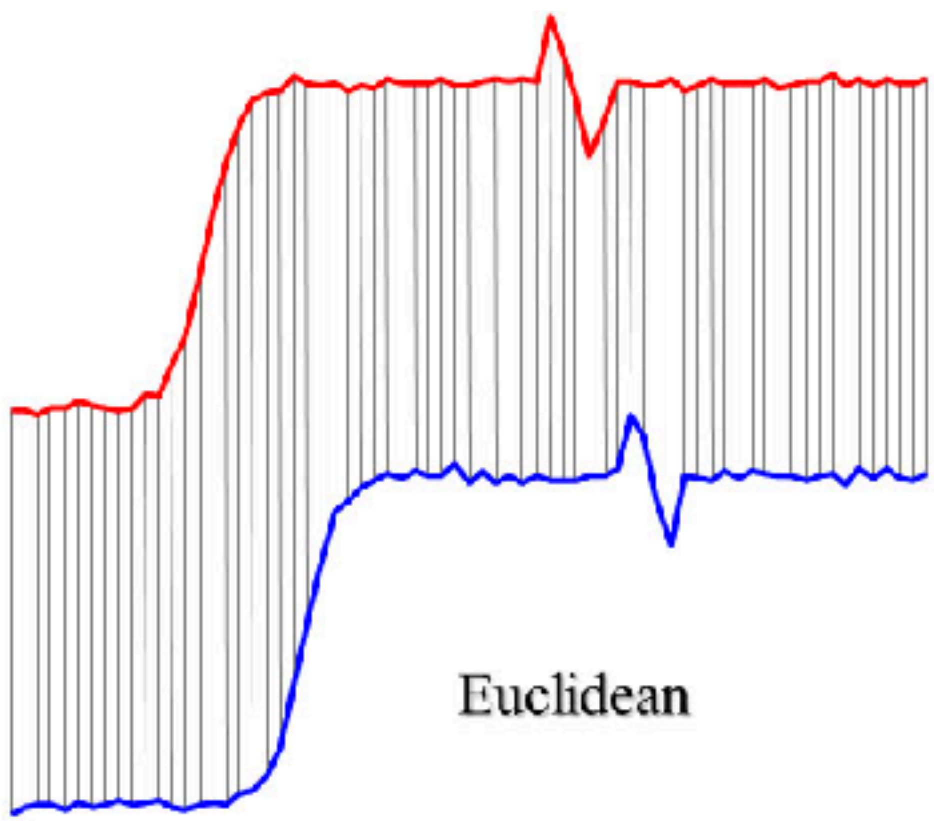
Dynamic Time Warping

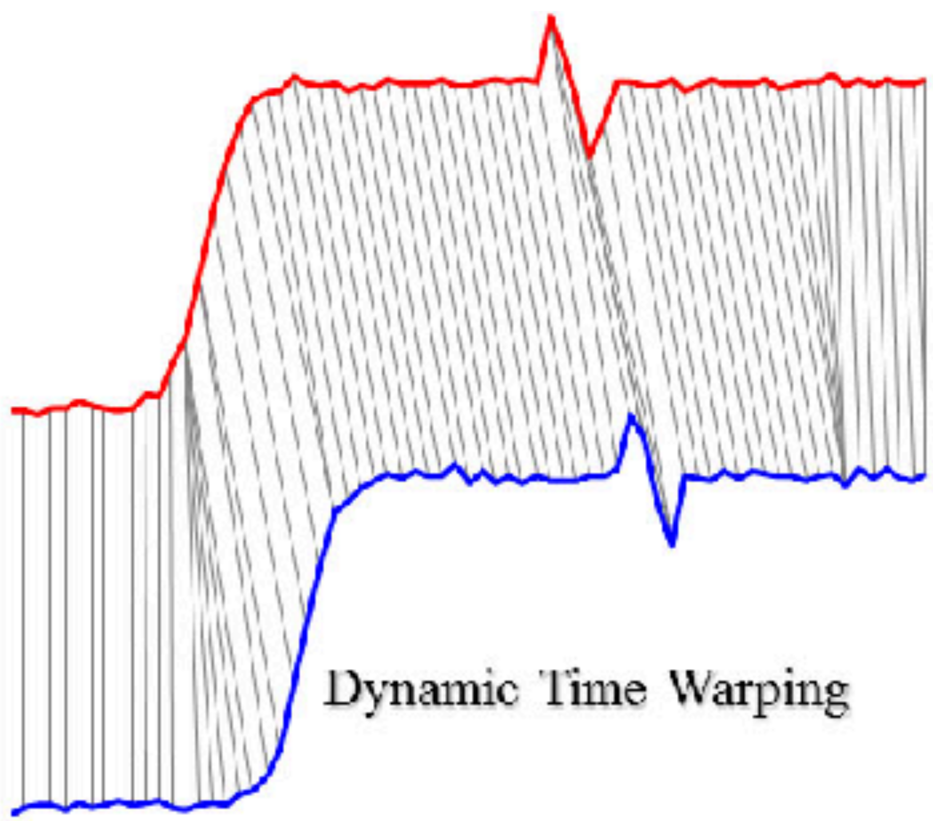


DTW is resistant to delays in the occurrence of events

2-4 seconds of
measuring

Dynamic Time Warping





DTW is resistant to delays in the occurrence of events

2-4 seconds of measuring

One trace for training

Web Page Identification

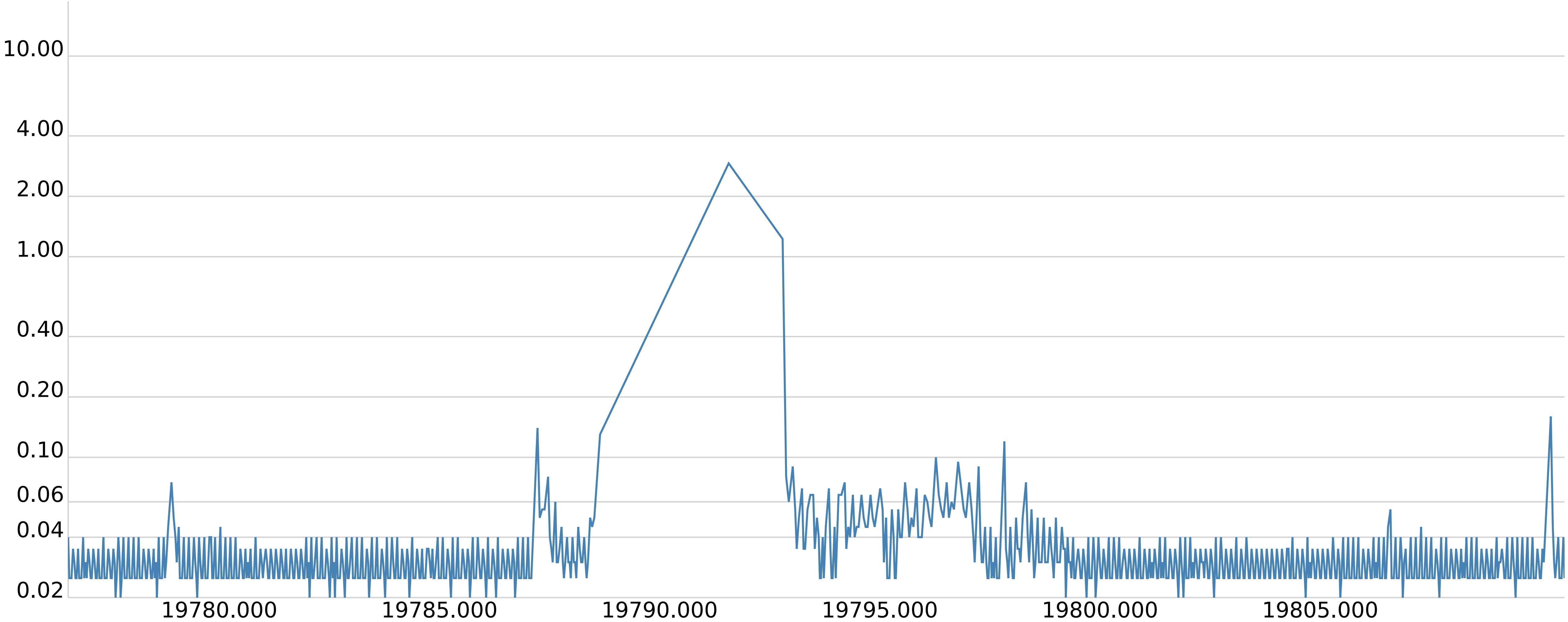
500 pages x 30 traces x 3 machines x 2 event loops

Renderer’s main thread:	**75%**	(Linux desktop)
Host’s I/O thread:	**23%**	(Macbook Pro)

(recognition rates below 5% across machines)

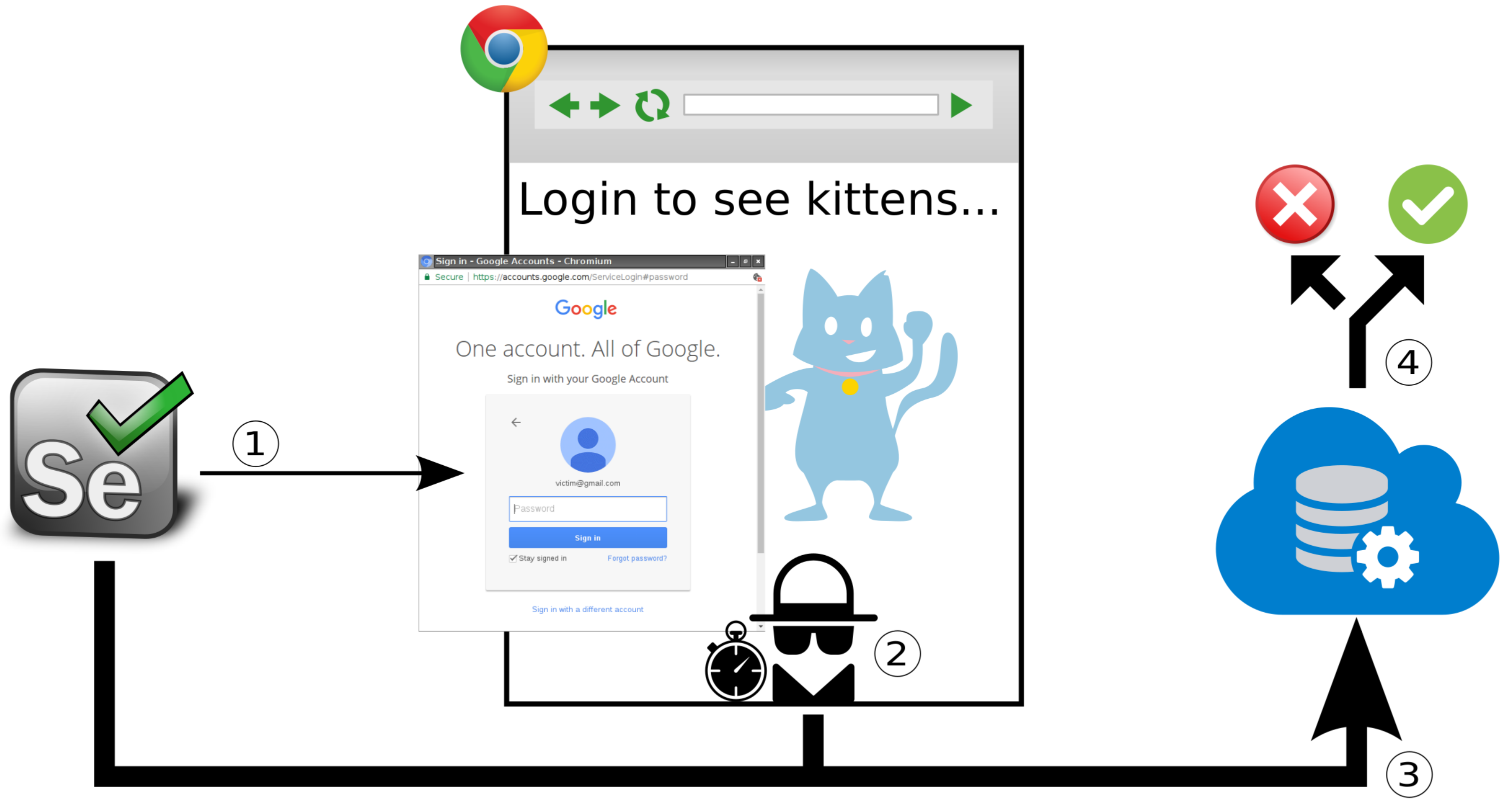
R-library and datasets:
https://github.com/cgvwzq/rlang-loophole

Inter-keystroke Timing



We obtain the password length and time between consecutive pressed keys

Inter-keystroke Timing

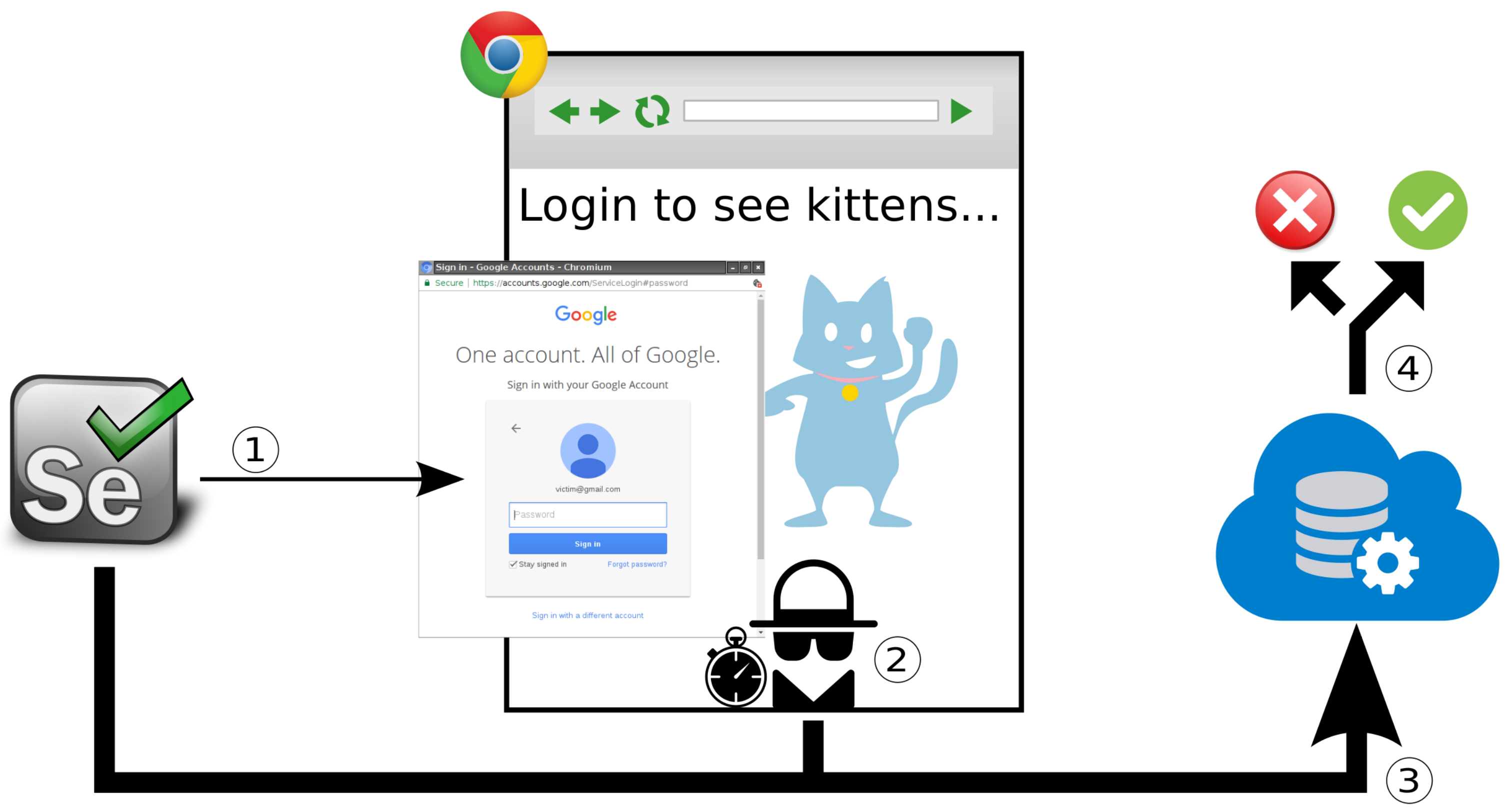


10.000 passwords

90% accuracy

precision: σ = 6.1 ms

Inter-keystroke Timing



10.000 passwords

90% accuracy

precision: σ = 6.1 ms

More precision than network based attacks.

Less noise than in micro-architectural attacks.

No privileges. No training.

Countermeasures

- Reduce clock resolution
- Site Isolation Project
- CPU throttling
- Rate limiting

Countermeasures

- ~~Reduce clock resolution~~
- Site Isolation Project
- CPU throttling
- Rate limiting

Conclusions

- Shared event loops in Chrome are vulnerable to timing side-channels
- We systematically study how this channel can be used for different attacks
- Fundamental design issues that need to be addressed

Thank you! :)

Questions?