Sinatra: Stateful Instantaneous Updates for Commercial Browsers through Multi-Version eXecution

Ugnius Rumsevicius
Siddhanth Venkateshwaran
Ellen Kidane
Luís Pina

University of Illinois Chicago

NSF CCF-2227183

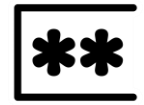
UIC ENGINEERING

"Google is aware that an exploit for CVE-2023-2136 exists in the wild," the company warned.

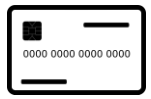
The official CVE report adds that exploiting the flaw (…) could pave a way for the hacker to (…) run untrusted malicious code on a computer

https://www.pcmag.com/news/google-detects-second-zero-day-chrome-exploit-days-after-patching-another

Browsers keep private data



Passwords

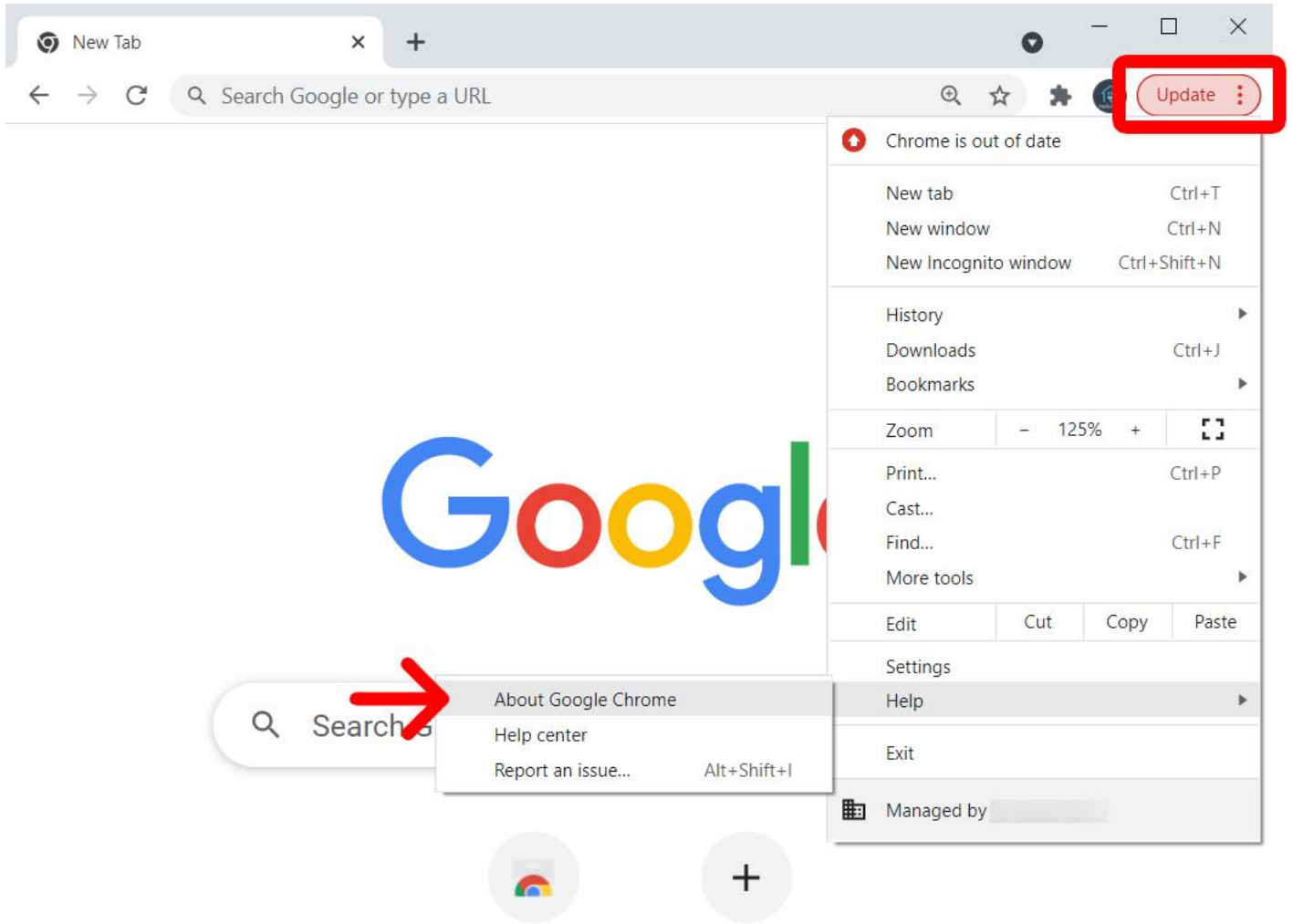


Credit Cards

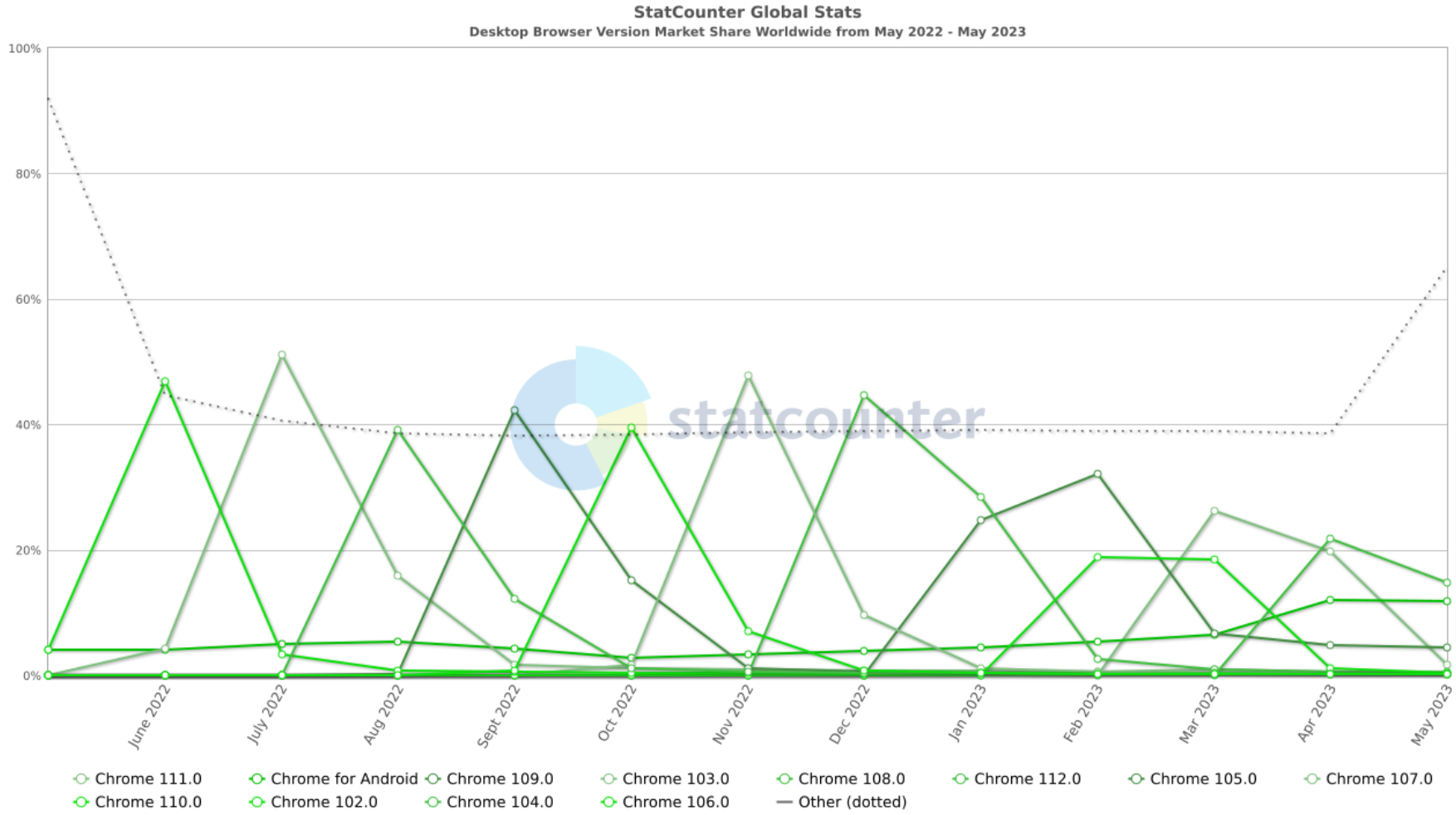
Home address

Cookies

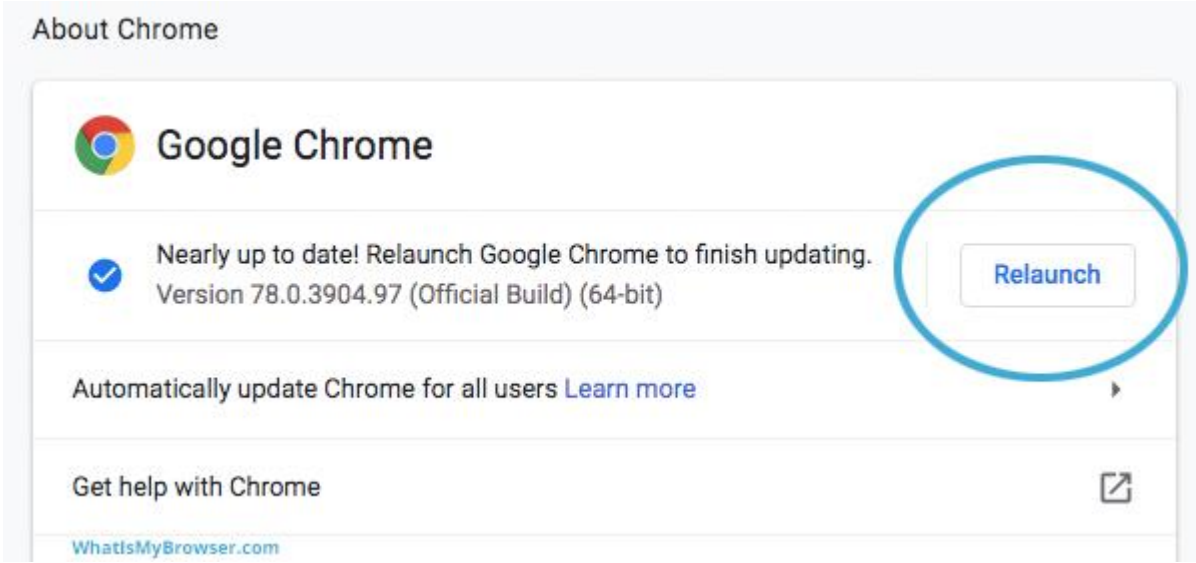
Best defense: Frequent updates



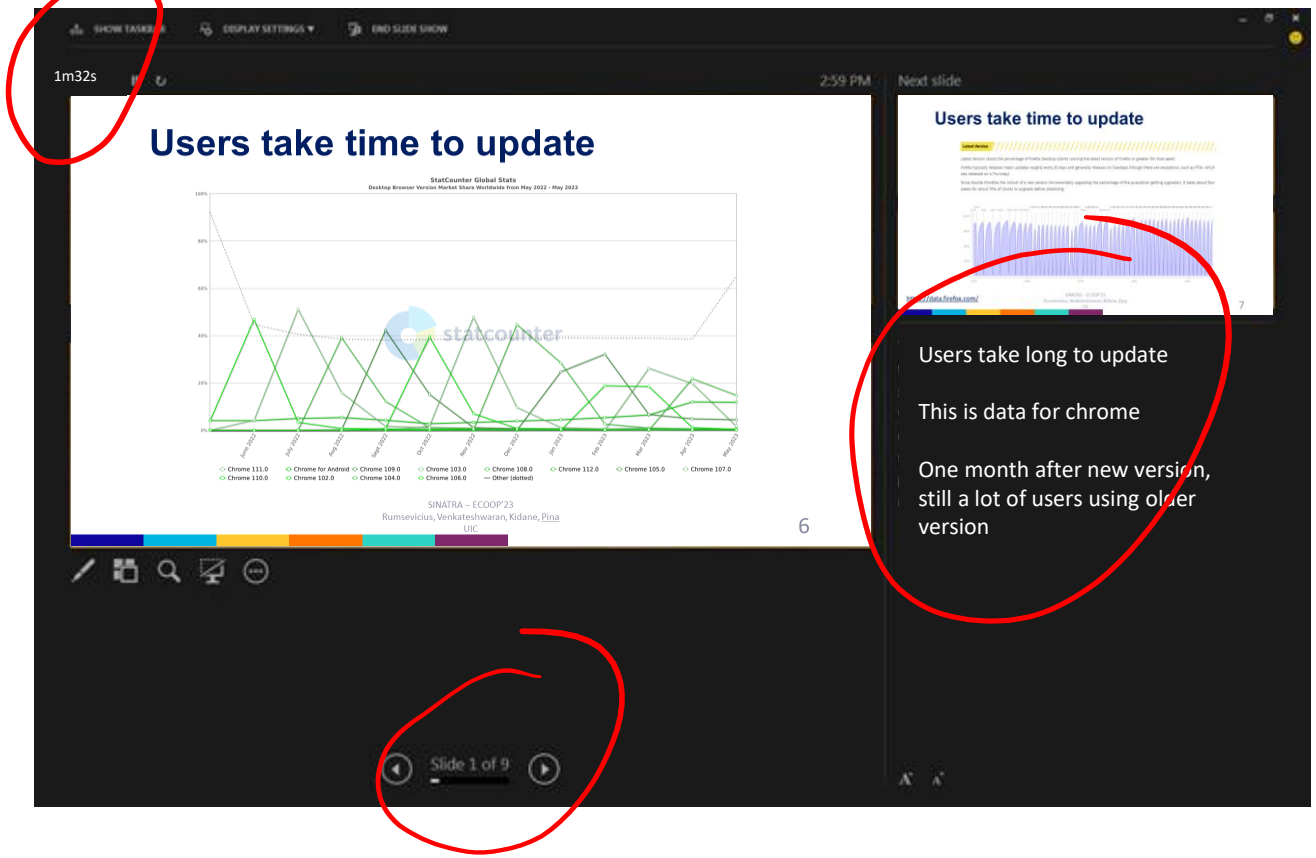
Users take time to update



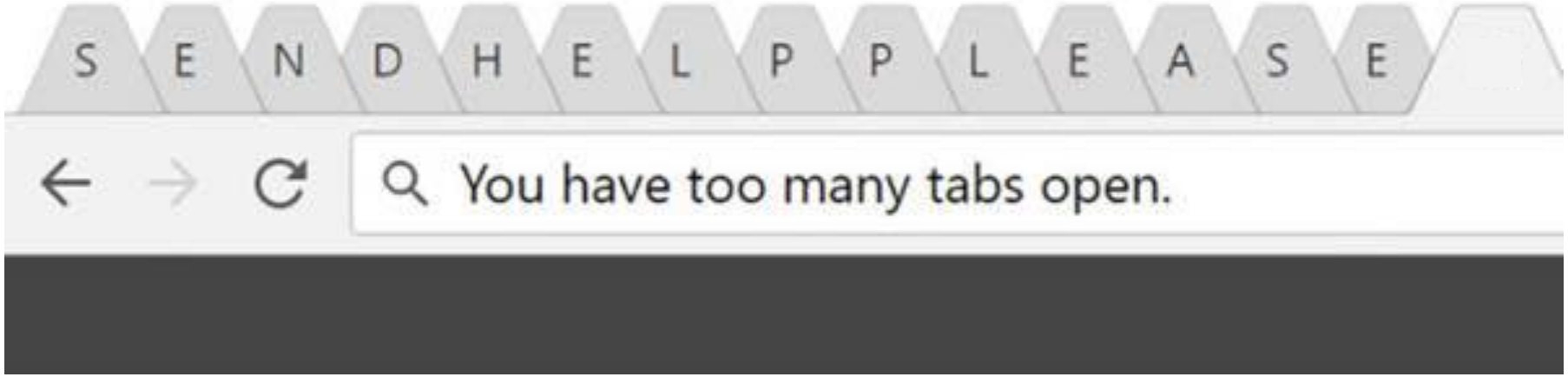
Browser updates take time



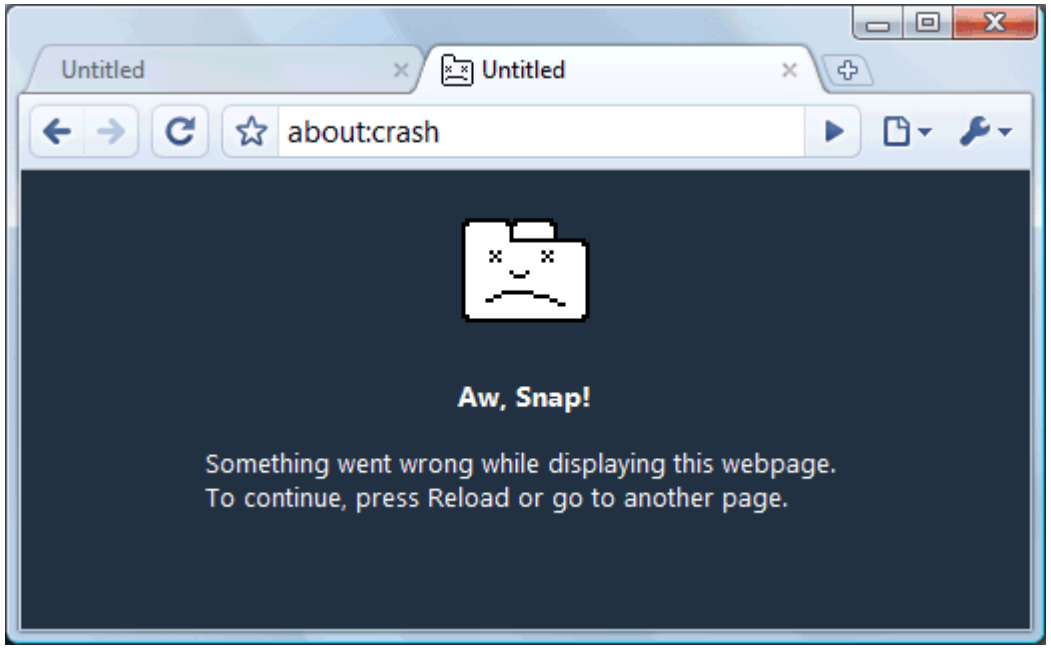
Browser updates lose state

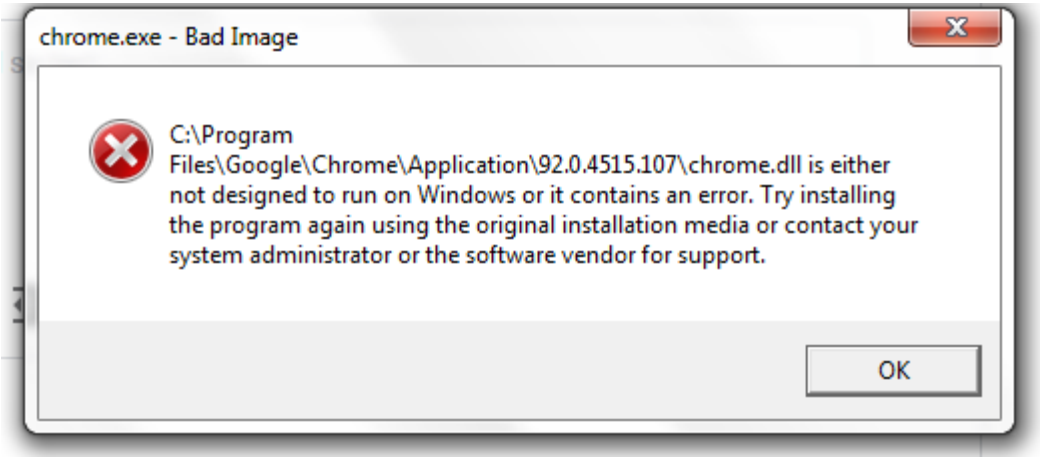


Browser updates lose state

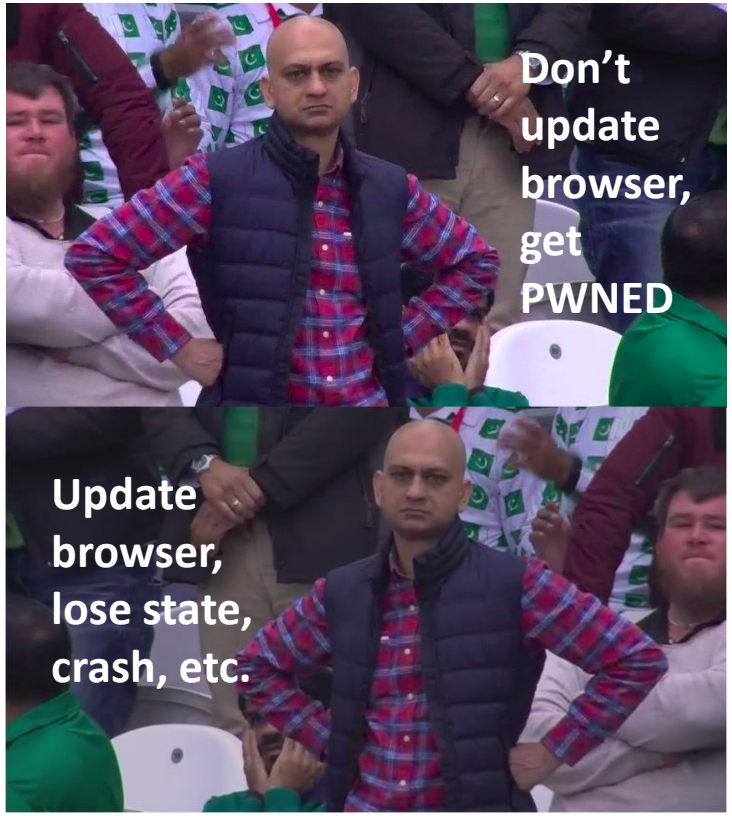


Browser updates may fail

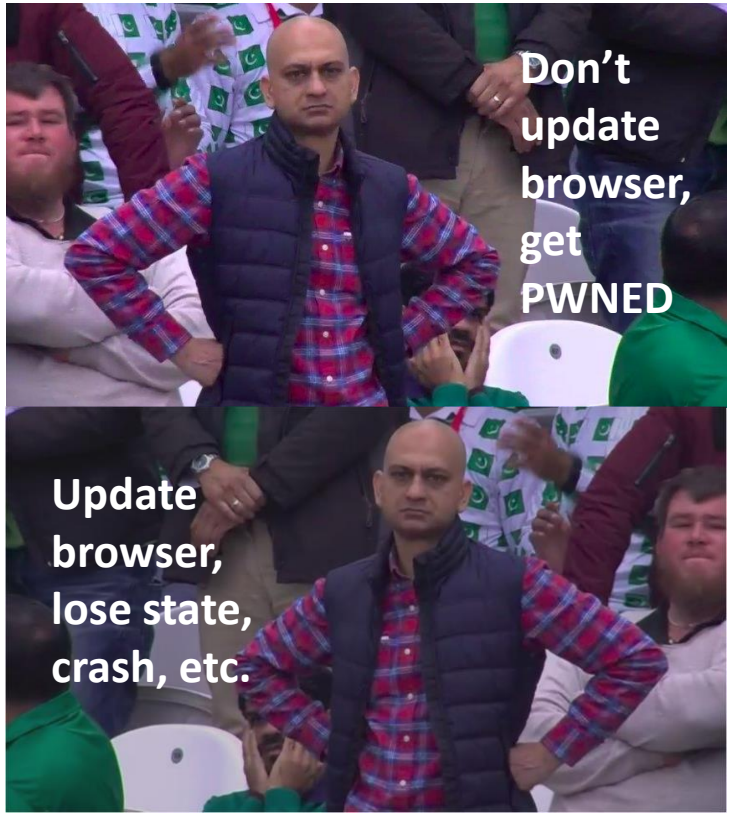




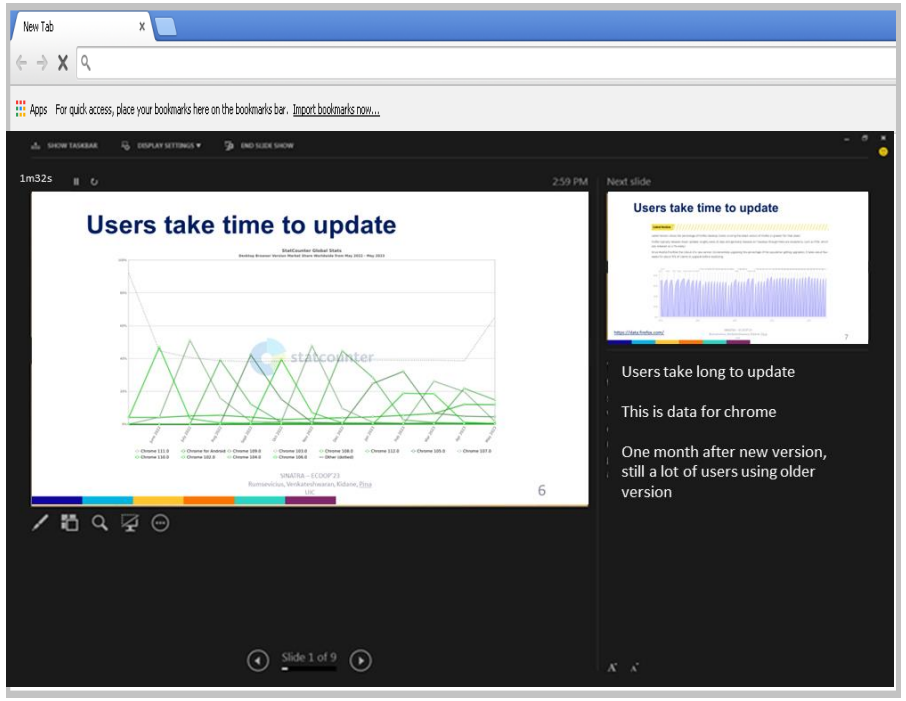
No good option



No good option



SINATRA



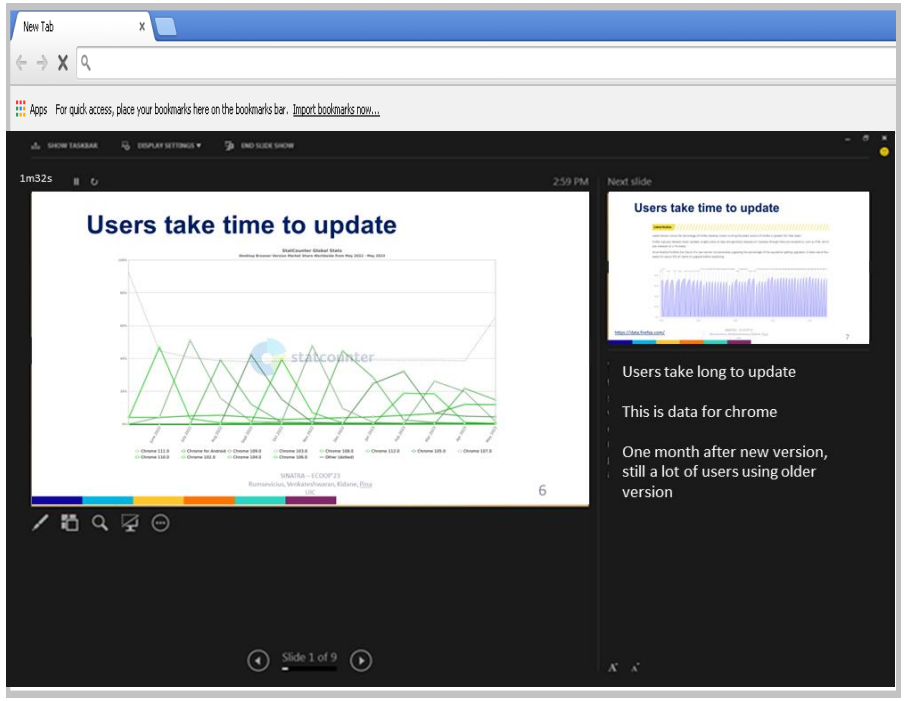
First step in automatic updates

Browser updates as soon as possible without user noticing

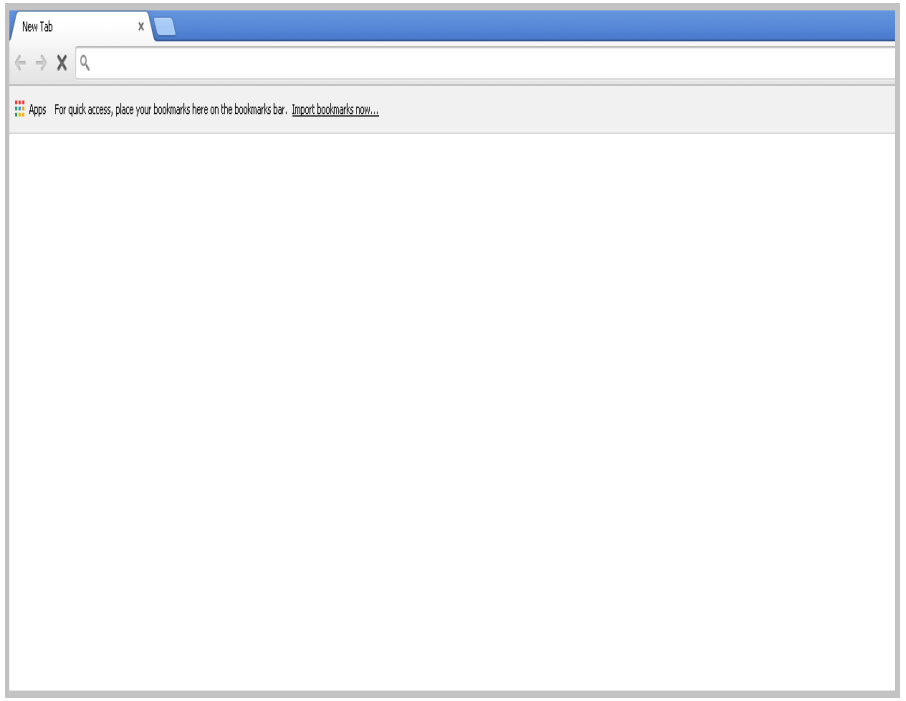
Only 10ms pause to perform updates

Browser agnostic

SINATRA

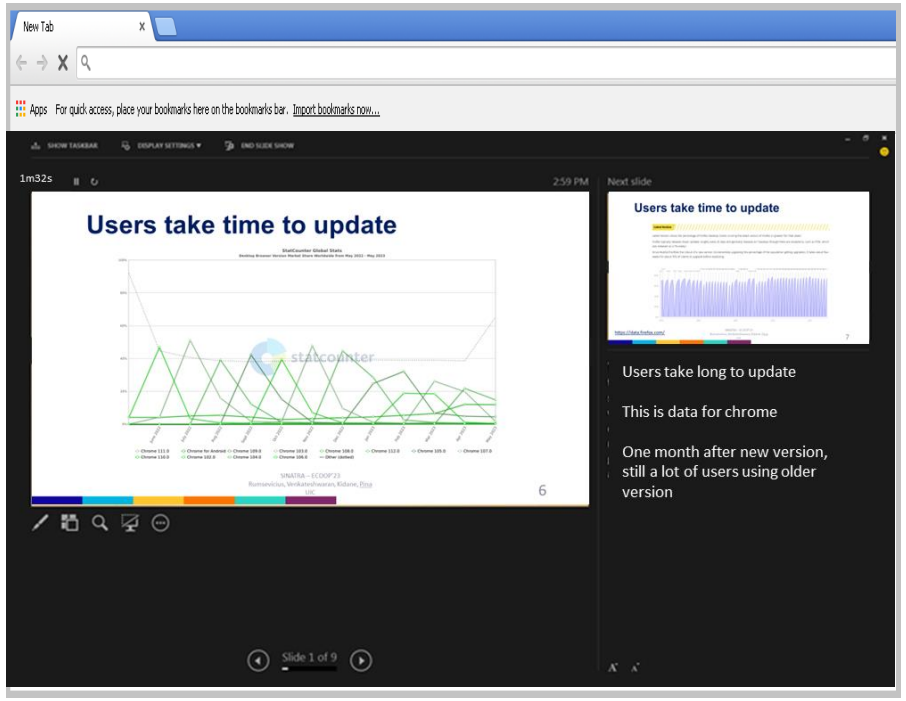


Old browser version

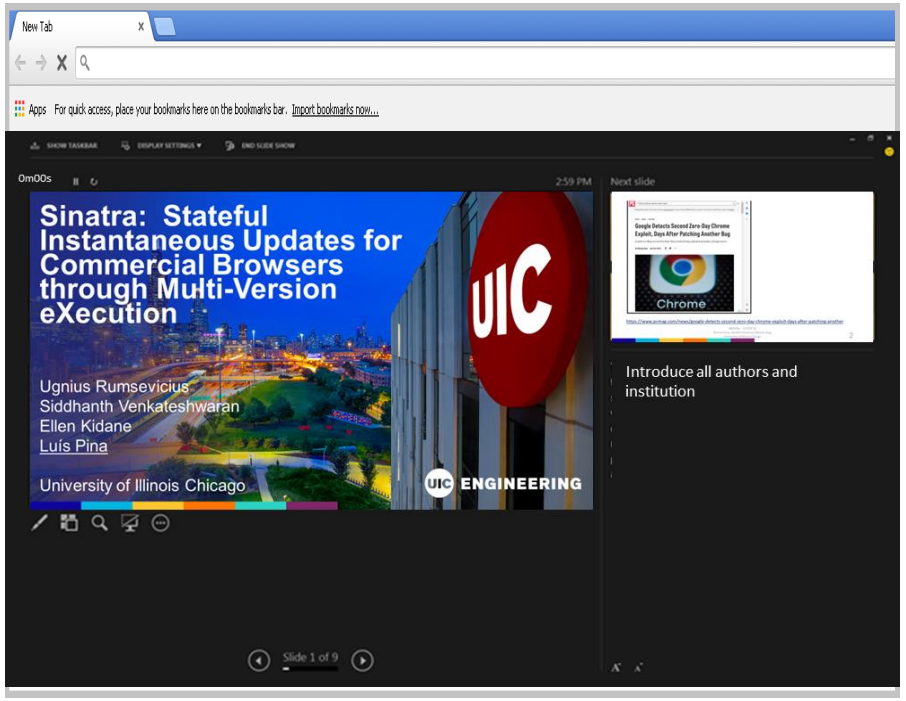


1. Launch new browser version

SINATRA

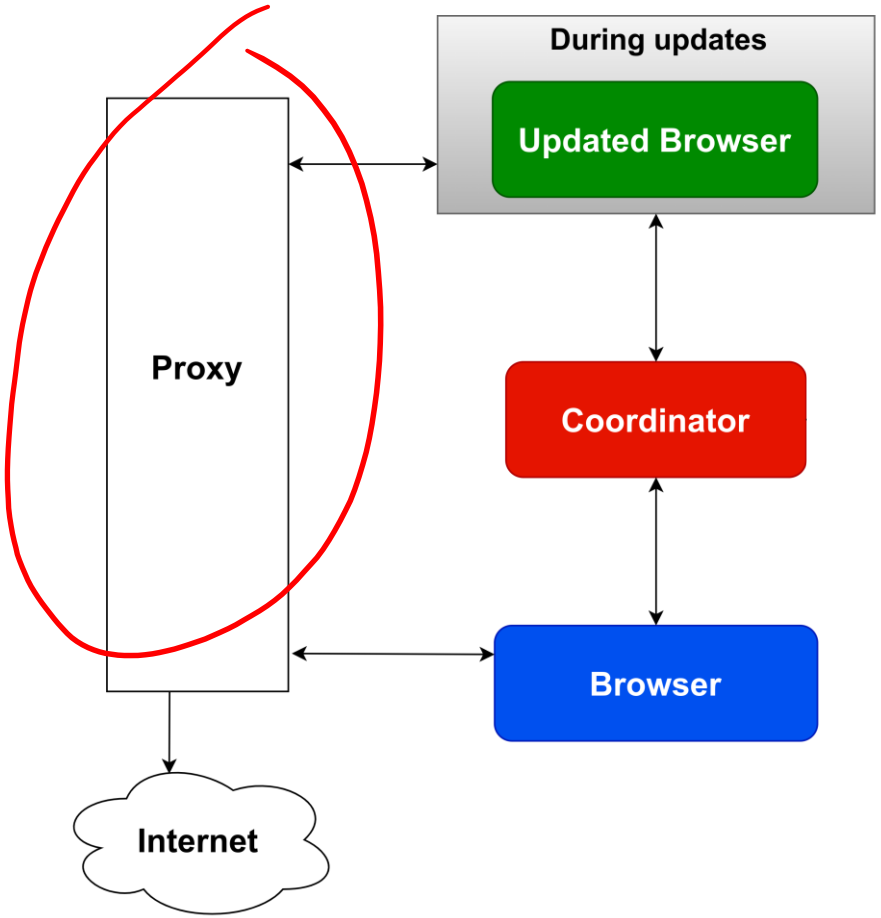


Old browser version

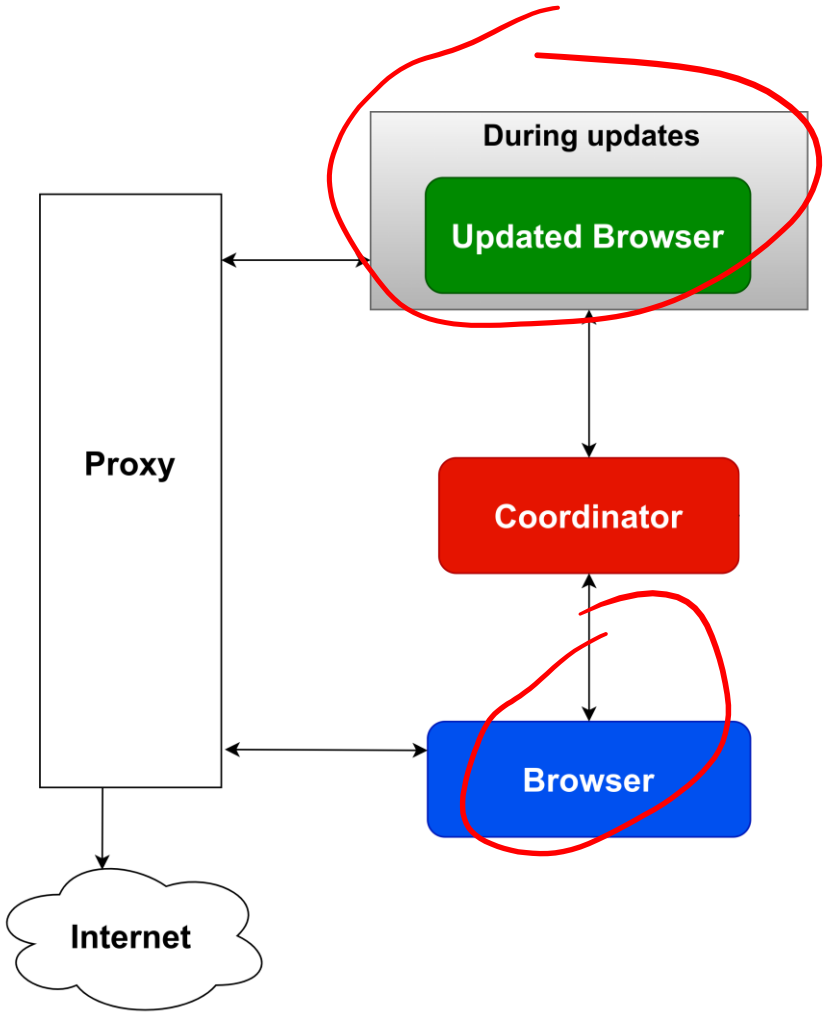


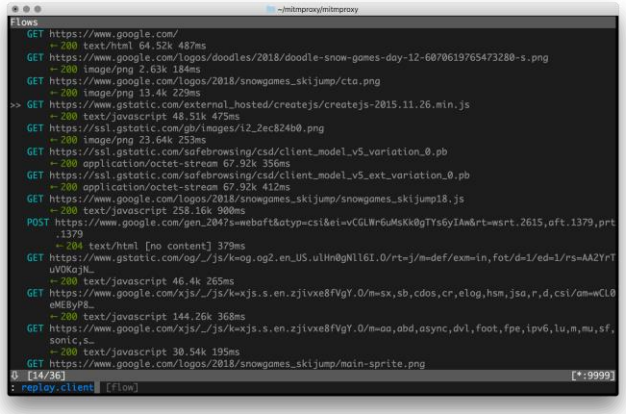
1. Launch new browser version
2. Open same page, get same contents

SINATRA architecture



SINATRA architecture

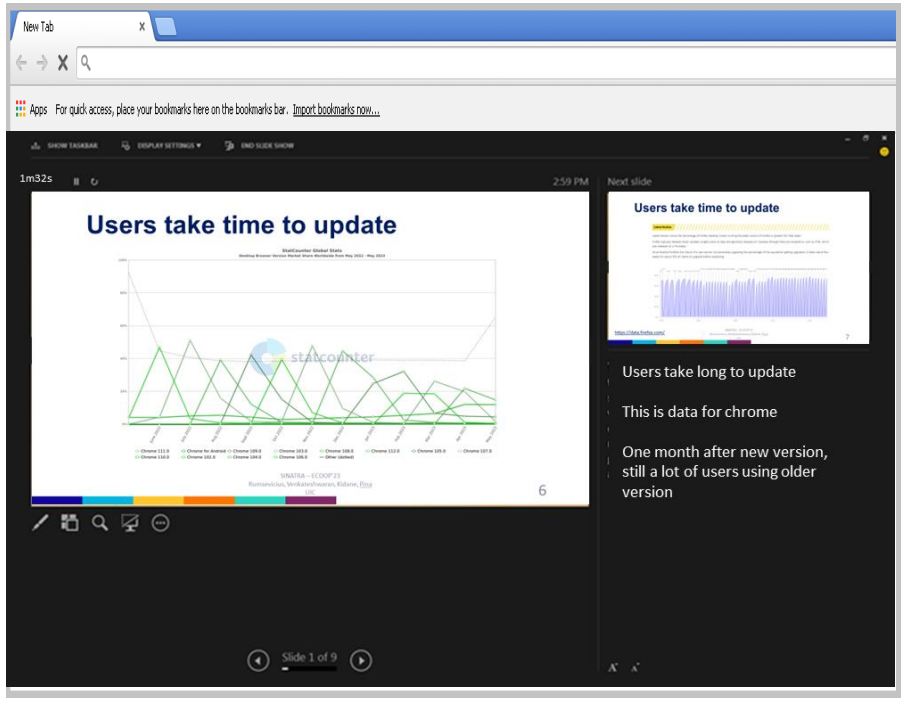




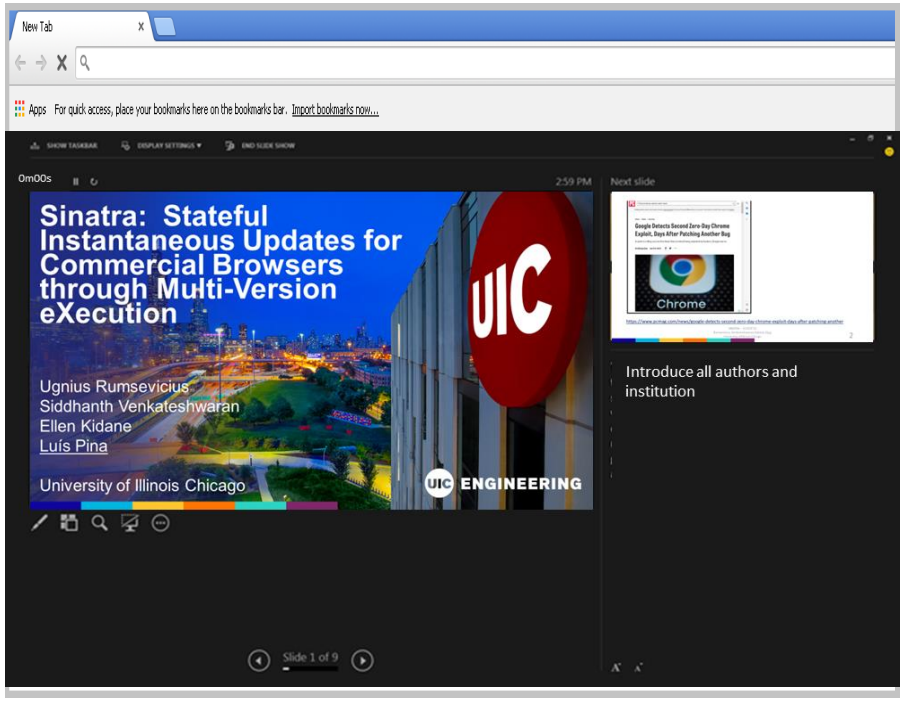
mitmproxy supports HTTPS and sophisticated rules

https://mitmproxy.org/

SINATRA

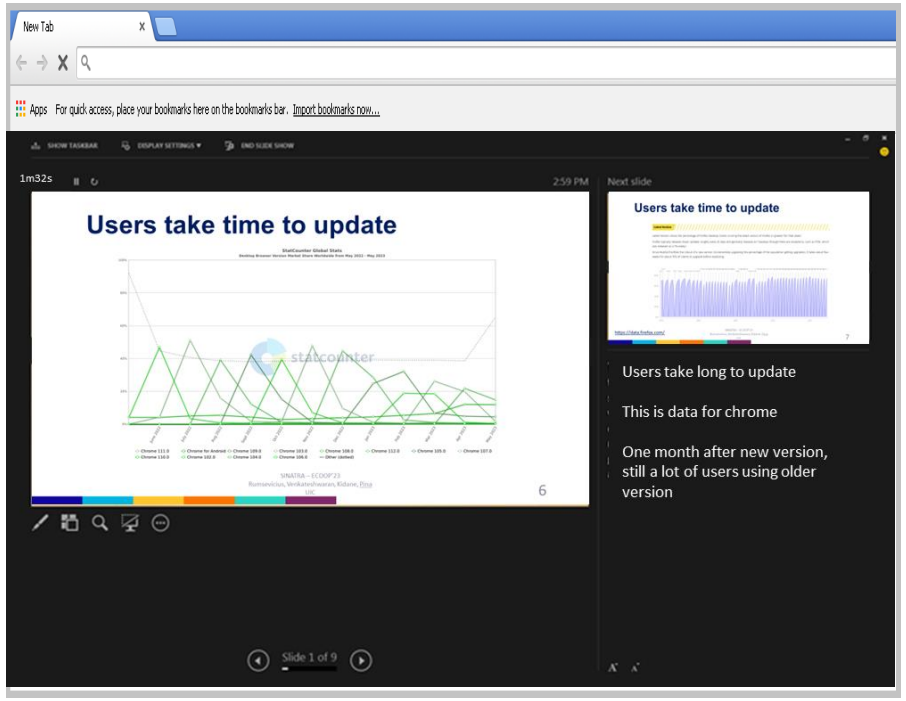


Old browser version

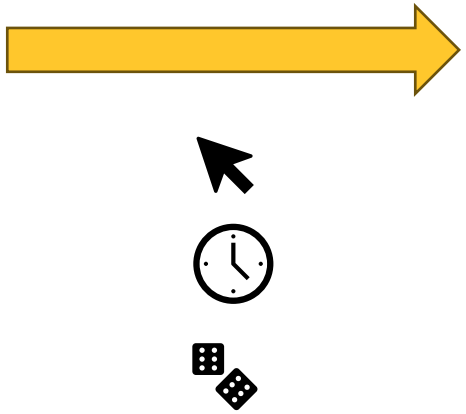


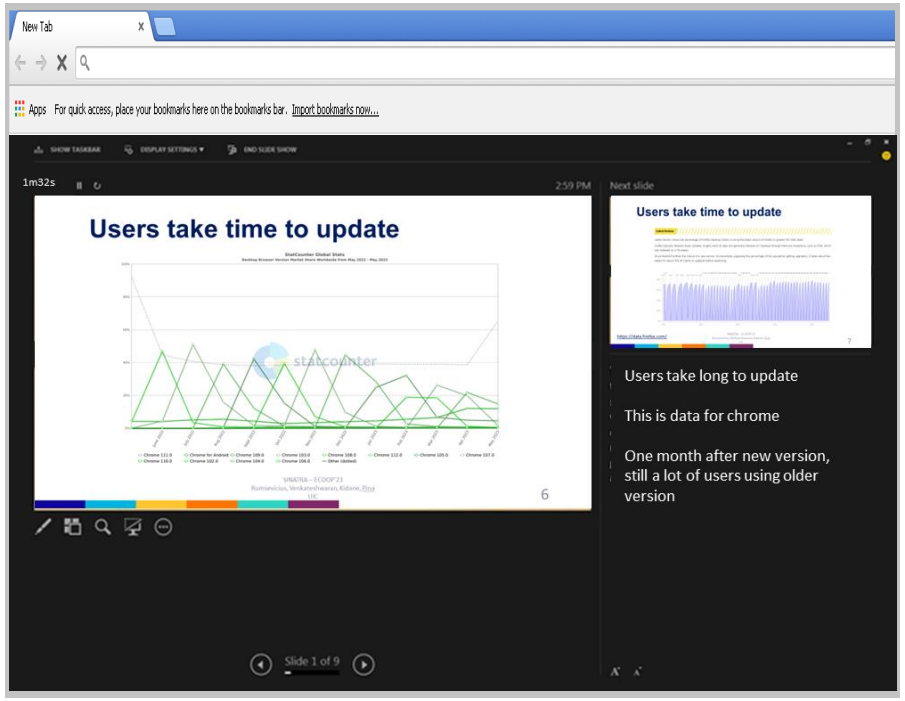
1. Launch new browser version
2. Open same page, get same contents
3. Replay all events from old browser

SINATRA



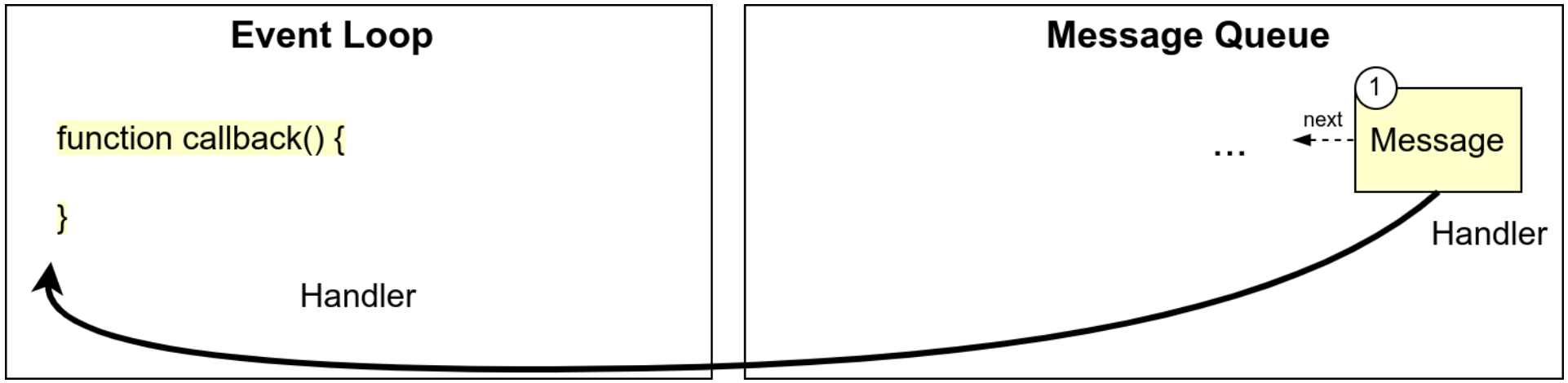
Old browser version



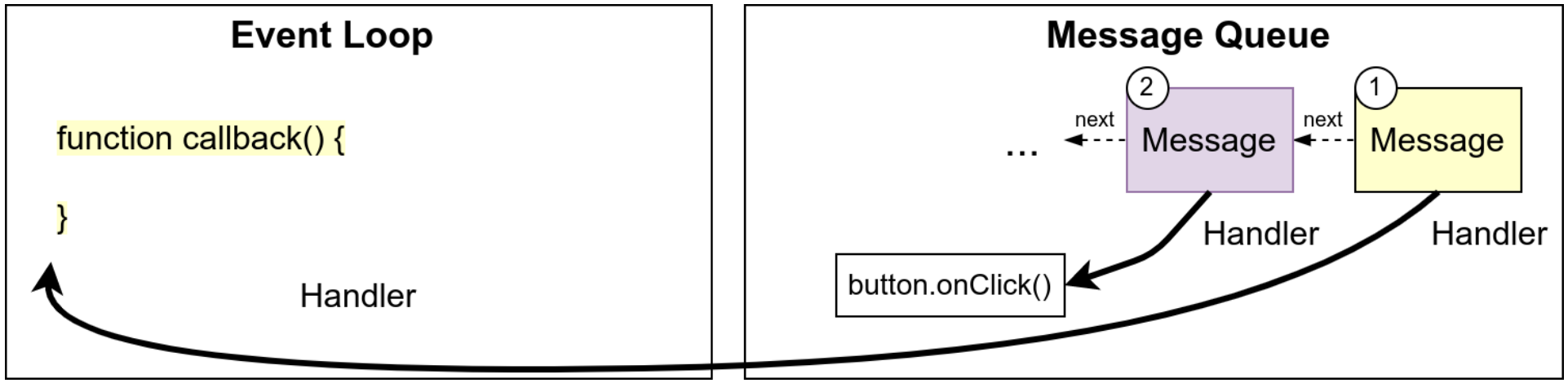


1. Launch new browser version
2. Open same page, get same contents
3. Replay all events from old browser

Javascript Events



Javascript Events



Intercepting Javascript Events

```
1   <html>
2       <head>
3   +       <script src="mvx/sinatra.js" type="text/javascript"></script>
4   +       <script src="mvx/sinatra_init.js" type="text/javascript" defer></script>
5           ...
6       </head>
7       <body>
8           ...
9       </body>
10  </html>
```

Intercepting Javascript Events

```
1 const originalAddEventListener = HTMLElement.prototype.addEventListener;
2 HTMLElement.prototype.addEventListener = function(evType, evListener, u) {
3
4
5
6
7
8 }
```

Intercepting Javascript Events

```
1 const originalAddEventListener = HTMLElement.prototype.addEventListener;
2 HTMLElement.prototype.addEventListener = function(evType, evListener, u) {
3   let closure = function (ev) {
4
5
6   }
7   originalAddEventListener.call(this, evType, closure, u);
8 }
```

Intercepting Javascript Events

```
1 const originalAddEventListener = HTMLElement.prototype.addEventListener;
2 HTMLElement.prototype.addEventListener = function(evType, evListener, u) {
3   let closure = function (ev) {
4
5     evListener.call(this, ev, u);
6   }
7   originalAddEventListener.call(this, evType, closure, u);
8 }
```

Intercepting Javascript Events

```
1 const originalAddEventListener = HTMLElement.prototype.addEventListener;
2 HTMLElement.prototype.addEventListener = function(evType, evListener, u) {
3   let closure = function (ev) {
4     sendToCoordinator(evType, ev);
5     evListener.call(this, ev, u);
6   }
7   originalAddEventListener.call(this, evType, closure, u);
8 }
```

Intercepting Javascript Events

Fully implemented in Javascript

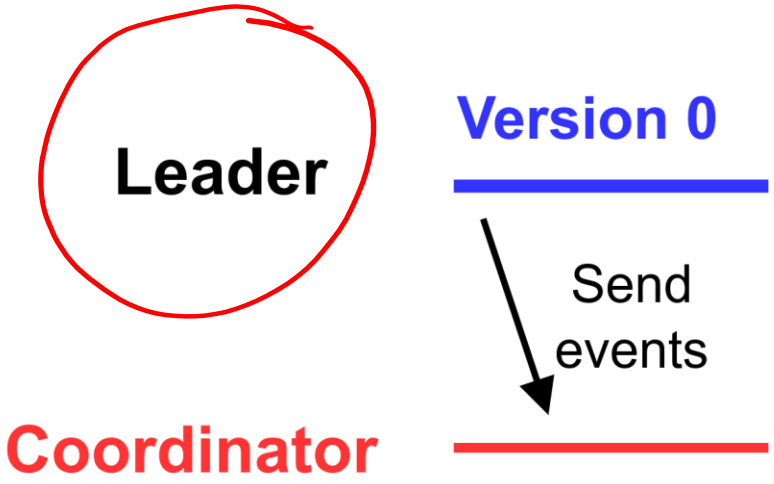
```
1                                                                    ;
2                                                                 u) {
3     let closure = function (ev) {
4       sendToCoordinator(evType, ev);
5       evListener.call(this, ev, u);
6     }
7     originalAddEventListener.call(this, evType, closure, u);
8   }
```

Intercepting Javascript Events

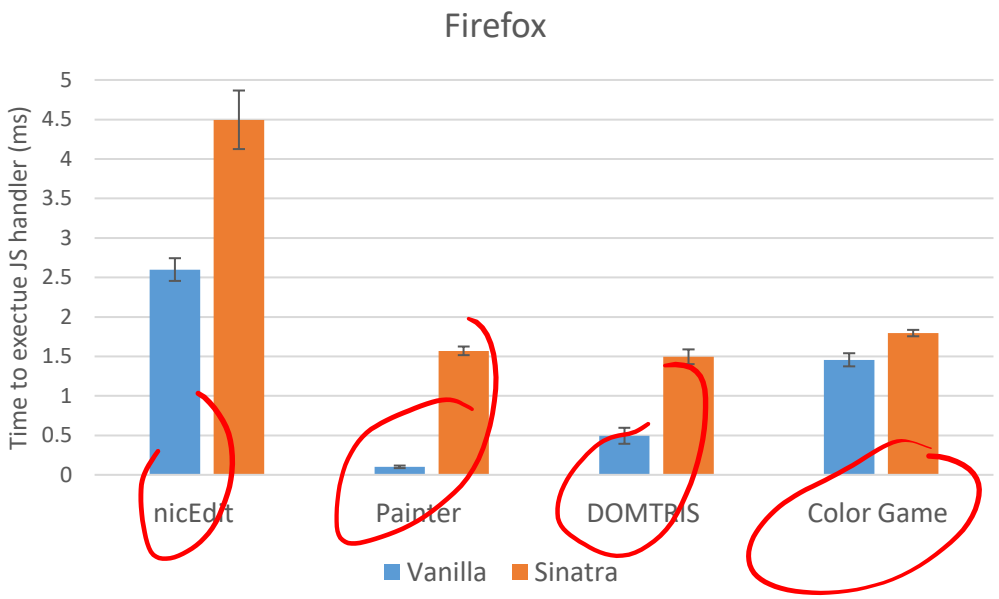
Fully implemented in Javascript

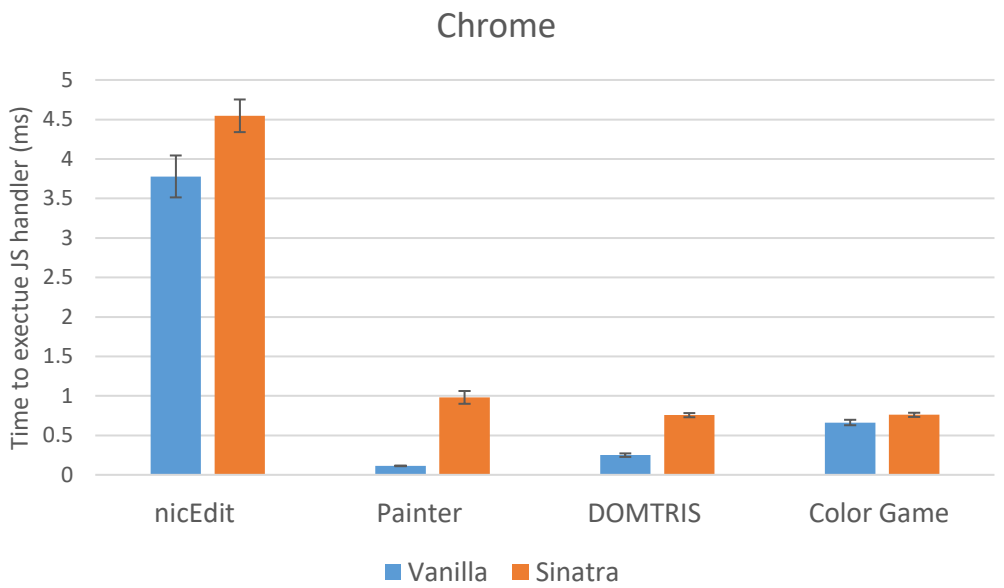
Browser agnostic

SINATRA phases



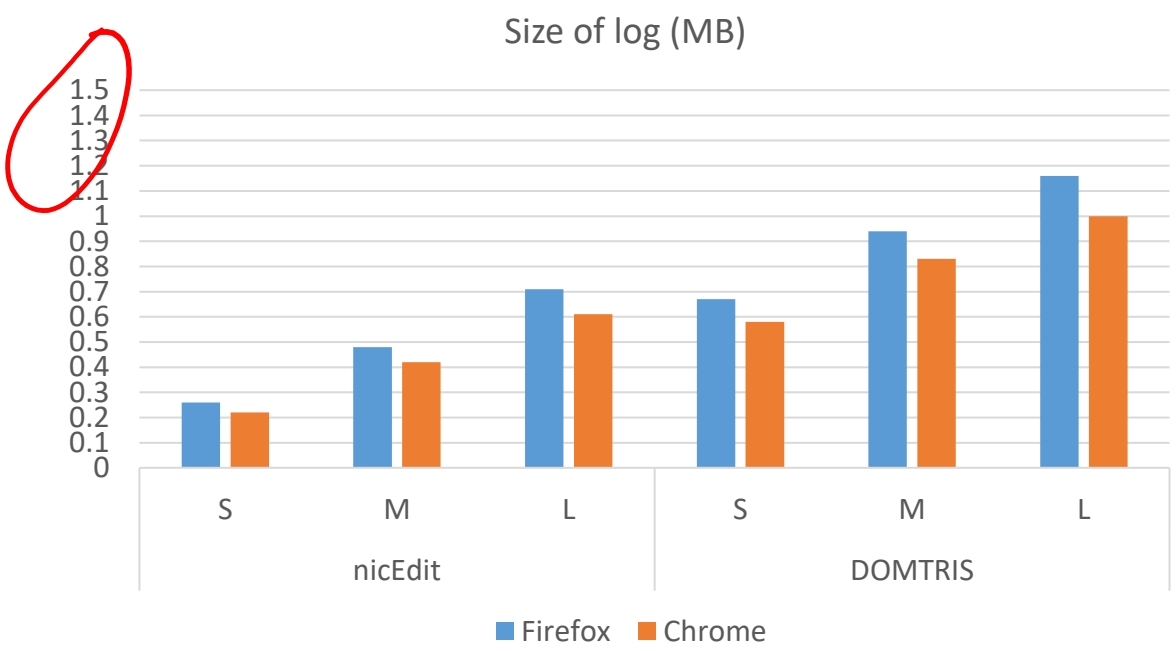
Intercepting Javascript Events

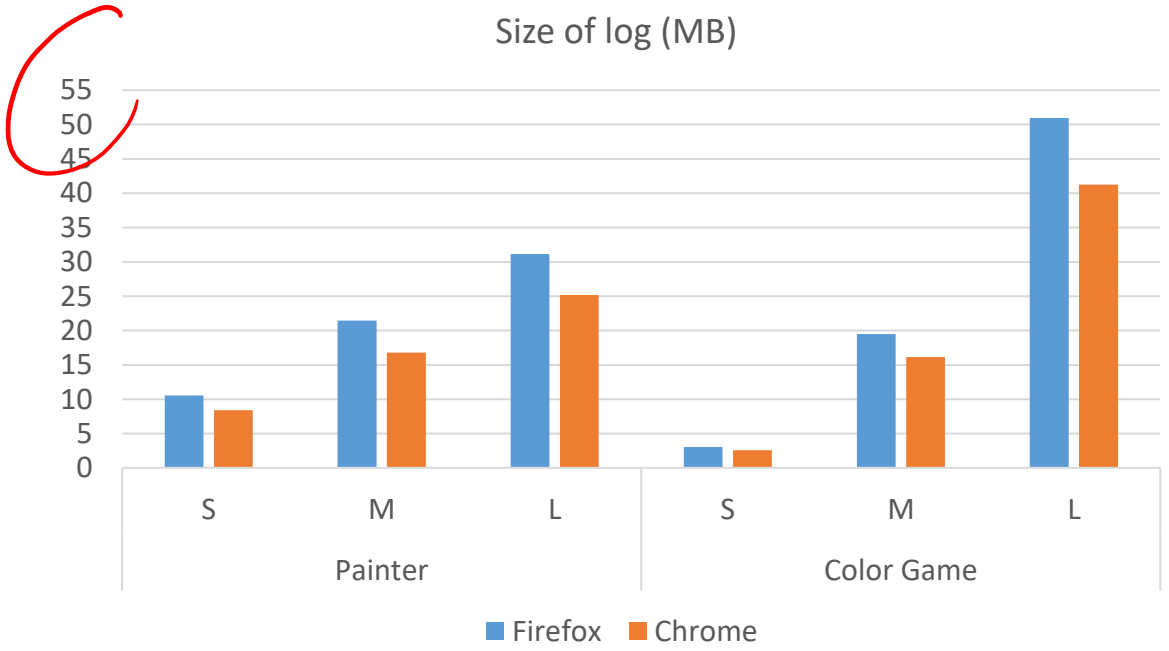




Low overhead (max +1.896ms per handler)

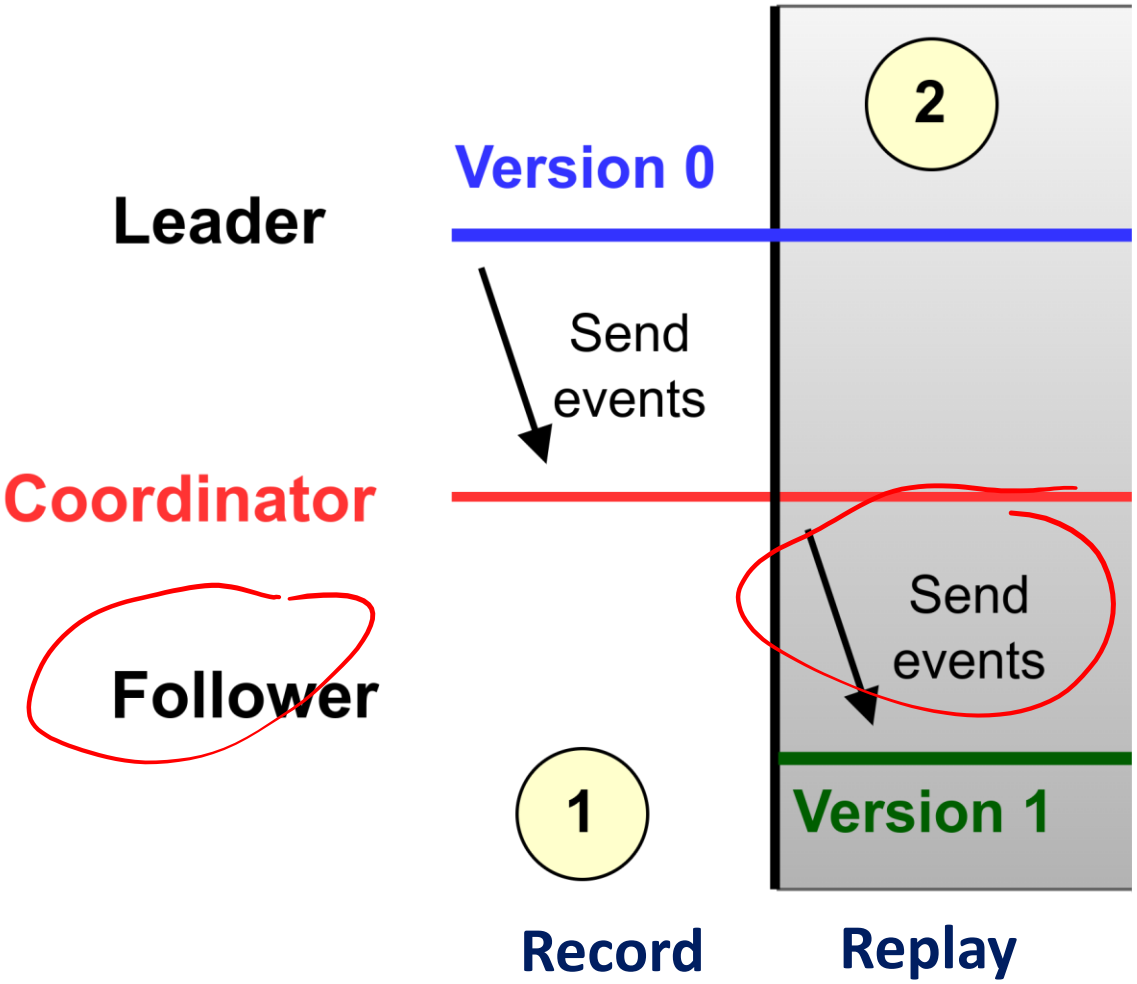
Intercepting Javascript Events



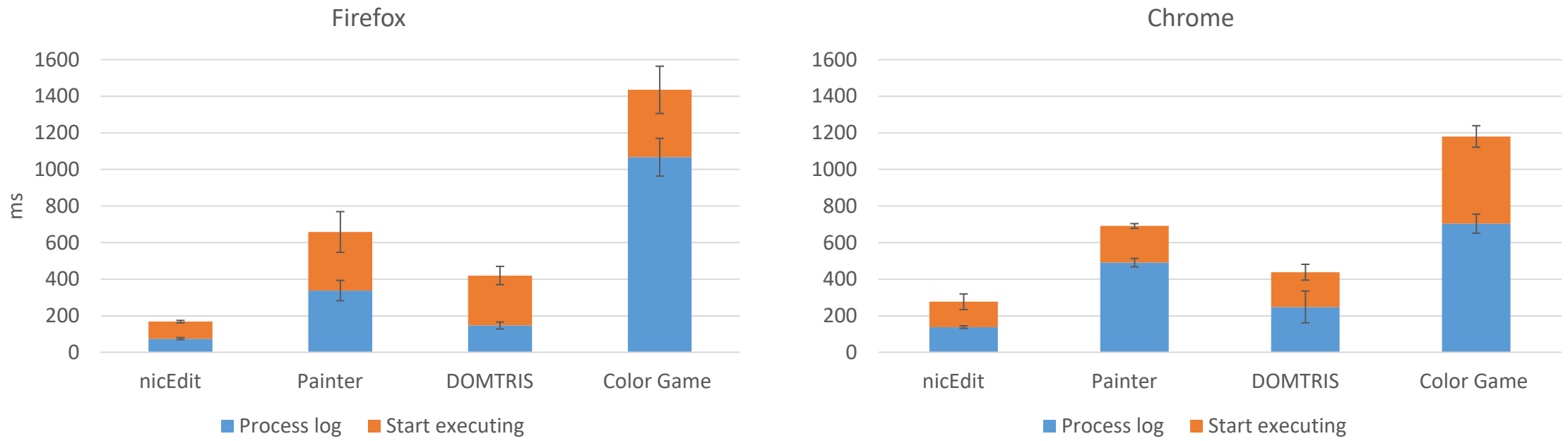


Not a lot of storage needed (max 50MB for long interaction)

SINATRA phases

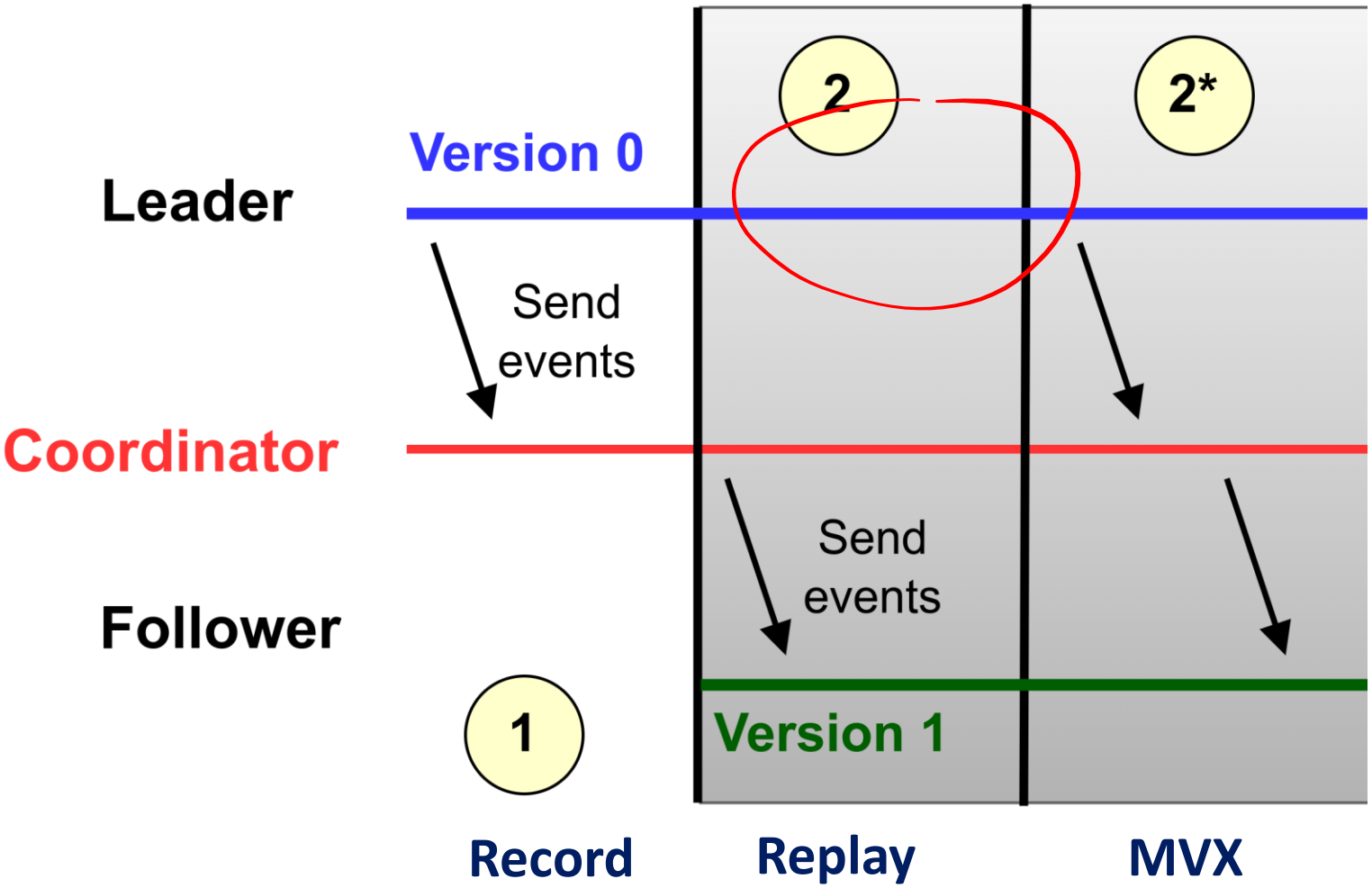


Replaying Javascript Events

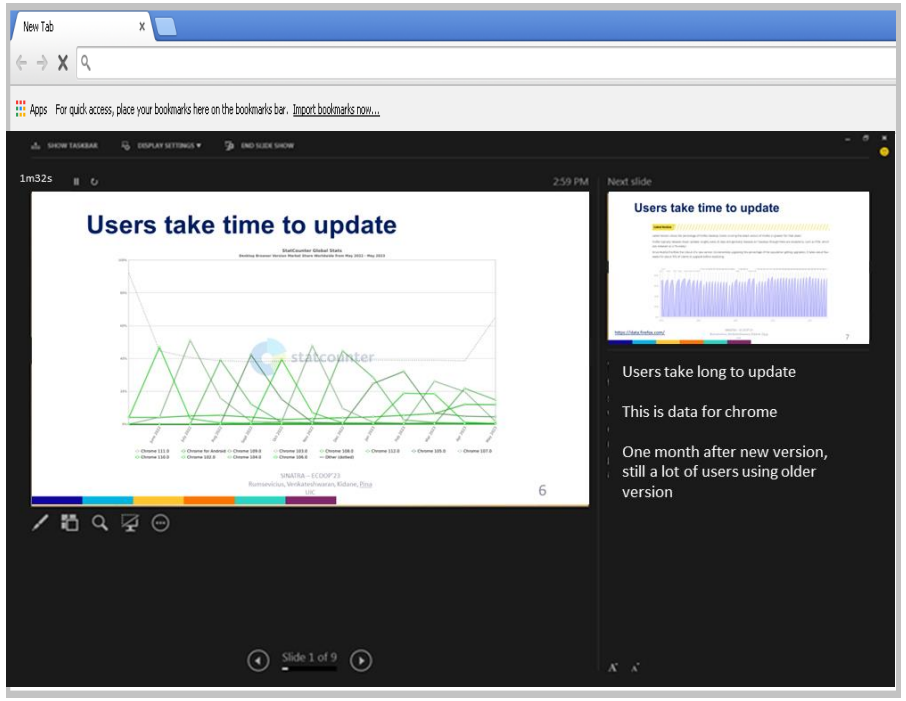


New browser version “catches up” in less than 2sec

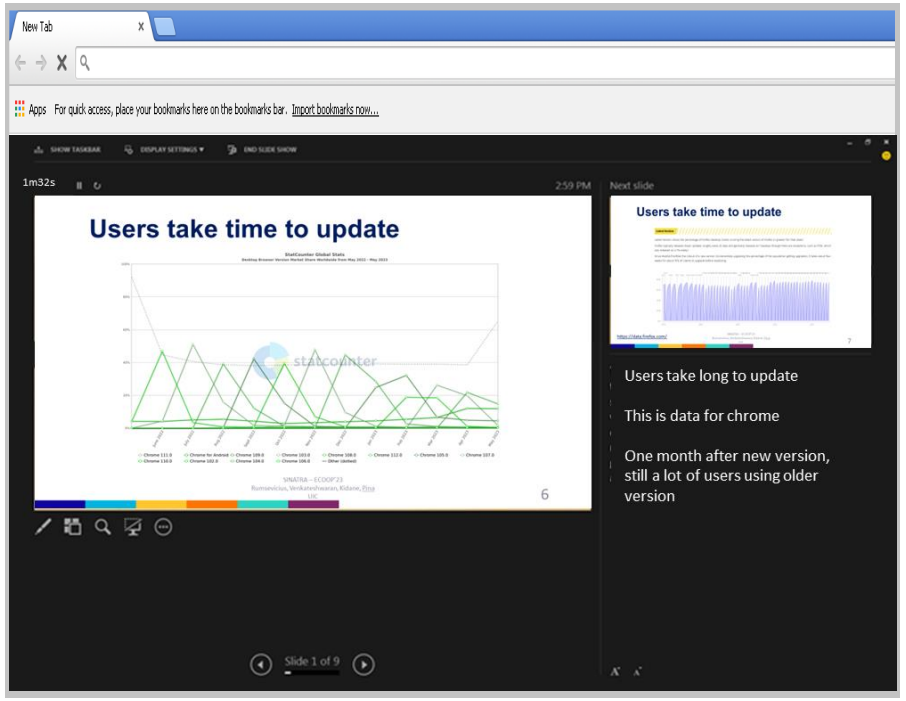
SINATRA phases



SINATRA

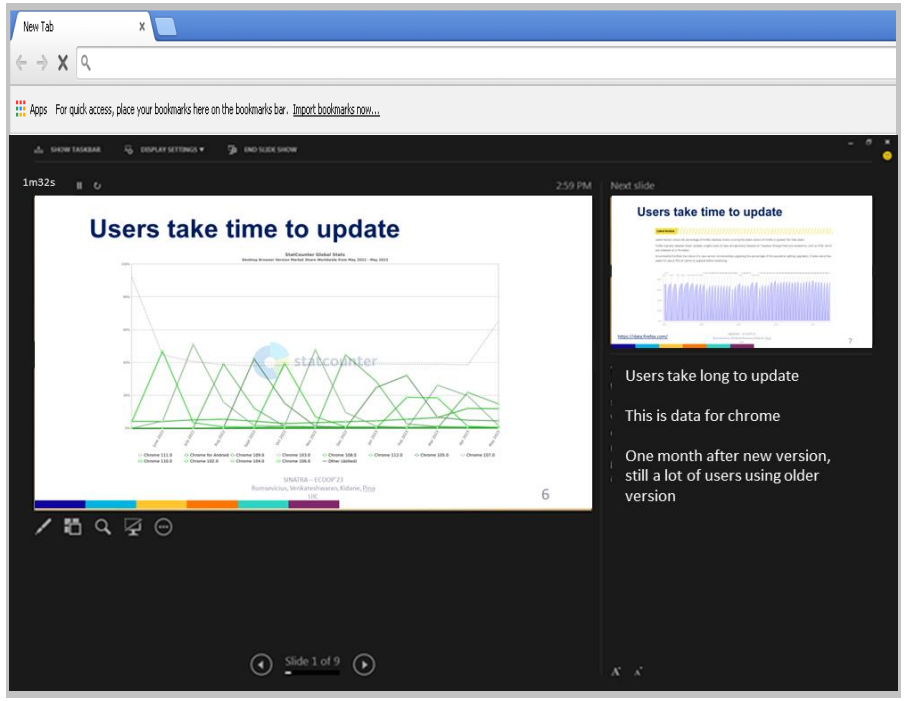


Old browser version

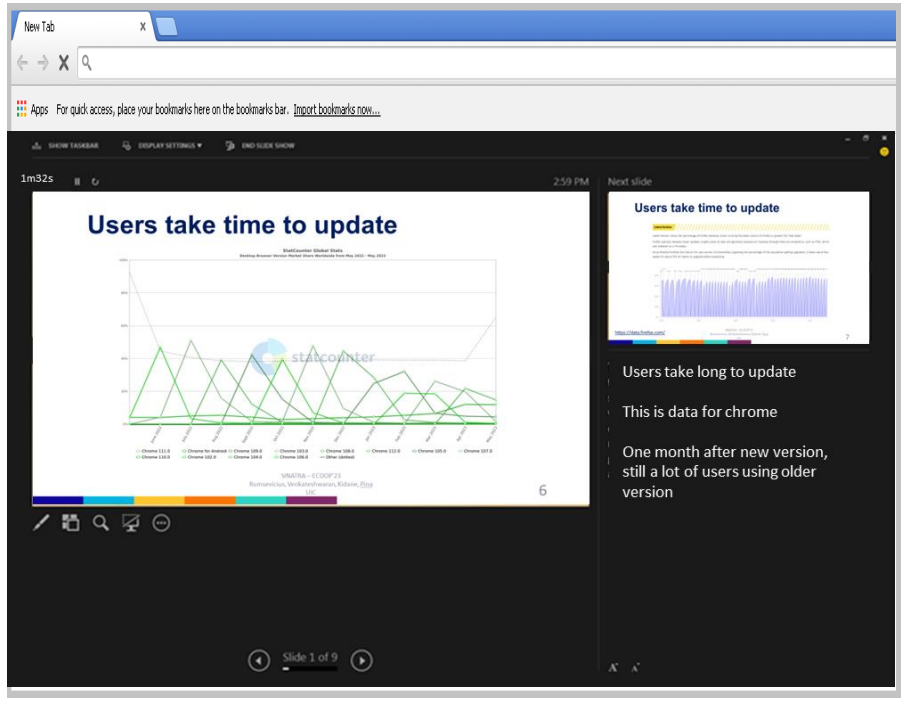


1. Launch new browser version
2. Open same page, get same contents
3. Replay all events from old browser
4. Swap roles

SINATRA

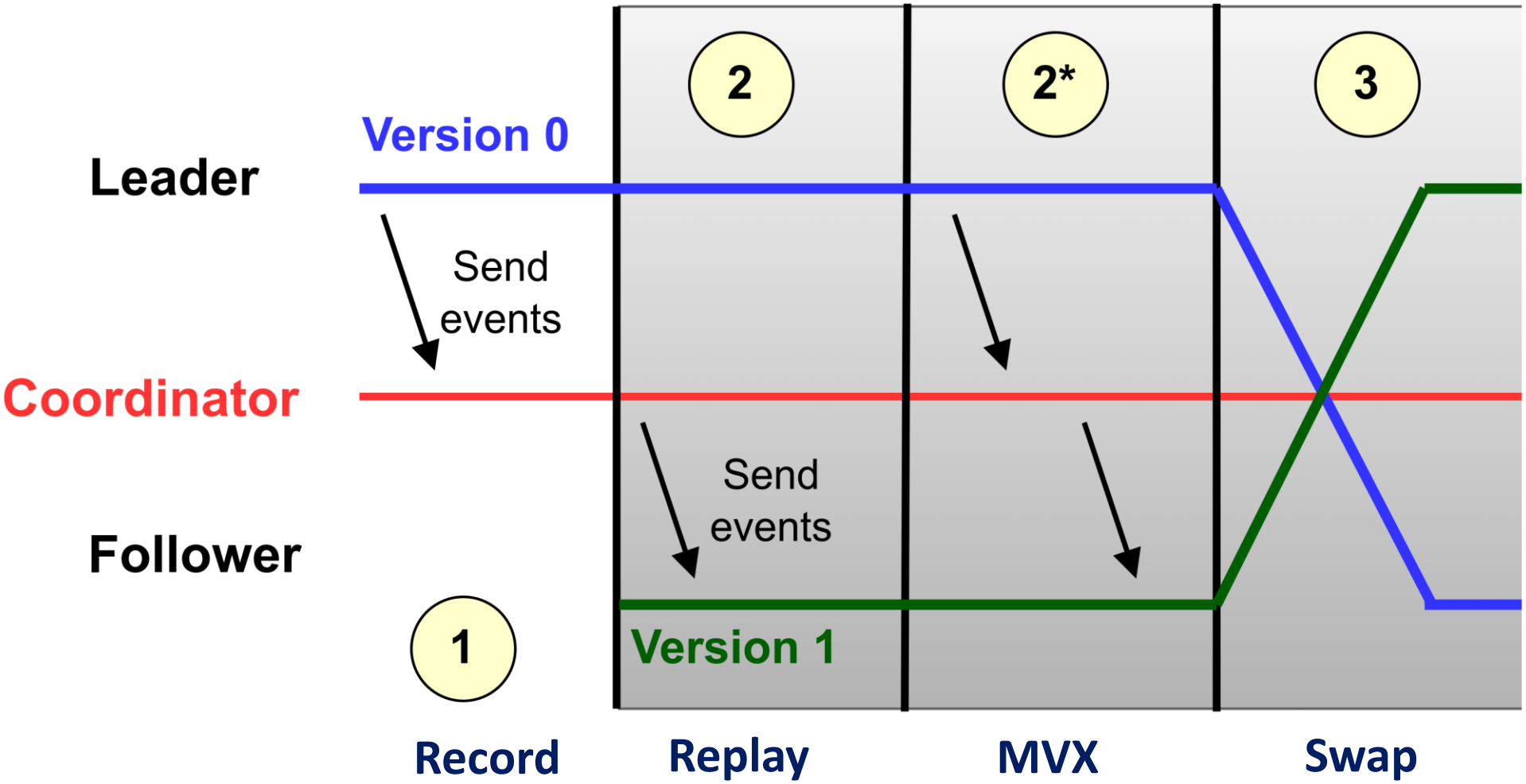


1. Launch new browser version
2. Open same page, get same contents
3. Replay all events from old browser
4. Swap roles

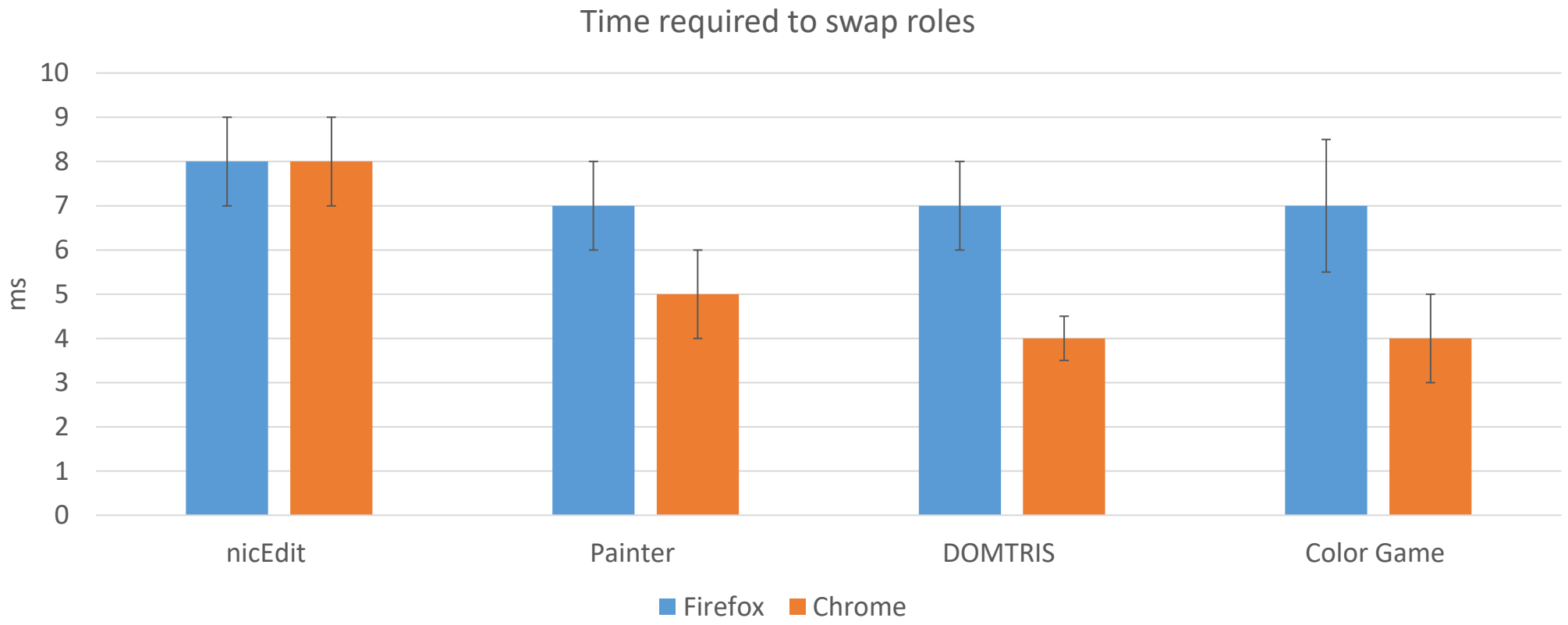


Old browser version

SINATRA phases

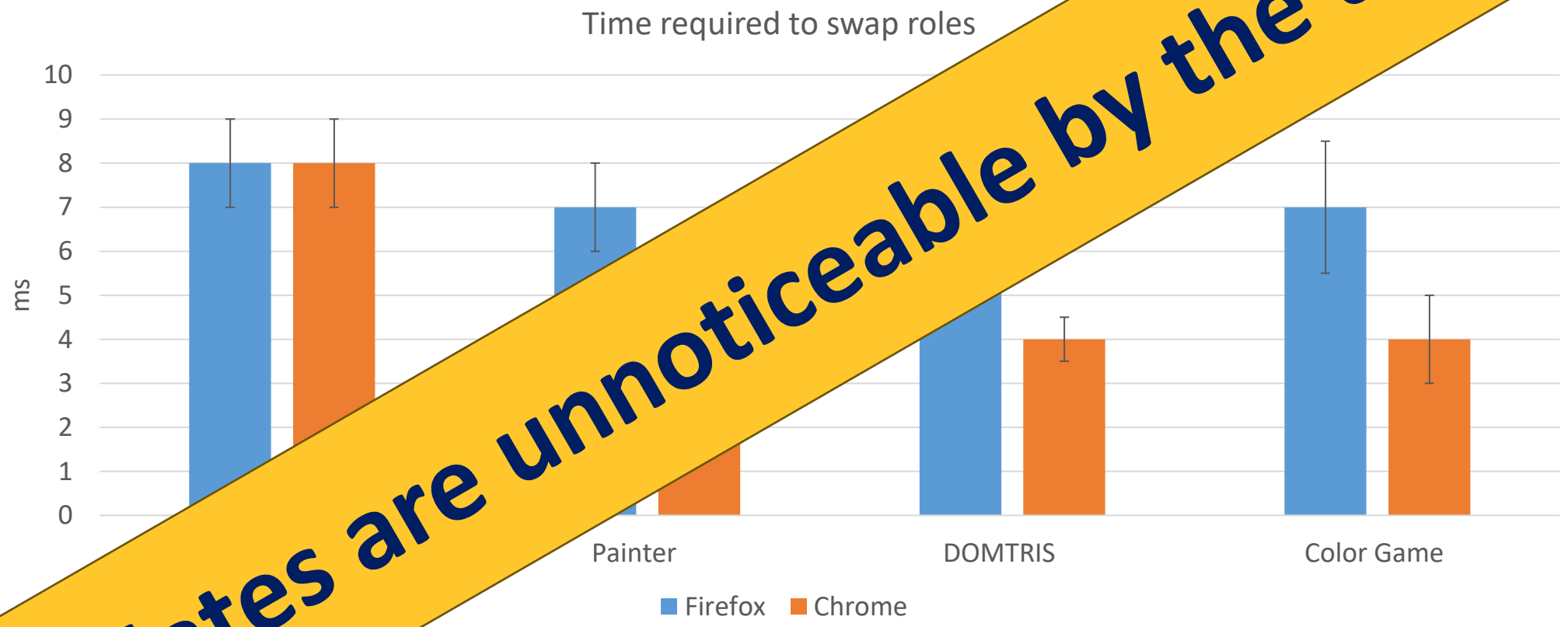


Pause introduced by SINATRA update



Users experience less than 10ms pause due to update

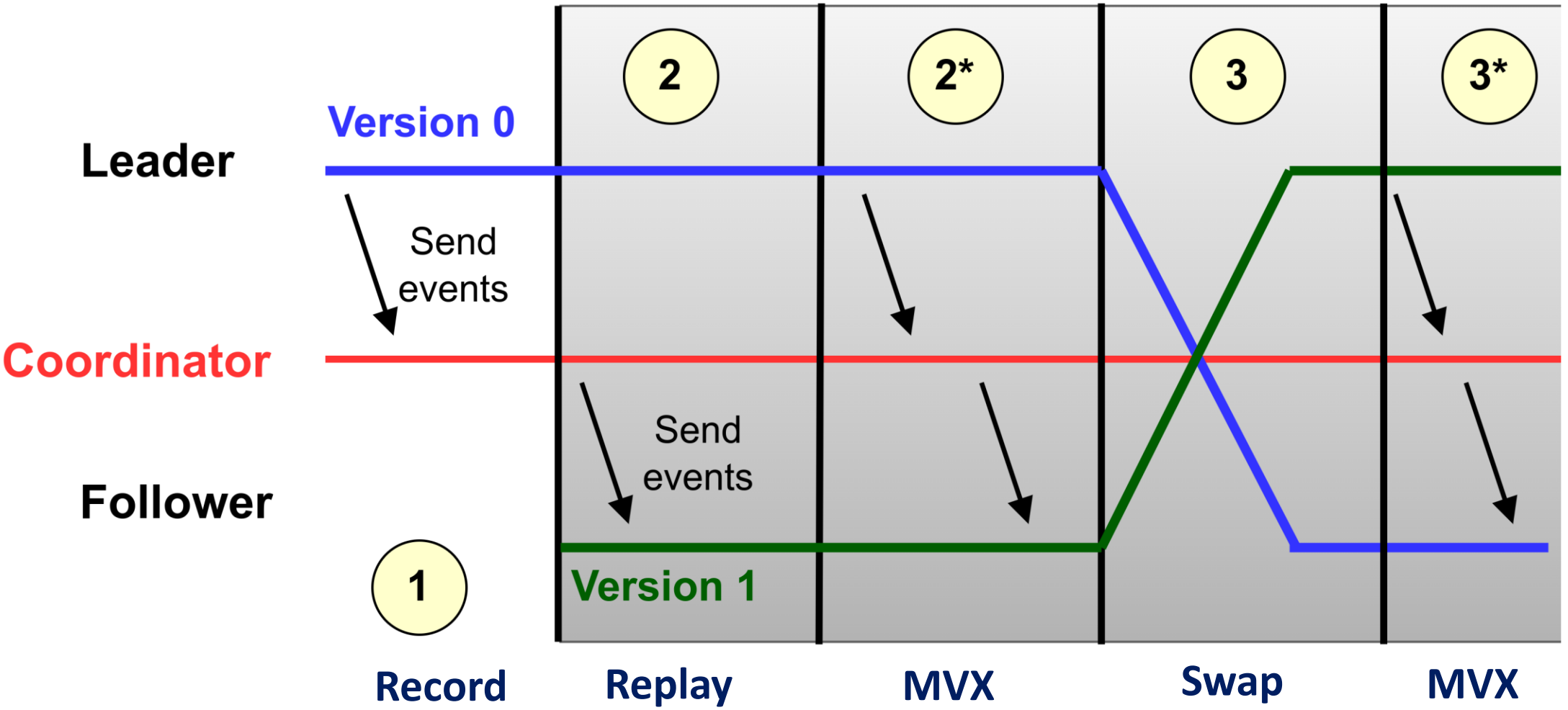
Pause introduced by SINATRA u



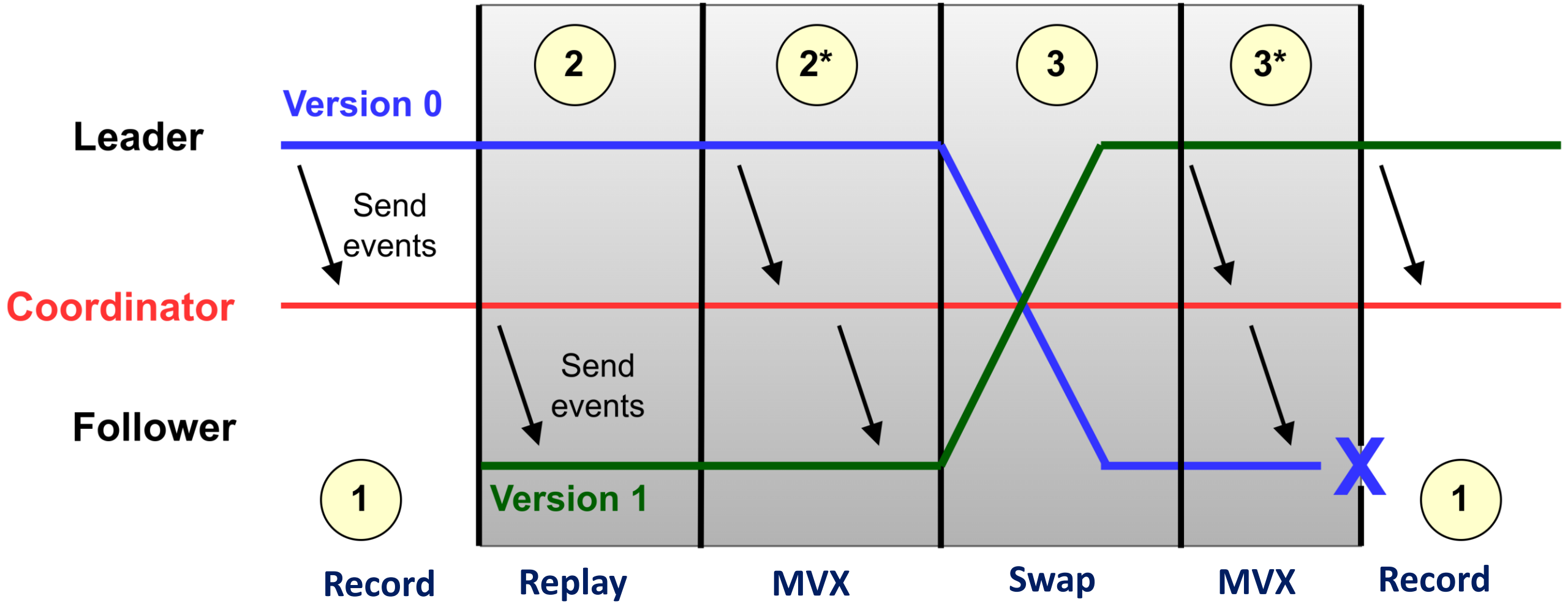
Updates are unnoticeable by the user

Users experience less than 10ms pause due to update

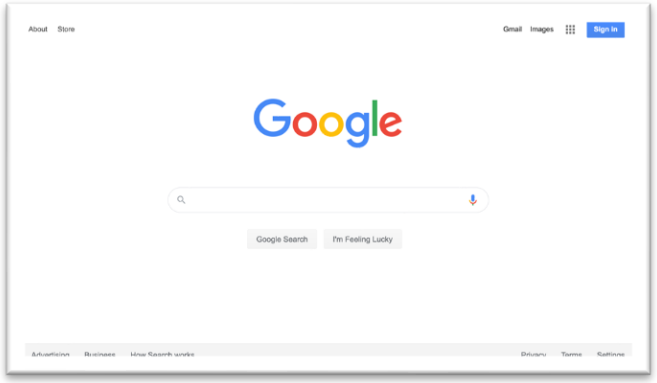
SINATRA phases



SINATRA phases



Realistic pages



JS Action

Realistic pages

- Captured 14h of traffic for high-volume twitter account
  - https://twitter.com/TotalTrafficCHI
  - Log sizes under 36MB

SINATRA is available!

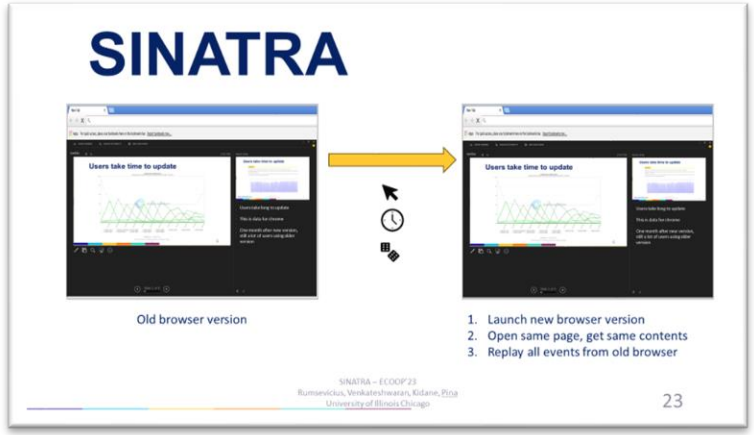
zenodo

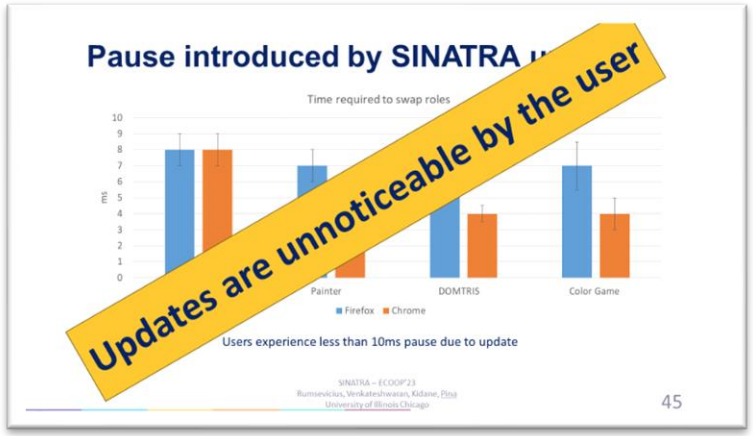
GitHub

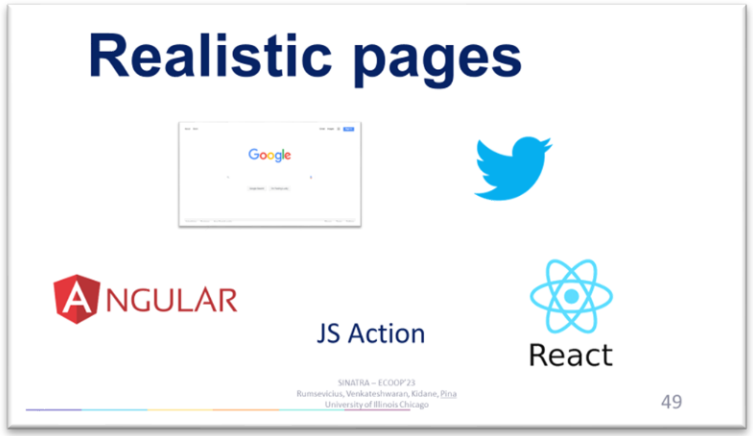
Conclusion

github.com/bitslab/sinatra

Fork me on Github



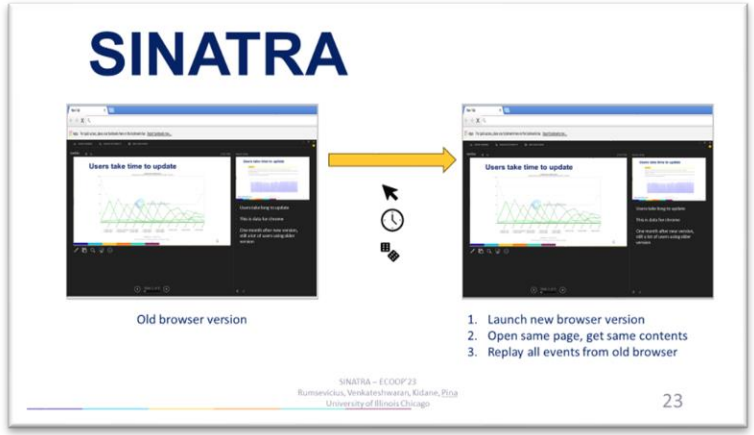


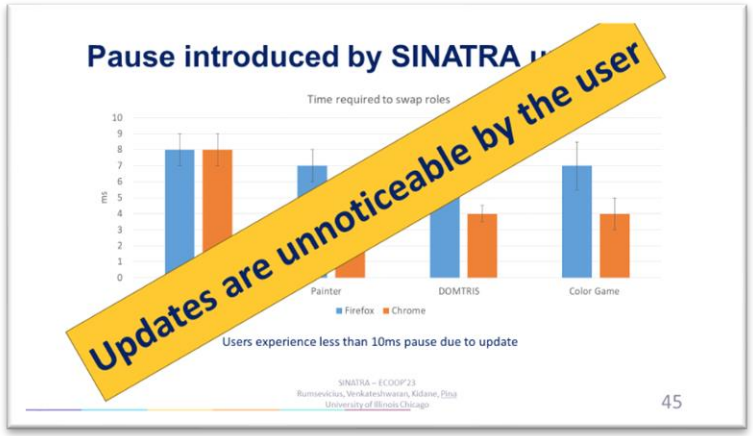


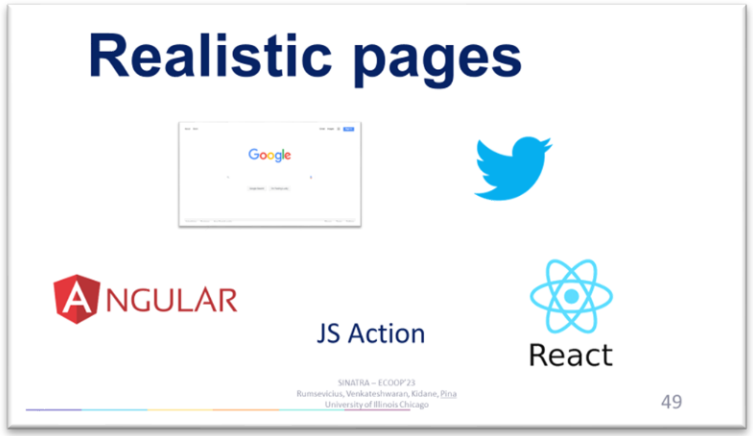
Conclusion

github.com/bitslab/sinatra

Fork me on Github







Thank you!

Questions?

CCF-2227183