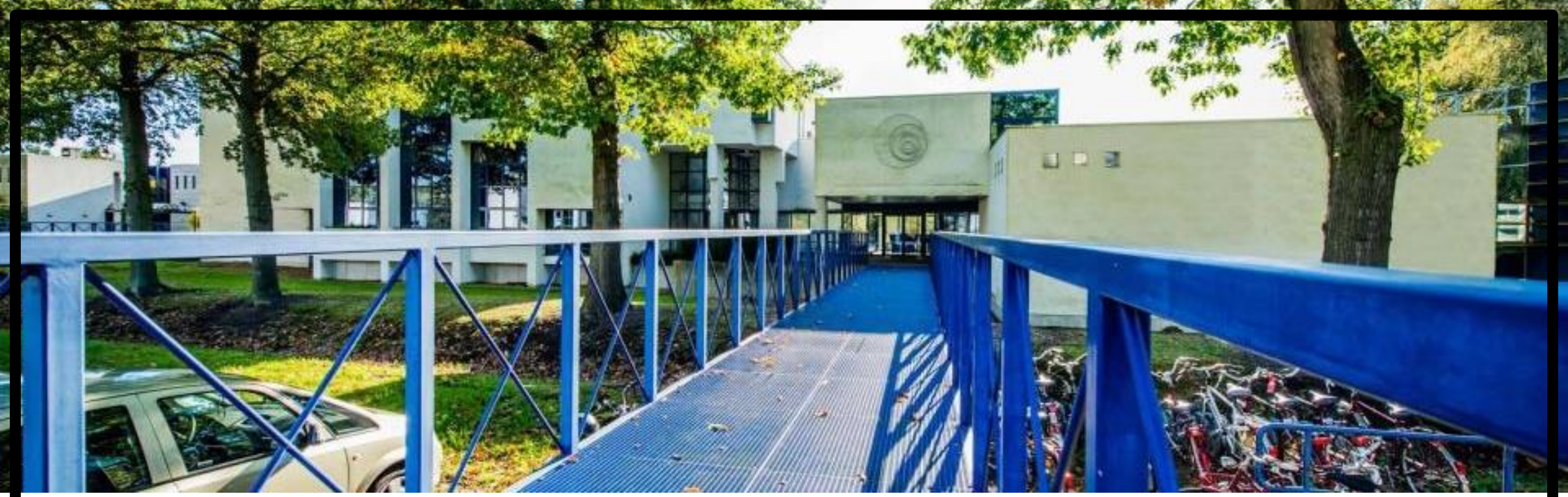


HTTP/2 Prioritization

and its Impact on Web Performance

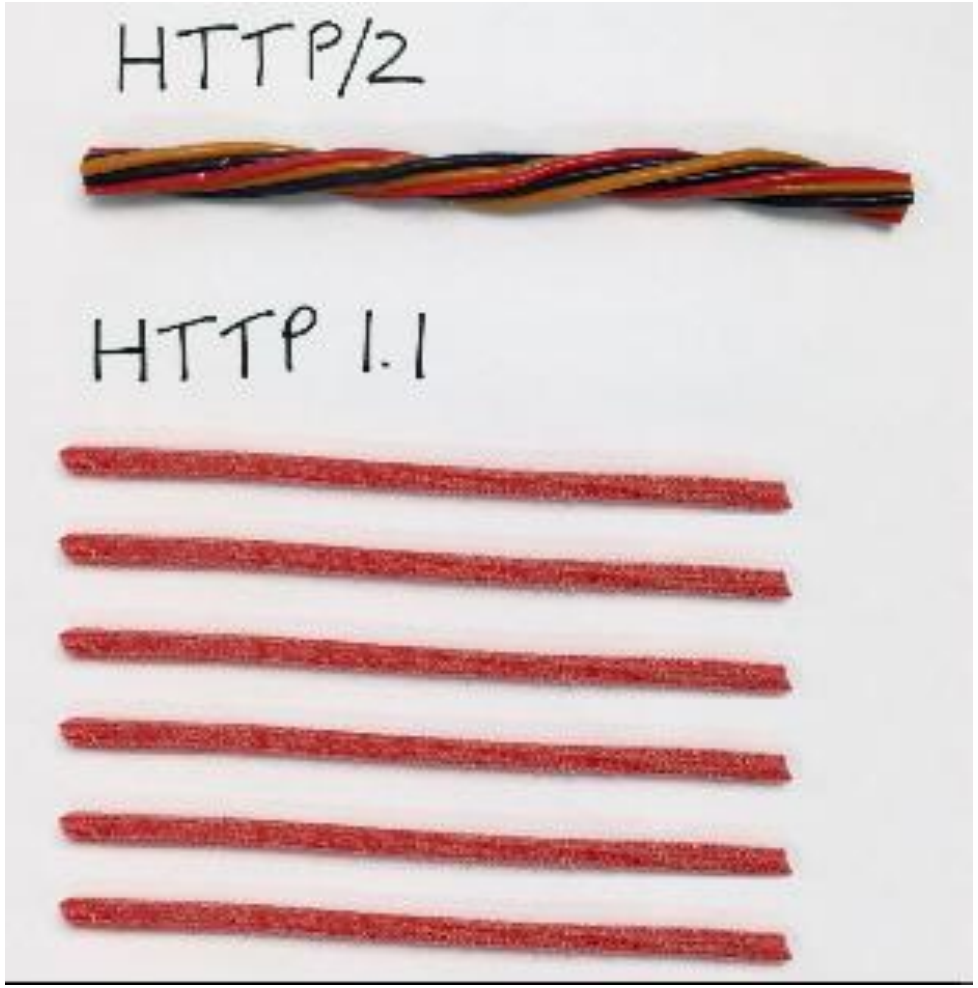
dr. Maarten Wijnants, ***Robin Marx***
Prof. dr. Peter Quax, Prof. dr. Wim Lamotte
https://speeder.edm.uhasselt.be

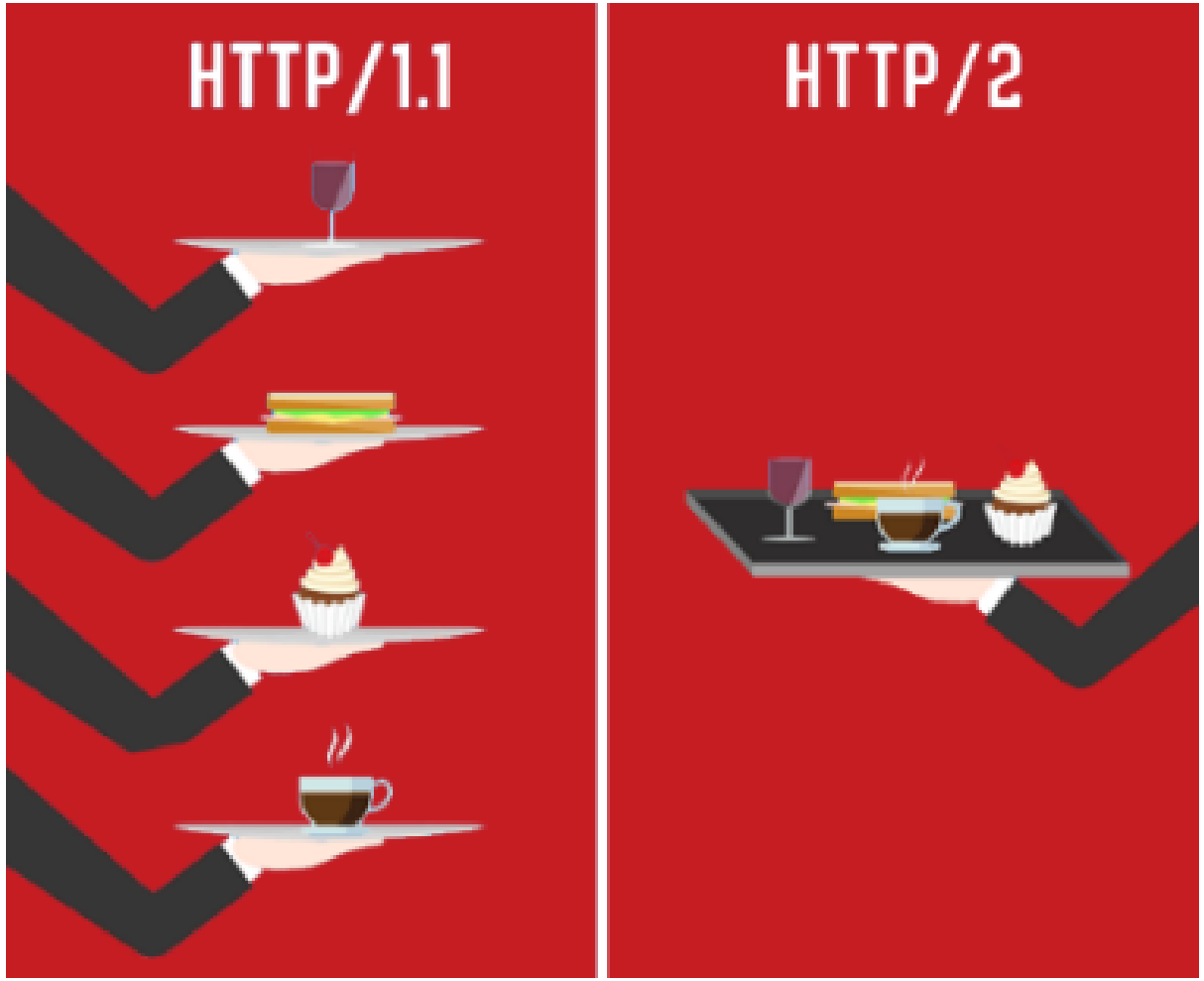
UHASSELT EDM

HTTP/1.1 : 6 – 17 Parallel TCP connections

http://www.splicemarketing.co.uk/_blog/Splice_Blog/post/the-arrival-of-http2/

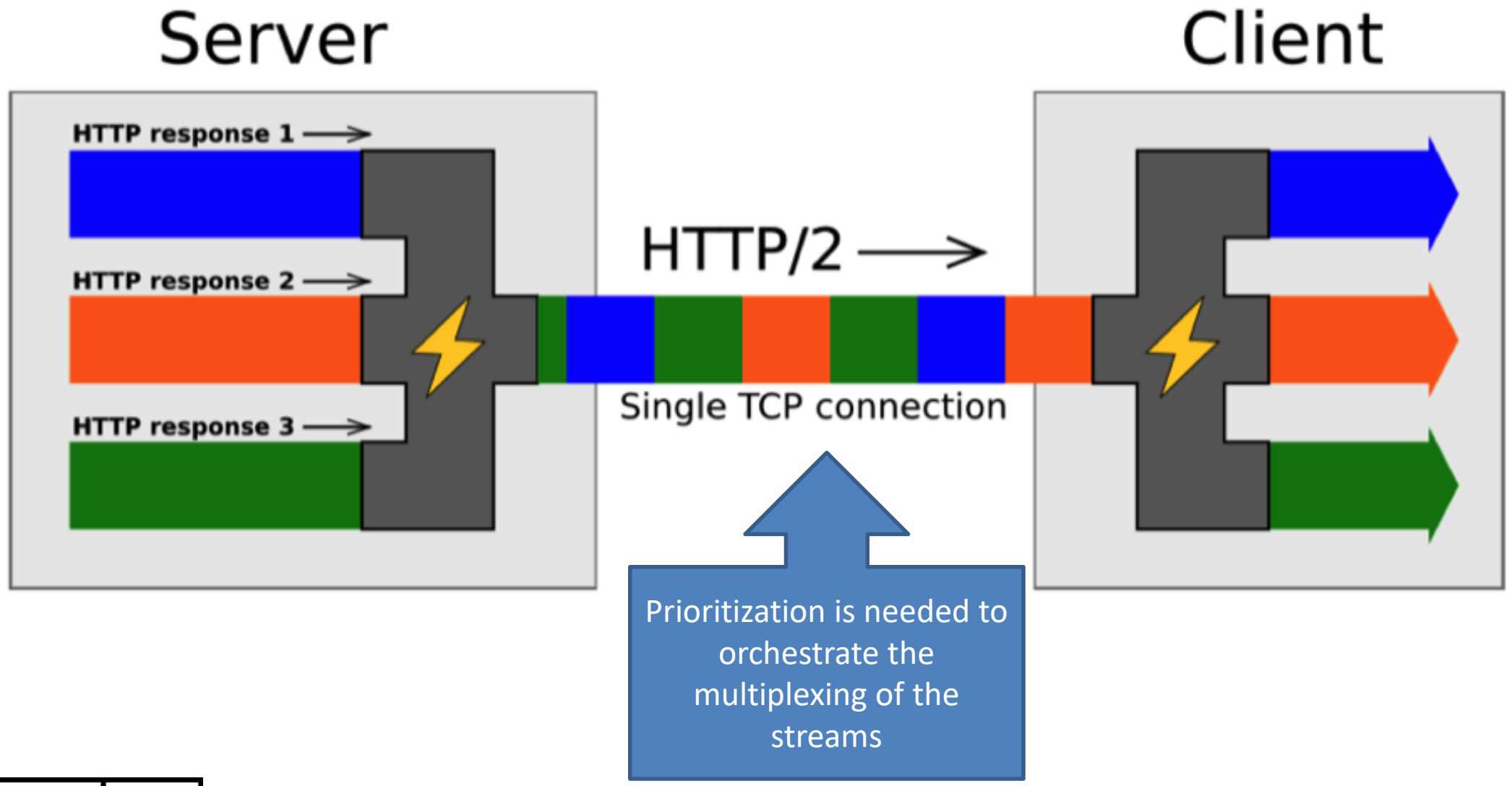
HTTP/2 : Single TCP connection + multiplexing





http://www.splicemarketing.co.uk/_blog/Splice_Blog/post/the-arrival-of-http2/

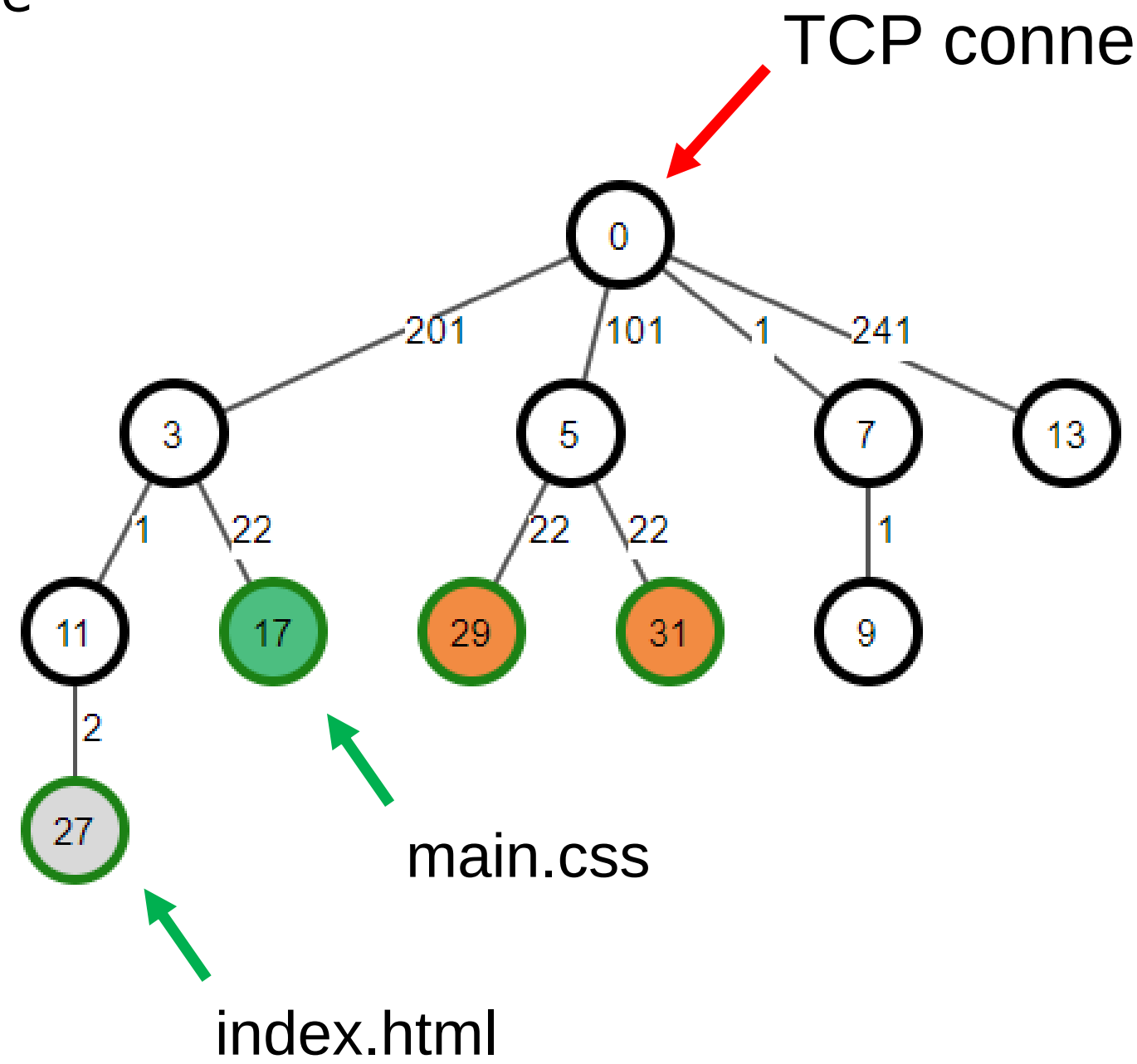
HTTP/2 : Single TCP connection + multiplexing



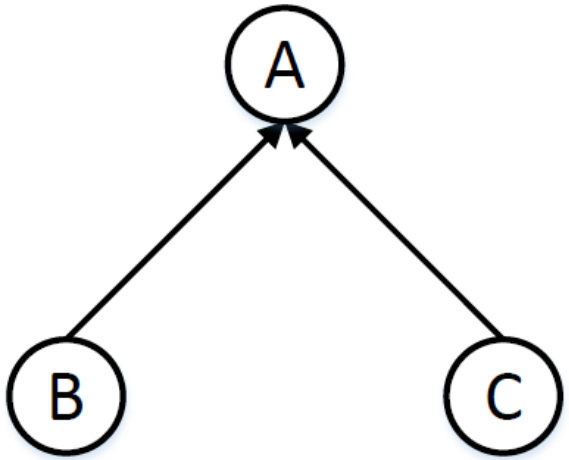
https://blog.sessionstack.com/how-javascript-works-deep-dive-into-websockets-and-http-2-with-sse-how-to-pick-the-right-path-584e6b8e3bf7

HTTP/2 Prioritization
Dependency Tree Mechanism

Dependency tree



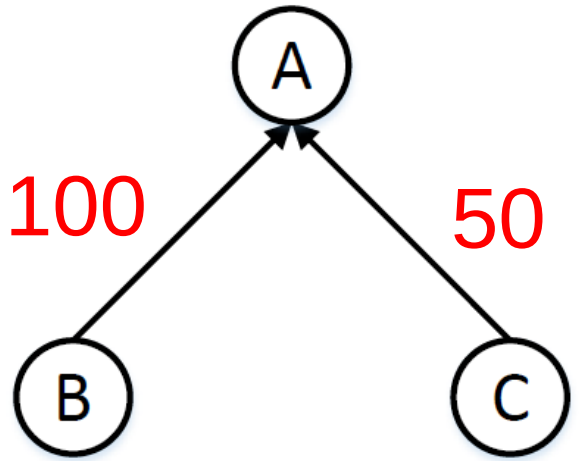
Parent / Child dependencies



Parent first, then children

Siblings share bandwidth

Resource weights

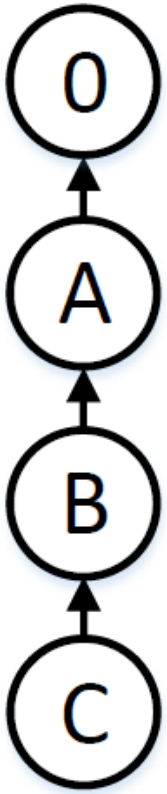


Parent first, then children

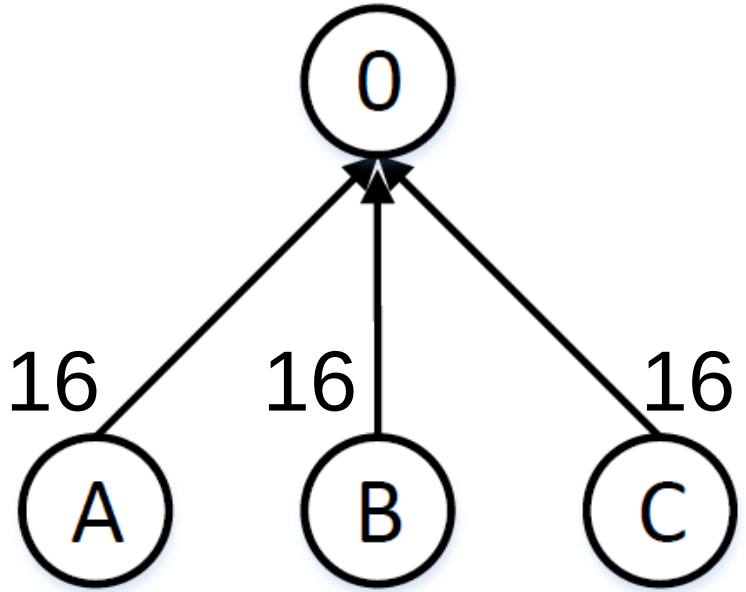
Siblings share bandwidth

B gets 66%

Two naïve approaches : FCFS vs RR



First-Come
First-Served

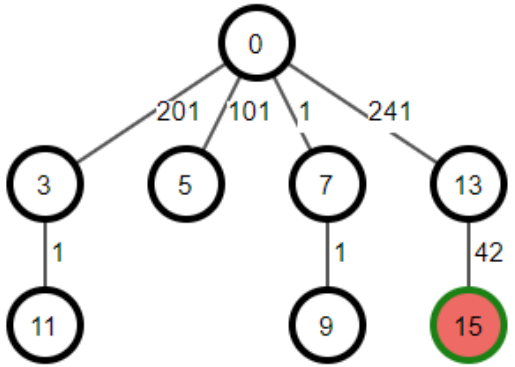


Round Robin

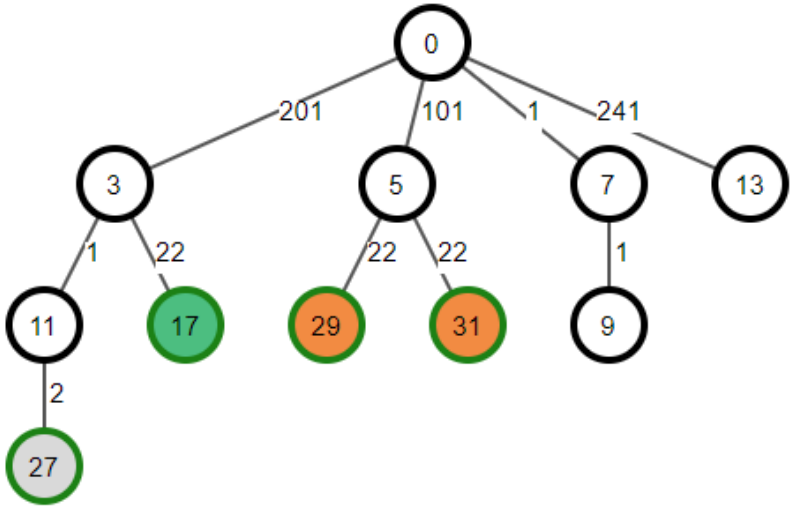
Real-world Browser Behavior

Real Browser Results Visualization with H2Vis

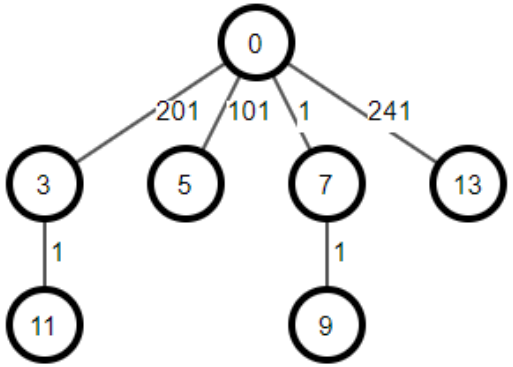
ID: 3	T: PRIORITY	S: CLIENT	P: 0	E: F	W: 201
ID: 5	T: PRIORITY	S: CLIENT	P: 0	E: F	W: 101
ID: 7	T: PRIORITY	S: CLIENT	P: 0	E: F	W: 1
ID: 9	T: PRIORITY	S: CLIENT	P: 7	E: F	W: 1
ID: 11	T: PRIORITY	S: CLIENT	P: 3	E: F	W: 1
ID: 13	T: PRIORITY	S: CLIENT	P: 0	E: F	W: 241
ID: 15	T: HEADERS	S: CLIENT	P: 13	W: 42	



ID: 15	T: DATA	S: SERVER	ES: T	
ID: 17	T: HEADERS	S: CLIENT	P: 3	W: 22
ID: 19	T: HEADERS	S: CLIENT	P: 11	W: 2
ID: 19	T: RST_STREAM	S: CLIENT		
ID: 21	T: HEADERS	S: CLIENT	P: 11	W: 2
ID: 21	T: RST_STREAM	S: CLIENT		
ID: 23	T: HEADERS	S: CLIENT	P: 11	W: 2
ID: 23	T: RST_STREAM	S: CLIENT		
ID: 25	T: HEADERS	S: CLIENT	P: 11	W: 2
ID: 25	T: RST_STREAM	S: CLIENT		
ID: 27	T: HEADERS	S: CLIENT	P: 11	W: 2
ID: 29	T: HEADERS	S: CLIENT	P: 5	W: 22
ID: 31	T: HEADERS	S: CLIENT	P: 5	W: 22



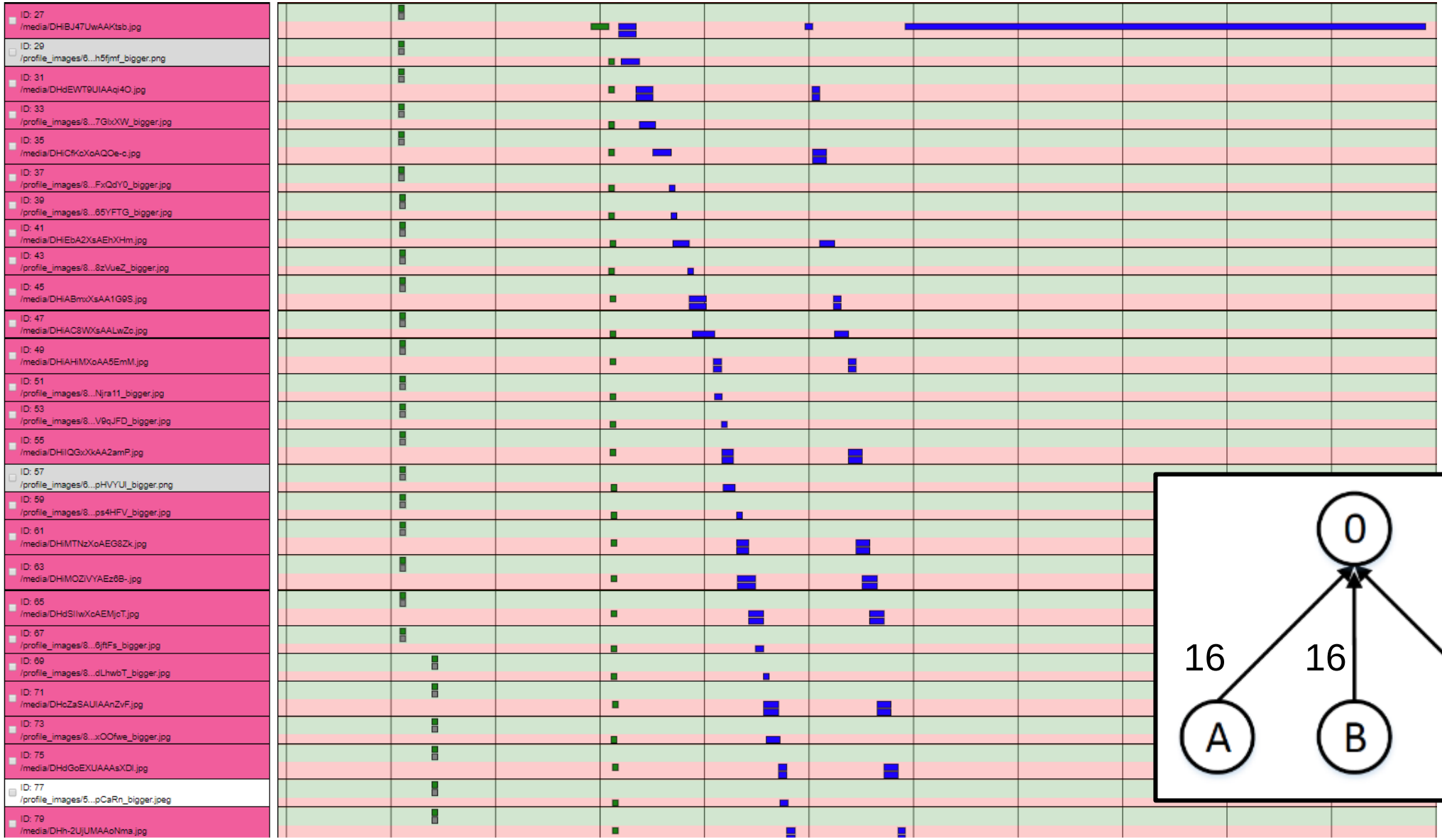
ID: 17	T: DATA	S: SERVER	ES: T
ID: 19	T: DATA	S: SERVER	ES: T
ID: 21	T: DATA	S: SERVER	ES: T
ID: 23	T: DATA	S: SERVER	ES: T
ID: 25	T: DATA	S: SERVER	ES: T
ID: 27	T: DATA	S: SERVER	ES: T
ID: 29	T: DATA	S: SERVER	ES: T
ID: 31	T: DATA	S: SERVER	ES: T



time

https://github.com/rmarx/h2vis

Real Browser Results Visualization with H2Vis

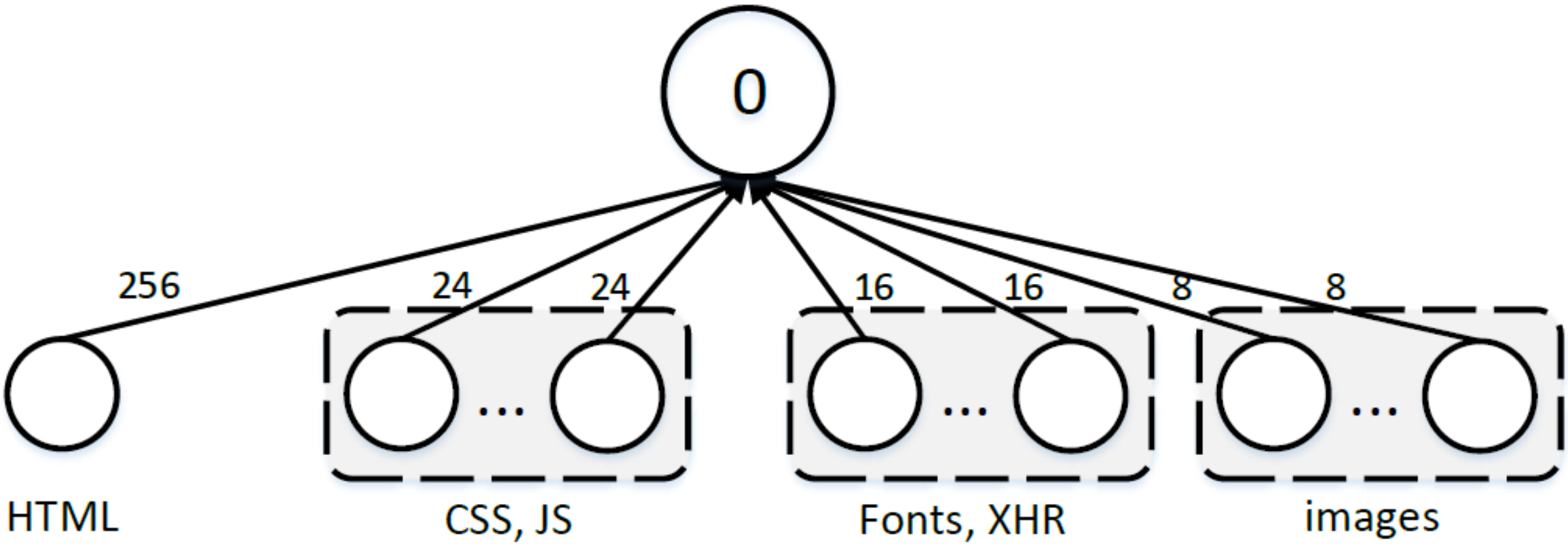


https://github.com/rmarx/h2vis

Real Browser Results

Priority approach	Browsers
Round Robin	Internet Explorer, Edge, Opera Mini   
Weighted Round Robin	Safari, UC Browser  
Dynamic First-Come First-Served	Chrome, Opera, Brave, Dolphin    
Complex Tree	Firefox 

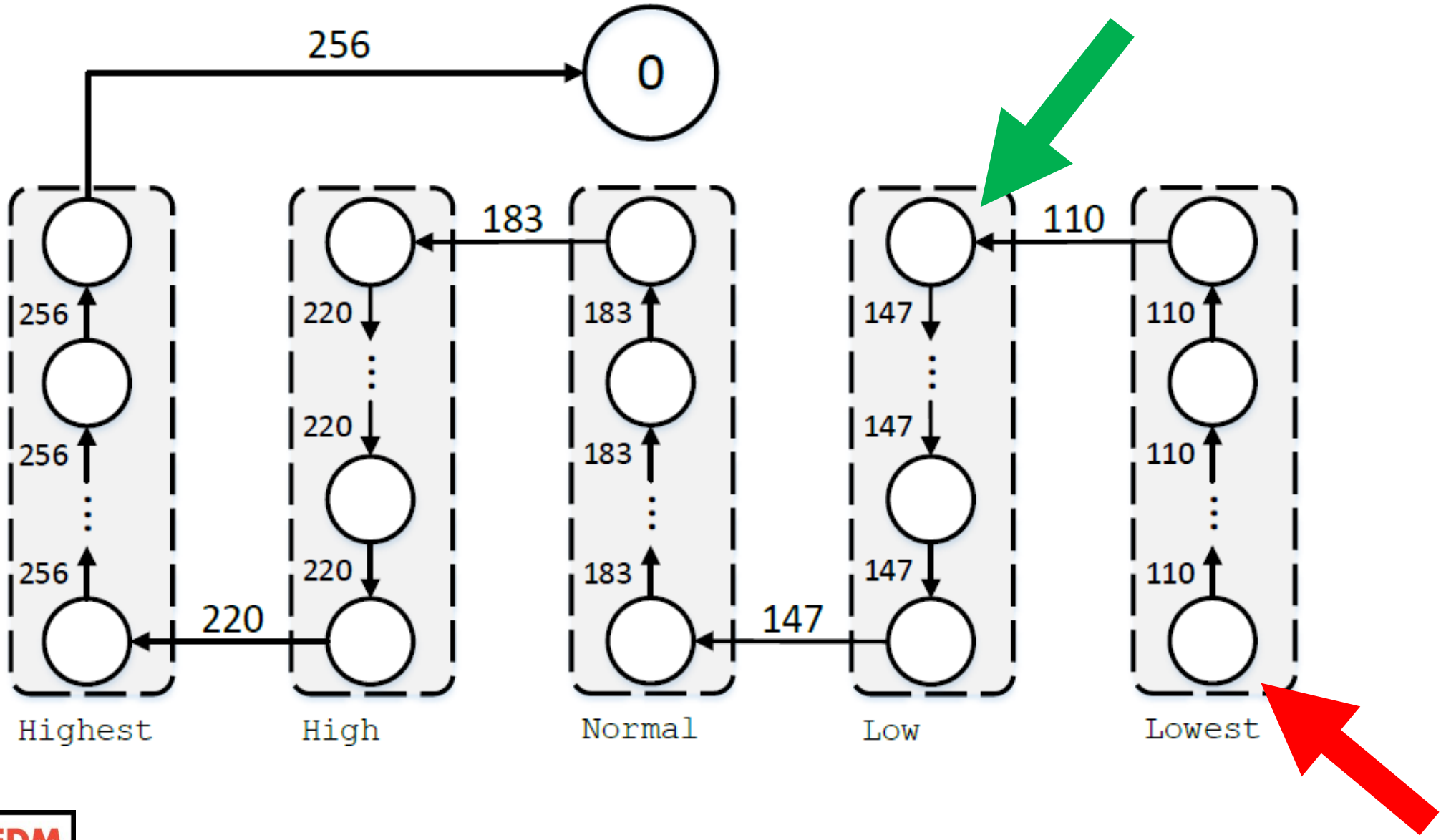
Weighted RR



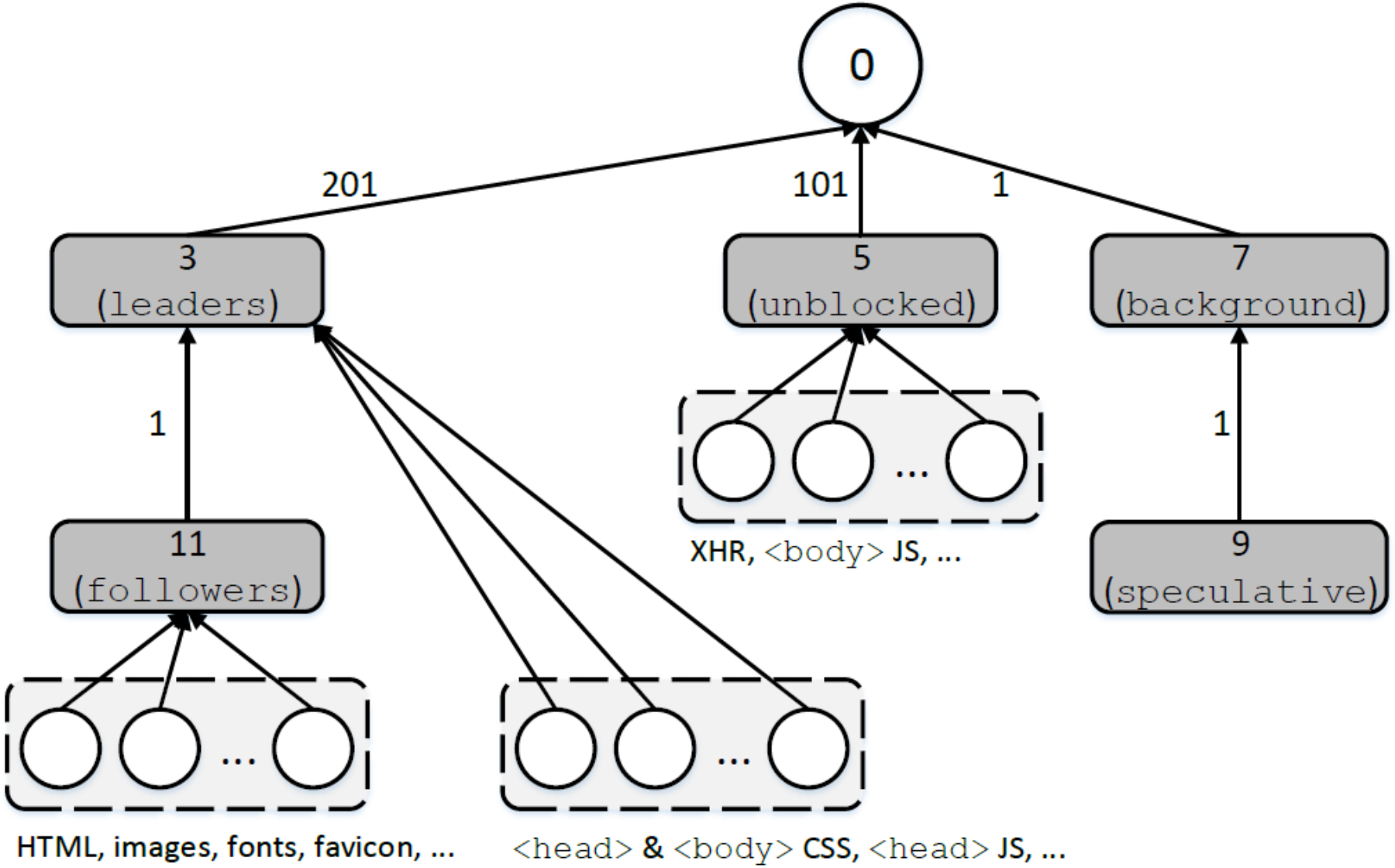
Real Browser heuristics

Bucket	Content Types	Content Types	Content Types
Lowest	pushed assets (initially)	pushed assets	pushed assets
Low	images, async and defer JS	images	
Normal	JS declared after first image	fonts, XHR	images
High	JS declared prior to first image, XHR	JS, CSS	HTML, JS, CSS, XHR
Highest	HTML, CSS, fonts	HTML	fonts

Dynamic/intra-bucket FCFS



Complex Tree



Experimental Evaluation

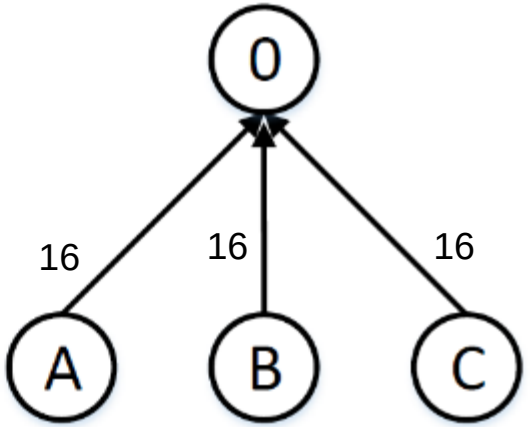
Which algorithm leads to the fastest page load times?

8 Prioritization Algorithms

- Default browser behaviour
- HTTPS/1.1
- RR
- FCFS

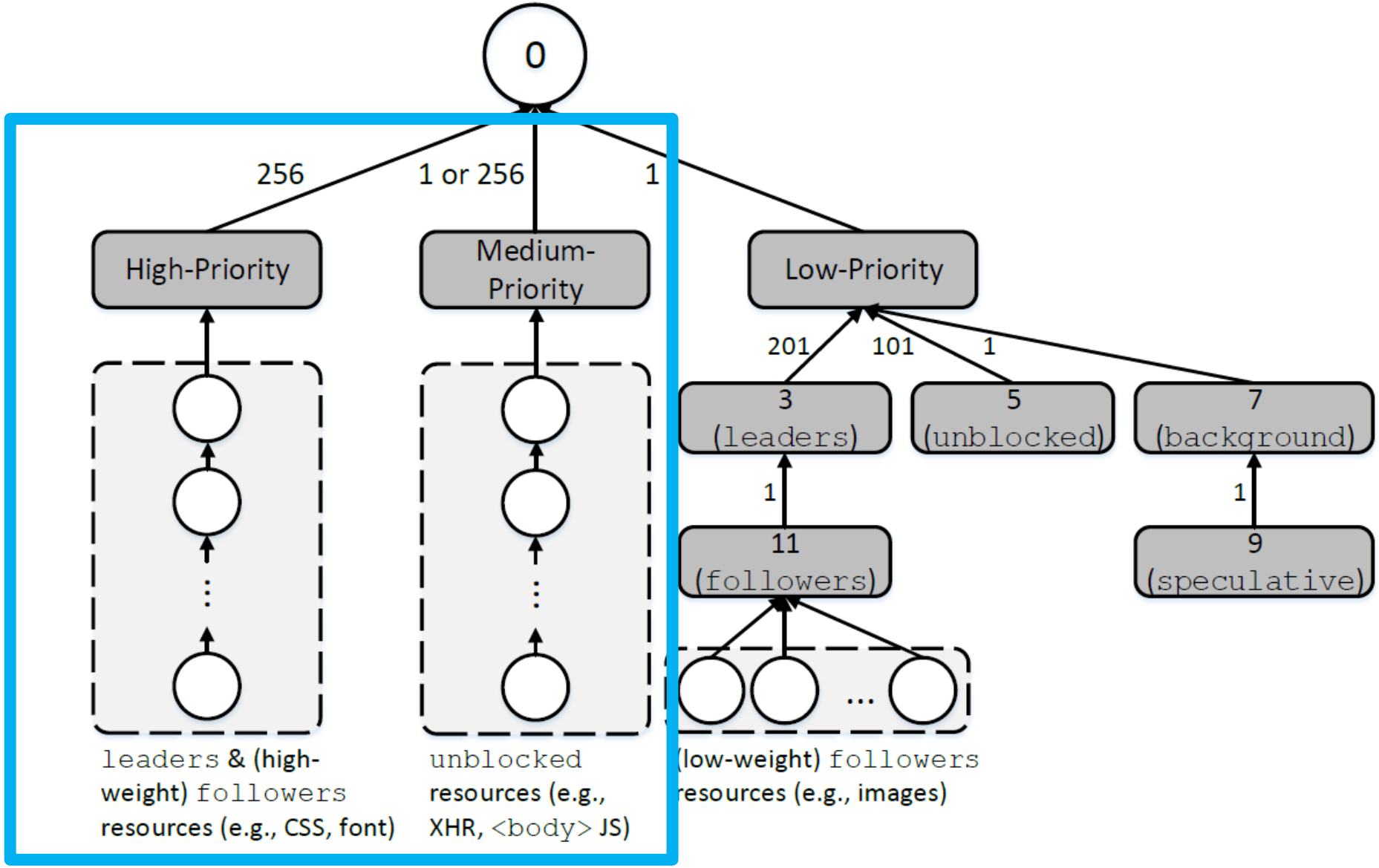
- Serial / Serial+
- Parallel / Parallel+

First-Come
First-Served

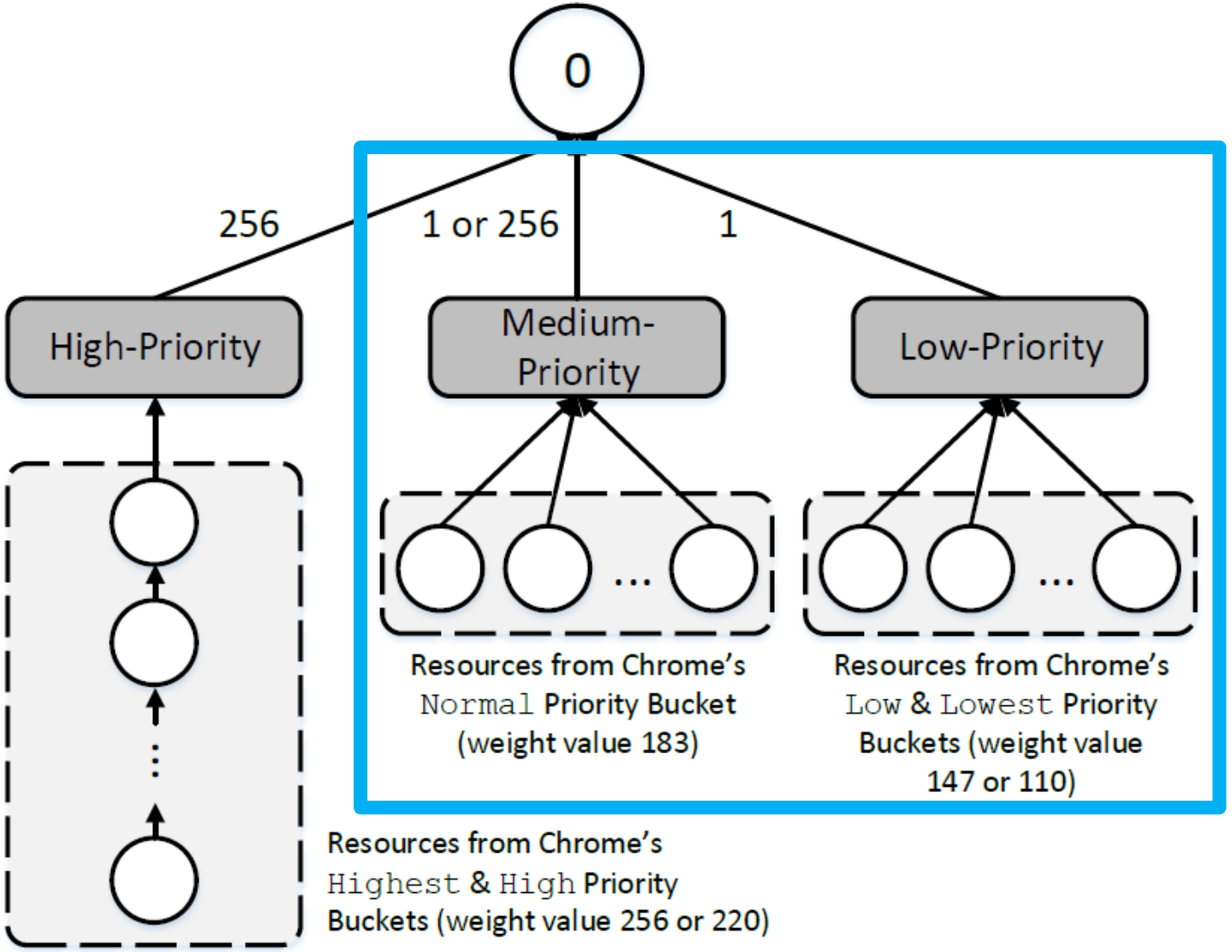


Round Robin

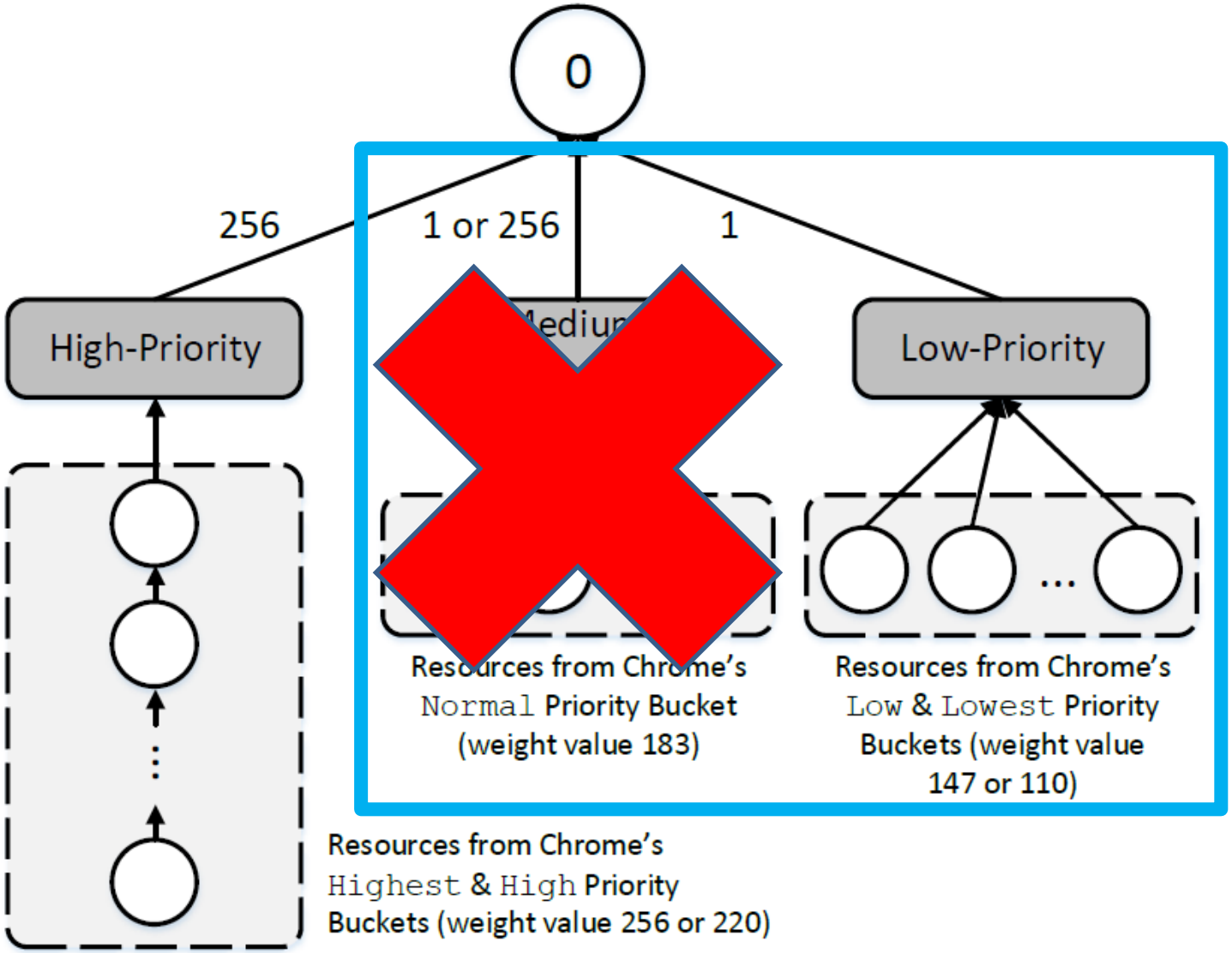
Algorithms : Serial / Serial+



Algorithms : Parallel / Parallel+



Algorithms : Parallel / Parallel+



Speeder Framework

- Full factorial evaluation
  - Modified H2O webserver overrides browser priorities
  - 40 websites
  - 2 different performance metrics (loadEventEnd and SpeedIndex)
  - Chrome and Firefox (driven by WebpageTest)
  - 5 advanced network emulations
    - Traces from actual cellular networks
  - 20 runs of each permutation

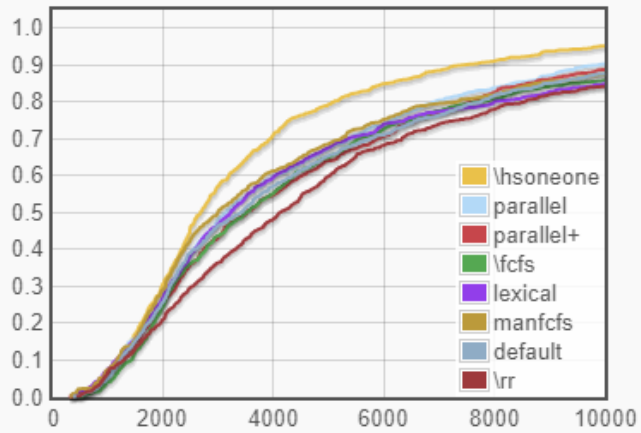
- Mann-Whitney U test
  - $p < 0.005$
  - Measure median deviation from “Default” browser behaviour

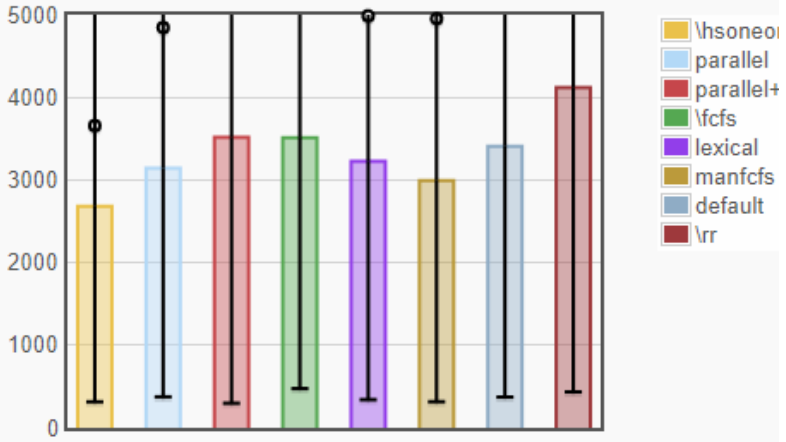
https://speeder.edm.uhasselt.be
https://docs.google.com/document/d/1oLhNg1skaWD4_DtaoCxdSRN5erEXrH-KnLrMwEpOtFY

Processing + Visualization

Chrome
SpeedIndex
Poor Cellular Network

Priority algo	#websites/ #values	No change	+any	≥ +5%	≥ +10%	≥ +25%	-any	≥ -5%	≥ -10%	≥ -25%
\hsoneone	41 / 820	33	8	8	8	7	0	0	0	0
parallel	39 / 776	34	3	3	3	3	2	2	2	2
parallel+	39 / 774	33	3	3	3	3	3	3	3	3
\fcfs	41 / 819	38	1	1	1	1	2	2	2	2
lexical	39 / 780	31	4	4	4	4	4	4	4	3
manfcfs	37 / 740	31	3	3	3	3	3	3	3	3
\rr	41 / 819	32	1	1	1	1	8	8	8	8

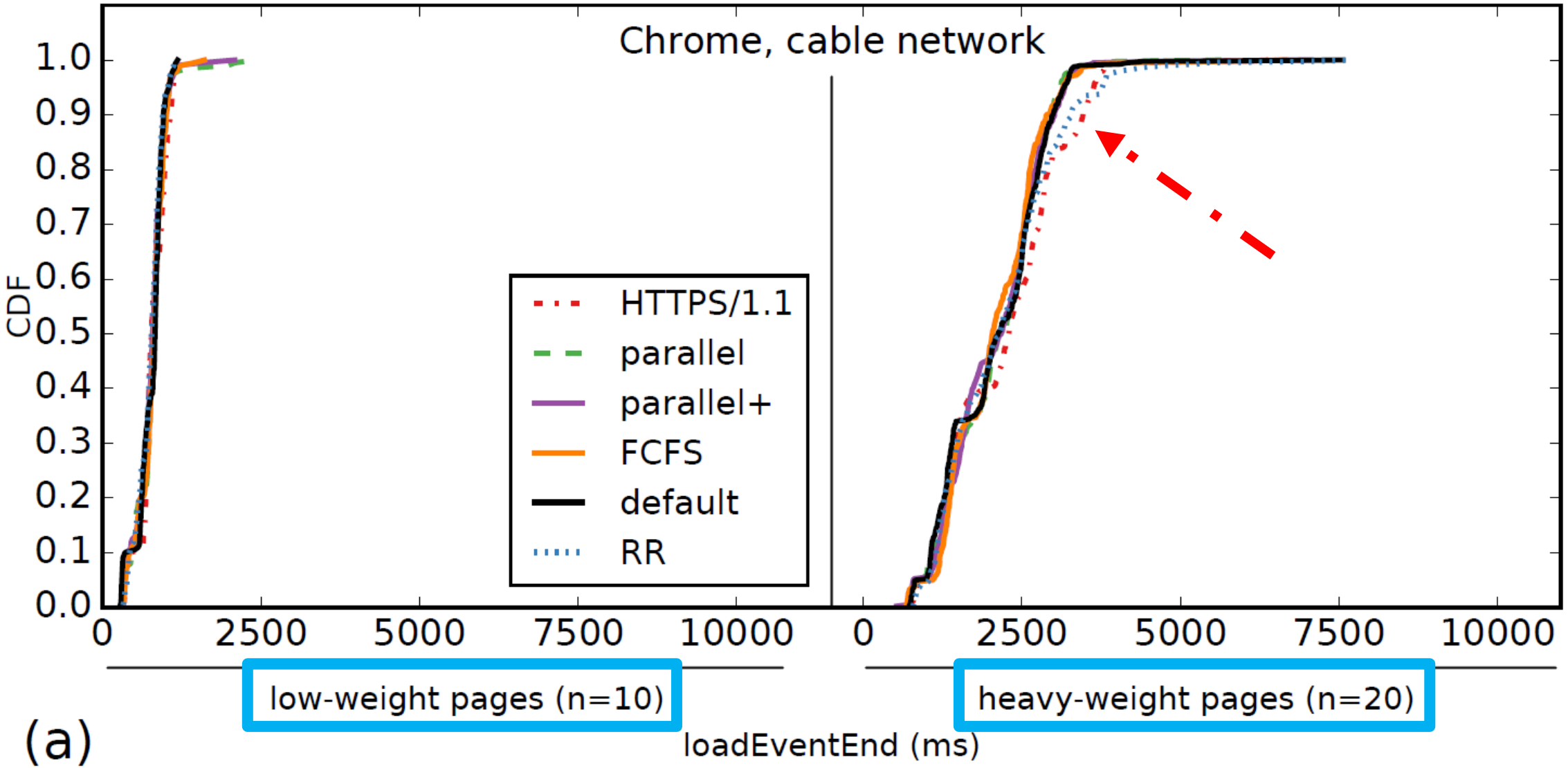




Tabular Visualization

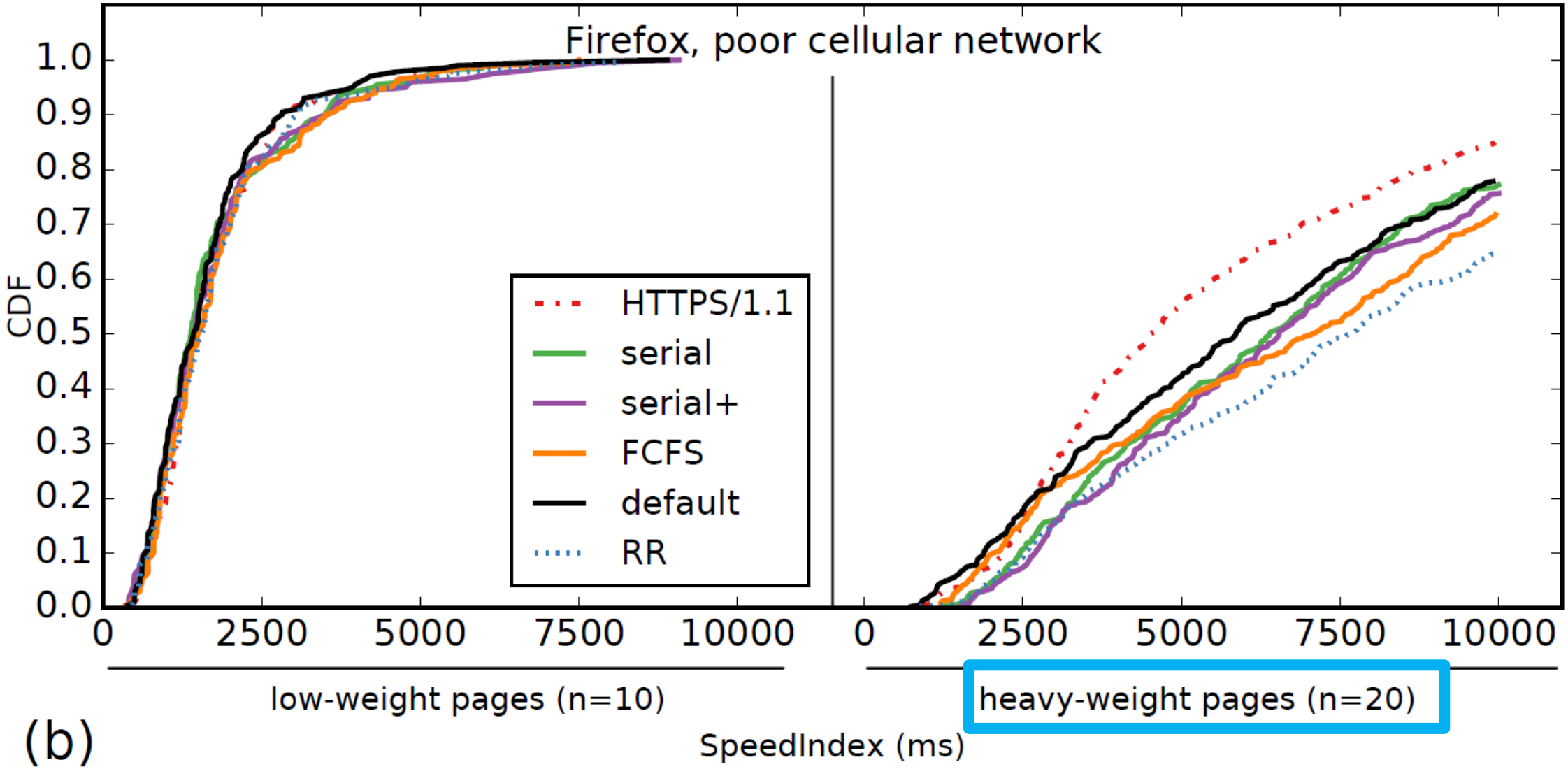
		Cable network					Good cellular network					Poor cellular network				
			speedup		slowdown			speedup		slowdown			speedup		slowdown	
setting	case	R	⩾5%	⩾25%	⩾5%	⩾25%	R	⩾5%	⩾25%	⩾5%	⩾25%	R	⩾5%	⩾25%	⩾5%	⩾25%
Chrome,	FCFS	**24**	**8**	1	**8**	**3**	40	0	0	0	0	40	0	0	0	0
`loadEventEnd`	RR	20	11	1	9	2	35	2	0	3	3	36	1	1	3	3
	vsFirefox	**3**	**37**	**15**	0	0	34	6	6	0	0	38	1	1	1	1
Chrome,	FCFS	**28**	6	1	6	3	**35**	1	0	4	3	**37**	1	1	2	2
`SpeedIndex`	RR	25	5	1	10	2	29	2	2	9	9	31	1	**1**	8	8
	vsFirefox	**4**	**35**	**17**	1	0	26	14	12	0	0	32	6	6	2	2
Firefox,	FCFS	27	8	0	5	0	38	1	0	1	1	38	2	1	0	0
`loadEventEnd`	RR	26	3	0	11	1	38	0	0	2	1	37	1	1	2	1
Firefox,	FCFS	27	**6**	0	7	1	34	3	1	3	3	36	0	0	4	4
`SpeedIndex`	RR	25	1	0	14	2	30	0	0	10	8	30	**1**	**1**	**9**	**9**

Fast network



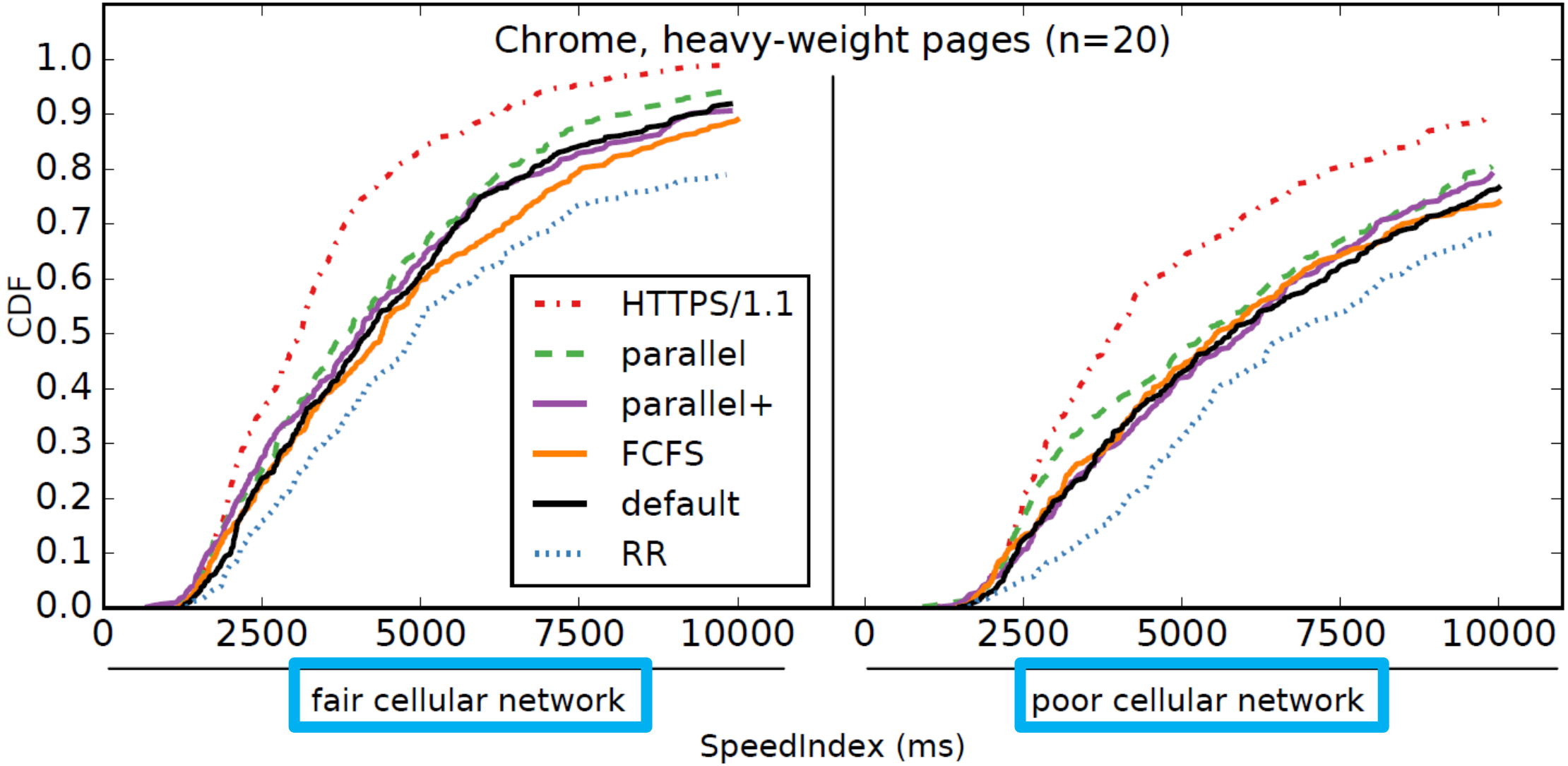
(a)

Slow network

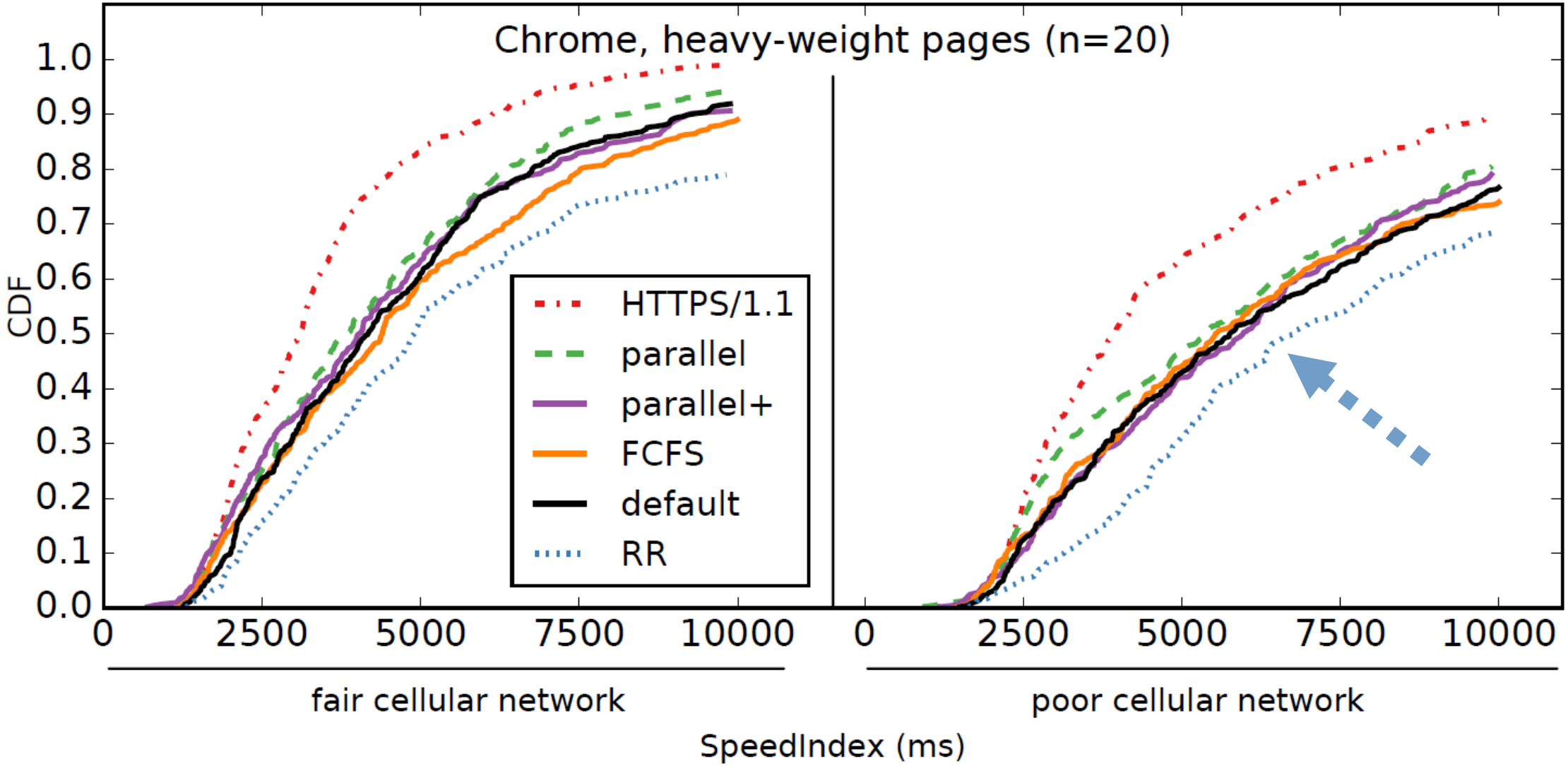


(b)

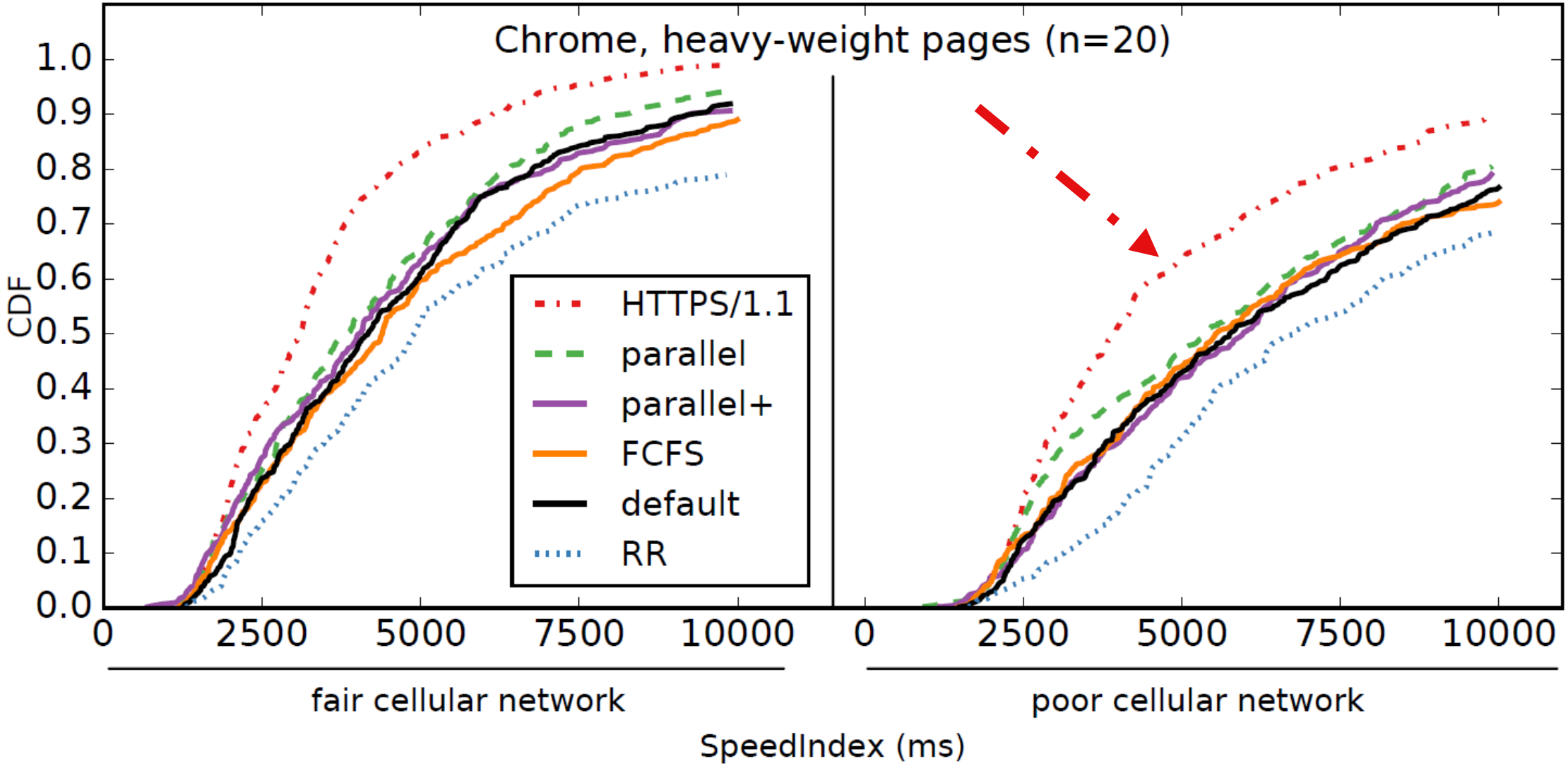
Slow networks



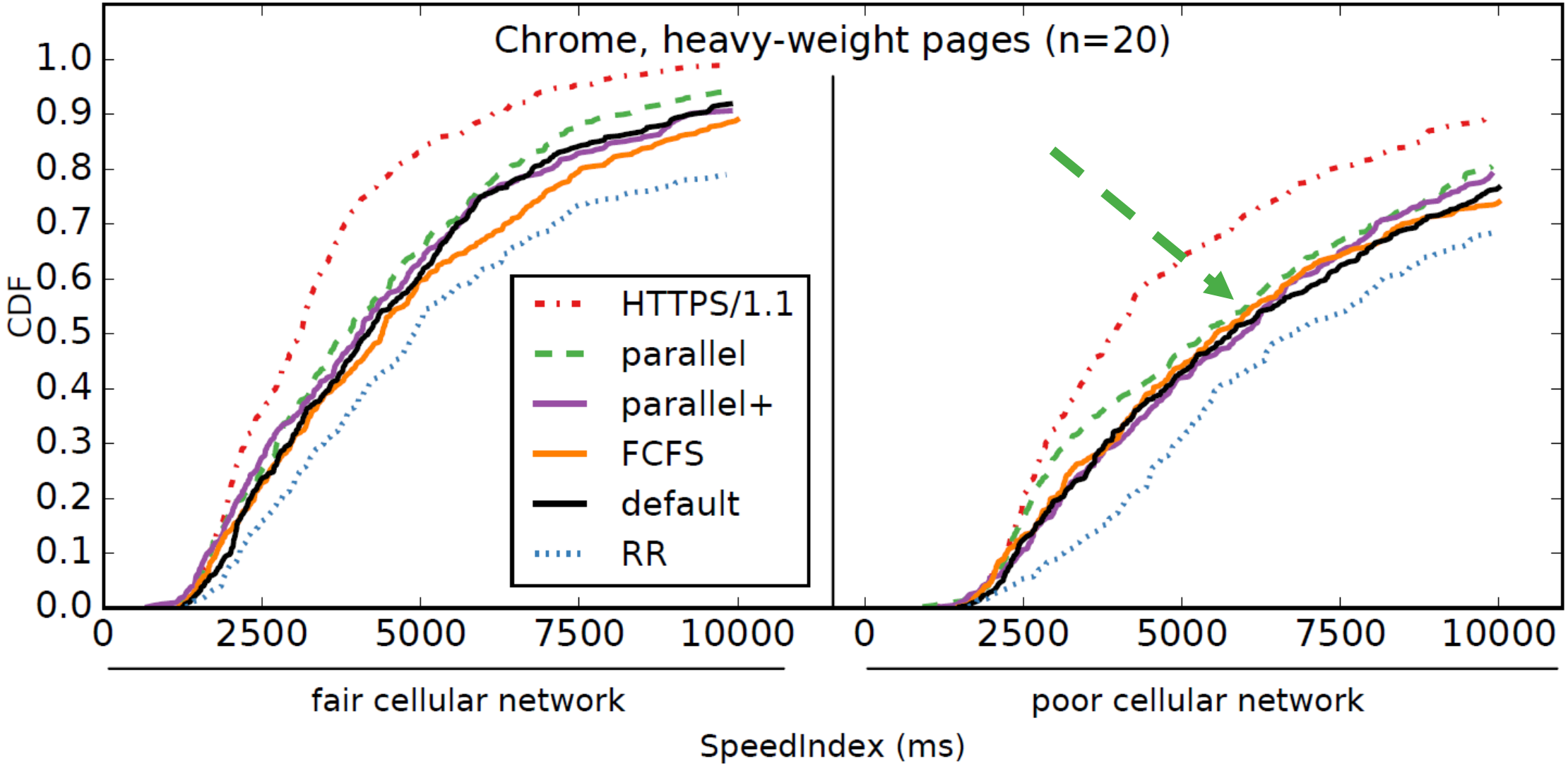
Slow networks : RR is slowest



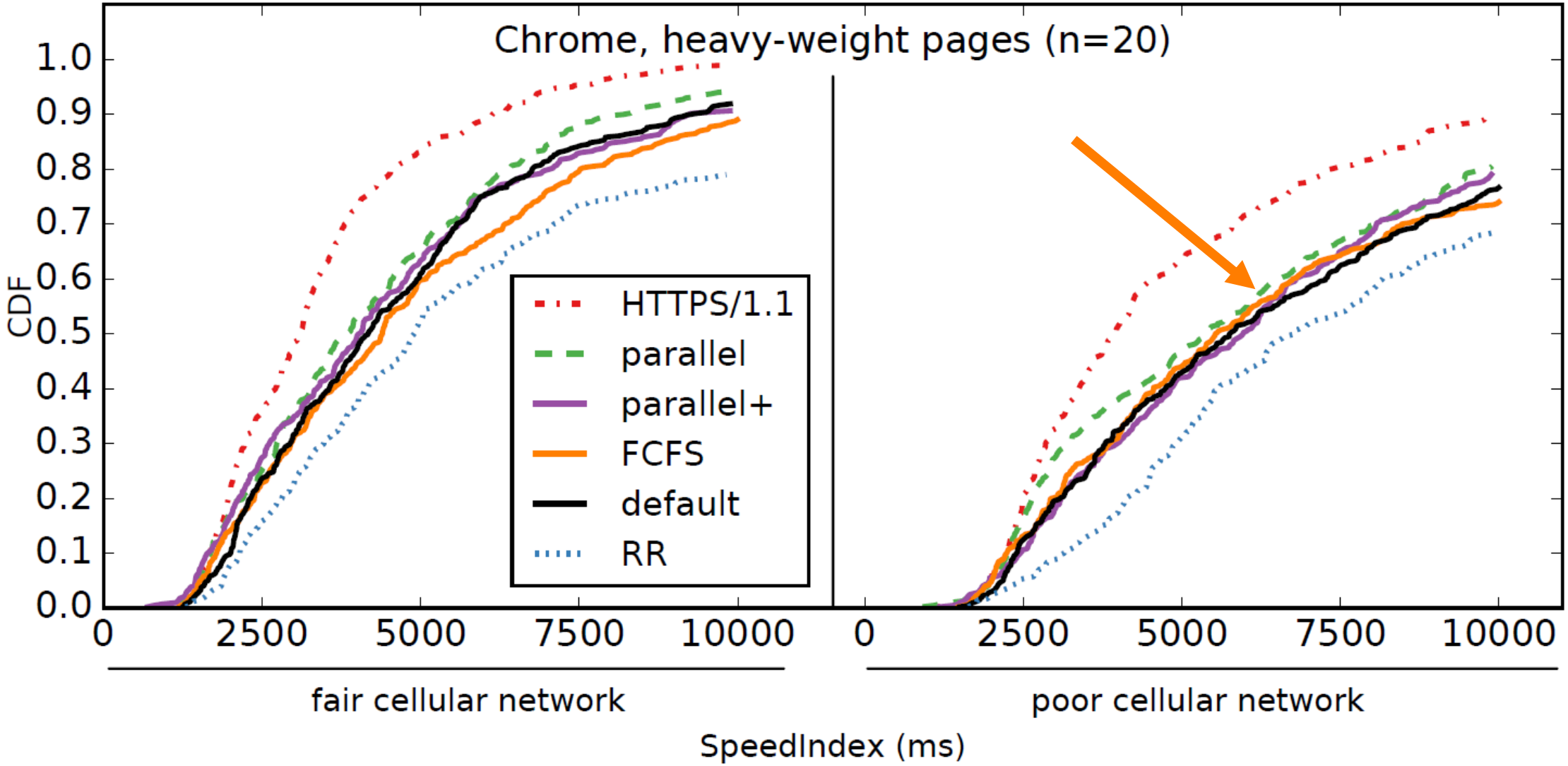
Slow networks : HTTP/1.1 is fastest



Slow networks : Parallel helps, but barely



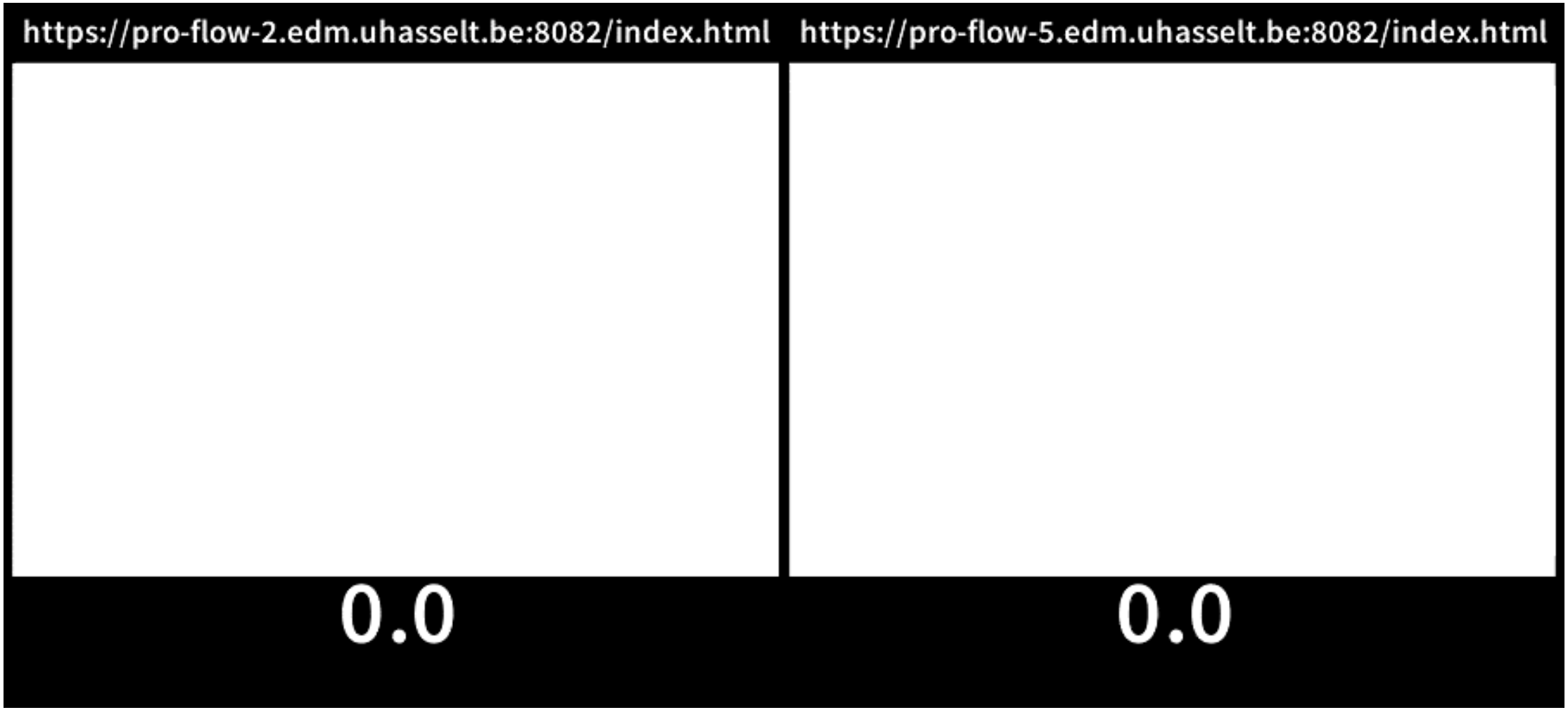
Slow networks : FCFS performs surprisingly well



Experimental Evaluation

Individual Case Studies (i.e., webpage-specific)

RR can yield extreme visual speedup

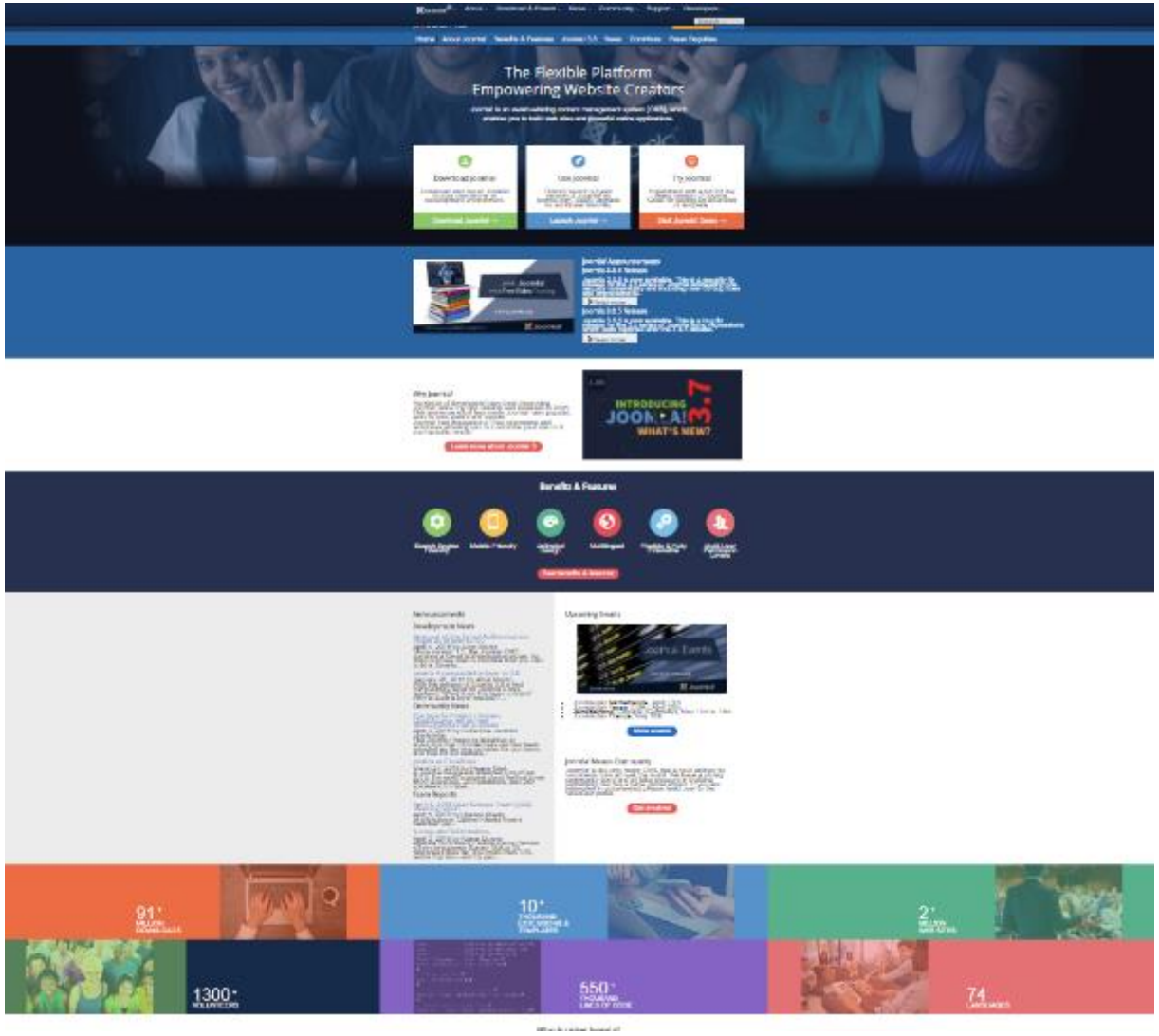


Round Robin

Browser Default

https://www.pinterest.com/

FCFS success story



Discussion and Outlook

Discussion

- Are priority setups tied to browser internals?
  - Still unclear, but tends towards NO

Discussion

HTTP/2 resource sequencing causes HOL blocking for images and async scripts

Project Member Reported by pmeenan@chromium.org, Sep 29 2016

I didn't pay too much attention when the H2 stream dependencies were added but it looks like Chrome is setting the exclusive bit on every request and building dependency chains that are reducing the effectiveness of prioritization.

Specifically, with the exclusive bit set, only one resource can download at a time. If you get the sequencing right, this makes sense for CSS and blocking JS but it can be bad for things like images and async scripts. It also doesn't behave well when higher priority resources are added after lower priority resources (causing the high priority resources to block until some of the queued low priority/exclusive requests have completed).

Comment 19 by y...@yoav.ws, Nov 16 2016

Cc: y...@yoav.ws

I strongly agree with Pat that strict in-order sending of each resource in its entirety is not the right way to go for resources that are progressively rendered. That is not only progressive images, but all images as well as any HTML resources. I also agree with Pat that these should not be special-cased in the network stack, but the "progressiveness" of the resource should be piped through from the renderer.

https://bugs.chromium.org/p/chromium/issues/detail?id=651538

Discussion

- Are priority setups tied to browser internals?
  - Unclear, but tends towards NO
- Current heuristics barely outperform naive setups
  - And it’s also difficult to improve with generic logic (serial+/parallel+)
- Shows the need for flexible per-resource prioritization options
  - See also related work (Wprof, Polaris, Vroom, …)
  - Priority-hints have been proposed, but only Google so far

https://groups.google.com/a/chromium.org/forum/m/#!topic/blink-dev/65IfM2f0eeM

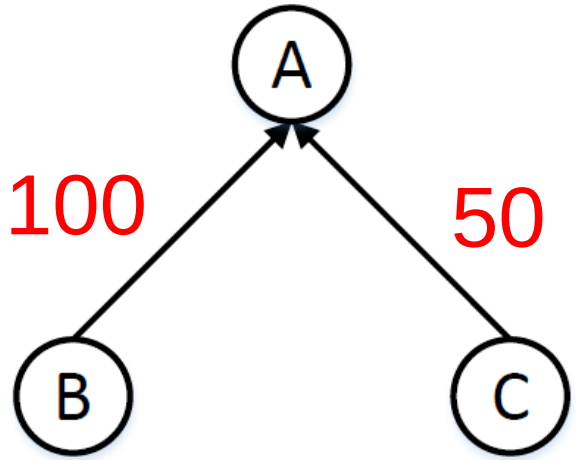
Outlook

- QUIC protocol
  - Prioritization at the transport layer
  - Server can impact priorities more in conjunction with congestion control
- Let’s change default from RR to FCFS in spec?
  - And should someone tell Microsoft?
- Up-front dependency graph calculation (for custom priorities!)
  - Already happening for JavaScript/CSS modules/web components + related academic work

Questions and Extra slides

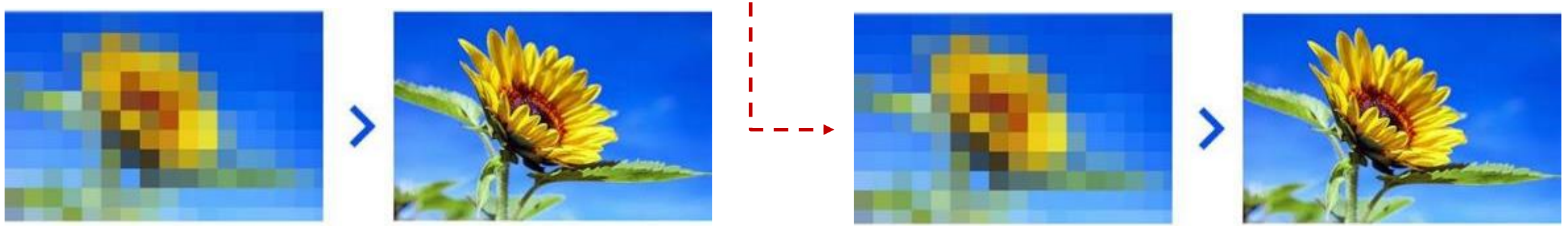
- Usefulness of bandwidth sharing
- Updates to browser behavior since June 2017
- Dependency buildup mechanism in detail
- Faster round robin pinterest waterfalls
- Nuances for algorithm behavior in Firefox
- Prioritization bugs in Service Workers, H2 push, async/defer, Resource Hints
- HTTP/1.1 speed differences explained

Parent / Child dependencies



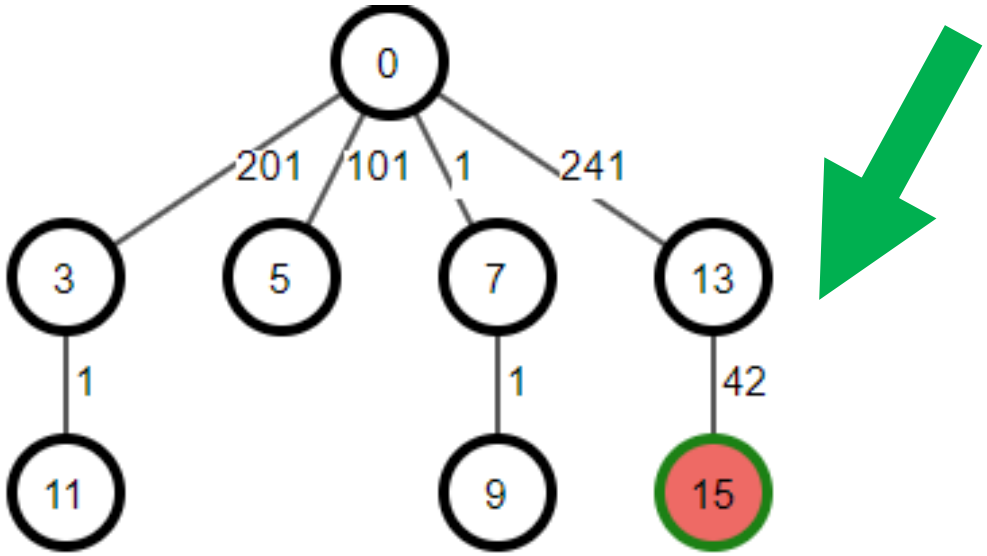
Parent first, then children

Siblings share bandwidth

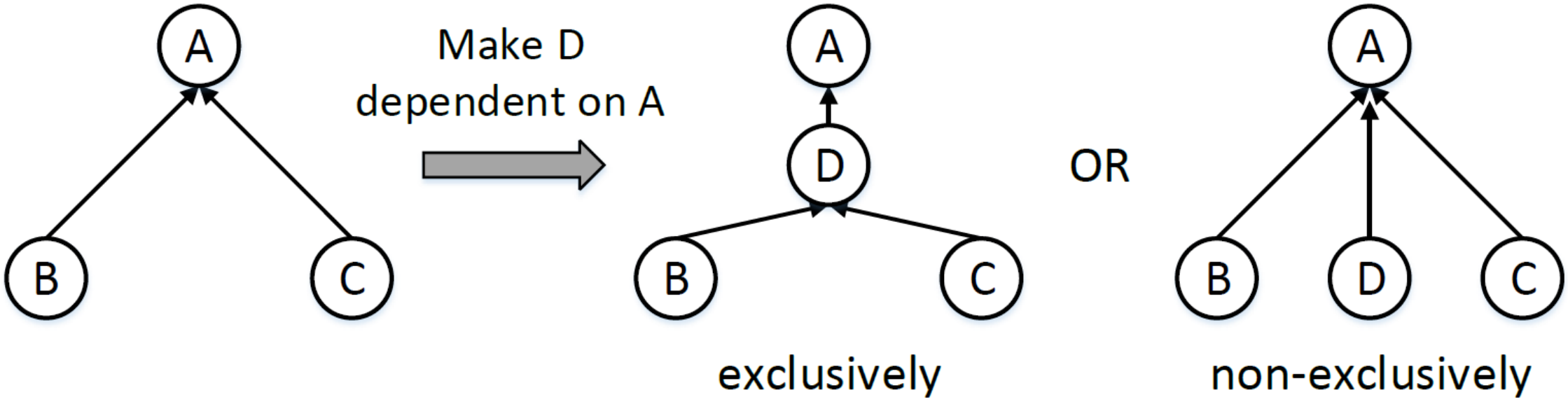


https://cloudpack.media/37251

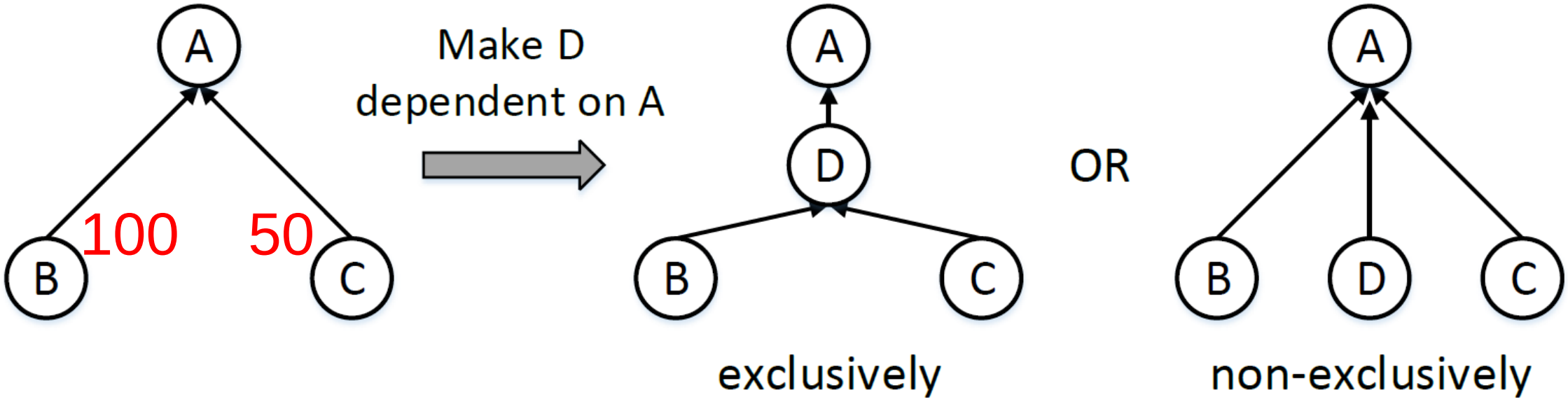
Evolution of Real Browser Prioritization Strategies



Exclusive vs Non-Exclusive adding of nodes

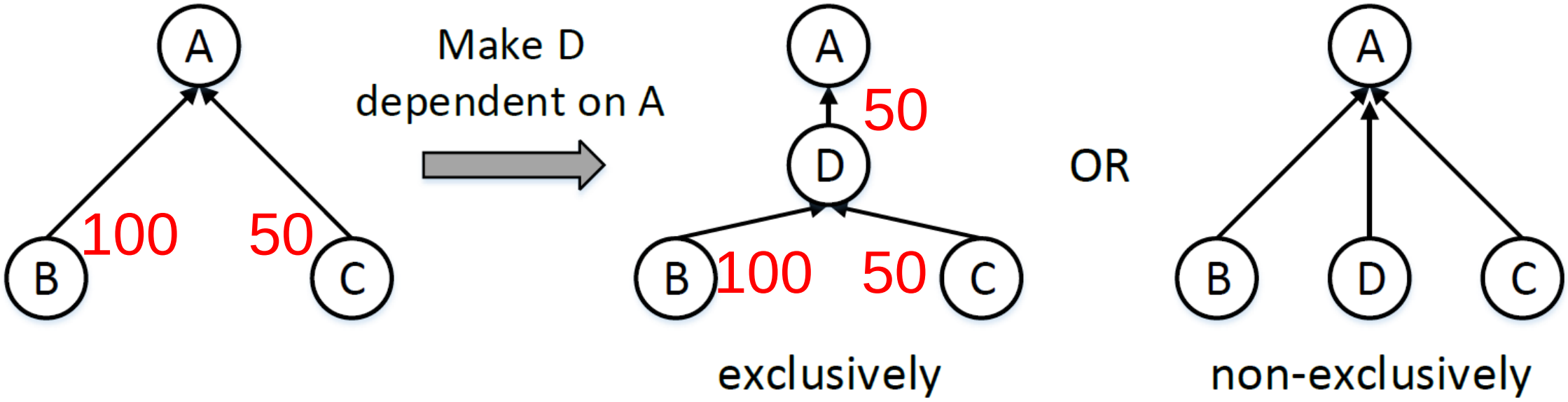


Weights on siblings



B gets 66%

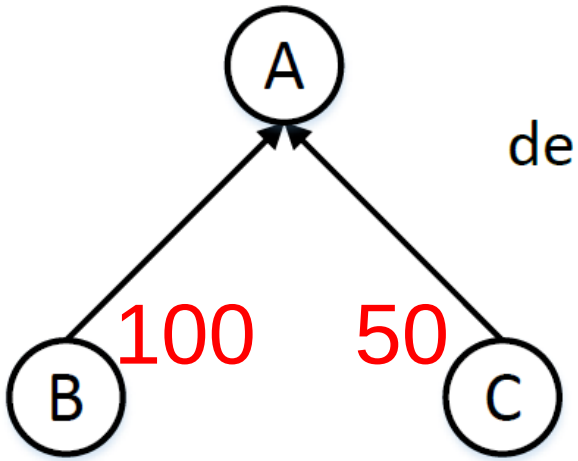
Weights on siblings



B gets 66%

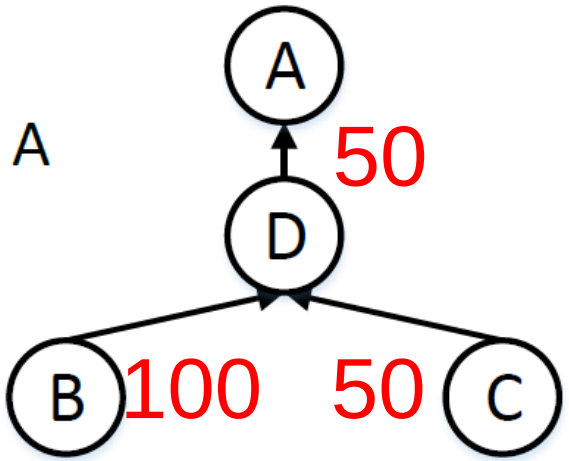
B gets 66%,
D gets 100%

Weights on siblings



B gets 66%

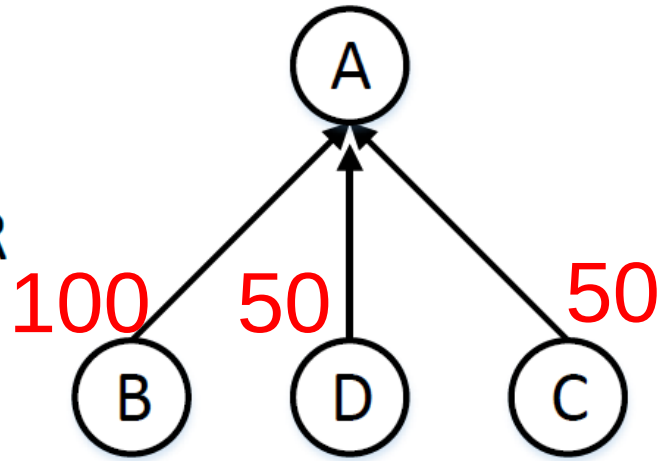
Make D
dependent on A



exclusively

B gets 66%,
D gets 100%

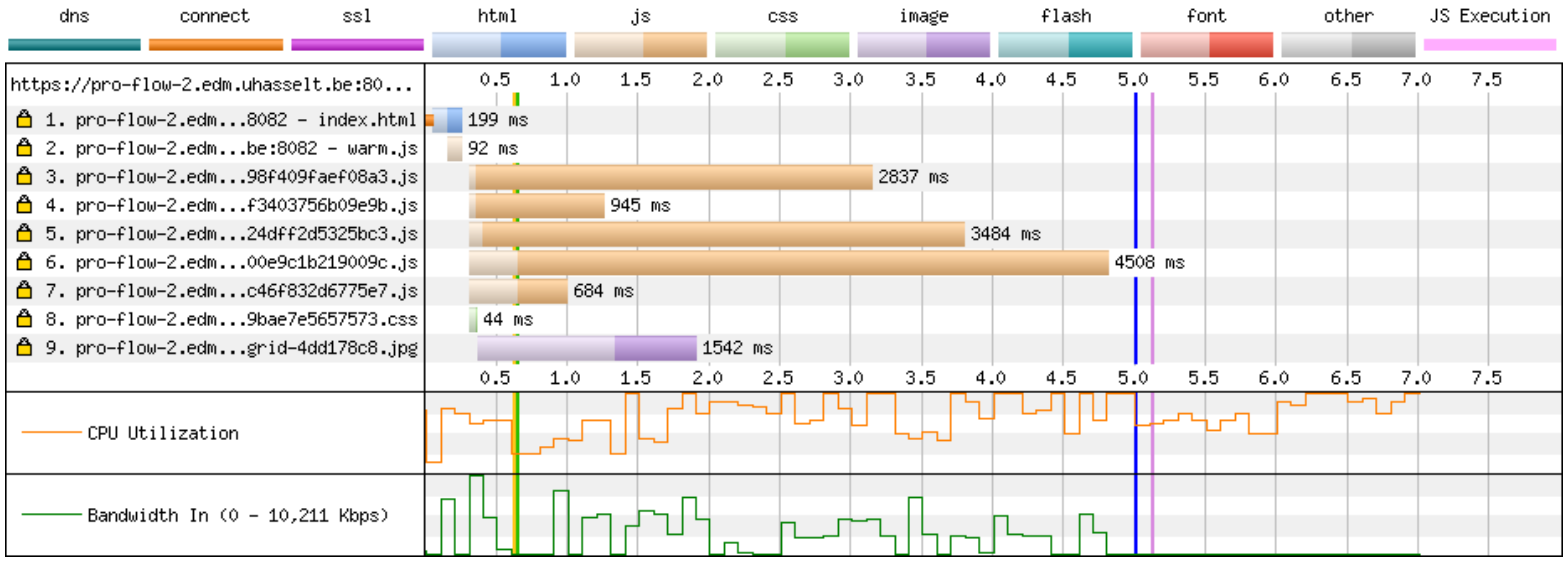
OR



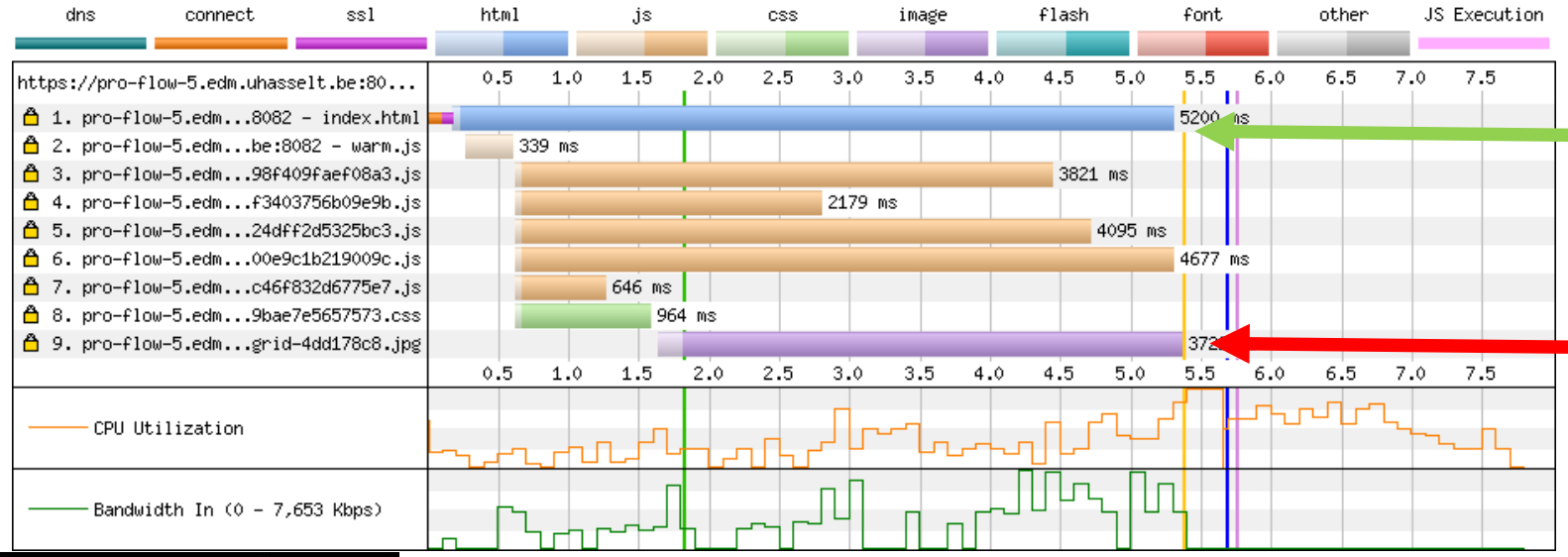
non-exclusively

B gets 50%,
D gets 25%

RR is faster on pinterest



Round Robin

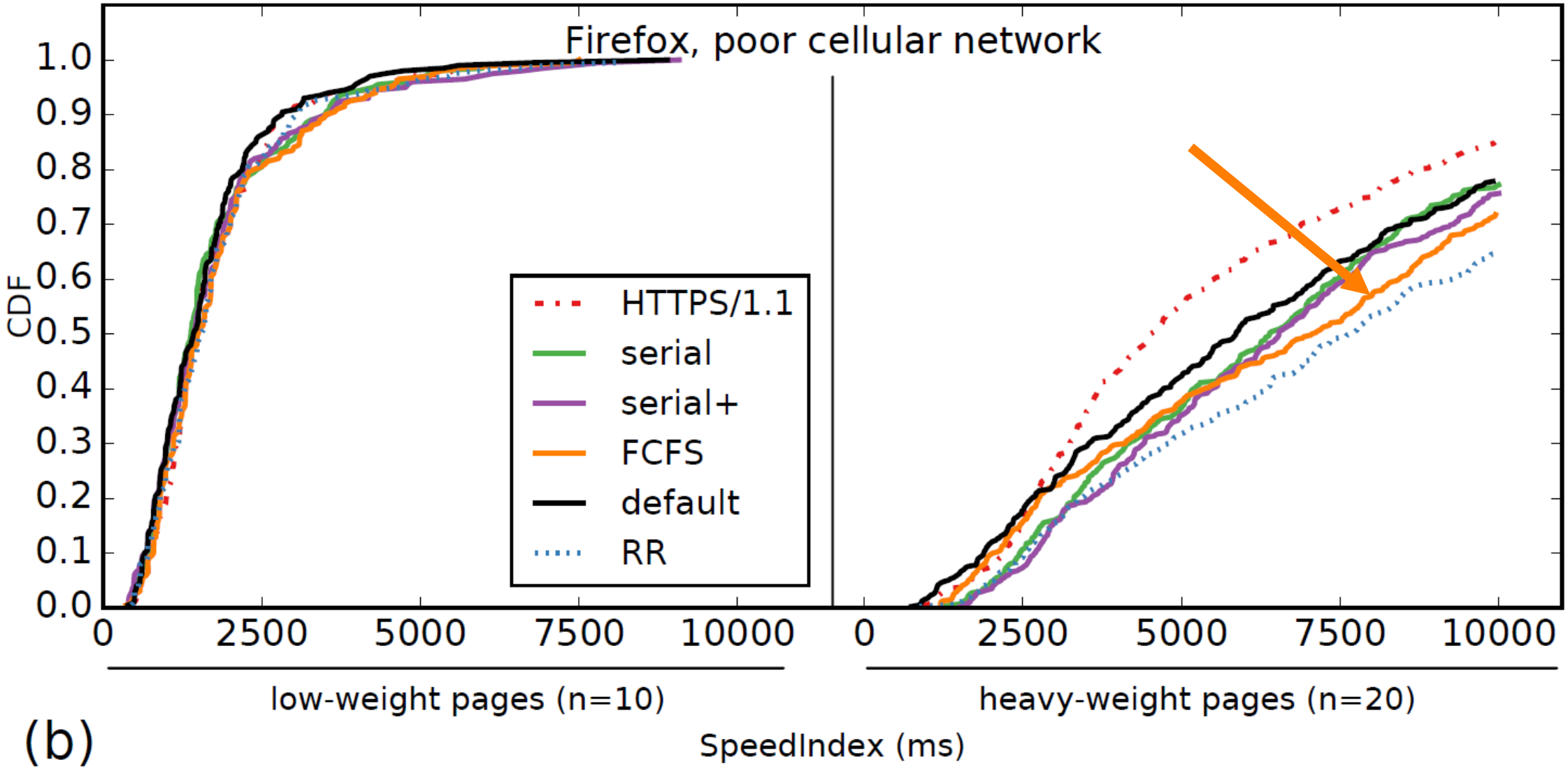


Browser Default

background image

https://www.pinterest.com/

On Firefox, FCFS isn't that great, heuristics best, RR still the worst

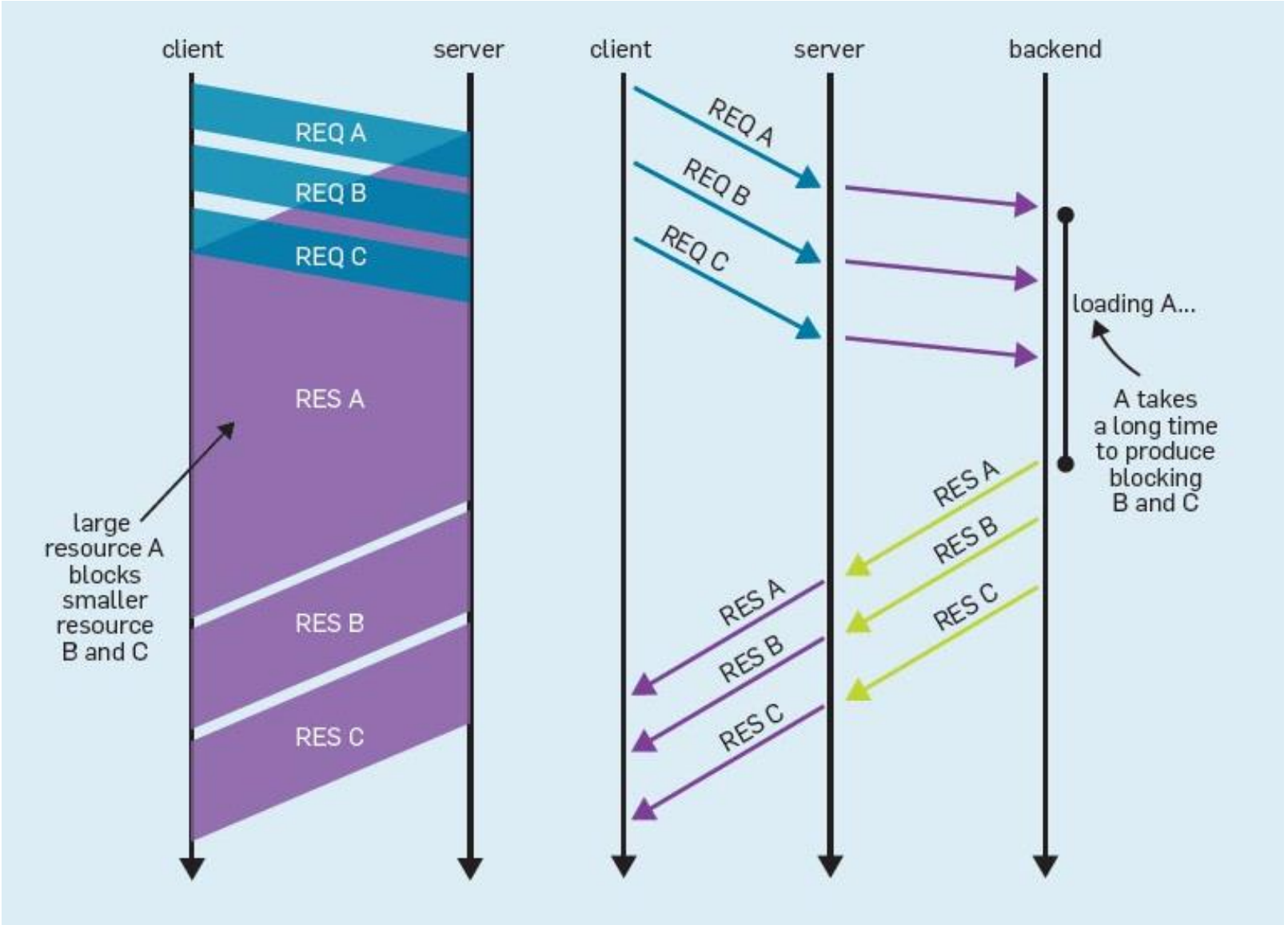


(b)

Real Browser Implementation issues

- Service Workers lose all priority info
- Weird positioning of PUSH resources in FF and Safari
  - Chrome promotes them to correct position
  - Firefox adds them directly to root node
  - Safari keeps weight of 16
- Weird behaviour for preload/prefetch and async/defer
  - Chrome: async/defer are Low priority always
  - Firefox: no special treatment, similar to sync versions

HTTP/1.1 : Head-of-Line Blocking



http://deliveryimages.acm.org/10.1145/2390000/2380673/figs/f5.jpg