# Untangling Header Bidding Lore

## Some Myths, Some Truths, and Some Hope

Waqar Aqeel[1(✉)], Debopam Bhattacherjee[4(✉)],
Balakrishnan Chandrasekaran[3(✉)], P. Brighten Godfrey[5],
Gregory Laughlin[6], Bruce Maggs[1,2], and Ankit Singla[4]

[1] Duke University, Durham, US
{waqeel,bmm}@cs.duke.edu
[2] Emerald Innovations, Cambridge, USA
[3] Max-Planck-Institut für Informatik, Saarbrücken, Germany
balac@mpi-inf.mpg.de
[4] ETH Zürich, Zürich, Switzerland
{debopam.bhattacherjee,ankit.singla}@inf.ethz.ch
[5] University of Illinois Urbana-Champaign, Champaign, US
pbg@illinois.edu
[6] Yale University, New Haven, US
greg.laughlin@yale.edu

**Abstract.** Header bidding (HB) is a relatively new online advertising technology that allows a content publisher to conduct a client-side (i.e., from within the end-user's browser), real-time auction for selling ad slots on a web page. We developed a new browser extension for Chrome and Firefox to observe this in-browser auction process from the user's perspective. We use real end-user measurements from 393,400 HB auctions to (a) quantify the ad revenue from HB auctions, (b) estimate latency overheads when integrating with ad exchanges and discuss their implications for ad revenue, and (c) break down the time spent in soliciting bids from ad exchanges into various factors and highlight areas for improvement. For the users in our study, we find that HB increases ad revenue for web sites by 28% compared to that in real-time bidding as reported in a prior work. We also find that the latency overheads in HB can be easily reduced or eliminated and outline a few solutions, and pitch the HB platform as an opportunity for privacy-preserving advertising.

## 1 Introduction

Online advertising is a multi-billion dollar industry, with estimated global revenues of more than 300 billion dollars (USD) in 2019 [19]. Revenues from advertising platforms exhibited a consistent positive growth rate over the last nine quarters [32], and are projected to reach 0.5 trillion USD within the next four years [19]. Programmatic advertising, which includes both real-time bidding (RTB) and header bidding (HB), dominates the online advertising space today: It accounts for 62% of the total advertising spend [32]. In this paper, we offer insights into the design and performance of HB auctions using *real* end-user measurements, which have not been available before.

A. Sperotto et al. (Eds.): PAM 2020, LNCS 12048, pp. 280–297, 2020.
https://doi.org/10.1007/978-3-030-44081-7_17

Header bidding, introduced around 2013[1] [9,50,57], is a nascent programmatic advertising technology that improves transparency and fairness in real-time bidding (RTB). In RTB, ad slots on a web page are offered to advertisers (or, more generally, buyers) following a *waterfall* model: one by one in a predetermined order, where the first one to bid a high enough price wins the slot. The ordering is, moreover, not determined by the publisher (or web site owner), but by an *ad server*, a third party that facilitates the auctioning of slots to buyers. HB, in contrast, enables the publisher to solicit bids simultaneously from multiple *ad exchanges*, where each exchange is a marketplace for advertisers to bid on ad slots. Under HB, the publisher typically places some JavaScript code within the web page's `HEAD` tag that, when loaded in an end-users' browser, launches an in-browser auction for the ad slots on that page. This in-browser, publisher-controlled, real-time ad auction permits publishers, as we show later, to significantly increase their ad revenues. Perhaps as a consequence, HB has already gained significant adoption: 22% of the Alexa top 3k web sites use HB [1], and a more recent study reports 22–23% adoption among the top 5k sites [35]. If we remove sites that are ad-free (e.g., government and non-profit web sites) or which use an in-house ad platform (e.g., Google and Facebook), HB adoption among the top 1k sites is at 80.2% and growing fast [1].

Users might also benefit from HB: It could be leveraged to build a privacy-preserving and transparent advertising ecosystem, where the end users have control over their data. They could decide, on a per-web-site basis, for instance, what information (e.g., concerning their interests or preferences) to barter for helpful ads from advertisers. If properly designed, these auctions can also provide the necessary oversight into end-user tracking, and transparency that users often expect when seeing ads [55,56]. Any debate on such a novel advertising ecosystem is possible, however, only if the underlying HB platform is proven to work well.

Real-time auctions such as those in RTB and HB are latency-sensitive. Google AdX (one of the largest ad exchanges) requires, for instance, that all advertisers respond within 120 ms of the bid request being sent [22]. Setting aside a recommended room of 20 ms for unexpected delays, and 40 ms for bid computations and data fetches, leaves only 60ms for the round trip between an advertiser and Google AdX [36]. Given the state of latency in the Internet [10], it is not surprising that Google AdX recommends that advertisers peer directly or co-locate with AdX to minimize latency. Ensuring low latency for bid requests and responses is even more challenging in HB, since users' browsers cannot be co-located with exchanges. Publishers thus set very long deadlines (from 500 ms to 3000 ms) to ensure that all ad exchanges in an HB auction have a chance to bid. These long deadlines are consistent with the widespread belief that the in-browser auction held in HB imposes significant latency overhead [17,35]. The central theme of this paper is that these concerns may be overblown. In particular, we identify the sources of overhead and outline several avenues for lowering it. We summarize our contributions as follows.

[1] The lack of any formal specification or standardization process makes it difficult to nail down the exact time header bidding was introduced.

⋆ We developed a web browser extension, for both the Google Chrome and Mozilla Firefox browsers, to dissect in-browser HB auctions. We released the source code of the extension as open source software [6].

⋆ Prior work on header bidding [35] relied on regularly crawling websites from a single vantage point. Crawling is valid for some of the analyses they do, such as how many ads are on a web page, and which exchanges are involved, but it cannot provide any useful insights into networking timing for real users. Revenue measurements will also be inaccurate as advertisers bid only token amounts for synthetic user profiles. We gathered measurements of in-browser HB auctions from about 400 real users, who volunteered to install and use the extension for a period of 8 months. We also made the data set constituting these measurements publicly available [6]. We call this data set Rum.

⋆ Using the Rum data set, we demonstrate that ad revenue (estimated using the median of bids from ad exchanges) from HB is significantly higher (28%) than that reported for RTB in other studies. We also estimate the publishers' latency overheads when integrating with ad exchanges and discuss their implications for publishers' ad revenue.

⋆ We break down the time spent in soliciting bids from ad exchanges into its contributing factors and highlight areas for improvement. We do not find any *fundamental* problem with client-side HB (i.e., in-browser auctions) implementations. It is not necessary to move these in-browser auctions to ad servers or, more generally, away from end users to lower auction duration.

## 2 A Brief History of Programmatic Advertising

The introduction of real-time bidding fundamentally changed the way ads were bought and sold: RTB, by leveraging programmatic advertising, facilitated the sale and purchase of ads on a per impression or view basis [23]. Under RTB, publishers (e.g., www.nytimes.com) announce their ad slots in *real-time* (i.e., when serving content to end users) to ad servers (e.g., DoubleClick for Publishers). The ad servers then reach out to typically several *demand sources* (e.g., privately negotiated advertisers, Google AdSense, or an ad exchange), where advertisers either bid for a chance to place ads in the available slots, or have previously negotiated contracts to show a certain volume of ads for a price.[2] A bid, typically expressed in *cost per mille (CPM)*, represents the amount that an advertiser is willing to pay for one thousand impressions or views of the ad [61]. Advertisers estimate the worth of each ad slot using user-specific data from one or more *data brokers*, which track end users to compile a database of user profiles (e.g., comprising details such as a user's gender, age, and location).[3]

*The Need for Header Bidding.* In RTB, ad servers contact demand sources in a rank order (referred to as the *waterfall model*) determined *a priori* by the publisher and/or ad server. For a given ad slot, the process terminates as soon

[2] Ad exchanges and advertisers are also collectively referred to as *buyers*.

[3] For more details on data brokers, we refer the reader to [4,47].

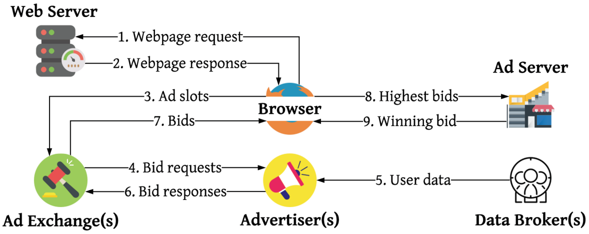


**Fig. 1.** Interactions between different elements in client-side header bidding

**Table 1.** A summary of the RUM data set

| Attribute(s) | Value |
|---|---|
| Users | ≈400 |
| Duration | 8 months |
| Cities; countries | 356; 51 |
| Web sites | 5362 |
| Ad exchanges | 255 |
| Page visits | 103,821 |
| Auctions | 393,400 |
| Bids | 462,075 |

as the slot is filled by a source, even if those appearing later in the ordering might have offered a higher price. This static ordering, hence, treats the sources, and in turn advertisers, unfairly. Publishers suffer from lower ad revenues—due to lost opportunities—and a lack of transparency—they do not know of the demands across different sources, especially ad exchanges, to inform a better ordering.

*Leveling the Playing Field.* Header bidding was introduced sometime around 2013 or 2014 [9,39,50,57], to address RTB's shortcomings. HB allows the publisher to contact different advertisers and ad exchanges concurrently. Then, these bids are sent to the ad server so they can be compared to other demand sources. With this model, ad exchanges have a fair chance to bid for the slots, and publishers can monitor the demand across different exchanges. Over time, three different kinds of header bidding implementations have emerged: *client-side*, *server-side*, and *hybrid* (see [35]), although client-side is the original and still dominant implementation. For the rest of this paper, we focus our attention on client-side HB.

*Client-Side HB.* The publisher adds JavaScript in the web page's header, i.e., content enclosed by the HEAD HTML-tag that when processed by an end-user's browser, kick-starts an *in-browser* auction (illustrated in Fig. 1). The auction concurrently solicits bids from different exchanges for the ad slots on that page. The bids received until the end of the auction are then sent to the ad server to compare with those retrieved via the waterfall-model auctions in the ad server. Finally, the ad server chooses the highest bid, i.e., with the highest CPM, and returns the winning bid to the browser. The browser then contacts (not shown in the illustration) the winning bidder to retrieve the ad and display it.

## 3 Real User Measurements

Our objective is to passively observe the in-browser auctions of the client-side header bidding process. To this end, we developed a browser extension, released it to public, and, from real end users who used the extension for 8 months, obtained measurements pertaining to HB auctions.

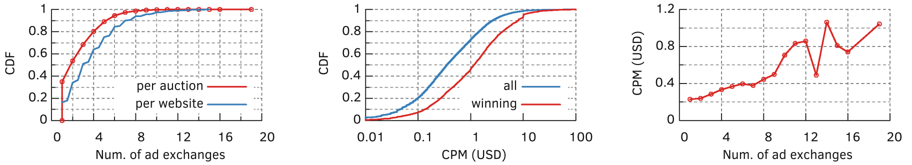
The browser extension utilizes the Prebid library [40], for it is the most widely used HB JavaScript library, with 63.7% of the publishers using it as of

August 2019 [1]. The extension, *MyAdPrice*, is available on both the Google Chrome web store and Firefox Add-ons web site. It uses the JavaScript APIs for WebExtensions [31] to access the document-object-model (DOM) tree [30]. Via the DOM, it learns of (a) the ad slots on the Web page, (b) the name and IP addresses of the ad exchanges that were contacted to fill up those slots, (c) the bids received from different exchanges, and (d) which bids, if any, won the auctions and for which ad slots. The extension also uses the Web Performance Timing API (WPT) [59] to capture the time spent in each step of the request such as DNS resolution, performing TCP/TLS handshakes, soliciting bids from exchanges (i.e., transferring the data carrying the requests to the exchange's servers) for various ad slots, and receiving bids (i.e., retrieving the response data from the exchange's servers) from the exchanges. Outgoing ad server requests are also checked for query parameters.

In addition to releasing the source code for the extension as open source software, we announced it on university mailing lists and public forums to increase adoption. We recruited approximately 400 volunteers from diverse locations, with nearly 50% of the users from the US. The rest were mostly from European countries including Bulgaria, the United Kingdom, France, Norway, Germany, and the Netherlands, and a small, but significant fraction, also from Canada and India. Table 1 presents the high-level characteristics of the `Rum` data set, comprising real user measurements over a period of 8 months. The end users visited about 5k web sites, for a total of about 100k web page fetches. The users' browsing activity resulted in about 400k auctions involving about 500k bids from 255 ad exchanges. In total, we observed 916,447 requests issued by the users' browsers to ad exchanges and servers; 247,869 (27%) of these were to ad servers, while the remaining 668,578 were to ad exchanges. Our browser extension recorded the timestamp of each request using the browser's Navigation Timing API [58]. Using these timestamped requests, we estimated the duration of auctions and investigated the factors that affect an auction's duration.

### 3.1 Privacy and Ethics

Our extension, by *default*, sends *no data* from the user's browser. The extension uses the browser's local storage to store data pertaining to ad slots in different pages and the bids received for each. The extension uses this data to compute the "ad-worthiness" of the user—the money that advertisers intend to make off of the user, and allows the user to view this locally-stored information. Users may *opt in* to share data including domain names of web pages they visit, i.e., only those that use header bidding, ad slots on the pages, exchanges contacted for ads, bids received, timing information on various phases of the in-browser auction, and, lastly, their geographical location at city level. The data shared does *not* have any information to uniquely identify a user. This opt-in data from real end users constitutes the `Rum` data set. When we consulted our institutional review board, they declared that we do not require an approval since we do not gather any personally identifiable information.



(a) (b) (c)

**Fig. 2.** (a) In the median, auctions involve only two ad exchanges and web sites (publishers) connect with only three ad exchanges. (b) Bid prices show significant variation, with approximately 30% of bids having at least $1 CPM. (c) The median CPM or ad revenue increases with number of ad exchanges contacted.

The strict privacy standards we set for ourselves also mean that our dataset has limitations. Since we don't upload any data by default, not all installations result in data collection. Also, since we don't identify users, we cannot tell how many unique users uploaded data to our servers. We also cannot track users across different websites, and cannot profile based on age, income etc.

We refrained from conducting any experiment that would harm end users or publishers or even the advertisers. The extension is merely a passive observer of the in-browser auctions. We did not crawl web sites, since that would generate synthetic ad impressions for which advertisers might have to pay the publishers. Crawling activities may also lead to exchanges flagging the publisher for suspicious activity. We did not craft synthetic user profiles for similar reasons.

## 4 Ad Exchanges, CPM, and Ad Revenue

The large number of ad exchanges observed in the RUM data set (in Table 1) suggests that publishers leverage HB to integrate with many buyers in order to maximize their ad revenue. To investigate further, we computed the number of ad exchanges contacted, derived from the count of distinct ad exchanges from which bids were received by the browser, per auction as well as per web site. The CDF of the number of ad exchanges contacted (Fig. 2a), across all auctions and web sites, reveals that most web sites (60%) use at most four ad exchanges, and 10% use at least twice as many. Per this figure more than a third (35%) of all auctions involve only one exchange and a fifth use at least four exchanges. Publishers seem conservative in connecting with many ad exchanges, even if HB libraries make it easy to establish such direct integrations. Prebid, the most widely used JavaScript HB library, for instance, offers more than 250 integration modules or "adapters" [43]; to integrate with an ad exchange, publishers simply have to enable or include the corresponding adapter.

The CDF of CPMs across all auctions, in Fig. 2b, shows a significant variation in bid values. While 20% of bids have at most $0.10 CPM, nearly 30% of the bids have at least $1 CPM. We also observed 2 bids with CPM between $500 – $1000 and 3 with more than $1000 CPM. We find that ad revenue in HB (for our volunteers) is not lower than that of RTB reported in other studies. For example, the median winning CPM that we observe ($1.15) is 28% higher than the RTB median of $0.90 reported in [37]. Furthermore, we grouped together ad slots based on the number of ad exchanges from which they solicited bids and computed the median value of bids in each group (Fig. 2c). The median value of bids increases significantly with the number of ad exchanges. It is indeed in the publishers' best interests to connect with many ad exchanges—at least more than the current number of ad exchanges (Fig. 2a) they are using.

Publishers could be contacting fewer exchanges for performance reasons. We investigate the implications of integrating with more ad exchanges for auction duration in the next section.

## 5 Auction Duration and Implications

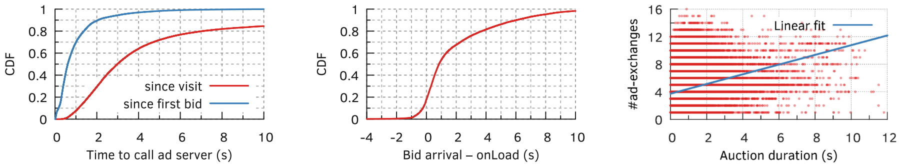
The Prebid Javascript library does not provide explicit timestamps for auction start, and end. As an approximation, we use the first bid request from the browser to an ad exchange to signal an auction's start. A call to the ad server marks the end of an auction (step 8 in Fig. 1). Hence we approximate the auction duration as the time between the first bid request, and the ad server call. The CDF of these estimates, in blue in Fig. 3a, shows that auctions last for 600 ms in the median and some 10% of auctions last longer than 2 s. Despite the publishers integrating with a small number of ad exchanges, auction durations are fairly high.[4]

The CDF of the elapsed time between when the user arrives at a given web page and the end of the auction ("since visit" line in Fig. 3a) reveals that the browsers spend a large amount of time prior to launching HB auctions. Perhaps web browsers spend this time prioritizing content over ads. Web pages may also refresh ads based on user activity, e.g., scrolling down or reactivating an inactive tab, triggering some auctions much later than when the user arrived at the web page. These are separate auctions that are triggered in response to these events.

To ascertain the implications of auction duration for end users, we focus on the page-load time (PLT), and measure the time it takes for the browser to fire the `onLoad` event after the user navigates to the web page. We subtract the `onLoad` time of a web page from the bid-arrival times associated with the ad slots or auctions on that page, and plot the CDF of the resulting values in Fig. 3b. Only a small fraction of bids (18%) arrive before the page load is complete; 82% of the bids arrive after the `onLoad` event is fired. Although the shortcomings of the PLT metric in reflecting end-users' experiences is well-known, it is still the most widely used metric [26], and according to this metric auction duration does not significantly impact end-user experiences.

Longer ad auctions could, however, affect publishers and advertisers. The negative effect of latency on e-commerce sales is well-known [3], and [8] concludes that increased response latency decreases click-through rates for search results.

[4] Appendix C presents additional results on factors that may influence the number of exchanges contacted by a publisher.



(a) (b) (c)

**Fig. 3.** (a) Auctions last for 600 ms in the median, and some 10% of auctions last more than 2 s. (b) Auctions, however, do not seem to affect the page load times: Most bids arrive much later than when the `onLoad` event fires. (c) Auction duration increases with the number of ad exchanges contacted.

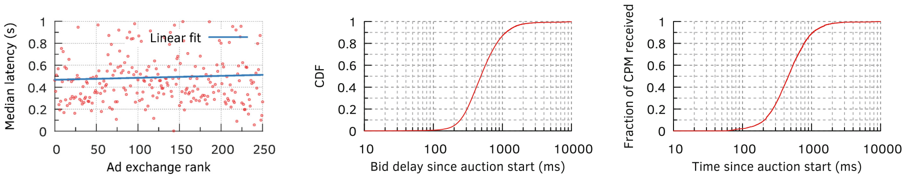
Delay in showing ads likely has the same effect, since a longer duration implies a longer time to display the ad and engage the user. Furthermore, the display of an ad might alter the visual elements or rendering of the web page. Auction duration also increases with the number of ad exchanges contacted by the browser, as the linear fit in Fig. 3c shows. While publishers can limit the auction duration, a smaller timeout could lead to lost revenues, since a higher bid may arrive after the timeout is triggered. Clearly, publishers have to manage the trade-off between maximizing revenue and minimizing auction duration.

A simple approach to managing the trade-off is to cherry-pick ad exchanges that deliver high-value bids. We thus rank-order ad exchanges by *median CPM* of bids sent across all web sites and users. Figure 4a shows, however, no correlation between ad-exchange CPM and the median latency of its bid responses.

Rather than limit the number of exchanges, which is clearly not efficient, publishers could perhaps specify an early timeout. Figure 4b shows the CDF of bid-response arrivals with respect to auction start (i.e., the timestamp of the first bid request). 87% of the bids arrive within 1 s of the start of the auction. Also, the CDF of CPMs of bids as a function of the time they were received since auction start, in Fig. 4c, indicates that 90% of the total CPM is received within the same time span. This observation is in stark contrast with the estimates of auction duration in Fig. 3a ("since first bid" line). More concretely, per Fig. 3a, 30% of the auctions take longer than 1 s, suggesting that publishers are conservative in setting auction timeouts or deadlines: A lot of time is, hence, unnecessarily wasted on waiting for bids that will likely have no significant effect on the auction.

## 6 Sources of Latency in HB Auctions

In this section we delve into the factors that fundamentally determine the duration of the in-browser HB auctions. To this end, we dissect the latency of a bid request into its contributing factors and identify, wherever possible, avenues for mitigating the latency overheads.



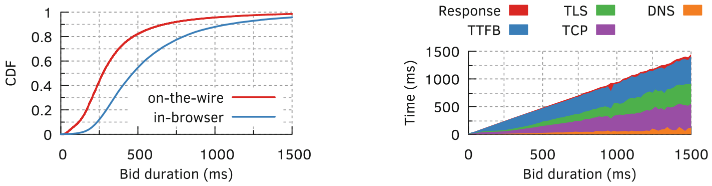
(a) (b) (c)

**Fig. 4.** (a) "High-CPM" ad exchanges are not any faster in responding with bids than "low-CPM" ad exchanges. (b) 87% of the bids and (c) 90% of the ad revenue, estimated through CPMs, arrive within 1 s of the start of the auction.

We define *bid duration* as the time between the start of the bid request being sent out and the end of the bid response being received. We can measure bid duration from two data sources—from within the Prebid JavaScript library (`in-browser`) and through the WPT API [59] (`on-the-wire`). `in-browser` measures the difference between the timestamps that Prebid records when it has prepared the bid request to be sent through the browser, and when it has finished parsing the bid response. `on-the-wire` is just the duration between the bid request and response as provided by the WPT API.

The CDF of bid durations calculated separately from these two sources, in Fig. 5a, shows, surprisingly, a difference of 174 ms in the median, which is fairly large. This difference is suggestive of poor implementation practices or bugs in HB libraries, specifically in the logic implemented in the adapters developed for integrating the publisher with an ad exchange or advertiser [41]; it could also be that publishers are using adapters incorrectly. Consider the scenario in which a publisher's web site contacts exchanges $\mathcal{A}$ and $\mathcal{B}$. Suppose that bid duration for exchanges $\mathcal{A}$ and $\mathcal{B}$ are 250 ms and 300 ms, respectively. In the ideal case, the adapters for $\mathcal{A}$ and $\mathcal{B}$ should be making concurrent, asynchronous requests. Suppose that $\mathcal{B}$ has a bug in its adapter: it makes a synchronous request. If the publisher integrated HB so that $\mathcal{B}$ is contacted before $\mathcal{A}$, given that $\mathcal{B}$ makes a synchronous call, the call to $\mathcal{A}$ will get delayed until the request to $\mathcal{B}$ completes. The auction now lasts for 550 ms instead of only 300 ms (in case of a correct implementation). Such pitfalls are detailed in [18] and [42].

The WPT API allows us to break down the bid duration into the various steps involved. We specifically gather the following measures: (a) the amount of time the bid request was waiting in the browser's queue ("Stall"), due to several factors such as preemption by requests with higher priority, exhaustion of the allowed number of simultaneous TCP connections (particularly with HTTP/1.0 and HTTP/1.1), and allocation of disk space for caching; (b) time spent in resolving the domain name ("DNS"); (c) time spent in TCP handshake; (d) time spent in TLS handshake; (d) time spent in waiting for the first byte of the response since start of request ("TTFB"); and (d) time spent in receiving the rest of the response ("Response"). We also marked an underlying TLS/TCP connection of a request as *persistent* if the time spent in TCP and TLS handshakes is zero.



(a)

(b)

**Fig. 5.** (a) The gap between the "in-browser" and "on-the-wire" bid request durations suggests room for improving HB implementations. (b) Breakdown of time spent by requests over non-persistent connections into key contributing factors.

In breaking down the request latency to its contributing factors, we separate requests over persistent connections from those over non-persistent connections.

### 6.1 Persistent vs. Non-persistent Connections

Only 60% of the ad requests in the RUM data set were made with persistent connections. They were 34.7% shorter, with a median duration of 230 ms, than those using non-persistent connections. If we break down the latency of such requests into contributing factors, TTFB accounts for 93% and 79% of the total duration, in the median and 80th percentile, respectively. "Response" contributes 2.3% while "Stall" contributes the rest. "Stall" time continues to increase consistently for requests beyond the 80th percentile.

Figure 5b shows the latency breakdown for the remaining 40% of the ad requests made using non-persistent connections; we omitted steps with negligible contributions. The requests take 352 ms in the median and spend, on average, 38% of their time in TCP and TLS handshakes. The handshake times can be reduced to nearly zero if exchanges adopt newer protocols that support low-RTT session resumption such as TCP Fast Open (TFO) [46], TLS 1.3 [49], and QUIC [27]. We tested 228 ad exchanges and found only minimal support for such features: Only 11.4% of the ad exchanges tested support TLS 1.3 and 6.6% support QUIC. We found, however, that 75.9% of the observed IP addresses belonging to ad exchanges support TFO. However, this observation is misleading because even though clients support TFO, they rarely have it enabled (see Appendix A).

Response contributes, on average, 2.4% to the total duration, with a 5 KB median size response from the ad exchanges. TTFB also includes the time spent in conducting the auctions in the exchange and indicates room for improving the exchange-side auctions. Overall, per Fig. 5b, bid durations increase primarily because of increases in time spent across TCP, TLS and TTFB. That TCP, TLS, and TTFB times increase in lockstep suggests RTTs between users and ad exchanges as a key contributor to latency.

### 6.2 Ad Infrastructure Deployments

Using a commercial geolocation service, we calculated the *geodesic* [62] between the end users and the ad exchange servers.[5] Figure 6a plots the CDF of these distances for four of the eight most popular exchanges; we omitted the rest, which had similar results, for clarity. Index Exchange's servers (IND), deployed at 88 different locations are the closest to end users: in the median, the servers are about 180 km away from the users. The remaining exchanges each have servers in only 20 locations and are quite far away from end users—median distances for Rubicon Project (RUB), AOL, and Criteo (CRT) are approximately 520 km, 910 km, and 2410 km, respectively. Criteo seems to be directing almost all North American users to a single US West Coast location. (Appendix B presents other inferences derived from the locations of the users and ad exchanges.)

Index Exchange's geographically widespread deployments help in ensuring a low handshake time, as shown in Fig. 6b. The handshake times to servers of Criteo and AOL, despite the exchanges' comparatively poor deployment, are surprisingly low. We found that Criteo supports TLS 1.3, while Index Exchange does not. This can result in a drastic improvement in handshake latency as TLS 1.3 saves one complete round-trip in the handshake. Another reason that Index Exchange is not seeing even lower latency is that perhaps most of the latency is in the last mile. Since 60% of the bid requests use persistent connections, TTFB, and not handshake time, accounts for most of the request duration. Figure 6c shows that Criteo does an exceptionally good job, especially compared to Index Exchange, in keeping the TTFB low: The server-side auctions at Criteo are perhaps better optimized than those at Index Exchange.

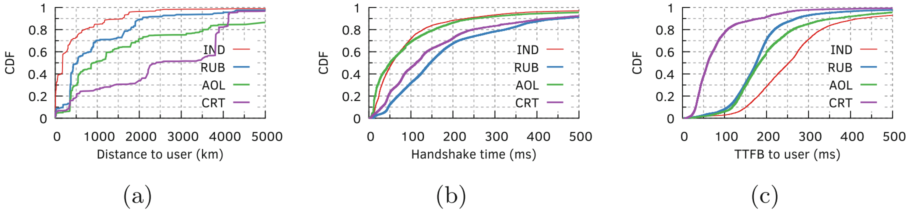


**Fig. 6.** (a) Ad exchanges typically are quite far from end users. (b) TCP/TLS handshakes account for a significant fraction of an ad request's duration. (c) Ad exchanges can quite effectively lower auction durations by optimizing the exchange-side auctions, and lowering the TTFB values.

# 7 Related Work

Header bidding, being a nascent technology, has received little attention in the literature. In [28], Jauvion et al. discuss how to optimize a buyer's bidding strategy in HB, while [45] presents a schotastic optimization model for optimizing

[5] We geolocate the end-user's IP address when the extension reports the opt-in data.

ad-exchange revenues. Cook et al. use machine learning models to identify relationships between data brokers and advertisers [16]. In [35], Pachilakis et al. present a measurement study of the HB platform. They focus on market aspects such as the most popular ad exchanges, number of ad slots found on web pages, and their sizes. They crawl web sites with blank user profiles from a single vantage point, so their revenue and network timing data does not reflect real users and network conditions. They also cannot identify the causes of HB latency. In contrast, our study uses real user measurements to study latency and its ad-revenue implications.

Orthogonal to header bidding, there is a rich body of work on online advertising, end-user tracking, and privacy that show how users attach monetary value to their personally identifiable information (e.g., [11,53]) and how to uncover advertising and tracking services by analyzing network traffic data (e.g., [48]). Venkatadri et al. propose a novel mechanism that enforces transparency on online advertising platforms [56]. Guha et al. and Toubiana et al. have presented designs for privacy preserving advertising that puts the client at the center [24,55]. These techniques, however, require sweeping changes for real-world deployments, and we argue that they can be ported over to the HB platform that is already enjoying widespread adoption.

## 8 Concluding Remarks

Within a span of roughly six years since its introduction, header bidding has gained a strong adoption: Among the top 1k web sites that use third-party ad platforms, 80% use HB. It decreases publishers' dependence on large advertising-market players, e.g., Google, and also improves publisher revenue [52]. Although there are widespread concerns that HB's in-browser auctions introduce significant latency overheads and affect end-users' browsing experiences [17] ([35] mentions high delays seen from one vantage point, and paints a gloomy picture without any analysis on what is causing the delay), our real-end-user measurements lessen these concerns. We showed that more than half of these overheads can be eliminated by adopting more modern protocols and also, perhaps, by fixing bugs in the JavaScript-based HB implementations. Since HB is widely adopted by publishers, shows promise in signficantly increasing the publishers' ad revenues (e.g., see Sect. 4), and has implementation overheads that are addressable with minimal engineering efforts, we propose that client-side HB be seen as an opportunity for privacy-preserving advertising.

The pervasive and commonplace tracking of users to improve targeted ads is unsustainable in the long term. Recent privacy violations and scandals [15,21,51] have raised users' awareness and lowered their tolerances: A recent study found 22% of surveyed users to be using *Adblock Plus*, the most popular ad-blocking browser extension [44], and, fueled by users' demands, Firefox ships bundled with a collection of privacy extensions (e.g., tracker and third-party cookie blocker) [33]. Such aggressive measures to block ads and trackers, nevertheless, is fundamentally at odds with the publishers' business model. Major news web sites have resorted to putting up paywalls [54], and asking for donations [60].

There is, unfortunately, an inherent flaw in today's approach to blocking ads and trackers: ads and trackers are treated equally. While users are sensitive about privacy, most do not mind seeing non-intrusive ads; users would be willing to share more if they had control over what is shared and with whom, and what kind of ads they would like to see [12]. Users also think that ad targeting based on tracking is often inaccurate: they see ads related to common stereotypes about their identity, or related to searches they made over a month ago [12].

The client-side HB platform gives us an opportunity to address these concerns: Since the browser has control over the in-browser auction, it can essentially control the entire ad-fetch process. Browsers must continue to block tracking mechanisms such as host fingerprinting [63] and third-party cookies [20], but could allow HB-based ad requests. They could even append such requests with user-profile data, obviating the exchanges' need for data brokers. The user-profile data could be based on a limited form of profiling or could consist of manually entered preferences as in Privad [24]. Regardless of the approach, the user has complete control and visibility into this data. Privacy-preserving designs for online advertising (e.g., [24,55]) are not novel, but they require sweeping changes for deployment in practice. Given HB's widespread adoption, porting over these techniques to work on top of the HB platform might mitigate the deployment issues.

When implemented correctly, these solutions will limit users' exposure to essentially IP-address-based tracking, which can be alleviated by tunneling the ad requests through a forward proxy operated by neutral or non-profit entitites such as Mozilla or Brave; since these ad requests are encrypted, we do not need to trust the proxy operator. Such public proxies have been operated by Google [2] and Opera [34], albeit for other purposes. We could also incentivize such proxies by devising a revenue-sharing mechanism between the end user, publisher, and the proxy operator using an in-browser cryptocurrency wallet (e.g., MetaMask [29]).

A detailed investigation of such mechanisms will constitute future work. For now, we have shown that HB is already popular and generating higher revenues for publishers, and the perceived latency limitations are addressable, and not fundamental to the protocol. We hope that our insights will encourage both academia and the industry to take a deeper look into header bidding.

## A Client-Side TFO Adoption

In this appendix, we complement the observations on server-side TFO adoption (in Sect. 6.1) with some comments on adoption on the client side. Measuring TFO adoption on the client side is challenging. The Linux kernel disables TFO globally if it sees 3 consecutive TCP timeouts, before or after the handshake, for any destination [13]. The rationale is to avoid the extra cost of TFO failure or client blacklisting in case of middlebox interference [25]. macOS implements a similar backoff strategy and disables TFO [5], although it is a bit less conservative. Windows implements an even more conservative backoff strategy [7].

Even if the operating system has TFO enabled, the browser usually does not. The Chromium project, on which Google Chrome and some other browsers are based, has removed TFO from all platforms [14], while Firefox supports TFO, but keeps it disabled by default.

# B NA and EU Users: GDPR, Ad-Worthiness and Latencies

In this appendix, we examine the role that user location plays in HB. We coarsely divided our users into regions of North America (NA), Europe (EU), Asia (AS), and Oceania (OC), we observe that web sites contact more ad exchanges in North America: 13% of web sites, when visited by users in North America, contact 8 or more ad exchanges, but in case of EU users 99% web sites contact at most 7 (Fig. 7a). Perhaps this effect can be attributed to the strict privacy requirements of GDPR. The difference between European and North American users is even more pronounced when it comes to bid amounts (or CPMs). Web sites generate 4 times more CPM through a visit from a North American user than they do from a European user as shown in Fig. 7b. It is hard to conclusively determine the reason for this large difference as there are a multitude of factors that determine the "ad-worthiness" of a user.

The CDF of `on-the-wire` bid durations for users in different regions (Fig. 7c) shows that, in the $80^{\text{th}}$ percentile, European (EU) users observe 12% higher bid durations than North American (NA) users. The auction durations for NA users are, however, 27% longer than that of their EU counterparts in the $80^{\text{th}}$ percentile (Fig. 8a). These observations can perhaps be attributed to NA users contacting more exchanges, and that, as we have seen earlier in Fig. 3c, increases auction duration. Bid durations for Oceania (OC) users are alarmingly high: 23% of bids take longer than 1 s (Fig. 7c), which precipitates in long auctions for OC users (Fig. 8a). Only 7% auctions of OC users take, however, longer than 2.5 s compared to 10% of auctions in case of NA users. For a large fraction of OC users, even though bids arrive late, the JavaScript perhaps times out and terminates the auction, potentially introducing some loss of ad revenue for publishers.

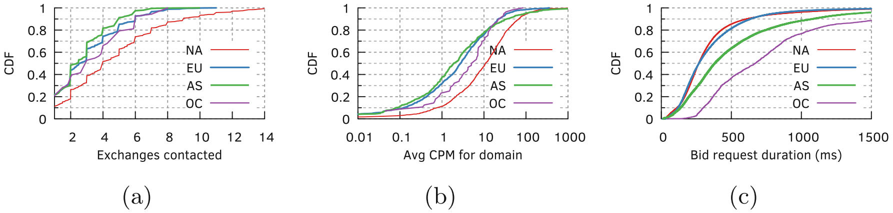
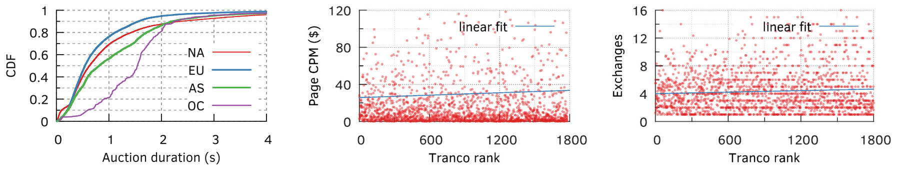


**Fig. 7.** Impact of a user's location on (a) the number of exchanges contacted, (b) the mean CPM obtained per web page, and (c) bid-request durations.



(a) (b) (c)

**Fig. 8.** (a) Impact of user's location on auction duration, and the impact of a web-site's ranking on (b) mean CPM and (c) number of exchanges contacted.

## C Popularity Correlations

We investigate, in this appendix, how the popularity ranking of a web site affects its HB implementation and the CPM it receives on its ad slots. For popularity rankings, we used the Tranco list [38], a stable top list hardened against manipulation. We used the relative ranks of second-level domains observed in our measurements and filtered out web sites that have fewer than 10 data points.

Figure 8b shows the mean CPM per web-page visit, of a given web site, as a function of that site's relative Tranco rank. The linear fit, with a slope of 0.008, reveals a weak correlation, suggesting that web-site popularity is not a strong indicator of "high-value" audience for advertisers. For instance, `imgur.com` (rank 51), an image-sharing web site outranks `wsj.com` (rank 152), a major business-focused publication.

Increasing the number of ad exchanges contacted increases the auction duration, which may have implications for end-users' browsing experiences (refer Sect. 5). Figure 8c shows, however, no correlation between the rank of a web site (based on Tranco) and the number of ad exchanges it contacts: Popular web sites do not contact fewer exchanges than unpopular ones to improve user experience.

We also repeated these analyses with the Majestic Million top list[6] instead of Tranco. Majestic Million ranks web sites by the number of subnets linking to them, which is more of a quality measure than raw traffic. Regardless, we did not observe any significant change in the results and inferences presented above.

## References

1. Adzerk: Ad Tech Insights - August 2019 Report, August 2019. https://adzerk.com/assets/reports/AdTechInsights_Aug2019.pdf
2. Agababov, V., et al.: Flywheel: Google's data compression proxy for the mobile web. In: NSDI (2015)
3. Akamai: Akamai "10For10", July 2015. https://www.akamai.com/us/en/multimedia/documents/brochure/akamai-10for10-brochure.pdf

[6] https://majestic.com/reports/majestic-million.

4. Anthes, G.: Data brokers are watching you. Commun. ACM **58**(1), 28–30 (2014)
5. Apple Open Source: tcp_cache.c, September 2017. https://github.com/opensource-apple/xnu/blob/master/bsd/netinet/tcp_cache.c
6. Aqeel, W.: MyAdPrice: an ad-tracking extension for Chrome and Firefox, January 2020. https://myadprice.github.io/
7. Balasubramanian, P.: Updates on Windows TCP, July 2017. https://datatracker.ietf.org/meeting/100/materials/slides-100-tcpm-updates-on-windows-tcp-00
8. Barreda-Ángeles, M., Arapakis, I., Bai, X., Cambazoglu, B.B., Pereda-Baños, A.: Unconscious physiological effects of search latency on users and their click behaviour. In: Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2015 (2015)
9. Benes, R.: 'An ad tech urban legend': an oral history of how header bidding became digital advertising's hottest buzzword, June 2017. https://digiday.com/media/header-bidding-oral-history/
10. Bozkurt, I.N., et al.: Why is the Internet so slow?!. In: Kaafar, M.A., Uhlig, S., Amann, J. (eds.) PAM 2017. LNCS, vol. 10176, pp. 173–187. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-54328-4_13
11. Carrascal, J.P., Riederer, C., Erramilli, V., Cherubini, M., de Oliveira, R.: Your browsing behavior for a big mac: economics of personal information online. In: WWW (2013)
12. Chanchary, F., Chiasson, S.: User perceptions of sharing, advertising, and tracking. In: Eleventh Symposium On Usable Privacy and Security (SOUPS), July 2015
13. Cheng, Y.: Pause fast open globally after third consecutive timeout. https://patchwork.ozlabs.org/patch/847640/. Accessed 16 Oct 2019
14. Chromium bugs: TCP fast open not supported on Windows 10 build 1607. https://bugs.chromium.org/p/chromium/issues/detail?id=635080. Accessed 16 Oct 2019
15. Confessore, N.: Cambridge Analytica and Facebook: The Scandal and the Fallout So Far. The New York Times, April 2018
16. Cook, J., Nithyanand, R., Shafiq, Z.: Inferring tracker-advertiser relationships in the online advertising ecosystem using header bidding. CoRR abs/1907.07275 (2019)
17. Davies, J.: Beware of page latency: the side effects to header bidding, September 2016. https://digiday.com/uk/beware-page-latency-side-effects-header-bidding/
18. DeWitt, G.: Improperly implemented header bidding tags cause page slowdown and decreased revenue for publishers, May 2017. https://www.indexexchange.com/improperly-implemented-header-bidding-tags-cause-page-slowdown-and-decreased-revenue-for-publishers/
19. Enberg, J.: Global Digital Ad Spending 2019, March 2019. https://www.emarketer.com/content/global-digital-ad-spending-2019
20. Englehardt, S., et al.: Cookies that give you away: the surveillance implications of web tracking. In: WWW (2015)
21. Gartenberg, C.: Seized documents reveal that Facebook knew about Russian data harvesting as early as 2014, The Verge, November 2018
22. Google: Latency Restrictions and Peering. https://developers.google.com/ad-exchange/rtb/peer-guide. Accessed 12 Oct 2019
23. Google: The arrival of real-time bidding, June 2011. https://www.rtbchina.com/wp-content/uploads/2012/03/Google-White-Paper-The-Arrival-of-Real-Time-Bidding-July-2011.pdf
24. Guha, S., Cheng, B., Francis, P.: Privad: practical privacy in online advertising. In: NSDI (2011)

25. Hesmans, B., Duchene, F., Paasch, C., Detal, G., Bonaventure, O.: Are TCP extensions Middlebox-proof? In: HotMiddlebox (2013)
26. da Hora, D.N., Asrese, A.S., Christophides, V., Teixeira, R., Rossi, D.: Narrowing the gap between QoS metrics and web QoE using above-the-fold metrics. In: PAM (2018)
27. Iyengar, J., Thomson, M.: QUIC: A UDP-Based Multiplexed and Secure Transport. Internet-draft, Internet Engineering Task Force, September 2019
28. Jauvion, G., Grislain, N., Dkengne Sielenou, P., Garivier, A., Gerchinovitz, S.: Optimization of a SSP's header bidding strategy using Thompson sampling. In: KDD (2018)
29. Lee, W.M.: Using the MetaMask Chrome extension, pp. 93–126. Apress (2019)
30. MDN web docs: Introduction to the DOM, May 2019. https://developer.mozilla.org/en-US/docs/Web/API/Document_Object_Model/Introduction
31. MDN web docs: JavaScript APIs for WebExtensions, March 2019. https://developer.mozilla.org/en-US/docs/Mozilla/Add-ons/WebExtensions/API
32. Meeker, M.: Internet Trends 2019, June 2019. https://www.bondcap.com/pdf/Internet_Trends_2019.pdf
33. Mozilla Firefox: Content blocking. https://support.mozilla.org/en-US/kb/content-blocking. Accessed 16 Oct 2019
34. Opera: Data savings and turbo mode. https://www.opera.com/turbo. Accessed 16 Oct 2019
35. Pachilakis, M., Papadopoulos, P., Markatos, E.P., Kourtellis, N.: No more chasing waterfalls: a measurement study of the header bidding ad-ecosystem. In: IMC (2019)
36. Pandey, P., Muthukumar, P.: Real-Time Ad Impression Bids Using DynamoDB, April 2013. https://aws.amazon.com/blogs/aws/real-time-ad-impression-bidsusing-dynamodb/
37. Papadopoulos, P., Kourtellis, N., Markatos, E.P.: The cost of digital advertisement: comparing user and advertiser views. In: WWW (2018)
38. Pochat, V.L., Goethem, T.V., Tajalizadehkhoob, S., Korczyński, M., Joosen, W.: Tranco: a research-oriented top sites ranking hardened against manipulation. NDSS (2019)
39. Prebid: A brief history of header bidding. http://prebid.org/overview/intro.html#a-brief-history-of-header-bidding. Accessed 9 Oct 2019
40. Prebid: Header Bidding Made Easy. http://prebid.org/index.html. Accessed 10 Oct 2019
41. Prebid: How to add a new bidder adapter, http://prebid.org/dev-docs/bidder-adaptor.html. Accessed 14 Oct 2019
42. Prebid: How to reduce the latency of header bidding with Prebid.js. http://prebid.org/overview/how-to-reduce-latency-of-header-bidding.html. Accessed 12 Oct 2019
43. Prebid: Prebid.org members . http://prebid.org/partners/partners.html. Accessed 15 Oct 2019
44. Pujol, E., Hohlfeld, O., Feldmann, A.: Annoyed users: ads and ad-block usage in the wild. In: IMC (2015)
45. Qin, R., Yuan, Y., Wang, F.: Optimizing the revenue for ad exchanges in header bidding advertising markets. In: 2017 IEEE International Conference on Systems, Man, and Cybernetics (SMC), October 2017
46. Radhakrishnan, S., Cheng, Y., Chu, J., Jain, A., Raghavan, B.: TCP fast open. In: CoNEXT, December 2011

47. Ramirez, E., Brill, J., Ohlhausen, M.K., Wright, J.D., McSweeney, T.: Data brokers: a call for transparency and accountability. Technical report, United States Federal Trade Commission, May 2014
48. Razaghpanah, A., et al.: Apps, trackers, privacy, and regulators: a global study of the mobile tracking ecosystem. In: NDSS (2018)
49. Rescorla, E.: The Transport Layer Security (TLS) Protocol Version 1.3. RFC 8446, August 2018
50. Sluis, S.: The rise of 'header bidding' and the end of the publisher waterfall, June 2015. https://adexchanger.com/publishers/the-rise-of-header-bidding-and-the-end-of-the-publisher-waterfall/
51. Snowden, E.J.: Permanent record. Metropolitan Books (2019)
52. Sovrn: The Past, Present, and Future of Header Bidding, February 2017. https://www.sovrn.com/blog/header-bidding-grows-up/
53. Staiano, J., Oliver, N., Lepri, B., de Oliveira, R., Caraviello, M., Sebe, N.: Money walks: a human-centric study on the economics of personal mobile data. In: Proceedings of the 2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing, UbiComp 2014 (2014)
54. The Times Open Team: We Re-Launched The New York Times Paywall and No One Noticed, Times Open, August 2019
55. Toubiana, V., Narayanan, A., Boneh, D., Nissenbaum, H., Barocas, S.: Adnostic: privacy preserving targeted advertising. In: NDSS (2010)
56. Venkatadri, G., Mislove, A., Gummadi, K.P.: Treads: transparency-enhancing ads. In: HotNets (2018)
57. Videology Knowledge Lab: Header bidding: a byte-sized overview, October 2015. https://www.iab.com/wp-content/uploads/2015/10/VidLab-HeaderBidding-3.27.17V10.pdf
58. W3C: Navigation Timing, December 2012. https://www.w3.org/TR/navigation-timing/
59. W3C: A Primer for Web Performance Timing APIs, April 2019. https://w3c.github.io/perf-timing-primer/
60. Jim, W.: More than a million readers contribute financially to the Guardian. The Guardian, November 2018
61. Wikipedia: Cost per mille. https://en.wikipedia.org/wiki/Cost_per_mille. Accessed 9 Oct 2019
62. Wikipedia: Geodesic. https://en.wikipedia.org/wiki/Geodesic. Accessed 29 Oct 2019
63. Yen, T.F., Xie, Y., Yu, F., Yu, R.P., Abadi, M.: Host fingerprinting and tracking on the web: privacy and security implications. In: Proceedings of the Network and Distributed System Security Symposium (NDSS) 2012, February 2012