No More Chasing Waterfalls: A Measurement Study of the Header Bidding Ad-Ecosystem

Michalis Pachilakis
University of Crete / FORTH, Greece
mipach@ics.forth.gr

Panagiotis Papadopoulos
Brave Software
panpap@brave.com

Evangelos P. Markatos
University of Crete / FORTH, Greece
markatos@ics.forth.gr

Nicolas Kourtellis
Telefonica Research, Spain
nicolas.kourtellis@telefonica.com

ABSTRACT

In recent years, Header Bidding (HB) has gained popularity among web publishers, challenging the status quo in the ad ecosystem. Contrary to the traditional waterfall standard, HB aims to give back to publishers control of their ad inventory, increase transparency, fairness and competition among advertisers, resulting in higher ad-slot prices. Although promising, little is known about how this ad protocol works: What are HB's possible implementations, who are the major players, and what is its network and UX overhead?

To address these questions, we design and implement *HBDetector*: a novel methodology to detect HB auctions on a website at real-time. By crawling 35,000 top Alexa websites, we collect and analyze a dataset of 800k auctions. We find that: (i) 14.28% of top websites utilize HB. (ii) Publishers prefer to collaborate with a few Demand Partners who also dominate the waterfall market. (iii) HB latency can be significantly higher (up to 3× in median case) than waterfall.

CCS CONCEPTS

•**Information systems → Online advertising; Display advertising;** *Web log analysis;*

KEYWORDS

Header Bidding, Digital Advertising, Waterfall, RTB

1 INTRODUCTION

The largest portion of the digital advertisements we receive today on the Web follows a *programmatic ad-purchase* model. Upon a website visit, a real time auction gets triggered, usually via the real-time bidding (RTB) protocol [27], for each and every available

ad-slot on the user's display. These auctions are hosted in remote marketplace platforms called Ad Exchanges (ADXs) that collect the bids from their affiliated Demand Site Platforms (DSPs). The highest bidder wins, and delivers its impression to the user's display.

However, there are more than one ad networks that can provide bids for an ad-slot. In the traditional standard for ad-buying, called *waterfalling*, the different ad networks (e.g., ADXs with their affiliated DSPs) are prioritized in hierarchical levels [32]. Thus, when there is no bid from ad network #1, a new auction is triggered for ad network #2, and so forth. Of course, apart from the auction-based ad purchase, there are still other non-programmatic channels like *direct orders* from advertisers who run static campaigns for a certain number of impressions [12]. Through these channels, advertisers target not a user but the entire audience of a specific website (e.g., an ad regarding Super Bowl on espn.com). Alternatively, if there is neither a direct order nor a bid in these auctions, the ad-slot may be filled via another channel for remnant inventory called *fallback or backfill* (e.g., Google AdSense) [25].

The process of ad prioritization among the above different channels and ad networks in waterfall is managed through the publisher's ad server or Supply Side Platform (SSP) (e.g., DoubleClick for Publishers (DFP)). Priorities are typically set not at real time but based on the average price of the past purchases for each channel. As a consequence, in waterfall not all ad partners have the ability to compete simultaneously. Therefore, *the publishers do not get the optimal charge price*, since an ad-slot may not be sold at the highest price (e.g., if the winning bid in the auction of ad network #1 is 0.2\$, the ad-slot will be sold even if there was a bid of 0.5\$ in ad network #2). Apart from the potential loss of revenue for the publishers, there is also a *significant lack of transparency*. Except from the winning bidder, the publishers do not know who else placed a bid for their ad-slot and for how much. In addition, the lack of control restricts the publishers from choosing Demand Partners, or different sale channels in real time (e.g., to get a high price through RTB when the quota of direct ads sold has not yet been depleted).

To remedy all the above, *Header Bidding* [55] (or *parallel bidding* in mobile apps [42]) has been recently proposed and has started to gain wide acceptance among publishers [20, 23, 36, 52]. As depicted in Figure 1, HB is a different auction that takes place not on the ad server as in waterfall, but inside the header field of a HTML page, before anything else is loaded on the page. It allows a publisher to simultaneously get bids from all sale channels (e.g., direct orders, programmatic auctions, fallback) and Demand Partners (e.g., DSPs, ADXs, ad agencies). HB not only gives the control back to the
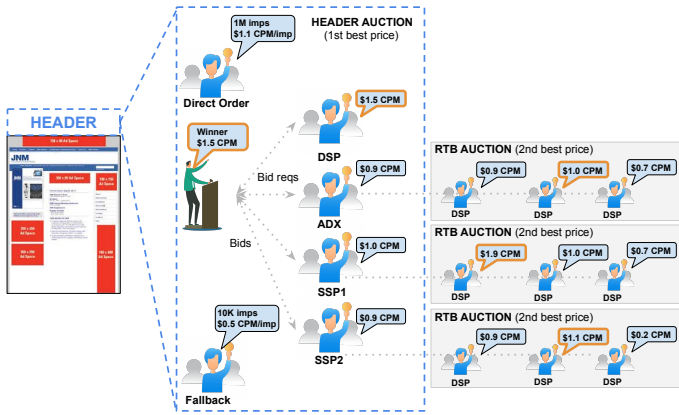
Figure 1: High level overview of the HB. The absence of priorities aims to provide (i) fairness and higher competition among advertisers and (ii) increased revenue for the publishers.

publisher but also allows higher revenues than waterfall, since it guarantees that the impressions with the higher price will be bought and rendered [10]. On the advertiser's side, HB promotes fairness since there are no priorities. Consequently, any advertiser could win any auction, as long as it bids higher than others. HB enables small advertisers to also be competitive, compared to big advertisers who would have higher priority on the waterfall model.

Although there is a lot of research regarding the waterfall standard [5, 6, 34, 35, 41], we know very little about the innovative and rapidly growing alternative of HB. How is it implemented? What is the current adoption of HB on the Web? What is the performance overhead and how it affects the page rendering time? How many bids the average publisher can receive? What are the average charge prices and how do these compare to the ones of the waterfall standard? Which are the big players and how is the market share divided?

To respond to all these questions, we study the different existing implementations of HB and we design *HBDetector*: a novel methodology to detect HB auctions on the Web. Our approach aims to increase transparency on the ad-ecosystem, by exposing at real-time the internals of the new and rapidly growing HB ad protocol: in which sites it exists, the prices and partners involved, etc. Using *HBDetector*, we crawl a number of popular websites, we collect a rich dataset of HB-enabled websites. Our tool helps us detect particular browser events triggered by the HB libraries embedded in such webpages, along with the ad partners participating in the HB and metadata for the auctions executed on these websites. We analyze and present the first full-scale study of HB aiming to shed light on how this innovative technology works and investigate the trade-off between the overhead it imposes on the user experience and the average earnings it brings to publishers. In this paper, we make the following main contributions:

(1) We propose and implement *HBDetector*, the first of its kind Web transparency tool, capable of detecting HB activity at real-time, on the Web. We provide it as an open-sourced browser extension for Google Chrome[1] .
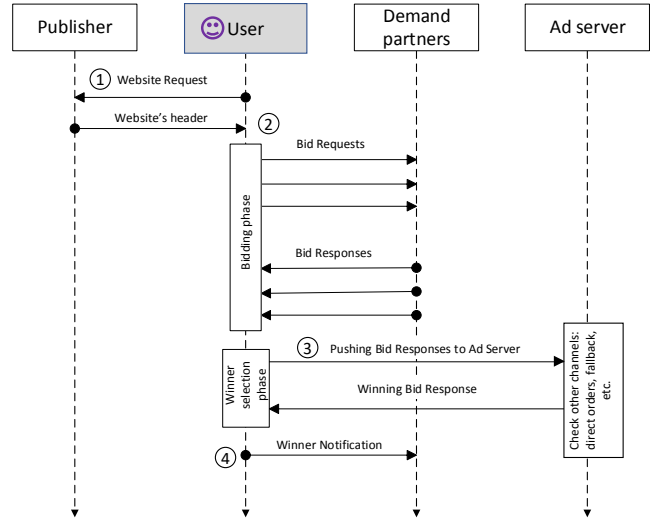
---

[1]https://www.github.com/mipach/HBDetector

Figure 2: Flow chart of the Header Bidding protocol.

(2) By running *HBDetector* across 35,000 top Alexa websites, we collect a dataset of 800k auctions. This work is the first to analyze the adoption and characteristics of HB.

(3) We extract a set of lessons on HB: (i) There are 3 different implementations of HB: Client-side, Server-side and Hybrid HB. (ii) There is at least 14.28% of top websites that use HB. (iii) Publishers tend to collaborate with a small number of Demand Partners, which are already reputable in the waterfall standard. (iv) HB latency can be significantly higher (up to 3× in the median case, and up to 15× in 10% of cases) than waterfall.

2 BACKGROUND ON HEADER BIDDING

In this section, we cover background knowledge required for our study regarding the most important aspects of HB.

2.1 HB Protocol Description

Contrary to the traditional waterfall standard, in HB the ad auction does not take place in a remote ADX, but on the user's browser. The HB process, depicted in Figure 2, is the following:

Step 1: When a user visits a website, the HTML page is fetched. As soon as the header of the HTML is rendered in the browser, user tracking code and the third-party library responsible for the procedure of the HB is loaded as well.

Step 2: Then, the HB library sends (in parallel) HTTP POST requests to the Demand Partners (e.g., DSPs, ad agencies, ADXs which conduct their own RTB auctions) requesting for bids. These bid requests also include information about the current user (such as interests and cookies). Such information can be used by the Demand Partners to decide whether and how much they will bid for an ad-slot in the particular user's display. Note, that if a Demand Partner does not respond within a predefined time threshold, its bid is considered late and not taken into account.

Step 3: As soon as the Demand Partners respond with their bids (and their impressions), the collected responses are sent to the publisher's ad server. The ad server will check the received bids and compare with the floor price agreed with the publisher, to

decide if the received prices are high enough [54]. If the floor price is met, the HB process was successful and the ad-slot is satisfied. Alternatively, the ad server can check the rest of the programmatic (or not) available channels (e.g., direct order, RTB, fallback) and will find the best next option for the specific ad-slot. This step entails communicating with SSPs for available direct orders which can provide higher revenues to the publisher than regular RTB auctions. The ad server can also communicate with Demand Partners for RTB auctions, or other SSPs who can provide fallback ads, such as Google AdSense, or even house ads.

Step 4: As soon as the impression is rendered on the user's display, a callback HTTP request notifies the winning Demand Partner that its impression was rendered successfully on the user's browser, and the ad price that was charged (winner notification).

In theory, with this new protocol, the publisher has total control over the ad inventory they provide, knowing exactly how much the Demand Partners value each slot, and the actual amount of money they are willing to pay for it. In addition, there is full transparency, since the publisher can have access to all bids and decide at real time the best strategy it should follow without the need to trust any intermediaries. In the future, HB could provide the means to publishers to reduce advertising that is not suitable for, or does not match the semantics of their websites, and even curb malvertising. However, as we will show later in Section 4, this transparency and control is not always applicable under the various types of HB we have detected.

2.2 HB Implementation & Performance

To implement the above protocol, publishers need to include HB third-party libraries in their webpages. Although there does not exist a common standard for HB yet, the great majority of publishers use the open-source library of `Prebid.js` [45], supported by all major ad companies. This library includes: (i) The core component which is responsible to issue the bid requests and collect the responses, which are later sent to the publisher's ad server. (ii) The adapters which are plugged into the core and provide all necessary functionality required for each specific Demand Partner. `Prebid.js` is supported by more than 200 Demand Partners (e.g., AppNexus, Criteo, OpenX, PulsePoint) that provide their own adapters [44].

We note that in traditional waterfall, the auction information is opaque to the client and the only information that can be inferred (if at all) is through the parameters of the notification URL (which acts as a callback to the winning bidder). In contrast, in the HB, and due to bidder responses, browser DOM events are triggered that contain metadata directly available at the user browser, and can be used to clearly distinguish between waterfall and HB activity.

The non-hierarchical model of HB produces much more network traffic than the waterfall standard. Indeed, HB sends one request for each and every collaborating Demand Partner. This can result to an increased page latency, especially when some Demand Partners take too long to respond. To make matters worse, as soon as they receive a bid request, some of these Demand Partners may run their own auctions inside their ad network, with their own affiliated bidders (as depicted in Figure 1). This increased page latency raises significant concerns. Indeed, 40% of the publishers already mention

that such latency is capable of impacting their users' browsing experience [8, 9, 15].

It is worth noting, that HB technology is still in its early stages and many ad networks are technologically not ready to move completely from the waterfall model to participate in this new model. In order not to miss bids from such networks, some ad mediators (e.g., Appodeal) mix the two techniques in an attempt to provide waterfall compatibility during this transitional period [42].

3 METHODOLOGY FOR MEASURING HB

In this section, we outline our methodology for detecting HB on webpages, and our effort to crawl top Alexa websites for HB activity.

3.1 Detection Mechanism

In order to detect if a webpage is using HB for delivering ads to its users, we need to detect HB-related activity originating from the said webpage. As explained above, the HB activity is performed over different channels than ad protocols such as RTB, using a library (implemented in JavaScript) embedded in the header of the page. Therefore, by monitoring the events triggered by such libraries, we can confidently distinguish HB activity from other models such as waterfalling.

There are three main ways to detect if HB is present in a webpage:

(1) Perform static analysis of the page and identify tags of scripts that load known HB libraries.
(2) Detect DOM-related events that are triggered due to HB embedded in the webpage.
(3) Detect web requests sent from the page to HB entities.

The first method is straightforward to implement with the following steps: Download the webpage source code and use regular expressions to detect all known HB libraries. However, we note that just detecting these libraries is not enough, as false positives or false negatives could occur. For example, static analysis is prone to false positives such as non HB libraries being misnamed using HB-related names, or HB-related libraries appearing in the HTML code but not executed Similarly, static analysis is vulnerable to false negatives such as renamed HB libraries to names that are not known yet, or new HB libraries that do not match our HB-related keywords from known libraries. To avoid such potential false positives and negatives, we chose not to use static analysis in the *HBDetector*.

The second method is more difficult to implement, but offers better detection rates with reduced false positives and negatives, and thus, harder to evade. This method monitors the DOM events that are triggered in a webpage, events that are sent to notify the code of interesting activity that has taken place on the page. Events can represent everything from basic user interactions to automated notifications happening on the page. Most HB libraries trigger events in several phases of an auction (initiation of the auction, bid collection, winning bidder, etc.). If such an event is detected, we are certain that it is because of HB. Even better, by "tapping" on these events [33], we can collect information about HB that the first method is not able to detect.

The third method is similar to the second, but operates at a different level in the browser: monitor the web requests of a page in real-time, and detect all the request sent to and received from known HB Demand Partners. By constructing a list containing all
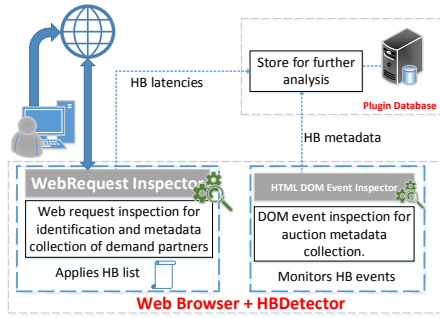
Figure 3: Overview of the *HBDetector* mechanism. After the user accesses a webpage, all the incoming and outgoing WebRequests are inspected to detect HB partners. A content script is also injected in the header of the webpage to detect HB events about the auction performed.

the known Demand Partners, we can check all the incoming and outgoing WebRequests to the browser, and keep the relevant to HB.

In this paper, we implemented *HBDetector*, a tool which combines the 2nd and 3rd methods to increase detection performance. An overview of the tool is illustrated in Figure 3. *HBDetector* adds a content script in the header of each webpage when the page is loaded. This script monitors the webpage's activity for various events and requests sent and received by the page, keeping the ones relevant to HB (e.g., incoming responses from DSPs for HB auctions). When such DOM events are triggered (which is a first sign of HB activity), the tool filters the web requests triggering these events by checking the parameters included in them. HB libraries use predefined parameters such as "*bidder*", "*hb_partner*", "*hb_price*", etc., which are not used by other ad-protocols such as RTB. Thus, the tool keeps all web requests that triggered a DOM event of a HB library, and also include HB parameters. It then proceeds to extract the values from these parameters for analysis. These parameters are typically fixed for each HB library, and all HB partners must use them as such, to participate successfully in HB auctions with that library. In contrast, in the RTB protocol, the parameter names used in the notification URLs are DSP dependent and do not utilize DOM events.

From the available HB libraries, we examined `prebid.js` (and its variants), being the most famous one (64% of client-side wrappers are built on prebid [28]), as well as `gpt.js` and `pubfood.js` libraries for their available codebase and/or documentation. We decided to focus our more in-depth reverse-engineering on `prebid.js` due to its popularity, available documentation and open-source code and APIs [24, 43]. By performing code and documentation analysis for the HB libraries that have such material available, we identified the following list of HB events that our tool can detect:

- *auctionInit*: the auction has started
- *requestBids*: bids have been requested
- *bidRequested*: a bid was requested from specific partner
- *bidResponse*: a response has arrived
- *auctionEnd*: the auction has ended
- *bidWon*: a bid has won
- *slotRenderEnded*: the ad's code is injected into a slot
- *adRenderFailed*: an ad failed to render

Table 1: Summary of collected data by crawling top Alexa webpages using the *HBDetector* for HB-related activity.

Data	Volume
# of websites crawled	35,000
# of websites with HB	4998
# of auctions detected	798,629
# of bids detected	241,392
# of competing Demand Partners	84
# weeks of crawling	5

In this work, we focus on three of these events: *auctionEnd*, *bidWon*, and *slotRenderEnded*. The *auctionEnd*, as its name states, is triggered after the auctions for the ad-slots have finished, i.e., the Demand Partners have submitted their offers. The *bidWon* event is triggered after the winning Demand Partner has been determined. Finally, the *slotRenderEnded* event is triggered when an ad has finished rendering successfully on an ad-slot. By analyzing these events, which can only be triggered by HB activity and not other libraries, we were able to collect several metadata about the auctions, such as the Demand Partners who bided, the ones who won, the CPM (cost per million impressions in USD) spent, the ad size, currency, dimensions, etc.

We also constructed a list with all the known HB Demand Partners. We collected and combined several lists used by HB tools designed to help publishers fine tune their HB on their websites. Using this list, we can infer all the *WebRequests* about HB without altering them, in order to detect when a request to a Demand Partner is sent, and when an answer is received. The *HBDetector* is written in a few hundred lines of JavaScript as a Google Chrome browser extension.

***HBDetector* limitations:** The tool does not analyze all libraries used by the HB ecosystem due to unavailability of documentation and/or code. Also, it cannot capture new DOM events if they get added to existing libraries it is analyzing. Finally, it cannot capture current DOM events if the events change format or parameters they are using. In addition, the tool does not capture waterfalling RTB activity, and therefore, does not allow direct comparison of the two protocols with respect to Demand Partners involved, ad-prices, etc. We plan to address these limitations in a future version of the tool.

3.2 Data Crawling

We used our tool to detect which websites employ HB, by crawling a set of websites, based on a large top list purchased from Alexa [3] on 01/2017. Given the changes anticipated in such website ranking list and especially in its long tail [48], we focus on the head of the Alexa list, to capture a more stable part of the ranking distribution through time. Due to equipment, network and time costs, we limited this list to 35,000 domains to crawl per day, during Feb'19. To confirm the representativeness of this older list, we compared it with the top 35k domains in 2017-2019 from [48], and found that it has an overlap of 78.36%(06/2017), 62.10%(06/2018), 58.36%(02/2019) and 55.34%(06/2019).

We used selenium and chromedriver loaded with *HBDetector* in order to automate the crawling. We initiated a clean slate instance before visiting each website, in order to keep the crawling process stateless (no previous history, no cookies, no user profile). When a
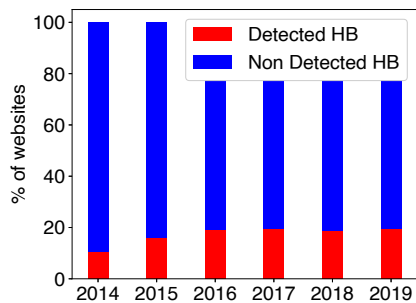
Figure 4: Header Bidding adoption in the last six years for the top 1k Alexa websites of every year.

webpage is visited, the crawler waits for the page to be completely loaded, and then allows an extra five seconds, in case additional content needs to be downloaded or pending responses to be concluded. We set the page load timeout to 60 seconds, so that if the page is not fully loaded in one minute, the crawler proceeds to the next webpage in the list, after killing the previous instance and initiating a new, clean instance.

With this crawling process, we detected HB in ~5,000 (14.28%) of the websites, in a well-distributed fashion. In particular, HB was found in 20-23% of the top 5k websites, 12-17% for the top 5k-15k, and 10-12% for the rest. Indeed, new top websites not included in this 35k list may have already adopted HB, leading to an underestimation of today's adoption rate. However, as found in our results in Sec. 4.1 were we use top 1k Alexa lists for 6 years, we show similar HB adoption rate with the head of the top 35k list, giving credence to our results. Then, we crawled these 5k websites every day for a period of 34 days in Feb'19, collecting metadata about the HB auctions, and performance exhibited from the various websites using HB. In Table 1, we provide a summary of the data collected.

We note that we detected 800k auctions but received 241k bids. One could expect that each auction should have at least a bid. Indeed this would be the case if actual users were involved and Demand Partners were interested in them. However, there are cases where bidders may avoid bidding when they know nothing about the user. In our case, we are interested in the vanilla case using a clean state crawler and no real user profiles.

4 THE 3 FACETS OF HEADER BIDDING

In this section, we analyze the crawled data and present results and observations we have made about the HB adoption over time and types of HB we identified from our exploration.

4.1 Header Bidding Adoption

Since this is a new programmatic ad-protocol (standardized in 2014[7]), we explore the general adoption of HB through the last 6 years. To do that, we downloaded snapshots of selected lists of webpages using the *Wayback Machine* [29]. Due to the involved network and time cost to crawl from the Wayback Machine, we focused on the top 1,000 publishers based on Alexa rankings, made available in a recent study [48] and https://toplists.github.io/. The list of top publishers was selected on a fixed day per year (6/6/2019,

6/6/2018, etc.). Since these historical webpages were static, we performed a static analysis looking for HB libraries and components in their websites' code. Someone could also try an analysis using the *HBDetector*, by attempting to render each website, or even fingerprinting the libraries. However, such analyses: i) Take more time to execute than static analysis. ii) The webpage must be renderable and its components must work (scripts should be downloadable, scripts should not fail to run, the page should not call unresponsive servers, etc.). Therefore, dynamic analysis cannot be applied on historical pages "played back", with potentially deprecated libraries or other scripts embedded, third-party partners not responding, etc., and expect 100% correctness on the results collected.

Figure 4 shows the yearly breakdown of HB found in these websites. Interestingly, we observe a steady increase of the HB adoption. About 10% of these websites were early adopters and started using HB 6 years ago. After the breakthrough of 2016, when HB became popular[49], there is a steady 20% of the websites using this ad protocol. These adoption rates, and the general rate of 14.28% in the 35k list, match industry-claimed numbers of ~15% in the last 15 months (14.66% in Jan'18 - 15.84% in March'19, computed for the top 1k out of 5k top Alexa websites that serve programmatic ads) for the US market [19, 28].

We note that *HBDetector* catches 100% of the HB activities for the libraries analyzed. Indeed, there are websites which could be using HB libraries that we didn't analyze at the time of data collection, and therefore were not flagged as HB-enabled websites. This means we get 100% precision but not 100% recall. However, the HB adoption experiment using the 1k lists shows a rate that aligns with the overall HB adoption rate in the 35k list, and these two rates closely match what industry is claiming. These observations point to low false positive and negative rates, and that the data collected by *HBDetector* (i.e., using dynamic analysis) have high recall rate and provide a representative picture of the HB ecosystem at the time of each crawl.

4.2 Types of Header Bidding Detected

Our in-depth investigation of the HB ecosystem and the data collected revealed that this new programmatic ad protocol is currently being deployed in three facets: (i) Client-Side HB, (ii) Server-Side HB, and (iii) Hybrid HB. This finding matches the 3 types of HB wrappers (client-side, server-side and hybrid) suggested by industry reports [26]. In the Client-Side HB and Hybrid HB models, the ad auctions are transparent, so we can distinguish them with a high degree of certainty due to the events sent and received by the browser. On the other hand, on Server-Side HB model it is less clear, since most of the ad-related actions happen at the server. However, after inspecting the responses received by the browser, we can discover the parameters referring to HB (e.g. hb_partner, hb_price, etc.). Next, we analyze each facet, including the steps taken for the protocol's execution, and potential consequences it may have.

4.3 Client-Side HB

In Client-Side HB, as the name implies, the HB process happens in the user's browser. As illustrated in Figure 5, during this HB type, the user's browser executes 8 steps, including the initiation of the
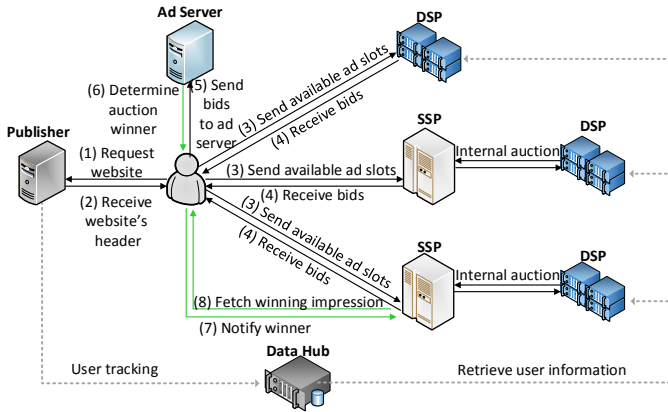
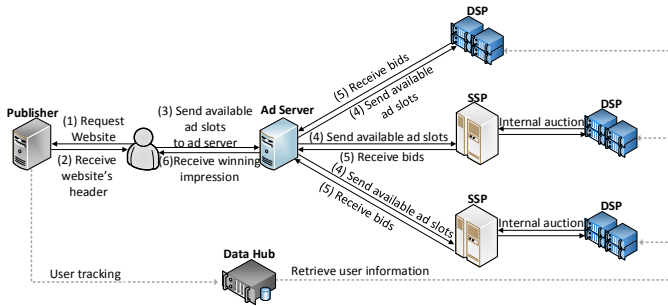Figure 5: Client-Side HB overview and steps followed.

Figure 6: Server-Side HB overview and steps followed.

HB auction, receiving of bids from Demand Partners and notifying the winning partner. Client-Side HB's main goal is to improve fairness and transparency. Publishers can choose the Demand Partners they want to collaborate with, regardless of their market cap. What matters is if their bids are competitive enough. Also, because the whole HB process is performed at the client side, and then sent to the publisher's ad server, it is completely transparent to the publisher and, in theory, to the user.

The publisher can know at any time which partners bid, for which ad-slots they were interested, how much they were willing to pay, etc. On the down side, Client-Side HB is harder to set up. Publishers need to have good technical understanding to set up and tune their HB library. Also, they need to operate their own ad server, a task which is not trivial. Finally, because of the increased number of messages to be exchanged, or due to a bad configuration in the HB library, longer latencies may be observed.

From the regular end-user's point of view, the only thing that can be observed is an increased latency for the loading of the webpage when it employs Client-Side HB. However, the regular user cannot be aware of all the HB (and other ad-tech) activity happening in the background. This is where our *HBDetector* tool can help increase transparency of the protocol from the point of view of the end-user, and measure non-obvious aspects such as the communication and time overhead for the browser during HB, winning bids, etc.

4.4 Server-Side HB

In Server-Side HB, a single request is sent to a Demand Partner's server, which is responsible to do the whole HB process and send
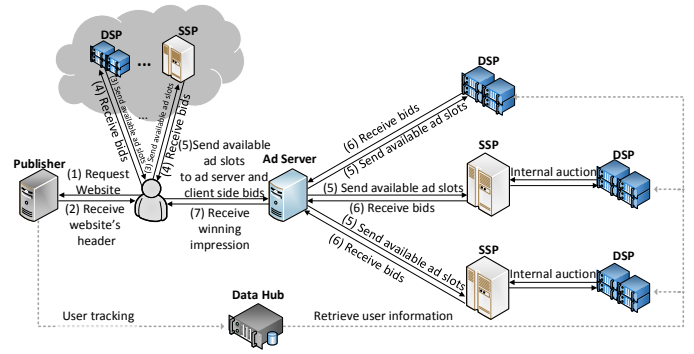

Figure 7: Hybrid HB overview and steps followed.

back to the client only the winning impressions. As Demand Partners, in this scenario, we consider all possible ad partners (SSPs, DSPs) that take part in the auction. Figure 6 shows the Server-Side HB model and the steps performed by the user's browser. The careful reader will note a similarity of this model with Client-Side HB with one Demand Partner. To distinguish Server-Side HB from Client-Side HB, we check the responses sent back from the Demand Partner involved to the browser, to filter out bid responses (which would reveal Client-Side HB cases). This filtering using HB-related keywords, also ensures that we are not mixing HB with traditional waterfall activity. Obviously, in this model the publisher needs to trust that the Demand Partner (i.e., the server handling all requests) is honest, will not execute waterfall in the backend instead of HB, and will select the best bids as winners, thus providing the best possible profits to the publisher.

Server-Side HB requires the least effort from the publishers to setup their HB. However, in exchange for setup convenience, it reduces transparency to the minimum, since the publishers have no way of knowing the Demand Partners participating in the auctions or their actual bids. Publishers don't need to tune their library, nor set up an ad server. They just add to their webpage a pre-configured library, provided by the Demand Partner they choose to collaborate with. Also, this setup could make small players less competitive, compared to big ones with better infrastructure and higher influence to the market, because publishers could tend to trust the latter ones. In effect, the Server-Side HB has re-enabled the dominant players in RTB to regain control of the ad-bidding process which was momentarily transferred on the user browser.

From the end-user's point of view, this setup lacks transparency and does not offer many insights on how the whole HB process either works, performs, or what impact it has on the user's browser: all auctions are done in the background, at the ad server's side. This setup brings back the pros and cons of the typical RTB with ADXs playing the crucial and controlling role in the protocol.

4.5 Hybrid HB

As its name states, this is a hybrid model that combines Client-Side HB with Server-Side HB (Figure 7). In this model, the user fetches the webpage which then requests bids from independent Demand Partners (as in the Client-Side HB model). When the browser (HB library) receives the bid responses, it sends them to the ad server along with the available slots. The ad server then performs its own
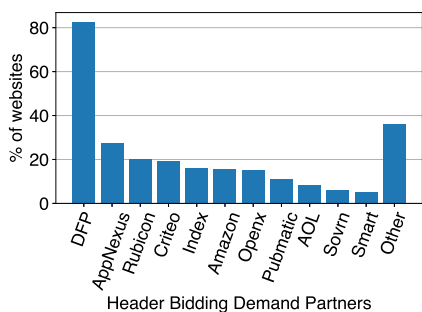
Figure 8: Top Demand Partners in HB. Google (with DFP) is present in 80% of websites with HB. The rest of Demand Partners (N=73) have presence in 36% of the websites with HB.
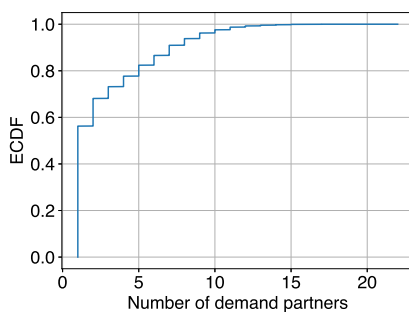
Figure 9: Demand Partners per website that employes HB. More than 50% of publishers use only one Demand Partner, but some use as many as 20.
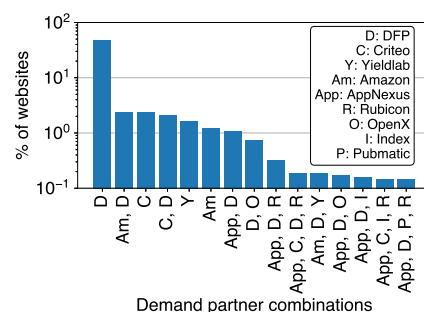
Figure 10: Most frequent Demand Partners combinations. DFP owns the majority of the market (48%). Criteo follows with 2.37% and Yieldlab with 1.68%.

auction (as in the Server-Side HB model) and picks the final winning impression(s) from all collected bids (both from client and server side). This model tries to combine the pros of Client-Side HB and Server-Side HB, while avoiding their cons. It is a semi-transparent model with a certain degree of fairness, which requires a moderate degree of effort for the setup. Publishers can choose the Demand Partners they will collaborate with directly, so they can know the bids they are willing to pay. Also they don't need to operate their own ad server, so the programmatic effort is reduced to tuning with the selected Demand Partners.

4.6 Facet Breakdown

The 3 facets of HB that we observed and described above, have the following breakdown as detected from the *HBDetector* in the wild (no other cases were observed that could comprise a 4th category). We find that the Server-Side HB currently comprises the larger portion of the market with 48%. Then, the Hybrid HB is second with 34.7%, and the Client-Side HB is third with 17.3%. This means that publishers prefer the centralization and control offered by a server-side (or hybrid) model, which imposes a smaller overhead and increases speed of transactions.

Indeed, the actors that provide both HB and waterfalling options need to respect the protocols' guidelines, otherwise they won't participate successfully in the HB process. Depending on the model they are called to use in each auction, they have to use the appropriate notification channel and parameters to notify the browser. As we will see in the next section, this highly skewed breakdown towards server-side or hybrid is due to the presence of Google's DFP, which participates in many of these HB auctions.

5 ANALYZING THE HB ECOSYSTEM

Here, we analyze the data crawled in different dimensions:

- Number, diversity and combinations of Demand Partners participating in HB (Section 5.1)
- Latencies measured with respect to overall HB process, publishers and participating partners (Section 5.2)
- Auctions performed, bids received, bids taken into account or got lost (Section 5.3)
- Properties of ads delivered: ad-slot prices paid and comparison with RTB prices. (Section 5.4)

5.1 Demand Partners Involved in HB

As a next step, we examine the properties of Demand Partners across the websites crawled and investigate who are the dominant Demand Partners, how many participate per website, and how they are combined together per webpage.

Who dominates the market?
First, we examine the popularity of each Demand Partner across all websites. We define as popularity the percentage of sites that a given Demand Partner participates in the site's HB process. In total, we find 84 unique Demand Partners. Figure 8 shows the 11 most popular Demand Partners. As we can see, Google's DoubleClick for Publishers (DFP) is the most popular partner, with more than 80% of publishers utilizing it. The DFP can be used both as an ad server and as a server-side HB solution. Thus, it is not strange that most of the publishers choose this option over setting their own ad server. We can also see that the list of top Demand Partners is full of popular partners that can be found also in the waterfall standard, as presented in past works[2] [35, 41]. These companies have already invested in the HB protocol and process early on, capitalizing on their knowledge and market share in RTB, and most publishers tend to choose these traditional big ad-partners over smaller ones.

How many Demand Partners are typically used?
A website can use more than one Demand Partner during the HB auction. But given that the more partners used could impact the loading time of the website, a question is what is typically employed by publishers. The number of unique Demand Partners participating in a HB auction are extracted from the incoming web requests that trigger corresponding HB events at the browser, and detected by the *HBDetector* (see Section 3.1 for details on the detection). Figure 9 shows the CDF of the number of Demand Partners found on each website. We can see that more than 50% of the websites use only one Demand Partner. However, about 20% of the publishers collaborate with 5 or more Demand Partners, and about 5% of publishers collaborate with ten or more Demand Partners.

Which Demand Partners are typically combined?
Demand Partners can appear on a website in different combinations. Given that we already identified 3 types of HB setup (client-side, server-side and hybrid), it is interesting to see how publishers select

---

[2] AppNexus, Index, Amazon, Rubicon, OpenX, AOL, Criteo, Pubmatic, and Sovrn, which match exactly what the industry claims as the top HB bidders in Aug'19 reports [28]
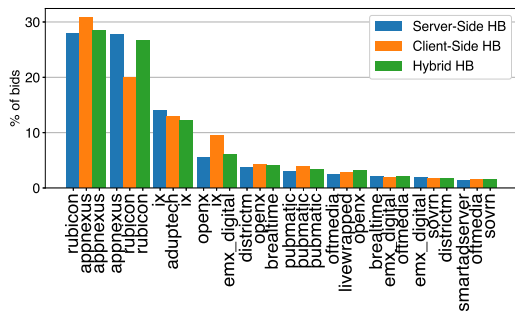
Figure 11: Top Demand Partners with respect to participation in HB auctions, per facet of HB. Top 2 partners in all types are Rubicon and AppNexus.

Figure 13: HB latency Vs. domain popularity with respect to Alexa ranking (in bins of 500 websites). Some outliers that goes as high as 10 seconds (removed for clarity).

different Demand Partners to participate in their HB auctions. We should keep in mind that the mixture of partners selected can impact the performance of HB with respect to delays and prices achieved. Also, frequently selected combinations may reveal typical or unlike competitions between Demand Partners. Therefore, we analyze the common combinations of Demand Partners found on webpages, and show the top 15 with respect to popularity in Figure 10, out of 753 possible groups of competitors found.

As expected, DFP holds a majority of the market on its own (i.e., appearing without any competitors) in 48% of the cases. Interestingly, besides DFP which dominates the market as single-partner, common groups of competitors include DFP in 51% of groups found. Furthermore, Criteo and Yieldlab follow as single partners with 2.37% and 1.68%, respectively. Some popular pairs of competitors include DFP and other companies such as Amazon, Criteo, and AppNexus. Finally, some triples include the above pairs with added entities such as Rubicon, OpenX, etc.

Which Demand Partners are used in each HB facet?
Given the three HB facets, we anticipate that some Demand Partners and publishers will prefer one facet of HB over another. Therefore, we analyze the participation of Demand Partners into each type, in Figure 11. In contrast to Client-Side HB, which all the bidders are transparent to the client, in Hybrid and Server-Side HB this is not the case. For this reason, we analyze the responses from the ad server (most commonly the DFP) to find the partners who won the
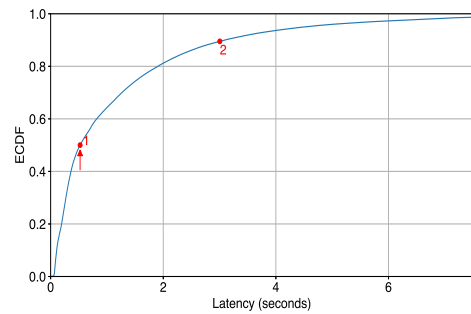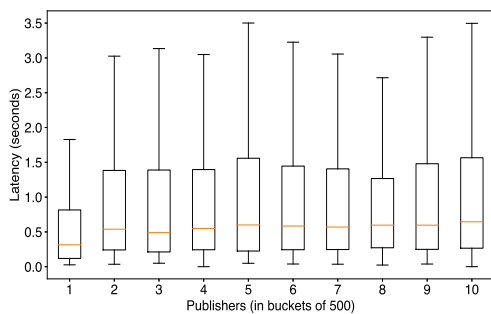

Figure 12: Total HB latency per website. (1) Marks median latency of 600ms. (2) Marks a commonly used industry threshold of 3 seconds which captures 90% of bid responses.

auctions. As expected, big DSPs like AppNexus and Rubicon hold the highest shares, followed by Index Exchange.

5.2 Header Bidding Latency

In this section, we explore various aspects of Header Bidding such as the imposed latency measured from different vantage points, with respect to overall latency, publishers using it, number of partners participating, etc. In all whiskers plots, we show 5th and 95th percentiles, and the boxes show 25th and 75th percentiles, with a red line for median (50th percentile).

How much latency does HB add?
The total latency of HB on a publisher's webpage is defined as the time from the first bid request to a Demand Partner (step 1 in Fig. 2) until the ad server is informed and responds (step 3 in Fig. 2). In Figure 12, we show the total time needed from the HB to process the bid requests and responses. We see that the median latency is about 600ms (point 1 in figure). However, some websites suffer a much higher overhead. Indeed, about 35% percent of the websites observe more than one second of latency, and as much as 4% of websites observe more than 5 seconds of latency for the HB process to conclude.

Based on our description so far, one might expect that a timeout would be used during HB, to cut off responses from slow Demand Partners. Although many of the wrappers use a timeout of 3 seconds, publishers are able to set their own threshold by making some changes in the wrappers. Unfortunately, our results indicate that at least 10% of the websites exceed the threshold of 3 seconds (point 2 in Figure 12), and some even need 20 seconds before the HB is completed (not shown in the figure for clarity of the other results).

Overall, even though most of the HB libraries strive to perform HB activities in an asynchronous fashion, it appears that HB can add significant overhead to a website if the library is badly tuned and Demand Partners are slow to respond. In a recent report [11], the average page load time (PLT) of a webpage was measured at 8.66 seconds, which is above the median latency measured here for HB. However, the industry recommends that the PLT should be kept under 3 seconds [11], which would lead 10% of websites with HB auctions experiencing time delays above this recommendation.

Does publisher popularity associate with HB latency?
As a next step, we study the latency measured with respect to the ranking of each website. Someone could expect that highly
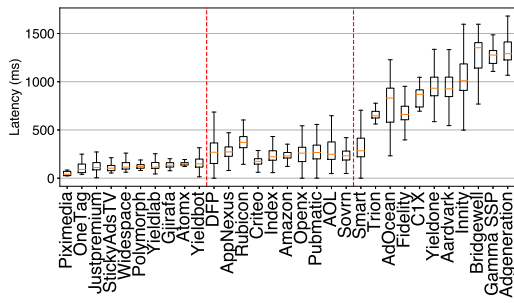
Figure 14: HB latency for the fastest partners (left-side of plot), top partners in market share (middle-section of plot), and slowest partners (right-side of plot). Top partners in market share are not the fastest.
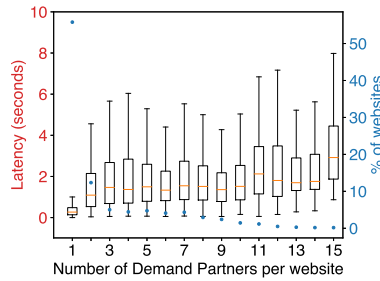
Figure 15: Total HB latency (left y-axis) and % of websites found (right y-axis) vs. number of Demand Partners per website. Publishers with more than one partner tend to have higher page load times.
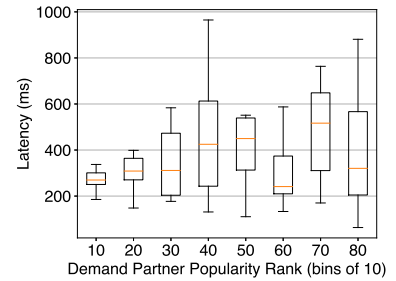
Figure 16: Distribution of latencies observed per Demand Partner across all the websites. Partners are ranked based on popularity. Popular partners tend to have latencies with smaller variability.

ranked publishers seek to have lower latencies for their websites, and therefore add partners in their HB process who demonstrate lower latencies. Also, higher-ranked websites may have available more resources to use in their HB planning, which could lead to reduced latencies and better performance. In Figure 13, we show the latency of publishers vs. their Alexa ranking. Indeed, we find that the highest-ranked publishers (i.e., the first 500 websites) exhibit significantly lower latencies (median = 310ms), than the rest of the ranked websites (median = 500ms).

Which are the fastest and slowest Demand Partners?

Figure 14 shows the fastest, top and slowest Demand Partners, respectively. We notice many small or unknown Demand Partners in these lists. The fastest (slowest) Demand Partners have median values in the range of 41-217ms (646-1290ms). Interestingly, the top Demand Partners with respect to market share have latencies that are small, but not low enough to qualify them for the fastest partners (with the exception of Criteo which has a median latency under 200ms).

Do multiple Demand Partners impact HB protocol's latency?
As we mentioned earlier, a publisher may choose to use several Demand Partners at the same time. Although this decision may increase competition for the ad-slots offered, and can drive-up the bidding prices, and consequently the publisher's revenue, it may also increase the latency of the webpage to load on the user's browser, and decrease the quality of the overall user experience. Therefore, we explore the impact that the number of Demand Partners can have on the user experience with respect to latency.

Figure 15 shows the latency of websites vs. the number of Demand Partners each website has. We observe that publishers who use only one Demand Partner have a small latency of 268.2 ms. As can be seen by the second y-axis, this is the majority of websites. Also, publishers with two Demand Partners have a latency of 1091.6 ms. Publishers with more than two Demand Partners have a median latency in the range of 1.3-3.0 seconds. **Does HB partner popularity associate with HB latency?**
Next, we study the latency of all 84 Demand Partners detected, ranked based on their popularity in our dataset. In Figure 16, we show the distribution of latencies observed per partner, when computed across all the websites each partner was found. We observe

that the most popular partners tend to have latencies with smaller variability (up to 200ms), in comparison to the less popular partners who may exhibit latency variability up to 500-1,000ms.
How many bids are late?
Here, we analyze the portion of bids that arrive too late to be included in the auction. As late bids we define all the responses about bids from Demand Partners which arrive too late, i.e., *after* the request to the ad server is sent from the browser. Thus, it is important to understand what is the portion (and number) of bids that were received from the browser, that came too late and were not considered in the HB auction. In Figure 17, we show the CDF of the portion of such late bids with respect to the total number of bids received at a website for the HB auction. We see that in 50% of the cases with late bids, almost 50% of the bid responses come too late to be considered in the auction by the ad server. Also, for 10% of the auctions, more than 80% of the bids are late. In results not show here due to space, we measured that in 60% of the auctions, there was only one late bid, in 40% of the auctions there was at least two late bids, and in 20% of auctions at least four late bids.

In Figure 18, we measure the percentage of late bids per Demand Partner. We notice that 21 Demand Partners bid too late in 50% of the auctions they participate. In some extreme cases, the Demand Partner loses 100% of the bids they send. All these late bids point to the possible loss of revenue from the publisher. This could be the result of a poorly defined wrapper that sends the request to the ad server the same time it sends the requests to Demand Partners, without waiting for their responses first, as well as Demand Partners that do not have the proper infrastructure to respond fast enough to all incoming requests.

5.3 Auctioned Ad-slots

In this section, we investigate the properties of the auctioned ad-slots, such as the size, the number of auctions per website, and how this impacts the overall performance of the protocol.
How many ad-slots are auctioned per webpage?
We start by investigating the number of ad slots that are available for auction. In Figure 19, we plot the CDF of the number of ad-slots across the websites, per HB type. In general, and for up to 70% of websites, the Hybrid HB type auctions more ad-slots than the other two types. For the other 30% of websites, Server-Side HB auctions
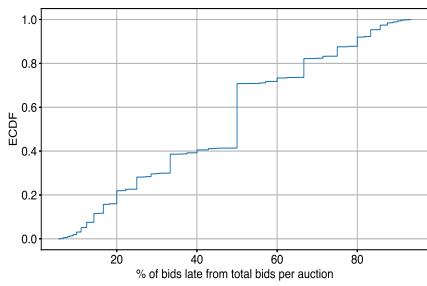
Figure 17: Portion of late bids over the total bids received, due to high latency of a partner to respond. For 10% of auctions, they have 80% or more late bids.
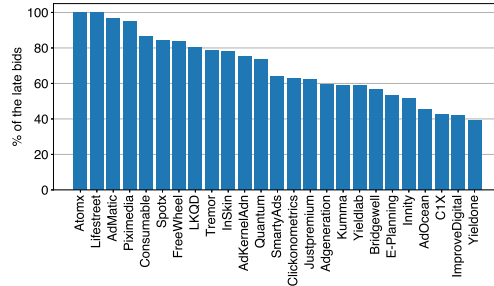

Figure 18: Percentage of late bids out of all bids sent per Demand Partner. Some partners have all their bids arriving too late to be considered for auction.
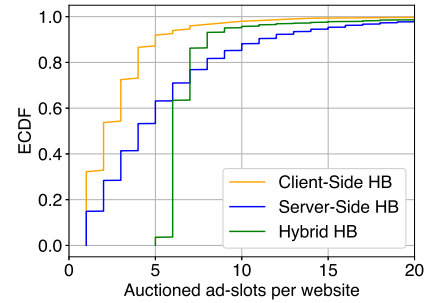

Figure 19: Auctioned ad-slots across websites, per HB facet. The median website has 2-6 available ad-slots. 90% of websites have 5-11 ad-slots (depending on the HB type).
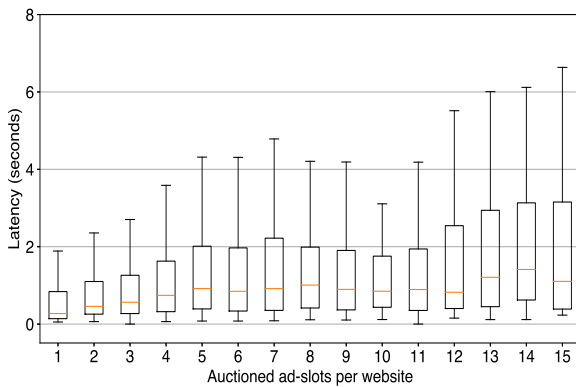

Figure 20: HB latency as a function of the number of ad-slots auctioned. More ad-slots result in higher median latency and variability in latency for the HB process.
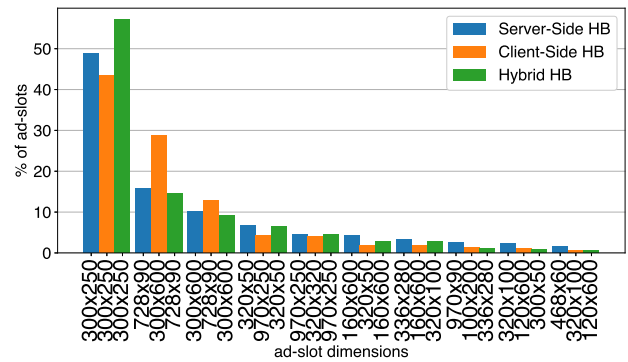

Figure 21: Portion of ads for different HB ad sizes, per HB facet. The side banner (300x250) and top banner (728x90) are among the most popular ad-slot sizes in all HB facets.

more ad-slots. The median website has 2-6 available ad-slots, and 90% of websites have up to 5-11 ad-slots (depending on the HB type). Also, 3% of the websites provide more than 20 slots for auction.

Requesting bids for 20 ad-slots on a single page can be considered odd, even a flag for suspicious or fraudulent behavior. Therefore, we manually investigated such cases, and to our surprise, we found that some publishers request auctions for *more* slots than they have available for display. After investigating this behavior further, we observed that these ad-slots refer to several different devices and screen sizes such as for tablet, smartphone, laptop, etc. We speculate they do that due to either bad configuration of their wrapper (i.e., they use the same HB wrapper for all the devices they serve without customizing the requests), or because they want to receive bids for multiple versions of the same ad-slots, for better optimization of the publisher's HB process later on. Indeed, this odd activity needs to be studied in depth in the future, to understand if it is a matter of bad practice or an effort for ad-fraud.

Does the number of auctioned ad-slots impact latency?
Next, we checked if the HB latency is associated with the number of ad-slots auctioned. Intuitively, we may expect that the more slots are to be auctioned, the more time the HB will take. However, given that a lot of Demand Partners invest significant computing

resources to parallelize and optimize bidding computations, the above statement may not hold. In Figure 20, we plot the latency of HB based on the number ad-slots auctioned in the website. In the majority of cases, this latency includes the communication to the ad server. In Client-Side HB, we cannot know the ad server (since each publisher uses their own), so we have no means to infer this latency. We observe that the total latency tends to increase with the number of slots auctioned. In fact, when there are 1-3 ad-slots auctioned, the median latency is 0.30-0.57 seconds, but when the slots are 3-5, the median latency ranges to 0.57-0.92 seconds. Interestingly, we observe that even if there is only one ad-slot to be auctioned, the latency can still vary per auction, from a few tens of milliseconds to almost two seconds. This variability can be due to extra latencies as result of internal auctions occurring at each Demand Partner.

What are the most popular ad-slots auctioned?
Finally, we analyze the most popular dimensions of HB ad slots. Our findings are presented in Figure 21, per facet of HB. The most common ad size is the 300x250 (side banner), for all 3 facets. The second most common is the 728x90 (top banner) and the 300x600 (for the Client-Side HB). These are generally popular banners in both mobile and desktop advertising, and they match results observed in the past for RTB [41]. Due to the increase of mobile
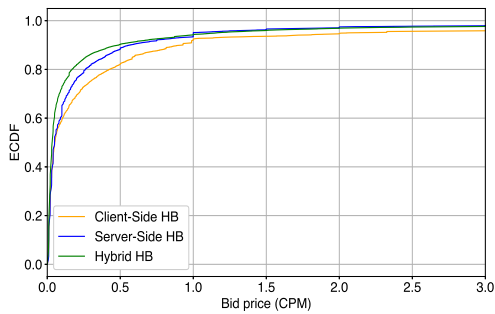
Figure 22: CDF of the auctioned ad slots bid prices, per HB facet. These are baseline prices that Demand Partners are willing to spend when they have no information for the user.
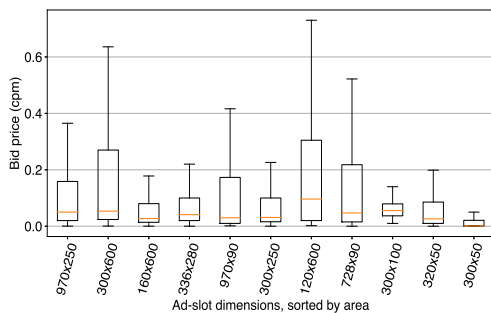

Figure 23: Distribution of bid prices in CPM per ad-slot size (x-axis sorted by area of ad-slot). Even in our crawler's baseline scenario, partners bid high prices to reach users.

browsing, publishers can choose these specific sizes to keep the HB configuration simple and well defined for multiple devices (as they don't need to set multiple sizes for different devices, and fewer auctions need to occur on the Demand Partners' side).

5.4 Ad-slot Bid Prices

In this section, we discuss the auctioned ad-slots' bid prices and how they vary depending on their size. We were able to detect the ad prices using *HBDetector*. In case of Hybrid and Client-Side HB, most of the prices are transparent to the client and easy to extract from the *bid response* messages. In contrast, in Server-Side HB the prices are not trivial to detect. We analyze in depth the auction metadata, and based on several heuristics we find and extract the prices whenever they are included.

What are the HB partners willing to pay?
First, we analyze the prices bided by the Demand Partners during the auctions. In Figure 22, we show the CDF of the baseline bid prices (in CPM or cost per thousand ad impressions, in USD) that advertisers are willing to spend for the ad-slots auctioned, per type of HB. In general, we note that Client-Side HB draws higher bid prices for the publisher, in comparison to the other two types. Also, more than 20% of the prices are more than 0.5 CPM, which is lower but comparable to regular waterfall prices, as claimed in past studies (found to be ~1 CPM [41]). Also, we should note that these prices are baseline, so they are much lower than if they were referring to targeted users.
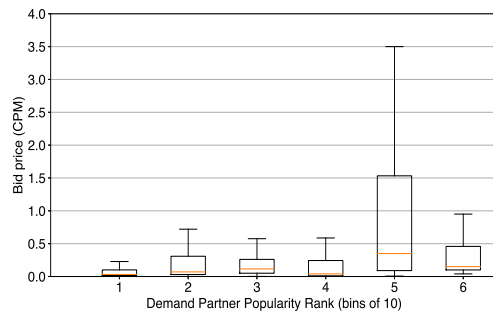

Figure 24: Distribution of prices that partners bid, ranked by popularity of Demand Partner (who are grouped in bins of 10).

What are the HB partners paying per ad-slot?
Second, we compare ad-slot sizes with bid prices for each size. In Figure 23 we plot the prices (in CPM) for each ad-slot. We see that in the recorded dimensions, the median cost ranges from 0.00084-0.096 CPM. The most expensive ad-slot (based on median price) is 120x600 with 0.096 CPM. The cheapest ad-slot is 300x50 (which also happens to have the least ad-area) with 0.00084 CPM. Also, the most popular ad-slot size, which is 300x250, has a median cost of 0.031 CPM. Previous studies on waterfall standard [41] find the prices of 300x250 slot ranging from 0.1 to 1.4 CPM, with a median of 0.19 CPM. These prices are higher than the ones found in our HB study, but we should again consider that our detected prices are for baseline users that Demand Partners have no prior knowledge, whereas in [41] it was for real users. Therefore, a follow-up work could apply real user profiles to collect HB prices, and thus, make a more fair comparison with RTB prices.

What is the variability of bid prices per DSP?
In Figure 24, we plot the prices (in CPM) that each Demand Partner bid to examine possible association between a partner's popularity and how high they bid in HB. The DSPs are ranked by popularity and grouped in buckets of 10 to ease illustration. The most popular partners (first bins) tend to be more consistent and bid lower prices. In contrast, less popular DSPs have higher median bid prices and variability in their bids. This observation could be explained when considering how the HB market works: for less popular DSPs to be competitive and win auctions, they bid higher prices than popular partners to reach sufficient number of users. Alternatively, this result could also indicate that more popular partners have technology that detects when browsing is of a baseline (or bot/unknown) user and therefore do not bid high, whereas the less popular partners bid high, hoping to target a real user. Finally, it can also be a side effect of how Demand Partners decide to spend their budget across the websites they collaborate with: more popular partners exist in more websites, and may chose to bid low in many of them, to cover a wider range of websites.

6 RELATED WORK

User data and their economics have long been an interesting topic and attracted a considerable body of research [2, 13, 21, 22, 35, 37, 41, 46, 47, 51, 53, 56]. In particular, Acquisti et al. discuss the value of privacy after defining two concepts (i) *Willingness To Pay*: the monetary amount users are willing to pay to protect their privacy,

and (ii) *Willingness To Accept*: the compensation that users are willing to accept for their privacy loss [2]. In two user-studies [13, 51] authors measure how much users value their own offline and online personal data, and consequently how much they would sell them to advertisers. In [47], authors propose "transactional" privacy to allow users to decide what personal information can be released and receive compensation from selling them.

Papadopoulos et al. set out to explore the cost advertisers pay to deliver an ad to the user in the waterfall standard and RTB auctions [41]. In addition, they study how the personal data that users leak while browsing (like location and interests) can affect the pricing dynamics. The authors propose a methodology to compute the total cost paid for the user even when advertisers hide the charged prices. Finally, they evaluate their methodology by using data from a large number of volunteering users. Olejnik et al. perform an analysis of cookie matching in association with the RTB advertising [35]. They leverage the RTB notification URL to observe the charge prices and they conduct a basic study to provide some insights into these prices, by analyzing different user profiles and visiting contexts. Their results confirm that when the users' browsing histories are leaked, the charge prices tend to be increased. In [39], the authors measure the costs of digital advertising on both the user's and the advertiser's side in an attempt to compare how fairly these costs are distributed between the two. In particular, they compare the cost advertisers pay in the waterfall standard with the costs imposed on the data plan, the battery efficiency and (by using cookie synchronization [1, 38, 40] as a metric) the privacy of the specific user.

In [31], the authors briefly describe HB and focus on optimizing its bidding strategy and the produced yield. They consider revenue optimization as a contextual bandit problem, where the context consists of the information available about the ad opportunity, such as properties of the internet user or of the provided ad slot. In [21], authors use a dataset of users' HTTP traces and provide rough estimates of the relative value of users by leveraging the suggested bid amounts for the visited websites, based on categories provided by the Google AdWords. FDTV [22] is a plugin to inform users in real-time about the economic value of the personal information associated to their Facebook activity. In [30], Iordanou et al. try to detect both programmatic and static advertisements in a webpage, using (i) a crowdsourcing, and (ii) a crawling approach to determine the criteria with which ads are displayed. They find biases on ads depending on age, income and gender of users.

Bashir et al. study the diffusion of user tracking caused by RTB-based programmatic ad-auctions [6]. Results of their study show that under specific assumptions, no less than 52 tracking companies can observe at least 91% of an average user's browsing history. In an attempt to shed light upon Facebook's ad ecosystem, Andreou et al. investigate the level of transparency provided by the mechanisms "Why am I seeing this?" and Ad Preferences Page [4]. The authors built a browser extension to collect Facebook ads and information extracted from these two mechanisms before performing their own ad campaigns and target users that used their browser extension. They show that ad explanations are often incomplete and misleading. In [5], the authors aim to enhance the transparency in ad ecosystem with regards to information sharing, by developing a content agnostic methodology to detect client- and server- side

flows of information between ad exchanges and leveraging retargeted ads. By using crawled data, the authors collected 35.4K ad impressions and identified 4 different kinds of information sharing behavior between ad exchanges.

7 SUMMARY & DISCUSSION

Header Bidding is gaining popularity among Web publishers, who want to regain the control of their ad inventory and what advertisers are paying for it. Proponents of HB have touted that this new ad-tech protocol increases transparency and fairness among advertisers, since more partners can directly compete for an ad-slot. HB, in theory, can boost the revenue of publishers, who can select the Demand Partners that are competing for the publishers' ad-slots, and also remove intermediaries from the ad-selling process.

In this study, we investigate and present in full detail the different implementations of HB and how each of them works. Based on these observations, we design and implement *HBDetector*: a first of its kind tool to measure in a systematic fashion the evolving ecosystem of HB, its performance and its properties. By running *HBDetector* across a list of top 35,000 Alexa websites, we collected data about 800k HB auctions and performed the first in-depth analysis of HB. We discuss our lessons from this study in the next paragraphs.

7.1 Commoditization of Ad Supply

Header Bidding was introduced to put Demand Partners under pressure for more competitive pricing (and loosen Google's grip on the market). Indeed, it has changed the hierarchy on the supply side. Depending on the publisher's needs, we found that Header Bidding can be implemented in 3 ways: (i) Client-Side HB, (ii) Server-Side HB, and (iii) Hybrid HB. Therefore, Demand Partners that could previously claim exclusive access to a publisher's inventory (and thus higher positions in the waterfall) are no longer able to do so. Instead, HB enabled all Demand Partners regardless of their size or relationship with publishers, to compete for the same inventory, thus commoditizing supply [16].

However, as measured in this study, big companies such as DoubleClick, AppNexus, Rubicon, Criteo, etc., took advantage of their existing dominance in the ad-market and placed themselves again in a very centralizing (and process controlling) position within the HB ecosystem (especially within the Server-Side HB and Hybrid HB models). In fact, we identified that Server-Side HB dominates this market with 48% of auctions handled by a single partner/ad server. Google, in particular, handles as much as 80% of HB auctions. DoubleClick for Publishers (DFP) dominates as a single partner, while it also appears in 51% of the competing groups of Demand Partners in HB. Also, most publishers use only one Demand Partner, but some use many (more than 10). Interestingly, this centralization is in direct contrast to the publishers' revenue. We found that websites utilizing Client-Side HB achieve higher bid prices than the other two models.

7.2 Non-Viable Performance Overheads

The fear of latency has kept some premium publishers away from header integrations and continues to make others wary about embracing HB. Results of this study verify the concerns of publishers [14, 18] regarding the latencies imposed on the user side. We measured up to 0.6 seconds for the median website and more than 3 seconds in 10% of websites. Furthermore, publishers with more than one Demand Partner experienced higher HB latencies: one Demand Partner imposes a small latency of 0.3 seconds, but 2 Demand Partners impose 1.1 seconds latency, and 3 Demand Partners can impose up to 3 seconds latency. It is of no doubt that for the publishers that do the utmost to provide readers with a high-quality experience, such latency is capable of significantly degrading the user experience. Interestingly, we find that the top 500 (in Alexa ranking) websites exhibited significantly lower HB latency than the rest of websites.

Although Header Bidding tech promises multiple, in-parallel bid requests to Demand Partners that can provide the best possible ad price to the publisher, Javascript on the users' end is single-threaded. This means that even if the HB provider's wrapper performs well-optimized asynchronous ad calls, these still need to stand in the network queue, thus increasing not only the overall HB execution time but also the entire webpage's loading time. These delays can have adverse effects on user's browsing experience while loading a HB-enabled webpage. Interestingly, we find that the 10 most popular Demand Partners exhibit lower variability in latency than the rest, demonstrating that they invest appropriate resources to reduce latencies at the user-end.

7.3 Late Bids: Revenue & Network Cost

The broadcasting nature of Header Bidding results in an enormous amount of bid requests to multiple Demand Partners. As measured in this study, a typical website has a small number of available ad-slots (i.e., 2-6 ad-slots for the median case, depending on the type of HB), but some auctions request for more ad-slots than they have available to show (even up to 20). Side banner and top banner are the most popular ad-slots auctioned in HB. For each ad-slot, a parallel auction takes place that requests bids from numerous DSPs. As expected, the more ad-slots present in a webpage, the higher the overall HB latency: when 1-3 ad-slots are auctioned, the median latency is 0.3-0.57 seconds, but for 3-5 slots, the median latency is 0.57-0.92 seconds.

This overwhelming volume of bid requests significantly increases the needed processing power for ADXs and the decision engines of DSPs, thus skyrocketing their infrastructure costs [17]. Indeed, companies that started supporting HB experienced increases of up to 100% in the bid requests they received [50] (i.e., between 5 million and 6 million requests per second) for the very same number of available ad-slots as before. Interestingly, the same partners may in fact compete for the same ad-slots more than once: in the HB, and then in the regular waterfall model, since the publisher may still fall back to the waterfall if the HB does not reach high enough prices for the auctioned slots [17].

Apart from skyrocketing the infrastructure costs, the increased amount of bid requests also increases the response time for DSPs, causing lots of delayed bids. We found that in more than 50% of auctions, half of bid responses arrive too late (after the publisher's set threshold) to be considered, due to high latency. These late bids not only are wasted network resources and processing power from the point of view of the Demand Partners, but also loss of potentially higher revenues for publishers.

7.4 Limitations & Future Work

The present work is a first, comprehensive study of the HB protocol, and an effort to measure the ecosystem and partners involved. Unfortunately, HB documentation is scarce at best, and it has been a feat to reverse-engineer the protocol, and understand the numerous HB libraries used by the crawled websites. Due to several limitations in the data collection process, in the present study, we focused on specific dimensions and left others for future work. In a follow-up work on HB, it is important to address the limitations of our *HBDetector*, and also perform extensions to study aspects not covered in this work:

- Perform extensive analysis of all available HB libraries, to increase coverage of the ecosystem and websites employing HB.
- Study in detail the impact that HB has on the UX and page load time of each website, as well as the various hosting infrastructures responsible for the crawled websites, locations of Demand Partners involved, and categories of websites to find associations with HB prices and latencies.
- Investigate the privacy of online users accessing HB-enabled websites for potential PII leaks, and measure the impact that HB may have on user anonymity, as well as the use of HTTP vs. HTTPs for HB transactions.

REFERENCES

[1] Gunes Acar, Christian Eubank, Steven Englehardt, Marc Juarez, Arvind Narayanan, and Claudia Diaz. The web never forgets: Persistent tracking mechanisms in the wild. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, 2014.
[2] Alessandro Acquisti, Leslie K John, and George Loewenstein. What is privacy worth? *The Journal of Legal Studies*, 2013.
[3] Alexa. Alexa, the top sites on the web. https://www.alexa.com/topsites.
[4] Athanasios Andreou, Giridhari Venkatadri, Oana Goga, Krishna Gummadi, Patrick Loiseau, and Alan Mislove. Investigating ad transparency mechanisms in social media: A case study of facebook's explanations. In *Proceedings of the Network and Distributed System Security Symposium (NDSS)*, 2018.
[5] Muhammad Ahmad Bashir, Sajjad Arshad, William Robertson, and Christo Wilson. Tracing information flows between ad exchanges using retargeted ads. In *25th USENIX Security Symposium*, 2016.
[6] Muhammad Ahmad Bashir and Christo Wilson. Diffusion of user tracking data in the online advertising ecosystem. *Proceedings on Privacy Enhancing Technologies*, 2018.
[7] Ross Benes. An ad tech urban legend: An oral history of how header bidding became digital advertising's hottest buzzword. https://digiday.com/media/header-bidding-oral-history/, 2017.

[8] Ross Benes. How latency emerged as publishers' worst user-experience headache. https://digiday.com/media/latency-emerged-publishers-worst-user-experience-headache/, 2017.

[9] Ross Benes. The state of header bidding in 4 charts. https://digiday.com/media/state-header-bidding-4-charts/, 2017.

[10] Ricardo Bilton. With header bidding, publishers are boosting cpms by as much as 50 percent. https://digiday.com/media/header-bidding-publishers-boosting-cpms-much-50-percent/, 2015.

[11] Machmetrics Speed Blog. Site speed tips for performance nerds. https://www.machmetrics.com/speed-blog/average-page-load-times-websites-2018/, 2018.

[12] The Ad-Juster Blog. Key differences between direct-sold and programmatic advertising. https://blog.ad-juster.com/differences-between-direct-sold-and-programmatic-advertising/, 2017.

[13] Juan Pablo Carrascal, Christopher Riederer, Vijay Erramilli, Mauro Cherubini, and Rodrigo de Oliveira. Your browsing behavior for a big mac: Economics of personal information online. In *Proceedings of the Conference on World Wide Web (WWW)*. ACM, 2013.

[14] Chris_13. #opspov: How header bidders affect latency. https://www.admonsters.com/opspov-how-header-bidders-affect-latency/, 2016.

[15] Jessica Davies. Beware of page latency: The side effects to header bidding. https://digiday.com/uk/beware-page-latency-side-effects-header-bidding/, 2016.

[16] Jessica Davies. Behind the bid-caching debate, signs of ad tech commoditization. https://digiday.com/media/behind-bid-caching-debate-signs-ad-tech-commoditization/, 2018.

[17] Jessica Davies. The winners and losers of header bidding. https://digiday.com/media/header-bidding-winners-losers/, 2018.

[18] Avin Dunaway. The truth about latency and the header: A chat with pubmatic. https://www.admonsters.com/truth-latency-header-pubmatic/, 2018.

[19] eMarketer Inc. Five charts: The state of header bidding. https://www.emarketer.com/content/five-charts-the-state-of-header-bidding, 2019.

[20] Lauren Fisher. Header bidding update 2018. https://www.emarketer.com/content/header-bidding-update-2018, 2018.

[21] Phillipa Gill, Vijay Erramilli, Augustin Chaintreau, Balachander Krishnamurthy, Konstantina Papagiannaki, and Pablo Rodriguez. Follow the money: Understanding economics of online aggregation and advertising. In *Proceedings of the Internet Measurement Conference (IMC)*. ACM, 2013.

[22] José González Cabañas, Angel Cuevas, and Rubén Cuevas. Fdvt: Data valuation tool for facebook users. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*. ACM, 2017.

[23] Goodway Group. 2018 header bidding trends: The update you need to read. https://www.mediapost.com/publications/article/332220/in-app-ads-header-bidding-enjoy-solid-growth.html, 2018.

[24] Google. Google publisher tag. https://developers.google.com/doubleclick-gpt/reference.

[25] Kean Graham. What is a bid request? https://www.monetizemore.com/blog/what-is-a-bid-request/, 2018.

[26] Goodway Group. 2018 header bidding trends: The update you need to read. https://goodwaygroup.com/blog/2018-header-bidding-trends-update, 2018.

[27] The Interactive Advertising Bureau (IAB). Openrtb (real-time bidding). https://www.iab.com/guidelines/real-time-bidding-rtb-project/, 2019.

[28] Adzerk Inc. Header bidding industry index. https://adzerk.com/hbix/, 2019.

[29] Internet Archive. Internet archive: The wayback machine. https://archive.org/web/, 2019.

[30] C. Iordanou, N. Kourtellis, J.M. Carrascosa, C. Soriente, R. Cuevas Rumin, and N. Laoutaris. Beyond content analysis: Detecting targeted ads via distributed counting. In *Proceedings of the International Conference on emerging Networking EXperiments and Technologies (CONEXT)*. ACM, 2019.

[31] Grégoire Jauvion, Nicolas Grislain, Pascal Dkengne Sielenou, Aurélien Garivier, and Sébastien Gerchinovitz. Optimization of a ssp's header bidding strategy using thompson sampling. *CoRR*, abs/1807.03299, 2018.

[32] OKO Ad Managment. Ad waterfalls explained. https://oko.uk/blog/ad-waterfalls-explained, 2019.

[33] Mozilla. Mdn web docs. https://developer.mozilla.org/en-US/docs/Web/API/EventTarget/addEventListener.

[34] Lukasz Olejnik and Claude Castelluccia. To bid or not to bid? measuring the value of privacy in rtb. Technical Report, 2015.

[35] Lukasz Olejnik, Minh-Dung Tran, and Claude Castelluccia. Selling off user privacy at auction. In *21st Annual Network and Distributed System Security Symposium, NDSS*, 2014.

[36] Gavin O'Malley. In-app ads, header bidding enjoy solid growth. https://www.mediapost.com/publications/article/332220/in-app-ads-header-bidding-enjoy-solid-growth.html, 2019.

[37] Michalis Pachilakis, Panagiotis Papadopoulos, Nikolaos Laoutaris, Evangelos P Markatos, and Nicolas Kourtellis. Measuring ad value without bankrupting user privacy. *arXiv preprint arXiv:1907.10331*, 2019.

[38] Panagiotis Papadopoulos, Nicolas Kourtellis, and Evangelos Markatos. Cookie synchronization: Everything you always wanted to know but were afraid to ask. In *Proceedings of the World Wide Web Conference (WWW)*. ACM, 2019.

[39] Panagiotis Papadopoulos, Nicolas Kourtellis, and Evangelos P Markatos. The cost of digital advertisement: Comparing user and advertiser views. In *Proceedings of the World Wide Web Conference (WWW)*. ACM, 2018.

[40] Panagiotis Papadopoulos, Nicolas Kourtellis, and Evangelos P. Markatos. Exclusive: How the (synced) cookie monster breached my encrypted vpn session. In *Proceedings of the 11th European Workshop on Systems Security*, EuroSec'18, pages 6:1–6:6, New York, NY, USA, 2018. ACM.

[41] Panagiotis Papadopoulos, Nicolas Kourtellis, Pablo Rodriguez Rodriguez, and Nikolaos Laoutaris. If you are not paying for it, you are the product: How much do advertisers pay to reach you? In *Proceedings of the Internet Measurement Conference (IMC)*. ACM, 2017.

[42] Wing Poon. Now that in-app header bidding is finally here, is the waterfall era truly over? https://blog.appodeal.com/waterfall-parallel-bidding-part-one/, 2018.

[43] Prebid. Publisher api reference. http://prebid.org/dev-docs/publisher-api-reference.html.

[44] Prebid. Bidder params. http://prebid.org/dev-docs/bidders.html, 2019.

[45] Prebid.org. Prebid - header bidding unwrapped. http://prebid.org, 2017.

[46] Abbas Razaghpanah, Rishab Nithyanand, Narseo Vallina-Rodriguez, Srikanth Sundaresan, Mark Allman, and Christian Kreibich Phillipa Gill. Apps, trackers, privacy, and regulators. In *Proceedings of the Network and Distributed System Security Symposium*, NDSS'18, 2018.

[47] Christopher Riederer, Vijay Erramilli, Augustin Chaintreau, Balachander Krishnamurthy, and Pablo Rodriguez. For sale : Your data: By : You. In *Proceedings of the 10th ACM Workshop on Hot Topics in Networks*, 2011.

[48] Quirin Scheitle, Oliver Hohlfeld, Julien Gamba, Jonas Jelten, Torsten Zimmermann, Stephen D. Strowes, and Narseo Vallina-Rodriguez. A long way to the top:, significance, structure, and stability of internet top lists. In *Proceedings of the Internet Measurement Conference (IMC)*. ACM, 2018.

[49] Sarah Sluis. The year header bidding went mainstream. https://adexchanger.com/publishers/year-header-bidding-went-mainstream/, 2016.

[50] Sarah Sluis. Header bidding unleashed a huge infrastructure problem and ad tech will either sink or swim. https://adexchanger.com/platforms/header-bidding-unleashed-huge-infrastructure-problem-ad-tech-will-either-sink-swim/, 2017.

[51] Jacopo Staiano, Nuria Oliver, Bruno Lepri, Rodrigo de Oliveira, Michele Caraviello, and Nicu Sebe. Money walks: A human-centric study on the economics of personal mobile data. In *Proceedings of the ACM International Joint Conference on Pervasive and Ubiquitous Computing*, 2014.

[52] Statista. Growth rates of monetized header bidding advertising impressions volume worldwide from 1st quarter 2017 to 3rd quarter 2018, by device. https://www.statista.com/statistics/812228/header-bidding-ad-impression-change/, 2019.

[53] Narseo Vallina-Rodriguez, Srikanth Sundaresan, Abbas Razaghpanah, Rishab Nithyanand, Mark Allman, Christian Kreibich, and Phillipa Gill. Tracking the trackers: Towards understanding the mobile advertising and tracking ecosystem. *arXiv preprint arXiv:1609.07190*, 2016.

[54] Ratko Vidakovic. Header bidding: What marketers need to know. https://marketingland.com/header-bidding-marketers-need-know-199311, 2017.

[55] Ted Vrountas. Simplifying the header bidding process & what marketers should know. https://instapage.com/blog/what-is-header-bidding, 2018.

[56] Apostolis Zarras, Alexandros Kapravelos, Gianluca Stringhini, Thorsten Holz, Christopher Kruegel, and Giovanni Vigna. The dark alleys of madison avenue: Understanding malicious advertisements. In *Proceedings of the Internet Measurement Conference (IMC)*. ACM, 2014.