the mediagrid

ENGINEERED BY IPONWEB

Header Bidding Demystified: Client-Side vs. Server-Side

For most web inventory, header bidding has become an essential component of most publishers' ad monetisation strategies, with over 80 percent of the top 1000 Alexa publishers running header bidding. It enables higher inventory fill rates and revenue by allowing publishers to receive bids from multiple trading partners at the same time, in contrast to the traditional 'waterfall' method of trading, in which inventory is passed to ad networks/platforms sequentially.

Before header bidding came along, the waterfall was the default technical setup for most publishers' ads monetization. But managing the waterfall was not just inefficient and resource intensive, but it also left money on the table. Header bidding unlocked better ways to get more yield from the same inventory pool, but it, too, is not a one-size-fits-all solution.

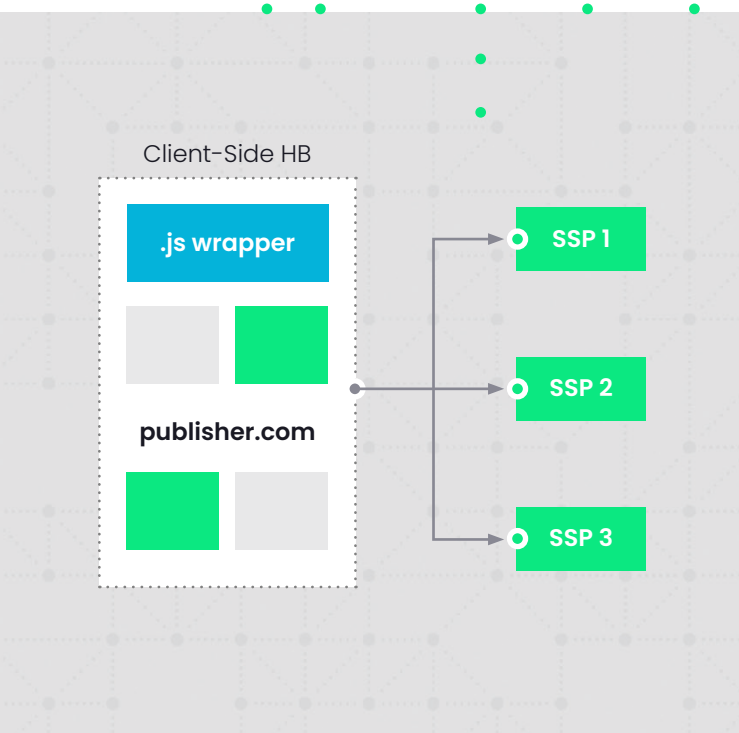
Header bidding simplifies the programmatic bidding process for publishers by allowing them to ask multiple demand partners simultaneously for bid values before sending the request to their ad server. Because every bidder gets a chance to compete in the auction regardless of where they sit in a publisher's stack, header bidding helps to: increase fill rates, give bidders a more even footing as well as increase transparency into how much impressions are worth. It also means that programmatic bids can compete more fairly with other line items in the ad server, including direct sold campaigns.

With 91% of desktop and 86% of mobile web inventory sold via header bidding in 2020, there's no question that it's become the industry standard since it gained widespread popularity. For most publishers today, it is not a choice of should we use header bidding, but of how best to configure and manage our header setup to drive maximum yield - through a client-side or server integration. Both can be used to initiate header calls, however there are benefits and drawbacks for each. We will explore both techniques further below.

What is client-side header bidding?

Perhaps the best way to think of client-side header bidding is as browser-based header bidding.

Client-side header bidding remains the most commonly used version of the technology, and it operates similarly to any tag you might find in a website's header. When a user loads a page, the header bidding JavaScript code in the header fires, which is typically within a header container or wrapper that sends a bid request to multiple demand partners simultaneously, who then submit a bid for the impression (see diagram below). Unlike the waterfall auction method, all of these requests are sent in parallel, democratizing the auction by design so that every bidder gets a fair shot at the impression.



Each header auction is given a set amount of time to take place, allowing the demand side partner to receive the request, assess the opportunity and respond with an appropriate bid amount. If demand sources do not respond within this window, the publisher will close the auction and select a winner from the responses it did receive. The frequency with which a demand-side partner fails to respond across auctions is called a **timeout rate**, and it's something publishers need to carefully balance. That's because any bidder which fails to respond is essentially a lost revenue opportunity, along with increasing the load-time burden on the browser. The fewer the bids, the less competition, the lower the potential return – and smart publishers are starting to remove demand partners with consistently high timeout rates as a result.

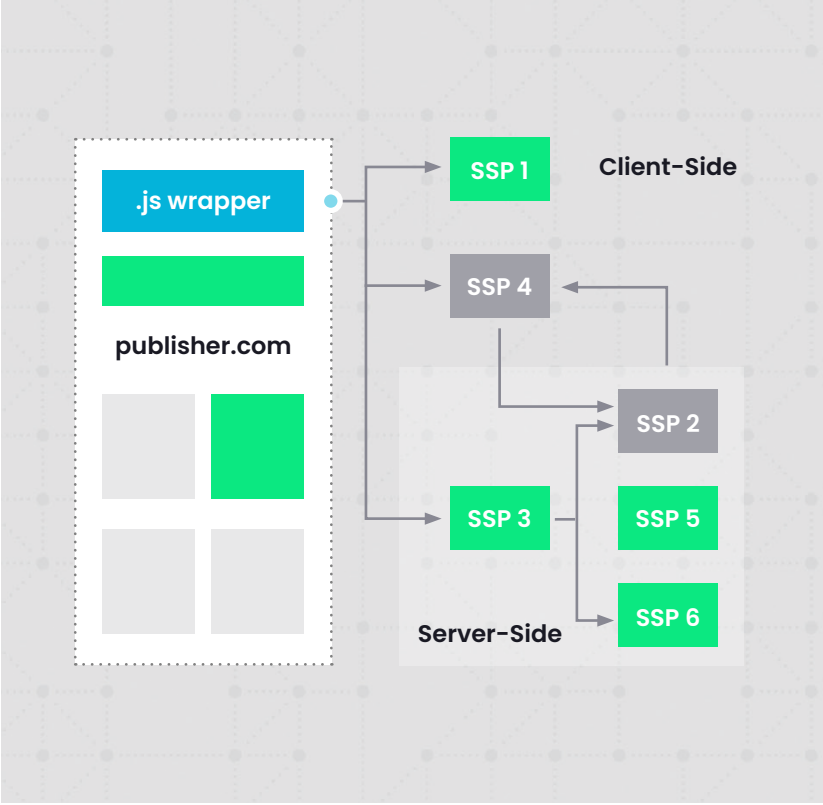
Of course, with multiple header calls going out to different partners in parallel from the user's browser, client-side header bidding impacts page loading speed, which can affect the publisher's user experience in some cases.

All of these issues around latency can be solved by reducing the number of bidders in the publisher's header. Nonetheless, if latency is a concern, there is another way.

What is server-side header bidding?

Quite simply, the main difference between client-side and server-side header bidding is that bid requests are sent out via a centralized server rather than a user's browser.

When a user loads a page, everything happens the same way as client-side, except that, rather than sending out multiple parallel bid requests to the chosen demand partners in a publisher's stack, the JavaScript tag instead pings the publisher's Server-to-Server (S2S) platform. This could be a specific vendor offering a paid header bidding S2S service via a platform like Prebid Server, a preferred SSP partner, or even a self-hosted implementation if you have the server horsepower.



This server then sends out bid requests to multiple demand partners – usually many more than is possible with a client-side implementation – and returns the highest bid to the publisher page. From here, all other bids are collected and returned to the ad server, where bids are attributed to the proper line items participating in the final auction.

The biggest reason that publishers might opt for an S2S header bidding implementation is the big reduction in latency. Server-to-server effectively offloads the burden of sending parallel bid requests and managing responses from the browser to the server, meaning it's not limited by browser technologies or page load speeds. The publisher's website simply sends one request to the S2S provider and the auction is taken care of in the cloud.

While server-side header bidding solves issues with client-side implementations such as latency, it's not without problems of its own. Server-to-server integrations require that you have a transparent partner; as a preferred partner collates the bids from all other demand partners it is necessary that you have a trustworthy and transparent vendor to provide fair auctions. In addition, as that vendor is the sole gate to downstream partners, there are also limitations around achieving accurate cookie match rates and identity resolution which we describe below.

Cookie Syncing and Match Rates

Without a direct relationship with the end user, SSPs and DSPs rely on cookie matching to 'sync' the users that are common to all trading partners. For client-side header bidding, the publisher controls the cookie syncs with its trading partners and is able to maintain a high match rate with its connected SSP partners.

The cookie sync determines a match rate, i.e. the percentage of shared known users, with this usually averaging 50-80% between each downstream participant; with higher match rates, publishers can command higher advertising revenues. As cookie syncs are performed 'downstream' in the media trading chain (publishers to SSPs, SSPs to DSPs, DSPs to brands), the reduction in match rate between downstream trading partners (that are not directly connected to the publisher) is compounded at each step and can result in a loss of revenue for the publisher.

For example, the typical match rate that a publisher has with a connected SSP is ~70%. If that SSP has a 60% match rate with a DSP, the overall publisher to DSP match rate is only 42% (i.e. 60% of the SSP-to-publisher 70% match rate).

When the SSPs move from a client-side to a server-side setup, the SSP has fewer controls over the browser cookie setup and the user sync calls, which now belong to the server, have a detrimental effect on the user match rate – which can impact both CPMs and overall performance/revenue.

Client-side vs. server-side: Which is right for your business?

Now that you're familiar with how each implementation of header bidding mechanics works, let's talk about which is right for you. Of course, there's never a single solution that will fit all businesses, so instead, we'll consider the pros and cons of each technology to help you decide.

The benefits and drawbacks of client-side header bidding include:

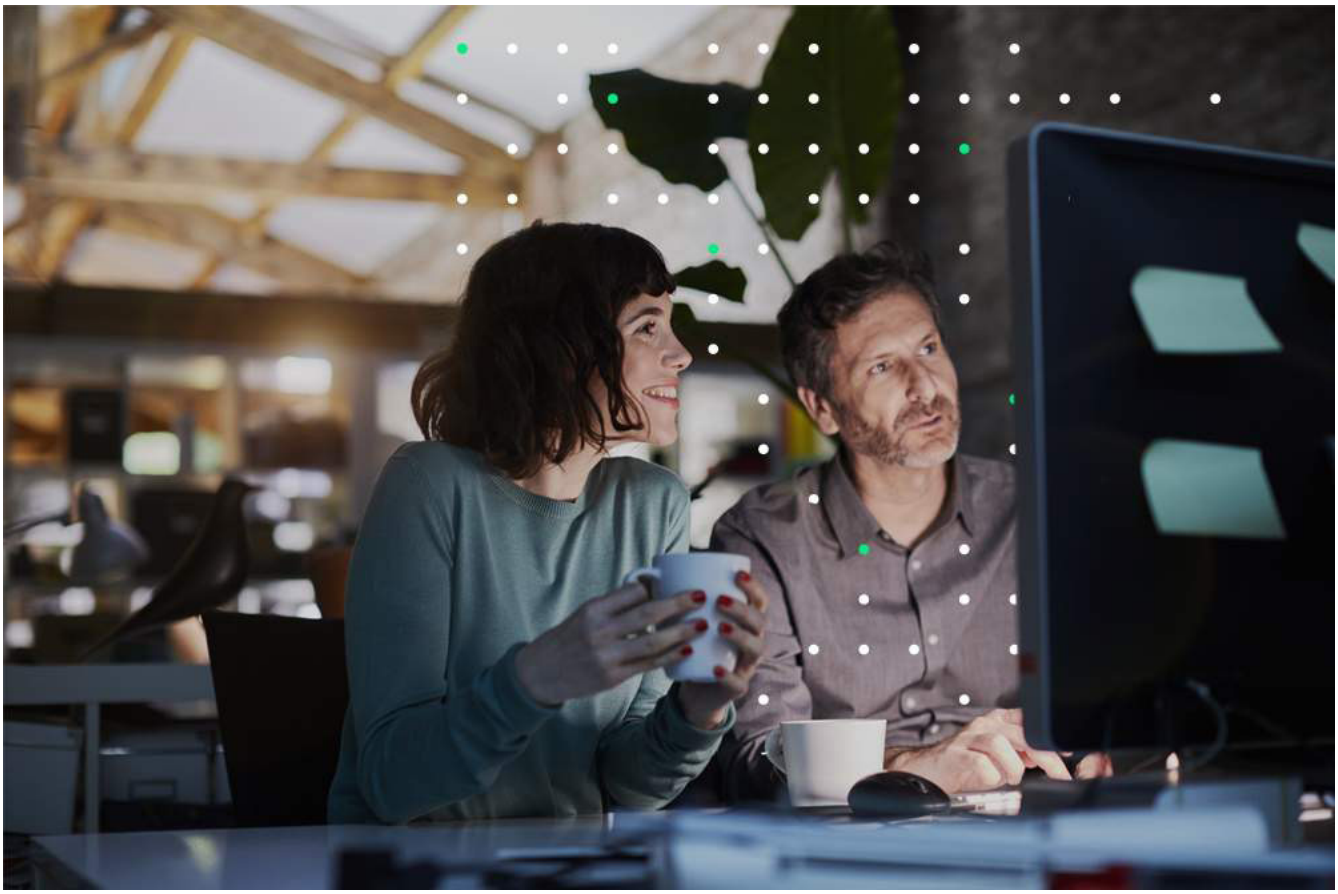
- ✔ Better identity resolution thanks to the ability to use browser-based cookie syncing.
- ✔ Transparency and control over demand partners, which can be added or removed at will by the publisher.

- ✕ Increased latency and slower page loads due to the need to send out parallel bid requests and await a response (though this can be mitigated by reducing bidder numbers).
- ✕ Limitations on the total number of parallel requests browsers can make.

The benefits and drawbacks of server-side header bidding include:

- ✔ Eliminates latency by removing the need to host a client-side auction and send out multiple bid requests in parallel.
- ✔ Removes limitations on demand partners, boosting the pool of potential bids.

- ✕ Reduction in buy-side transparency. As the server-side partner is now hosting the auction, publishers may not have full visibility into the auction participants and processes
- ✕ The addition of a third-party in the chain results in lower cookie syncing match rates, which can decrease CPMs and impact overall revenue.



As with many aspects of programmatic advertising, the best solution for publishers here is to try things out and see what works best for them. For example, a publisher could have bidders X, Y, and Z integrated via a client-side integration while having bidders A, B, and C via a server-side setup. Or they may want to use specific bidders for specific formats – it's totally flexible.

The bottom line here is that publishers need to dedicate the time to assess which bidders you should have server vs. client, and how that might differ for different format types, regions, etc. At The MediaGrid, we've found that through a combination of A/B testing, changes to timeout rates, adjustments to line items priorities, and other yield optimization techniques, publishers can easily identify what works best for their unique setup.

To learn more about the best header bidding setup for your business, or for help testing out which technologies will deliver the best yield, get in touch with The MediaGrid team today.

Connect with The MediaGrid.

Email us directly at info@themediagrid.com for more information, or visit us at www.themediagrid.com.

the mediagrid
ENGINEERED BY IPONWEB