Lag

From Wikipedia, the free encyclopedia

[Jump to navigation](https://en.wikipedia.org/wiki/Lag#mw-head)[Jump to search](https://en.wikipedia.org/wiki/Lag#searchInput)

*This article is about delay in online games. For other uses, see*[*Lag (disambiguation)*](https://en.wikipedia.org/wiki/Lag_(disambiguation))*.*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | hide**This article has multiple issues.** Please help [**improve it**](https://en.wikipedia.org/w/index.php?title=Lag&action=edit) or discuss these issues on the [**talk page**](https://en.wikipedia.org/wiki/Talk:Lag). *(*[*Learn how and when to remove these template messages*](https://en.wikipedia.org/wiki/Help:Maintenance_template_removal)*)*   |  | | --- | | This article **needs additional citations for**[**verification**](https://en.wikipedia.org/wiki/Wikipedia:Verifiability). *(April 2011)* |  |  | | --- | | This article **possibly contains**[**original research**](https://en.wikipedia.org/wiki/Wikipedia:No_original_research). *(December 2008)* |  |  | | --- | | This article **contains**[**instructions, advice, or how-to content**](https://en.wikipedia.org/wiki/Wikipedia:What_Wikipedia_is_not#NOTHOWTO). *(April 2016)* | |

In [online gaming](https://en.wikipedia.org/wiki/Online_game), **lag** is a noticeable delay ([latency](https://en.wikipedia.org/wiki/Latency_(engineering))) between the action of [players](https://en.wikipedia.org/wiki/Gamer) (input) and the reaction of the [server](https://en.wikipedia.org/wiki/Server_(computing)) supporting the [game](https://en.wikipedia.org/wiki/Video_game), which has to be sent back to the [client](https://en.wikipedia.org/wiki/Client_(computing)).

The player's ability to tolerate lag depends on the type of game being played. For instance, a [strategy game](https://en.wikipedia.org/wiki/Strategy_game) or a [turn-based game](https://en.wikipedia.org/wiki/Turn-based_game) with a slow pace may have a high threshold or even be mostly unaffected by high lag. A game with [twitch gameplay](https://en.wikipedia.org/wiki/Twitch_gameplay) such as a [first-person shooter](https://en.wikipedia.org/wiki/First-person_shooter) with a considerably faster pace may require a significantly lower lag to provide satisfying [gameplay](https://en.wikipedia.org/wiki/Gameplay).



**Contents**

* [1Ping time](https://en.wikipedia.org/wiki/Lag#Ping_time)
* [2Causes](https://en.wikipedia.org/wiki/Lag#Causes)
* [3Effects](https://en.wikipedia.org/wiki/Lag#Effects)
* [4Solutions and lag compensation](https://en.wikipedia.org/wiki/Lag#Solutions_and_lag_compensation)
  + [4.1Client-side](https://en.wikipedia.org/wiki/Lag#Client-side)
  + [4.2Server-side](https://en.wikipedia.org/wiki/Lag#Server-side)
    - [4.2.1Rewind time](https://en.wikipedia.org/wiki/Lag#Rewind_time)
    - [4.2.2Trust clients](https://en.wikipedia.org/wiki/Lag#Trust_clients)
    - [4.2.3Make clients extrapolate](https://en.wikipedia.org/wiki/Lag#Make_clients_extrapolate)
  + [4.3Design](https://en.wikipedia.org/wiki/Lag#Design)
* [5Cloud gaming](https://en.wikipedia.org/wiki/Lag#Cloud_gaming)
* [6See also](https://en.wikipedia.org/wiki/Lag#See_also)
* [7References](https://en.wikipedia.org/wiki/Lag#References)

Ping time[[edit](https://en.wikipedia.org/w/index.php?title=Lag&action=edit&section=1)]

Ping time is the network [latency](https://en.wikipedia.org/wiki/Latency_(engineering)) between a player's [client](https://en.wikipedia.org/wiki/Client_(computing)) and the game [server](https://en.wikipedia.org/wiki/Server_(computing)) as measured with the [ping utility](https://en.wikipedia.org/wiki/Ping_utility) or equivalent. Ping time is an average time in [milliseconds](https://en.wikipedia.org/wiki/Millisecond) (ms). The lower one's ping is, the lower the latency is and the less lag the player will experience. *High ping* and *low ping* are commonly used terms in online gaming, where *high ping* refers to a ping that causes a severe amount of lag; while any level of ping may cause lag, severe lag is usually indicated by a ping of over 100 ms.[[1]](https://en.wikipedia.org/wiki/Lag#cite_note-1) This usage is a gaming cultural colloquialism and is not commonly found or used in professional computer networking circles. In games where timing is key, such as [first-person shooter](https://en.wikipedia.org/wiki/First-person_shooter) and [real-time strategy](https://en.wikipedia.org/wiki/Real-time_strategy) games, a low ping is always desirable, as a low ping means smoother [gameplay](https://en.wikipedia.org/wiki/Gameplay) by allowing faster updates of game data between the players' clients and game server.

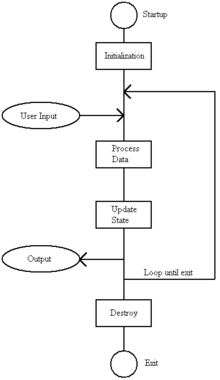
High latency can cause lag. Game servers may disconnect a client if the latency is too high and may pose a detriment to other players' gameplay. Similarly, client [software](https://en.wikipedia.org/wiki/Software) will often mandate disconnection if the latency is too high. High ping may also cause servers to crash due to instability.

In some first-person shooter games, a high ping may cause the player to unintentionally gain unfair advantages, such as disappearing from one location and instantaneously reappearing in another, simulating the effect of teleportation, thus making it hard for other players to judge their character's position and subsequently making the player much more difficult to target. To counter this, many game servers automatically kick players with a ping higher than average. Conversely, a high ping can make it very difficult for the player to play the game due to negative effects occurring, making it difficult for the player to track other players and even move their character.

Rather than using the traditional [ICMP](https://en.wikipedia.org/wiki/Internet_Control_Message_Protocol) echo request and reply [network packets](https://en.wikipedia.org/wiki/Network_packet) to determine ping times, [video game programmers](https://en.wikipedia.org/wiki/Video_game_programmer) often build their own latency detection into existing game packets (usually based on the [UDP](https://en.wikipedia.org/wiki/User_Datagram_Protocol) protocol) instead.

Some factors that might affect ping include: [communication protocol](https://en.wikipedia.org/wiki/Communication_protocol) used, Internet [throughput](https://en.wikipedia.org/wiki/Throughput) (connection speed), the quality of a user's Internet service provider and the configuration of [firewalls](https://en.wikipedia.org/wiki/Firewall_(computing)). Ping is also affected by geographical location. For instance, if someone is in India, playing on a server located in the United States, the distance between the two is greater than it would be for players located within the US, and therefore it takes longer for data to be transmitted. However, the amount of packet-switching and network hardware in between the two computers is often more significant. For instance, [wireless network interface cards](https://en.wikipedia.org/wiki/Wireless_network_interface_cards) must modulate digital signals into [radio signals](https://en.wikipedia.org/wiki/Radio_wave), which is often more costly than the time it takes an electrical signal to traverse a typical span of cable. As such, lower ping can result in faster Internet download and upload rates.

Causes[[edit](https://en.wikipedia.org/w/index.php?title=Lag&action=edit&section=2)]

[](https://en.wikipedia.org/wiki/File:Gameloop.PNG)

A simplified game architecture

While a [single-player game](https://en.wikipedia.org/wiki/Single-player_game) maintains the main game state on the local machine, an [online game](https://en.wikipedia.org/wiki/Online_game) requires it to be maintained on a central server in order to avoid inconsistencies between individual clients. As such, the client has no direct control over the central game state and may only send change requests to the server, and can only update the local game state by receiving updates from the server. This need to communicate causes a delay between the clients and the server, and is the fundamental cause behind lag. While there may be numerous underlying reasons for why a player experiences lag, they can be summarized as insufficient hardware in either the client or the server, or a poor connection between the client and server.[[2]](https://en.wikipedia.org/wiki/Lag#cite_note-2)

Hardware related issues cause lag due to the fundamental structure of the game architecture. Generally, games consist of a looped sequence of states, or "frames". During each frame, the game accepts user input and performs necessary calculations (AI, graphics etc.). When all processing is finished, the game will update the game state and produce an output, such as a new image on the screen and/or a packet to be sent to the server. The frequency at which frames are generated is often referred to as the [frame rate](https://en.wikipedia.org/wiki/Frame_rate). As the central game state is located on the server, the updated information must be sent from the client to the server in order to take effect. In addition, the client must receive the necessary information from the server in order to fully update the state. Generating packets to send to the server and processing the received packets can only be done as often as the client is able to update its local state. Although packets could theoretically be generated and sent faster than this, it would only result in sending redundant data if the game state cannot be updated between each packet. A low frame rate would, therefore, make the game less responsive to updates and may force it to skip outdated data.

Conversely, the same holds true for the server. The frame rate (or tick rate) of the server determines how often it can process data from clients and send updates. This type of problem is difficult to predict and compensate for. Apart from enforcing minimum hardware requirements and attempting to optimize the game for better performance, there are no feasible ways to deal with it.

Perhaps the most common type of lag is caused by [network performance](https://en.wikipedia.org/wiki/Network_performance) problems. [Losses](https://en.wikipedia.org/wiki/Packet_loss), [corruption](https://en.wikipedia.org/wiki/Data_corruption) or [jitter](https://en.wikipedia.org/wiki/Packet_delay_variation) (an outdated packet is in effect a loss) may all cause problems, but these problems are relatively rare in a network with sufficient [bandwidth](https://en.wikipedia.org/wiki/Bandwidth_(computing)) and no or little [congestion](https://en.wikipedia.org/wiki/Network_congestion). Instead, the latency involved in transmitting data between clients and server plays a significant role. Latency varies depending on a number of factors, such as the physical distance between the end-systems, as a longer distance means additional transmission length and [routing](https://en.wikipedia.org/wiki/Routing) required and therefore higher latency. Routing over the [Internet](https://en.wikipedia.org/wiki/Internet) may be extremely indirect, resulting in far more transmission length (and consequent latency) than a direct route, although the cloud gaming service [OnLive](https://en.wikipedia.org/wiki/OnLive) has developed a solution to this issue by establishing [peering](https://en.wikipedia.org/wiki/Peering) relationships with multiple [Tier 1 network](https://en.wikipedia.org/wiki/Tier_1_network) Internet Service Providers and choosing an optimal route between server and user.[[3]](https://en.wikipedia.org/wiki/Lag#cite_note-Columbia-video-3) In addition, insufficient bandwidth and congestion, even if not severe enough to cause losses, may cause additional delays regardless of distance. As with the hardware issues, packets that arrive slowly or not at all will make both the client and server unable to update the game state in a timely manner.

Online game systems utilizing a [wireless network](https://en.wikipedia.org/wiki/Wireless_network) may be subject to significant lag, depending on the architecture of the wireless network and local [electromagnetic interference](https://en.wikipedia.org/wiki/Electromagnetic_interference) impacting that network. Electromagnetic interference (e.g. from a [microwave oven](https://en.wikipedia.org/wiki/Microwave_oven)) can cause transmitted packets to be lost, requiring a retransmission which incurs latency. Although [radio propagation](https://en.wikipedia.org/wiki/Radio_propagation) through the air is faster than light through an optical fiber, wireless systems are often shared among many users and may suffer from latency incurred due to [network congestion](https://en.wikipedia.org/wiki/Network_congestion), or due to [network protocols](https://en.wikipedia.org/wiki/Network_protocols) that introduce latency.

Effects[[edit](https://en.wikipedia.org/w/index.php?title=Lag&action=edit&section=3)]

The noticeable effects of lag vary not only depending on the exact cause, but also on any and all techniques for lag compensation that the game may implement (described below). As all clients experience some delay, implementing these methods to minimize the effect on players is important for smooth gameplay. Lag causes numerous problems for issues such as accurate rendering of the game state and hit detection.[[4]](https://en.wikipedia.org/wiki/Lag#cite_note-4) In many games, lag is often frowned upon because it disrupts normal gameplay. The severity of lag depends on the type of game and its inherent tolerance for lag. Some games with a slower pace can tolerate significant delays without any need to compensate at all, whereas others with a faster pace are considerably more sensitive and require extensive use of compensation to be playable (such as the first-person shooter genre). Due to the various problems lag can cause, players that have an insufficiently fast Internet connection are sometimes not permitted, or discouraged from playing with other players or servers that have a distant server host or have high latency to one another. Extreme cases of lag may result in extensive desynchronization of the game state.

Lag due to an insufficient update rate between client and server can cause some problems, but these are generally limited to the client itself. Other players may notice jerky movement and similar problems with the player associated with the affected client, but the real problem lies with the client itself. If the client cannot update the game state at a quick enough pace, the player may be shown outdated renditions of the game, which in turn cause various problems with hit- and collision detection.[[5]](https://en.wikipedia.org/wiki/Lag#cite_note-5) If the low update rate is caused by a low frame rate (as opposed to a setting on the client, as some games allow), these problems are usually overshadowed by numerous problems related to the client-side processing itself. Both the display and controls will be sluggish and unresponsive. While this may increase the perceived lag, it is important to note that it is of a different kind than network-related delays. In comparison, the same problem on the server may cause significant problems for all clients involved. If the server is unable or unwilling to accept packets from clients fast enough and process these in a timely manner, client actions may never be registered. When the server then sends out updates to the clients, they may experience freezing (unresponsive game) and/or [rollbacks](https://en.wikipedia.org/wiki/Rollback_(data_management)), depending on what types of lag compensation, if any, the game uses.

Lag due to network delay is, in contrast, often less of a problem. Though more common, the actual effects are generally smaller, and it is possible to compensate for these types of delays. Without any form of lag compensation, the clients will notice that the game responds only a short time after an action is performed. This is especially problematic in first-person shooters, where enemies are likely to move as a player attempts to shoot them and the margin for errors is often small.

Solutions and lag compensation[[edit](https://en.wikipedia.org/w/index.php?title=Lag&action=edit&section=4)]

There are various methods for reducing or disguising delays, though many of these have their drawbacks and may not be applicable in all cases. If synchronization is not possible by the game itself, the clients may be able to choose to play on servers in geographical proximity to themselves in order to reduce latencies, or the servers may simply opt to drop clients with high latencies in order to avoid having to deal with the resulting problems. However, these are hardly optimal solutions. Instead, games will often be designed with lag compensation in mind.[[6]](https://en.wikipedia.org/wiki/Lag#cite_note-6)

Many problems can be solved simply by allowing the clients to keep track of their own state and send absolute states to the server or directly to other clients.[[7]](https://en.wikipedia.org/wiki/Lag#cite_note-bernier01-7) For example, the client can state exactly at what position a player's character is or who the character shot. This solution works and will all but eliminate most problems related to lag. Unfortunately, it also relies on the assumption that the client is honest. There is nothing that prevents a player from modifying the data they send, directly at the client or indirectly via a proxy, in order to ensure they will always hit their targets. In online games, the risk of cheating may make this solution unfeasible, and clients will be limited to sending relative states (i.e. which vector it moved on or shot in).

**Client-side**[[edit](https://en.wikipedia.org/w/index.php?title=Lag&action=edit&section=5)]

As clients are normally not allowed to define the main game state, but rather receive it from the server, the main task of the client-side compensation is to render the virtual world as accurately as possible. As updates come with a delay and may even be dropped, it is sometimes necessary for the client to [predict](https://en.wikipedia.org/wiki/Client-side_prediction) the flow of the game. Since the state is updated in discrete steps, the client must be able to estimate a movement based on available samples. Two basic methods can be used to accomplish this; [extrapolation](https://en.wikipedia.org/wiki/Extrapolation) and [interpolation](https://en.wikipedia.org/wiki/Interpolation).[[7]](https://en.wikipedia.org/wiki/Lag#cite_note-bernier01-7)

Extrapolation is an attempt to estimate a future game state. As soon as a packet from the server is received, the position of an object is updated to the new position. Awaiting the next update, the next position is extrapolated based on the current position and the movement at the time of the update. Essentially, the client will assume that a moving object will continue in the same direction. When a new packet is received, the position may be corrected slightly.

Interpolation works by essentially buffering a game state and rendering the game state to the player with a slight, constant delay. When a packet from the server arrives, instead of updating the position of an object immediately, the client will start to interpolate the position, starting from the last known position. Over an interpolation interval, the object will be rendered moving smoothly between the two positions. Ideally, this interval should exactly match the delay between packets, but due to loss and variable delay, this is rarely the case.

Both methods have advantages and drawbacks.

* Interpolation ensures that objects will move between valid positions only and will produce good results with constant delay and no loss. Should dropped or out-of-order packets overflow the interpolation buffer the client will have to either freeze the object in position until a new packet arrives, or fall back on extrapolation instead. The downside of interpolation is that it causes the world to be rendered with additional latency, increasing the need for some form of lag compensation to be implemented.
* The problem with extrapolating positions is fairly obvious: it is impossible to accurately predict the future. It will render movement correctly only if the movement is constant, but this will not always be the case. Players may change both speed and direction at random. This may result in a small amount of "warping" as new updates arrive and the estimated positions are corrected, and also cause problems for hit detection as players may be rendered in positions they are not actually in.

Often, in order to allow smooth gameplay, the client is allowed to do soft changes to the game state. While the server may ultimately keep track of ammunition, health, position, etc., the client may be allowed to predict the new server-side game state based on the player's actions, such as allowing a player to start moving before the server has responded to the command. These changes will generally be accepted under normal conditions and make delay mostly transparent. Problems will arise only in the case of high delays or losses, when the client's predictions are very noticeably undone by the server. Sometimes, in the case of minor differences, the server may even allow "incorrect" changes to the state based on updates from the client.

**Server-side**[[edit](https://en.wikipedia.org/w/index.php?title=Lag&action=edit&section=6)]

Unlike clients, the server knows the exact current game state, and as such prediction is unnecessary. The main purpose of server-side lag compensation is instead to provide accurate effects of client actions. This is important because by the time a player's command has arrived time will have moved on, and the world will no longer be in the state that the player saw when issuing their command. A very explicit example of this is hit detection for weapons fired in first-person shooters, where margins are small and can potentially cause significant problems if not properly handled.

**Rewind time**[[edit](https://en.wikipedia.org/w/index.php?title=Lag&action=edit&section=7)]

Another way to address the issue is to store past game states for a certain length of time, then rewind player locations when processing a command.[[7]](https://en.wikipedia.org/wiki/Lag#cite_note-bernier01-7) The server uses the latency of the player (including any inherent delay due to interpolation; see above) to rewind time by an appropriate amount in order to determine what the shooting client saw at the time the shot was fired. This will usually result in the server seeing the client firing at the target's old position and thus hitting. In the worst case, a player will be so far behind that the server runs out of historical data and they have to start leading their targets.

This is a [WYSIWYG](https://en.wikipedia.org/wiki/WYSIWYG) solution that allows players to aim directly at what they are seeing. But the price is an aggravation of the effects of latency when a player is under fire: not only does their own latency play a part, but their attacker's too. In many situations, this is not noticeable, but players who have just taken cover will notice that they carry on receiving damage/death messages from the server for longer than their own latency can justify. This can lead more often to the (false) impression that they were shot through cover and the (not entirely inaccurate) impression of "laggy [hitboxes](https://en.wikipedia.org/wiki/Hitbox)".[[7]](https://en.wikipedia.org/wiki/Lag#cite_note-bernier01-7)

One design issue that arises from rewinding is whether to stop rewinding a dead player's lagged commands as soon as they die on the server, or to continue running them until they "catch up" to the time of death. Cutting compensation off immediately prevents victims from posthumously attacking their killers, which meets expectations, but preserves the natural advantage of moving players who round a corner, acquire a target and kill them in less time than a round trip to the stationary victim's client.

Rewinding can be criticised for allowing the high latency of one player to negatively affect the experience of low-latency players. Servers with lag compensation will sometimes reduce the length of player history stored, or enforce ping limits, to reduce this problem.

**Trust clients**[[edit](https://en.wikipedia.org/w/index.php?title=Lag&action=edit&section=8)]

It is possible for clients to tell the server what they are doing and for the server to trust the data it receives. This method is avoided if at all possible due to its susceptibility to [cheating](https://en.wikipedia.org/wiki/Cheating_in_video_games): it is a simple matter to route network data through a second computer which inserts fabricated hit messages or modifies existing ones, a technique which cannot be detected by [anti-cheat](https://en.wikipedia.org/wiki/Anti-cheat) tools.[[7]](https://en.wikipedia.org/wiki/Lag#cite_note-bernier01-7)

However, the sheer scale of some games makes computationally expensive solutions like [rewinding](https://en.wikipedia.org/wiki/Lag#Rewind_time) impossible. In [*Battlefield 3*](https://en.wikipedia.org/wiki/Battlefield_3), for example, a "hybrid hit detection" system is used where clients tell the server that they hit and the server performs only a vague test of plausibility before accepting the claim.[[8]](https://en.wikipedia.org/wiki/Lag#cite_note-bf3-clientside-8)

Trusting a client's results otherwise has the same advantages and disadvantages as [rewinding](https://en.wikipedia.org/wiki/Lag#Rewind_time).

**Make clients extrapolate**[[edit](https://en.wikipedia.org/w/index.php?title=Lag&action=edit&section=9)]

A less common lag solution is to do nothing on the server and to have each client extrapolate (see above) to cover its latency.[[9]](https://en.wikipedia.org/wiki/Lag#cite_note-ro2-lag-9) This produces incorrect results unless remote players maintain a constant velocity, granting an advantage to those who dodge back and forth or simply start/stop moving.

Extended extrapolation also results in remote players becoming visible (though not vulnerable) when they should not be: for example if a remote player sprints up to a corner then stops abruptly at the edge, other clients will render them sprinting onward, into the open, for the duration of their own latency. On the other side of this problem, clients have to give remote players who just started moving an extra burst of speed in order to push them into a theoretically-accurate predicted location.

**Design**[[edit](https://en.wikipedia.org/w/index.php?title=Lag&action=edit&section=10)]

It is possible to reduce the perception of lag through [game design](https://en.wikipedia.org/wiki/Game_design). Techniques include playing client-side animations as if the action took place immediately, reducing/removing built-in timers on the host machine, and using camera transitions to hide warping.[[10]](https://en.wikipedia.org/wiki/Lag#cite_note-aldridge11-10)

Cloud gaming[[edit](https://en.wikipedia.org/w/index.php?title=Lag&action=edit&section=11)]

[Cloud gaming](https://en.wikipedia.org/wiki/Cloud_gaming) is a type of online gaming where the entire game is hosted on a game server in a data center, and the user is only running a [thin client](https://en.wikipedia.org/wiki/Thin_client) locally that forwards [game controller](https://en.wikipedia.org/wiki/Game_controller) actions upstream to the game server. The game server then renders the next frame of the game video which is compressed using low-lag [video compression](https://en.wikipedia.org/wiki/Video_compression) and is sent downstream and decompressed by the thin client. For the cloud gaming experience to be acceptable, the round-trip lag of all elements of the cloud gaming system (the thin client, the Internet and/or LAN connection the game server, the game execution on the game server, the video and audio compression and decompression, and the display of the video on a [display device](https://en.wikipedia.org/wiki/Display_device)) must be low enough that the user perception is that the game is running locally.[[3]](https://en.wikipedia.org/wiki/Lag#cite_note-Columbia-video-3)[[11]](https://en.wikipedia.org/wiki/Lag#cite_note-D8_Video:_OnLive_demoed_on_iPad,_PC,_Mac,_Console,_iPhone-11) Because of such tight lag requirements, distance considerations of the [speed of light](https://en.wikipedia.org/wiki/Speed_of_light) through [optical fiber](https://en.wikipedia.org/wiki/Optical_fiber) come into play, currently limiting the distance between a user and a cloud gaming game server to approximately 1000 miles, according to [OnLive](https://en.wikipedia.org/wiki/OnLive), the only company thus far operating a cloud gaming service.[[12]](https://en.wikipedia.org/wiki/Lag#cite_note-12) There is also much controversy about the lag associated with cloud gaming. In multiplayer games using a client/server network architecture, the player's computer renders the game's graphics locally and only information about the player's in-game actions are sent to the server. For example, when the player presses a button, the character on-screen instantly performs the corresponding action. However, the consequences of the action such as an enemy being killed are only seen after a short delay due to the time taken for the action to reach the server. This is only acceptable as long as the response to the player's input is fast enough.

When using cloud gaming, inputs by the player can lead to short delays until a response can be seen by them. Inputs must first be transmitted to the remote server, then the server must start rendering the graphics of the action being performed and stream the video back to the player over the network, taking additional time. Thus, the player experiences a noticeable delay between pressing a button and seeing something happen on-screen. Depending on the skill and experience of the player, this can cause disorientation and confusion similar to [Delayed Auditory Feedback](https://en.wikipedia.org/wiki/Delayed_Auditory_Feedback) and hampers navigation and aiming in the game world. When rapidly inputting a long combination move, the on-screen character will not be synchronized with the button presses. This usually causes severe confusion in the player resulting in the failure of the combination move.

The extra input lag can also make it very difficult to play certain single player games. For example, if an enemy takes a swing at the player and the player is expected to block, then by the time the player's screen shows that the enemy has commenced attacking, the enemy would have already struck and killed the player on the server.

See also[[edit](https://en.wikipedia.org/w/index.php?title=Lag&action=edit&section=12)]

* [Lagometer](https://en.wikipedia.org/wiki/Lagometer)

References[[edit](https://en.wikipedia.org/w/index.php?title=Lag&action=edit&section=13)]

* 1. [**^**](https://en.wikipedia.org/wiki/Lag#cite_ref-1) [*"How to Get Rid of Lag | GeForce"*](https://www.geforce.com/whats-new/guides/how-to-get-rid-of-lag-guide)*. www.geforce.com. Retrieved 2018-09-13.*
  2. [**^**](https://en.wikipedia.org/wiki/Lag#cite_ref-2) *Cronin, Eric; Filstrup, Burton; Anthony, Kurc.*[*"A Distributed Multiplayer Game Server System"*](http://www.eecg.toronto.edu/~ashvin/courses/ece1746/2003/reading/cronin-umtr01.pdf)*(PDF). University of Michigan. Retrieved 16 July 2014.*
  3. ^ [Jump up to:***a***](https://en.wikipedia.org/wiki/Lag#cite_ref-Columbia-video_3-0) [***b***](https://en.wikipedia.org/wiki/Lag#cite_ref-Columbia-video_3-1) [*"The Process of Invention: OnLive Video Game Service"*](http://tv.seas.columbia.edu/videos/545/60/79)*. The FU Foundation School of Engineering & Applied Science (Columbia University). Retrieved 2010-01-23.*
  4. [**^**](https://en.wikipedia.org/wiki/Lag#cite_ref-4) *Smith, Joshua.*[*"Distributed Game Architecture To Overcome System Latency"*](https://docs.google.com/viewer?url=patentimages.storage.googleapis.com/pdfs/US5899810.pdf)*(PDF). United States Patent. Retrieved 16 July2014.*
  5. [**^**](https://en.wikipedia.org/wiki/Lag#cite_ref-5) *Claypool, Mark; Claypool, Kajal.*[*"Latency Can Kill: Precision and Deadline in Online Games"*](https://archive.is/20140629140345/http:/boostelo.net/latency)*. Archived from*[*the original*](http://boostelo.net/latency)*on 29 June 2014. Retrieved 16 July 2014.*
  6. [**^**](https://en.wikipedia.org/wiki/Lag#cite_ref-6) *Roelofs, Gregory.*[*"Compensating For Network Latency In A Multi-Player Game"*](https://docs.google.com/viewer?url=patentimages.storage.googleapis.com/pdfs/US6475090.pdf)*(PDF). United States Patent. Retrieved 16 July2014.*
  7. ^ [Jump up to:***a***](https://en.wikipedia.org/wiki/Lag#cite_ref-bernier01_7-0) [***b***](https://en.wikipedia.org/wiki/Lag#cite_ref-bernier01_7-1) [***c***](https://en.wikipedia.org/wiki/Lag#cite_ref-bernier01_7-2) [***d***](https://en.wikipedia.org/wiki/Lag#cite_ref-bernier01_7-3) [***e***](https://en.wikipedia.org/wiki/Lag#cite_ref-bernier01_7-4) *Bernier, Yahn (2001).*[*"Latency Compensating Methods in Client/Server In-game Protocol Design and Optimization"*](http://developer.valvesoftware.com/wiki/Latency_Compensating_Methods_in_Client/Server_In-game_Protocol_Design_and_Optimization#Lag_Compensation)*.*[*Valve*](https://en.wikipedia.org/wiki/Valve_Corporation)*. Retrieved 17 September 2011.*
  8. [**^**](https://en.wikipedia.org/wiki/Lag#cite_ref-bf3-clientside_8-0) *Kertz, Alan (December 11, 2011).*[*"Re: We need someone to create a guide for the new Network Interpolation Setting slider"*](https://www.reddit.com/r/battlefield3/comments/n2oiy/we_need_someone_to_create_a_guide_for_the_new/c35xc2m)*. Retrieved 4 November 2013. BF3's hit model uses a combined client server model, a Hybrid Hit Detection. The client says to the server "Hey, I shot him!" and the server does a check against the position of the two targets and determines if the player could reasonably have hit that target and then applies the damage.*
  9. [**^**](https://en.wikipedia.org/wiki/Lag#cite_ref-ro2-lag_9-0) *Gibson, John (5 December 2010).*[*"Re: Will HoS present the netcode disadvantages of UE3?"*](http://forums.tripwireinteractive.com/showpost.php?p=662931&postcount=9)*.*[*Tripwire Interactive*](https://en.wikipedia.org/wiki/Tripwire_Interactive)*. Retrieved 18 September 2011.*
  10. [**^**](https://en.wikipedia.org/wiki/Lag#cite_ref-aldridge11_10-0) *Aldridge, David (2011).*[*"I Shot You First: Networking the Gameplay of HALO: REACH"*](http://www.gdcvault.com/play/1014345/I-Shot-You-First-Networking)*. Game Developers Conference 2011.*[*GDC*](https://en.wikipedia.org/wiki/Game_Developers_Conference)*Vault.*
  11. [**^**](https://en.wikipedia.org/wiki/Lag#cite_ref-D8_Video:_OnLive_demoed_on_iPad,_PC,_Mac,_Console,_iPhone_11-0) [*"D8 Video:OnLive demoed on iPad, PC, Mac, Console, iPhone"*](http://video.allthingsd.com/video/d8-video-onlive-demo/9D57A2C6-24ED-4351-8266-F3F7BA0C4D18/)*. Wall Street Journal. 2010-08-09. Retrieved 2010-08-19.*
  12. [**^**](https://en.wikipedia.org/wiki/Lag#cite_ref-12) [*"Beta Testing at the Speed of Light"*](https://web.archive.org/web/20121216005734/http:/blog.onlive.com/2010/01/21/beta-testing-at-the-speed-of-light/)*. OnLive. 2010-01-21. Archived from*[*the original*](http://blog.onlive.com/2010/01/21/beta-testing-at-the-speed-of-light/)*on 2012-12-16. Retrieved 2010-01-23.*

[Categories](https://en.wikipedia.org/wiki/Help:Category):

* [Online games](https://en.wikipedia.org/wiki/Category:Online_games)
* [Computer networks](https://en.wikipedia.org/wiki/Category:Computer_networks)
* [Multiplayer video games](https://en.wikipedia.org/wiki/Category:Multiplayer_video_games)
* [Video game terminology](https://en.wikipedia.org/wiki/Category:Video_game_terminology)