**Stateful vs Stateless: Full Difference**

February 23, 2022

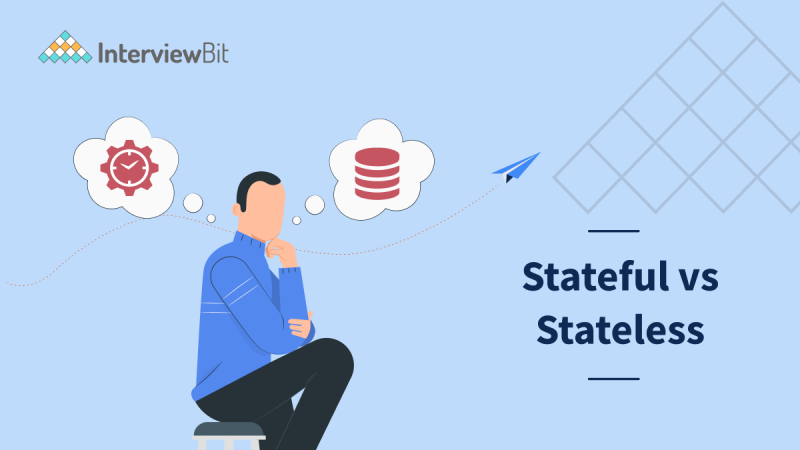


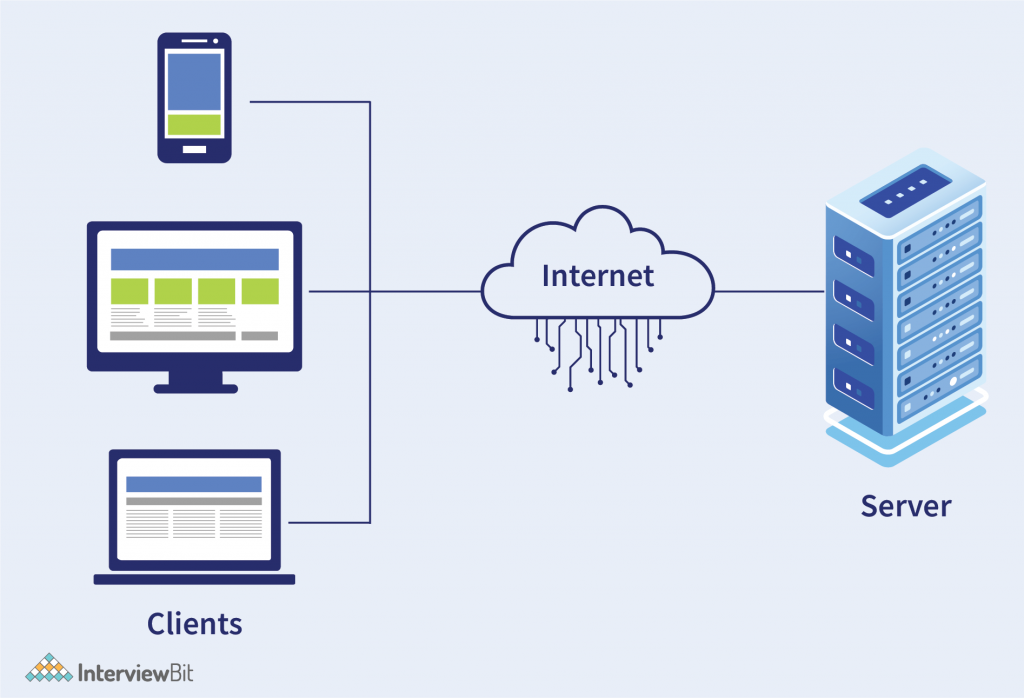
Table Of Contents

show

* [Introduction](https://www.interviewbit.com/blog/stateful-vs-stateless/#introduction)
* [What is Stateful?](https://www.interviewbit.com/blog/stateful-vs-stateless/#what-is-stateful)
  + [Advantages of Stateful](https://www.interviewbit.com/blog/stateful-vs-stateless/#advantages-of-stateful)
  + [Disadvantages of Stateful](https://www.interviewbit.com/blog/stateful-vs-stateless/#disadvantages-of-stateful)
* [What is Stateless?](https://www.interviewbit.com/blog/stateful-vs-stateless/#what-is-stateless)
  + [Advantages of Stateless](https://www.interviewbit.com/blog/stateful-vs-stateless/#advantages-of-stateless)
  + [Disadvantages of Stateless](https://www.interviewbit.com/blog/stateful-vs-stateless/#disadvantages-of-stateless)
* [Key Differences](https://www.interviewbit.com/blog/stateful-vs-stateless/#key-differences)
* [Stateful vs Stateless: Full Difference](https://www.interviewbit.com/blog/stateful-vs-stateless/#stateful-vs-stateless-full-difference)
* [Conclusion](https://www.interviewbit.com/blog/stateful-vs-stateless/#conclusion)
* [FAQs](https://www.interviewbit.com/blog/stateful-vs-stateless/#faqs)

**Introduction**

As the world rapidly moves towards a future where technology plays an integral part in our lives, we may have heard the word – ‘network protocol’ quite often. So what is network protocol? A network protocol is a set of rules that govern how data is formatted, sent, and received by computer network devices, ranging from servers and routers to endpoints, regardless of their underlying infrastructures, designs, or standards. Devices on both sides of a communication exchange must accept and follow protocol norms in order to send and receive data correctly.

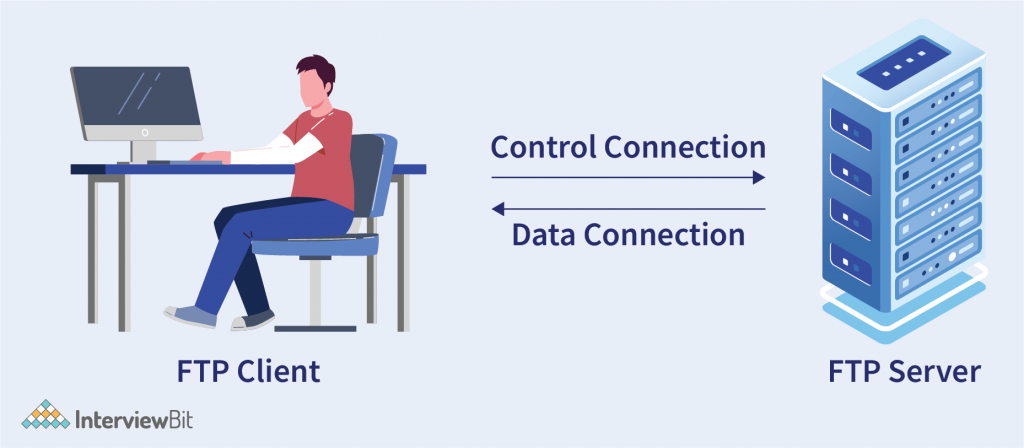


Network protocols can be broadly classified into two types – Stateful and Stateless.  
The word ‘state’ refers to the condition or the quality of being at a given point of time. Whether a protocol is stateful or stateless depends on the length of interaction a client has with it and how much of the information is stored. We need to look at the requirement basis of server or server-side software to save information on the status or details of sessions (here, a session refers to the temporary interchange of information between server and client), in order to find out the differences between the two. When it comes to definitions,’ stateful’ and ‘stateless’ have a form of mutually reliant relationship. To comprehend one, you must comprehend the other. Let us explore the topic of Stateful vs Stateless in the next few sections.

**What is Stateful?**

To understand what the word Stateful refers to, let us look at an analogy. Consider the scenario of a telephone call. In this case, the connection is maintained from the beginning to the end of the call to ensure continuous communication. The connection is validated first, and only then will the session be established indefinitely until the end of the talk. This is exactly what a stateful protocol is. If a client delivers a request to the server in a stateful protocol, it expects a response of some sort. If it does not receive a response, it will send the request again.

However, the major feature of stateful is that it maintains the state of all its sessions, be it an authentication session, or a client’s request for information. Stateful are those that may be used repeatedly, such as online banking or email. They’re carried out in the context of prior transactions in which the states are stored, and what happened in previous transactions may have an impact on the current transaction. Because of this, stateful apps use the same servers every time they perform a user request. An example of stateful is FTP (File Transfer Protocol). For the course of an FTP session, which often includes many data transfers, the client establishes a Control Connection. After this, the data transfer takes place.



So, what are the advantages and disadvantages of being Stateful?

**Advantages of Stateful**

* Stateful protocol keeps track of the connection information, and as a result, delivers superior performance because of continually keeping track of information.
* Stateful protocols are more intuitive because they can maintain data on the server between two requests.
* They can improve performance when data retrieval is required only once.

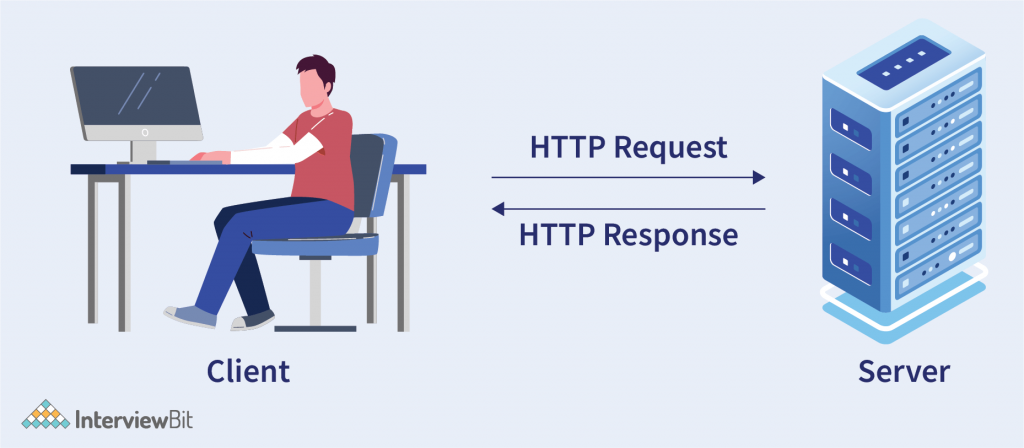
**Disadvantages of Stateful**

* Stateful protocol requires memory allocation in order to store data.
* In the event of inefficient maintenance of session storage, there can be a decrease in the performance. It requires continuous management of the service’s full lifecycle.
* These protocols are highly dependent on the server-side state.
* Usually, stateful protocols require backing storage.
* Since the state is maintained, stateful is not very secure.

**What is Stateless?**

In order to comprehend what Stateless means, let us consider a scenario just like we did in the case of Stateful. Consider the event of sending an SMS. Here, the receiver’s availability is not confirmed, and the sender just sends the SMS to the recipient. There is no confirmation from the receiving device to the sending device that the message has been received. Despite being transmitted, the communication may or may not be received. There can be no cross-verification of status or retries. This is what stateless is all about.

A stateless protocol is one in which the receiver is not required to keep session state from previous requests. The sender sends relevant session state to the receiver in such a way that each request may be interpreted without reference to prior requests’ session state, which the receiver retains. HTTP (HyperText Transfer Protocol) is an example of Stateless Protocol because each request is executed independently of the requests that came before it. This implies that once a transaction is completed, the connection between the browser and the server is also terminated.



**Advantages of Stateless**

* Since the monitoring system does not have to look beyond a single request to determine its whole nature, visibility of the protocol is improved.
* It is easier to recover from partial failures like crashes since no state is maintained, which improves reliability.
* The server does not have to store session state between requests, hence, scalability  is enhanced as deploying the services to any number of servers is possible, and implementation is simplified even more.
* It only necessitates a small number of resources because the system doesn’t need to keep track of communication over numerous lines, as well as session information.
* In Stateless Protocols, each individual communication is unconnected and distinct from the ones that come before or after it.
* Here, each packet of data travels on its own. There is no need to refer to another packet in these packets.

**Disadvantages of Stateless**

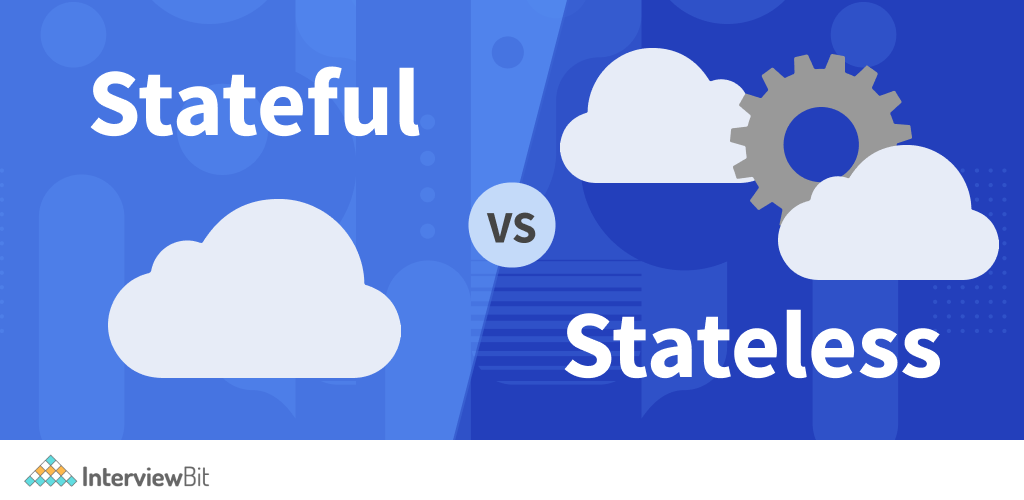
* It may be essential to include additional information in each request, and as a result, the server will need to interpret this new information.
* They may degrade network performance by increasing the amount of repetitive data delivered in a series of requests, which cannot be saved and reused.
* They are inherently less capable as they do not store information about a particular user session.

**Key Differences**

Now that we have a basic understanding of what each comprises, we can begin to clarify the primary differences between stateful and stateless.

The most significant distinction between stateful and stateless is that stateless do not “save” data, whereas stateful applications do. And as a result, the server doesn’t need to preserve server information or details of its sessions, whereas this needs to be done in stateful. A stateful application’s ability to maintain its state is crucial, but any data that goes through a stateless service is often short-lived. Any linked storage is usually transient. If the container is restarted, for example, any data stored will be lost. Running a single instance of a database for testing can be a fairly simple task when it comes to stateless. However, managing the whole life cycle of a stateful app’s service can be quite difficult. Production deployment and operation are two factors that contribute to this complexity. Both of these require highly available deployment, scaling, and error handling techniques. Each stateful data service requires or supports a certain form of storage. Furthermore, identifying the type of backing storage for a stateful application is notoriously difficult. When it comes to statelessness, none of this is required.

**Stateful vs Stateless: Full Difference**



| **Characteristics** | **Stateful** | **Stateless** |
| --- | --- | --- |
| Definition | Stateful Protocols require the server to save the state of a process. | Stateless Protocols do not need the server to save the state of a process. |
| Response mechanism | Stateful expects a response and if no answer is received, the request is resent. | In stateless, the client sends a request to a server, which the server responds to based on the state of the request. |
| Design complexity | This makes the design heavy and complex since data needs to be stored. | Server design is simplified in this case. |
| Requirement of Server | The server is required to store and save status information and details of sessions. | No server is needed for data storage. |
| Dependency | Server and Client are tightly coupled, as in extremely interdependent on each other. | Server and Client are more independent and hence, loosely coupled. |
| Transaction Handling | Transaction handling is relatively slow in the stateful protocol. | This is relatively faster in the stateless protocol. |
| Implementation | They are logically heavy to implement. | They are easy to implement. |
| Functioning after a crash | Since stateful protocols need to store data regarding the sessions, once the crash occurs, all the stored data is lost. Hence, it doesn’t work very well after a crash occurs. | In the event of a crash, stateless protocols work better because there doesn’t exist a state that needs to be restored. A server that failed during the crash can simply be restarted. |
| Design | The server design is complex to implement. | The server design is simpler to implement. |
| Working State | They react only by the current state of a transaction or request. | They act independently by taking the previous or next request into consideration. |
| Requests | Requests are always dependent on the server-side. | Requests are self-contained and not dependent on the server side. |
| Userbase | These are a thing of the past, and the dynamic user base is very less. | These are the future because more and more industries are moving towards statelessness. |
| Servers Specifications | The same server must be utilized to process every request. | Different servers can be used to process different information at a time. |
| Scaling Architecture | Scaling architecture is difficult and complex. | It is relatively easier to scale architecture. |
| Programming | It is difficult to code as one of the salient features here is data storage. | It is much easier to code. |
| Examples | Telnet, FTP (File Transfer Protocol), etc. | HTTP, UDP (User Datagram Protocol), DNS (Domain Name System), etc. |

**Conclusion**

In a world where clients were only thin interfaces to more powerful servers, stateful architecture made sense and was a logical solution. Having a strong coupling between users and servers wasn’t an issue when services just needed to scale to hundreds or thousands of users. We’ve had to change not only hardware and software, but also design patterns and concepts, now that we have powerful client computers and web services are frequently required to grow to millions or even billions of users. And this is where stateless comes into the picture. When it comes to choosing between stateless and stateful, the answer depends upon the requirements that need to be fulfilled. Both have their own sets of pros and cons. Hopefully, the distinctions with stateful vs stateless programming are now much clearer. Grasping the differences between them is crucial when it comes to API, which leads us to gain a deeper understanding of API.

**FAQs**

**Q. What is the difference between stateful and stateless?**  
A. The major difference between stateful and stateless is whether or not they store data regarding their sessions, and how they respond to requests. Stateful services keep track of sessions or transactions and respond to the same inputs in different ways depending on their history. Clients maintain sessions for stateless services, which are focused on activities that manipulate resources rather than the state.

**Q. Is stateless better than stateful?**  
A. In most cases, stateless is a better option when compared with stateful. However, in the end, it all comes down to your requirements. If you only require information in a transient, rapid, and temporary manner, stateless is the way to go. Stateful, on the other hand, might be the way to go if your app requires more memory of what happens from one session to the next.

**Q. Is HTTP stateful or stateless?**  
A. HTTP is stateless because it doesn’t keep track of any state information. In HTTP, each order or request is carried out in its own right, with no awareness of the demands that came before it.

**Q. Is REST API stateless or stateful?**  
A. REST APIs are stateless because, rather than relying on the server remembering previous requests, REST applications require each request to contain all of the information necessary for the server to understand it. Storing session state on the server violates the REST architecture’s stateless requirement. As a result, the client must handle the complete session state.