The English language is one of the most expressive, and confusing, in existence. Words can have different meaning based not only on context, but on placement within a given sentence. Add in the twists that come from technical jargon and suddenly you’ve got words meaning completely different things. This is evident in the use of persistent and persistence.

While the conceptual basis of persistence and persistent are essentially the same, in reality they refer to two different technical concepts.

Both persistent and persistence relate to the handling of connections. The former is often used as a general description of the behavior of HTTP and, necessarily, TCP connections, though it is also used in the context of database connections. The latter is most often related to TCP/HTTP connection handling but almost exclusively in the context of load-balancing.

**Persistent**

Persistent connections are connections that are kept open and reused. The most commonly implemented form of persistent connections are HTTP, with database connections a close second.

Persistent HTTP connections were implemented as part of the HTTP 1.1 specification as a method of improving the efficiency of HTTP in general. Before HTTP 1.1 a browser would generally open one connection per object on a page in order to retrieve all the appropriate resources. As the number of objects in a page grew, this became increasingly inefficient and significantly reduced the capacity of web servers while causing browsers to appear slow to retrieve data. HTTP 1.1 and the Keep-Alive header in HTTP 1.0 were aimed at improving the performance of HTTP by reusing TCP connections to retrieve objects. They made the connections persistent such that they could be reused to send multiple HTTP requests using the same TCP connection.

Similarly, this notion was implemented by proxy-based load-balancers as a way to improve performance of web applications and increase capacity on web servers. Persistent connections between a load-balancer and web servers is usually referred to as TCP multiplexing. Just like browsers, the load-balancer opens a few TCP connections to the servers and then reuses them to send multiple HTTP requests.

Persistent connections, both in browsers and load-balancers, have several advantages:

- Less network traffic due to less TCP setup/teardown. It requires no less than 7 exchanges of data to set up and tear down a TCP connection, thus each connection that can be reused reduces the number of exchanges required resulting in less traffic.
- Improved performance. Because subsequent requests do not need to setup and tear down a TCP connection, requests arrive faster and responses are returned quicker.
- TCP has built-in mechanisms, for example window sizing, to address network congestion. Persistent connections give TCP the time to adjust itself appropriately to current network conditions, thus improving overall performance. Non-persistent connections are not able to adjust because they are open and almost immediately closed.
- Less server overhead. Servers are able to increase the number of concurrent users served because each user requires fewer connections through which to complete requests.
Persistence

Persistence, on the other hand, is related to the ability of a load-balancer or other traffic management solution to maintain a virtual connection between a client and a specific server.

Persistence is often referred to in the application delivery networking world as "stickiness" while in the web and application server demesne it is called "server affinity". Persistence ensures that once a client has made a connection to a specific server that subsequent requests are sent to the same server. This is very important to maintain state and session-specific information in some application architectures and for handling of SSL-enabled applications.

![Examples of Persistence](image)

- **Hash Load Balancing and Persistence**
- **LTM Source Address Persistence**
- **Enabling Session Persistence**
- **20 Lines or Less #7: JSessionID Persistence**

Examples of Persistence: When the first request is seen by the load-balancer it chooses a server. On subsequent requests the load-balancer will automatically choose the same server to ensure continuity of the application or, in the case of SSL, to avoid the compute intensive process of renegotiation. This persistence is often implemented using cookies but can be based on other identifying attributes such as IP address. Load-balancers that have evolved into application delivery controllers are capable of implementing persistence based on any piece of data in the application message (payload), headers, or at in the transport protocol (TCP) and network protocol (IP) layers.

Some advantages of persistence are:

- **Avoid renegotiation of SSL.** By ensuring that SSL enabled connections are directed to the same server throughout a session, it is possible to avoid renegotiating the keys associated with the session, which is compute and resource intensive. This improves performance and reduces overhead on servers.
- **No need to rewrite applications.** Applications developed without load-balancing in mind may break when deployed in a load-balanced architecture because they depend on session data that is stored only on the original server on which the session was initiated. Load-balancers capable of session persistence ensure that those applications do not break by always directing requests to the same server, preserving the session data without requiring that applications be rewritten.

**Summize**

So **persistent** connections are connections that are kept open so they can be reused to send multiple requests, while **persistence** is the process of ensuring that connections and subsequent requests are sent to the same server through a load-balancer or other proxy device.

Both are important facets of communication between clients, servers, and mediators like load-balancers, and increase the overall performance and efficiency of the infrastructure as well as improving the end-user experience.
Persistent and Persistence, What's the Difference?

Lori MacVittie, 2008-11-07

The English language is one of the most persistent languages. While the conceptual basis of persistence relates to the handling of connections. The former is often used as a general description of the behavior of HTTP and, necessarily, TCP connections, though it is also used in the context of database delivery controllers. Although the term persistence is related to the ability of a load-balancer or other traffic management solution to maintain a virtual connection between a client and a specific server, it is possible to avoid renegotiating the keys associated with the session, which is computationally expensive.

So what is the difference between persistent and session persistence?

Both are important facets of communication between clients, servers, and mediators like load-balancers, and increase the overall performance and efficiency of the infrastructure as well as improving the end-user experience.

Persistent connections are connections that are kept open so they can be reused to send multiple requests, while session persistence refers to the ability of an application or, in the case of SSL, to avoid the compute intensive renegotiation of encryption keys associated with the session. When the first request is seen by the load-balancer it chooses a specific server. On subsequent requests the load-balancer will automatically choose the same server to ensure continuity of the session.

The first notion was implemented by traditional load-balancers as a way to improve performance of web applications and increase capacity on web servers. Persistent connections between a load-balancer and web servers is similar. This notion was implemented by proxy-based load-balancers as a way to improve performance of web applications by reusing the TCP connection. HTTP 1.0 header in HTTP 1.0 were aimed at improving the performance of HTTP connections a close second. HTTP 1.1 specification as a method of improving the persistent connections a close second. Persistent HTTP connections were implemented as part of the HTTP 1.1 specification as a method of improving the persistent connections a close second. Persistent HTTP connections were implemented as part of the HTTP 1.1 specification as a method of improving the persistent connections a close second. Persistent HTTP connections were implemented as part of the HTTP 1.1 specification as a method of improving the performance.