

# Setting TIME\_WAIT TCP

Asked 16 years ago   Modified 8 years, 6 months ago   Viewed 147k times

---



83



We're trying to tune an application that accepts messages via TCP and also uses TCP for some of its internal messaging. While load testing, we noticed that response time degrades significantly (and then stops altogether) as more simultaneous requests are made to the system. During this time, we see a lot of TCP connections in `TIME_WAIT` status and someone suggested lowering the `TIME_WAIT` environment variable from it's default 60 seconds to 30.

From [what I understand](#), the `TIME_WAIT` setting essentially sets the time a TCP resource is made available to the system again after the connection is closed.

I'm not a "network guy" and know very little about these things. I need a lot of what's in that linked post, but "dumbed down" a little.

- I think I understand why the `TIME_WAIT` value can't be set to 0, but can it safely be set to 5? What about 10? What determines a "safe" setting for this value?
- Why is the default for this value 60? I'm guessing that people a lot smarter than me had good reason for selecting this as a reasonable default.

- What else should I know about the potential risks and benefits of overriding this value?

tcp

network-protocols

Share

Improve this question

Follow

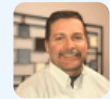
edited Jun 19, 2015 at 14:05



Mr Fooz

112k ● 7 ● 76 ● 103

asked Dec 3, 2008 at 13:35



Vinnie

12.7k ● 15 ● 61 ● 80

---

Also you don't want to set it to too high:

[stackoverflow.com/questions/1803566/...](https://stackoverflow.com/questions/1803566/...) – Pacerier Jan 23, 2016 at 4:04

---

7 Answers

Sorted by:

Highest score (default)





A TCP connection is specified by the tuple (source IP, source port, destination IP, destination port).

**117**



The reason why there is a `TIME_WAIT` state following session shutdown is because there may still be live packets out in the network on their way to you (or from you which may solicit a response of some sort). If you were to re-create that same tuple and one of those packets showed up, it would be treated as a valid packet for your connection (and probably cause an error due to sequencing).



So the `TIME_WAIT` time is generally set to double the packets maximum age. This value is the maximum age your packets will be allowed to get to before the network discards them.

That guarantees that, before you're allowed to create a connection with the same tuple, all the packets belonging to previous incarnations of that tuple will be dead.

That generally dictates the minimum value you should use. The maximum packet age is dictated by network properties, an example being that satellite lifetimes are higher than LAN lifetimes since the packets have much further to go.

Share Improve this answer

[edited Jun 14, 2013 at 1:43](#)

Follow

answered Dec 3, 2008 at 13:46



paxdiablo

880k ● 241 ● 1.6k ● 2k

---

2 How can I determine the "maximum packet age"? Is this set be the OS, something on the network, or some software setting? BTW, the code "generating" most of these connections is a third party platform we don't have source for. Thanks for the great response! – [Vinnie](#) Dec 3, 2008 at 16:33

---

5 The real name for it is maximum segment lifetime, MSL. Not sure you can change this in Windows or even if you should - it's meant to be set based on network characteristics. Windows sets it to 120s, I think. All TCP params are in HKEY\_LOCAL\_MACHINE\System\CurrentControlSet\Services\Tcpip\Parameters. – [paxdiablo](#) Dec 3, 2008 at 22:19

---

Not sure you can change this in Windows or even if you should - it's meant to be set based on network characteristics. Windows sets it to 120s, I think. ☹ You can definitely [change it in Linux](#) and 120 is way too long. I haven't tested to confirm, but it looks like having such a long delay makes P2P hell on most routers unnecessarily. – [Synetech](#) Mar 24, 2013 at 0:41

---

@Synetech, it should only cause problems if you're cycling through a *huge* number of sessions. Otherwise, session cleanup on timeout should take care of it. I believe even torrents keep sessions open for a while, reusing them if possible though, of course, it depends on the client. – [paxdiablo](#) Mar 24, 2013 at 3:40 ✎

---

@paxdiablo, yes, *should*. I replaced my good old DI-524 with a "newer, better" DIR-625 which I *do* like, but since then, whenever I merely *open* a torrent client, my connection dies (no HTTP, NTP, ping, nothing). Other than the router, everything else is the same. I checked TCPView and found that the torrent client has plenty of lingering connections.

Closing it almost immediately lets me surf again, so obviously the router is being overwhelmed. Reducing the delay helps (I already reduced the #connections a lot). (Also, that Chrome opens countless connections even for the NTP doesn't help.)  
– [Synetech](#) Mar 24, 2013 at 13:39

---



23



Usually, only the endpoint that issues an 'active close' should go into TIME\_WAIT state. So, if possible, have your clients issue the active close which will leave the TIME\_WAIT on the client and NOT on the server.

See here:



<http://www.serverframework.com/asynchronevents/2011/01/time-wait-and-its-design-implications-for-protocols-and-scalable-servers.html> and <http://www.isi.edu/touch/pubs/infocomm99/infocomm99-web/> for details (the later also explains why it's not always possible due to protocol design that doesn't take TIME\_WAIT into consideration).

Share Improve this answer

edited Jun 24, 2011 at 17:36

Follow

answered Dec 3, 2008 at 15:40



[Len Holgate](#)

21.6k ● 4 ● 47 ● 94

---

2 Good point, server will still have to wait for the ACK from its FIN but that should take less time. It's also good practice for the initiator to shut down the session since only it generally knows when it's finished. – [paxdiablo](#) Dec 4, 2008 at 3:40

---

- 1 2nd link has expired. Try this instead:  
[web.archive.org/web/20170521010646/http://www.isi.edu/touch/...](http://web.archive.org/web/20170521010646/http://www.isi.edu/touch/) – [Abhishek Kumar](#) Feb 5, 2019 at 19:00
- 



10

Pax is correct about the reasons for `TIME_WAIT`, and why you should be careful about lowering the default setting.



A better solution is to vary the port numbers used for the originating end of your sockets. Once you do this, you won't really care about time wait for individual sockets.



For listening sockets, you can use `SO_REUSEADDR` to allow the listening socket to bind despite the `TIME_WAIT` sockets sitting around.

Share Improve this answer

answered Dec 3, 2008 at 14:05

Follow



[Darron](#)

21.6k ● 5 ● 51 ● 54

- 
- 18 I'll upvote any answer that begins with the phrase "Pax is correct". :-)) – [paxdiablo](#) Dec 4, 2008 at 3:29
- 

- 5 For very active machines with many thousands of active sockets it's actually possible to tie up all of the ephemeral ports in `TIME_WAIT`. Once that happens you can't open any more connections until some sockets finish waiting. Reducing the `TIME_WAIT` duration can help a lot. As mentioned by Len Holgate, it's much better to have the client initiate active close if at all possible, since that obviates the server of `TIME_WAIT` duties entirely. – [Sam Hanes](#) Aug 2, 2011 at 13:31
-

- 1 Originating port numbers are automatically varied by default, and that doesn't help TIME\_WAIT states at the server. The correct solution is to ensure the client closes first.  
– [user207421](#) Feb 13, 2017 at 0:10
- 



5



In Windows, you [can change](#) it [through the registry](#):

```
; Set the TIME_WAIT delay to 30 seconds (0x1E)
```

```
[HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services  
"TcpTimedWaitDelay"=dword:0000001E
```



Share Improve this answer

edited May 10, 2016 at 23:07

Follow

answered Mar 24, 2013 at 0:47



[Synetech](#)

9,915 ● 9 ● 69 ● 97

- 6 setting tcp\_fin\_timeout doesn't affect time\_wait - this is a common misconception. It's for a totally different thing (a FIN timeout, obviously). Lots of guides and such point to this setting but they're wrong. Look at tcp.h and you'll see that it's hard-coded (Linux). – [mcauth](#) Aug 11, 2014 at 10:49
- 



2

setting the tcp\_reuse is more useful than changing time\_wait, as long as you have the parameter (kernels 3.2 and above, unfortunately that disqualifies all versions of RHEL and XenServer).



Dropping the value, particularly for VPN connected users, can result in constant recreation of proxy tunnels on the outbound connection. With the default Netscaler (XenServer) config, which is lower than the default Linux config, Chrome will sometimes have to recreate the proxy tunnel up to a dozen times to retrieve one web page. Applications that don't retry, such as Maven and Eclipse P2, simply fail.

The original motive for the parameter (avoid duplication) was made redundant by a TCP RFC that specifies timestamp inclusion on all TCP requests.

Share Improve this answer

answered Jun 2, 2016 at 19:38

Follow



[andrew glynn](#)

81 ● 2



0



I have been load testing a server application (on linux) by using a test program with 20 threads.

In 959,000 connect / close cycles I had 44,000 failed connections and many thousands of sockets in TIME\_WAIT.



I set SO\_LINGER to 0 before the close call and in subsequent runs of the test program had no connect failures and less than 20 sockets in TIME\_WAIT.

Share Improve this answer

answered Apr 28, 2016 at 6:54

Follow



[Michael Taylor](#)

19 ● 3



---

2 And you also ran the risk of losing data. This is not the correct solution. SO\_LINGER is not something to be trifled with. The correct solution is to ensure the client closes first, typically by connection-pooling at the client and writing the server so as to handle multiple requests per connection. Just like HTTP 1.1 in fact. – [user207421](#) Feb 13, 2017 at 0:12

---

I should have been clearer in my response. I am not advocating setting the linger time to 0 just pointing out the problem I observed on Windows. On linux this problem did not happen. So it is system dependent and, probably, dependent on what system settings are active.  
– [Michael Taylor](#) May 4, 2017 at 0:43

---

It is not system-dependent. Resetting a socket like this loses all the data in flight. – [user207421](#) Sep 6, 2017 at 3:27 

---



TIME\_WAIT might not be the culprit.

-2

```
int listen(int sockfd, int backlog);
```



According to Unix Network Programming Volume1, backlog is defined to be the sum of completed connection queue and incomplete connection queue.



Let's say the backlog is 5. If you have 3 completed connections (ESTABLISHED state), and 2 incomplete connections (SYN\_RCVD state), and there is another connect request with SYN. The TCP stack just ignores the SYN packet, knowing it'll be retransmitted some other time. This might be causing the degradation.

At least that's what I've been reading. ;)

Share Improve this answer

answered Dec 3, 2008 at 18:04

Follow



yogman

4,131 ● 4 ● 26 ● 27


---

I'm pretty sure the backlog is only for connections that haven't yet reached ESTABLISHED; once they have, they're removed from the queue; they're only blocking more incoming connections until the (SYN,SYN/ACK,ACK) handshaking is complete, basically once the server returns from accept(). – [paxdiablo](#) Dec 4, 2008 at 3:35

---

- 1 (-1) No, the listen backlog is purely for connections that are not completely established; i.e. they have arrived at the TCP/IP stack but not yet been 'accepted'. If your listen backlog is too small then your server will refuse connections if connections come in more quickly than it can accept them. – [Len Holgate](#) Dec 4, 2008 at 9:06
- 

- 2 A minor misunderstanding. "completed connection queue. This queue contains an entry for each connection for which the three way handshake is completed. The socket is in the ESTABLISHED state. Each call to accept() removes the front entry of the queue." [sean.de/Solaris/soltune.html](http://sean.de/Solaris/soltune.html) – [yogman](#) Dec 4, 2008 at 18:55
- 

- 1 @LenHolgate A connection that hasn't been accepted can still be completely established, and such connections go on the backlog queue. There are other queues for incomplete connections. Whether incoming connections that would overflow the backlog are refused or ignored is system-dependent: Windows refuses, Unix etc ignores. – [user207421](#) Feb 13, 2017 at 0:13 
-