

UDP vs TCP, how much faster is it?

[closed]

Asked 16 years, 3 months ago Modified 1 year, 11 months ago

Viewed 285k times



231



Closed. This question is [opinion-based](#). It is not currently accepting answers.

💡 **Want to improve this question?** Update the question so it can be answered with facts and citations by [editing this post](#).

Closed 5 years ago.

The community reviewed whether to reopen this question 2 years ago and left it closed:

Original close reason(s) were not resolved

[Improve this question](#)

For general protocol message exchange, which can tolerate some packet loss. How much more efficient is UDP over TCP?

network-programming

tcp

udp

Share

Improve this question

Follow

edited May 14, 2014 at 14:16



Étienne

4,984 ● 2 ● 38 ● 64

asked Sep 6, 2008 at 22:46



Net Citizen

5,324 ● 8 ● 40 ● 51

-
- 1 You could also add the "tcp" tag since the question is about TCP, too. – [Cristian Ciupitu](#) Sep 17, 2008 at 17:03
-
- 5 What does "general protocol message exchange" mean ?
The question needs to clarify what efficiency is about. Do we want less latency for a small message ? Or, do we want a higher throughput for a continuous stream of data ?
– [Johan Boulé](#) Jun 1, 2014 at 23:27
-
- Tcp has more better features than UDP except the Speed.
– [RAJKUMAR NAGARETHINAM](#) Aug 20, 2015 at 10:49
-
- 4 The question of TCP vs. UDP speed is moot. The question in your headline actually doesn't match the body of the question. Both TCP and UDP packets travel at exactly the same speed on the same medium. – [Ron Maupin](#) Sep 18, 2015 at 15:02 ✎
-
- BBR, FEC, gradient ascent/descent are all part of programming and maths - common lets be sensible with this - the question needs editing, but it's very very relevant and is a good question in essence. – [Mrk Fldig](#) Dec 30, 2021 at 14:52 ✎
-

14 Answers

Sorted by:

Highest score (default)





306



People say that the major thing TCP gives you is reliability. But that's not really true. The most important thing TCP gives you is congestion control: you can run 100 TCP connections across a DSL link all going at max speed, and all 100 connections will be productive, because they all "sense" the available bandwidth. Try that with 100 different UDP applications, all pushing packets as fast as they can go, and see how well things work out for you.

On a larger scale, this TCP behavior is what keeps the Internet from locking up into "congestion collapse".

Things that tend to push applications towards UDP:

- Group delivery semantics: it's possible to do reliable delivery to a group of people much more efficiently than TCP's point-to-point acknowledgement.
- Out-of-order delivery: in lots of applications, as long as you get all the data, you don't care what order it arrives in; you can reduce app-level latency by accepting an out-of-order block.
- Unfriendliness: on a LAN party, you may not care if your web browser functions nicely as long as you're blitting updates to the network as fast as you possibly can.

But even if you care about performance, you probably don't want to go with UDP:

- You're on the hook for reliability now, and a lot of the things you might do to implement reliability can end up being slower than what TCP already does.
- Now you're network-unfriendly, which can cause problems in shared environments.
- Most importantly, firewalls will block you.

You can potentially overcome some TCP performance and latency issues by "trunking" multiple TCP connections together; iSCSI does this to get around congestion control on local area networks, but you can also do it to create a low-latency "urgent" message channel (TCP's "URGENT" behavior is totally broken).

Share Improve this answer

answered Sep 11, 2008 at 20:06

Follow



tqbf

9,101 ● 3 ● 24 ● 13

23 Good answer, I'd even go more general, "flow control" (as opposed to congestion control, which is a subset of flow control). Not only multiple TCP connections can share one link, but it would also prevent sender from overflowing receiver's buffer if they pause processing incoming data for any reason. – [Alex B](#) Jan 5, 2010 at 10:07


1 @AaronLS: *packet loss* and *RTT (roundtrip time)* increases (which can be seen as a proxy for *queuing delay*) are/can be (e.g: WiFi networks may lose packets without real congestion, fooling some TCP congestion control algorithms into congestion avoidance) congestion indicators. – [ninjalj](#) Mar 31, 2014 at 11:34 ✎

3 UDP is a bastard to deal with... I'm burnt out on this already. No matter what I do, I can't seem to find a balance of

performance, latency, throughput, reliability. Great for real-time things like things on timers... but I'm working on a TCP replacement using UDP and Forward Error Correction and this is much harder than I thought it was going to be.

Congestion control. A universal system that works on 1GB networks and Wireless networks all the same is a work of art. I feel like I'm trying to reassemble packets that were loaded into a shotgun. – [Jason Nelson](#) Feb 28, 2016 at 8:31

- 1 @AaronLS My own UDP implementation is working pretty great now actually. I've even tested it to china and back. Great performance on the LAN and around the world. – [Jason Nelson](#) Aug 21, 2016 at 16:28
-

- 1 Btw another one in TCP's favor is that it's inherently connection oriented, which greatly simplifies application client handling logic (`listen` -> `accept` -> client state is naturally independent of other clients). Handling multiple connections from a single client in particular becomes gnarly with UDP. And a point in UDP's favor is UDP stacks are *really* easy to implement, which is a huge plus on embedded systems (microcontrollers, FPGAs, etc., in particular a TCP implementation for an FPGA is generally something you just want to purchase from somebody else and not think about). – [Jason C](#) Mar 5, 2017 at 6:19 
-



In some applications TCP is faster (better throughput) than UDP.

105



This is the case when doing lots of small writes relative to the MTU size. For example, I read an experiment in which a stream of 300 byte packets was being sent over Ethernet (1500 byte MTU) and **TCP was 50% faster than UDP.**



The reason is because TCP will try and buffer the data and fill a full network segment thus making more efficient use of the available bandwidth.

UDP on the other hand puts the packet on the wire immediately thus congesting the network with lots of small packets.

You probably shouldn't use UDP unless you have a very specific reason for doing so. Especially since you can give TCP the same sort of latency as UDP by disabling the [Nagle algorithm](#) (for example if you're transmitting real-time sensor data and you're not worried about congesting the network with lot's of small packets).

Share Improve this answer

Follow

edited Jul 25, 2013 at 0:02



[TravisHendrickson](#)

51 ● 5

answered Mar 12, 2009 at 12:41



[Robert S. Barnes](#)

40.5k ● 32 ● 134 ● 182

5 I've actually done benchmarks to this effect. I was sending packets that were as large as UDP would support without throwing exceptions (in Java) and TCP was much faster. I would guess a lot of OS, driver, and hardware optimizations are part of this as well. – [Charlie](#) Aug 1, 2009 at 21:47

15 @Myforwik: First, this is not implementation defined, it is part of the TCP protocol. It's called the Nagle algorithm. It helps prevent what's commonly known as Silly Window Syndrome. Look up both terms. Second, there is no concept of packets from TCP's pov. Lastly, the book "Effective TCP/IP

Programming" dedicates a whole chapter to this subject and multiple other chapters to the related subject of knowing when to use TCP vs. UDP. I bring up this situation (which is actually quite common) because the OP asked a general question, and this is one of the many possible answers.

– [Robert S. Barnes](#) Jun 30, 2011 at 11:59

54 @Myforwik. When suggesting moronism in others, try to realise that we all have gaps in our knowledge -- you included. SO is, after all, a forum for knowledge-sharing. Pretty much all first person shooters use UDP, and it's rare for them to send packets at sizes anywhere near as large as the MTU. If you'd like to go and suggest to John Carmack what a moron he was for coming up with this approach, I'd encourage you to educate yourself thoroughly in this regard, first. 15 years, and an industry's worth of high-performance networking code doesn't lie down and die because you think its "moronic". – [Engineer](#) Dec 12, 2011 at 14:24 ✎

4 *"I read an experiment in which a stream of 300 byte packets was being sent over Ethernet (1500 byte MTU) and TCP was 50% faster than UDP."* - could you link this experiment?
– [Leviathan](#) Nov 17, 2014 at 13:53

4 @Leviathan It's in the book Effective TCP/IP Programming.
– [Robert S. Barnes](#) Nov 17, 2014 at 14:35



94



UDP is faster than TCP, and the simple reason is because its non-existent acknowledge packet (ACK) that permits a continuous packet stream, instead of TCP that acknowledges a set of packets, calculated by using the TCP window size and round-trip time (RTT).



For more information, I recommend the simple, but very comprehensible [Skullbox explanation \(TCP vs. UDP\)](#).



Share Improve this answer

edited Nov 14, 2019 at 6:44

Follow



Jarvis

8,564 ● 3 ● 32 ● 61

answered Sep 6, 2008 at 23:03



Fernando Barrocal

13.1k ● 9 ● 45 ● 51

24 There are actually many cases where TCP is actually faster than UDP. See my answer below. – [Robert S. Barnes](#) Feb 15, 2011 at 6:53

4 Which is faster depends entirely on the traffic characteristics. – [user172783](#) Feb 28, 2013 at 18:02

7 Although the answer may be correct, it doesn't answer the question, and repeats knowledge mentioned elsewhere repeatedly. Also fails to mention that Reliable UDP methods with ACK can be notably faster than TCP. – [Elliot Woods](#) Dec 10, 2015 at 12:56

Why would the additional ACK make TCP slower? From the receiver's perspective, once the packet has arrived it's passed to the application and the ACK is sent. It's not like the app is waiting for the server to ack the ack. – [lionello](#) Jan 6, 2021 at 0:00



with loss tolerant

34

Do you mean "with loss tolerance" ?



Basically, UDP is not "loss tolerant". You can send 100 packets to someone, and they might only get 95 of those packets, and some might be in the wrong order.





For things like video streaming, and multiplayer gaming, where it is better to miss a packet than to delay all the other packets behind it, this is the obvious choice

For most other things though, a missing or 'rearranged' packet is critical. You'd have to write some extra code to run on top of UDP to retry if things got missed, and enforce correct order. This would add a small bit of overhead in certain places.

Thankfully, some very very smart people have done this, and they called it TCP.

Think of it this way: If a packet goes missing, would you rather just get the next packet as quickly as possible and continue (use UDP), or do you actually need that missing data (use TCP). The overhead won't matter unless you're in a really edge-case scenario.

Share Improve this answer

answered Sep 7, 2008 at 20:19

Follow



[Orion Edwards](#)

123k ● 66 ● 245 ● 339

2 5 packets out of 100? It's quite a lot. I guess it's just an example. Question: in real situation how many packets can be lost? Because if it is for example 2 out of 10000 (plus minus 1), then I wouldn't worry about that. – [freakish](#) Mar 28, 2013 at 12:40

2 @freakish, yeah it was just an example. The actual amount of packet loss depends on your connection, upstream networks, etc. I used to play a lot of online games, and I would find that if it was just me using the internet connection,

I'd get no packet loss, but as soon as I'd launch a background download, I'd start to get some (maybe 10%-20%). This was about 5 years ago though, and faster internet connections may help. – [Orion Edwards](#) Mar 29, 2013 at 18:52

3 Internet routers drop udp before tcp – [user1496062](#) May 30, 2014 at 5:54



When speaking of "what is faster" - there are at least two very different aspects: throughput and latency.

31



If speaking about *throughput* - TCP's flow control (as mentioned in other answers), is extremely important and doing anything comparable over UDP, while certainly possible, would be a Big Headache(tm). As a result - using UDP when you need *throughput*, rarely qualifies as a good idea (unless you want to get an unfair advantage over TCP).



However, if speaking about *latencies* - the whole thing is completely different. While in the absence of packet loss TCP and UDP behave extremely similar (any differences, if any, being marginal) - after the packet is lost, the whole pattern changes drastically.

After any packet loss, TCP will wait for retransmit for at least 200ms (1sec per paragraph 2.4 of RFC6298, but practical modern implementations tend to reduce it to 200ms). Moreover, with TCP, even those packets which did reach destination host - will not be delivered to your app until the missing packet is received (i.e., the whole

communication is delayed by $\sim 200\text{ms}$) - BTW, this effect, known as Head-of-Line Blocking, is inherent to all reliable ordered streams, whether TCP or reliable+ordered UDP. To make things even worse - if the retransmitted packet is also lost, then we'll be speaking about delay of $\sim 600\text{ms}$ (due to so-called exponential backoff, 1st retransmit is 200ms , and second one is $200 \times 2 = 400\text{ms}$). If our channel has 1% packet loss (which is not bad by today's standards), and we have a game with 20 updates per second - such 600ms delays will occur on average every 8 minutes. And as 600ms is more than enough to get you killed in a fast-paced game - well, it is pretty bad for gameplay. These effects are exactly why gamedevs often prefer UDP over TCP.

However, when using UDP to reduce latencies - it is important to realize that merely "using UDP" is not sufficient to get substantial latency improvement, it is all about HOW you're using UDP. In particular, while RUDP libraries usually avoid that "exponential backoff" and use shorter retransmit times - if they are used as a "reliable ordered" stream, they still have to suffer from Head-of-Line Blocking (so in case of a double packet loss, instead of that 600ms we'll get about $1.5 \times 2 \times \text{RTT}$ - or for a pretty good 80ms RTT, it is a $\sim 250\text{ms}$ delay, which is an improvement, but it is still possible to do better). On the other hand, if using techniques discussed in <http://gafferongames.com/networked-physics/snapshot-compression/> and/or <http://ithare.com/udp-from-mog-perspective/#low-latency-compression> , it IS possible to eliminate Head-of-Line blocking entirely (so for a double-

packet loss for a game with 20 updates/second, the delay will be 100ms regardless of RTT).

And as a side note - if you happen to have access only to TCP but no UDP (such as in browser, or if your client is behind one of 6-9% of ugly firewalls blocking UDP) - there *seems* to be a way to implement UDP-over-TCP without incurring too much latencies, see here:

<http://ithare.com/almost-zero-additional-latency-udp-over-tcp/> (make sure to read comments too(!)).

Share Improve this answer

edited Jun 14, 2017 at 5:10

Follow

answered Jun 14, 2017 at 4:53



No-Bugs Hare

1,638 ● 15 ● 15



20



Which protocol performs better (in terms of throughput) - UDP or TCP - really depends on the network characteristics and the network traffic. Robert S. Barnes, for example, points out a scenario where TCP performs better (small-sized writes). Now, consider a scenario in which the network is congested and has both TCP and UDP traffic. Senders in the network that are using TCP, will sense the 'congestion' and cut down on their sending rates. However, UDP doesn't have any congestion avoidance or congestion control mechanisms, and senders using UDP would continue to pump in data at the same rate. Gradually, TCP senders would reduce their

sending rates to bare minimum and if UDP senders have enough data to be sent over the network, they would hog up the majority of bandwidth available. So, in such a case, UDP senders will have greater throughput, as they get the bigger pie of the network bandwidth. In fact, this is an active research topic - How to improve TCP throughput in presence of UDP traffic. One way, that I know of, using which TCP applications can improve throughput is by opening multiple TCP connections. That way, even though, each TCP connection's throughput might be limited, the sum total of the throughput of all TCP connections may be greater than the throughput for an application using UDP.

Share Improve this answer

answered Feb 28, 2012 at 13:59

Follow



[gjain](#)

4,518 ● 5 ● 40 ● 48

-
- 4 This is not correct routers will drop UDP before TCP . On a shared wire you can get drowned by UDP but what is likely to happen in an oversupply situation depends on the technology but its quite easy for UDP to degrade to the point of very little being sent just collisions. – [user1496062](#) May 30, 2014 at 5:56

I like your explanation but don't get one point. If UDP connections can get all the traffic (if bandwidth is low or data is high) in that case your application if using TCP is basicly held hostage to those using UDP. If all applications are using TCP then they "play nice" with each other. Then why allow UDP on the rauter in the first place? – [Igor Čordaš](#) Jun 15, 2014 at 19:26

@PSIXO: Well, TCP and UDP serve different application requirements, so both are used by applications. The implication of your suggestion is that we should have different networking infrastructure for TCP and UDP traffic - an expensive proposition, and certainly not something we can do now, especially with all the investment already done. That's why researchers are busy finding out alternative ways to balance the conflict in 'software'. – [gjain](#) Jun 16, 2014 at 18:43

Well essentially yes, having two infrastructures would be a perfect solution but unfortunately it is not plausible. What I wanted to say with my comment was that you are overstating UDP hit to TCP because if it was that high of a factor people would just disable UDP on the router (As they sometimes do in companies) if they need TCP to function fast. Keep in mind also that UDP packets have higher chance of being dropped than TCP. About the rest of the facts in your answer I fully agree and find it quite helpful but I just think you are overestimating certain effects. – [Igor Čordaš](#) Jun 18, 2014 at 11:29



13



Each TCP connection requires an initial handshake before data is transmitted. Also, the TCP header contains a lot of overhead intended for different signals and message delivery detection. For a message exchange, UDP will probably suffice if a small chance of failure is acceptable. If receipt must be verified, TCP is your best option.

Share Improve this answer

Follow

answered Sep 6, 2008 at 22:52



[Kyle Cronin](#)

79k ● 45 ● 151 ● 167



13



I will just make things clear. **TCP/UDP** are two cars are that being driven on the road. suppose that traffic signs & obstacles are Errors **TCP** cares for traffic signs, respects everything around. Slow driving because something may happen to the car. While **UDP** just drives off, full speed no respect to street signs. Nothing, a mad driver. **UDP** doesn't have error recovery, If there's an obstacle, it will just collide with it then continue. While **TCP** makes sure that all packets are sent & received perfectly, No errors , so , the car just passes obstacles without colliding. I hope this is a good example for you to understand, Why **UDP** is preferred in gaming. Gaming needs speed. **TCP** is preferred in downloads, or downloaded files may be corrupted.

Share Improve this answer

answered Sep 5, 2016 at 4:21

Follow



[Ahmed I. Elsayed](#)

2,110 ● 2 ● 18 ● 32

-
- 1 To be clear: "all packets are sent & received perfectly" is a pretty bad exaggeration. With TCP having only a 16-bit checksum, errors which go undetected by TCP, are much more frequent then they should be. In fact, there are chances to see such errors when downloading a mere 1G file (over a particularly bad connection). – [No-Bugs Hare](#) Aug 14, 2020 at 9:02
-



7

UDP is slightly quicker in my experience, but not by much. The choice shouldn't be made on performance but on the message content and compression techniques.



If it's a protocol with message *exchange*, I'd suggest that the very slight performance hit you take with TCP is more than worth it. You're given a connection between two end points that will give you everything you need. Don't try and manufacture your own reliable two-way protocol on top of UDP unless you're really, really confident in what you're undertaking.

Share Improve this answer

answered Sep 6, 2008 at 22:52

Follow



Andrew

12k ● 12 ● 71 ● 85



6

There has been some work done to allow the programmer to have the benefits of both worlds.

SCTP



It is an independent transport layer protocol, but it can be used as a library providing additional layer over UDP. The basic unit of communication is a message (mapped to one or more UDP packets). There is congestion control built in. The protocol has knobs and twiddles to switch on

- in order delivery of messages
- automatic retransmission of lost messages, with user defined parameters

if any of this is needed for your particular application.

One issue with this is that the connection establishment is a complicated (and therefore slow process)

Other similar stuff

- https://en.wikipedia.org/wiki/Reliable_User_Datagram_Protocol

One more similar proprietary experimental thing

- <https://en.wikipedia.org/wiki/QUIC>

This also tries to improve on the triple way handshake of TCP and change the congestion control to better deal with fast lines.

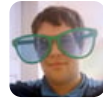
Update 2022: Quic and HTTP/3

QUIC (mentioned above) has been standardized through RFCs and even became the basis of HTTP/3 since the original answer was written. There are various libraries such as [lucas-clemente/quic-go](https://github.com/lucas-clemente/quic-go) or [microsoft/msquic](https://github.com/microsoft/msquic) or [google/quiche](https://github.com/google/quiche) or [mozilla/neqo](https://github.com/mozilla/neqo) (web-browsers need to be implementing this).

These libraries expose to the programmer reliable TCP-like streams on top the UDP transport. [RFC 9221 \(An Unreliable Datagram Extension to QUIC\)](https://www.rfc-editor.org/rfc/rfc9221) adds working with individual unreliable data packets.

Follow

answered Aug 2, 2013 at 10:02



[user7610](#)

28.5k ● 17 ● 142 ● 165



5



Keep in mind that TCP usually keeps multiple messages on wire. If you want to implement this in UDP you'll have quite a lot of work if you want to do it reliably. Your solution is either going to be less reliable, less fast or an incredible amount of work. There are valid applications of UDP, but if you're asking this question yours probably is not.

[Share](#) [Improve this answer](#)

answered Sep 6, 2008 at 23:04

[Follow](#)



[Leon Timmermans](#)

30.2k ● 2 ● 64 ● 110



4



If you need to quickly blast a message across the net between two IP's that haven't even talked yet, then a UDP is going to arrive at least 3 times faster, usually 5 times faster.

[Share](#) [Improve this answer](#)

answered Jun 30, 2011 at 11:06

[Follow](#)



[Myforwik](#)

3,568 ● 5 ● 36 ● 43

3 Any references? – [Gerard](#) Mar 31, 2020 at 14:28

3 UDP is going to arrive 3 to 5 times faster - or not going to arrive at all. ;-) – [No-Bugs Hare](#) Aug 14, 2020 at 9:03

"At least 3 times faster" screams for any credible source – [Nico Haase](#) Feb 9, 2021 at 10:13

OK wanna reopen the question, I'll give you why - UDP is A LOT faster if you get it right. – [Mrk Fldig](#) Dec 30, 2021 at 15:46



4



It is meaningless to talk about TCP or UDP without taking the network condition into account. If the network between the two point have a very high quality, UDP is absolutely faster than TCP, but in some other case such as the GPRS network, TCP may be faster and more reliability than UDP.



Share Improve this answer

Follow

edited Sep 18, 2015 at 7:26



[Michael Dorner](#)

20k ● 15 ● 94 ● 125

answered Jan 26, 2014 at 12:25



[zhanglongpan](#)

41 ● 1



2



The network setup is crucial for any measurements. It makes a huge difference, if you are communicating via sockets on your local machine or with the other end of the world.

Three things I want to add to the discussion:



1. You can find [here](#) a very good article about TCP vs. UDP in the context of game development.
2. Additionally, [iperf](#) ([jperf](#) enhance iperf with a GUI) is a very nice tool for answering your question yourself by measuring.
3. I implemented a benchmark in Python (see [this SO question](#)). In average of 10^6 iterations the difference for sending 8 bytes is about 1-2 microseconds for UDP.

Share Improve this answer

Follow

edited May 23, 2017 at 12:34



Community Bot

1 • 1

answered Sep 18, 2015 at 6:56



Michael Dorner

20k • 15 • 94 • 125

-
- 2 To make the benchmark relevant to the real world Internet, you need to re-run it with a packet loss simulator such as netem. If doing it right (and with simulated RTT of say 100ms and simulated packet loss of 1%), results will differ drastically. In short - while TCP and UDP latencies are indeed the same for zero packet loss - they start to differ A LOT for each lost packet. – [No-Bugs Hare](#) Jun 14, 2017 at 4:57
-