

Does TCP send a SYN/ACK on every packet or only on the first connection?

Asked 14 years, 3 months ago Modified 5 years, 2 months ago

Viewed 74k times



65



I have a TCP server that listens for an incoming client, then sends it one packet of data every second. I was wondering, does the SYN/ACK packet only get sent on initial connection, so it looks like this:

```
<client connect>
SYN
ACK
DATA
DATA
DATA
<client disconnect>
```

Or does it get sent with every packet, like this?

```
<client connect>
SYN
ACK
DATA

SYN
ACK
DATA

SYN
ACK
```

```
DATA
<client disconnect>
```

Also, if it's the first case, are there any benefits of UDP over TCP if you just keep the connection open over a long period of time?

network-programming

tcp

udp

client-server

packet

Share

Improve this question

Follow

asked Aug 30, 2010 at 21:47



Daniel T.

38.3k ● 37 ● 144 ● 207

3 There are no "packets" in TCP/IP. See the correct terminology here: [stackoverflow.com/questions/955369/...](http://stackoverflow.com/questions/955369/) – Joe Phillips Aug 30, 2010 at 23:31

8 @Phillips - TCP is a protocol layered over IP. There is no concept of segments until processed by TCP. During this process it is definitely acceptable to refer to incoming data as packets rather than segments, because they are after all only IP packets at such a point. Goes into TCP as IP packets, comes out as segments, messages etc. – JSON Nov 15, 2015 at 20:39

3 Answers

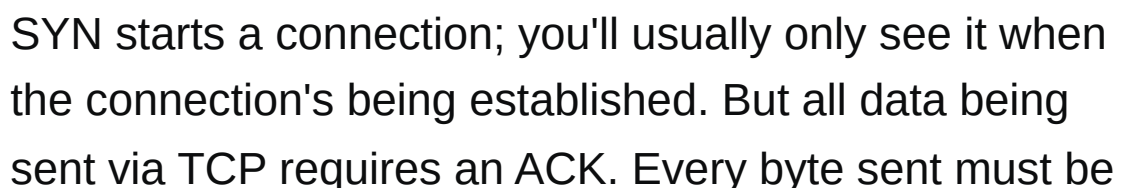
Sorted by:

Highest score (default)



It's kinda like:

123

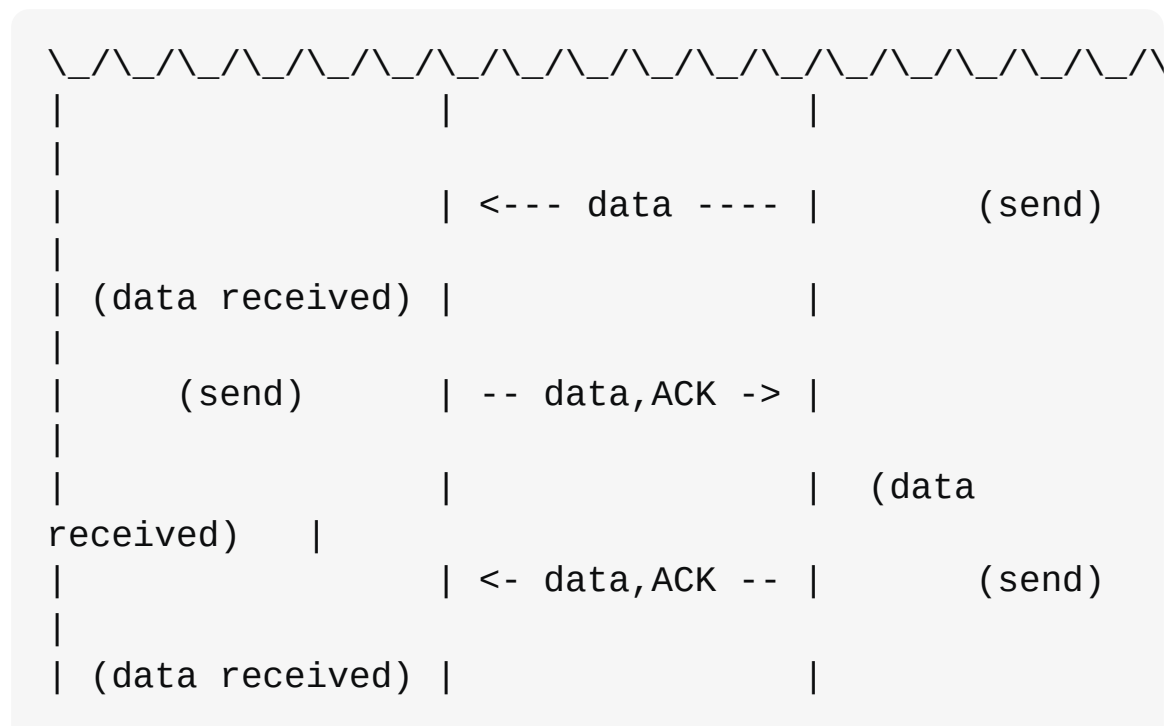


accounted for, or it will be retransmitted (or the connection reset (closed), in severe cases).

Actual connections aren't usually *exactly* like the diagram above, though, for two reasons:

- ACKs can build up, so one ACK can acknowledge everything received up to that point. That means you can acknowledge two or more sends with one ACK.
- An ACK is simply a flag and field in a TCP header. Sending one requires at least a header's worth of bandwidth, plus whatever the lower layers tack on. But data segments already include all that...so if you're sending data, you can send an ACK at the same time for free.

Most TCP/IP stacks try to reduce the number of naked ACKs without unduly risking retransmission or a connection reset. So a conversation like this one is quite possible:





As for UDP, there's no built-in concept of SYN and ACK -- UDP is by nature "unreliable", and not connection-oriented, so the concepts don't apply as much. Your acknowledgement will usually just be the server's response. But some application-layer protocols built on top of UDP will have some protocol-specific way of acknowledging data sent and received.

Share Improve this answer

[edited Sep 2, 2017 at 21:34](#)

Follow

answered Aug 30, 2010 at 21:57

cHao

86.4k ● 20 ● 146 ● 177

37 ACK can get complicated. It isn't for every data packet, but for however many have been received so there might be one ACK every 8 packets. The sending side has a *window* which is how much it will send before it *must* receive an ACK. Then there is Selective ACK which is used to say "Received bytes 2000-8000, but not 0-2000" – [Zan Lynx](#) Aug 30, 2010 at 22:22

1 User Datagram Protocol is often used in query-response protocols where a response to a query will demonstrate that it was received, and the lack of a repeated query will demonstrate to the responder that either its response was received or the originator of the query has given up (and the responder doesn't care which, since its proper response in either case is to do nothing further). – [supercat](#) Apr 30, 2013 at 22:31



21



SYN is only at the beginning.


ACK is on subsequent segments in either direction. The ACK will also define a window size. If the window size is 100 for example, the sender can send 100 segments before it expects to receive an ACK. E.g If sender sends 100 segments but segment number 50 gets lost, then receiver will get 1-49 & 51 -100. Receiver will then ACK for 50 (next segment it expects) and set window size to 1. Sender will resend 1 segment with sequence number 50. Receiver will then ACK for 101 and set the window size back up to a higher number.

Both are actually fields in the TCP header and can be sent with data, though the SYN and the first ACK typically are data-less.

So neither of the scenarios you describe is quite correct. The first is actually closer to reality, but all the data packets after the SYN do have to include an ACK, and also an acknowledgement number field which identifies the number of the next packet expected.

The end of a session also involves handshakes with FIN flagged packets and ACKs relating to them.

The exchanged sequence numbers are used to identify lost packets and enable a retry mechanism, and also to reassemble the entire stream of packets in the correct order.



Also, if it's the first case, are there any benefits of UDP over TCP if you just keep the connection open over a long period of time?

With UDP you can't just keep the connection open over a long period of time. There is no connection.

This sequence of SYN/ACK/FIN flags is what makes a connection.

With UDP, there are no SYNs or ACKs, so communication is one-way, delivery is not guaranteed and order is not preserved. But it has less overhead, so it's useful when

speed is more important than reliability, as for example in streaming media.

This is a bit simplified yet, but it's the best I can do at the moment.

There's much more on this in the [wikipedia entry on TCP](#) and of course in the RFCs.

Share Improve this answer

Follow

edited Oct 6, 2019 at 13:23



Ondrej Slinták

31.9k ● 21 ● 96 ● 127

answered Aug 30, 2010 at 23:07



Don Roby

41.1k ● 6 ● 95 ● 114

-
- 3 I'd also recommend the book "TCP/IP Illustrated, Volume 1 - The Protocols" by W. Richard Stevens in addition to reading Wikipedia and the RFCs. It's a bit easier on the brain :)

– [Michael J. Gray](#) Jun 20, 2014 at 5:27

Sender will resend 1 segment with sequence number 50. Receiver will then ACK for 101 shouldn't it be *Receiver will then ACK for 51*, since the last received segment was 50?

– [Rafael Eying](#) Jun 16, 2016 at 21:09

I don't understand the comment about 'communication is one-way'. That makes no sense at all. UDP is just a trivial, extremely thin layer over IP, and since it is merely IP with a small amount of chocolate sauce on top, you can send UDP packets in *both* directions. – [Cecil Ward](#) Jul 19, 2017 at 5:12

If a designer chooses to use UDP, it is done to gain higher speed performance and minimise the amount of traffic exchanged, or alternatively it is to allow full control over

communication methods. Using UDP, a designer can, if desired, build a new kind of protocol with complete freedom of choice. Some applications may not require reliable delivery, in-order delivery guarantees, or other benefits that protocols such as TCP or SCTP provide. However, a designer may have to do a lot more design work if using UDP, complicating the application code or ending up inventing a custom protocol. – [Cecil Ward](#) Jul 19, 2017 at 5:33

- 1 @RafaelEyng no since segments 51-100 are buffered on the receiver's end. When he receives the missing segment in the middle, he puts them in the correct order and now has everything from 1-100. There is no need to request segments you already have. – [David Trevor](#) Nov 2, 2018 at 14:31
-



Picture this: The original TCP standard RFC 793 allowed data to be sent with the first SYN packet though.

-1



However, that's not the case today. What you get is a separate SYN packet during initiation of the Three-Way-Handshake from the requestor of the connection.

Suppose A requests to connect with B, thus A sends a packet with a SYN bit set. B responds with an ACK to acknowledge receipt and sends A the ACK + SYN packets. Data can then be transmitted henceforth.



[Dordal has a very good explanation on this matter. Click this link here.](#)

Share Improve this answer

answered Dec 30, 2017 at 18:54

Follow



[StacknormalFlow](#)

19 ● 4

2 That link is broken. Which is why you shouldn't link to external stuff for answers. – [neuhhaus](#) Jun 8, 2021 at 10:34

RFC 793, as amended, is still the case today. Please provide an *RFC reference* for your claim. – [user207421](#) Aug 21, 2023 at 5:22
