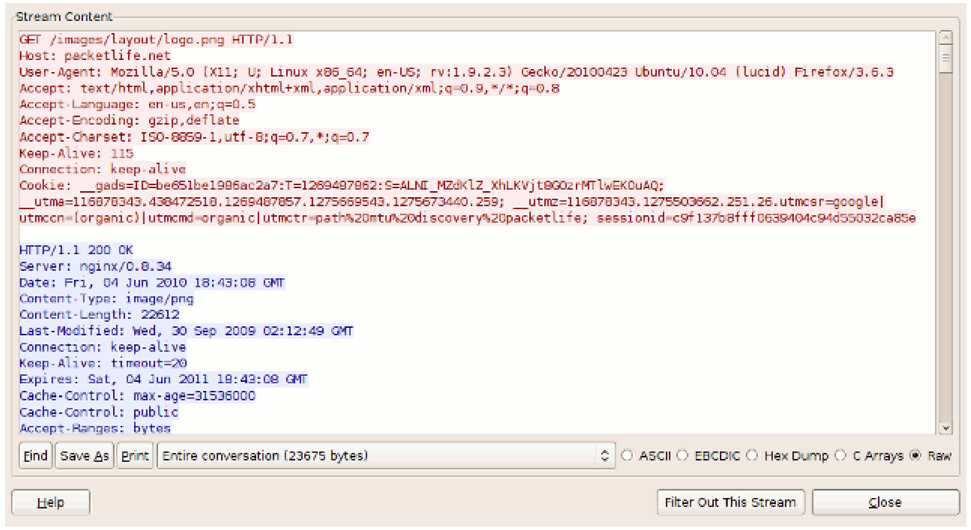
Understanding TCP Sequence and Acknowledgment Numbers

By stretch | Monday, June 7, 2010 at 2:15 a.m. UTC

If you're reading this, odds are that you're already familiar with TCP's infamous "three-way handshake," or "SYN, SYN/ACK, ACK." Unfortunately, that's where TCP education ends for many networkers. Despite its age, TCP is a relatively complex protocol and well worth knowing intimately. This article aims to help you become more comfortable examining TCP sequence and acknowledgement numbers in the Wireshark packet analyzer.

Before we start, be sure to open the example capture in Wireshark and play along.

The example capture contains a single HTTP request to a web server, in which the client web browser requests a single image file, and the server returns an HTTP/1.1 200 (OK) response which includes the file requested. You can right-click on any of the TCP packets within this capture and select **Follow TCP Stream** to open the raw contents of the TCP stream in a separate window for inspection. Traffic from the client is shown in red, and traffic from the server in blue.



```
GET /images/layout/logo.png HTTP/1.1
Host: packetlife.net
User-Agent: Mozilla/5.0 (X11; U; Linux x86_64; en-US; rv:1.9.2.3) Gecko/20100423 Ubuntu/10.04 (lucid) Firefox/3.6.3
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-us,en;q=0.5
Accept-Encoding: gzip,deflate
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
Keep-Alive: 115
Connection: keep-alive
Cookie: __gads=ID=be651be1986ac2a7:T=1269487862:S=ALNI_MZdKlZ_XhLKVjt8GOzrMTlwEKOuAQ;
__utma=116878343.438472518.1269487857.1275669543.1275673440.259; __utmz=116878343.1275503662.251.26.utmcsr=google|
utmccn=(organic)|utmcmd=organic|utmctr=path%20mtu%20discovery%20packetlife; sessionid=c9f137b8fff0639404c94d55032ca85e

HTTP/1.1 200 OK
Server: nginx/0.8.34
Date: Fri, 04 Jun 2010 18:43:08 GMT
Content-Type: image/png
Content-Length: 22612
Last-Modified: Wed, 30 Sep 2009 02:12:49 GMT
Connection: keep-alive
Keep-Alive: timeout=20
Expires: Sat, 04 Jun 2011 18:43:08 GMT
Cache-Control: max-age=31536000
Cache-Control: public
Accept-Ranges: bytes
```

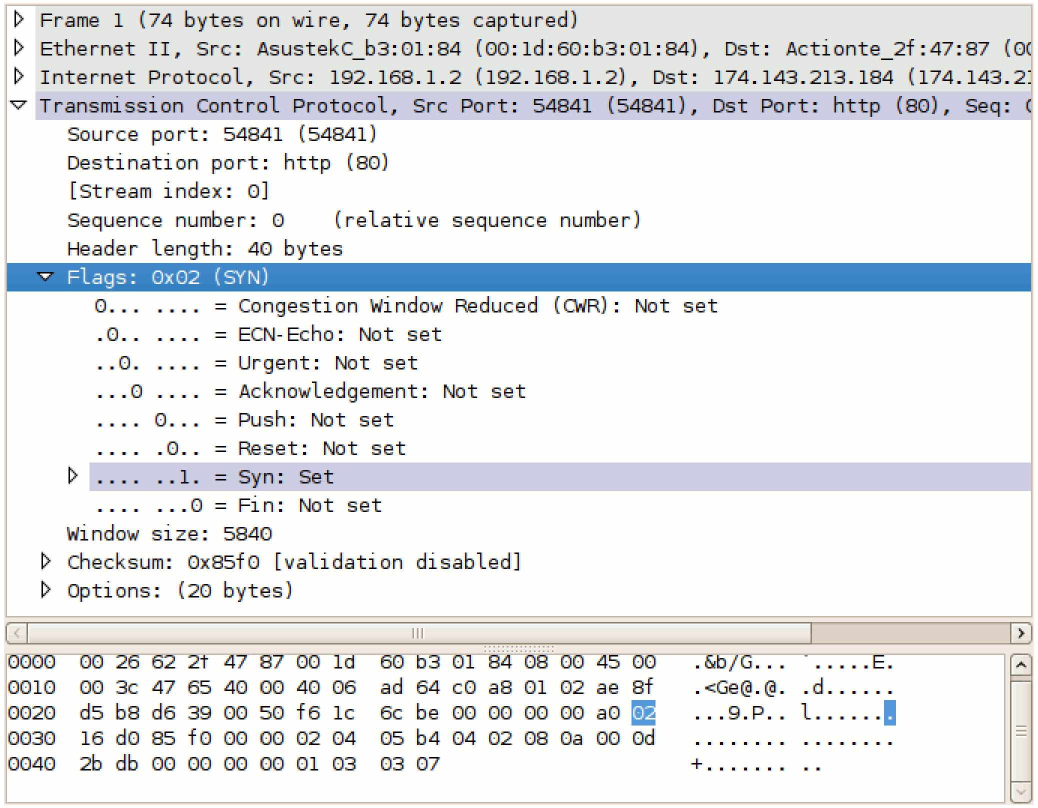
The Three-Way Handshake

TCP utilizes a number of flags, or 1-bit boolean fields, in its header to control the state of a connection. The three we're most interested in here are:

- **SYN** - (Synchronize) Initiates a connection
- **FIN** - (Final) Cleanly terminates a connection
- **ACK** - Acknowledges received data

As we'll see, a packet can have multiple flags set.

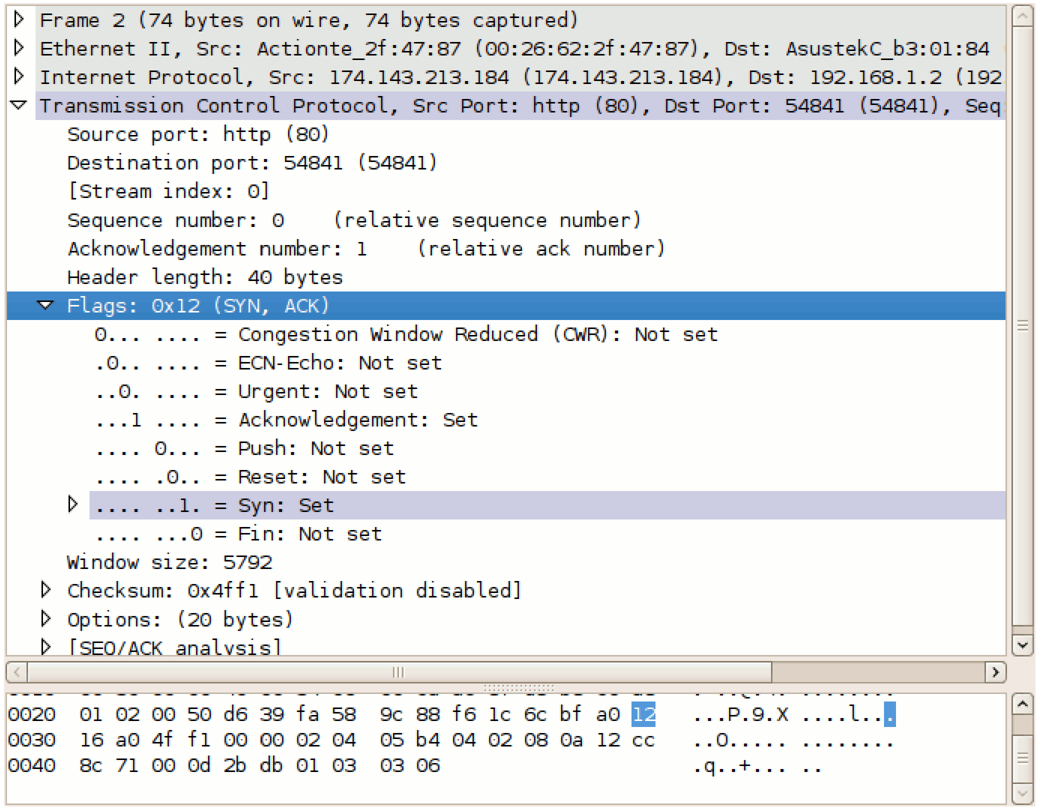
Select packet #1 in Wireshark and expand the TCP layer analysis in the middle pane, and further expand the "Flags" field within the TCP header. Here we can see all of the TCP flags broken down. Note that the SYN flag is on (set to 1).



```
▷ Frame 1 (74 bytes on wire, 74 bytes captured)
▷ Ethernet II, Src: AsustekC_b3:01:84 (00:1d:60:b3:01:84), Dst: Actionte_2f:47:87 (00
▷ Internet Protocol, Src: 192.168.1.2 (192.168.1.2), Dst: 174.143.213.184 (174.143.2
▽ Transmission Control Protocol, Src Port: 54841 (54841), Dst Port: http (80), Seq: 0
    Source port: 54841 (54841)
    Destination port: http (80)
    [Stream index: 0]
    Sequence number: 0    (relative sequence number)
    Header length: 40 bytes
  ▽ Flags: 0x02 (SYN)
      0... .... = Congestion Window Reduced (CWR): Not set
      .0.. .... = ECN-Echo: Not set
      ..0. .... = Urgent: Not set
      ...0 .... = Acknowledgement: Not set
      .... 0... = Push: Not set
      .... .0.. = Reset: Not set
    ▷ .... ..1. = Syn: Set
      .... ...0 = Fin: Not set
    Window size: 5840
  ▷ Checksum: 0x85f0 [validation disabled]
  ▷ Options: (20 bytes)
```

```
0000  00 26 62 2f 47 87 00 1d  60 b3 01 84 08 00 45 00   .&b/G... `.....E.
0010  00 3c 47 65 40 00 40 06  ad 64 c0 a8 01 02 ae 8f   .<Ge@.@. .d......
0020  d5 b8 d6 39 00 50 f6 1c  6c be 00 00 00 00 a0 02   ...9.P.. l.......
0030  16 d0 85 f0 00 00 02 04  05 b4 04 02 08 0a 00 0d   ........ ........
0040  2b db 00 00 00 00 01 03  03 07                     +....... ..
```

Now do the same for packet #2. Notice that it has two flags set: ACK to acknowledge the receipt of the client's SYN packet, and SYN to indicate that the server also wishes to establish a TCP connection.



```
▷ Frame 2 (74 bytes on wire, 74 bytes captured)
▷ Ethernet II, Src: Actionte_2f:47:87 (00:26:62:2f:47:87), Dst: AsustekC_b3:01:84
▷ Internet Protocol, Src: 174.143.213.184 (174.143.213.184), Dst: 192.168.1.2 (192
▽ Transmission Control Protocol, Src Port: http (80), Dst Port: 54841 (54841), Seq
    Source port: http (80)
    Destination port: 54841 (54841)
    [Stream index: 0]
    Sequence number: 0    (relative sequence number)
    Acknowledgement number: 1    (relative ack number)
    Header length: 40 bytes
  ▽ Flags: 0x12 (SYN, ACK)
      0... .... = Congestion Window Reduced (CWR): Not set
      .0.. .... = ECN-Echo: Not set
      ..0. .... = Urgent: Not set
      ...1 .... = Acknowledgement: Set
      .... 0... = Push: Not set
      .... .0.. = Reset: Not set
    ▷ .... ..1. = Syn: Set
      .... ...0 = Fin: Not set
    Window size: 5792
  ▷ Checksum: 0x4ff1 [validation disabled]
  ▷ Options: (20 bytes)
  ▷ [SEQ/ACK analysis]

0020  01 02 00 50 d6 39 fa 58  9c 88 f6 1c 6c bf a0 12   ...P.9.X ....l...
0030  16 a0 4f f1 00 00 02 04  05 b4 04 02 08 0a 12 cc   ..O..... ........
0040  8c 71 00 0d 2b db 01 03  03 06                     .q..+... ..
```

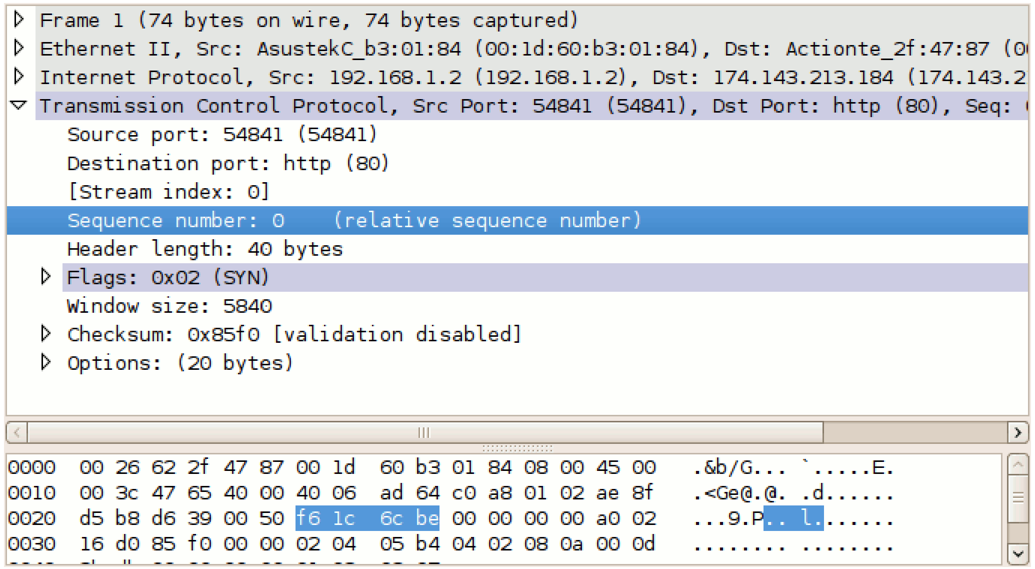
Packet #3, from the client, has only the ACK flag set. These three packets complete the initial TCP three-way handshake.

Sequence and Acknowledgment Numbers

The client on either side of a TCP session maintains a 32-bit *sequence number* it uses to keep track of how much data it has sent. This sequence number is included on each transmitted packet, and acknowledged by the opposite host as an *acknowledgement number* to inform the sending host that the transmitted data was received successfully.

When a host initiates a TCP session, its initial sequence number is effectively random; it may be any value between 0 and 4,294,967,295, inclusive. However, protocol analyzers like Wireshark will typically display *relative* sequence and acknowledgement numbers in place of the actual values. These numbers are relative to the initial sequence number of that stream. This is handy, as it is much easier to keep track of relatively small, predictable numbers rather than the actual numbers sent on the wire.

For example, the initial relative sequence number shown in packet #1 is 0 (naturally), while the ASCII decode in the third pane shows that the actual sequence number is 0xf61c6cbe, or 4129057982 decimal.

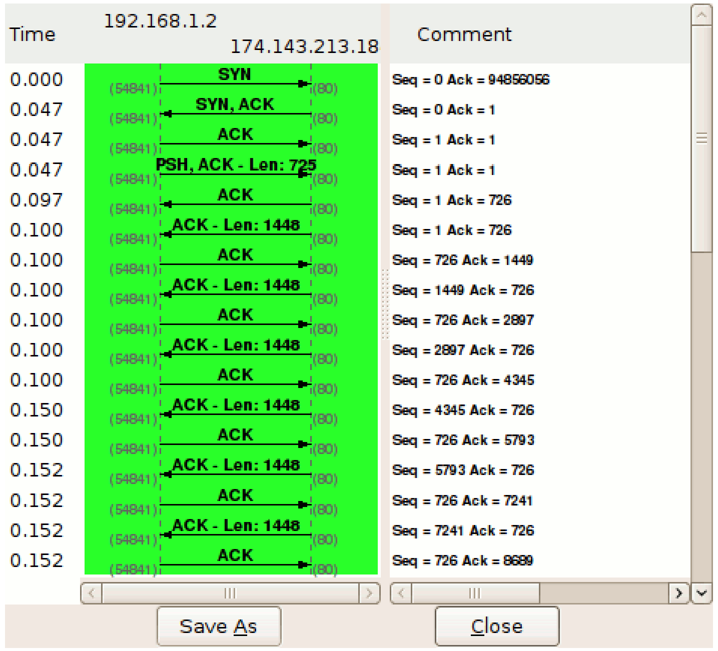


```
▷ Frame 1 (74 bytes on wire, 74 bytes captured)
▷ Ethernet II, Src: AsustekC_b3:01:84 (00:1d:60:b3:01:84), Dst: Actionte_2f:47:87 (0
▷ Internet Protocol, Src: 192.168.1.2 (192.168.1.2), Dst: 174.143.213.184 (174.143.2
▽ Transmission Control Protocol, Src Port: 54841 (54841), Dst Port: http (80), Seq:
    Source port: 54841 (54841)
    Destination port: http (80)
    [Stream index: 0]
    Sequence number: 0    (relative sequence number)
    Header length: 40 bytes
  ▷ Flags: 0x02 (SYN)
    Window size: 5840
  ▷ Checksum: 0x85f0 [validation disabled]
  ▷ Options: (20 bytes)

0000  00 26 62 2f 47 87 00 1d  60 b3 01 84 08 00 45 00   .&b/G... `.....E.
0010  00 3c 47 65 40 00 40 06  ad 64 c0 a8 01 02 ae 8f   .<Ge@.@. .d......
0020  d5 b8 d6 39 00 50 f6 1c  6c be 00 00 00 00 a0 02   ...9.P.. l.......
0030  16 d0 85 f0 00 00 02 04  05 b4 04 02 08 0a 00 0d   ........ ........
```

The display of relative sequence numbers can optionally be disabled by navigating to **Edit > Preferences...** and un-checking **Relative sequence numbers and window scaling** under TCP protocol preferences. However, be aware that the remainder of this article will reference relative sequence and acknowledgement numbers only.

To better understand how sequence and acknowledgement numbers are used throughout the duration of a TCP session, we can utilize Wireshark's built-in flow graphing ability. Navigate to **Statistics > Flow Graph...**, select **TCP flow** and click **OK**. Wireshark automatically builds a graphical summary of the TCP flow.



Time	192.168.1.2		174.143.213.18	Comment
0.000	(54841)	SYN →	(80)	Seq = 0 Ack = 94856056
0.047	(54841)	← SYN, ACK	(80)	Seq = 0 Ack = 1
0.047	(54841)	ACK →	(80)	Seq = 1 Ack = 1
0.047	(54841)	PSH, ACK - Len: 725 →	(80)	Seq = 1 Ack = 1
0.097	(54841)	← ACK	(80)	Seq = 1 Ack = 726
0.100	(54841)	← ACK - Len: 1448	(80)	Seq = 1 Ack = 726
0.100	(54841)	ACK →	(80)	Seq = 726 Ack = 1449
0.100	(54841)	← ACK - Len: 1448	(80)	Seq = 1449 Ack = 726
0.100	(54841)	ACK →	(80)	Seq = 726 Ack = 2897
0.100	(54841)	← ACK - Len: 1448	(80)	Seq = 2897 Ack = 726
0.100	(54841)	ACK →	(80)	Seq = 726 Ack = 4345
0.150	(54841)	← ACK - Len: 1448	(80)	Seq = 4345 Ack = 726
0.150	(54841)	ACK →	(80)	Seq = 726 Ack = 5793
0.152	(54841)	← ACK - Len: 1448	(80)	Seq = 5793 Ack = 726
0.152	(54841)	ACK →	(80)	Seq = 726 Ack = 7241
0.152	(54841)	← ACK - Len: 1448	(80)	Seq = 7241 Ack = 726
0.152	(54841)	ACK →	(80)	Seq = 726 Ack = 8689

Each row represents a single TCP packet. The left column indicates the direction of the packet, TCP ports, segment length, and the flag(s) set. The column at right lists the relative sequence and acknowledgement numbers in decimal. Selecting a row in this column also highlights the corresponding packet in the main window.

We can use this flow graph to better understand how sequence and acknowledgement numbers work.

Packet #1

Each side of a TCP session starts out with a (relative) sequence number of zero. Likewise, the acknowledgement number is also zero, as there is not yet a complementary side of the conversation to acknowledge.

(Note: The version of Wireshark used for this demonstration, 1.2.7, shows the acknowledgement number as an apparently random number. This believed to be a software bug; the initial acknowledgement number of a session should always be zero, as you can see from inspecting the hex dump of the packet.)

Packet #2

The server responds to the client with a sequence number of zero, as this is its first packet in this TCP session, and a relative acknowledgement number of 1. The acknowledgement number is set to 1 to indicate the receipt of the client's SYN flag in packet #1.

Notice that the acknowledgement number has been increased by 1 although no payload data has yet been sent by the client. This is because the presence of the SYN or FIN flag in a received packet triggers an increase of 1 in the sequence. (This does not interfere with the accounting of payload data, because packets with the SYN or FIN flag set do not carry a payload.)

Packet #3

Like in packet #2, the client responds to the server's sequence number of zero with an acknowledgement number of 1. The client includes its own sequence number of 1 (incremented from zero because of the SYN).

At this point, the sequence number for both hosts is 1. This initial increment of 1 on both hosts' sequence numbers occurs during the establishment of all TCP sessions.

Packet #4

This is the first packet in the stream which carries an actual payload (specifically, the client's HTTP request). The sequence number is left at 1, since no data has been transmitted since the last packet in this stream. The acknowledgement number is also left at 1, since no data has been received from the server, either.

Note that this packet's payload is 725 bytes in length.

Packet #5

This packet is sent by the server solely to acknowledge the data sent by the client in packet #4 while upper layers process the HTTP request. Notice that the acknowledgement number has increased by 725 (the length of the payload in packet #4) to 726; e.g., "I have received 726 bytes so far." The server's sequence number remains at 1.

Packet #6

This packet marks the beginning of the server's HTTP response. Its sequence number is still 1, since none of its packets prior to this one have carried a payload. This packet carries a payload of 1448 bytes.

Packet #7

The sequence number of the client has been increased to 726 because of the last packet it sent. Having received 1448 bytes of data from the server, the client increases its acknowledgement number from 1 to 1449.

For the majority of the capture, we will see this cycle repeat. The client's sequence number will remain steady at 726, because it has no data to transmit beyond the initial 725 byte request. The server's sequence number, in contrast, continues to grow as it sends more segments of the HTTP response.

Tear-down

Packet #38

After acknowledging the last segment of data from the server, the client processes the HTTP response as a whole and decides no further communication is needed. Packet #38 is sent by the client with the FIN flag set. Its acknowledgement number remains the same as in the prior packet (#37).

Packet #39

The server acknowledges the client's desire to terminate the connection by increasing the acknowledgement number by one (similar to what was done in packet #2 to acknowledge the SYN flag) and setting the FIN flag as well.

Packet #40

The client sends its final sequence number of 727, and acknowledges the server's FIN packet by incrementing the acknowledgement number by 1 to 22952.

At this point, both hosts have terminated the session and can release the software resources dedicated to its maintenance.

Posted in Packet Analysis

Comments

Cd-MaN
June 7, 2010 at 4:39 a.m. UTC

Fun fact: the 3-way hanshake can be in fact a four way handshake - see here (although I don't believe that it is implemented like that anywhere):

http://www.breakingpointsystems.com/community/blog/tcp-portals-the-handshakes-a-lie/

eaadams
June 7, 2010 at 8:38 a.m. UTC

Great stuff Jeremy! Excellent explanation of TCP operation PLUS how to use Wireshark features to examine and learn about how a protocol works. Expect a flood of hits from my students!

Aubrey

Mimo
June 7, 2010 at 3:27 p.m. UTC

Hi Jeremy,

Great article.

When I studied TCP first, I was thought that ack no. is the next byte receiver is expecting. i.e if ack no. is 726 that means, receiver received 725 bytes of data and sender can send data starting from 726th byte.

So to me this sentence doesn't make much sense:

"Notice that the acknowledgment number has been increased by 1 although no payload data has yet been sent by the client. This is because the presence of the SYN or FIN flag in a received packet triggers an increase of 1 in the sequence."

As it seems to me that is is expecting data stream starting from byte 1. So Ack no. is not increased.

Though the meaning is same.

PS: Thank you for the demo of ability of wireshark. I still don't know much abt this tool.

---

stretch
June 7, 2010 at 5:02 p.m. UTC

@Mimo: In this instance, as is typical with network protocols in general, our counter is *zero-indexed*; the first byte is byte 0, not byte 1. the SYN flag increases the counter by one to 1, which is actually the second possible value.

Hope that helps.

---

yohan900
June 8, 2010 at 12:13 a.m. UTC

Mimo, you are correct. Do a search on google for TCP phantom byte and that will explain why there is an increment of 1 in the setup and teardown processes.

Also check out Laura Chappells WireShark training, good stuff.

http://support.novell.com/techcenter/articles/nc2001_05e.html

Note. During the TCP startup and teardown sequence, a "phantom byte" causes the sequence number and acknowledgment number fields to increment by 1 even though no data is exchanged. This phantom byte can be confusing when you have just learned that the sequence number field increments only when data is sent.

---

tormentum
June 8, 2010 at 12:51 a.m. UTC

G'day,

Great post! I've just recently started down the road to CCNA having always been interested in networking. I've used wireshark in the past for very basic troubleshooting, but this is a nice deep (comparitivley speaking!) look at TCP.

One question: packet 36 contains a PSH or push flag. What is this in relation to the HTTP stream?

Cheers,
Adam.

Mimo
June 8, 2010 at 3:00 p.m. UTC

Thank you yohan900 and stretch for the information!!

@ tormentum
http://support.microsoft.com/kb/169292
If the PUSH bit set, data will be delivered up to the application right away, but the ack may still be delayed.

Hope this helps....

Daren
June 9, 2010 at 7:49 a.m. UTC

What a bit of luck!!! I have a second-stage telephone interview today and one of the subjects I will be tested on is TCP. The first interview covered it in a little detail (3-way handshake / sliding window) but this one today will delve further into it. This will help enormously - thank you very much Jeremy.

Ned
June 9, 2010 at 7:01 p.m. UTC

Hi Jeremy, Nice Article. Thx for posting. If you get a chance can you continue on in this series and dive deeper into TCP with regards to the various algorithms like slow start and other concepts like windowing, congestion control, PAWS, Reno, SACKs etc. Tx

AaronDhiman
June 9, 2010 at 7:21 p.m. UTC

Awesome Article!

Dano
June 10, 2010 at 6:18 p.m. UTC

It would have been much better to call this a "Three-step handshake" or "Three-phase handshake". To me, "Three-way handshake" implies that here are three different parties shaking hands when in fact it is a method that requires three steps for two parties to communicate.

Just an observation . .

ecbanks
June 17, 2010 at 1:38 p.m. UTC

Succinct explanation. Helped straighten this out in my head.

raylook
July 1, 2010 at 3:21 p.m. UTC

great article , thanks ;)

Angelo
November 20, 2010 at 6:39 p.m. UTC

Hi, So if client is sending a data over different TCP segments the acknowledge number on client side will not increase untill all the data has been sent over..???

I have a client which sends data to server and when I capture it I see the data is not in single IP datagram but I see that the whole data sent to server spans over multiple TCP segments and in Two IP datagrams.

My questions is if I need to collect all the data sent by client to server how do I reassemble them at TCP lever. I am capturing IP packets which encapsulates TCP....

Thanks,
Angelo

natraj
December 9, 2010 at 12:34 p.m. UTC

Hi,
This is good stuff, but why didn't u talk about packet#36 and packet#37? If those are also added to this article, then this will be more useful for beginners.

Thanks,

Natraj.

Wajih
May 30, 2011 at 5:05 p.m. UTC

Thanks indeed. God bless you!

Taki
June 4, 2011 at 6:19 p.m. UTC

Good tutorial, many thanks for it.

Why does wireshark keep displaying len=0 for each tcp segment ?

Himanshu
June 8, 2011 at 8:10 p.m. UTC

Itz really a nice work .
Good help for beginners.

matti
July 29, 2011 at 8:20 a.m. UTC

A good explanation on TCP basics. Your decision to use relative numbers is clear, but I always tend to stick with actual numbers in the 3-way handshake part. This is to emphasize my students that both ends use their own, independent seq numbers that gets increased while data flow goes on. It's easy to mix them when both start from zero, as is the case with wireshark using relative seq numbers.

Keep up the good work!

Vijay
October 11, 2011 at 5:05 p.m. UTC

Great Explanation !!! It cleared most of the doubts that I had.

Earlier my assumption was the sequence number would be sequential and I did not know that Acknowledgement was related to the amount of bytes transferred. But you made it clear. Thanks a lot.

---

mkriesten
October 27, 2011 at 11:18 a.m. UTC

Excellent explanation. This article is a wonderful piece for educational purposes. Stumbled upon it yesterday while talking about TCP with one of my younger colleagues.

---

jrreyes
November 29, 2011 at 11:37 p.m. UTC

Very clearly explained. Thank you !!!

---

leffe
December 9, 2011 at 9:06 p.m. UTC

Thank you for this great explanation!

A guest
December 20, 2011 at 11:20 a.m. UTC

Very good explained. Thanks a lot. Guys like you makes life easier for us.

Amit Gupta
January 27, 2012 at 4:27 a.m. UTC

Excellent explanation. I was just hunting for this type of explanation on the complete cycle of TCP packet. Got more interest after reading this article.

Thank you very much Jeremy.

---

kr105
March 24, 2012 at 8:12 p.m. UTC

Awesome, thank you!!

---

mani
April 27, 2012 at 8:00 a.m. UTC

its awesome man its great effort i really appreciate it good work it is very helpful for beginners

---

Zeo
May 16, 2012 at 5:31 p.m. UTC

Thank you Jeremy it is a great article!

Looking at the connection termination:

"The server acknowledges the client's desire to terminate the connection by increasing the acknowledgement number by one (similar to what was done in packet #2 to acknowledge the SYN flag) and setting the FIN flag as well."

Please share your thoughts on this because I thought that the ACK and FIN can not be combined in the same segment because the the server might still be processing a transaction and not ready to close yet so it waits until it finishes processing those transactions before setting FIN flag. Thus it is a way hand shake when tearing down a connection.

Joe
May 21, 2012 at 7:02 p.m. UTC

Wow, I've been looking for a simple explanation like this for the last week, thank you so much!!!!

Mitchell
May 23, 2012 at 3:17 p.m. UTC

Hi, thanks for this great tutorial, have an exam in a few days and this had made things a lot clearer although realy, I didn't need to go this much is depth.

Thank you.

James Goulding
May 29, 2012 at 12:01 a.m. UTC

After a zillion sites I finally understand the concept of ACK numbers. Your step by step break down of the diagram was key for me. Note: I'm going to a very well-known University and none of their material explains this concept like you have done.

The following was what did it for me:
"Packet #: 7 The sequence number of the client has been increased to 726 because of the last packet it sent. Having received 1448 bytes of data from the server, the client increases its acknowledgement number from 1 to 1449.

For the majority of the capture, we will see this cycle repeat. The client's sequence number will remain steady at 726, because it has no data to transmit beyond the initial 725 byte request. The server's sequence number, in contrast, continues to grow as it sends more segments of the HTTP response."

I realized that the server could send more than the client requested. We were asked to break down 8 packets into a diagram and explain their movements, in Wireshark, with several criteria. In ALL of their examples the server NEVER sent more data. After 3 hours I was ready to blow.

Thanks for your posting.

Student
July 2, 2012 at 1:41 a.m. UTC

Thank you for the explanation and visuals on WireShark! Made understanding TCP/IP a lot easier!

Heshan
July 20, 2012 at 10:51 a.m. UTC

Thank you very much! A great explanation.

A guest
July 30, 2012 at 1:58 p.m. UTC

Thanks, It was very useful for me.

Novin Mathew
August 3, 2012 at 1:17 p.m. UTC

Jeremy, Good one! A quick insight into few important TCP/IP internals. Really helpful.
Cheers, Novin

Ondrej
August 26, 2012 at 11:09 a.m. UTC

Amazing, thank you so much. I've been looking for explanation like this for ages. Cool stuff, keep up on your great work!

Suryakanta Mandal
August 27, 2012 at 8:53 a.m. UTC

Great stuff Explained very well

Mohsen
September 17, 2012 at 3:23 p.m. UTC

Thanks, this gives an intuitive understanding of TCP/IP.

Thanks
Regards
Mohsen hs

sam
September 17, 2012 at 4:24 p.m. UTC

Wonderful information, Thanks

SteveO
September 19, 2012 at 1:51 a.m. UTC

Difficult to find a document like this. Cisco needs to hire you to write this in their books! This will always help me and others.

Rohit
October 11, 2012 at 10:44 a.m. UTC

Thank you very much....

jatin
October 14, 2012 at 12:00 p.m. UTC

Thank you so much for providing the knowledge about TCP. this is really helpful and much more deep than i expected.

cheers
jatin.

yared
November 2, 2012 at 2:49 p.m. UTC

Thank you very much about Jeremy for your detailed explanation .you make me fell confident for my mid term exam

Msurni
November 5, 2012 at 8:21 a.m. UTC

Very nice article....
How do we indicate that this is the last chunk in the streaming? do we need to send rn in the last data chunk or how?

Koop
November 6, 2012 at 1:08 p.m. UTC

I learned that ACK number was "bytes sent + 1", but here we have ACK number = "bytes sent". (Indeed the three way handshake show seq = 1 (byte))

BigB
December 10, 2012 at 4:40 p.m. UTC

Explained very clearly and thoroughly, thanks!

Balaji
December 11, 2012 at 4:11 a.m. UTC

Thanks :) It was really helpful

shady
January 28, 2013 at 11:52 a.m. UTC

Thanks, this helped.

hatcreek
February 23, 2013 at 8:50 p.m. UTC

This made it much clearer for me, thank you. How's the Lab coming? Hope your having a good 2013.

-Hatcreek

Vince
February 24, 2013 at 7:30 p.m. UTC

It's the first post / article I've read about the subject that is clear and understandable.

Thanks

Askar
March 12, 2013 at 10:24 a.m. UTC

Thank you very nice and informative, keep the good work.

Vitor
March 13, 2013 at 5:38 p.m. UTC

Anothre great article, Jeremy! Realy thnks for that. U can explain in plain english 'complicated' (or not so understand things)

sameer
March 25, 2013 at 10:03 a.m. UTC

Thanks for the detailed article.

Zaygham
March 27, 2013 at 2:41 p.m. UTC

What is the rational for having a random assignment of sequence number at the start of a TCP session. It should always start with zero

Christopher
March 29, 2013 at 3:10 p.m. UTC

Your explanation is very helpful!Thanks a lot!

laud
April 3, 2013 at 12:09 p.m. UTC

Thank you, this is really a great article. It definitely halped preparing my Network exam :)

golha
April 23, 2013 at 5:32 a.m. UTC

Very helpful! tnx.

Kishore
May 3, 2013 at 2:56 p.m. UTC

Great article indeed Jeremy. Good explanation of TCP negotiations. I know about the TCP handshakes but not to such extent. Thanks for the wonderful explanation.

nocomment
May 17, 2013 at 11:35 a.m. UTC

excellent !!

Thyag
July 25, 2013 at 8:25 p.m. UTC

Jeremy,

You simply rock dude lml !

ROHIT
July 27, 2013 at 10:21 a.m. UTC

Nice

Haridas N
August 8, 2013 at 7:11 a.m. UTC

Hi,

Your blog entries are very clear and clears the basic concepts going under these complex protocols. Thanks for the good explanation, this helped me to concrete my idea about how the TCP communication works.

Warm Regards,
Haridas N.

Guest
August 25, 2013 at 9:33 p.m. UTC

Hi, thanks a lot, nice post.

BTW Professor Philip Levis in the "Introduction to Networking" Fall 2012 Stanford Online course explains (Week 3, Tcp Header lecture) that the TCP ack number is the last received byte PLUS ONE, that is refers to the next byte that one wants to accept, not the last byte received.

Kind of helps to grasp the counts.

Keep up the good work,
Guest

Sidd
October 30, 2013 at 5:33 p.m. UTC

Excellent post and a good challenge question :-) to post the comment... thanks

---

Satish
November 22, 2013 at 2:29 p.m. UTC

Superb! Really nice post with a very clear explanation. Thanks for posting such an nice article.

Jesper Gustafsson
December 18, 2013 at 2:41 p.m. UTC

Oh good lord this helped me! Cisco surely didn't do much help on this matter!

Dan B
January 3, 2014 at 8:23 p.m. UTC

FYI, the use of an apparently random number for the initial sequence number is not a bug in Wireshark and is a very intentional implementation detail.

See http://tools.ietf.org/html/rfc1948 but, in brief, making the sequence number less predictable helps to prevent certain types of malicious attacks.

See also the wiki page on TCP:
http://en.wikipedia.org/wiki/Transmission_Control_Protocol#Reliable_transmission

"In the first two steps of the 3-way handshake, both computers exchange an initial sequence number (ISN). This number can be arbitrary, and should in fact be unpredictable to defend against TCP sequence prediction attacks."

HP1
January 10, 2014 at 10:41 p.m. UTC

Awesome stuff like always!

---

kaps16
February 3, 2014 at 12:33 p.m. UTC

I didn't understand this under packet #3-

At this point, the sequence number for both hosts is 1. This initial increment of 1 on both hosts' sequence numbers occurs during the establishment of all TCP sessions.

I guess the SEQ # of Server is still 0.

I know that in packet # 3 (from client) ACK = 1 still how does it increment the SEQ # at the server side?

Andrew
February 19, 2014 at 3:02 p.m. UTC

Great article!

What you write about SEQ N and ACK N and how they are incremented is not written in any TCP article I read, but that's exactly how "real" TCP works!

Thank you so much!!!

jenil
March 3, 2014 at 1:56 a.m. UTC

superb article!

its clear your all doubt and also provides you basic ideas. Thankx man

rj
March 19, 2014 at 7:45 p.m. UTC

great article & superb explanation.
Please help me in understand this
- how did the client know that it is the last segment from server?
- is it client which always initiates the termination of connection?

A guest
March 20, 2014 at 9:20 a.m. UTC

Thank you sir for such a great article. No words to explain

Johnny
March 31, 2014 at 11:12 a.m. UTC

You know that moment when something just all of a sudden makes complete sense? Yeah, reading this article gave me that moment!

Thank you!

Kishore
May 21, 2014 at 4:44 p.m. UTC

Great write-up!

Kanishk
May 23, 2014 at 6:36 a.m. UTC

Great help!!! :) Thanks man....

09127
May 28, 2014 at 6:38 p.m. UTC

Thank you a lot for this guide: it was really useful both to learn more about how TCP works and about Wireshark.

Francisco Zanatta
June 12, 2014 at 5:37 p.m. UTC

Very useful article, specially because it uses Wireshark screencaptures and shows the real thing happening. Very useful.

Sumsam Ali
June 22, 2014 at 9:19 a.m. UTC

Great explanation.

naveen
June 24, 2014 at 4:34 p.m. UTC

Loved the explanation from Author. This is what i was searching for. Nice work Jeremy. Keep sharing. :)

Maks
July 2, 2014 at 8:52 a.m. UTC

Thanks a lot , now it really become clear

Noel
July 5, 2014 at 2:44 a.m. UTC

Thanks for the useful information, I've just started reading it but it already answered some of the question marks on my head. I want to thank you for your easy to understand approach about this complicated world of TCP/IP. Kudos to the team. thanks

Diego
July 6, 2014 at 4:53 p.m. UTC

Thanks a lot for taking the time to do this !! It really help catching up really fast :)

blacklotus
August 3, 2014 at 10:12 p.m. UTC

Thanks a ton - super helpful.

Mudit
October 1, 2014 at 9:50 a.m. UTC

Great Article. Thanks :)

Dave
October 3, 2014 at 9:47 p.m. UTC

Nice stuff!

Can someone explain why in the packet #4 (HTTP GET) the sender has the ACK flag bit ON. Is that mandatory? I have noticed the same for the packet #38. I have been trying to find a reference on how mandatory the ACK flag is but can't find.

Harpreet Singh
October 16, 2014 at 11:56 a.m. UTC

Essential and complete. Thanks!

NN
October 27, 2014 at 10:20 a.m. UTC

Thank you Jeremy for this blog, it's value won't be deprecated as long as TCP protocol is used.

Doubt: You mentioned that if the SYN/FIN bit is set in the received packet, the outgoing Ack no will be +1. I don't understand the statement in tear down process "acknowledges the server's FIN packet by incrementing the acknowledgement number by 1 to 22952" why is there a random increment of the last packet? Or Did i miss something?

Phil Kim
December 3, 2014 at 1:37 p.m. UTC

Is it possible to make an application to send packets with same sequence number for just health checking?

Sofía
December 3, 2014 at 8:32 p.m. UTC

Thank you so muuuuuuuch!

AlexTee
December 5, 2014 at 4:59 p.m. UTC

Thanks for another great article.

I'd like to note, that some Cisco books explain Ack number a bit differently. In particular, Doyle's Routing TCP/IP states following:

"Acknowledgment Number is a 32-bit field that identifies the sequence number the source next expects to receive from the destination. If a host receives an acknowledgment number that does not match the next sequence number it intends to send (or has sent), it knows that packets have been lost."

Miljana
January 30, 2015 at 2:31 a.m. UTC

Thank you Jeremy, this is really fantastic explanation. :)

tacy
March 19, 2015 at 3:49 a.m. UTC

Awesome, thanks Jeremy

Harsha
March 27, 2015 at 10:22 a.m. UTC

fantastic explanation!

chethan
May 20, 2015 at 2:03 a.m. UTC

Very informative.

Alfredo
May 31, 2015 at 10:23 a.m. UTC

Thank you! Excellent explanation!

Viet
June 5, 2015 at 6:32 a.m. UTC

Good stuff, refresh my knowledge

kooth
June 10, 2015 at 2:46 p.m. UTC

This was quite helpful! I'm a developer and thought I knew all I needed to know about packet analysis -- boy was I wrong! After reading this post, I immediately signed up!

Thanks so much for sharing your knowledge!

Sincerely, Keith E. Cooper Tampa, FL.

Allan
July 4, 2015 at 11:46 a.m. UTC

Hi, Is there any way I can compare tcp packet payload with the memory. Please help. Thank.

A guest
July 17, 2015 at 11:09 a.m. UTC

Why thee HTTP Response is shown as TCP and not HTTP on Wireshark? Does anyone know?

Henrik
July 31, 2015 at 12:36 p.m. UTC

Kudos on the explanation from someone who is really new to the TCP protocol.

chaoticflanagan
August 28, 2015 at 5:03 a.m. UTC

Maybe i'm just missing something, but why does the sequence number change so much, fluctuating up and down?

Todd
September 1, 2015 at 9:54 p.m. UTC

Another explanation comes from Richard Stevens' TCP/IP Illustrated vol. 2: (slightly paraphrased):

Every byte of data exchanged across a TCP connection, along with the SYN and FIN flags, is assigned a seq. #. The seq # in a packet contains the number of the first byte in the seqment, which may be zero if the relative seq # starts at zero.

The ack # contains the next seq # the sender of the ack expects to receive, thus acknowledging all data up to the ack # minus 1. Thus, the ack # is the next seq # expected by the sender of the ack. The ack # is valid only if the ACK flag is set in the header.

I believe the inclusion of the SYN/FIN flags explains the "phantom byte" discussed elsewhere.

Pablo
September 6, 2015 at 6:22 p.m. UTC

Great work¡ I am trying to understand TCP in a better way, so thank you very much¡¡

Russoue
September 9, 2015 at 9:12 p.m. UTC

Great tutorial! Thanks.

Ali
September 13, 2015 at 2:45 p.m. UTC

Very helpful .Thank you

Markus
October 1, 2015 at 9:40 a.m. UTC

Excellent article! This really helped me refreshing my tcp knowledge for analysis!

If you need an idea for a new article, how about the SSL handshake explained in the flow chart? :)

Jason
October 23, 2015 at 5:45 a.m. UTC

@chaoticflanagan: The sequence numbers are not fluctuating; there are two sets of sequence numbers, one for each direction. The Server and the Client each hace their own sequence number count incrementing as they send data. You must observe the direction of the arrows in this flowgraph.

alma
October 30, 2015 at 8:42 p.m. UTC

Thank you sir, a very good tutorial

Nilesh
December 12, 2015 at 2:31 a.m. UTC

Great Explanation for TCP packet.

Bincy
December 25, 2015 at 4:55 a.m. UTC

Amazing article :)

Ming
January 20, 2016 at 2:47 a.m. UTC

Many thanks for the clear explanation of TCP stream. Make the TCP much less mysterious!!!

Giri
January 27, 2016 at 3:51 p.m. UTC

Excellent explanation! I am wiser now. Cleared all my doubts, and now I can finish my assignment. Thank You!

Devendra Gupta
February 6, 2016 at 1:24 p.m. UTC

Hi, Thanks for the post. I was wondering that in stop and wait the sequence number is one bit. Does this mean that out of 32 bit we only use one bit and rest 31 bits are useless. And we toggle this one bit sequence number from 0 to 1 and vice versa. If this is not true where does this one bit sequence number fits in the protocol itself.

Thanks for clearing my doubts in advance

Ajo
February 25, 2016 at 6:26 p.m. UTC

Thank you ! Well explained :)

Mayank
March 16, 2016 at 11:12 p.m. UTC

Great Job man..!! Thanks

jessicaParker
March 17, 2016 at 12:20 a.m. UTC

hey Stretch, great info on here, just wanted to see what you thought about analyzing tcp using netflow? according to this article http://www.pcwdld.com/what-is-netflow , netflow is an IP flow export protocol, but many folks are using it to manage and monitor network influx, usage and more? what are you thoughts, i'm considering downloading a netflow analyzer to test it out more, but just wondering if it would catch a TCP sequence as well? thx

---

Vinay
March 21, 2016 at 2:54 a.m. UTC

Does the server send 200OK only after sending all bits of the huge file requested by client? or Does the server send a 200OK as soon as it receives the GET from the client ?

Ruwan
May 3, 2016 at 8:51 a.m. UTC

Excellent, I learnt alot from this article.

Mritunjay
July 1, 2016 at 4:01 p.m. UTC

great Article..Thanks Man

DeanRasmus
August 17, 2016 at 1:19 p.m. UTC

What are the odds! By googling my questions, I ended up with reading two of your excellent articles. Thanks so much.

Tony
October 4, 2016 at 5:34 a.m. UTC

Excellent article. Loved how you step through one packet at a time.

EBrant
October 20, 2016 at 8:16 a.m. UTC

Thanks very much for posting this was an excellent explanation :)

Comments have closed for this article due to its age.