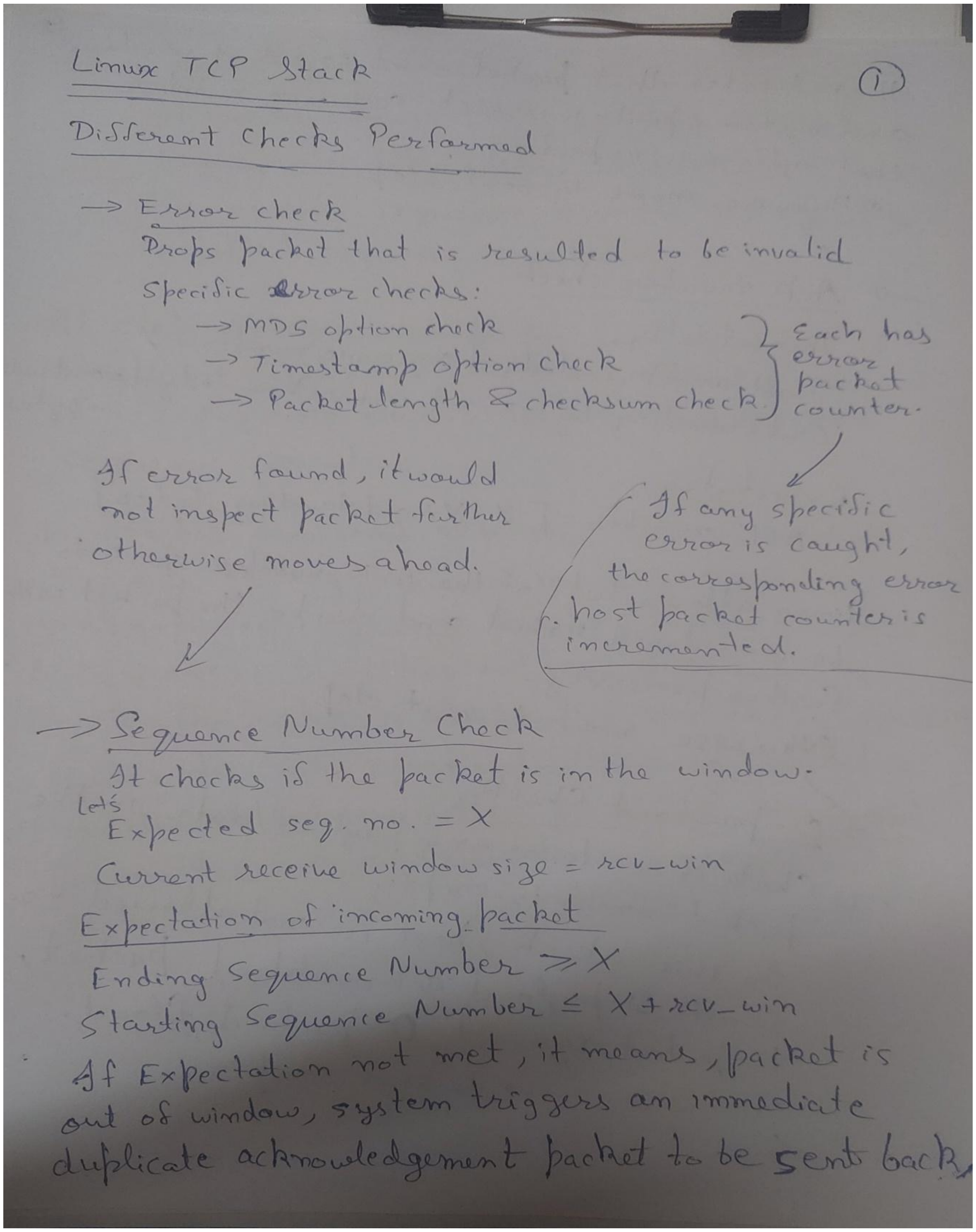
Q1: How are sequence no. generated for TCP in Linux, Windows, and MacOS?

Ans:

Linux



Linux TCP Stack

①

Different Checks Performed

→ Error check

Drops packet that is resulted to be invalid

Specific error checks:

→ MD5 option check

→ Timestamp option check

→ Packet length & checksum check

} Each has error packet counter.

If error found, it would not inspect packet further otherwise moves ahead.

If any specific error is caught, the corresponding error host packet counter is incremented.

→ Sequence Number Check

It checks if the packet is in the window.

Let's Expected seg. no. = X

Current receive window size = rcv_win

Expectation of incoming packet

Ending Sequence Number $\geq X$

Starting Sequence Number $\leq X + rcv\_win$

If Expectation not met, it means, packet is out of window, system triggers an immediate duplicate acknowledgement packet to be sent back.

It indicates that packet sent is out of window, and system specifies correct sequence number that system is expecting.
Otherwise move to next step:-
↙

→ Ack number check

Theoretically,
First unacknowledged sequence number = Y
Outstanding bytes, not yet acknowledged = oustanding-bytes

Expectation
ACK number = [Y, Y+outstanding_bytes]

Linux allows half of the ACK number space to be considered valid and drops the packet without further processing.
Otherwise move to next step:-
↙

→ 0-payload Check.
Check if contains any payload/data in the packet or it is empty.
In this check, host is not allowed to send 0-payload acknowledged packet, to protect against 0-payload flood.
Otherwise move to next step:-
↙

→ Retransmission Check.

It works almost same as Step-2 check, Sequence number check, It check if the ending sequence number not equal to expected sequence number.

At step-2, system sends tcp_sent_dupack() if condition is not met. By this any attacker can find if its sequence number is less than expected sequence number.

An attacker can send non-zero payload packet that has ending sequence number same as next expected sequence number with a guessed ACK number. If it does not pass ACK number check, the packet is dropped & the DelayedACKlost counter does not increment, otherwise, packet is considered retransmitted packet & increments the counter. In this way by using search algo's attacker can guess ACK number as well.

Linux is taking advantage of RFC1948. The hashing function implementation seems to be flawless, and additional randomness is introduced. Thus, the observed ISN generator characteristics are not related to any specific TCP connection parameters. ISNs are more easily predictable when using exactly the same source port, source address, destination port and destination address, but hashing function "secret" value is modified in relatively short time intervals (and thus frequently changing observed characteristics), do not exposing system security. The initial sequence numbers are intended to be more or less random. More precisely, RFC 793 specifies that the 32-bit counter be incremented by 1 in the low-order position about every 4 microseconds.

BSD/Mac OS

Sequence Number Based Counters

(1) → rcvduppack & rcvdupbyte
(2) → rcvpackafterwin & rcvbyte afterwin
(3) → rcvoopack & rcvoobyte

} ·used to infer server side sequence number

(4) → rcvdupack & rcvacktoomuch } infer client side sequence number

rcvduppack $\equiv$ Delayed ACK Lost

rcvduppack = no. of bytes (payload) in incoming packet with old sequence number considered duplicate.

(2) is equivalent to (1), but in (2) (rcvpackafterwin & rcvbyteafterwin), expected sequence number is x plus receive window size. Else they work same as (1).

(3) is incremented only when packet is out of order or incremented when sequence number is bigger than expected sequence number yet smaller than expected sequence number + receive window size.

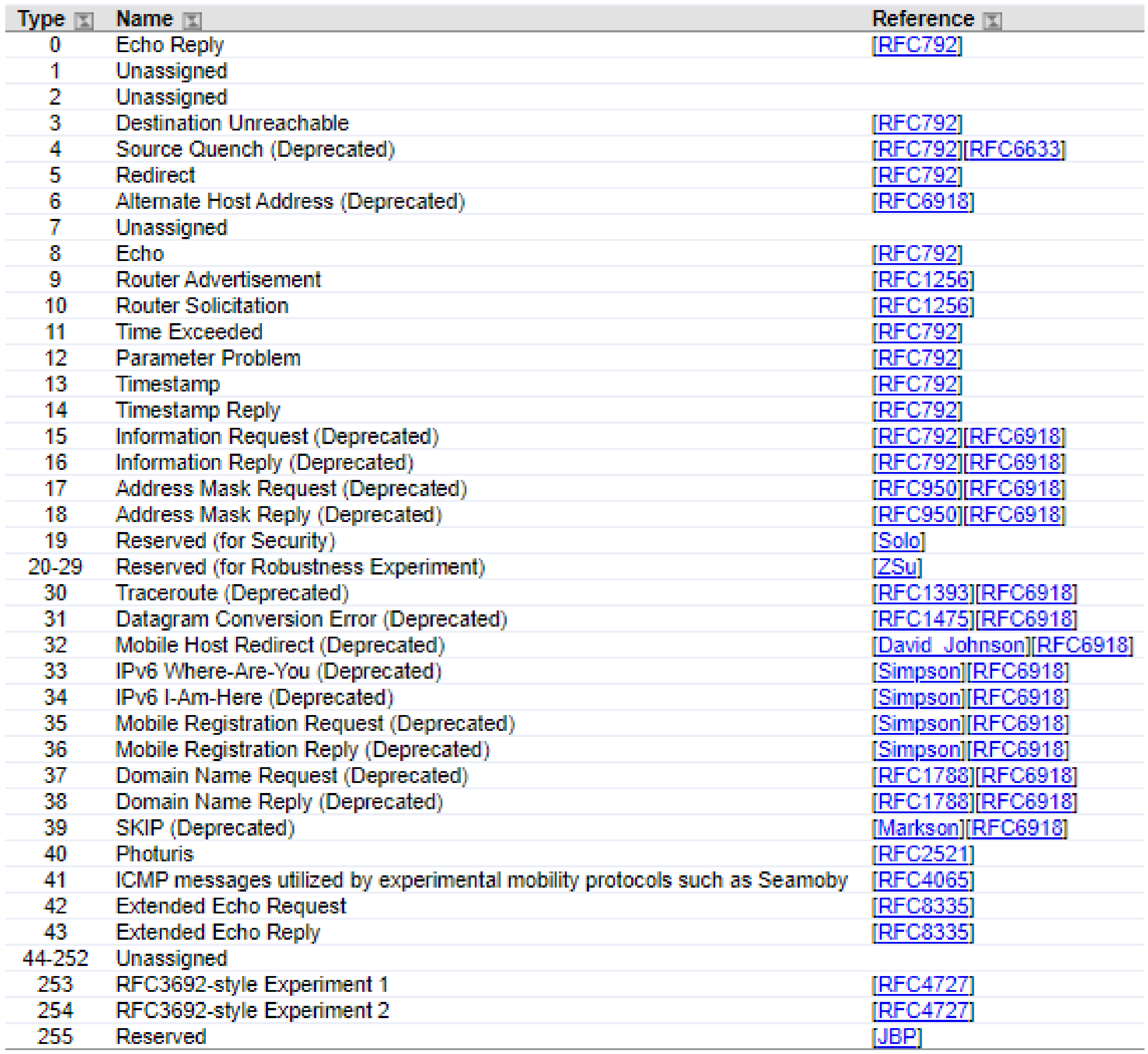
(4) rcvdupack is ~~increcre~~ incremented when (5)
ACK number of an incoming packet is smaller
than or equal to unacknowledged number.
(SND.UNA).

~~rcvdu~~
rcvacktoomuch is incremented when ACK
number is greater than the sequence number
of the next original transmit (SND.MAX)

Windows

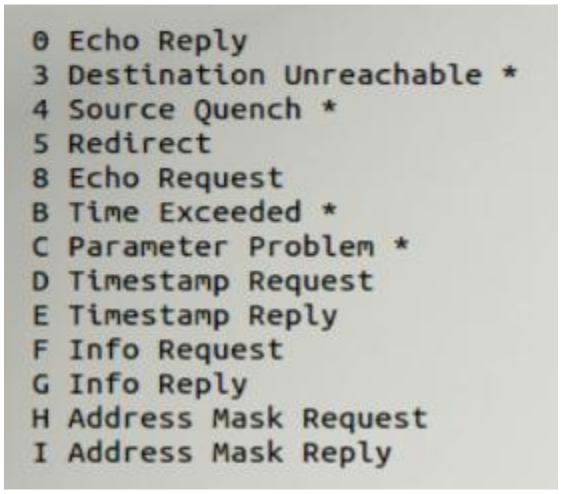
Microsoft Windows OSes do not appear to
expose such sequence-number-dependent counters
and thus safe.
From output of netstat -s
It shows only segments received, sent &
retransmitted.
Hard to infer sequence number from these,
quite impossible.

Q2: How many number of ICMP packets can be?
Ans:
Types of ICMP packets:



Type	Name	Reference
0	Echo Reply	[RFC792]
1	Unassigned	
2	Unassigned	
3	Destination Unreachable	[RFC792]
4	Source Quench (Deprecated)	[RFC792][RFC6633]
5	Redirect	[RFC792]
6	Alternate Host Address (Deprecated)	[RFC6918]
7	Unassigned	
8	Echo	[RFC792]
9	Router Advertisement	[RFC1256]
10	Router Solicitation	[RFC1256]
11	Time Exceeded	[RFC792]
12	Parameter Problem	[RFC792]
13	Timestamp	[RFC792]
14	Timestamp Reply	[RFC792]
15	Information Request (Deprecated)	[RFC792][RFC6918]
16	Information Reply (Deprecated)	[RFC792][RFC6918]
17	Address Mask Request (Deprecated)	[RFC950][RFC6918]
18	Address Mask Reply (Deprecated)	[RFC950][RFC6918]
19	Reserved (for Security)	[Solo]
20-29	Reserved (for Robustness Experiment)	[ZSu]
30	Traceroute (Deprecated)	[RFC1393][RFC6918]
31	Datagram Conversion Error (Deprecated)	[RFC1475][RFC6918]
32	Mobile Host Redirect (Deprecated)	[David_Johnson][RFC6918]
33	IPv6 Where-Are-You (Deprecated)	[Simpson][RFC6918]
34	IPv6 I-Am-Here (Deprecated)	[Simpson][RFC6918]
35	Mobile Registration Request (Deprecated)	[Simpson][RFC6918]
36	Mobile Registration Reply (Deprecated)	[Simpson][RFC6918]
37	Domain Name Request (Deprecated)	[RFC1788][RFC6918]
38	Domain Name Reply (Deprecated)	[RFC1788][RFC6918]
39	SKIP (Deprecated)	[Markson][RFC6918]
40	Photuris	[RFC2521]
41	ICMP messages utilized by experimental mobility protocols such as Seamoby	[RFC4065]
42	Extended Echo Request	[RFC8335]
43	Extended Echo Reply	[RFC8335]
44-252	Unassigned	
253	RFC3692-style Experiment 1	[RFC4727]
254	RFC3692-style Experiment 2	[RFC4727]
255	Reserved	[JBP]

Most common types of ICMP types as specified by
>man icmp
on SEED Ubuntu



```
0 Echo Reply
3 Destination Unreachable *
4 Source Quench *
5 Redirect
8 Echo Request
B Time Exceeded *
C Parameter Problem *
D Timestamp Request
E Timestamp Reply
F Info Request
G Info Reply
H Address Mask Request
I Address Mask Reply
```

Explanation of ICMP message types:

- **Echo Request, Echo Reply**
  Used to test destination accessibility and status. A host sends an **Echo Request** and listens for a corresponding **Echo Reply**. This is most commonly done using the ping command.
- **Destination Unreachable, Echo Reply**
  Sent by a router when it cannot deliver an IP datagram. A datagram is the unit of data, or packet, transmitted in a TCP/IP network.
- **Source Quench**
  Sent by a host or router if it is receiving data too quickly for it to handle. The message is a request that the source reduce its rate of datagram transmission.
- **Redirect Message**
  Sent by a router if it receives a datagram that should have been sent to a different router. The message contains the address to which the source should direct future datagrams. This is used to optimize the routing of network traffic.
- **Router Advertisement, Router Solicitation**
  Allow hosts to discover the existence of routers. Routers periodically broadcast their IP addresses via Router Advertisement messages. Hosts may also request a router address by broadcasting a Router Solicitation message to which a router replies with a Router Advertisement.
- **Time Exceeded**
  Sent by a router if the datagram has reached the maximum limit of routers through which it can travel.
- **Parameter Problem**
  Sent by a router if a problem occurs during the transmission of a datagram such that it cannot complete processing. One potential source of such a problem is invalid datagram header.

- **Timestamp Request, Timestamp Reply**
  Used to synchronize the clocks between hosts and to estimate transit time.
- **Information Request, Information Reply**
  Obsolete. These messages were used earlier by hosts to determine their inter-network addresses, but are now considered outdated and should not be used.
- **Address Mask Request, Address Mask Reply**
  Used to find the mask of the subnet (i.e. what address bits define the network). A host sends an Address Mask Request to a router and receives an Address Mask Reply in return.

References:

- https://www.rfc-editor.org/rfc/rfc1948.txt
- https://lcamtuf.coredump.cx/oldtcp/tcpseq.html
- https://www.iana.org/assignments/icmp-parameters/icmp-parameters.xhtml
- https://docs.sophos.com/esg/enterprise-console/5-5/help/en-us/esg/Enterprise-Console/concepts/Further_information_on_ICMP_traffic.html
- https://man7.org/linux/man-pages/man7/tcp.7.html
- https://docstore.mik.ua/orelly/linux/lnut/ch02_03.htm