Clock Driven Scheme to Initialize the Sequence Number

Using TCP to deal with delayed duplicates of both old and new connections

By Chong Kim

Introduction

I have done some work on an original paper by Sunshine entitled "Communication Management in Transport Protocols" in *Computer Networks* vol.2, 1978. He introduced a mechanism called "Clock Driven Scheme to initialize the sequence number". It is about initializing the sequence number of data packets when a transport layer in a host attempts to establish a connection with a transport layer in a remote host. I have discovered that the original mechanism introduced by Sunshine fails theoretically. I have developed my own theory or theorem in its place.

Anybody who has attended a seminar course in computer networks should have enough background knowledge to read this article. It will be better if the reader has knowledge about the design issues of transport layer protocols, particularly TCP. The Clock Driven Scheme to ISN here requires the understanding of the network of a datagram service such as Internets. For those readers whose memories need refreshing, I have included a discussion of transport layer protocols in the Appendix at the end of this article.

Clock Driven Scheme to Initialize the Sequence Number

A clock is running in the source/sender host. Note that the clock is running even when the host crashes. The clock is not necessarily global throughout the network. We also assume that the sequence number range is large in relation to the lifetime of a data packet in the subnet so that the data octets in packets of the current connection would arrive before their sequence numbers wrapped around or would have been killed off in the subnet by the time their sequence numbers wrapped around. That is, data octets carrying their sequence numbers are new data octets (i.e. not the data octets sent/emitted away before their sequence numbers wrapped around). We also assume that transport layers use sliding windows whose maximum sizes are very small in relation to the sequence number range, and the network service in question is connection-oriented.

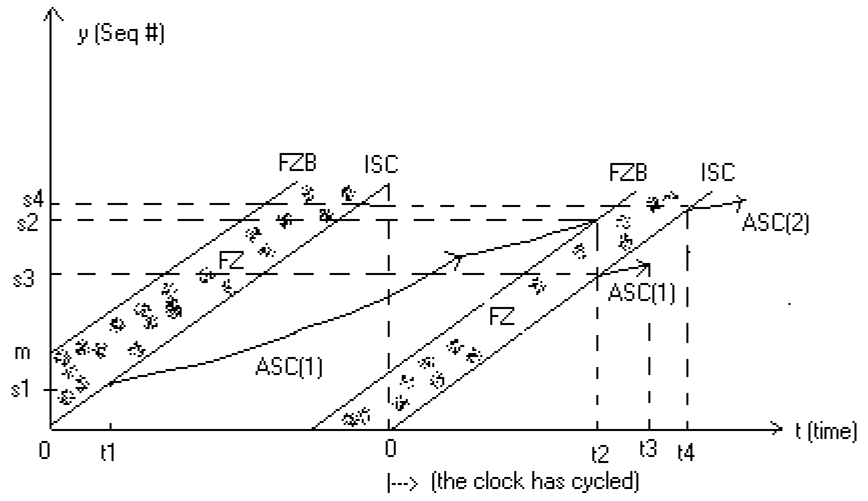
With the assumption of the large sequence number and the limited lifetime of a data packet in the subnet, this scheme is designed to handle Type 2 and Type 3 problems with delayed duplicates of old connections explicitly:

Type 2: Delayed data duplicates of an old connection arrive at the destination host and there is a new identical connection between the same hosts. If their sequence numbers are in the range of the receiver's window and ACKs are acceptable, they can be accepted in place of correct data octets to come later.

Type 3: A delayed duplicate of an old connection for connection-closure procedure arrives at the destination while a data communication is taking place on a new connection. If its sequence number is in the receiver's window, it can initiate a false connection-closure procedure of the current connection.

Note that Types 1 and 2 are really a single issue about delayed duplicates of the old connection arriving at the destination/receiver while there is a new connection between the two hosts.

The sending transport layer takes the following procedure:



FZB : Forbidden Zone Boundary where ASC is fobidden to cross
FZ : Fobidden Zone where ASC is forbidden to enter
ISC : Initial Sequence Number Curve which is the seq # curve of the maximum data transmission by the sender at time t
ASC : Actual Sequence Number Curve which is the seq # curve of the actual data transmission by the sender at time t

(1) When the transport layer wants to open a connection with another host at time t = t1, it checks its clock and selects its INS (Initial Sequence Number) = s1 from ISC at t = t1. Once the connection has been established (cf. the receiver does not need to know about the sender's clock except INS), the sender begins to emit data octets in packets and ASC shows the sequence number of data octet emitted at time t. Note that, throughout the data communication on the connection, ASC will remain below ISC until the clock cycles (because ISC reflects the maximum data emission rate by the sending host).

(2) Suppose that the clock cycles and the sender (the source transport layer) are still emitting data octets to the subnet. If ASC is just about to intersect FZB at t = t2 when the sender wants to emit the next data octets in a packet, the sender pauses to resynchronize the connection with the other end (i.e. the receiver) to inform that the sequence number of the new data octet in a packet starts at s3. ASC(2) is the new ASC after the resynchronization.

(3) If the sender (i.e. the source host of the data octet emission) is inactive for a period of time (but not so long to suggest a lack of activity on the connection because this could cause the other side to close the connection!) and ASC progresses horizontally to enter

FZ -- i.e. ASC enters FZ horizontally while the sender is waiting for ACKnowledgements -- then the sender must wait until ASC has exited from FZ horizontally.

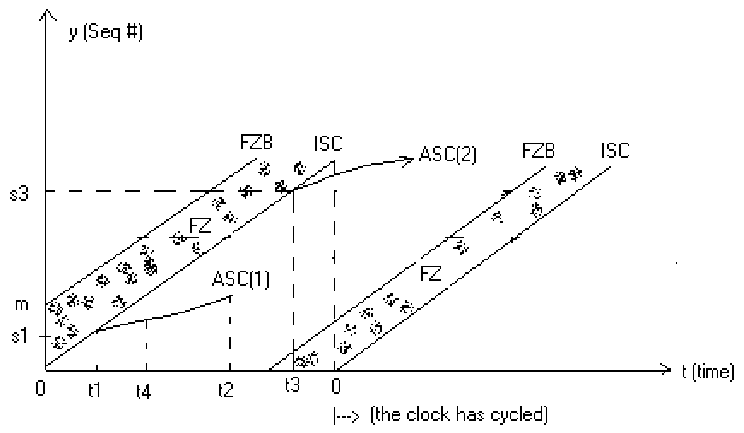
(4) In the diagram above, the connection closed at t = t3. If the sender opens the connection again at t = t4, it checks the clock and ISC at t = t4 and chooses INS = s4 for the new connection.

The idea is simple. If delayed duplicates of an old connection arrive at the destination, the receiver for the new connection discards them by determining that their sequence numbers are unacceptable. The value m is chosen to ensure that either

(1) The delayed duplicates of an old connection will have been killed off in the subnet by the time the receiving window of the destination transport layer for the new connection has advanced to cover their sequence numbers, or

(2) If delayed duplicates of an old connection arrive, their sequence numbers are outside/beyond the receiving window of the destination transport layer for the new connection. Therefore they will be discarded since their sequence numbers are outside the receiver's window.

Note that this scheme does not solve the problem type (1): a delayed duplicate of the old connection for connection request arrives at the destination and presently there is no identical connection open between the two hosts. As we will discuss shortly, this problem is dealt by "Three Ways Handshake".



Example 1

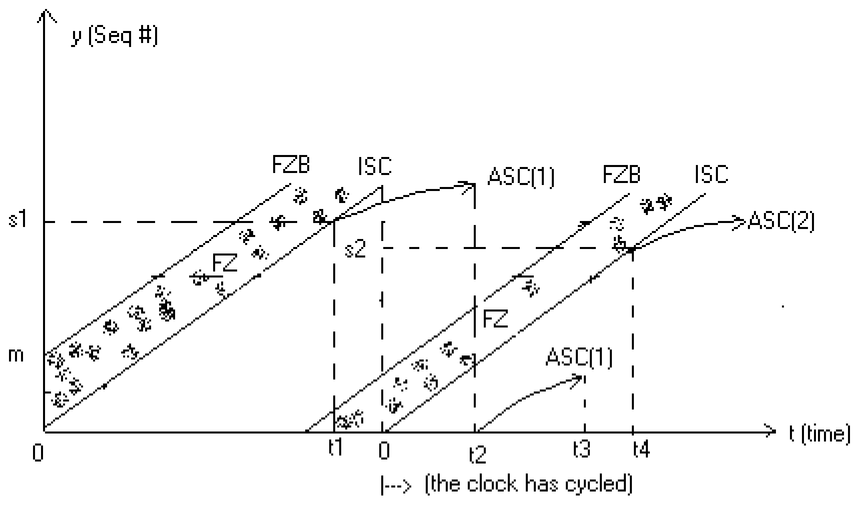
1) At t = t1, the connection opens and the sender emits data octets with ISN (Initial Sequence Number) = s1. ASC(1) is ASC for this connection.

2) At t = t2, the connection is closed.

3) At t = t3, the connection is opened again with ISN = s3. ASC(2) is ASC for the new connection.

4) Suppose that the data octet of the old connection emitted at t = t4 or its duplicate has wondered in the subnet for a period of time equal to or less than MSL (Maximum Segment Lifetime in the subnet). If this data octet of the old connection arrives at the destination/receiver (cf. the data octet's life time has not yet exceeded MSL), it will be discarded by the receiver because its sequence number is below ISN for the new connection and therefore is not covered/included in the advancing receiver's window.

Example 2



1) At time t = t1, the connection opens between two hosts and the sender (i.e. the owner of this clock) sets ISN (Initial Sequence Number) = s1 and starts emitting data octets. ASC(1) is ASC for this connection.

2) At t = t2, the sequence number wraps around for the sender and ASC(1) is below ISC.

3) At t = t3, the connection is closed successfully.

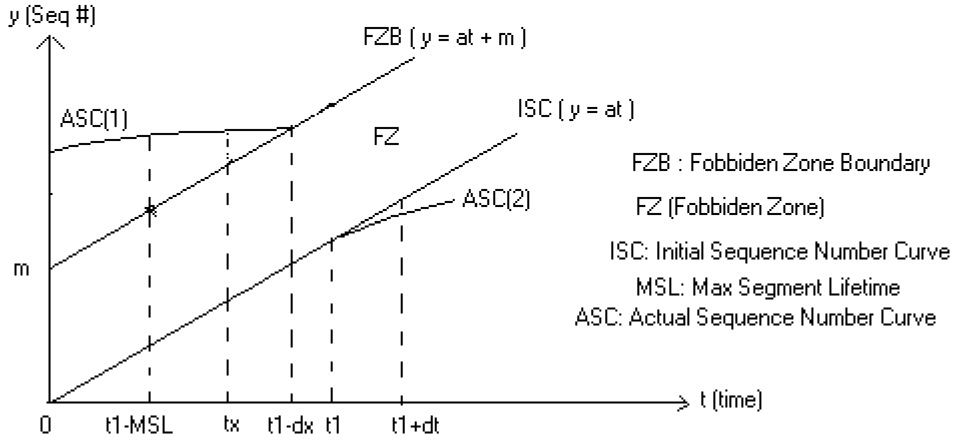
4) At t = t4, the same connection opens again with ISN = s2 for the new connection.

5) Any delayed duplicate data octet of the previous/old connection, if it arrives at the receiver before MSL in time has passed, has its sequence number below the lower boundary (or above the upper boundary) of the receiver's window for the new connection. Therefore, the receiver will discard the delayed duplicate data octet of the old connection.

How to Determine the m value

The value of m can be determined by considering the worst case scenarios where delayed duplicates of the old/previous connection arrive at the receiver of the new/present connection. We devote the rest of this article to the determination of the m value.

The evaluation of m – case (1)


y (Seq #)
FZB (y = at + m)
ASC(1)
ISC (y = at)
FZ
ASC(2)
m
FZB : Fobbiden Zone Boundary
FZ (Fobbiden Zone)
ISC: Initial Sequence Number Curve
MSL: Max Segment Lifetime
ASC: Actual Sequence Number Curve
t (time)
0
t1-MSL
tx
t1-dx
t1
t1+dt

0 < dx <= MSL (dx is at least 1). 0 <= dt <= MSL (Maximum Segment Lifetime).

The situation in the diagram above is that the old connection was closed at time t = t1 – dx and the new connection was opened at t = t1. ASC(1) belongs to the old connection and ASC(2) to the new connection. We do not consider any data octet, whether duplicate or not, emitted before t = t1 – MSL exclusive because such data will not exist in the network after t = t1 inclusive (i.e. after the new connection opened).

The possible minimum seq # of the data octet, whether duplicate or not, emitted at t = tx is above

y = a·tx + m – MSSW (Maximum Size of the Sender's Window) + 1 (inclusive) - (1)

where (t1 – MSL) <= tx <= (t1 –dx)

(i.e. Imagine that the seq # of ASC(1) at t = tx is the upper boundary of the sender's window. At t = tx, the seq # of any data octet, whether it is/was duplicate or not, that has been emitted is in this window of the sender. Any data octet whose seq # is below the lower boundary of this window of the sender has been acknowledged and no duplicate of it will be emitted after t = tx inclusive.)

Suppose that this data octet arrives at the receiver at t = t1 + dt, then we have

(t1 + dt) – tx <= MSL (cf. note that the situation for equality exists) - (2)

At t = t1 + dt, the upper boundary of the receiver's window is below

y = a (t1 + dt) + MSRW(Maximum Size of the Receiver's Window) (inclusive) - (3)

(Note that ASC(2) is below ISC inclusive and the sequence number curve for the received data octets for the receiver is below ASC(2) inclusive)

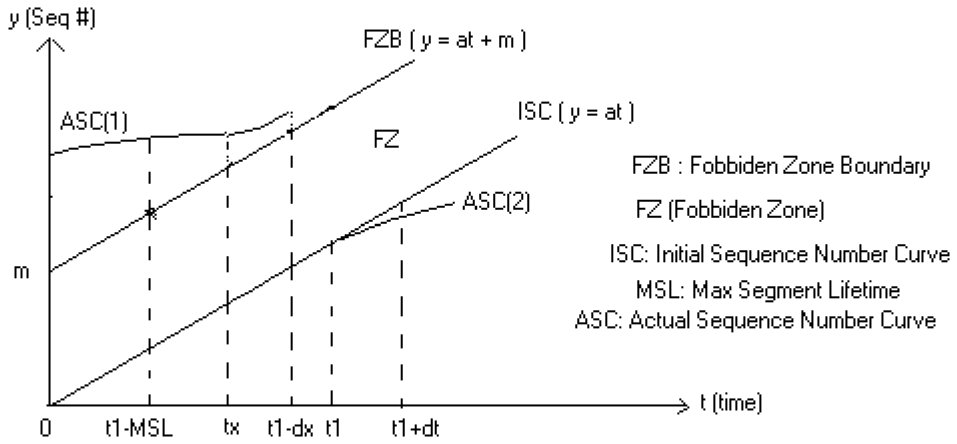
(1) – (3) > 0 means that, if the delayed data octet emitted at t = tx for the old connection arrives at the receiver of the new connection, the receiver for the new connection will discard this delayed data octet because its sequence number is above the upper boundary of the receiver's sliding window.

(1) – (3) = a·tx + m – MSSW + 1 – a·t1 – a·dt – MSRW
= – a (t1 – tx + dt) + m – MSSW + 1 – MSRW
(1) – (3) > 0 gives m > a (t1 – tx + dt) + MSSW + MSRW – 1
==> m >= a (t1 – tx + dt) + MSSW + MSRW since 1 is the unit for seq # - (4)
Then m >= a·MSL + MSSW + MSRW (using (2))
(i.e. y >= x + 3 and x <= 2 ==> y >= 2 + 3)

Therefore, the condition that m >= a MSL + MSRW + MSSW guarantees that in this worst case scenario the receiver of the new connection rejects any delayed data octet of the old connection, if it arrives, because the sequence number of the delayed data octet of the old connection is above the upper boundary of the receiver's window for the new connection.

Note that, even with MSRW and MSSW relatively small to MSL, we can still get (1) – (3) <= 0 if (t1 – tx) = MSL, dt = 0 and m = a·MSL for example. i.e. given m = a·MSL (i.e. the width of FZ = MSL) the seq # of the duplicate data octet emitted at t = tx is in the range of the receiver's window when it arrives at t = t1.

The Evaluation of m – case (2)

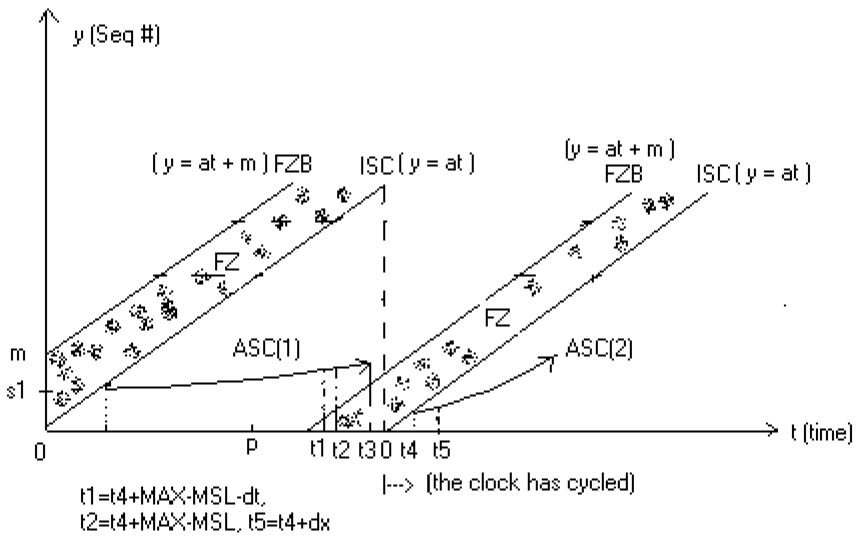

y (Seq #)
FZB (y = at + m)
ASC(1)
ISC (y = at)
FZ
FZB : Fobbiden Zone Boundary
ASC(2)
FZ (Fobbiden Zone)
ISC: Initial Sequence Number Curve
m
MSL: Max Segment Lifetime
ASC: Actual Sequence Number Curve
t (time)
0
t1-MSL
tx
t1-dx
t1
t1+dt

0 <= dx <= MSL. 0 <= dt <= MSL(Maximum Segment Lifetime).

The situation in the diagram above is that the old connection was closed at time t = t1 – dx and the new connection was opened at t = t1. ASC(1) belongs to the old connection and ASC(2) to the new connection. We do not consider any data octet, whether duplicate or not, emitted before t = t1 – MSL exclusive because such a data will not exist in the network after t = t1 inclusive (i.e. after the new connection opened).

Using the same argument and mathematics as in the case (1), we get the condition that m <= a·MSL + MSSW + MSRW for the receiver for the present/new connection rejecting any delayed data octet for the previous/old connection if it arrives at all.

The Evaluation of m – case (3)



dt is the distance between t = t3 and t =t4. 0 <= dx <= MSL. MSL >= dt > 0.
0 < (dt +dx) <= MSL. t4 – t1 = MSL.

At t = t3, the connection was successfully closed. The sender transport layer has opened the same connection with the same host again at t = t4. ASC(1) belongs to the old connection and ASC(2) to the new connection.

Clearly this is the same as the case (2). The only difference is that the ASCs of our concern have fallen down vertically and moved to the left horizontally. This is intuitively an identical situation to the case (2). So there is no need to be engaged in complicated mathematics. The condition that m >= a·MSL + MSRW + MSSW will ensure that any delayed duplicate of the previous/old connection is rejected by the receiver of the new connection if the delayed duplicate of the old connection arrives at all.

Note that all the other case scenarios where delayed duplicates of the old connection arrive at the receiver of the new connection are identical situations to the case scenarios that we have covered. If you zoom in to or out of the ASCs, ISC and FZB of our concern, you will realize that they are not different from the cases we have dealt so far. The difference will be that they have been moved around to the left or right or horizontally or vertically. Intuitively they are the same pictures as those in cases (1) and (2), which leads to the same conclusion.

Conclusion

We have covered all the worst case scenarios where delayed duplicates of the old connection arrive at the receiver of the new connection. They have led us to determine the m value. Our conclusion is that we condition that $m >= a \cdot MSL$(Maximum Segment Lifetime) + MSRW (Maximum Size of the Receiver's Window) + MSSW (Maximum Size of the Sender's Window) for the sender using the clock driven scheme. **T**his condition guarantees that any delayed data octet, whether duplicate or not, for the previous/old connection is discarded by the receiver for the present/new connection, if it arrives at all, because the sequence number of the delayed data octet is outside the range of the receiver's advancing window.

Note also that the m value can not be too large. The upper boundary of m (cf. I expect that it will be quite large since the ranges of time t and the sequence number are very large with relatively small MSL) should be set/limited to ensure that data octets for the current connection arrive before their sequence numbers wrap around or they have been killed off in the subnet by the time their sequence numbers wrap around. In simple words, it must be ensured that data octets carrying their sequence numbers are new data octets (i.e. not the data octets sent/emitted away before their sequence numbers wrapped around). This means that we limit the upper boundaries of MSRW and MSSW. In addition, it is also clear that all the hosts using this Clock Driven Scheme to communicate with each other have to agree on the single value for MSRW (i.e. We can not have one host using one value for its own MSRW and another using a different value for MSRW. In a word, the value for MSRW is universal among all networking software such as TCP using this scheme if they want to communicate with each other).

Note also that we have not taken into consideration a factor regarding the time taken by the data octet to travel from the sending host to the receiving host. It is non-deterministic. However, suppose that the possible minimum time taken by the data octet to arrive at the receiving host since its birth at the sending host is B. Then the equation (3) in case (1) becomes $y = a(t1 + dt – B) +MSRW$ (i.e. to the receiving host, ISC appears to have moved by B along t). Then we will have $m>= a(MSL – B) + MSRW + MSSW$. For the developer of the communication software, he can simply put B = zero since it will safely raise the lower boundary of m.

There is another factor we have not considered. We have assumed that MSL is the maximum life time of the data octet in the subnet such as the Internet. This means that, in the diagram in case (1), a delayed (duplicate) data octet emitted before $t = t1 – MSL$ may arrive at the receiving host. This will be possible if we consider the time taken by the data octet to arrive at the subnet since its birth (i.e. it may have to travel through its own local network before reaching the subnet). Therefore we conclude that, if we let C = the maximum time taken by the data octet to arrive at the subnet since its birth at the sending host, we have to consider every data octet emitted after time $t = t1 – MSL – C$ inclusive. So the equation (2) in case (1) becomes $(t1 + dt) – tx <= MSL + C$, which leads the equation (4) to become $m >= a(MSL + C) + MSSW + MSRW$ (or $m >= a(MSL – B + C)$

+ MSSW + MSRW if we also consider the time taken by the data octet to arrive at the receiving host since its birth at the sending host).

Finally (hopefully!), there is one more issue to cover. We have defined MSL as the maximum life time of a data octet or packet in the subnet (i.e. Internet). If you have a look at the IP header of a datagram which contains/carries the data octet in the subnet, you will notice TTL (Time To Live) bits. In the manual that I have, the length of TTL is said to be 8 bits long and each unit is a second. Therefore the maximum lifetime of the data octet in the subnet (i.e. the Internet) is 256 seconds. Unless all gateways and hosts are equipped with universally synchronized clocks and the datagram shows the time of the data octet's birth, theoretically this calculation seems unreliable because, when a gateway receives a data octet or packet from another, it just increases the TTL value in its header. We may ask how the receiving gateway computer knows how long it has taken the data octet to travel from the sending gateway to the receiving gateway. If this time is not calculated exactly and not incorporated into TTL, we may have to argue that there may occur a marginal error (i.e. a positive number) in the value of MSL of the subnet provider. If such a marginal error exists, we should be aware that a data octet may wander in the subnet (i.e. the Internet) for longer than MSL, by that marginal error, which the sending transport layer knows. If this is the case, we have to calculate that marginal error and increase MSL, by that marginal error, which the sending transport layer knows. In such a case, the sending transport layer protocol should use this updated MSL value. To summarize, if you suspect that a data octet may live in the subnet for longer than the MSL value given by the subnet provider, you may increase your MSL value for your sending transport layer by that marginal error (i.e. see in equations (2) & (4) in case (1). cf. I expect this marginal error to be a very small positive number.) However, I suspect that gateways have been designed to take into consideration this possible positive marginal error and increase TTL by time larger than or equal to the actual time taken by the data octet to travel between the two gateways. If this is the case, the MSL value of the subnet provider which the transport layer knows is larger than or equal to the true MSL value for the data octet in the subnet (i.e. a data octet's true maximum life time in the subnet is less than or equal to the MSL value of the subnet provider) and the transport layer does not have to update the MSL value provided by the subnet provider. i.e. the lower boundary of m using the MSL value of the subnet provider can be trusted for our purpose because such lower boundary of m is lager than or equal to that using the true MSL for the data octet in the subnet.

Summary

To solve the problems related to the delayed duplicates of both the old connection and the present connection, we use "Clock Driven Scheme to initialize the initial number" and "Three Ways Handshake."

Clock Driven Scheme with a large sequence number range relative to MSL (Maximum Segment Lifetime) resolves all the problems of delayed duplicates of both old and new connections, except that it can not handle the problem type (3): a delayed duplicate of an old connection for connection-request arrives to initialize a false connection and presently there is no such a connection between the two hosts. This problem type (3) is resolved by Three Ways Handshake.

TCP implements both Clock Driven Scheme and Three Ways Handshake to deal with delayed duplicates of both old and new connections.

Appendix

Transport layer protocols

If the subnet (i.e. Internet)'s packet delivery service is reliable, there is little for the transport layer to do to enhance the quality of the subnet's service. Most subnets are said to be reliable. The question is, how reliable should they be? For example, if a user wishes to send an electronic mail through a network that loses a packet per week on average, this network may well be regarded as very reliable and simple transport layer protocols may suffice to operate over this network. On the other hand, a bank that uses this network for multi-million pound transactions may regard this network as unreliable and insist on a heavy duty transport protocol to prevent any mistake.

Normally, several transport protocols work in parallel on top of a network layer and the user chooses which protocol to use for a particular user service. For example, in the TCP/IP protocol suit, TCP is used as a reliable transport protocol for a file transfer and remote login while UDP, a transaction-oriented protocol, is used for an electronic mail service.

If the subnet's packet delivery service is not reliable and the user does care about it, the transport layer protocol must be designed to deal with all possible problems of the subnet's packet delivery service explicitly. If the network in question is an Internet that consists of several different networks, the global reliability of the packet delivery service must be questioned and the transport protocol is designed to deal with problems of the subnet and to provide reliable end-to-end data transmission service.

We assume that the most unreliable data transmission service on a packet-switching network is a datagram service. Theoretically datagrams can be lost and damaged in the subnet. They can take different routes in the subnet and arrive out of order. Some may wander around in the subnet even after the two communicating hosts have closed down the connection. However, if we impose a constraint on the lifetime of a datagram in the subnet and allow for a large sequence number, this problem is manageable. If we ignore delayed duplicates of *old* connections for the time being, the data transmission seems to have few problems. For example,

> If the packet is lost or damaged, the sender will not receive ACKnowledgement and retransmit it when the packet's timer times out.
>
> If ACK from the receiver is lost, then the sender's sliding window stops advancing and the receiver may keep retransmitting ACK until it gets to the sender (in some implementation, if a transport protocol notices a lack of activity on the other side, the transport protocol aborts its side of the connection).

If datagrams arrive out of order, they will be reassembled according to their sequence numbers.

If delayed duplicates of the current connection arrives at the destination, they will be detected and discarded as duplicates because the small lifetime of the datagram in comparison with a large sequence number space will ensure that those duplicates will arrive -- if they do at all -- before their sequence numbers wrap around respectively - in other words, either delayed duplicates of the current connection have arrived at the receiver or been killed off by the subnet by the time their sequence numbers wrap around.

We now consider the problems with delayed duplicates of ***old*** connections. That is, delayed duplicates wander around in the subnet and arrive at the receiving end after the connection has been closed; Most computer communication is very short. Suppose that a data communication on a connection lasted for two minutes and some delayed duplicates of this connection are still wandering in the network, i.e. the life time of the packet is 5 minutes, for example. Suppose now that the connection has opened again. Then the communication on the same connection may suffer from the delayed duplicates of the old connection if this new communication opens within 5 minutes of the closure of the old connection. We may consider three types of problems with delayed duplicates of *old* connections:

1. If a delayed duplicate of an old connection for connection-request arrives at the destination after the connection has been closed, it could initiate a false connection procedure. (Cf. currently the connection is not in use.)

2. Delayed data duplicates of an old connection arrive at the destination and there is a new connection. If their sequence numbers are in the receiver's window and ACKs are acceptable, they can be accepted in place of correct packets to come later.

3. A delayed duplicate of an old connection for connection-closure arrives at the destination while a data communication is taking place on a new connection. If its sequence number is in the receiver's window, it can initiate a false connection-closure of the current connection.

To prevent delayed duplicates of old connections from disturbing the communication on the new connection, we assume that packets have MSL (Maximum Segment Lifetime), i.e. a lifetime of a datagram in the subnet. So any packet and its possible acknowledgement are long gone by the time MSL elapses from the birth of those packets -- either these packets have arrived at the receiving host or been killed off by the subnet before their sequence numbers wrap around.

<u>The first method</u> we may try is that both hosts wait for MSL before opening the same connection. This ensures that all delayed duplicates of old connections and their possible acknowledgements are drained from the subnet. Problems (1), (2) and (3) will not occur since the subnet is clear of any packet of old connections.

<u>The second method</u> was that the both hosts keep the information about the connection for MSL after the connection closure. If the host opens the connection within MSL from the

time of the closure, it chooses the initial sequence number one larger than the last sequence number used for the last connection. Problems (2) and (3) will not occur since sequence numbers of delayed duplicates of old connections would be below the receiver's window for the new connection. However, problem (1) still can happen -- i.e. suppose that the connection has been closed. No new connection has been requested from both sides. Then this delayed duplicate of the old connection for connection request arrives. The receiver may think that the other side wants to open the connection again.

The third method is "Clock Driven Scheme to initialise the sequence number"; A clock is running in the source/sender host and the initial sequence number is selected from the clock time. Note that this clock is not global throughout the network. The method was introduced by Tomlinson (1975) and its implementation prevents problems (2) and (3) with delayed duplicates of old connections. The advantage of this method over the other two is that a host does not have to wait for MSL or keep the record of the previous connection for MSL. To deal with the problem (1), Tomlinson introduced "Three Ways Handshake to open a connection". It can be suggested that "Clock driven scheme to ISN" and "Three Ways Handshake" based on a large sequence number space in comparison to the life time of a packet will resolve all these problems (1), (2) and (3) with delayed duplicates (of both old and new connections).

Chong Kim has worked as a computer programmer and researcher with a number of engineering and computer companies. His interests are operating systems, computer networks and artificial intelligence.