AR-006-797

ELECTRONICS RESEARCH LABORATORY

Communications Division

RESEARCH REPORT
ERL-0582-RR

ABOUT THE SIGNALING SYSTEM NO 7 SIGNALING CONNECTION CONTROL PART PROTOCOLS

by

M.K.F. Lai

SUMMARY

A description of the Signaling System No 7 Signaling Connection Control Part Protocol is given. A scenario where a connection can be released by an off-line not belonging to the connection is demonstrated. Finally, an improvement of the protocol for eliminating this particular flaw is proposed.

© COMMONWEALTH OF AUSTRALIA 1991

NOV 91 COPY No.

APPROVED FOR PUBLIC RELEASE

This work is Copyright. Apart from any fair dealing for the purpose of study, research, criticism or review, as permitted under the Copyright Act 1968, no part may be reproduced by any process without written permission. Copyright is the responsibility of the Director Publishing and Marketing, AGPS. Inquiries should be directed to the Manager, AGPS Press, Australian Government Publishing Service, GPO Box 84, Canberra ACT 2601.

Contents

1 The signaling Connection Control Part

The main function of the signaling connection control part (SCCP) of the Signaling System Number 7 [Q.700 88] is the catering for both connectionless as well as connection-oriented network services to transfer circuit related and non-circuit related signaling information and other type of information between exchanges and specialized centres (particularly for management and maintenance purposes) in the telecommunication networks via a Signaling System No. 7 (SS7) network. One of its overall objectives is to provide the means for establishing the logical signaling connections within the SS7 network and for transferring the signaling data units with the use of the logical signaling connections.

2 Connection-Oriented Connection

A SCCP user, in general, is any subsubsystem located at a signaling point of the network. A connection-oriented logical signaling connection established between the calling SCCP user and the called SCCP user in the network may consist of one or more connection sections in tandem. The originating node (a signaling point) of the connection is where the calling SCCP user is located, while the destination node (also a signaling point) is where the called SCCP user is located. The originating node of the connection is the originating node of the first connection section while the destination node of the connection is the destination node of the last connection section.

In the case of two or more connection sections, intermediate nodes (which are also a signaling point and are normally called signaling transfer points) are required for relaying messages. They are located between two connection sections. A message is received by an intermediate node from one connection section and is sent on to the other. In the setting up (establishment), passing data and releasing phases of a signaling connection, a protocol (a set of procedures) is followed by both the originating and destination node and also by the intermediate nodes involved.

A number of protocols are given in CCITT Q.714 [Q.700 88]. We are particularly interested in the connection-oriented protocol of Class 2 which is found in Section 3 of Q.714 [Q.700 88]. It is the lower connection-oriented protocol class. As far as the connection establishment and releasing procedures are concerned, it is the same as the higher protocol class except that no negotiation of protocol classes and initial credits is required.

2.1 Exchanged Messages

The basic SCCP messages being exchanged in the connection-oriented protocol can be classified into the following types: *Connection Request*, *Connection Confirm*, *Connection Refused*, *Data*, *Released* and *Release Complete*. Every message has a type code which uniquely identifies the type of the message. Their formats are given in Q.713 [Q.700 88].

3 Connection Request Messages

The main components of a *Connection Request* message include called party address and source local reference, both of which are mandatory, and also include an optional calling party address, Table 3/Q.713 of Q.713 [Q.700 88]. Although the calling party address is only optional, we feel that it is required in the connection-oriented connection establishment in order to allow a reply to the request to be sent back to the originating node.

3.1 Party Addresses

Party addresses are used to identify SCCP users in the network, and can be formatted in a number of ways (Q.713 [Q.700 88]). One possible format of a party address is a (signaling point code, subsystem number) pair. In this case, the signaling point code identifies the signaling point where the user is located and the subsystem number identifies the user at the signaling point. However, a party address can also be a global title and there are four possible formats.

In order to determine a signaling point code or to determine a pair of signaling point code and subsystem number from a global title, a translation is required. At every signaling point, a translation function is provided. However, the translations of the same global title performed at two different signaling points may be different [1]. Hence, it is important to note that a translation of global title of a SCCP user does not necessarily determine the signaling point where the user is located. The detail can be found in Section 2 of Q.714 [Q.700 88].

In the following, we shall assume that a signaling point code can always be determined by the translation function at every signaling point.

[1] For otherwise, intermediate nodes would not be required at all in any signaling connection.

3.2 Local Reference Numbers

Local reference numbers are assigned independently to a connection section by the signaling points at the two ends of the connection section. They are then used by the two ends to identify the connection section. As it will be described in the later sections, parts of a message are local reference numbers. Their functions are to prevent the initiation of incorrect procedures on a connection section by one of the end nodes due to receipt of a message (which may not even come from the other end of that connection section), Section 3.3 of Q.714 and Section 3.3.2 of Q.714 [Q.700 88].

The node at one end independently selects its source local reference number that will be used by the node at the other end as the destination local reference number, Section 3.1.2 of Q.714 [Q.700 88]. In doing so, it is the node's responsibility to ensure that the selected number has not been used for assigning in a reasonable number of connections immediately prior to the present connection.

As a result, every connection section is registered by the node at each end twice: with its selected source local reference number and with the destination local reference number. If the number for registering a connection section is selected by the node itself, then this number is the node's source local reference number for the connection section. On the other hand, if the number is not selected by the node itself, then it is the node's destination local reference number for the connection section.

Obviously, when the connection establishment is completed, the originating (respectively destination) node will have one pair of source/destination local reference numbers for the outgoing (respectively incoming) connection section. On the hand, every intermediate node will have two pair of source/destination local reference numbers catering for both the incoming and outgoing connection sections.

3.3 Sending Connection Request Messages

When a SCCP user at the originating node requests the establishment of a signaling connection to a remote SCCP user, it supplies the party address of the remote SCCP user (*i.e.* the called party address). Such details are provided by the calling SCCP user via the invocation of the **N-CONNECT REQUEST** primitive. For simplicity, we do not mention the determining of protocol class. The detail can be found in Section 3.1.4.1 of Q.714 [Q.700 88].

The originating node checks whether the resources are available. If they are not, the calling SCCP user is informed that the attempt to set up a signaling connection was unsuccessful via the **N-DISCONNECT INDICATION** primitive.

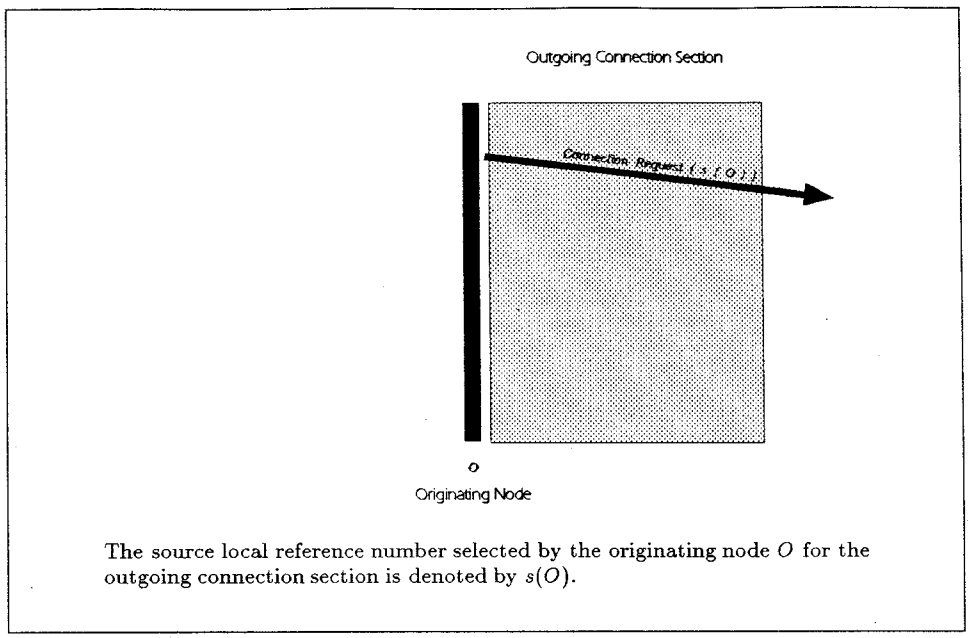


Figure 1: Sending *Connection Request* at an Originating Node

On the other hand, if the resources are available, the SCCP translation function at the originating node determines a signaling point code from the called party address. Notice that if there is a signaling point code given in the called party address, we can say that the translation function trivially determines the signaling point code (*i.e.* no translation is required).

After a number is selected and placed in the source local reference component and also after the calling/called SCCP user's party addresses are placed in the corresponding calling/called party address components, the *Connection Request* is sent to the signaling point whose point code is that just determined by the translation function. This is therefore the basic setting up of the outgoing connection section from the originating node to the remote node determined by the translation function. From then on, the outgoing connection section is registered with the selected source local reference number. Furthermore, the node is now expecting to receive either a *Connection Confirm* or a *Connection Refused* message on this registered outgoing connection section either before its connection establishment timer expires or before its calling SCCP user wishes otherwise. We remark that it is, generally, required that the availability of the determined signaling point should be checked before the sending of a *Connection Request* message. However, for simplicity, we shall assume that all the signaling points in the network are available.

3.4 Receiving Connection Request Messages at an Intermediate Node

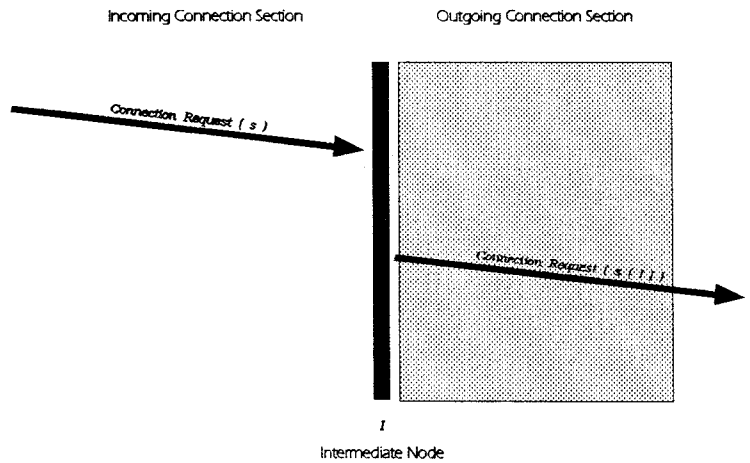
When a *Connection Request* message is received at an intermediate node, its SCCP translation function determines that the called party address in the message does not belong to a local SCCP user. Furthermore, in doing so, a new signaling point code is generated. Also, the node may decide that an association of connection sections is required. Again, for simplicity, we ignore the possibility that the association is not required, or that the protocol class should be considered.

The node then selects a number. To set up an outgoing connection section from this node to the one determined by the translation, a *Connection Request* message is constructed with the selected number placed in the source local reference component, and with the same calling and called party address components (*i.e.* the same addressing information) as found in the incoming *Connection Request* message. This *Connection Request* message is then sent out to the determined node. Full detail is contained in Section 3.1.5.1 of Q.714 [Q.700 88].

Additionally, the incoming connection is registered with the number found in the source local reference component of the received *Connection Request* message. The number becomes the destination local reference number for the incoming connection. In addition, the outgoing connection section is also registered with the selected source local reference number. Furthermore, the node is now expecting to receive either a *Connection Confirm* or a *Connection Refused* message on this registered outgoing connection section before the expiration of its connection establishment timer.

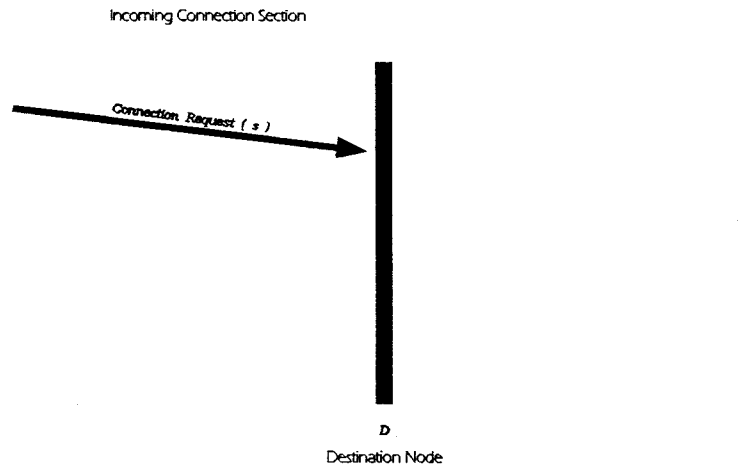
3.5 Receiving Connection Request Messages at the Destination Node

When a *Connection Request* message is received at the destination node, its SCCP translation function determines that the called party address in the message is a local SCCP user. Again, we assume that the resources are available, and that the determination of protocol class is not required. After the node informs the SCCP user of a request to establish a connection via a **N-CONNECT INDICATION** primitive, the user will invoke either a **N-CONNECT RESPONSE** or a **N-DISCONNECT REQUEST** primitive. In the mean time, the incoming connection section is registered with the number found in the source local reference component of the received *Connection Request* message so that the number is referred as the destination local reference number for the incoming connection. The detail is in Section 3.1.6.1 of Q.714 [Q.700 88].



The number found in the source local reference component of the incoming *Connection Request* message is denoted by s. The source local reference number selected by the intermediate node I for the outgoing connection section is denoted by $s(I)$.

Figure 2: Relaying Request of Connection at an Intermediate Node



The number found in the source local reference component of the incoming *Connection Request* message is denoted by s.

Figure 3: Receiving *Connection Request* at a Destination Node

4 Connection Confirm Messages

The important components of a *Connection Confirm* message include destination local reference and source local reference, which are both mandatory, and include the called party address, Table 4/Q.713 [Q.700 88].

4.1 Sending Connection Confirm Messages

A confirmation of connection can only be initiated at the destination node of a connection. It is only required when the SCCP user at the destination node has invoked the **N-CONNECT RESPONSE** primitive. Again the consideration of protocol class is ignored for simplicity. The destination point selects a number as its source local reference number for the incoming connection section. When constructing the *Connection Confirm* message,

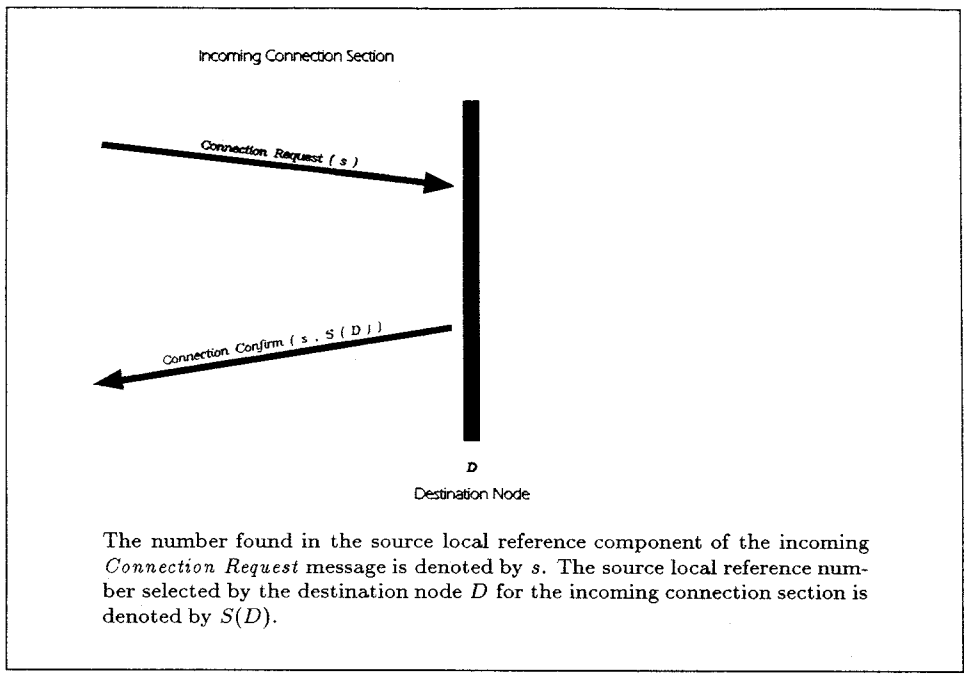
- the selected source local reference number is placed in the source local reference component, and
- the destination local reference number registered for the incoming connection section is placed in the destination local reference component, and
- the content in the calling party address component of the incoming *Connection Request* message is placed in the called party component.

Subsequently, the *Connection Confirm* message is ready to be sent on the incoming connection section. From then on, the node is expecting to receive either a *Data* message or a *Released* message on the registered incoming connection section before either the expiration of its inactivity timer or its called SCCP user wishes otherwise.

4.2 Receiving Connection Confirm Messages at Intermediate Nodes

When an intermediate node receives a *Connection Confirm* message before the expiration of its connection establishment timer on its registered outgoing connection section, it checks that the number in the destination local reference component of the received message matches its source local reference number for the outgoing connection section. If it does, the node records the number found in the source local reference component of the received message as its destination local reference number for the outgoing connection. Furthermore, another number is selected for assigning to the associated incoming connection section as its source local reference number for the section.

A *Connection Confirm* message with



The number found in the source local reference component of the incoming *Connection Request* message is denoted by s. The source local reference number selected by the destination node D for the incoming connection section is denoted by $S(D)$.

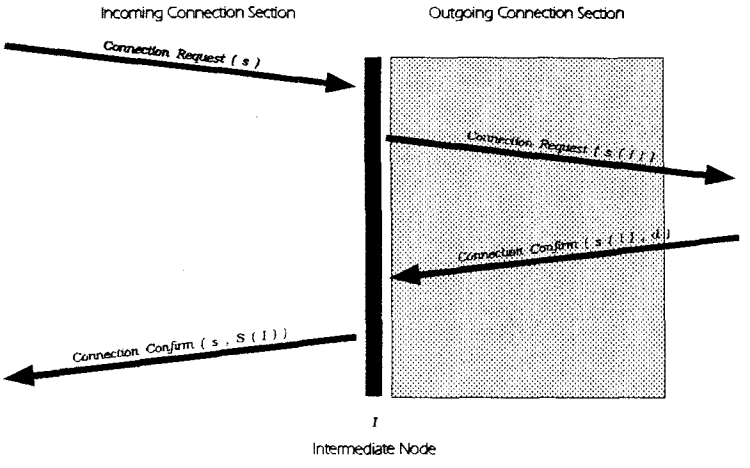
Figure 4: Sending *Connection Confirm* at a Destination Node

- the selected local reference number placed in the source local reference component,
- the recorded destination local reference number for the incoming connection section placed in the destination local reference component,
- the called party address component of the received *Connection Confirm* message transferred without changes to the corresponding called party address component

is subsequently sent on the incoming connection section. From then on, the node is expecting to receive either a *Data* or a *Released* message on the registered incoming or outgoing connection section before the expiration of its inactivity timer on the incoming connection section or the expiration of its inactivity timer on the outgoing connection section.

4.3 Receiving Connection Confirm Messages at the Originating Node

When the originating node receives a *Connection Confirm* message, it checks that the number in the destination local reference component of the received message marches its source local reference number for the outgoing connection section. If it does, it is a genuine reply to the connection request that it earlier made on the outgoing connection.



The number $s(I)$ has been selected by the intermediate node I for the outgoing connection section earlier and it matches the number found in the destination local reference component of the receiving *Connection Confirm* message. The source local reference number selected by I for the incoming connection section is denoted by $S(I)$.

Figure 5: Relaying Confirmation of Connection at an Intermediate Node

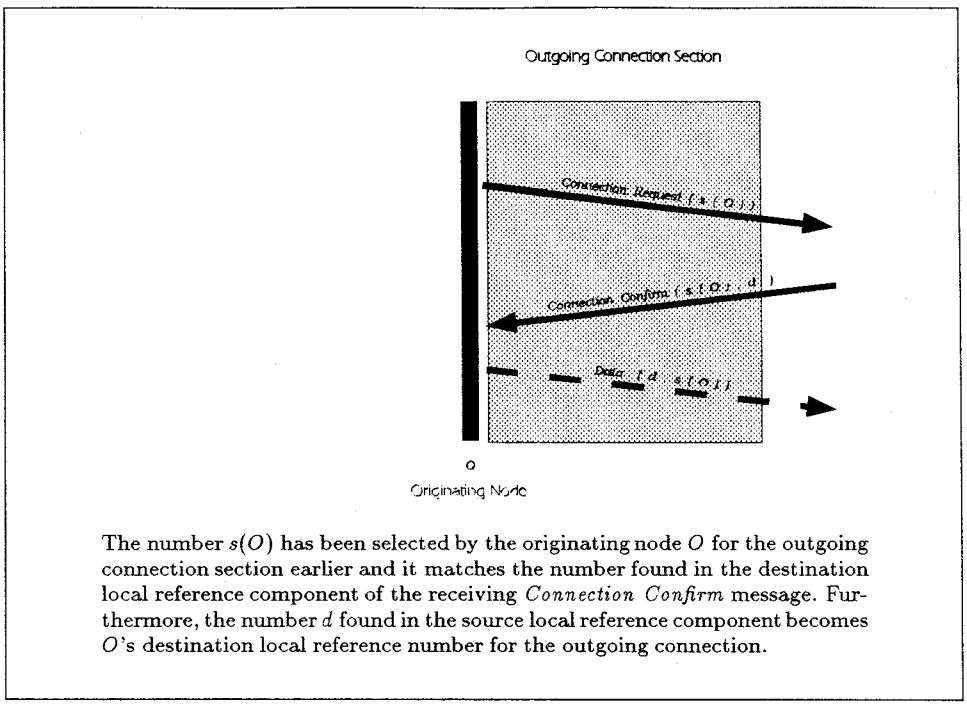
As a result the connection establishment between the originating and destination node is completed. Unless either

- the calling SCCP user at the originating node has since wished or wishes otherwise, or
- the connection establishment timer on the incoming connection section has expired,

data of the calling user can now be placed in a *Data* message together with the necessary source and destination local reference numbers. Subsequently, the *Data* message is sent on the outgoing connection section. From then on, the node is expecting to receive either a *Data* message or a *Released* message on the registered outgoing connection section before the expiration of its inactivity timer on the outgoing connection section.

5 Connection Refused Messages

The purpose of sending a *Connection Refused* message is to indicate to the calling SCCP user that the attempt to set up a signaling connection was unsuccessful. The only important component of a *Connection Refused* message is the destination local reference which is mandatory, Table 5/Q.713 [Q.700 88].



The number $s(O)$ has been selected by the originating node O for the outgoing connection section earlier and it matches the number found in the destination local reference component of the receiving *Connection Confirm* message. Furthermore, the number d found in the source local reference component becomes O's destination local reference number for the outgoing connection.

Figure 6: Receiving *Connection Confirm* at an Originating Node

5.1 Sending Connection Refused Messages

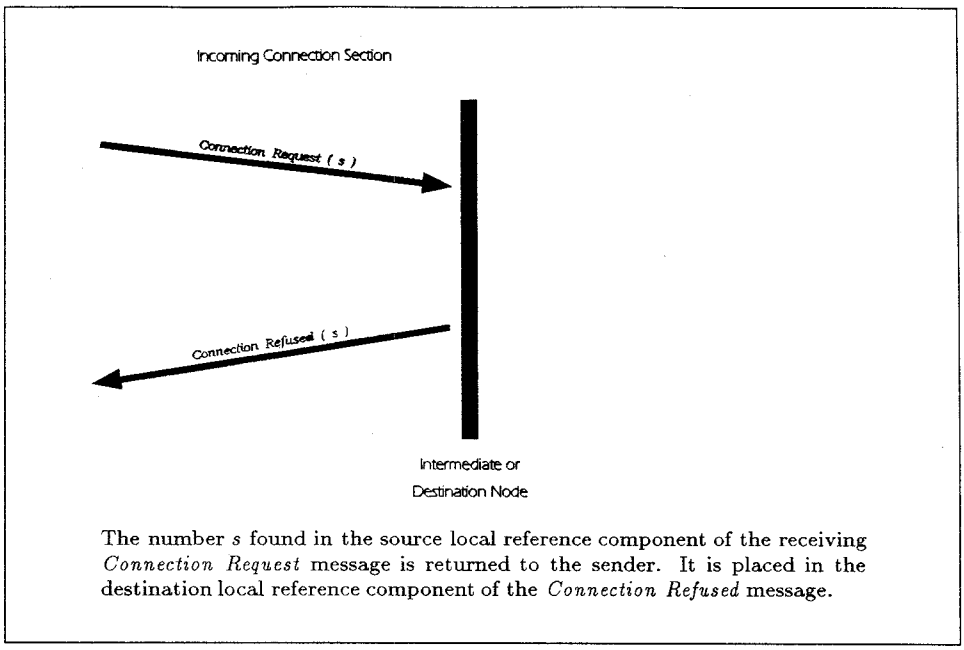
When a *Connection Request* message for an incoming connection section is received by an intermediate or a destination node, a refusal may be initiated.

For instance, an intermediate node may refuse the establishment due

- to the lack of resources, or
- to the failure of the translation of the global title found in the received *Connection Request* message, or
- to the unavailability of the signaling point determined by the translation of the global title.

On the other hand, the destination node may also refuse the establishment due to the lack of resources, or to the invocation of a **N-DISCONNECT REQUEST** primitive by the called SCCP user. In addition, an intermediate node also sends out a *Connection Refused* message on the incoming connection section if the outgoing connection establishment timer expires.

Finally, if a *Connection Refused* message is received at an intermediate node before the expiration of the outgoing connection establishment timer, it checks that the number in the destination local reference component matches its source local reference number for



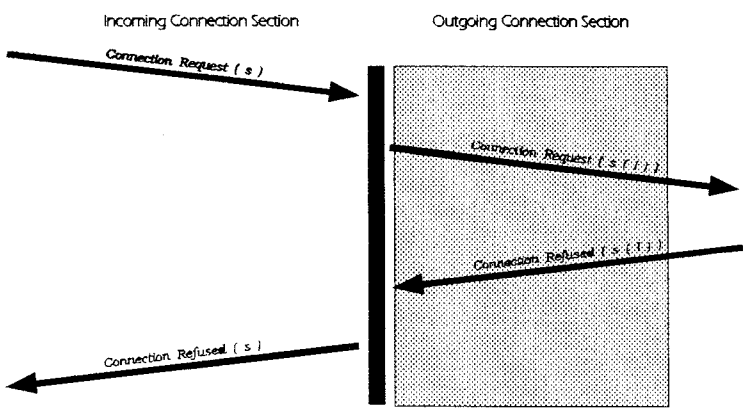
The number s found in the source local reference component of the receiving *Connection Request* message is returned to the sender. It is placed in the destination local reference component of the *Connection Refused* message.

Figure 7: Sending *Connection Refused* at a Destination or an Intermediate Node

the outgoing connection section. If it does, it is a genuine reply to the connection request that it earlier made on the outgoing connection section. In this case, the intermediate node is forced to refuse the connection request received earlier on the incoming connection section. For an intermediate or the destination node to construct a *Connection Refused* message as a reply to the received *Connection Request* message, the number found in the source local reference component of the received *Connection Request* message is placed in the destination source local reference component. Subsequently, the *Connection Refused* message is sent on the incoming connection section. After that, the resources associated with the incoming connection section are released and no further messages are sent or received on that section.

5.2 Receiving Connection Refused Messages at the Originating Node

When the originating node receives a *Connection Refused* message before either its connection establishment timer expires or its calling SCCP user wishes otherwise, it checks that the number in the destination local reference component of the received messages marches its source local reference number for the outgoing connection section. If it does, it is a genuine reply to the connection request that it earlier made on the outgoing connection. As a result, the calling SCCP user is informed about the unsuccessful attempt to set up a connection with the called SCCP user by invoking the **N-DISCONNECT INDICATION** primitive. After that, no further messages are sent and the resources



The number $s(I)$ has been selected by the intermediate node I for the outgoing connection section earlier and it matches the number found in the destination local reference component of the receiving *Connection Refused* message. Furthermore, the number s found in the source local reference component of the *Connection Request* message received earlier from the incoming connection section is returned to the sender. It is placed in the destination local reference component of the *Connection Refused* message.

Figure 8: Relaying Refusal of Connection at an Intermediate Node

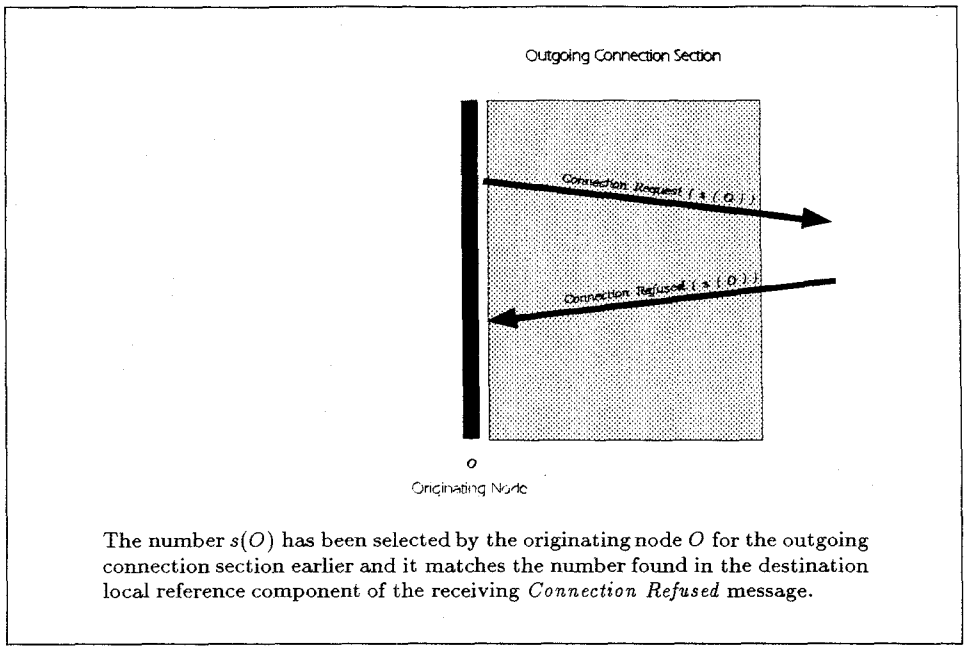
associated with the outgoing connection are released.

5.2.1 Expiration of the Outgoing Connection Establishment Timer at the Originating Node

Regardless whether a *Connection Confirm* or a *Connection Refused* message is received or not, as soon as the originating node's connection establishment timer on the outgoing connection section expires, the node informs the calling SCCP user about the unsuccessful attempt to set up a connection with the called SCCP user by invoking the **N-DISCONNECT INDICATION** primitive. Moreover, the resources associated with the connection section are released.

6 Released and Release Complete Messages

A signaling connection between two users of SCCP can be released. Two types of messages are required to initiate and complete the releasing of a connection, namely *Released* and *Release Complete*. Unlike the case of the *Connection Refused* messages, which are exclusively used during the connection establishment phase. In the case of the *Released*



The number $s(O)$ has been selected by the originating node O for the outgoing connection section earlier and it matches the number found in the destination local reference component of the receiving *Connection Refused* message.

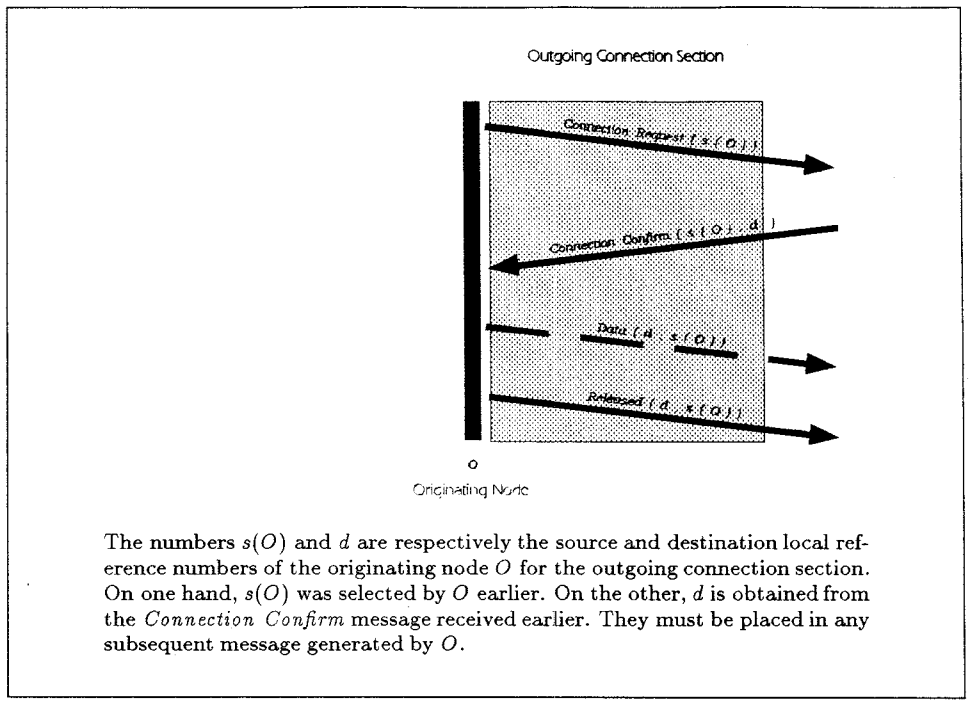
Figure 9: Receiving *Connection Refused* at an Originating Node

and *Release Complete* messages, they are used when the connection between two users is established. However, there is also one instance, which is described in the protocol of Q.714 [Q.700 88], where a *Released* message is sent out in conjunction with the sending of a *Connection Refused* message by an intermediate node, although to different nodes. The *Connection Refused* message is sent to the other end of the incoming connection section while the *Released* message is sent to the other end of the outgoing connection section. What leads to this particular instance is the expiration of the outgoing connection establishment timer at the intermediate node. We shall discuss this scenario in greater detail in the later subsections.

The important components of a *Released* message include destination and source local reference which are both mandatory, Table 6/Q.713 [Q.700 88]. Similarly, a *Release Complete* message also contains destination and source local reference components mandatorily, Table 7/Q.714 [Q.700 88].

6.1 Sending Released Messages After Establishment of Connection

The release of an established connection can be initiated at any node in the connection. Notice that in an established connection, a *Connection Confirm* message must have been sent by the destination end of every connection section in the connection and must have subsequently been received by the originating end of the connection section. Conse-



The numbers $s(O)$ and d are respectively the source and destination local reference numbers of the originating node O for the outgoing connection section. On one hand, $s(O)$ was selected by O earlier. On the other, d is obtained from the *Connection Confirm* message received earlier. They must be placed in any subsequent message generated by O.

Figure 10: Sending *Released* at an Originating Node

quently, each end node of every connection section must have recorded its source and destination local reference numbers of the connection section.

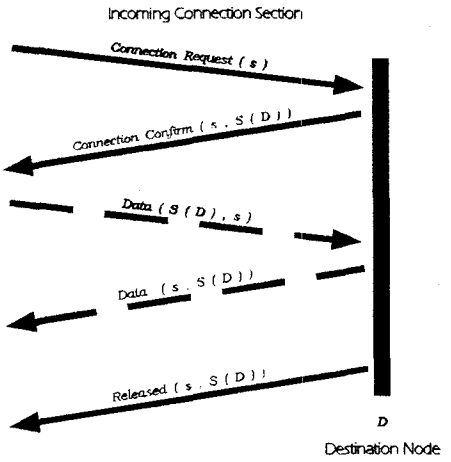
The cause-list for releasing a connection section is given as follows.

When the calling/called SCCP user at the originating/destination node wishes to disconnect the present connection, the node is informed via **N-DISCONNECT REQUEST** primitive [2]. Subsequently, the release of the outgoing/incoming connection section follows.

On the other hand, a node in the connection (including the originating or destination) may also wish to release the connection due to a failure to maintain a connection section or to the expiration of the receive inactivity control timer on a connection section.

In the case of the originating node, the failure or the expiration of the receive inactivity timer may occur on the outgoing connection section. Releasing the connection section subsequently follows. In the case of the destination node, the failure or the expiration of the receive inactivity timer may occur on the incoming connection section; and similarly it is followed by the release of the connection section. In the case of an intermediate node, the failure or the expiration of the timer may occur on either the incoming or the outgoing connection section. Both connection sections will subsequently be released. Finally, if

[2]We remark that the originating node may have already been informed by its calling SCCP user before the arrival of the *Connection Confirm* message at the node.



The numbers $S(D)$ and s are respectively the source and destination local reference numbers of the destination node D for the incoming connection section. On one hand, $S(D)$ was selected by D earlier. On the other, s is obtained from the *Connection Request* message received earlier. They must be placed in any subsequent message generated by D.

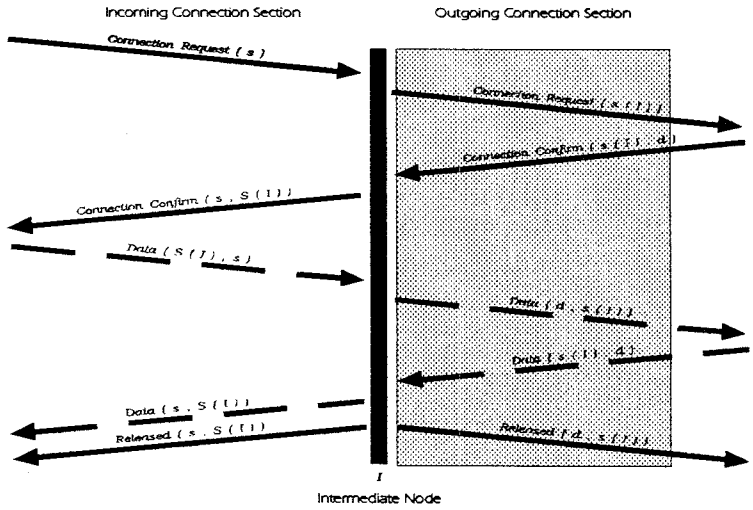
Figure 11: Sending *Released* at a Destination Node

an intermediate node receives a *Released* message associated with a connection section (*i.e.* the numbers found in the source and destination local reference components match the node's destination and source local reference numbers for the connection section respectively), then the release of the other connection section is also followed. This scenario will further be discussed in Section 6.3.4.

To construct a *Released* message for a connection section, a node is required to place its destination and source local reference numbers of the connection section into the respective destination and source local reference components. Subsequently, the message is ready to be sent on the connection section. From then on, unless there is an internal error, it is expecting to receive either a *Released* or a *Release Complete* message on the connection section.

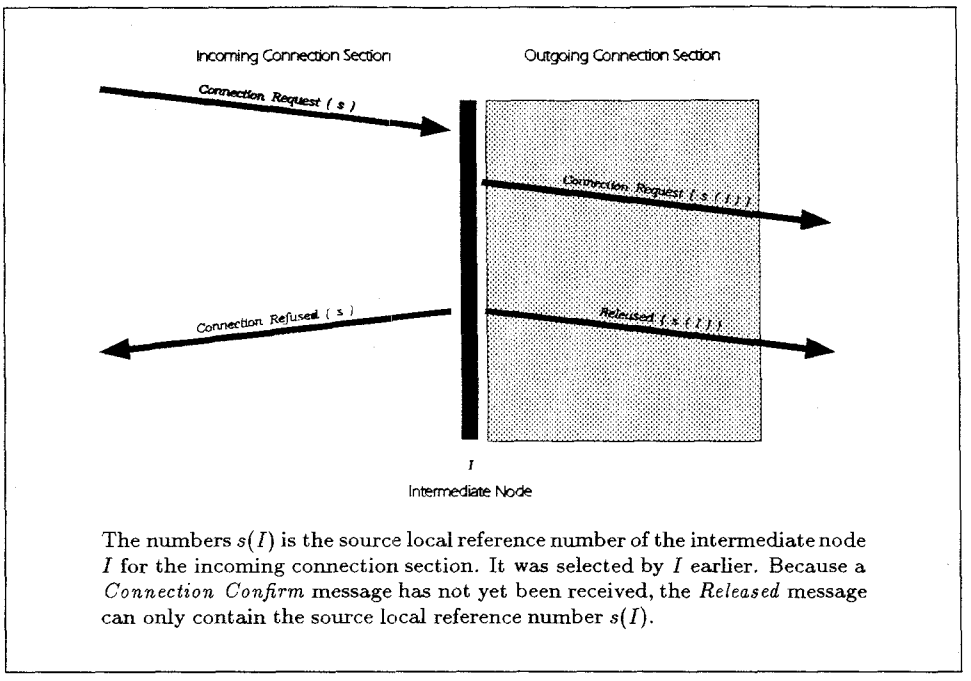
6.2 Sending Released Messages in Conjunction with Sending of Connection Refused Messages by Intermediate Nodes

As mentioned in Section 5.1, an intermediate node sends out a *Connection Refused* message on the incoming connection section if the outgoing connection establishment timer expires. In this case, the intermediate node is required to send a *Released* message on the outgoing connection section in order to release that connection section, Section 3.2.1 of Q.714 [Q.700 88]. However, it is important to note that no message (*Connection Con-*



The numbers $S(I)$ and s are respectively the source and destination local reference numbers of the intermediate node I for the incoming connection section. On one hand, $S(I)$ was selected by I earlier. On the other, s is obtained from the *Connection Request* message received earlier. They must be placed in any subsequent message generated by I for sending on the incoming connection section.
The numbers $s(I)$ and d are respectively the source and destination local reference numbers of the intermediate node I for the outgoing connection section. On one hand, $s(I)$ was selected by I earlier. On the other, d is obtained from the *Connection Confirm* message received earlier. They must be placed in any subsequent message generated by I for sending on the outgoing connection section.

Figure 12: Sending *Released* at an Intermediate Node



The numbers $s(I)$ is the source local reference number of the intermediate node I for the incoming connection section. It was selected by I earlier. Because a *Connection Confirm* message has not yet been received, the *Released* message can only contain the source local reference number $s(I)$.

Figure 13: Sending *Released* in Conjunction with the Sending of *Connection Refused* at an Intermediate Node

firm or *Refused*) has yet been received from this outgoing connection section. Hence, the intermediate node has not been able to record the destination local reference number of this outgoing connection section. Consequently, when the *Released* message for sending on the outgoing connection section is constructed, the destination local reference component of the message can not be filled. Although the destination local reference is one of the mandatory components according to Table 6/Q.713 [Q.700 88], it follows from Section 3.1.2 of Q.714 [Q.700 88] that this particular mandatory requirement is only a conditional one. We quote the following from Section 3.1.2 of Q.714 [Q.700 88] :

"*Once the destination local reference number is known, it is a mandatory field for all messages transferred on that connection section.*"

In other words, the *Released* message constructed without the destination local reference component during the connection establishment is still valid. This *Released* message would contain only the intermediate node's source local reference number of the outgoing connection section which is being released. After it has been sent on the outgoing connection section, unless there is an internal error, the intermediate node is expecting to receive either a *Released* or a *Release Complete* message from the outgoing connection section. However, one of the messages that the node is not expecting is a *Connection Refused* message. In the following subsections, the subsequent actions of the nodes after the arrival of a *Released* or *Release Complete* message are described.

6.3 Receiving Released Messages

When a node (including originating, intermediate and destination) in a connection receives a *Released* message, it needs to check if the message relates to any of its current connection sections (*i.e.* if the message indicates the release of one of its current connection sections by the node at the other end).

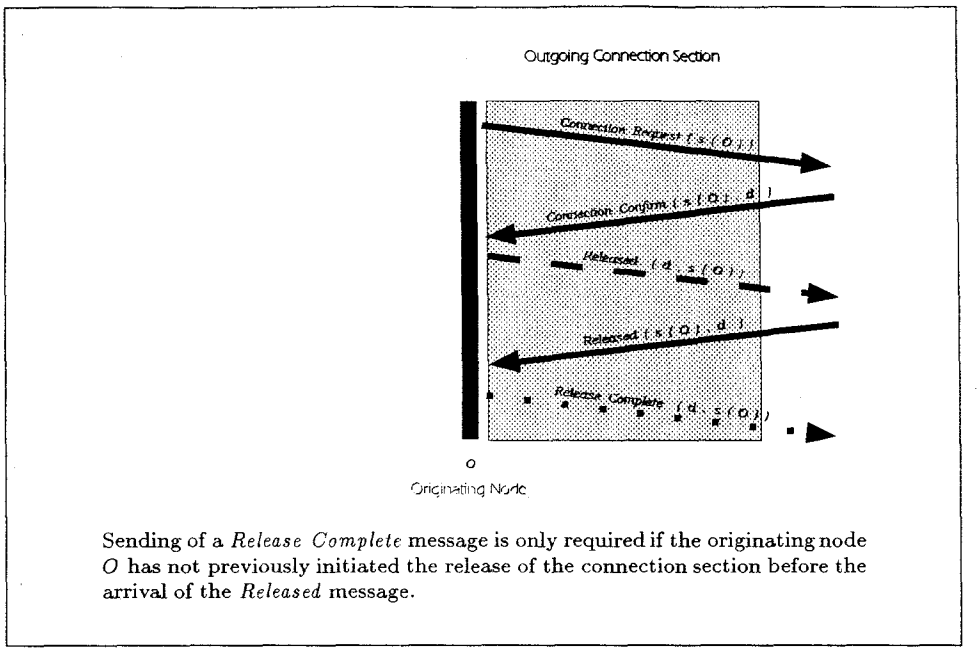
6.3.1 Validity of Released Messages Coming from the Outgoing Connection Section

It is important to note that for the originating and intermediate nodes, each of them has previously received a *Connection Confirm* message from its outgoing connection section. As a *Connection Confirm* message includes both source and destination local reference components mandatorily, each of the nodes can conclude that the node at the other end of the outgoing connection section must have known the destination local reference number for the connection section. As a result, each of the nodes can not accept a *Released* message coming from its outgoing connection section to be valid if the message contains empty destination local reference component.

6.3.2 Validity of Released Messages Coming from the Incoming Connection Section

On the other hand, the same can not be said for the intermediate and destination nodes when they receive a *Released* message from their respective incoming connection sections. If a message is received for the first time from the incoming connection section by a node after it has sent a *Connection Confirm* message, and also this message is a *Released* message, then the node has to accept this *Released* message to be valid even if the destination local reference component may be empty. It is not possible for the node to conclude that the node at the other end of the incoming connection section had received the *Connection Confirm* message from the node before the *Released* message was sent [3]. However, if some *Data* message has since been received from the incoming connection section after the sending of the *Connection Confirm* message, then the node only accepts a subsequent *Released* message to be valid if both local reference components of the message are non-empty. The reason is that a *Data* message can only be sent if a *Connection Confirm* message is received by the node at the other end of the incoming connection section.

[3]For example, the connection establishment timer at the other end of the connection section may have expired and subsequently sends a *Released* message before the arrival of the *Connection Confirm* message.



Sending of a *Release Complete* message is only required if the originating node O has not previously initiated the release of the connection section before the arrival of the *Released* message.

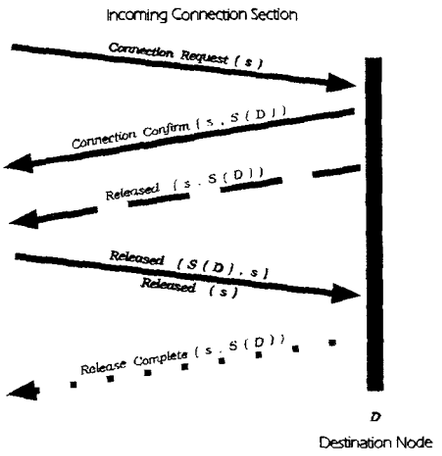
Figure 14: Receiving *Released* at an Originating Node

6.3.3 Actions by the Originating or Destination Node

In the case of the originating or destination node, there is only one connection section; it is outgoing in the case of originating node while it is incoming in the case of destination node. The node checks that the numbers in the source and destination (if it exists) local reference component match its destination and source local reference numbers for the connection section respectively. If they match, the message is therefore a *Released* message coming from the node at the other end of the connection section. If the node itself has not initiated a release on the connection section, it informs its SCCP user via **N-DISCONNECT INDICATION** primitive, and it also replies to the *Released* message by sending a *Release Complete* message containing its source and destination local reference numbers of the connection section. After that, the resources associated with the connection section are released. However, if the node itself has already initiated a release on the connection section, no further message will be sent and the resources associated with the connection section are released.

6.3.4 Actions by the Intermediate Nodes

In the case of an intermediate node, there are the two associated connection sections, namely the incoming and outgoing connection sections. The node checks that the numbers in the source and destination (if it exists) local reference components match its destination and source local reference numbers for either one of the two connection sec-



Sending of a *Release Complete* message is only required if the destination node D has not previously initiated the release of the connection section before the arrival of the *Released* message.

Figure 15: Receiving *Released* at a Destination Node

tions. Note that even through the node's destination local reference numbers for the incoming connection section and for the outgoing connection section may be the same [4], the node's source local reference numbers for the two connection sections must be different because of the way they are selected by the node. Hence, the numbers in the source and destination (if it exists) local reference components match its destination and source local reference numbers for at most one connection section. Furthermore, if the destination local reference component does not exist, then it cannot be a valid *Released* message coming from the outgoing connection section.

Suppose that they match the node's destination and source local reference numbers for one connection section. The message is therefore a *Released* message coming from the node at the other end of the connection section. If the node itself has not initiated a release on the connection section, then it has also not initiated a release on the associated connection section. It firstly replies to the *Released* message by sending a *Release Complete* message containing its source and destination local reference numbers of the connection section before the resources associated with the connection section are released. Secondly, the node is required to construct a *Released* message containing its source and destination local reference numbers for the associated connection section. Subsequently, the message is sent on the associated connection section.

For the remaining possibility: the node has already initiated a release on the connection section, it follows that it has also initiated a release on the associated connection

[4]They are respectively generated independently by the node at the other end of its incoming connection section and the node at the other end of its outgoing connection section.

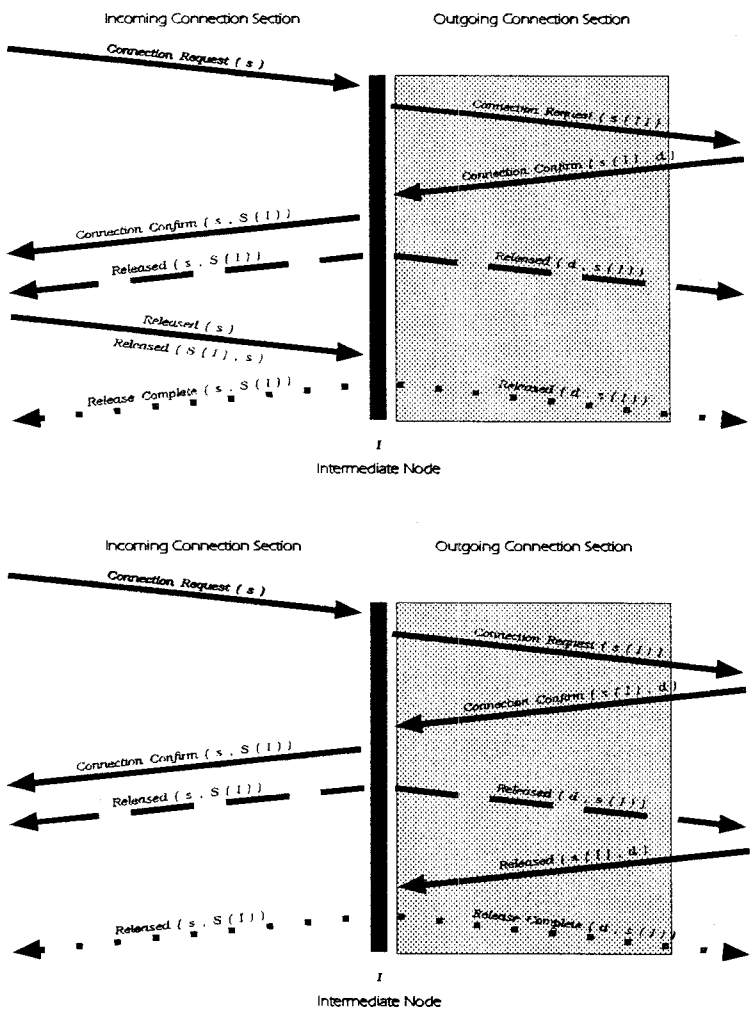
section. In this case, no further message will be sent on either connection section and the resources associated with the connection section (*i.e.* the one on which the *Released* message is received) are released. Notice that the resources associated with the associated connection section may or may not have been released. If they have not been released, the node should still maintain them until a *Released* or a *Release Complete* message on that connection section is received.

6.4 Receiving Release Complete Messages

When a node (including originating, intermediate and destination) in a connection receives a *Release Complete* message, it needs to check if the message concerns with any of its connection sections where the release has previously been initiated by itself.

In other words, it examines the received *Release Complete* message by matching respectively the numbers found in the destination and source local reference components with its source and destination (if it exists) local reference numbers of a connection section where the release has previously been initiated by itself. If the numbers match, then it is a genuine reply to the *Released* message it has earlier sent out on the connection section. Subsequently, no further message will be sent and the resources associated with the connection section are released.

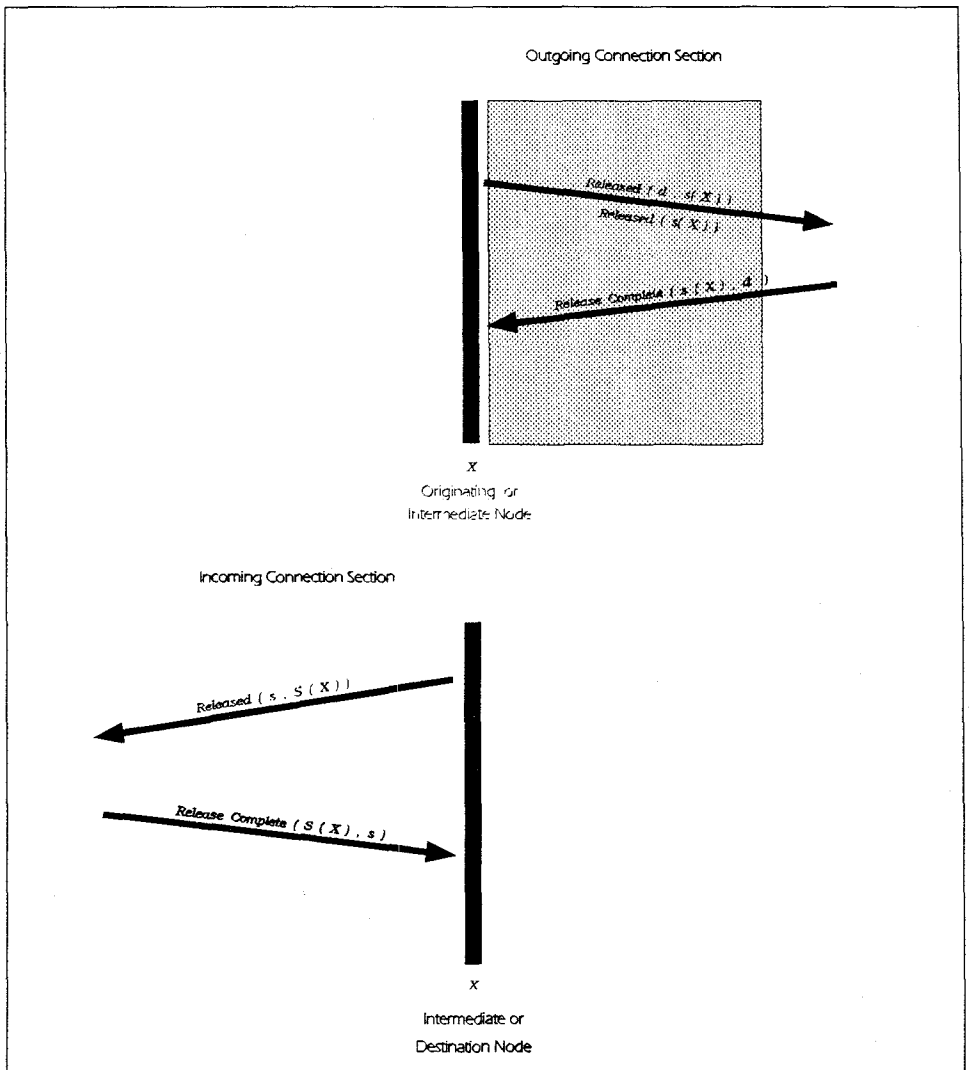
On the other hand, if the numbers do not match, the node understands that the received message does not concern with any of its current connection sections. In this case, the received message is simply ignored.



If the intermediate node I has not previously initiated the release of both associated connection sections before the arrival of the *Released* message, then it is required to

- send a *Release Complete* message on the connection section where the *Released* message arrived,
- send a *Released* message on the other connection section.

Figure 16: Receiving *Released* at an Intermediate Node



When a node X receives a *Release Complete* message on a connection section, it is required to check that the numbers found in the source and destination local reference components match its destination and source local reference numbers for that connection section respectively.

Figure 17: Receiving *Release Complete* at a Node

7 Initiating the Release of a Connection by a Node not in the Connection

In this section, we shall use the description of the behaviour of the nodes in the establishment and release phases of a connection given in the last sections to demonstrate a scenario where the connection between the originating and destination nodes may be released by a node which is not in the connection. Furthermore, it cannot be detected by any node in the connection. Every node in the connection would believe that the node at the other end of one of its connection sections has initiated the release and not the off-line node.

7.1 A Connection

We let $\mathcal{C}$ be a connection consisting of three nodes A, B and C [5], where A, B and C are the originating, intermediate and destination nodes respectively. In the following, we shall assume that the calling SCCP user at A informs A the global title of the called SCCP user located at C when it makes its connection establishment request to A. Furthermore, B is the node that is determined from the global title by the translation function at A. In addition, C can be determined from the global title also by the the translation function at B.

7.2 Making the Connection Establishment Request

The originating node A begins the establishment of $\mathcal{C}$ by sending a *Connection Request* message with its selected source local reference number $s(A)$ as in Figure 1.

As described in Figure 2, when B receives the *Connection Request*, which requires relaying, from A on its incoming connection section, it associates the connection section with the number $s(A)$ found in the *Connection Request*. In other words, $s(A)$ becomes B's destination local reference number of the incoming connection section. In addition, it selects its own source local reference $s(B)$ for the outgoing connection section and includes it in a *Connection Request* for sending on the outgoing connection section. Subsequently, it starts its connection establishment timer on the outgoing connection and waits for the arrival of either a *Connection Confirm* or a *Connection Refused* message.

After a *Connection Request* message has arrived at the destination node C, Figure 3, either it refuses the establishment of $\mathcal{C}$ by sending a *Connection Refused* message or it confirms the establishment of $\mathcal{C}$ by sending a *Connection Refused* message.

[5] In fact, A and B may also be intermediate nodes.

7.3 Refusing the Request

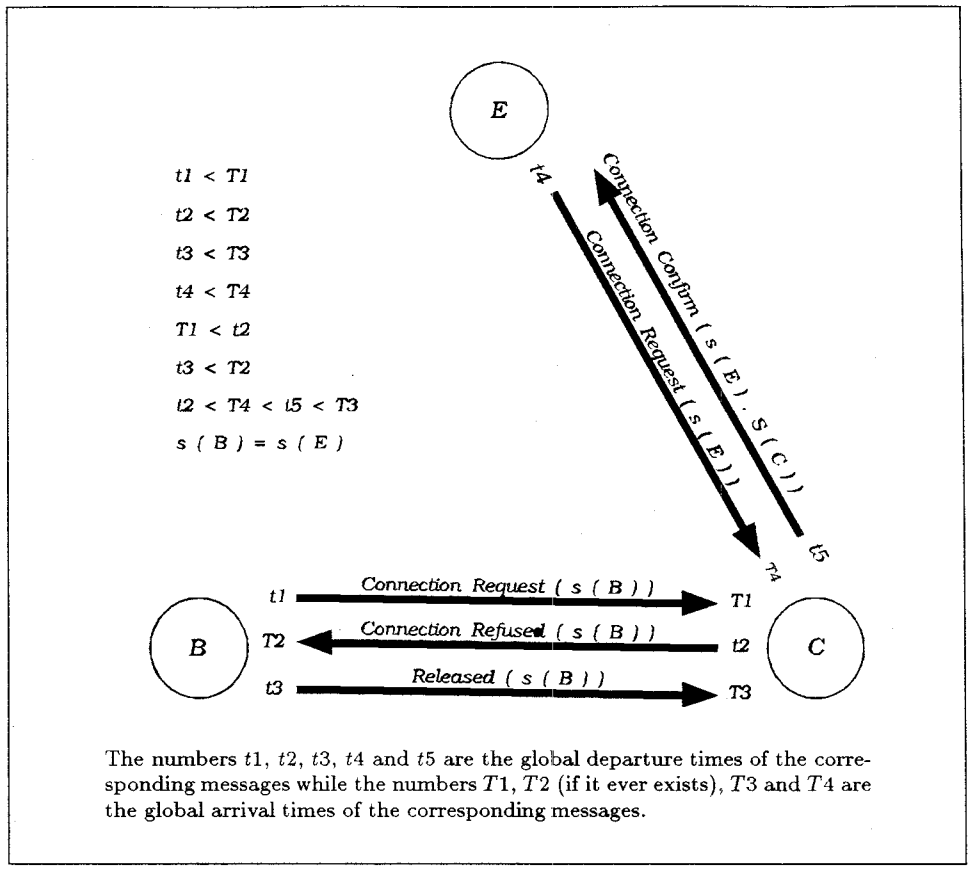
We shall assume that C takes up the option of refusing $\mathcal{C}$. Hence C sends out a *Connection Refused* message as described in Figure 7. After that, C is free to receive another *Connection Request* message, if any, again.

In the CCITT Q.700 series [Q.700 88], there is no mention how long a destination node is allowed to decide what type of messages it should send out after the arrival of a *Connection Request* message. In addition, the maximum time-out limit of a connection establishment timer at an intermediate node is not given neither. As a result, it is possible that B's connection establishment timer on the outgoing connection has expired before the arrival of the *Connection Refused* message sent by C. In this case, according to Figure 13, B sends a *Connection Refused* message with its destination local reference number of the incoming connection section, namely $s(A)$, on the incoming connection section. At the same time, B also sends a *Released* message on the outgoing connection section to C. It is important to note that only the source local reference number of the outgoing connection section, namely $s(B)$, is included in this message. Let us denote this message by *Released*$(s(B))$.

7.4 Establishment of Another Connection

As mentioned in the last subsection, the node C is ready to receive another *Connection Request* message after refusing the connection establishment requested by B.

We suppose that $\mathcal{C}'$ is another connection including a node E and C as consecutive nodes. In other words, E and C are the end nodes of a connection section of $\mathcal{C}'$ and E sends a *Connection Request* message with its selected source local reference number $s(E)$ on this connection section to C. We suppose that C receives this *Connection Request* message from E right after it has refused the request from B. Notice that C may either be an intermediate or a destination node in the connection $\mathcal{C}'$ as a connection consists of one or more connection sections. In either case, it is possible, as indicated by Figure 4 or Figure 5, for C to reply positively to the connection request from E; *i.e.* C sends out a *Connection Confirm* message with its selected source local reference number $S(C)$ and also with $s(E)$ (its destination local reference number). This message is sent on the incoming connection section to E.



The numbers $t1$, $t2$, $t3$, $t4$ and $t5$ are the global departure times of the corresponding messages while the numbers $T1$, $T2$ (if it ever exists), $T3$ and $T4$ are the global arrival times of the corresponding messages.

Figure 18: The Scenario

7.5 Arrival of a Released Message Concerning a Previous Connection Section

At this moment, the message *Released*$(s(B))$ can in fact arrive at C; and we assume that it is indeed the first one to do so after the sending of the *Connection Confirm* message. The node C checks this message and discovers that it contains only the source local reference but not the destination local reference component. Since C did not receive any other message between the departure of its *Connection Confirm* message and the arrival of the *Released* message, the arrived *Released* message can still be valid even if it contains only the source local reference but not the destination local reference component, Figure 15 or Figure 16. Consequently, matching the number found in the source local reference component of the received message against the number $s(E)$, which has been found in the same component of the *Connection Request* message received earlier for the present incoming connection section and which has been associated with the present incoming connection section, is the only way to validate if this received *Released* message concerns with the present incoming connection section.

7.6 Coincidence of Two Destination Local Reference Numbers

However, both numbers $s(E)$ and $s(B)$ are not selected by the node C itself. The number $s(E)$ was selected independently by E while $s(B)$ was selected independently by B. Hence, it is possible that they can be identical.

7.7 Validation of the Released Message and Subsequent Completion of Releasing of the Connection Section

If the above two numbers $s(E)$ and $s(B)$ are identical, the node C would therefore conclude that the message *Released*$(s(B))$ is valid and also that it concerns with the present incoming connection section, with which the number $s(E)$ is associated.

Consequently, it sends out a *Release Complete* message with the numbers $s(E)$ and $S(C)$ placed in the destination and source local reference components respectively on the incoming connection section before releasing the resources; furthermore, if there is an associated connection section, it also initiates the release on that connection section, Figure 17.

7.8 Completion of Releasing at the Other Nodes

After describing the release of the connection section(s) at C, it remains to consider the possible behaviour at the nodes E and B.

7.8.1 Behaviour at E

Firstly, there are a number of scenarios that may occur at E. We recall that C has previously sent out a *Connection Confirm* message on the connection section which connects E and C. The following figures: Figures 12, 13 and 16 are all applicable to the node E. In each case, a *Released* message with destination local reference number $S(C)$ and source local reference number $s(E)$ is sent by E on the connection section which connects E and C. Furthermore, it subsequently waits for the arrival of a *Release Complete* message with destination local reference number $s(E)$ and source local reference number $S(C)$. However, as mentioned in the last subsection, C has already sent out such a message.

7.8.2 Behaviour at B

Finally, we consider the behaviour of B. As described in Subsection 7.3 above, B has sent out the message *Released*($s(B)$) and has since waited for a *Release Complete* message. Unfortunately, such a message would never arrive since its generator (*i.e.* C) has already refused the connection section which connects B and C from the assumption in Subsection 7.3. As a result, B's release timer on the outgoing connection section will eventually expire. The node B will therefore inform the maintenance function, Section 3.3.4.2 of Q.714 [Q.700 88]. However, this maintenance function is subjected for further study as mentioned in Q.714 [Q.700 88]. Nevertheless, as a minimal requirement of the maintenance function we would expect that B would be informed about the C's refusal of the connection section so that it is allowed to complete its own release of all the resources of the connection section.

This therefore completes our description of the scenario where a connection is released by a node not belonging to the connection.

Remark 7.1 (Assumed Capability of the Undefined Maintenance Function)

We remark that it is not enough for the maintenance function to just monitor C's current source and destination local reference numbers for C's incoming connection section and match them to B's current destination and source local reference numbers for B's outgoing connection section. The reason is mainly due to the absence of B's destination local reference number for its outgoing connection section. To uniquely identify a connection section, both local and source reference numbers are required to be present.

Consequently, an audit trail on the history of the messages departing from C is required and it may not be too practical.

8 Possible Improvement

We begin the discussion of a possible improvement of the SCCP connection-oriented protocol by describing the following observations.

In the first observation, the behaviour of an originating node with an expired outgoing connection establishment timer is described.

Observation 8.1

In this observation, we shall reason why the scenario in the last section (Section 7) would not occur if the connection $\mathcal{C}$ consists of only two nodes, namely B and C, rather than

three nodes as stated in Section 7. In other words, we assume that B is the originating node and C is the destination node in $\mathcal{C}$ which consists of the single connection section.

In this case, when B's outgoing connection establishment timer expires, it does not initiate the release of the connection section by sending a *Released* message as in the case described in Section 7, but it simply releases all the resources of the connection section and informs its calling SCCP user.

Consequently, after refusing the request of connection from B and accepting the request of connection from E, it is not possible that C would receive a *Released* message which theoretically concerns with one of its previous incoming connection sections but may practically be confused as one concerning with the current incoming connection section.

In the second observation, we describe the behaviour of an intermediate or a destination node after it sends out or relays a *Connection Confirm* message on the incoming connection section and subsequently waits for a non-existing message from that connection section.

Observation 8.2

Let us now consider the behaviour of C in the case where it accepts the request of connection from B but its *Connection Confirm* message cannot arrive before the expiration of B's outgoing connection establishment timer. It is important to note that C has its destination as well as its source local reference numbers for the incoming connection section, namely $s(B)$ and $S(C)$. The number $s(B)$ has been found in the *Connection Request* from B while the number $S(C)$ was selected by itself.

Obviously, C would not receive any further message from B as B has already released all the resources of its outgoing connection section. Hence, C's inactivity timer will eventually expire and a *Released* message with $s(B)$ and $S(C)$ is sent on its incoming connection section. After that, for the same reason, C's release timer on the incoming connection will also expire and subsequently the maintenance function will be informed. However, unlike the case as described in Remark 7.1, it is enough for the maintenance functions to simply monitor B's source and destination local reference numbers to conclude that B is no longer belonging to the same connection section that C is currently is in. The maintenance function matches C's destination and source local reference numbers with B's source and destination local reference numbers (if they exist) respectively. There are two possibilities:

- B does not currently have a source (and hence destination) local reference number(s) for its outgoing connection,
- B currently has a source (and possibly destination) local reference number(s) for its outgoing connection section.

In the first possibility, B is in an idle state. It therefore cannot be in the same connection section that C is currently in. In the second possibility, B has a source local reference number, N say, for it outgoing connection section. Both $s(B)$ and N are selected by the node B itself for its outgoing connection section, and the same number is not selected twice either consecutively or within a short space of time. Consequently, B and C are simultaneously belonging to the same connection section if and only if $s(B)$ and N are identical. However, since B has in fact released all the resources of the outgoing connection section that C is in by our assumption, N must be different from $s(B)$. As a result, the maintenance function is able to inform C about the disconnection more efficiently.

8.1 Proposed Alternative to the SCCP Connection-Oriented Protocol

From the comparison of the behaviour between an intermediate node (Figure 13 and Section 7.3) and an originating node (Observation 8) when both of their outgoing connection establishment timers expire, it follows that the intermediate node may be able to release a connection, to which it does not belong, but not the originating node.

Furthermore, Observation 8.2 shows how the node at the other end of the outgoing connection section of the originating node releases the resources of the connection section. The same should also be possible if an intermediate node also releases all the resources of the outgoing connection section rather than sending a *Released* message and waiting for a reply when its connection establishment timer expires.

We are therefore proposing that releasing all the resources of the connection section, where the connection establishment expires, should be followed by intermediate nodes as well as originating nodes. In other words, we simply ignore Figure 13. As a consequence, there will be no sending of *Released* message during a connection establishment phase. There will not be any valid *Released* message which has only the source local reference component but not the destination local reference component. All valid *Released* messages include the two components (*i.e.* they are truly mandatory) and there are used only after connections have been established.

Finally, we mention that the overall performance would not suffer when the proposed alternative is preferred. The reason is simply due to the fact that if the behaviour of the nodes and their SCCP users is allowed in the alternative protocol, then it is also possible in the original SCCP protocol.

9 Conclusion

In this paper, we have described the Signaling System Number 7 Signaling Connection Control Part Protocols as stated in [Q.700 88]. We have demonstrated that a security flaw is possible, namely a connection can be released by an off-line node that does not belong to the connection. Nevertheless, we are able to propose an equally efficient alternative that can eliminate the above security flaw.

References

[Q.700 88] CCITT recommendations Q.700-Q.716 Specifications of Signaling System No. 7, IXTH PLENARY ASSEMBLY, MELBOURNE, 14-25 NOVEMBER 1988, Geneva 1989

Department of Defence

DOCUMENT CONTROL DATA SHEET

Page Classification	
UNCLASSIFIED	
Privacy Marking/Caveat (of document)	

1a. AR Number	1b. Establishment Number	2. Document Date	3. Task Number
AR-006-797	ERL-0582-RR	NOV 91	ADF 89/229

4. Title

ABOUT THE SIGNALING SYSTEM NO 7 SIGNALING CONNECTION CONTROL PART PROTOCOLS

5. Security Classification

Document	Title	Abstract
U	U	U

S (Secret) C (Confi) R (Rest) U (Unclass)

* For UNCLASSIFIED docs with a secondary distribution LIMITATION, use (L) in document box.

6. No. of Pages	38
7. No. of Refs.	1

8. Author(s)

M.K.F. Lai

9. Downgrading/Delimiting Instructions

None

10a. Corporate Author and Address

Electronics Research Laboratory
PO Box 1600
SALISBURY SA 5108

10b. Task Sponsor

DGCIS

11. Officer/Position responsible for

Security.....

Downgrading.....

Approval for Release.....DERL.....

12. Secondary Distribution of this Document

APPROVED FOR PUBLIC RELEASE

Any enquiries outside stated limitations should be referred through DSTIC, Defence Information Services, Department of Defence, Anzac Park West, Canberra, ACT 2600.

13a. Deliberate Announcement

No limitation

13b. Casual Announcement (for citation in other documents)

[X] No Limitation

[ ] Ref. by Author , Doc No. and date only.

14. DEFTEST Descriptors

Signaling Connection Control Part (SCCP), Signaling System NO 7 (SS7), Networks, Signal processing

15. DISCAT Subject Codes

2501

16. Abstract

A description of the Signaling System No 7 Signaling Connection Control Part Protocol is given. A scenario where a connection can be released by an off-line not belonging to the connection is demonstrated. Finally an improvement of the protocol for eliminating this particular flaw is proposed.

Page Classification

UNCLASSIFIED

16. Abstract (CONT.)

17. Imprint

Electronics Research Laboratory
PO Box 1600
SALISBURY SA 5108

18. Document Series and Number	19. Cost Code	20. Type of Report and Period Covered
ERL-0582-RR	723238	RESEARCH REPORT

21. Computer Programs Used

NONE

22. Establishment File Reference(s)

23. Additional information (if required)