Accredited Standards Committee*

X3, Information Processing Systems

Doc. No.: X3S3.3/92-246

Date: June 11, 1992

Reply to: James Moulton
ONS
PO Box 669
Sterling, VA 22170
(703)430-2668
(703)430-5932
moulton@ons.com

To: X3S3.3 Members

From: James Moulton

Title: Proposed USA Contribution to SC 6 on Multicast Network Service

Abstract: The attached contribution is a completed draft of the Section 4 - Multipeer Network Service. This service description is based on the MPDT architecture submitted to SC 21 for review and is in the format of the new ISO 8348/CCITT X.213.

Section 4 - MultiPeer Data Transmission

20 Features of the MultiPeer Data Transmission Service

The MultiPeer Data Transmission (MPDT) NS provides the following features to the NS user:

a) The means to activate an MPDT Group Conversation (GC) with one or more NS users for the purpose of transferring NS-user-data in the form of NSDUs under the control of AGI. More than one conversation may exist between the same group of NS users.

b) The means to join an on-going MPDT group conversation.

c) The establishment of an agreement between the group of NS users and the NS provider for a certain QOS associated with the group conversation.

d) The means of transferring NSDUs in a MPDT group conversation under the constraints imposed by QOS. The transfer of NSDUs, which consist of an integer number of octets, is transparent, in that the boundaries of NSDUs and the contents of NSDUs are preserved unchanged by the Network Service, there are no constrainTS on the NSDU content imposed by the Network Service. It may not be known whether any or all of the potential receivers receive the NSDUs.

The transfer of NSDUs are subject to the control of QOS. QOS specifies whether the NSDUs are maintained in sequence, error controlled, or flow controlled.

e) The means by which the receiving NS users may flow control the rate at which sending NS users may send NSDUs. The coordination of flow control between the group of NS users receiving data is for further study.

f) The means to leave an on-going MPDT group conversation.

g) The unconditional, and therefore possible destructive deactivation of an MPDT group conversation by the initiator of the group conversation or the NS provider.

21 Model of MultiPeer Data Transmission Network Service

21.1 *General*

This Service Definition uses the abstract model for a layer service defined in [CCITT X.210 | DIS 10731]. The model defines the interactions between the NS users and the NS provider which take place at the NSAPs. Information is passed between the NS user and the NS provider by service primitives, which may convey parameters.

For a given instance of communication, the mode of service provided to all NS users is the same (i.e., unicast or MPDT). Choice of provision of the unicast or MPDT service is made in accordance with the procedures in Clause 24.

21.2 *Model of a MultiPeer Data Transmission Group Conversation*

Among the endpoints of a GC, there exits a flow control function which relates the behavior of the NS users receiving NS-user-data (e.g., accepting data from the NS provider) to the ability of another NS user to send data. As a means of specifying this flow control feature and its relationship with other capabilities provided by the Network Service, a queue model of a GC is used.

This queue model of a GC is discussed only to aid in the understanding of the end-to-end service features perceived by users of the Network Service. It is not intended to serve as a substitute for a precise, formal description of the Network Service, nor as a complete specification of all allowable sequences of NS primitives.

In addition, this model does not attempt to describe all the functions or operations of Network Layer entities which are used to provide the Network Service. No attempt to specify or constrain Network Service implementations is implied.

The queue model for a GC consists of a model for a single sender and a model for a single receiver. It is for further study whether a model of more than one sender or more than one receiver is required.

The queue model is described in terms of objects. objects consist of service primitives, octets of normal data, or other indications that relate to service primitives.

The queue model defines the allowed sequence of events that can occur on the queue and objects in the queue. It is limited to normal operation in that an error (e.g., loss of data) is not modeled as an allowed event.

21.2.1 *General*

The operation of a GC may be described in an abstract way as an association among NSAPs. The association is between the senders of NSDUs and a group of one or more receivers. The potential set of senders is defined in the establishment of the group which is a part of System Management.

The characteristics of the association between sending NS users and receiving NS users participating in the GC is governed the QOS agreed during GC activation.

The queue model for MPDT is described in terms of send queues and receive queues. The queue model is used to describe the flow of multiple NSDUs from a single sender to multiple receivers. The send queue model describes the view from the perspective of the NSDU sender. The receive queue model describes the view from the NSDU receiver.

For senders of NSDUs, each sender is modeled as having one send queue. The queue describes the interactions between the NSDU sender, the NS provider, and the NSDU receivers' receive queues.

Receivers will have a single queue from all senders in the GC.

The ability of an NS user to add objects to a queue may be determined by the behaviors of all NS users removing objects for these separate queues and the state of individual queues. At each NS user participating in the GC there will be a pair of queues. One queue is for sending and one is for receiving. objects are entered and removed from the queues as a result of interactions at the NSAPs.

21.2.2 *Sender*

The objects which may be placed in a queue by a NS user are:

a) activate objects (each representing all parameters contained in an N-GC.ACTIVATE.requestor.submit or N-GC.ACTIVATE.acceptor.deliver primitive);

b) join objects (each representing all parameters contained in a N-GC.JOIN.requestor.submit or a N-GC.JOIN.acceptor.deliver primitive);

c) octets of normal data;

c) indication of end-of-NSDU (completion of a N-GC.DATA.requestor.submit primitive);

d) expedited NSDUs (representing all parameters of a N-GC.DATA.requestor.submit primitive) (for further study);

e) leave objects (each representing all parameters contained in a N-GC.LEAVE.requestor.submit);

f) deactivate objects (each representing all parameters contained in a N-GC.DEACTIVATE.requestor.submit primitive).

The only objects which can be placed in a queue by the NS provider are deactivate objects (representing N-GC.DEACTIVATE.acceptor.deliver primitives and parameters) and leave objects (representing N-GC.LEAVE.acceptor.deliver primitives and parameters.)

A NS user, which attempts to activate a GC, enters an activate object (representing a N-GC.ACTIVATE.requestor.submit) into its send queue associated with the NS provider. The NS user is not allowed to enter any other object into this queue except a deactivate object until after an object representing a N-GC.ACTIVATE.requestor.deliver has been removed from its receive queue.

If the AGI for the group defines a minimum number or set of active NS users then the sending NS user may not enter any object other than a deactivate or leave object into its send queue until after the minimum number of N-GC.JOIN.accepter.deliver objects have been removed from the receive queue.

From the perspective of the sender, a queue relates an ordered set of distinct objects in the following ways:

a) queues are empty before an activate object has been added and can be return to this state with a loss of their contents, by the GC provider under the circumstances described in h) below.

b) objects are added to the queue, subject to the control of the GC provider.

c) a queue has limited capacity, but this capacity is not necessarily either fixed or determinable.

d) the management of the queue capacity shall be such that normal data and end-of-NSDU indications cannot be added to the queue when its addition would prevent the addition of an expedited NSDU object (for further study), leave object, or deactivate object.

In addition the GC provider may manipulate pairs of adjacent objects in the queue to allow:

e) reordering, the order of any pair of objects may be reversed if, and only if, the following object is of a type defined to take precedence over the preceding object. Expedited NSDUs take precedence over octets of normal data and end-of-NSDU indications.

f) deletion, leave and deactivate objects take precedence over any other object. Any object other than a a leave or deactivate object may be deleted by the GC provider if, and only if, the following object is a deactivate or leave object.

21.2.3 *Receiver*

Each receiver participating in a GC maintains a single receive queue. This queue relates the objects from all senders. The use of a single receive queue for each NS user implies that there is no ordering of objects from different senders. The ordering of objects at different receivers may be different due to logical differences in distance from the senders The objects which may be removed from a queue by a NS user are:

a) activate objects (each representing a N-GC.ACTIVATE.acceptor.deliver primitive);

b) join objects (each representing a N-GC.JOIN.acceptor.deliver primitive);

c) notifications (each representing a change in state of the GC as indicated by the NS provider);

d) octets of normal data;

e) indication of end-of-NSDU (completion of a N-GC.DATA.requestor.submit primitive);

f) expedited NSDUs (for further study);

g) leave objects; and

h) deactivate objects.

The only objects which can be placed in a queue by the NS provider are deactivate objects, leave objects, and notification objects.

From the perspective of the receiver a queue relates an ordered set of distinct objects in the following ways:

a) a queue has limited capacity, but this capacity is not necessarily either fixed or determinable.

b) TBD

21.2.4 *GC Activation*

21.2.4.1 *Activate*

A sender queue and receiver queue are established during the activation of a GC.

21.2.4.2 *Join*

A pair of queues is associated with a GC between the NSAP and the provider when the NS provider receives an N-GC.JOIN.requestor.submit request primitive at the NSAP. A join object is entered into one of the queues (the send queue). From the standpoint of the joining NS user, the queues remain associated with the GC until a deactivate or leave object is either entered or removed from the receive queue at that NSAP.

21.2.5 *Data transfer operations*

Flow control on the GC is represented in this queue model by the management of the queue capacities, allowing objects of certain types to be added to the queues. The conditions affecting entry of deactivate and leave objects are described in item a) below and in 21.2.6. The flow control relationship between the other types of objects is summarized in Table 1.

Once in the queue , the NS provider may manipulate pairs of adjacent objects, resulting in

a) change in order; the order of any pair of objects may be reversed if, and only if, the following object is of a type defined to be able to advance ahead of the preceding object. No object is defined to be able to advance ahead of another object of the same type. (Note: Changes in the order of objects may occur due to errors in the processing of the objects.)

b) deletion; any object may be deleted if, and only if, the following object is defined to be destructive with respect to the preceding object. If necessary, the last object in the queue will be deleted to allow a destructive object to be entered. Destructive objects may therefore always be added to the queue. Release and leave objects are defined to be destructive with respect to all other objects.

Table 1-Flow control relationship between objects

The addition of object x may prevent further addition of object y	octets of NS-user-data or end-of-NSDU	Expedited NSDU	Data acknowledgement
octets of normal NS-user-data or end-of-NSDU	Yes	Yes	No
Expedited NSDU (for further study)	No	Yes	No
Data acknowledgement	No	No	No

21.2.6 *GC Deactivation*

21.2.6.1 *Leave*

The insertion into a queue of a leave object, which may occur at any time, represents the initiation of a GC leave procedure. The leave procedure may be destructive with respect to other objects in the two queues and eventually results in the emptying of the queues and the disassociation of the queues with the GC.

The insertion of a leave object may also represent the rejection of a GC join attempt or the failure to activate a GC.

21.2.6.2 *Deactivate*

The insertion of a deactivation object, which may occur at any time, represenTS the initiation of a GC deactivate procedure. The deactivate procedure may be destructive with respect to other objects in all queues associated with the GC and eventually result in the emptying of the queues and the disassociation of the queues with the GC.

22. Quality of Service

The specification of QOS is for further study in conjunction with the work in SC 21/WG 1.

22.1 *Determination of QOS*

The specification of QOS is for further study in conjunction with the work in SC 21/WG 1.

22.2 *Definition of MPDT QOS-parameters*

The specification of QOS is for further study in conjunction with the work in SC 12/WG 1.

23. Sequence of MPDT primitives at one NSAP

This Clause defines the constraints on the sequences in which the primitives defined in Clauses 24 to 26 may occur. The constrainTS determine the order in which primitives occur, but do not fully specify when they may occur. Other constrainTS, such as flow control of data, will affect the ability of a NS user or a NS provider to issue a primitive at any particular time.

Table 2 is a summary of the NS primitives and their parameters.

TABLE 2 - Summary of Network Service primitives and parameters

Phase	Service	Primitive	Parameters
GC activation	GC activate	N-GC.ACTIVATE.requestor. submit	(Called group address, calling address, expedited data selection, QOS-parameter set, AGI-parameter set)
		N-GC.ACTIVATE.acceptor. deliver	(Called group address, calling address, expedited data selection, QOS-parameter set, AGI-parameter set)
	GC.JOIN	N-GC.JOIN requestor.submit	(Called group address, calling address, QOS-parameter set, NS-user-data)
		N-GC.JOIN.acceptor.deliver	(Called address, calling address, QOS-parameter set, NS-user-data)
Notification	Notification	N-GC.NOTE.requestor.submit	(Called group address, calling address, notification parameters)
		N-GC.NOTE.acceptor.deliver	(Called group address, calling address, notification data)
Data transfer	Data transfer	N-GC.DATA.requestor.submit	(Called group address, Calling address, QOS requirements, NS-user-data
		N-GC.DATA.acceptor.deliver	(Called group address, Calling address, QOS requiremenTS, NS-user-data)
	Expedited data transfer (see Note)	N-GC.EXPEDITED—DATA. requestor.submit	(Called group address, Calling address, QOS requiremenTS, NS-user-data)
		N-GC.EXPEDITED—DATA. acceptor.deliver	(Called group address, Calling address, QOS requirements, NS-user-data)
GC deactivate	GC leave	N-GC.LEAVE.requestor.submit	(Called group address, Calling address, Reason, NS-user-data)
		N-GC.LEAVE.acceptor.deliver	(Called group address, Calling address, reason, NS-user-data)
	GC deactivate	N-GC.DEACTIVATE. requestor.submit	
		N-GC.DEACTIVATE. acceptor.deliver	

Note — A NS provider-option service: it may be provided in every Network Service. The provision for an expedited service is for further study.

A primitive issued at one GC end point will, in general, have consequences at the other GC end points. The relations of primitives of each type to primitives at the other GC end poinTS are defined in the appropriate Clauses 24 to 26. However, a N-GC..DEACTIVATE.requestor.submit, N-GC.DEACTIVATE.acceptor.deliver, N-GC.LEAVE.requestor.submit, or N-GC.LEAVE.acceptor.deliver primitive may terminate any of the other sequences before completion.

23.1 *Sequence of primitives at one GC endpoint*

The possible overall sequences of primitives at an GC endpoint are defined in the state transition diagram, Figure 1. In the diagram:

a) a primitive which is not shown as resulting in a transition (from one state to the same state, or from one state to a different state) is not permitted in that state (however, see Clause 23 above concerning the effect of N-GC.DEACTIVATE, and N-GC.LEAVE);

b) T.GC.DEACTIVATE and N-GC.LEAVE stand for either the request or the indication form of the primitive in all cases;

c) the Idle state (state 1) reflects the absence of a GC. It is the initial and final state of any sequence, and upon returning to this state, the GC is deactivated;

d) the use of a state transition diagram to describe the allowable sequences of service primitives does not impose any requirements or constraints on the internal organization of any implementations of the Network Service.

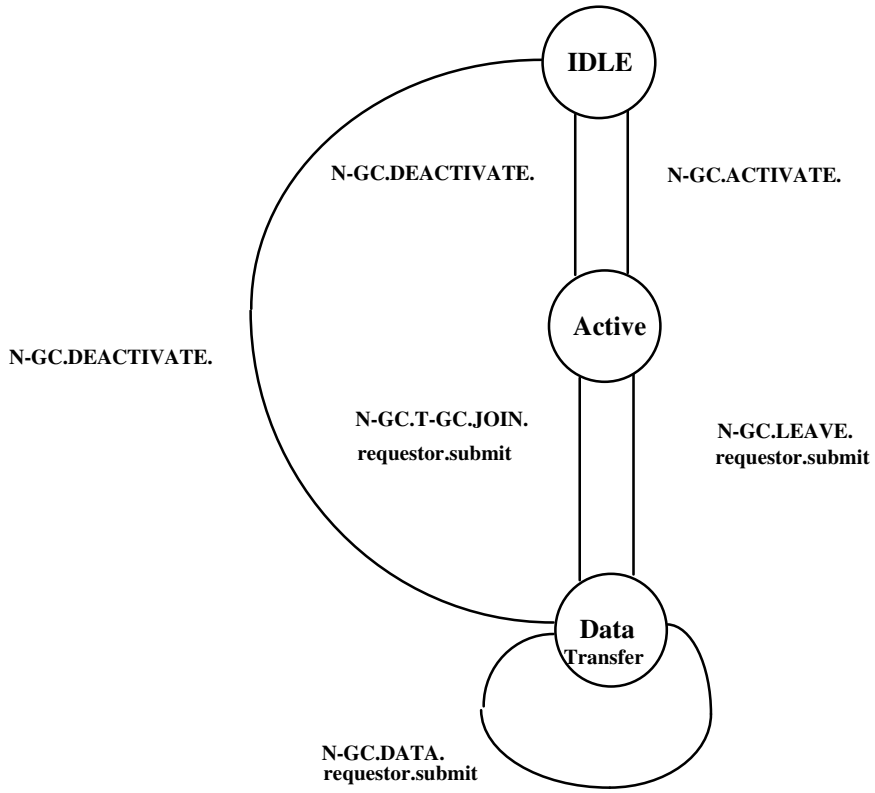

Figure 1 - State Transition Diagram

24 Network group conversation activation phase

24.1 *Activate*

24.1.1 *Function*

The GC activate service primitives are used to initialize a GC.

Simultaneous N-GC.ACTIVATE.requestor.submits at two or more NSAPs are handled independently by the NS provider; they may result in zero, one, or more GCs.

The activation of a group is a necessary precondition before a GC may be used for data transfer. Activation is an end-to-end service where all defined group members receive a notification (N-GC.ACTIVATE.acceptor.deliver) that the group has been activated. As a part of the activation service, information about the AGI and QOS are delivered in the indication.

Figure 2 shows the interaction of SUBMIT and DELIVER primitives.

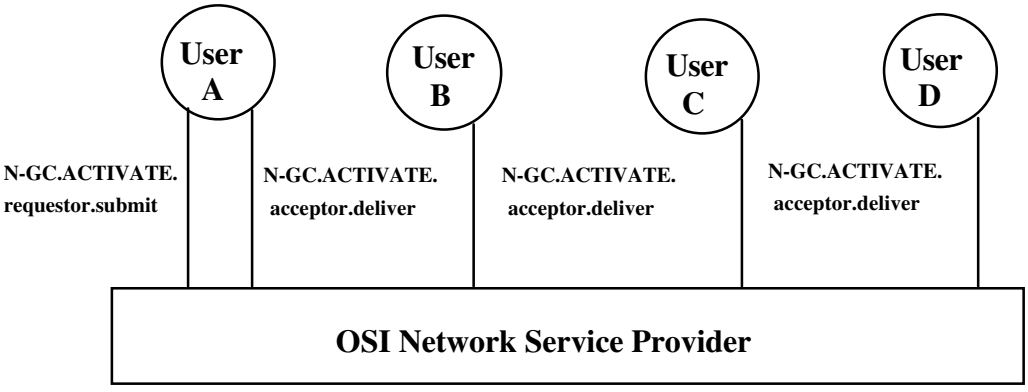

Figure 2 - Interaction of ACTIVATE primitives

24.1.1.1 *Activate Semantics*

When a NS user wishes to begin a new GC among a group of NS users, it initiates an N-GC.ACTIVATE.requestor.submit service request specifying the group address, AGI, QOS, and other parameters as required. The acceptance of the service request by the NS provider establishes the binding between an instance of communication and the group defined by the group address.

The NS provider will establish an identification for the GC so that multiple simultaneous instances of conversations can occur among a group.

The NS provider will deliver a N-GC.ACTIVATE.acceptor.deliver to each NS user which is a member of the group. The indication announces that the GC is available for communication and it contains the parameters as specified in the N-GC.ACTIVATE.requestor.submit primitive and the GC id..

If the NS provider must reject the activate request, it utilizes the GC deactivate primitives as given in Clause 25.

The activation is performed independently of GC join. The NS user activating the GC is in the active state and must wait for the indication that the GC has been successfully activated. If the NS user wishes to be an active member of the GC then it must use the join service as given in 24.2.

24.1.1.2 *Successful Activate*

The activation of a GC is successful only when the initiator of the activate service primitive receives the notifications N-GC.ACTIVATE.acceptor.deliver).

24.1.1.3 *Rejected Activation*

The initiator may receive a N-GC.DEACTIVATE.acceptor.deliver indicating that the activate attempt failed. The reason code may provide additional information about the rejection.

24.1.2 *Types of primitives and parameters*

Table 3 indicates the types of primitives and the parameters needed for GC activation.

24.1.3 *Addresses*

The parameters which take addresses as values (Clauses 24 to 27) all refer to NSAP addresses, lists of NSAP address, or group addresses. The NSAP address parameters will accommodate variable length addresses.

NOTE: The maximum length of the called address field when a list of NSAPs is proposed is for further study.

NOTE: The use of group ids as a side effect of an activate is for further study.

TABLE 3 - GC activation primitives and parameters

Primitive / Parameter	N-GC.ACTIVATE. requestor.submit	N-GC.ACTIVATE. acceptor.deliver
Called group address	X	X(=) (see Note)
Calling address	X (see Note)	X(=)
Expedited data selection	X	X
QOS--parameter set	X	X
AGI parameter set	X	X
GC id		X

Note — This parameter may be implicitly associated with the NSAP at which the primitive is issued.

The values of these addresses as supplied by the NS user are not necessarily checked or authenticated by the NS provider.

24.1.3.1 *Called address parameter*

The called address parameter conveys an address identifying the set of NSAP or group to which the GC is to be established.

24.1.3.2 *Calling address parameter*

The calling address parameter conveys the address of the NSAP from which the GC has been requested.

24.1.3.3 *Expedited data selection parameter*

The use of expedited data is for further study.

24.1.3.4 *QOS-parameter set*

This is for further study.

24.1.3.5 *AGI parameter set*

This is for further study.

24.1.3.6 *Group id*

The Group id parameter provides the identification of the GC within the context of the group address.

24.2 *Join*

24.2.1 *Function*

The join service is used by a NS user that wishes to begin to send and/or receive NS data from a GC. Whereas the activation of a group may span long time periods, the join service is used to bound the time in which a particular NS user is active in the GC.

The join service consists of two primitives: N-GC.JOIN.requestor.submit initiating the join and N-GC-JOIN.acceptor.deliver primitive received by all active GC members announcing the new active GC member..

The NS user wishing to join an active GC issues the N-GC.JOIN.requestor.submit primitive with the appropriate GC identification and addresses. If the NS user does not know the GC id it may issue the primitive with the id set to 0 which will allow the NS user to join all activate GCs for that group.

If the NS provider rejects the join it responds using the leave procedures specified in Clause 26.

It is for further study other means of rejecting the join by participating NS users.

24.2.1.1 Join Semantics

Each NS user which wishes to be active must explicitly join the GC through the use of the join primitives. There are two separate uses of the join primitives:

1. the NS user knows the GC id and explicitly requests to join a particular GC by issuing the N-GC.JOIN.requestor.submit primitive.
2. the NS user does not know the GC id and implicitly joins all active GC on the group address by issuing the N-GC.JOIN.requestor.submit primitive with the GC id set to 0.

The join service is not a confirmed service. The NS user that issues the N-GC.JOIN.requestor.submit primitive does not receive a response indicating that it successfully joined the GC. Rather, until the NS user leaves the GC either by the NS provider or the NS user, or the GC is deactivated, the NS user is able to send and receive data over the GC.

Upon issuing the N-GC.JOIN.requestor.submit primitive the NS user enters the data transfer phase. The NS user may receive one or more notifications specifying the parameters of the GC.

Upon receiving a T-JOIN.acceptor.deliver, a NS user may respond with a notification. The notification will include the parameters associated with the GC and the GC id in use by that NS user. (The cases where notifications are mandatory is related to QOS and AGI and is for further study.)

There are no collision considerations with the join service.

Figure 3 shows the relationship of the join primitives.

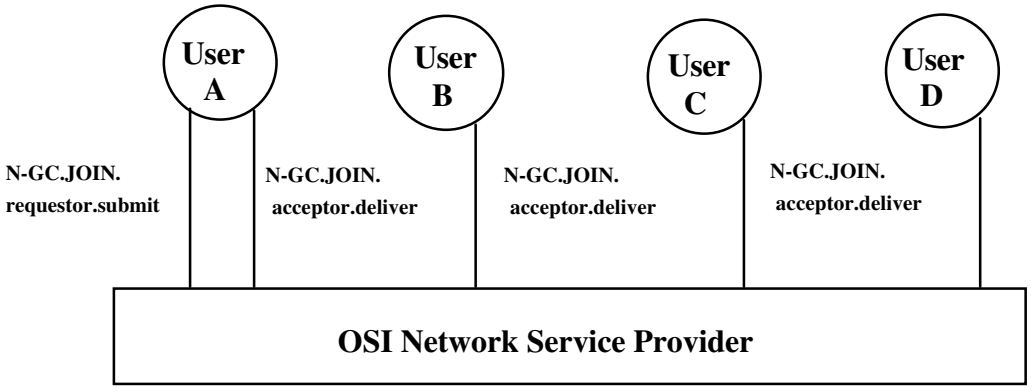

Figure 3 - Join primitives

24.2.2 *Types of primitives and parameters*

Table 4 provides the primitives and parameters associated with the join.

Table 4 - Primitives and parameters associated with the JOIN Service

Primitive / Parameter	N-GC.JOIN. requestor.submit	N-GC.JOIN. acceptor.deliver
Called group address	X	X(=)
Calling address	X (see note)	X
GC id	X	X
NS-user data	X	X

Note:- This parameter may be implicitly associated with the NSAP at which the primitive is issued.

24.2.3 *Addresses*

The parameters which take addresses as values (Clause 24 to 27) all refer to NSAP addresses, lisTS of NSAP address, or group addresses. The NSAP address parameters will accommodate variable length addresses.

NOTE: The maximum length of the called address field when a list of NSAPs is proposed is for further study.

NOTE: The use of group ids as a side effect of an activate is for further study.

The values of these addresses as supplied by the NS user are not necessarily checked or authenticated by the NS provider.

24.2.3.1 *Called address parameter*

The called address parameter conveys an address identifying the set of NSAP or group to which the GC is to be established.

24.2.3.2 *Calling address parameter*

The calling address parameter conveys the address of the NSAP from which the GC has been requested.

24.2.4 *Group id*

The Group id is the unambiguous identification assigned to the GC at activation.

24.2.5 *User Data*

The NS-user-data parameter allows the transfer of NS-user data among NS users, without modification by the NS provider. A NS user invoking a NC-GC join may send any integer number of octets of NS-user-data up to the maximum SDU size selected during GC activation.

25 Group Conversation Release Phase

25.1 *Group Conversation Leave*

25.1.1 *Function*

The GC leave service primitives are used to remove a receiver from a GC. The GC leave may be performed:

a) by any NS users to remove itself from an active GC;

b) by the NS provider to remove a NS user from an active GC.

d) by the NS provider to indicate its inability to join a requested GC.

GC leave is permitted at any time regardless of the current phase of the GC. Once an GC leave procedure has been invoked, the NS user will be removed from the GC; a request for GC leave cannot be rejected. After GC leave has been invoked at one GC endpoint, the NS provider may discard any normal or expedited NS-user-data that has not yet been sent or delivered at that endpoint.

Figure 4 describes the relationship of the primitives.

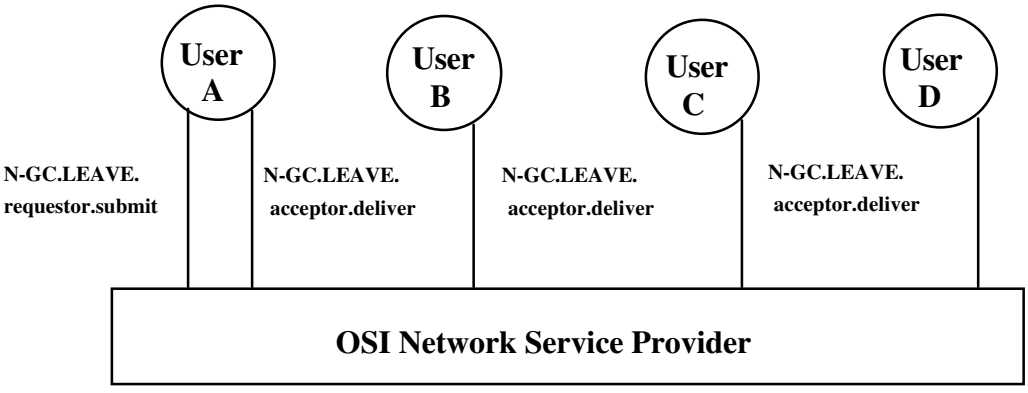

Figure 4 - Leave primitives

25.1.1.1 *Leave Semantics*

The leave service provides the opposite function from the join. NS users may use the leave service to remove itself from the active GC. The leave service is abrupt and potentially destructive. There is no attempt by the NS provider to ensure that queued data is delivered.

25.1.2 *Types of primitive and parameters*

Table 5 indicates the types of primitives and the parameters needed for GC leave.

TABLE 5- N-GC.LEAVE primitives and parameters

Primitive / Parameter	N-GC.LEAVE. requestor.submit	N-GC.LEAVE.acceptor.deliver
Called group address	X	X
Calling address	X	X
Originator	X	X(=)
Group id	X	X(=)
Reason	X(C)	X(C=)
NS-user -data	X(C) (see Note)	X(C=)

Note — This parameter may be implicitly associated with the NSAP at which the primitive is issued.

25.1.3 *Addresses*

The parameters which take addresses as values (Clause 24 to 27) all refer to NSAP addresses, lisTS of NSAP address, or group addresses. The NSAP address parameters will accommodate variable length addresses.

NOTE: The maximum length of the called address field when a list of NSAPs is proposed is for further study.

NOTE: The use of group ids as a side effect of an activate is for further study.

The values of these addresses as supplied by the NS user are not necessarily checked or authenticated by the NS provider.

25.1.3.1 *Called address parameter*

The called address parameter conveys an address identifying the set of NSAP or group to which the GC is to be established.

25.1.3.2 *Calling address parameter*

The calling address parameter conveys the address of the NSAP from which the GC has been requested.

25.1.4 Group id

The Group id is the unambiguous identification assigned to the GC at activation.

25.1.5 *Originator parameter*

The originator parameter indicates the source of the GC leave. INS value indicates either the “NS user”, “NS provider”, or “undefined”.

25.1.6 *Reason parameter*

The reason parameter gives information about the cause of the GC leave. The value conveyed in this parameter will be as follows:

a) When the originator parameter indicates an NS provider invoked leave, the value is one of:
   1) disconnection—permanent condition;
   2) disconnection—transient condition;
   3) join rejection—NSAP address unknown (permanent condition);
   4) join rejection—NSAP unreachable/transient condition;
   5) join rejection—NSAP unreachable/permanent condition;
   6) join rejection—QOS not available/permanent condition;
   7) join rejection—QOS not available/transient condition;
   8) join rejection—reason unspecified/permanent condition;
   9) join rejection—reason unspecified/transient condition;

b) When the originator parameter indicates an NS user invoked leave, the value is one of:
   1) leave—normal condition;
   2) leave—abnormal condition;
   3) join rejection—permanent condition;
   4) join rejection—transient condition;
   5) join rejection—QOS not available/transient condition;
   6) join rejection—QOS not available/permanent condition;
   7) join rejection-incompatible information in NS-user-data.

c) When the originator parameter value is “undefined”, then the value of the reason parameter shall also be “undefined”.

25.1.2.3 *NS-user-data parameter*

The NS-user-data parameter allows the transfer of NS-user-data among NS users, without modification by the NS provider. An NS user invoking N-GC leave may send any integer number of octets of NS-user-data up to the limit imposed by the maximum SDU size selected at GC activation. In an N-GC.LEAVE indication, this parameter can have a non—zero number of octets of NS-user-data only if the originator parameter has the value of “NS user”.

The NS-user-data sent is lost if GC leave is simultaneously invoked by either the NS provider or an intended receiving NS user.

25.2 *Group Conversation Deactivate*

25.2.1 *Function*

The GC deactivate service primitives are used to terminate a GC. The GC deactivate may be performed:

a) by the NS provider,

b) by the initiator of the GC, or

c) by any NS user under control of the AGI. (The deactivation of a GC by any NS user is for further study.)

GC deactivate is permitted at any time regardless of the current phase of the GC. Once a GC deactivate procedure has been invoked, the GC will be deactivated; a request for a GC deactivate cannot be rejected. After GC activate has been invoked at one endpoint, the NS provider may discard any normal or expedited data. Figure 5 describes the relationship of the deactivate primitives.

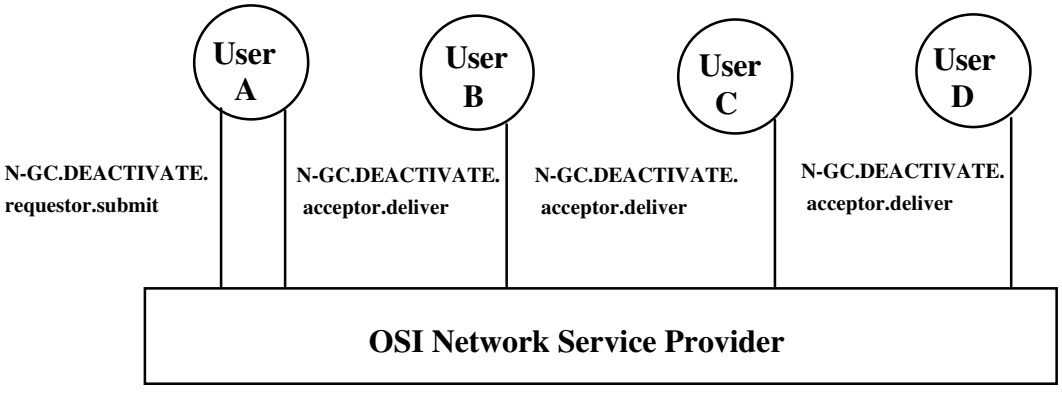

Figure 5 - Deactivate primitives

25.2.2 *Semantics*

The deactivate service destructively terminates the GC. All in transit data is lost. The state of the entities active in the GC is indeterminable.

25.2.3 *Types of primitives and parameters*

Table 7 provides the primitives and parameters associated with the deactivate service.

Table 7-Deactivate primitives and parameters

Primitive / Parameter	T.GC.DEACTIVATE request	T.GC.DEACTIVATE indication
Called group address	X	X(=)
Calling address	X	X(=)
Group id	X	X(=)
Originator	X	X
Reason	X	X

25.2.3.1 *Addresses*

The parameters which take addresses as values (Clause 24 to 27) all refer to NSAP addresses, lisTS of NSAP address, or group addresses. The NSAP address parameters will accommodate variable length addresses.

NOTE: The maximum length of the called address field when a list of NSAPs is proposed is for further study.

NOTE: The use of group ids as a side effect of an activate is for further study.

The values of these addresses as supplied by the NS user are not necessarily checked or authenticated by the NS provider.

25.2.3.1.1 *Called address parameter*

The called address parameter conveys an address identifying the set of NSAP or group to which the GC is to be established.

25.2.3.1.2 Calling address parameter

The calling address parameter conveys the address of the NSAP from which the GC has been requested.

25.2.3.2 *Group id*

The Group id is the unambiguous identification assigned to the GC at activation.

25.2.3.3 *Originator parameter*

The originator parameter indicates the source of the GC leave. INS value indicates either the “NS user”, “NS provider”, or “undefined”.

Note — The value “undefined” is not permitted when an N-GC.DEACTIVATE.acceptor.deliver is issued by an NS user or an NS provider in order to reject an GC join attempt.

25.2.3.4 *Reason parameter*

The reason parameter gives information about the cause of the GC deactivate.

26 Data transfer phase

26.1 *Data transfer*

26.1.1 *Function*

The data transfer service primitives provide for an exchange of NS-user-data called Network-Service-data-units (NSDUs), in either direction or in both directions simultaneously on a network GC. The Network Service preserves both the sequence and the boundaries of the NSDUs.

Note — Designers of higher layer protocols using the Network Service should realize that the requested QOS applies to complete NSDUs, and that divisions of available NS—user—data into small NSDUs may have cost implications because of the impact on cost optimization mechanisms operated by the NS provider. Figure 6 presents the data transfer primitives.

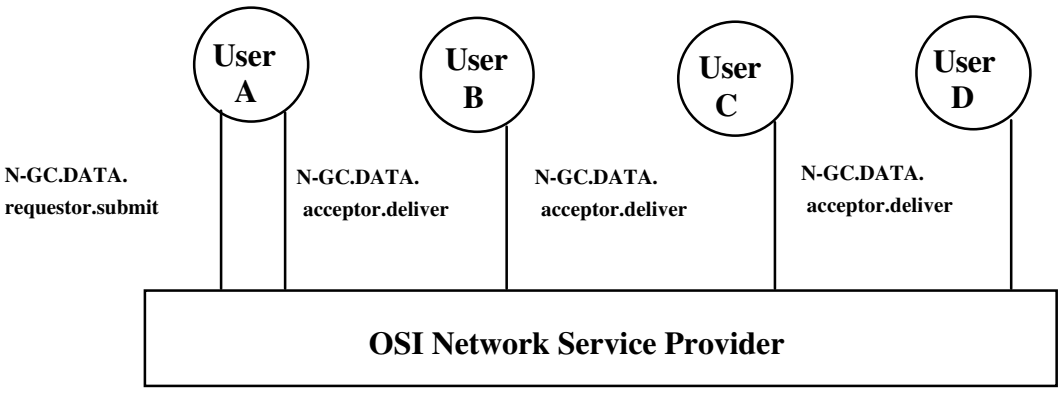

Figure 6 - Data transfer primitives

26.1.2 *Semantics*

The semantics of the data transfer service is the transfer of octets from one NS user to all other NS users active in the GC. The relationship of one transfer to another is determined by the QOS. The reliability of the transfer is determined by the QOS and the type of service selected. The NS service makes no guarantees about the order or success of any individual transfer within the allowable range of the selected QOS.

Missing or misordered NSDUS may be permitted if the QOS is selected to allow such events. With no reliability selected, there is no relationship between individual invocations of the data transfer service.

26.1.2 *Types of primitives and parameters*

Table 8 indicates the types of primitives and the parameters needed for data transfer.

TABLE 8 - Data transfer primitives and parameters

Primitive / Parameter	N-GC.DATA.requestor. submit	N-GC.DATA.acceptor. deliver
Called group address	X	X(=)
Calling address	X	X(=)
Group id	X	X(=)
QOS	X	X(=)
NS—user—data	X	X(=)

26.1.2.1 *Addresses*

The parameters which take addresses as values (Clause 24 to 27) all refer to NSAP addresses, lisTS of NSAP address, or group addresses. The NSAP address parameters will accommodate variable length addresses.

NOTE: The maximum length of the called address field when a list of NSAPs is proposed is for further study.
NOTE: The use of group ids as a side effect of an activate is for further study.

The values of these addresses as supplied by the NS user are not necessarily checked or authenticated by the NS provider.

26.1.3.1.1 *Called address parameter*

The called address parameter conveys an address identifying the set of NSAP or group

26.1.3.1.2 *Calling address parameter*

The calling address parameter conveys the address of the NSAP from which the N-GC.DATA request was made.

26.1.3.2 *Group id*

The group id parameter specifies the id of the GC as provided during GC activation.

26.1.3.3 *NS-user-data parameter*

The NS-user-data parameter allows the transfer of an NSDU among NS users, without modification by the NS provider. The NS user may send any integer number of octets, one or greater, of NS-user-data that forms the NSDU.

26.1.3 *Sequence of primitives*

The operation of the Network Service in transferring NSDUs can be modeled as a queue of unknown size within the NS provider (see Clause 21). The ability of an NS user to issue an N-GC.DATA.requestor.submit or of the NS provider to issue an N-GC.DATA.acceptor.deliver depends on the behavior of the receiving NS users and the resulting state of the queues.

27 Notification

27.1 *Function*

The notification service provides a means of notifying NS users of evenTS relating to the operation of a GC.

The service consisTS of two primitives: N-GC.NOTE.requestor.submit and N-GC.NOTE.acceptor.deliver.

27.2 *Semantics*

The notification service is used by NS users and the NS provider to notify other NS users of evenTS that effect the operation of the GC. In particular, the notification service is used to indicate the joining of new NS users, leaving of NS users, and errors detected by the NS provider. Each notification service invocation is independent of all others. The information contained in the notification service cannot be related to other notification service requesTS of delivers by either source or time.

Since there is no dependence between notifications, there are no collision considerations.

Figure 7 shows the sequence of primitives for the notification service.

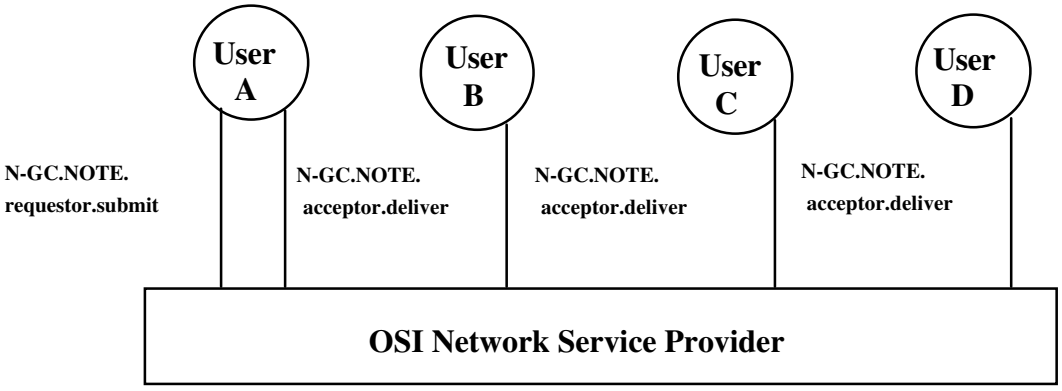

Figure 7 - Notification Service Primitives

27.3 *Types of primitives and parameters*

Table 9 provides the primitives and parameters associated with the notification.

TABLE 9 - Note primitives and parameters

Primitive / Parameter	N-GC.NOTE.requestor. submit	N-GC.NOTE.acceptor. deliver
Called group address	X	X(=)
Calling address	X	X(=)
Group id	X	X(=)
QOS	X	X(=)
NS—notification—data	X	X(=)

27.1.2.1 *Addresses*

The parameters which take addresses as values (Clauses 24 to 27) all refer to NSAP addresses, lisTS of NSAP address, or group addresses. The NSAP address parameters will accommodate variable length addresses.

NOTE: The maximum length of the called address field when a list of NSAPs is proposed is for further study.
NOTE: The use of group ids as a side effect of an activate is for further study.

The values of these addresses as supplied by the NS user are not necessarily checked or authenticated by the NS provider.

27.1.2.1.1 *Called address parameter*

The called address parameter conveys an address identifying the set of NSAP or group

27.1.2.1.2 *Calling address parameter*

The calling address parameter conveys the address of the NSAP from which the N-GC.NOTE request was made. When the N-GC.NOTE.acceptor.deliver is the result of a NS provider action, this parameter is set to 0 (NULL).

27.1.2.2 *Group id*

The group id parameter specifies the id of the GC as provided during GC activation.

26.1.2.3 *NS-user-notification parameter*

The NS-user-notification parameter allows the transfer of notification data among NS users, The NS user may send any integer number of octets, one or greater, of NS-user-data that forms the NSDU.

The values for the notification data is for further study.