System Specifications

Communication

Network Layer

3

3

3

Summary

This document specifies Network Layer of the KNX System.

Version 01.01.02 is a KNX Approved Standard.

This document is part of the KNX Specifications v2.1.

Document updates

Version	Date	Modifications
1.0	2001.12.17	Preparation of the Approved Standard.
1.1	2007.08.06	• S16 “System Broadcast Services” v1.0 AS
		• AN047 “System Broadcast Services in media coupler” v03
1.1	2008.08.28	• AN102 “TP1 Bridges” integrated.
1.1	2008.10.28	Editorial update. • clause 2.4: update of hexadecimal notations • clause 2.4: integration of some text from 9/3 “Couplers” in which the routing algorithm is removed.
1.1	2008.12.19	Preparation of the Approved Standard v1.1.
1.01.01	2011.08.03	Editorial corrections.
01.01.02	2013.10.28	Editorial updates for the publication of KNX Specifications 2.1.

References

[01] Chapter 3/3/2 “Data Link Layer – General Requirements”
[02] Chapter 3/3/4 “Transport Layer”
[03] Chapter 3/3/7 “Application Layer”

Filename:	03_03_03 Network Layer v01.01.02 AS.docx
Version:	01.01.02
Status:	Approved Standard
Savedate:	2013.10.28
Number of pages:	16

Contents

1 Overview 4

2 Network services and protocol 6

2.1 NPDU 6

2.2 Network Layer services 6

2.2.1 N_Data_Individual service 6

2.2.2 N_Data_Group service 7

2.2.3 N_Data_Broadcast service 8

2.2.4 N_Data_SystemBroadcast service 9

2.3 Parameters of Network Layer 10

2.4 Network Layer State Machines 11

2.4.1 State Machine of Network Layer for normal devices 11

2.4.2 State Machines for Couplers 11

1 Overview

The Network Layer shall be the layer between the Data Link Layer and the Network Layer User. This Network Layer shall conform to the definitions of the ISO/OSI model (ISO 7498) Network Layer.

The Network Layer shall use the L_Data - and the L_SystemBroadcast services of the Data Link Layer and shall provide N_Data_Individual, N_Data_Group, N_Data_Broadcast and N_Data_SystemBroadcast services to the Network Layer user, see Figure 1.

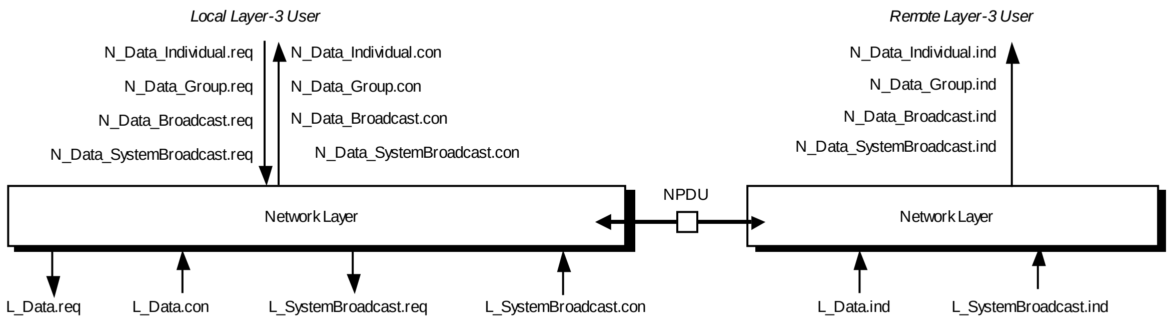

Figure 1 - Interactivity of the Network Layer (not for Bridges or Routers)

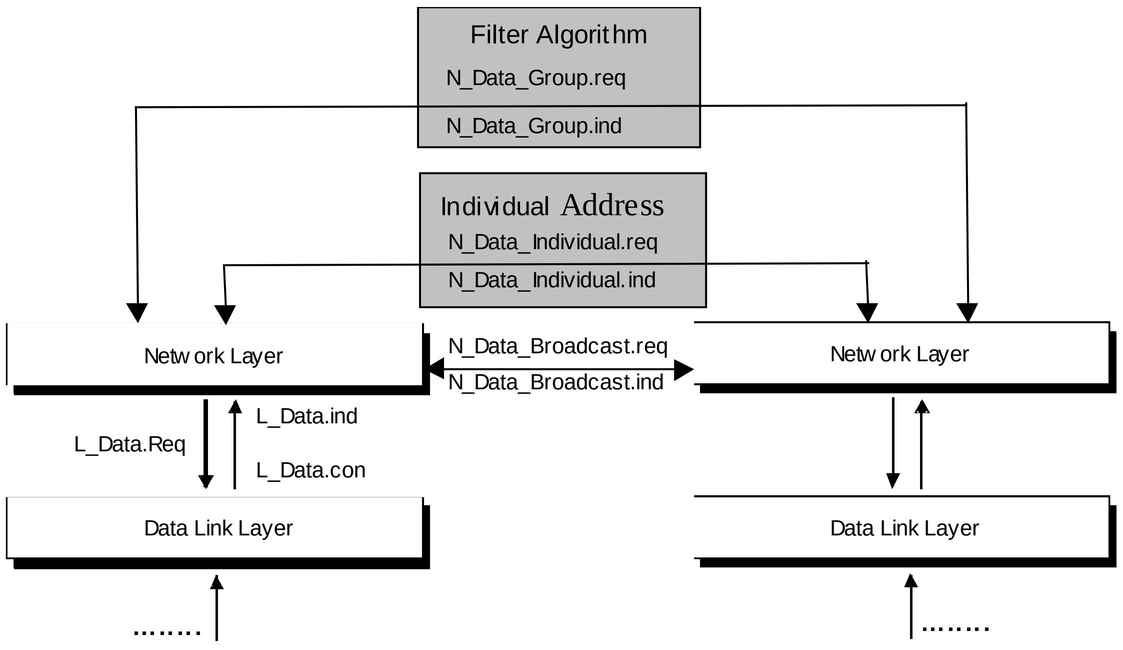

Figure 2 - General Functionality of a Router or Bridge

Communication across Subnetworks needs devices called Routers, see clause 2.4.3 “State Machine of Network Layer for Routers”. Routers are devices that allow to couple two Data Link Layer protocol instances together, which are connected to different Subnetworks. For routing frames from one Subnetwork to the other the Router uses a filter algorithm. Furthermore a Router allows removing misrouted messages so that they cannot flood the network. All the filter algorithms of a KNX network together define the allowed communication paths between any two devices.

Communication across Subnetworks without filter characteristics needs devices called Bridges, see also Figure 2. Like a Router a Bridge allows to couple two Data Link Layer protocol instances together, which are connected to different Subnetworks. But a Bridge does not have the filter property of a Router and therefore does not need any filter algorithm.

Two different mechanisms for routing are used. For Group Addressing a filter algorithm is used. For point-to-point addressing the routing is done by interpreting the Individual Address.

Two different Network Layer users must be distinguished:

1. The Network Layer user in a standard device: This is the Transport Layer, see [02].
2. The Network Layer user in a Router: this is the filter algorithm.

2 Network services and protocol

2.1 NPDU

The NPDU shall correspond to the LPDU of an L_Data-Frame without the Control Field, Source Address, Destination Address, Address Type Flag and the octet count. The NPDU is shown in Figure 3 below.

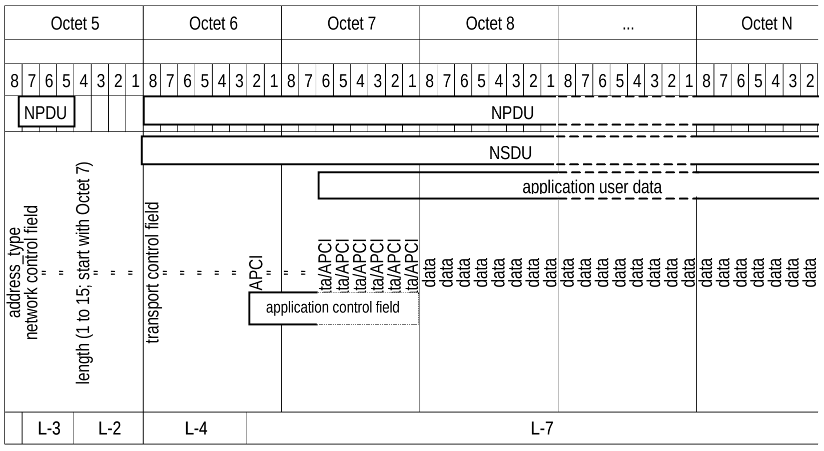

Figure 3 - Format of the NPDU

2.2 Network Layer services

2.2.1 N_Data_Individual service

The local user of Network Layer shall prepare an NSDU for the remote user of Network Layer; the destination shall be addressed with an Individual Address. The local user of Network Layer shall apply the N_Data_Individual.req primitive to pass the NSDU to the local Network Layer. The local Network Layer shall accept the service request and shall pass it with an L_Data.req with address_type = ‘individual’ to the local Data Link Layer.

The local Network Layer shall encode the NSDU to the NPDU by adding the hop_count with the value according to the parameter hop_count_type and mapping the arguments ack_request, destination_address, octet_count and priority, and to the corresponding arguments ack_request, destination_address, octet_count and priority of the L_Data.req primitive.

The remote Network Layer shall map an L_Data.ind primitive with address_type = ’individual’ to an N_Data_Individual.ind primitive. It shall remove the hop_count and shall generate the parameter hop_count_type according to its value. The argument lsdu shall be decoded to the argument nsdu. The arguments octet_count, priority and source_address shall be mapped to the corresponding arguments octet_count, priority and source_address of the N_Data_Individual.ind primitive.

The local Network Layer shall map the L_Data.con primitive to the N_Data_Individual.con primitive. The argument l_status shall be mapped to the corresponding argument n_status of the N_Data_Individual.con primitive.

N_Data_Individual.req(ack_request, destination_address, hop_count_type, octet_count, priority, nsdu)

ack_request:	Data Link Layer acknowledge requested or don't care
destination_address:	Individual Address of the destination
hop_count_type:	hop count 7 or Network Layer Parameter
octet_count:	length information as described in Data Link Layer
priority:	system, urgent, normal or low priority
nsdu:	this is the user data to be transferred by the Network Layer

N_Data_Individual.con(ack_request, destination_address, hop_count_type, octet_count, priority, nsdu, n_status)

ack_request:		Data Link Layer acknowledge requested or don't care
destination_address:		Individual Address of the destination
hop_count_type:		hop count 7 or Network Layer Parameter
octet_count:		length information as described in Data Link Layer
priority:		system, urgent, normal or low priority
nsdu:		this is the user data that has been transferred by Network Layer
n_status:	ok:	N_Data_Individual sent successfully with L_Data
	not_ok:	transmission of the associated L_Data request frame did not succeed

N_Data_Individual.ind(destination_address, hop_count_type, octet_count, priority, source_address, nsdu)

destination_address:	the Individual Address of this device
hop_count_type:	hop count equals 7 or not
octet_count:	length information as described in Data Link Layer
priority:	system, urgent, normal or low priority
source_address:	Individual Address of the device that requested the N_Data_Individual service
nsdu:	this is the user data that has been transferred by Network Layer

2.2.2 N_Data_Group service

The N_Data_Group service shall be confirmed locally. The local user of Network Layer shall prepare an NSDU for the remote user of Network Layer; the destination shall be addressed with a Group Address. The local user of Network Layer shall apply the N_Data_Group.req primitive to pass the NSDU to the local Network Layer. The local Network Layer shall accept the service request and shall pass it with an L_Data.req with address_type = 'multicast' to the local Data Link Layer.

The local Network Layer shall encode the NSDU to the LSDU by adding the hop_count with the value according to the parameter hop_count_type and mapping the arguments ack_request, destination_address, octet_count and priority to the corresponding arguments ack_request, destination_address, octet_count and priority of the L_Data.req primitive.

The remote Network Layer shall map an L_Data.ind primitive with address_type = 'multicast' and destination_address<>'0' to an N_Data_Group.ind primitive. It shall remove the hop_count and generates the parameter hop_count_type according to its value. The arguments destination_address, octet_count and priority are mapped to the corresponding arguments destination_address, octet_count and priority of the N_Data_Group.ind primitive.

The local Network Layer maps the L_Data.con primitive to the N_Data_Group.con primitive. The argument l_status is mapped to the corresponding argument n_status of the N_Data_Group.con primitive.

N_Data_Group.req(ack_request, destination_address, hop_count_type, octet_count, priority, nsdu)

ack_request:	Data Link Layer acknowledge requested or not
destination_address:	Group Address of the destination
hop_count_type:	hop count 7 or Network Layer Parameter
octet_count:	length information as described in Data Link Layer
priority:	system, urgent, normal or low priority
nsdu:	this is the user data to be transferred by Network Layer

N_Data_Group.con(ack_request, destination_address, hop_count_type, octet_count, priority, nsdu, n_status)

ack_request:		Data Link Layer acknwoledge requested or not
destination_address:		Group Address of the destination
hop_count_type:		hop count 7 or Network Layer Parameter
octet_count:		length information as described in Data Link Layer
priority:		system, urgent, normal or low priority
nsdu:		this is the user data that has been transferred by Network Layer
n_status:	ok:	N_Data_Group sent successfully with L_Data service
	not_ok:	transmission of the associated L_Data request frame did not succeed

N_Data_Group.ind(destination_address, hop_count_type, octet_count, priority, nsdu)

destination_address:	the addressed Group Address of this device
hop_count_type:	hop count equals 7 or not
octet_count:	length information as described in Data Link Layer
priority:	system, urgent, normal or low priority
nsdu:	this is the user data that has been transferred by Network Layer

2.2.3 N_Data_Broadcast service

The local user of Network Layer shall prepare an NSDU for all the remote Network Layer users within the same domain, the destination shall be addressed with the Broadcast Address (Destination Address = ´0´ and address_type = ´multicast´). The local user of Network Layer shall apply the N_Data_Broadcast.req primitive to pass the NSDU to the local Network Layer. The local Network Layer shall accept the service request and shall pass it with an L_Data.req with address_type = ´ multicast ´ to the local Data Link Layer.

The local Network Layer shall encode the NSDU to the LSDU by adding the hop_count with the value according to the parameter hop_count_type and mapping the arguments ack_request octet_count and priority to the corresponding arguments ack_request, octet_count and priority of the L_Data.req primitive and setting the L_Data.req parameter destination_address to ‘0’.

The remote Network Layer shall map an L_Data.ind primitive with address_type = ´multicast´ and destination_address = ´0´ to an N_Data_Broadcast.ind primitive. It shall remove the hop_count and shall generate the parameter hop_count_type according to its value. The argument lsdu shall be mapped to the argument nsdu. The argument priority shall be mapped to the corresponding argument priority of the N_Data_Broadcast.ind primitive.

The local Network Layer shall map the L_Data.con primitive to the N_Data_Broadcast.con primitive. The argument l_status shall be mapped to the corresponding argument n_status of the N_Data_Broadcast.con primitive.

N_Data_Broadcast.req(ack_request, hop_count_type, octet_count, priority, nsdu)

ack_request:	Data Link Layer acknowledge requested or not
hop_count_type:	hop count 7 or Network Layer Parameter
octet_count:	length information as described in Data Link Layer
priority:	system, urgent, normal or low priority
nsdu:	this is the user data to be transferred by Network Layer

N_Data_Broadcast.con(ack_request, hop_count_type, octet_count, priority, nsdu, n_status)

ack_request:		Data Link Layer acknowledge requested or not
hop_count_type:		hop count 7 or Network Layer Parameter
octet_count:		length information as described in Data Link Layer
priority:		system, urgent, normal or low priority
nsdu:		this is the user data that has been transferred by Network Layer
n_status:	ok:	N_Data_Broadcast sent successfully with L_Data service
	not_ok:	transmission of the associated L_Data request frame did not succeed

N_Data_Broadcast.ind(hop_count_type, octet_count, priority, source_address, nsdu)

hop_count_type:	hop count equals 7 or not
octet_count:	length information as described in Data Link Layer
priority:	system, urgent, normal or low priority
source_address:	Individual Address of the device that requested the N_Data_Broadcast service
nsdu:	this is the user data that has been transferred by Network Layer

2.2.4 N_Data_SystemBroadcast service

The local user of Network Layer shall prepare an NSDU for all the remote Network Layer users; the destination shall be addressed with the system broadcast address (Destination Address = 0000h and address_type = “multicast”). The local user of Network Layer shall apply the N_Data_System-Broadcast.req primitive to pass the NSDU to the local Network Layer. The local Network Layer shall accept the service request and shall pass it with an L_SystemBroadcast.req with address_type = “multicast” to the local Data Link Layer.

The local Network Layer shall encode the NSDU to the LSDU by adding the hop_count with the value according to the parameter hop_count_type and mapping the arguments ack_request octet_count and priority to the corresponding arguments ack_request, octet_count and priority of the L_System-Broadcast.req primitive and setting the L_SystemBroadcast.req parameter destination_address to 0000h.

The remote Network Layer shall map an L_SystemBroadcast.ind primitive with address_type = “multicast” and destination_address = 0000h to an N_Data_SystemBroadcast.ind primitive. It shall remove the hop_count and generate the parameter hop_count_type according to its value. The argument lsdu shall be mapped to the argument nsdu. The argument priority shall be mapped to the corresponding argument priority of the N_Data_SystemBroadcast.ind primitive.

The local Network Layer shall map the L_SystemBroadcast.con primitive to the N_Data_SystemBroadcast.con primitive. The argument l_status shall be mapped to the corresponding argument n_status of the N_Data_SystemBroadcast.con primitive.

N_Data_SystemBroadcast.req(ack_request, hop_count_type, nsdu, octet_count, priority)

ack_request:	This parameter shall be used to indicate whether a Data Link Layer acknowledge is mandatory or optional.
hop_count_type:	This parameter shall be used to indicate whether the hop_count shall be set to 7 or if the Network Layer parameter shall be used.
nsdu:	This parameter shall be used to contain the user data that shall be transferred by the Network Layer.
octet_count:	This parameter shall be used to indicate the length information of the requested frame.
priority:	This parameter shall be used to indicate the priority that shall be used to transmit the requested frame; it shall be “system”, “urgent”, “normal” or “low”.

N_Data_SystemBroadcast.con(ack_request, hop_count_type, nsdu, octet_count, priority, n_status)

ack_request:	This parameter shall be used to indicate whether a Data Link Layer acknowledge is indicated as mandatory or optional in the transmitted frame.	
hop_count_type:	This parameter shall be used to indicate whether the hop_count of the transmitted frame is set to 7 or if the Network Layer parameter is used.	
nsdu:	This parameter shall be used to contain the user data that is transferred by the Network Layer.	
octet_count:	This parameter shall be used to indicate the length information of the transmitted frame.	
priority:	This parameter shall be used to indicate the priority that is used to transmit the requested frame; it shall be “system”, “urgent”, “normal” or “low”.	
n_status:	ok:	This value of this parameter shall be used to indicate that the transmission of the N_Data_SystemBroadcast is successful.
	not_ok:	This value of this parameter shall be used to indicate that the transmission of the N_Data_SystemBroadcast.req did not succeed.

N_Data_SystemBroadcast.ind(hop_count_type, nsdu, octet_count, priority, source_address)

hop_count_type:	This parameter shall be used to indicate whether the hop count of the received frame equals 7 or not.
nsdu:	This parameter shall be used to contain the user data that is received by the Network Layer.
octet_count:	This parameter shall be used to contain the length information of the received frame.
priority:	This parameter shall be used to indicate the priority of the received frame; it shall be “system”, “urgent”, “normal” or “low”.
source_address:	This parameter shall be used to indicate the Source Address of the received frame; it shall be the Individual Address of the device that has transmitted the N_Data_SystemBroadcast-PDU.

2.3 Parameters of Network Layer

The following parameters influence the behaviour of Network Layer and are required inside Network Layer in order to operate correctly:

hop_count	will be added to the frame by Network Layer
device_type	information about the device: either normal device or Bridge or Router.

2.4 Network Layer State Machines

2.4.1 State Machine of Network Layer for normal devices

The state machine of Network Layer for normal devices shall map services as described in clause 2.2. The value of the hop count shall be added when the Transport Layer applies a Network Layer request primitive.

In sending direction, the Network Layer shall set the value of the hop_count field in the NSDU to the value according to the value service primitive parameter ‘hop_count_type’ of the request service primitive:

• hop_count_type = ‘7’	the hop_count shall be set to the value 7
• hop_count_type = ‘Network Layer Parameter’	the hop_count shall be set to the value of the Network Layer parameter ‘hop_count’ (see clause 2.3 “Parameters of Network Layer”)

In reception direction, the Network Layer shall set the service parameter hop_count_type of the indication- and confirmation service primitives according to the value of the hop_count field in the received NSDU:

• hop_count = 7	the hop_count_type shall be set to ‘equal to 7’
• hop_count ≠ 7	the hop_count_type shall be set to ‘not equal to 7’

2.4.2 State Machines for Couplers

2.4.2.1 General requirements

2.4.2.1.1 Introduction

Please refer to [01] for the overview of the Coupler types.

Couplers (KNX Router, KNX TP1 Bridge and KNX TP1 repeater) do also have a Network Layer but their Network Layer state machine differs from normal devices.

2.4.2.1.2 Busload requirements

- The Coupler shall be able to route telegrams in the communication mode
  - point-to-multipoint, connectionless and
  - point-to-point connectionless and –connection-oriented

  if the busload at the primary side and the secondary side (without the Coupler) is lower than or equal to 90 % (to be confirmed). If the bus load exceeds this value on the primary side and the secondary side, it may acknowledge with BUSY any incoming telegram (up to 6 times).
- In case the Coupler is a TP1 Bridge or a TP1 Repeater it shall be possible to reduce the number of repetitions from the primary side to the secondary side and vice versa for telegrams in the communication mode
  - point-to-multipoint, connectionless and
  - point-to-point connectionless and –connection-oriented.
- In case of power down at the primary side, it is recommended that according to parameterization the Coupler generates an error message.

2.4.2.2 State Machine of Network Layer for a TP1 Repeater

If an L_Data.ind with a hop_count in [1 … 6] is received, the TP1 Repeater shall decrement the hop_count and transmit the service parameters of the L_Data.ind with the corresponding service parameters (source address, destination_address, address_type, priority, ack_request, octet_count, lsdu) of an L_Data.req to the other side.

If an L_Data.ind with a hop_count value of seven is received, the TP1 Repeater shall transmit the service parameters of the L_Data.ind with the corresponding service parameters of an L_Data.req to the other side.

Otherwise the Network Layer of the TP1 Repeater shall discard the L_Data.ind.

2.4.2.3 State Machine of Network Layer for a TP1 Bridge

If an L_Data.ind with address_type = ´multicast´ (Group Addressed, LTE-addressed) is received on one side, the TP1 Bridge shall transmit the service parameters of the L_Data.ind with the corresponding service parameters of an L_Data.req to the other side. Any hop count in [0 … 7] shall remain unchanged.

If an L_Data.ind with address_type = ‘multicast’ and Destination Address = 0000h is received on one side, the TP1 Bridge shall process the L_Data.ind in its local stack and shall transmit the service parameters of the L_Data.ind with the corresponding service parameters of an L_Data.req to the other side. Any hop count in [0 … 7] shall remain unchanged.

If an L_Data.ind with address_type = ´individual´ and Destination Address equal to the Individual Address of the TP1 Bridge is received, the TP1 Bridge shall process the L_Data.ind identical to a normal device, see clause 2.2.1.

If an L_Data.ind with address_type = ‘individual’ and Destination Address not equal to the Individual Address of the Bridge is received on one side, the TP1 Bridge shall transmit the service parameters of the L_Data.ind with the corresponding service parameters of a L_Data.req to the other side. Any hop count in [0 … 7] shall remain unchanged.

Otherwise the Network Layer of the TP1 Bridge shall discard the L_Data.ind.

2.4.2.4 State Machine of Network Layer for Routers

If an L_Data.ind with address_type = ´multicast´ and hop_count in [1...6] is received and the filter condition for the Destination Address is true, the Router shall decrement the hop_count and shall transmit the service parameters of the L_Data.ind with the corresponding service parameters of a L_Data.req to the other side.

If an L_Data.ind with address_type = ´multicast´ and hop_count equal to seven is received, the Router shall transmit the service parameters of the L_Data.ind with the corresponding service parameters of an L_Data.req to the other side.

If an L_Data.ind with address_type = ´individual´ and Destination Address equal to the Individual Address of the Router is received, the Router shall process the L_Data.ind identical to a normal device, see clause 2.2.1.

If an L_Data.ind with address_type = ‘individual’ is received and the Destination Address matches the conditions for routing, the Router shall transmit the service parameters of the L_Data.ind with the corresponding service parameters of an L_Data.req to the other side. Additionally, if the hop counter value was in [1 … 6] the Router shall decrement it.

Otherwise the Network layer of the Router shall discard the L_Data.ind.

An N_Data_Individual.req service invoked by the Network Layer user at the Router shall be processed as described in clause 2.2.1.

More detailed Routing Algorithm

For a Router there are five possible courses of action in response to a received LSDU.

- ROUTE_UNMODIFIED

  The LSDU shall be routed from one Subnetwork (on the primary side or the secondary side) to the second Subnetwork side (on the secondary side or the primary side) without modification of the hop count value. The LSDU shall only be routed to the second Subnetwork after it has been positively acknowledged on the first Subnetwork.

- ROUTE_DECREMENTED

  The LSDU shall be routed from one Subnetwork (on the primary side or the secondary side) to the second Subnetwork side (on the secondary side or the primary side) after the hop count value is decremented. The LSDU shall only be routed to the second Subnetwork after it has been positively acknowledged on the first Subnetwork.

- ROUTE_LAST

  The LSDU shall be routed from the first Subnetwork to the second Subnetwork and the hop count value shall be set to zero. The telegram shall be acknowledged according the Data Link Layer specifications of the medium of the first Subnetwork and then the LSDU shall be routed to the second Subnetwork.

- FORWARD_LOCALLY

  The LSDU shall be processed to an NSDU and given to the local Network Layer User after it has been acknowledged positively on the Subnetwork from which it has arrived.

- IGNORE_TOTALLY

  The LSDU shall be ignored; no acknowledgment shall be sent back to the originator of the LSDU.

- IGNORE_ACKED

  The LSDU shall be ignored, nonetheless it shall be acknowledged on the Subnetwork from which it has arrived.

It shall be possible to parameterize the Router in such a way, that independent of the communication mode and the hop count the Router always acknowledges any received telegram.

It shall be possible to at least route Group Addresses from Main Group 0 to 13 individually and from Main Group 14 and 15 either generally or individually.

A Router shall be a Line Coupler or a Backbone Coupler. This shall depend on its position in the topology. This shall be reflected in the value of the Individual Address of the device. (The Individual Address is specified in [01].)

♦ If used as Backbone Coupler

– Device Address:	shall be 0
– Line Address:	shall be 0
– Area Address:	shall differ from 0

The Backbone Coupler shall separate the Backbone Line from the Main Line.

♦ If used as Line Coupler

– Device Address:	shall be 0
– Line Address:	shall differ from 0
– Area Address:	shall differ from 0

The following clauses specify the routing algorithm for a Router, which can either be a Line Coupler or a Backbone Coupler, depending on his position in the topology.

Abbreviations:

C	hop count value contained in the N-protocol header
D	low order octet of the Destination Address, i.e. Device Address part
G	Group Address
SD	low nibble of high order octet plus low order octet, i.e. Line Address + Device Address
Z	high nibble of high order octet of the Destination Address, i.e. Area Address
ZS	high order octet of the Destination Address, i.e. hierarchy information part: Area Address + Line Address

2.4.2.4.1 Routing in case of a Group Destination Address

Depending on parameter setting of the Router Filter Table, telegrams on point-to-multipoint connectionless communication mode with standard Group Addresses, shall be routed or blocked from the primary side to the secondary side and vice versa. The routing can be summarized as follows.

```
if      routing condition = TRUE and 0h < C < 7h then ROUTE_DECREMENTED
if      routing condition = TRUE and C = 0h then IGNORE_ACKED [1]
elsif   C = 7h    then ROUTE_UNMODIFIED
else    IGNORE_TOTALLY
```

The above applies regardless whether the Coupler is used as a Backbone Coupler or a Line Coupler.

For these telegrams on point-to-multipoint connectionless communication mode with standard Group Addresses it shall via parameterization of the Coupler be possible to:

- generally block routing,
- generally route or
- route according to the Filter Table.

2.4.2.4.2 Routing in case of an Individual Destination Address: Line Coupler

Depending on parameterization of the Line Coupler, telegrams on point-to-point connectionless - or connection-oriented communication mode shall be routed or blocked from the Main Line to the Subline and vice versa. The routing can be summarized as follows.

2.4.2.4.2.1 Main Line to Subline Routing

```
if      ZS = own Subnetwork Address then
            if D <> 00h then
                if      C = 7h          then ROUTE_UNMODIFIED
                elsif   0h < C < 7h     then ROUTE_DECREMENTED
                else    IGNORE_ACKED
            else FORWARD_LOCALLY
else    IGNORE_TOTALLY
```

[1] The ACK is sent by the Data Link Layer.

2.4.2.4.2.2 Subline to Main Line Routing

```
if      ZS <> own Subnetwork Address then
            if      C = 7h          then ROUTE_UNMODIFIED
            elsif   0h < C < 7h     then ROUTE_DECREMENTED
            else    IGNORE_ACKED
elseif D = 00h      then FORWARD_LOCALLY
else    IGNORE_TOTALLY
```

2.4.2.4.3 Routing in case of an Individual Destination Address: Backbone Coupler

2.4.2.4.3.1 Backbone Line to Main Line Routing

```
if      Z = own Area Address then
            if      SD <> 00h then
                    if      C = 7h          then ROUTE_UNMODIFIED
                    elsif   0h < C < 7h     then ROUTE_DECREMENTED
                    else    IGNORE_ACKED
            else FORWARD_LOCALLY
else    IGNORE_TOTALLY
```

2.4.2.4.3.2 Main Line to Backbone Routing

```
if Z <> own Area Address then
            if      C = 7h          then ROUTE_UNMODIFIED
            elsif   0h < C < 7h     then ROUTE_DECREMENTED
            else    IGNORE_ACKED
elseif SD = 00h then FORWARD_LOCALLY
else        IGNORE_TOTALLY
```

2.4.2.4.4 Routing in case of a Broadcast Destination Address

2.4.2.4.4.1 Between closed media

```
  if        C = 7h          then ROUTE_UNMODIFIED
            elsif   0h < C < 7h     then ROUTE_DECREMENTED
  else      IGNORE_ACKED
```

2.4.2.4.5 Routing in case of System Broadcast Destination Address – Media Coupler TP1-PL110

2.4.2.4.5.1 Routing

```
if      C = 7h          then    ROUTE_UNMODIFIED
elseif      0h < C < 7h     then    ROUTE_LAST
else        IGNORE_ACKED
```

2.4.2.4.5.1.1 System broadcast conversion

Due to the fact that on closed media the Domain Address is not transmitted, the Media Coupler shall provide a conversion procedure that transforms a broadcast to a system broadcast and vice versa.

After conversion of messages according to the rules below the routing shall be performed according to the above mentioned routing algorithms.

2.4.2.4.5.2 Conversion algorithm from open media to closed media:

```
if L_SystemBroadcast.ind then    route message as L_Data.req
```

2.4.2.4.5.3 Conversion algorithm from closed media to open media:

```
if      APCI = open media specific service primitive
        then route message as L_SystemBroadcast.req

else         L_Data.req
```

The open media specific service primitives are specified in [03].

These are the service primitives

- APCI_DomainAddress_Write,
- APCI_DomainAddress_Read,
- APCI_DomainAddress_Response and
- APCI_DomainAddress_SelectiveRead.