System Specifications

KNXnet/IP

Routing

Summary

This document provides the KNXnet/IP Routing specification.

Version 01.05.01 is a KNX Approved Standard.

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

3

8

5

Document updates

Version	Date	Modifications
1.0 DP	2004.01.08	Final version for Release for Voting
1.1 DV	2005.05.30	Draft for Voting provided for Final Voting.
1.3 AS	2008.07.02	Publication of the Approved Standard.
1.4 AS	2009.06.29	Editorial update in view of inclusion in the KNX Specifications v2.0.
1.5.00 AS	2010.07.26	• **AN117 "KNX IP Communication Medium"** integrated
01.05.01	2013.10.28	Editorial updates for the publication of KNX Specifications 2.1.

References

A general reference is made to the RFCs [1)] defining the Internet Protocol. These documents can be obtained on the Internet at http://www.ietf.org/rfc.html.

[1] Chapter 3/3/3 "Network Layer"
[2] Chapter 3/5/1 "Resources"
[3] Chapter 3/5/2 "Management Procedures"
[4] Chapter 3/6/3 "External Message Interface", v1.0 AS
[5] Chapter 3/8/1 "Overview" (KNXnet/IP)
[6] Chapter 3/8/2 "Core" (KNXnet/IP)
[7] Chapter 3/8/3 "Device Management" (KNXnet/IP)
[8] Chapter 3/8/7 "Remote Configuration and Diagnosis"
[9] Part 10/1 "Logical Tag Extended"

Filename: 03_08_05 Routing v01.05.01 AS.docx
Version: 01.05.01
Status: Approved Standard
Savedate: 2013.10.28
Number of pages: 21

1) Request for Comment: Internet Standards defined by the Internet Engineering Task Force (IETF) are firstly published as RFCs.

Content

1 Introduction

1.1 Scope

This specification defines the integration of KNX protocol implementations on top of Internet Protocol (IP) networks, called KNXnet/IP. It describes a standard protocol for KNX devices connected to an IP network, called KNXnet/IP devices. The IP network acts as a fast (compared to KNX transmission speed) backbone in KNX installations.

An overview of KNXnet/IP is presented in [5].

General Frame descriptions and data exchange protocols between KNXnet/IP devices are described in [6].

General device management and configuration of KNXnet/IP devices is specified in [7].

This Chapter 3/8/5 “Routing” describes a point-to-multipoint standard protocol for routing between KNX devices, called KNXnet/IP Routers, over an IP network.

This document defines a standard protocol that is implemented within KNX devices and the Engineering Tool Software (ETS) to support KNX data exchange over non-KNX networks.

1.2 Definitions, acronyms and abbreviations

Refer to [5] for a list of definitions for the KNXnet/IP specification.

2 KNXnet/IP Routing of KNX telegrams

2.1 Introduction

KNXnet/IP Routing is defined as a set of KNXnet/IP Routers communicating over a one-to-many communication relationship (multicast), in which KNX data shall be transferred from one device to one or more other devices simultaneously over an IP network. A set of KNXnet/IP Routers can replace KNX Line- and Backbone Couplers and connected Main Lines respectively Backbone Lines, allowing usage of existing cabling (e.g. Ethernet) and faster transmission times (and simultaneousness) between KNX subnets. The IP network acts as a fast backbone that connects KNX subnets and is a high-speed replacement for the KNX backbone.

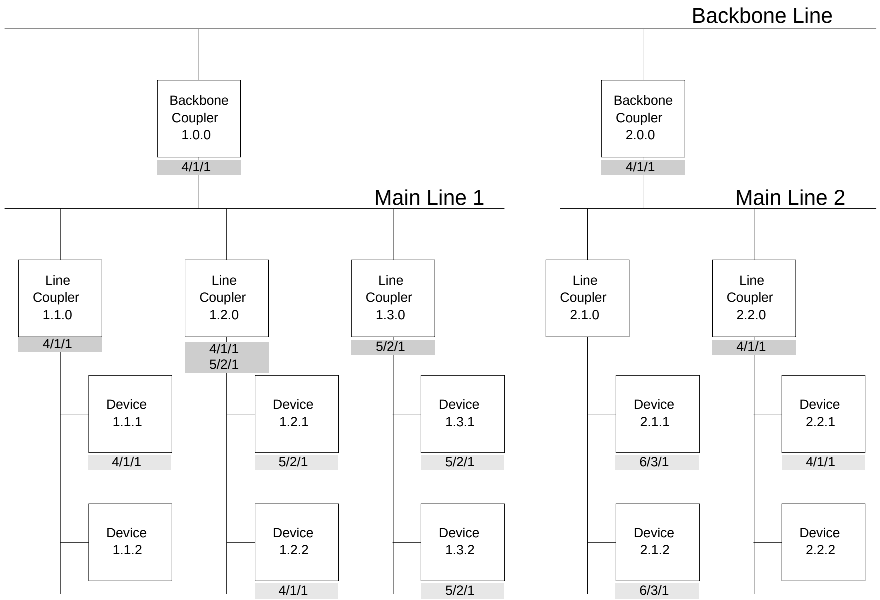

Figure 1 – KNX Group Telegram Routing

In Figure 1, a part of a KNX network is displayed. Every KNX device ("Device", Line Coupler and Backbone Coupler) has its unique KNX Individual Address that shall reflect the bus topology. Couplers make use of this correlation when filtering telegrams in point-to-point communication mode, using a filter mask defined by their own KNX Individual Address and telegram direction (from hierarchically lower Subnetwork to hierarchically higher Subnetwork or vice versa).

A group telegram sent by a KNX device shall be passed through the network from Coupler to Coupler (provided Filter Tables and/or routing rules allow the telegram to pass) until it reaches the target device(s). Depending on bus topology and KNX Group Address usage, it may take multiple Couplers to receive and retransmit a group telegram until it reaches all targets; the same obtains for point-to-point connectionless and - connection-oriented telegrams, only there is exactly one target device.

EXAMPLE A group telegram (on Group Address 4/1/1) transmitted by device 1.1.1 is retransmitted by Coupler 1.1.0 on Main Line 1.0, then by Coupler 1.2.0 on its Line and by Coupler 1.0.0 on the Backbone Line, by Coupler 2.0.0 on Main Line 2.0 and finally by Coupler 2.2.0 on its Line 2.2, which makes a total of five consecutive transmissions (the sixth one is handled at the same time as another transmission and is not counted here).

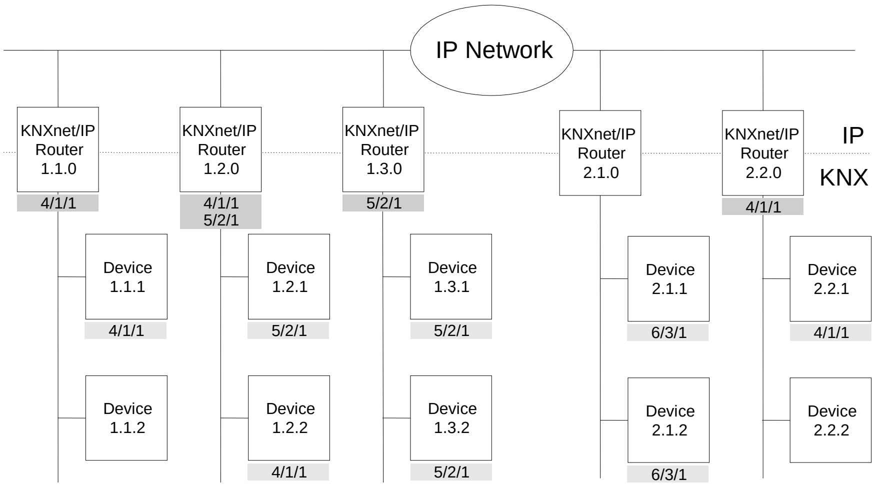

Figure 2 – KNXnet/IP Group Telegram Routing

In Figure 2 the same KNX devices as in Figure 1 are connected to KNXnet/IP Routers, which in turn are all connected to the same IP network. Every telegram from a Subnetwork that matches the filter criteria in its KNXnet/IP Router is encapsulated in an IP datagram and transferred to the IP network. All KNXnet/IP Routers connected to the IP network will receive the datagram simultaneously and, when in turn the filter criteria are matched, transmit the KNX telegram to their Subnetwork.

Compared to the example above: a group telegram (on Group Address 4/1/1) transmitted by device 1.1.1 is put on the IP network by KNXnet/IP Router 1.1.0 and finally by KNXnet/IP Routers 1.2.0 and 2.2.0 on their respective subnetworks, which makes a total of three consecutive transmissions. One of these transmissions is over the IP backbone and therefore very fast.

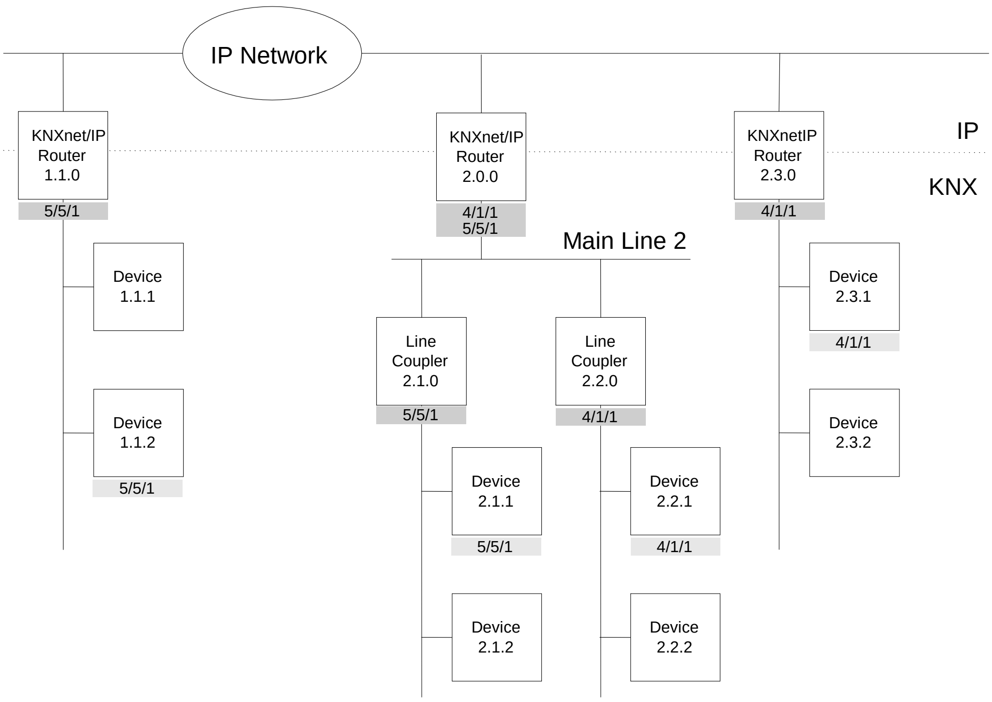

Figure 3 – Mixed topology (undesired subnetwork addressing)

Figure 3 shows a mixed topology of IP and KNX network. One KNXnet/IP Router couples a KNX Main Line to the IP network, others couple KNX Lines. It should be noticed that one of the KNX Lines would normally be located below the KNX Main Line, but it is coupled to the IP network directly.

Considering a telegram in point-to-point (connectionless or connection-oriented) communication mode sent by device 2.2.1 targeting device 2.3.2, which is routed by Line Coupler 2.2.0 to the KNX Main Line 2, Router 2.0.0 would normally not expect to route this telegram over IP. Or, reversed, for a telegram in point-to-point communication mode sent by device 1.1.1 targeting the same device 2.3.2, which is routed by router 1.1.0 to IP, router 2.0.0 would normally expect that target device to be located down its own KNX route and therefore transmit it to KNX Main Line 2, where it is, in this example, not needed.

The mixed topology is a result of the demand to use IP as a fast backbone: to take as much advantage of the fast IP network transmission as possible, most KNX Lines should be attached directly (by use of a KNXnet/IP Router) to the IP network. As KNX end devices can also be connected to KNX Main Lines, it must be possible to attach the latter to the IP network as well. As a result, to maintain standard routing of telegrams in point-to-point connectionless and connection-oriented communication mode (and routing Filter Table computation) ETS address assignment rules have to be extended.

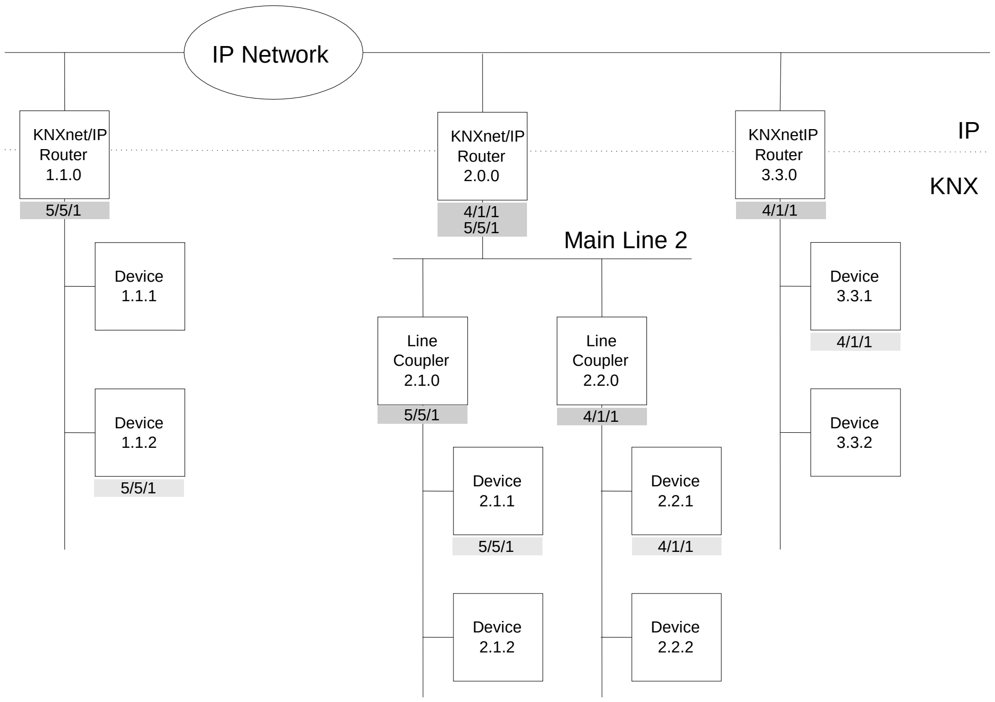

Figure 4 – Mixed topology (desired Subnetwork address assignment)

ETS shall implement the following rules for address assignment to KNXnet/IP devices.

A KNXnet/IP Router shall not be assigned to a KNX Line Coupler address (x.y.0) if the KNX Backbone Coupler address (x.0.0) is already assigned to another KNXnet/IP Router or if one or more KNX Line Couplers are already assigned to a KNX Line Coupler address (x.y.0), where y = [1 … 15]. A KNXnet/IP Router shall not be assigned to a KNX Backbone Coupler address (x.0.0) if one or more KNXnet/IP Routers are already assigned to a KNX Line Coupler address (x.y.0), where y = [1 ... 15].

These rules ensure proper routing of KNX telegrams with KNXnet/IP Routers.

Any KNXnet/IP device may be assigned to the KNX Individual Addresses in the range [(0.0.1)…(0.0.255)]. These KNXnet/IP devices shall implement KNXnet/IP Routing as the standard means of communication even though these KNXnet/IP devices are no KNXnet/IP Routers.

2.2 IP Multicasts

The one-to-many relationship requires IP multicasts to use a connectionless transport protocol. Every KNXnet/IP Router shall therefore support UDP/IP multicasts. Successful transmission of data from one router to another is not guaranteed, but on IP networks data loss is extremely unlikely. KNXnet/IP Routers shall use IGMP (Internet Group Management Protocol) to inform local IP multicast routers of the new multicast address user (IGMP v2 Membership Report), allowing KNXnet/IP Routing datagrams to cross IP Routers into other segments or subnetworks of the IP network.

By setting the TTL of outgoing datagrams, a KNXnet/IP Router can control how many “hops” the datagram can make, that is, how many IP routers the datagram may pass until it is discarded. As a result, the KNXnet/IP Router can be parameterised to transmit its multicast datagrams to the local IP network only. On the other hand, there is no way to prohibit multicast datagrams from other parts of the IP network to reach the KNXnet/IP Router, even when this does not announce its multicast group membership to local IP routers, because other devices may do so.

A sequence of telegrams sent by one router may also arrive at another router out of order, if these KNXnet/IP Routers are not situated on the same IP network.

2.3 Routing

2.3.1 Basics

Every installation shall use the same IP multicast address and port for the transmission of routing information, as stated in the “KNXnet/IP Routing HPAI”. The port number 3671 is registered at the Internet Authority for Number Assignment (IANA) for this purpose.

2.3.2 Multicast group membership

KNXnet/IP Routers do not establish communication channels between each other. A KNXnet/IP Router shall always transmit its data to all other KNXnet/IP Routers in the same IP multicast (group) address residing on the same network or even, depending on the TTL of the routing datagram, other networks reachable over IP routers.

KNXnet/IP Routers shall by default be assigned to the KNXnet/IP System Setup Multicast Address. Any telegram from the KNX network shall be sent to and received from this KNXnet/IP Routing Multicast Address.

Yet, if more than one KNX installation will use the same IP network it must do so without interfering with other KNX installations.

The same is the case with installations exceeding 180 (= 15 * 12) KNX Subnetworks i.e. where more than one KNX installation is installed.

In those cases KNXnet/IP Routers of separate installations shall use different KNXnet/IP Routing Multicast Addresses.

The KNXnet/IP Routing Multicast Address shall denote the KNX installation a KNXnet/IP Router is assigned to.

A KNXnet/IP Router may support routing from one KNX installation to another.

2.3.3 Hardware requirements

As every KNXnet/IP Router receives every routing datagram, a KNXnet/IP Router should be able to handle the theoretical maximum IP media traffic, mostly depending on the used Physical Layer.

2.3.4 Lost message handling

Depending on the configuration a KNXnet/IP Router can receive more messages from the LAN than it can send to the KNX Subnetwork. This can lead to an overflow of the LAN-to-KNX queue and subsequent loss of a KNXnet/IP message because it can not be transferred from the network buffer to the queue.

In this event the Property PID_QUEUE_OVERFLOW_TO_KNX value shall be incremented and a ROUTING_LOST_MESSAGE notification shall be multicast to the KNXnet/IP Routing Multicast Address.

If the connection of the KNXnet/IP Router to the LAN is broken it cannot forward telegrams from the KNX Subnetwork to the LAN. This can lead to an overflow of the KNX-to-LAN queue and subsequent loss of a KNX message because it can not be transferred from the KNX Subnetwork buffer to the queue.

In this event the Property PID_QUEUE_OVERFLOW_TO_IP value shall be incremented.

This notification allows a central supervising entity to log the routing traffic and determine potential problems with the system network design.

2.3.5 Flow control handling

Depending on the configuration a KNXnet/IP Router could receive more Datagrams from the LAN than it can send to the KNX Subnetwork. This could lead to an overflow of the LAN-to-KNX queue and subsequent loss of one or more KNXnet/IP telegrams because they could not be transferred from the network buffer to the queue to the underlying KNX Subnetwork.

Flow control shall for KNXnet/IP Routers and KNX IP devices avoid the loss of Datagrams due to overflowing queues in KNXnet/IP Routers and KNX IP devices.

Limiting the data rate of sending devices is not a solution for flow control as it does not guarantee that the incoming queue on a specific device (e.g. a KNXnet/IP Router) does not overflow because it is receiving Datagrams to be sent onto the local Subnetwork from more than one sending device. The solution is for a receiving device to indicate to all other devices that its incoming queue is filling up and it may loose Datagrams if they do not stop sending.

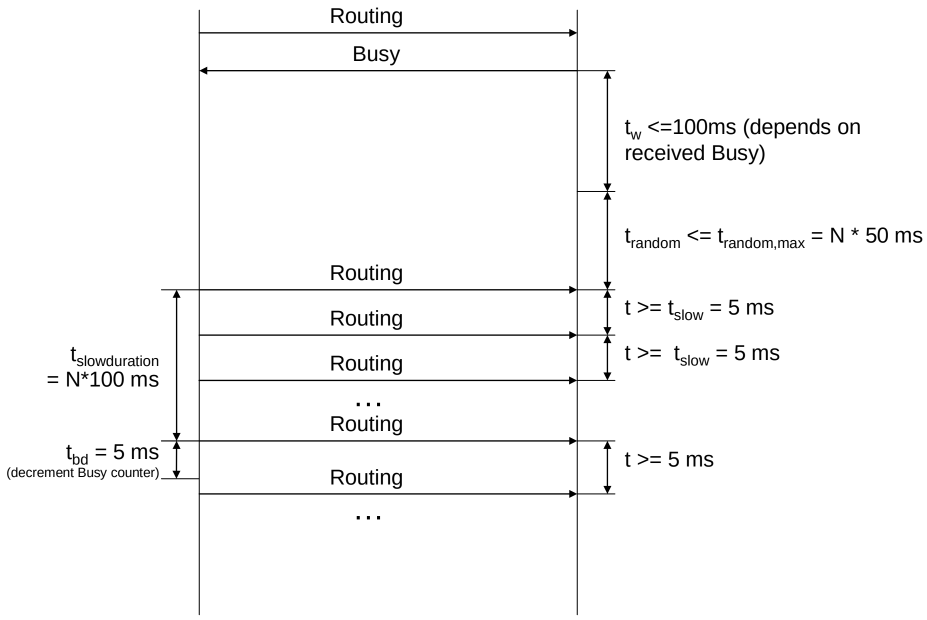

Figure 5 – Flow control with ROUTING_BUSY

Device sending ROUTING_BUSY

If the incoming queue (e.g. KNX IP to KNX TP1) of a KNXnet/IP Router or KNX IP device exceeds the number of Datagrams that can be processed (e.g. sent to the local KNX Subnetwork by a KNXnet/IP Router) within a period of $T_{process}$ then this device SHALL send a ROUTING_BUSY Frame with a wait time t_w. The default value for $T_{process}$ SHOULD be 100 ms and MAY be greater than 100 ms. t_w SHOULD resemble the time required to empty the incoming queue.

The value of t_w used by a device shall be stored in Property PID_ROUTING_BUSY_WAIT_TIME. t_w SHALL be at least 20 ms and SHALL not exceed 100 ms.

The ROUTING_BUSY Frame shall contain the routing_busy_control field. By default this routing_busy_control field SHALL be set to 0000h. This default value requires all KNX/IP devices and KNXnet/IP Routers to act upon receiving the ROUTING_BUSY Frame.

The threshold for sending a ROUTING_BUSY Frame with the Individual Address from the last ROUTING_INDICATION Frame SHOULD be set at five messages in the incoming queue [2)].

The threshold for sending a ROUTING_BUSY Frame to all KNX/IP devices and KNXnet/IP Routers SHOULD be set at ten messages in the incoming queue.

The incoming queue SHOULD be able to hold at least 30 messages.

Device receiving ROUTING_BUSY

Any KNX IP device and KNXnet/IP Router SHALL stop sending ROUTING_INDICATION Frames as soon as it receives a ROUTING_BUSY Frame for the time t_w with a routing busy control field set to 0000h. This is where the ROUTING_BUSY Frame acts as general flow control.

If another ROUTING_BUSY Frame is received before the time t_w has elapsed the resulting time t_w SHALL be determined by the higher value of the remaining time of a previous ROUTING_BUSY and the value t_w received with this last ROUTING_BUSY.

A KNX IP device or KNXnet/IP Router MAY resume sending after the wait time t_w has expired and an additional random wait time t_{random} has passed.
For an individual device the total time from receiving the ROUTING_BUSY to resuming sending shall be

(1) $t_{w,\ total} = t_w + t_{random}$

(2) $t_{random} = [0\ldots1]_{random} * N * 50\ ms;$ $\{ 0 \leq t_{random,} \leq N * 50\ ms \}$

The additional random wait time t_{random} shall be derived from a random real number in the range [0…1] multiplied by 50 ms to transmit and process a datagram times N.
N is defined as the number of ROUTING_BUSY Frames received in a moving period. N shall be incremented by one with each ROUTING_BUSY Frame received after 10 ms [3)] have passed since the last ROUTING_BUSY and decremented by one every t_{bd} = 5 ms after $t_{slowduration}$ has elapsed.

(3) $t_{slowduration} = N * 100\ ms$

The ROUTING_BUSY Frame allows a central supervising entity to log the routing traffic and determine potential problems with the system network design.

If the ROUTING_BUSY Frame contains a routing busy control field value not equal to 0000h then any device that does not interpret this routing busy control field SHALL stop sending for the time t_w. In this case the rules for the general flow control also apply.

---

2) The recommended values are based on a system simulation assuming that up to 255 devices send 50 ROUTING_INDICATION datagrams per second.

3) This value assumes that incoming ROUTING_BUSY datagrams are not due to network delays past 10 ms after the first ROUTING_BUSY is received. This avoids incrementing the counter N because more than one device exceeds the buffer trigger for sending ROUTING_BUSY at the same time.

3 Implementation rules and guidelines

3.1 Introduction

This clause of the specification describes the implementation details and features of a KNXnet/IP Router. KNX Extended Frames are routed in the same way as KNX Standard Frames.

3.2 Discovery and self description

Every KNXnet/IP Router shall support discovery and self description according to [6].

3.3 Group Address Filtering

3.3.1 Standard Group Addresses

All KNX Group Address telegrams received by a KNXnet/IP Router shall be subject to Group Address filtering, unless filtering is switched off completely.

Details regarding KNX Group Address filtering are specified in [1]; the Resources used to this (Parameters, Filter Tables) are specified in [2].

3.3.2 LTE Group Address Filtering

LTE Group Address Filtering requires the presence of the LTE Routing functionality in the KNXnet/IP Router. This is specified in [9] in the clause “LTE Routers”. This requires the presence of the Interface Object “LTE Address Routing Table Object” as specified in [2].

3.4 Individual Address Filtering

All KNXnet/IP Routers shall have a KNX Individual Address of type x.y.0 or x.0.0, where x denotes the Area Address and y the Line Address. As with Line Couplers all Individual Address telegrams shall be routed to the Area respectively Line denoted by the Individual Address of the KNXnet/IP Router.

This normal behaviour may be changed by setting parameters for handling Frames in point-to-point connectionless or – connection-oriented communication mode in the KNXnet/IP Router (PID_MAIN_-LCCONFIG and PID_SUB_LCCONFIG) as specified in [2].

3.5 Telegram from KNX network

Every telegram received from the KNX Subnetwork shall be subject to forwarding rules according to 3.3 and 3.4. In addition, the routing counter shall be taken into consideration, see 3.9.

3.6 Datagram from IP network

3.6.1 Overview

The IP network adapter shall receive every UDP/IP datagram that targets the KNXnet/IP Routing multicast IP address. The KNXnet/IP Router software shall then filter the data by the following criteria:

- port number, and
- multicast IP address, and
- filter table and/or mask.

In addition, the routing counter shall be taken into consideration, see 3.9.

3.6.2 Port number filtering

All KNXnet/IP Routing datagrams shall target a UDP/IP port number 3671. Only datagrams on this port shall be taken into consideration for transmission to the KNX Subnetwork.

3.6.3 Group - and Individual Address Filtering

The Group Address shall be filtered according to 3.3; the Individual Address shall be filtered according to 3.4. Note the KNX routing counter decrementation rules (see 3.9).

3.6.4 Router Object in the KNXnet/IP Router

The Router Object is specified in [2].

3.7 Telegram queuing and forwarding rules

3.7.1 Telegram queuing

A KNXnet/IP Router shall provide two telegram queues for at least two telegrams from the KNX Subnetwork and two telegrams from the IP network. Any telegram that cannot be instantly forwarded to the target network shall be queued in the respective telegram queue.

3.7.2 Forwarding rules

The KNXnet/IP Router may support two forwarding rules that shall be applied if more than one telegram is waiting in the telegram queue: priority/FIFO and normal FIFO mode.

1. In **priority/FIFO mode**, telegrams with the highest KNX priority (system, urgent, normal or low) shall be routed firstly, even when they where queued later than other telegrams with lower priority. If more than one telegram has the highest priority, then the one that stayed in the queue for the longest time shall be transmitted firstly (like in normal FIFO mode).
2. In **normal FIFO mode**, the telegram that stayed in the queue for the longest time shall be transmitted firstly, regardless of KNX priorities.

Normal FIFO mode shall be implemented; priority/FIFO mode may be supported by a KNXnet/IP device.

The KNXnet/IP Router shall discard any telegrams in the queue if the device detects that the connection to the medium is lost.

3.7.3 Queue overflow handling

If a telegram is to be queued when the appropriate queue is already full, one telegram has to be discarded. The telegram to be routed last (determined by the forwarding rules), taking into account the last telegram received, shall be cast off.

3.8 Routing data format

Telegrams received from the KNX subnet shall be transported as cEMI (Data Link Layer, L_Data.ind = cEMI message code 29h) Frames.

3.9 KNX Routing Counter

Provided the KNX telegram Routing Counter (RC) is not zero and not seven, it shall be decremented every time a telegram passes through a KNXnet/IP Router from the KNX Subnetwork to the IP network or vice versa.

If the RC is zero on reception of the Frame from the KNX Subnetwork or IP network, then the received KNX telegram shall not be routed.

If the RC is seven on reception of the Frame from the KNX Subnetwork or IP network, then the telegram shall be routed without decrementing the RC.

3.10 Security

KNXnet/IP Routing Frames shall not be encrypted.

This does not prevent routing KNXnet/IP Routing Frames through Virtual Private Networks (VPN), which by its very nature encrypts these Frames invisible from the KNXnet/IP Routers.

3.11 Error handling

3.11.1 Introduction

Some errors may occur in normal operation, e.g. due to unplugged network connections. These errors shall be reflected in the KNXnet/IP Parameter Object.

3.11.2 KNX net failure

If the KNXnet/IP Router is not be able to transmit telegrams over the KNX Subnetwork for five seconds then the corresponding bit in the Property PID_KNXNETIP_DEVICE_STATE (PID = 69; as specified in [7]) in the KNXnet/IP Parameter Object shall be set to ‘1’. If communication can be resumed, the bit shall be set to ‘0’.

3.11.3 IP net failure

If the communication to the IP network fails for more than five seconds then the corresponding bit in the Property PID_KNXNETIP_DEVICE_STATE (PID = 69; as specified in [7]) in the KNXnet/IP Parameter Object shall be set to ‘1’. If communication can be resumed, the bit shall be set to ‘0’.

3.11.4 Queue overflow

If one of the two routing queues overflows (see 3.7.3) and queue overflow statistics are implemented and activated (see [7], PID_KNXNETIP_ROUTING_CAPABILITIES), the corresponding counter PID_QUEUE_OVERFLOW_TO_IP or PID_QUEUE_OVERFLOW_TO_KNX shall be incremented.

3.12 Router statistics and status information

A KNXnet/IP Router shall provide statistics and status information, see [7] (clause “KNXnet/IP Parameter Object”). All values shall be unsigned, and all counts shall not wrap around if the maximum is reached (property can then be set to 0). This Property shall not be writable to protect the value from being changed accidentally. It may be reset to zero by removing power from the device or by a device hard reset or by another means provided by and at the discretion of the manufacturer.

4 Configuration and Management

4.1 General

General device management and configuration of KNXnet/IP devices is described in [7].

4.2 Property ID Definitions

In document [7] the specification of the KNXnet/IP Parameters Object in clause 2.5 contains Properties specific to KNXnet/IP Routing.

Additional Properties are not defined in this document.

4.3 Object Types

4.3.1 KNXnet/IP Parameter Object

The KNXnet/IP Parameter Object shall include the IP parameters for the KNXnet/IP device’s routing service container.

Refer to [7] for a detailed specification of this Interface Object Type.

A KNXnet/IP Router shall implement this Interface Object Type.

4.3.2 Router Object

The Router Interface Object is specified in [2].

A KNXnet/IP Router shall implement this Interface Object Type.

5 Data packet structures

5.1 KNXnet/IP services

Table 1 lists the KNXnet/IP Routing services.

Table 1 – KNXnet/IP Routing service type identifier

Service name	Code	V.	Description
ROUTING_INDICATION	0530h	1	Used for sending KNX Frames over IP networks. This service shall be unconfirmed.
ROUTING_LOST_MESSAGE	0531h	1	Used for indication of lost KNXnet/IP Routing Frames. This service shall be unconfirmed.

5.2 ROUTING_INDICATION

```
                        KNXnet/IP header
+-7-+-6-+-5-+-4-+-3-+-2-+-1-+-0-+-7-+-6-+-5-+-4-+-3-+-2-+-1-+-0-+
|   HEADER_SIZE_10              |   KNXNETIP_VERSION            |
|   (06h)                       |   (10h)                       |
+-7-+-6-+-5-+-4-+-3-+-2-+-1-+-0-+-7-+-6-+-5-+-4-+-3-+-2-+-1-+-0-+
|   ROUTING_INDICATION                                          |
|   (0530h)                                                     |
+-7-+-6-+-5-+-4-+-3-+-2-+-1-+-0-+-7-+-6-+-5-+-4-+-3-+-2-+-1-+-0-+
|   HEADER_SIZE_10 + sizeof(cEMI Frame)                         |
|                                                               |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+

                         KNXnet/IP body
+-7-+-6-+-5-+-4-+-3-+-2-+-1-+-0-+-7-+-6-+-5-+-4-+-3-+-2-+-1-+-0-+
|   cEMI Frame                                                  |
|   (variable length)                                           |
+-7-+-6-+-5-+-4-+-3-+-2-+-1-+-0-+-7-+-6-+-5-+-4-+-3-+-2-+-1-+-0-+
```

Figure 6 – ROUTING_INDICATION Frame binary format

5.3 ROUTING_LOST_MESSAGE

```
                        KNXnet/IP header
+-7-+-6-+-5-+-4-+-3-+-2-+-1-+-0-+-7-+-6-+-5-+-4-+-3-+-2-+-1-+-0-+
|   HEADER_SIZE_10              |   KNXNETIP_VERSION            |
|   (06h)                       |   (10h)                       |
+-7-+-6-+-5-+-4-+-3-+-2-+-1-+-0-+-7-+-6-+-5-+-4-+-3-+-2-+-1-+-0-+
|   ROUTING_LOST_MESSAGE                                        |
|   (0531h)                                                     |
+-7-+-6-+-5-+-4-+-3-+-2-+-1-+-0-+-7-+-6-+-5-+-4-+-3-+-2-+-1-+-0-+
|   HEADER_SIZE_10 + 4                                          |
|                                                               |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
                         LostMessageInfo
+-7-+-6-+-5-+-4-+-3-+-2-+-1-+-0-+-7-+-6-+-5-+-4-+-3-+-2-+-1-+-0-+
|   Structure Length            |   DeviceState                 |
|   (1 octet)                   |   (1 octet)                   |
+-7-+-6-+-5-+-4-+-3-+-2-+-1-+-0-+-7-+-6-+-5-+-4-+-3-+-2-+-1-+-0-+
|   NumberOfLostMessages                                        |
|   (2 octets)                                                  |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
```

Figure 7 – ROUTING_LOST_MESSAGE Frame binary format

5.4 ROUTING_BUSY

```
KNXnet/IP header
+-7-+-6-+-5-+-4-+-3-+-2-+-1-+-0-+-7-+-6-+-5-+-4-+-3-+-2-+-1-+-0-+
|   HEADER_SIZE_10              |   KNXNETIP_VERSION            |
|   (06h)                       |   (10h)                       |
+-7-+-6-+-5-+-4-+-3-+-2-+-1-+-0-+-7-+-6-+-5-+-4-+-3-+-2-+-1-+-0-+
|   ROUTING_BUSY                                                |
|   (0532h)                                                     |
+-7-+-6-+-5-+-4-+-3-+-2-+-1-+-0-+-7-+-6-+-5-+-4-+-3-+-2-+-1-+-0-+
|   HEADER_SIZE_10 + 6                                          |
|                                                               |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
BusyInfo
+-7-+-6-+-5-+-4-+-3-+-2-+-1-+-0-+-7-+-6-+-5-+-4-+-3-+-2-+-1-+-0-+
|   Structure Length            |   DeviceState                 |
|   (1 octet)                   |   (1 octet)                   |
+-7-+-6-+-5-+-4-+-3-+-2-+-1-+-0-+-7-+-6-+-5-+-4-+-3-+-2-+-1-+-0-+
|   ROUTING_BUSY_WAIT_TIME (in milliseconds)                    |
|   (2 octets)                                                  |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|   ROUTING_BUSY_CONTROL_FIELD                                  |
|   (2 octets)                                                  |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
```

Figure 8 – ROUTING_BUSY Frame binary format

6 Binary examples of KNXnet Frames

6.1 ROUTING_INDICATION

```
         +-------------------------------+
  1      |              06h              |    header size
         +-------------------------------+
  2      |              10h              |    protocol version
         +-------------------------------+
  3      |              05h              | \
         +- - - - - - - - - - - - - - - -+  > service type identifier 0530h
  4      |              30h              | /
         +-------------------------------+
  5      |              00h              | \
         +- - - - - - - - - - - - - - - -+  > total length, L+08h octets
  6      |            L+08h              | /
         +-------------------------------+ - - - - cEMI Frame - - - -
  7      |              11h              |    message code (e.g. L_Data.req message)
         +-------------------------------+
  8      |              00h              |    additional information (none)
         +-------------------------------+
  9      |              ...              | \
         +-------------------------------+ |
 10      |              ...              |  > Service Information (L bytes)
         +-------------------------------+ |
L+8      |              ...              | /
         +-------------------------------+
```

Figure 9 – ROUTING_INDICATION Frame binary format: example

6.2 ROUTING_LOST_MESSAGE

```
         +-------------------------------+
  1      |              06h              |    header size
         +-------------------------------+
  2      |              10h              |    protocol version
         +-------------------------------+
  3      |              05h              | \
         +- - - - - - - - - - - - - - - -+  > service type identifier 0531h
  4      |              31h              | /
         +-------------------------------+
  5      |              00h              | \
         +- - - - - - - - - - - - - - - -+  > total length, 0Ah octets
  6      |              0Ah              | /
         +-------------------------------+
  7      |              04h              |    structure length
         +-------------------------------+
  8      |              00h              |    device state
         +-------------------------------+
  9      |              00h              | \
         +- - - - - - - - - - - - - - - -+  > number of lost messages,
 10      |              05h              | /   e.g. 5
         +-------------------------------+
```

Figure 10 – ROUTING_LOST_MESSAGE Frame binary format: example

6.3 ROUTING_BUSY

```
       +-------------------------------+
  1    |             06h               |    header size
       +-------------------------------+
  2    |             10h               |    protocol version
       +-------------------------------+
  3    |             05h               | \
       +- - - - - - - - - - - - - - - -+  > service type identifier 0532h
  4    |             32h               | /
       +-------------------------------+
  5    |             00h               | \
       +- - - - - - - - - - - - - - - -+  > total length, 0Ch octets
  6    |             0Ch               | /
       +-------------------------------+
  7    |             04h               |    structure length
       +-------------------------------+
  8    |             00h               |    device state
       +-------------------------------+
  9    |             00h               | \
       +- - - - - - - - - - - - - - - -+  > number of milliseconds to wait,
 10    |             64h               | /    e.g. 100 ms
       +-------------------------------+
 11    |             00h               | \
       +- - - - - - - - - - - - - - - -+  > routing busy control value
 12    |             00h               | /  (default value: 0000h)
       +-------------------------------+
```

Figure 11 – ROUTING_BUSY frame binary format: example

7 Certification

7.1 Introduction

This clause provides information on the test procedures and requirements of the certification process.

7.2 Service support matrix

Service name	sent from ... to ...	implementation is
ROUTING_INDICATION	Server → Client	M
ROUTING_LOST_MESSAGE	Server → Client	M

Legend: “M” = Mandatory, “O” = Optional, “n.a.” = not applicable

7.3 Test cases

7.3.1 Introduction

Listed here are a number of test cases a KNXnet/IP Router implementation should be checked against.

7.3.2 Normal Operation

Normal operation means single telegram routing.

Nr.	Description	Expected Result
1.1	Router receives group telegram from KNX subnet	If filter settings allow routing, telegram is passed to IP (via MC).
		If filter settings interdict routing, telegram is discarded
1.2	Router receives group telegram from IP network.	If filter settings allow routing, telegram is passed to KNX
		If filter settings interdict routing, telegram is discarded
1.3	Router receives individually addressed telegram from KNX subnet	If filter settings allow routing, telegram is passed to IP (via MC).
		If filter settings interdict routing, telegram is discarded
1.4	Router receives individually addressed telegram from IP network	If filter settings allow routing, telegram is passed to KNX
		If filter settings interdict routing, telegram is discarded

7.3.3 Error cases

Nr.	Description	Expected Result
3.1	Router receives routable telegrams from KNX subnet without IP network connection.	Queue overflow after some telegrams (depends on queue size). If statistics is implemented and enabled, discarded telegrams are counted in PID_QUEUE_OVERFLOW_TO_IP.
3.2	Router receives routable telegrams from IP network without KNX network connection	Queue overflow after some telegrams (depends on queue size). If statistics is implemented and enabled, discarded telegrams are counted in PID_QUEUE_OVERFLOW_TO_KNX.