GSM Fax & Data Services

Test Specification

IP

Author: Texas Instruments Berlin AG
Alt-Moabit 90a
D-10559 Berlin
Germany

Date: 30 August 2003
Document No.: 8444.401.00.006
File: IP.DOC

Table of Contents

0 Document Control

Texas Instruments Berlin AG
Alt Moabit 90a
10559 Berlin
Germany

Telephone: +49.30.3983-0
Fax: +49.30.3983-300
Internet: http://www.ti.com

0.1 Document History

Document Id.	Date	Author	Remarks
8444.401.00.001	9-13 July 2000	Ola Flathus	Changes made by testing
8444.401.00.002	30 September 2000	Ola Flathus	Changes made by testing
8444.401.00.003	09 October 2000	Manfred Gutheins	Symbols of SAP remove
8444.401.00.004	11 October 2000	Ola Flathus	Corrected bugs
8444.401.00.005	06 December 2002	KJF	New IPA SAP
8444.401.00.006	30-August-2003	Jacek Kwasnik	Nomenclature for DTI is clarified and corrected

0.2 References

[1] Rec. T.4 Standardisation of group 3 facsimile apparatus for document transmission;
(CCITT-T.4, 1984)

[2] ITU-T Recommendation T.30; Series T: Terminal equipments and protocols for telematic services;
Procedures for document facsimile transmission in the general switched
telephone network;
(ITU-T.30, 1996)

[3] ITU-T Recommendation T.31; Terminals for telematic services;
Asynchronous facsimile DCE control - service class 1
(ITU-T.31, 1995)

[4] ITU-T Recommendation T.32; Terminals for telematic services;
Asynchronous facsimile DCE control - service class 2
(ITU-T.32, 1995)

[5] Rec. T.35; Terminal equipment and protocols for telematic services;
Procedures for the allocation of CCITT definde codes for non-standard facilities;
(CCITT-T.35, 1991)

[6] ITU-T Recommendation V.25 ter; Series V: data communication over the telephone network;
Interfaces and voiceband modems; Serial asynchronous automatic dialling and control
(ITU-T V.25 ter, 1997)

[7] Rec. V.42 bis Data compression procedures for data circuit terminating equipment (DCE) using
error correction procedures;
(CCITT-V.42 bis, 1990)

[8] Rec. V.110 (Blue book, Vol. VIII, Fascicle VIII.1) Support of data terminal equipments (DTEs)
with V-series type interfaces by an integrated services digital network (ISDN);
(CCITT-V.110, 1988)

[9] European digital cellular telecommunications system (Phase 2);
GSM Public Land Mobile Network (PLMN) connection types;
(GSM 3.10, September 1994, version 4.3.1)

[10] European digital cellular telecommunications system (Phase 2);
Technical realisation of facsimile group 3 transparent;
(GSM 3.45, September 1995, version 4.5.0)

[11] Digital cellular telecommunications system (Phase 2);
Mobile radio interface layer 3 specification;
(GSM 4.08, November 1996, version 4.17.0)

[12] European digital cellular telecommunications system (Phase 2);
Rate adaptation on the Mobile Station - Base Station System (MS - BSS) Interface;
(GSM 4.21, May 1995, version 4.6.0)

[13] European digital cellular telecommunications system (Phase 2);
Radio Link Protocol (RLP) for data and telematic services on the Mobile Station - Base Station
System (MS - BSS) interface and the Base Station System - Mobile-service Switching Centre
(BSS - MSC) interface
(GSM 4.22, September 1994, version 4.3.0)

[14] European digital cellular telecommunications system (Phase 2);
Radio Link Protocol (RLP) for data and telematic services on the Mobile Station - Base Station

System (MS - BSS) interface and the Base Station System - Mobile-service Switching Centre (BSS - MSC) interface
(Amendment prA1 for GSM 4.22, version 4.3.0)
(GSM 4.22, March 1995, version 4.4.0)

[15] European digital cellular telecommunications system (Phase 2);
General on Terminal Adaptation Functions (TAF) for Mobile Stations (MS);
(GSM 7.01, December 1995, version 4.10.0)

[16] European digital cellular telecommunications system (Phase 2);
Terminal Adaptation Functions (TAF) for services using asynchronous bearer capabilities;
(GSM 7.02, September 1994, version 4.5.1)

[17] European digital cellular telecommunications system (Phase 2);
Terminal Adaptation Functions (TAF) for services using synchronous bearer capabilities;
(GSM 7.03, September 1994, version 4.5.1)

[18] Digital cellular telecommunications system (Phase 2);
Use of Data Terminal Equipment - Data Circuit terminating Equipment (DTE - DCE) interface for Short Message Service (SMS) and Cell Broadcast Services (CBS);
(GSM 7.05, November 1996, version 4.8.0)

[19] Digital cellular telecommunications system (Phase 2);
AT command set for GSM Mobile Equipment (ME)
(GSM 7.07, May 1996, version 4.1.0)

[20] Digital cellular telecommunication system (Phase 2);
Mobile Station (MS) conformance specification;
Part 1: Conformance specification
(GSM 11.10-1, November 1996, version 4.17.0)

[21] Digital cellular telecommunications system (Phase 2);
Mobile Station (MS) conformance specification;
Part 2: Protocol Implementation Conformance Statement (PICS)
proforma specification
(GSM 11.10-2, May 1996, version 4.15.0)

[22] Digital cellular telecommunications system (Phase 2);
Mobile Station (MS) conformance specification;
Part 3: Layer 3 (L3) Abstract Test Suite (ATS)
(GSM 11.10-3, November 1996, version 4.17.0)

[23] Proposal for Rate Adaptation implemented on a DSP;
(C. Bianconi, Texas Instruments, January 1998, version 1.0)

[24] MCU-DSP Interfaces for Data Applications;
Specification S844
(C. Bianconi, Texas Instruments, March 1998, version 0.1)

[25] Users Guide
6147.300.96.100; Condat GmbH

[26] Service Access Point RA
8411.100.98.100; Condat GmbH

[27] Service Access Point RLP
8411.101.98.100; Condat GmbH

[28] Service Access Point L2R
8411.102.98.100; Condat GmbH

[29] Service Access Point FAD
8411.103.98.100; Condat GmbH
[30] Service Access Point T30
8411.104.98.100; Condat GmbH
[31] Service Access Point ACI
8411.105.98.100; Condat GmbH
[32] Message Sequence Charts RLP
8411.201.98.100; Condat GmbH
[33] Message Sequence Charts L2R
8411.202.98.100; Condat GmbH
[34] Message Sequence Charts FAD
8411.203.98.100; Condat GmbH
[35] Message Sequence Charts T30
8411.204.98.100; Condat GmbH
[36] Message Sequence Charts ACI
8411.205.98.100; Condat GmbH
[37] Proposal for Fax & Data Integration; March 1998
8411.300.98.100; Condat GmbH
[38] Test Specification RLP
8411.401.98.100; Condat GmbH
[39] Test Specification L2R
8411.402.98.100; Condat GmbH
[40] Test Specification FAD
8411.403.98.100; Condat GmbH
[41] Test Specification T30
8411.404.98.100; Condat GmbH
[42] Test Specification ACI
8411.405.98.100; Condat GmbH
[43] SDL Specification RLP
8411.501.98.100; Condat GmbH
[44] SDL Specification L2R
8411.502.98.100; Condat GmbH
[45] SDL Specification FAD
8411.503.98.100; Condat GmbH
[46] SDL Specification T30
8411.504.98.100; Condat GmbH
[47] SDL Specification ACI
8411.505.98.100; Condat GmbH
[48] Technical Documentation RLP
8411.701.98.100; Condat GmbH
[49] Technical Documentation L2R
8411.702.98.100; Condat GmbH
[50] Technical Documentation FAD
8411.703.98.100; Condat GmbH
[51] Technical Documentation T30
8411.704.98.100; Condat GmbH

[52] Technical Documentation ACI
8411.705.98.100; Condat GmbH

0.3 Abbreviations

ACI AT Command Interpreter
AGCH Access Grant Channel
AT Attention sequence "AT" to indicate valid commands of the ACI

BCCH Broadcast Control Channel
BCS Binary Coded Signals
BS Base Station
BSIC Base Station Identification Code

C/R Command/Response
C1 Path Loss Criterion
C2 Reselection Criterion
CBCH Cell Broadcast Channel
CBQ Cell Bar Qualify
CC Call Control
CCCH Common Control Channel
CCD Condat Coder Decoder
CKSN Ciphering Key Sequence Number
CRC Cyclic Redundancy Check

DCCH Dedicated Control Channel
DISC Disconnect Frame
DL Data Link Layer
DM Disconnected Mode Frame
DTX Discontinuous Transmission

EA Extension Bit Address Field
EL Extension Bit Length Field
EMMI Electrical Man Machine Interface
EOL End Of Line

F Final Bit
F&D Fax and Data Protocol Stack
FACCH Fast Associated Control Channel
FHO Forced Handover

GP Guard Period
GSM Global System for Mobile Communication

HDLC High level Data Link Control
HISR High level Interrupt Service Routine
HPLMN Home Public Land Mobile Network

I Information Frame
IMEI International Mobile Equipment Identity
IMSI International Mobile Subscriber Identity
ITU International Telecommunication Union
IWF Interworking Function

Kc	Authentication Key
L	Length Indicator
LAI	Location Area Information
LISR	Low level Interrupt Service Routine
LPD	Link Protocol Discriminator
M	More Data Bit
MCC	Mobile Country Code
MM	Mobility Management
MMI	Man Machine Interface
MNC	Mobile Network Code
MS	Mobile Station
MSG	Message phase in the GSM 3.45 protocol
N(R)	Receive Number
N(S)	Send Number
NCC	National Colour Code
NECI	New Establishment Causes included
OTD	Observed Time Difference
P	Poll Bit
P/F	Poll/Final Bit
PCH	Paging Channel
PCO	Point of Control and Observation
PDU	Protocol Description Unit
PL	Physical Layer
PLMN	Public Land Mobile Network
RACH	Random Access Channel
REJ	Reject Frame
RNR	Receive Not Ready Frame
RR	Radio Resource Management
RR	Receive Ready Frame
RTD	Real Time Difference
RTOS	Real Time Operating System
SABM	Set Asynchronous Balanced Mode
SACCH	Slow Associated Control Channel
SAP	Service Access Point
SAPI	Service Access Point Identifier
SDCCH	Slow Dedicated Control Channel
SIM	Subscriber Identity Module
SMS	Short Message Service
SMSCB	Short Message Service Cell Broadcast
SS	Supplementary Services
T.4	CCITT Standardisation for Document coding of Group 3 Facsimile Apparatus
TAP	Test Application Program
TCH	Traffic Channel

TCH/F	Traffic Channel Full Rate
TCH/H	Traffic Channel Half Rate
TDMA	Time Division Multiple Access
TE	Terminal Equipment - e. g. a PC
TMSI	Temporary Mobile Subscriber Identity
UA	Unnumbered Acknowledgement Frame
UI	Unnumbered Information Frame
V(A)	Acknowledgement State Variable
V(R)	Receive State Variable
V(S)	Send State Variable
VPLMN	Visiting Public Land Mobile Network

0.4 Terms

Entity: Program which executes the functions of a layer

Message: A message is a data unit which is transferred between the entities of the same layer (peer-to-peer) of the mobile and infrastructure side. Message is used as a synonym to protocol data unit (PDU). A message may contain several information elements.

Primitive: A primitive is a data unit which is transferred between layers on one component (mobile station or infrastructure). The primitive has an operation code which identifies the primitive and its parameters.

Service Access Point: A Service Access Point is a data interface between two layers on one component (mobile station or infrastructure).

1 Overview

The Protocol Stacks are used to define the functionality of the GSM protocols for interfaces. The GSM specifications are normative when used to describe the functionality of interfaces, but the stacks and the subdivision of protocol layers does not imply or restrict any implementation.

The protocol stack for fax and data transmission consists of several entities. Each entity has one ore more service access points, over which the entity provides a service for the upper entity. The entity, which is described in this document, is coloured grey in the following figure :

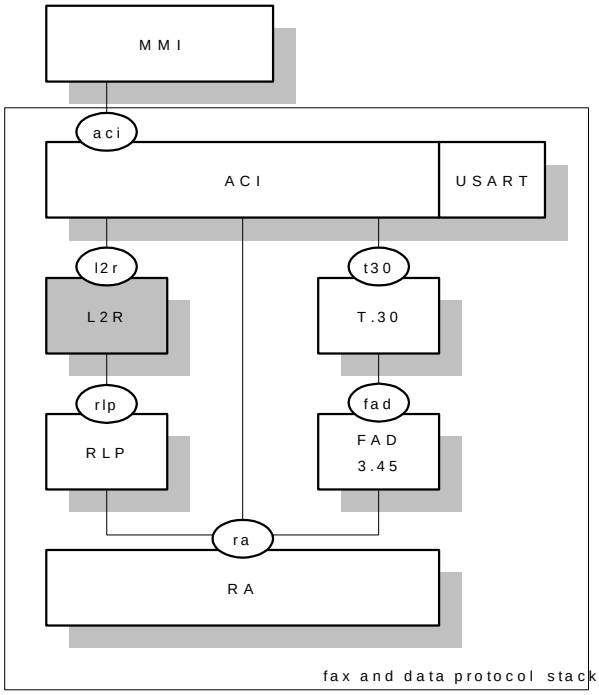


Figure 1-1: Architecture of the fax and data protocol stack

The information units passed via the SAPs are called primitives and consists of an operation code and several parameters. See the Users Guide for details.

The entities of the fax and data protocol stack are:

1.1 RA - Rate Adaptation

This entity performs an adaptation between an asynchronous or synchronous data stream with several bit rates on to the fixed bit rate used at the TCH. This is performed by the rate adaptation functions RA1' and RA0 described in GSM 04.21.

1.2 RLP - Radio Link Protocol

This entity provides a Layer 2 protocol for asynchronous reliable data transfer as specified in GSM 04.22. It includes error correction, sequence numbers and a mechanism for repeating corrupted and lost messages.

1.3 L2R - Layer 2 Relay Functionality

The L2R provides relay functions in order to adapt the character-oriented data received from the TE via USART to the bit-oriented RLP protocol.

1.4 FAD 03.45 - Fax Adaptation Protocol

The fax adaptation protocol, as specified in GSM 03.45, provides synchronisation with the BCS and MSG modems of the peer entity. It uses byte repetition in conjunction with a voting algorithm to handle corruption on the TCH data stream. The non-transparent fax protocol in accordance with GSM 03.46 is not part of this implementation.

The fax adapter enables T.30 to send BCS at 300 BPS and T.4 MSG in 2400, 4800, 7200 and 9600 BPS.

1.5 T.30 - Fax Protocol Entity

The protocol uses binary coded signals packed in HDLC frames to set up and release a connection in the message phase of the FAX transmission. This entity is specified in the ITU-T.30. The main tasks of this unit are:

- Building the HDLC frames with CRC.
- Performing bit stuffing/de-stuffing.
- Executing a sequence of 5 phases: 1.) set up, 2.) pre-message procedures, 3.) transmission/reception, 4.) post message procedures, 5.) waiting for call release.

1.6 ACI - AT Command Interpreter

The ACI is specified in GSM 07.07. It is responsible for call establishment via the GSM voice protocol stack and terminal adaptation for asynchronous transparent character-oriented data transmission. The ACI is able to receive AT commands and send the replies over the USART driver to a remote PC. This makes it possible to control the voice and data protocol stack from a remote application running on a PC. The ACI also provides a unique interface for an internal MMI in the MS.

1.7 USART - Universal Synchronous Asynchronous Receiver Transmitter Driver

The USART is a hardware component that facilitates a connection between the mobile station and terminal equipment (e.g. a PC). This interface uses some of the circuits described in V.24.

The data exchange provided by this unit is serial and asynchronous (synchronous communication is not in the scope of this document). A driver that uses interrupts to manage a circular buffer for the sending and receiving direction is necessary in order to use this component in the F&D. The driver has to be able to perform flow control.

2 Parameters

NOTE: Some of the primitive names described in this document carry "DTI" component in them, the other "DTI2" although actually the same "dti2" type of interface is meant and used. The same applies to parameter names where only "dti" occurs.

```
/* IP header (sans options):        */
/* VER.IHL.TOS.....LEN.............. */
/* ID..............FL.OFFSET....... */
/* TTL.....PROTO...CKSUM........... */
/* SRC.............................. */
/* DST.............................. */
/*                                  */
/* UDP header:                      */
/* SPORT...........DPORT........... */
/* UDPLEN..........CKSUM........... */

STRING (HL_NAME,"UDP")
STRING (LL_NAME,"PPP")

DECLARATION (DTI_PARAMETER_UDP)
DECLARATION (DTI_PARAMETER_IP)
DECLARATION (DTI_PARAMETER_PPP)

DECLARATION (DTI_PARA_ST_LINES_SASB_ON)
DECLARATION (DTI_PARA_ST_LINES_SA_OFF)
DECLARATION (DTI_PARA_ST_LINES_SB_OFF)
DECLARATION (DTI_PARA_ST_LINES_SASB_OFF)

/* IPGEN: processing input file "ip300.packets" */

FIELD(PKT_UDP_NODATA_IN_300)
/* incoming UDP packet with no data bytes */
                    0xe0, 0x00, 0x00, 0x00,
                    /* IP proto 17 len 28 10.11.12.14 -> 10.11.12.13 */
                    0x45, 0x00, 0x00, 0x1c, 0x11, 0xc6, 0x00, 0x00,
                    0x40, 0x11, 0x3c, 0xdb, 0x0a, 0x0b, 0x0c, 0x0e,
                    0x0a, 0x0b, 0x0c, 0x0d,
                    /* UDP sport 1201 dport 9 ulen 8 */
                    0x04, 0xb1, 0x00, 0x09, 0x00, 0x08, 0xce, 0xf3
ENDFIELD(PKT_UDP_NODATA_IN_300, 32)


/* IPGEN: processing input file "ip301.packets" */

FIELD(PKT_UDP_NODATA_IN_301)
/* incoming UDP packet with no data and checksum error */
```

```
                        0xe0, 0x00, 0x00, 0x00,
                        /* IP proto 17 len 28 10.11.12.14 -> 10.11.12.13 */
                        0x45, 0x00, 0x00, 0x1c, 0x11, 0xc6, 0x00, 0x00,
                        0x40, 0x11, 0x04, 0xd2, 0x0a, 0x0b, 0x0c, 0x0e,
                        0x0a, 0x0b, 0x0c, 0x0d,
                        /* UDP sport 1201 dport 9 ulen 8 */
                        0x04, 0xb1, 0x00, 0x09, 0x00, 0x08, 0xce, 0xf3
ENDFIELD(PKT_UDP_NODATA_IN_301, 32)


/* IPGEN: processing input file "ip302.packets" */

FIELD(PKT_UDP_NODATA_OUT_1_302)
/* outgoing UDP packet with no data bytes UDP -> IP */
                        0xe0, 0x00, 0x00, 0x00,
                        /* IP proto 17 len 0 10.11.12.13 -> 10.11.12.14 */
                        0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
                        0x00, 0x11, 0x00, 0x00, 0x0a, 0x0b, 0x0c, 0x0d,
                        0x0a, 0x0b, 0x0c, 0x0e,
                        /* UDP sport 1201 dport 9 ulen 8 */
                        0x04, 0xb1, 0x00, 0x09, 0x00, 0x08, 0xce, 0xf3
ENDFIELD(PKT_UDP_NODATA_OUT_1_302, 32)

FIELD(PKT_UDP_NODATA_OUT_2_302)
/* outgoing UDP packet with no data bytes IP -> PPP */
                        0xe0, 0x00, 0x00, 0x00,
                        /* IP proto 17 len 28 10.11.12.14 -> 10.11.12.13 */
                        0x45, 0x00, 0x00, 0x1c, 0x00, 0x00, 0x00, 0x00,
                        0x40, 0x11, 0x4e, 0xa1, 0x0a, 0x0b, 0x0c, 0x0d,
                        0x0a, 0x0b, 0x0c, 0x0e,
                        /* UDP sport 1201 dport 9 ulen 8 */
                        0x04, 0xb1, 0x00, 0x09, 0x00, 0x08, 0xce, 0xf3
ENDFIELD(PKT_UDP_NODATA_OUT_2_302, 32)


/* IPGEN: processing input file "ip303.packets" */

FIELD(PKT_UDP_NODATA_IN_303)
/* incoming UDP packet, ip_len > packet length */
                        0xe0, 0x00, 0x00, 0x00,
                        /* IP proto 17 len 300 10.11.12.14 -> 10.11.12.13 */
                        0x45, 0x00, 0x01, 0x2c, 0x11, 0xc6, 0x00, 0x00,
                        0x40, 0x11, 0x3b, 0xcb, 0x0a, 0x0b, 0x0c, 0x0e,
                        0x0a, 0x0b, 0x0c, 0x0d,
                        /* UDP sport 1201 dport 9 ulen 8 */
                        0x04, 0xb1, 0x00, 0x09, 0x00, 0x08, 0xce, 0xf3
ENDFIELD(PKT_UDP_NODATA_IN_303, 32)
```

```
/* IPGEN: processing input file "ip304.packets" */

FIELD(PKT_UDP_IN_304)
/* incoming UDP packet, ip_len < packet length */
                        0x60, 0x09, 0x00, 0x00,
                        /* IP proto 17 len 28 10.11.12.14 -> 10.11.12.13 */
                        0x45, 0x00, 0x00, 0x1c, 0x11, 0xc6, 0x00, 0x00,
                        0x40, 0x11, 0x3c, 0xdb, 0x0a, 0x0b, 0x0c, 0x0e,
                        0x0a, 0x0b, 0x0c, 0x0d,
                        /* UDP sport 1201 dport 9 ulen 8 */
                        0x04, 0xb1, 0x00, 0x09, 0x00, 0x08, 0xce, 0xf3,
                        /* data */
                        0x67, 0xc6, 0x69, 0x73, 0x51, 0xff, 0x4a, 0xec,
                        0x29, 0xcd, 0xba, 0xab, 0xf2, 0xfb, 0xe3, 0x46,
                        0x7c, 0xc2, 0x54, 0xf8, 0x1b, 0xe8, 0xe7, 0x8d,
                        0x76, 0x5a, 0x2e, 0x63, 0x33, 0x9f, 0xc9, 0x9a,
                        0x66, 0x32, 0x0d, 0xb7, 0x31, 0x58, 0xa3, 0x5a,
                        0x25, 0x5d, 0x05, 0x17, 0x58, 0xe9, 0x5e, 0xd4,
                        0xab, 0xb2, 0xcd, 0xc6, 0x9b, 0xb4, 0x54, 0x11,
                        0x0e, 0x82, 0x74, 0x41, 0x21, 0x3d, 0xdc, 0x87,
                        0x70, 0xe9, 0x3e, 0xa1, 0x41, 0xe1, 0xfc, 0x67,
                        0x3e, 0x01, 0x7e, 0x97, 0xea, 0xdc, 0x6b, 0x96,
                        0x8f, 0x38, 0x5c, 0x2a, 0xec, 0xb0, 0x3b, 0xfb,
                        0x32, 0xaf, 0x3c, 0x54, 0xec, 0x18, 0xdb, 0x5c,
                        0x02, 0x1a, 0xfe, 0x43, 0xfb, 0xfa, 0xaa, 0x3a,
                        0xfb, 0x29, 0xd1, 0xe6, 0x05, 0x3c, 0x7c, 0x94,
                        0x75, 0xd8, 0xbe, 0x61, 0x89, 0xf9, 0x5c, 0xbb,
                        0xa8, 0x99, 0x0f, 0x95, 0xb1, 0xeb, 0xf1, 0xb3,
                        0x05, 0xef, 0xf7, 0x00, 0xe9, 0xa1, 0x3a, 0xe5,
                        0xca, 0x0b, 0xcb, 0xd0, 0x48, 0x47, 0x64, 0xbd,
                        0x1f, 0x23, 0x1e, 0xa8, 0x1c, 0x7b, 0x64, 0xc5,
                        0x14, 0x73, 0x5a, 0xc5, 0x5e, 0x4b, 0x79, 0x63,
                        0x3b, 0x70, 0x64, 0x24, 0x11, 0x9e, 0x09, 0xdc,
                        0xaa, 0xd4, 0xac, 0xf2, 0x1b, 0x10, 0xaf, 0x3b,
                        0x33, 0xcd, 0xe3, 0x50, 0x48, 0x47, 0x15, 0x5c,
                        0xbb, 0x6f, 0x22, 0x19, 0xba, 0x9b, 0x7d, 0xf5,
                        0x0b, 0xe1, 0x1a, 0x1c, 0x7f, 0x23, 0xf8, 0x29,
                        0xf8, 0xa4, 0x1b, 0x13, 0xb5, 0xca, 0x4e, 0xe8,
                        0x98, 0x32, 0x38, 0xe0, 0x79, 0x4d, 0x3d, 0x34,
                        0xbc, 0x5f, 0x4e, 0x77, 0xfa, 0xcb, 0x6c, 0x05,
                        0xac, 0x86, 0x21, 0x2b, 0xaa, 0x1a, 0x55, 0xa2,
                        0xbe, 0x70, 0xb5, 0x73, 0x3b, 0x04, 0x5c, 0xd3,
                        0x36, 0x94, 0xb3, 0xaf, 0xe2, 0xf0, 0xe4, 0x9e,
                        0x4f, 0x32, 0x15, 0x49, 0xfd, 0x82, 0x4e, 0xa9,
                        0x08, 0x70, 0xd4, 0xb2, 0x8a, 0x29, 0x54, 0x48,
```

```
                        0x9a, 0x0a, 0xbc, 0xd5, 0x0e, 0x18, 0xa8, 0x44
ENDFIELD(PKT_UDP_IN_304, 304)

FIELD(PKT_UDP_IN_2_304)
/* incoming UDP packet, packet length adjusted */
                        0xe0, 0x00, 0x00, 0x00,
                        /* IP proto 17 len 28 10.11.12.14 -> 10.11.12.13 */
                        0x45, 0x00, 0x00, 0x1c, 0x11, 0xc6, 0x00, 0x00,
                        0x40, 0x11, 0x3c, 0xdb, 0x0a, 0x0b, 0x0c, 0x0e,
                        0x0a, 0x0b, 0x0c, 0x0d,
                        /* UDP sport 1201 dport 9 ulen 8 */
                        0x04, 0xb1, 0x00, 0x09, 0x00, 0x08, 0xce, 0xf3
ENDFIELD(PKT_UDP_IN_2_304, 32)

/* IPGEN: processing input file "ip305.packets" */

FIELD(PKT_UDP_IN_305)
/* incoming UDP packet, length > MTU */
                        0x88, 0x0c, 0x00, 0x00,
                        /* IP proto 17 len 401 10.11.12.14 -> 10.11.12.13 */
                        0x45, 0x00, 0x01, 0x91, 0x11, 0xc6, 0x00, 0x00,
                        0x40, 0x11, 0x3b, 0x66, 0x0a, 0x0b, 0x0c, 0x0e,
                        0x0a, 0x0b, 0x0c, 0x0d,
                        /* UDP sport 1201 dport 9 ulen 8 */
                        0x04, 0xb1, 0x00, 0x09, 0x00, 0x08, 0xce, 0xf3,
                        /* data */
                        0xac, 0x5b, 0xf3, 0x8e, 0x4c, 0xd7, 0x2d, 0x9b,
                        0x09, 0x42, 0xe5, 0x06, 0xc4, 0x33, 0xaf, 0xcd,
                        0xa3, 0x84, 0x7f, 0x2d, 0xad, 0xd4, 0x76, 0x47,
                        0xde, 0x32, 0x1c, 0xec, 0x4a, 0xc4, 0x30, 0xf6,
                        0x20, 0x23, 0x85, 0x6c, 0xfb, 0xb2, 0x07, 0x04,
                        0xf4, 0xec, 0x0b, 0xb9, 0x20, 0xba, 0x86, 0xc3,
                        0x3e, 0x05, 0xf1, 0xec, 0xd9, 0x67, 0x33, 0xb7,
                        0x99, 0x50, 0xa3, 0xe3, 0x14, 0xd3, 0xd9, 0x34,
                        0xf7, 0x5e, 0xa0, 0xf2, 0x10, 0xa8, 0xf6, 0x05,
                        0x94, 0x01, 0xbe, 0xb4, 0xbc, 0x44, 0x78, 0xfa,
                        0x49, 0x69, 0xe6, 0x23, 0xd0, 0x1a, 0xda, 0x69,
                        0x6a, 0x7e, 0x4c, 0x7e, 0x51, 0x25, 0xb3, 0x48,
                        0x84, 0x53, 0x3a, 0x94, 0xfb, 0x31, 0x99, 0x90,
                        0x32, 0x57, 0x44, 0xee, 0x9b, 0xbc, 0xe9, 0xe5,
                        0x25, 0xcf, 0x08, 0xf5, 0xe9, 0xe2, 0x5e, 0x53,
                        0x60, 0xaa, 0xd2, 0xb2, 0xd0, 0x85, 0xfa, 0x54,
                        0xd8, 0x35, 0xe8, 0xd4, 0x66, 0x82, 0x64, 0x98,
                        0xd9, 0xa8, 0x87, 0x75, 0x65, 0x70, 0x5a, 0x8a,
                        0x3f, 0x62, 0x80, 0x29, 0x44, 0xde, 0x7c, 0xa5,
                        0x89, 0x4e, 0x57, 0x59, 0xd3, 0x51, 0xad, 0xac,
                        0x86, 0x95, 0x80, 0xec, 0x17, 0xe4, 0x85, 0xf1,
```

```
                        0x8c, 0x0c, 0x66, 0xf1, 0x7c, 0xc0, 0x7c, 0xbb,
                        0x22, 0xfc, 0xe4, 0x66, 0xda, 0x61, 0x0b, 0x63,
                        0xaf, 0x62, 0xbc, 0x83, 0xb4, 0x69, 0x2f, 0x3a,
                        0xff, 0xaf, 0x27, 0x16, 0x93, 0xac, 0x07, 0x1f,
                        0xb8, 0x6d, 0x11, 0x34, 0x2d, 0x8d, 0xef, 0x4f,
                        0x89, 0xd4, 0xb6, 0x63, 0x35, 0xc1, 0xc7, 0xe4,
                        0x24, 0x83, 0x67, 0xd8, 0xed, 0x96, 0x12, 0xec,
                        0x45, 0x39, 0x02, 0xd8, 0xe5, 0x0a, 0xf8, 0x9d,
                        0x77, 0x09, 0xd1, 0xa5, 0x96, 0xc1, 0xf4, 0x1f,
                        0x95, 0xaa, 0x82, 0xca, 0x6c, 0x49, 0xae, 0x90,
                        0xcd, 0x16, 0x68, 0xba, 0xac, 0x7a, 0xa6, 0xf2,
                        0xb4, 0xa8, 0xca, 0x99, 0xb2, 0xc2, 0x37, 0x2a,
                        0xcb, 0x08, 0xcf, 0x61, 0xc9, 0xc3, 0x80, 0x5e,
                        0x6e, 0x03, 0x28, 0xda, 0x4c, 0xd7, 0x6a, 0x19,
                        0xed, 0xd2, 0xd3, 0x99, 0x4c, 0x79, 0x8b, 0x00,
                        0x22, 0x56, 0x9a, 0xd4, 0x18, 0xd1, 0xfe, 0xe4,
                        0xd9, 0xcd, 0x45, 0xa3, 0x91, 0xc6, 0x01, 0xff,
                        0xc9, 0x2a, 0xd9, 0x15, 0x01, 0x43, 0x2f, 0xee,
                        0x15, 0x02, 0x87, 0x61, 0x7c, 0x13, 0x62, 0x9e,
                        0x69, 0xfc, 0x72, 0x81, 0xcd, 0x71, 0x65, 0xa6,
                        0x3e, 0xab, 0x49, 0xcf, 0x71, 0x4b, 0xce, 0x3a,
                        0x75, 0xa7, 0x4f, 0x76, 0xea, 0x7e, 0x64, 0xff,
                        0x81, 0xeb, 0x61, 0xfd, 0xfe, 0xc3, 0x9b, 0x67,
                        0xbf, 0x0d, 0xe9, 0x8c, 0x7e, 0x4e, 0x32, 0xbd,
                        0xf9, 0x7c, 0x8c, 0x6a, 0xc7, 0x5b, 0xa4, 0x3c,
                        0x02, 0xf4, 0xb2, 0xed, 0x72
ENDFIELD(PKT_UDP_IN_305, 405)


/* IPGEN: processing input file "ip306.packets" */

FIELD(PKT_UDP_IN_306)
/* incoming UDP packet, ip_len == MTU, size > MTU */
                        0x88, 0x0c, 0x00, 0x00,
                        /* IP proto 17 len 400 10.11.12.14 -> 10.11.12.13 */
                        0x45, 0x00, 0x01, 0x90, 0x11, 0xc6, 0x00, 0x00,
                        0x40, 0x11, 0x3b, 0x67, 0x0a, 0x0b, 0x0c, 0x0e,
                        0x0a, 0x0b, 0x0c, 0x0d,
                        /* UDP sport 1201 dport 9 ulen 409 */
                        0x04, 0xb1, 0x00, 0x09, 0x01, 0x99, 0xc1, 0x6a,
                        /* data */
                        0x12, 0x23, 0x34, 0x45, 0x56, 0x67, 0x78, 0x89,
                        0x12, 0x23, 0x34, 0x45, 0x56, 0x67, 0x78, 0x89,
                        0x12, 0x23, 0x34, 0x45, 0x56, 0x67, 0x78, 0x89,
                        0x12, 0x23, 0x34, 0x45, 0x56, 0x67, 0x78, 0x89,
                        0x12, 0x23, 0x34, 0x45, 0x56, 0x67, 0x78, 0x89,
                        0x12, 0x23, 0x34, 0x45, 0x56, 0x67, 0x78, 0x89,
```

```
                    0x12, 0x23, 0x34, 0x45, 0x56, 0x67, 0x78, 0x89,
                    0x12, 0x23, 0x34, 0x45, 0x56, 0x67, 0x78, 0x89,
                    0x12, 0x23, 0x34, 0x45, 0x56, 0x67, 0x78, 0x89,
                    0x12, 0x23, 0x34, 0x45, 0x56, 0x67, 0x78, 0x89,
                    0x12, 0x23, 0x34, 0x45, 0x56, 0x67, 0x78, 0x89,
                    0x12, 0x23, 0x34, 0x45, 0x56, 0x67, 0x78, 0x89,
                    0x12, 0x23, 0x34, 0x45, 0x56, 0x67, 0x78, 0x89,
                    0x12, 0x23, 0x34, 0x45, 0x56, 0x67, 0x78, 0x89,
                    0x12, 0x23, 0x34, 0x45, 0x56, 0x67, 0x78, 0x89,
                    0x12, 0x23, 0x34, 0x45, 0x56, 0x67, 0x78, 0x89,
                    0x12, 0x23, 0x34, 0x45, 0x56, 0x67, 0x78, 0x89,
                    0x12, 0x23, 0x34, 0x45, 0x56, 0x67, 0x78, 0x89,
                    0x12, 0x23, 0x34, 0x45, 0x56, 0x67, 0x78, 0x89,
                    0x12, 0x23, 0x34, 0x45, 0x56, 0x67, 0x78, 0x89,
                    0x12, 0x23, 0x34, 0x45, 0x56, 0x67, 0x78, 0x89,
                    0x12, 0x23, 0x34, 0x45, 0x56, 0x67, 0x78, 0x89,
                    0x12, 0x23, 0x34, 0x45, 0x56, 0x67, 0x78, 0x89,
                    0x12, 0x23, 0x34, 0x45, 0x56, 0x67, 0x78, 0x89,
                    0x12, 0x23, 0x34, 0x45, 0x56, 0x67, 0x78, 0x89,
                    0x12, 0x23, 0x34, 0x45, 0x56, 0x67, 0x78, 0x89,
                    0x12, 0x23, 0x34, 0x45, 0x56, 0x67, 0x78, 0x89,
                    0x12, 0x23, 0x34, 0x45, 0x56, 0x67, 0x78, 0x89,
                    0x12, 0x23, 0x34, 0x45, 0x56, 0x67, 0x78, 0x89,
                    0x12, 0x23, 0x34, 0x45, 0x56, 0x67, 0x78, 0x89,
                    0x12, 0x23, 0x34, 0x45, 0x56, 0x67, 0x78, 0x89,
                    0x12, 0x23, 0x34, 0x45, 0x56, 0x67, 0x78, 0x89,
                    0x12, 0x23, 0x34, 0x45, 0x56, 0x67, 0x78, 0x89,
                    0x12, 0x23, 0x34, 0x45, 0x56, 0x67, 0x78, 0x89,
                    0x12, 0x23, 0x34, 0x45, 0x56, 0x67, 0x78, 0x89,
                    0x12, 0x23, 0x34, 0x45, 0x56, 0x67, 0x78, 0x89,
                    0x12, 0x23, 0x34, 0x45, 0x56, 0x67, 0x78, 0x89,
                    0x12, 0x23, 0x34, 0x45, 0x56, 0x67, 0x78, 0x89,
                    0x12, 0x23, 0x34, 0x45, 0x56, 0x67, 0x78, 0x89,
                    0x12, 0x23, 0x34, 0x45, 0x56, 0x67, 0x78, 0x89,
                    0x12, 0x23, 0x34, 0x45, 0x56, 0x67, 0x78, 0x89,
                    0x12, 0x23, 0x34, 0x45, 0x56, 0x67, 0x78, 0x89,
                    0x12, 0x23, 0x34, 0x45, 0x56, 0x67, 0x78, 0x89,
                    0x12, 0x23, 0x34, 0x45, 0x56, 0x67, 0x78, 0x89,
                    0x12, 0x23, 0x34, 0x45, 0x56, 0x67, 0x78, 0x89,
                    0x12, 0x23, 0x34, 0x45, 0x56, 0x67, 0x78, 0x89,
                    0x12, 0x23, 0x34, 0x45, 0x56
ENDFIELD(PKT_UDP_IN_306, 405)

FIELD(PKT_UDP_IN_2_306)
/* incoming UDP packet, ip_len == MTU, size > MTU */
                    0x80, 0x0c, 0x00, 0x00,
                    /* IP proto 17 len 400 10.11.12.14 -> 10.11.12.13 */
```



```
0x45, 0x00, 0x01, 0x90, 0x11, 0xc6, 0x00, 0x00,
0x40, 0x11, 0x3b, 0x67, 0x0a, 0x0b, 0x0c, 0x0e,
0x0a, 0x0b, 0x0c, 0x0d,
/* UDP sport 1201 dport 9 ulen 408 */
0x04, 0xb1, 0x00, 0x09, 0x01, 0x99, 0xc1, 0x6a,
/* data */
0x12, 0x23, 0x34, 0x45, 0x56, 0x67, 0x78, 0x89,
0x12, 0x23, 0x34, 0x45, 0x56, 0x67, 0x78, 0x89,
0x12, 0x23, 0x34, 0x45, 0x56, 0x67, 0x78, 0x89,
0x12, 0x23, 0x34, 0x45, 0x56, 0x67, 0x78, 0x89,
0x12, 0x23, 0x34, 0x45, 0x56, 0x67, 0x78, 0x89,
0x12, 0x23, 0x34, 0x45, 0x56, 0x67, 0x78, 0x89,
0x12, 0x23, 0x34, 0x45, 0x56, 0x67, 0x78, 0x89,
0x12, 0x23, 0x34, 0x45, 0x56, 0x67, 0x78, 0x89,
0x12, 0x23, 0x34, 0x45, 0x56, 0x67, 0x78, 0x89,
0x12, 0x23, 0x34, 0x45, 0x56, 0x67, 0x78, 0x89,
0x12, 0x23, 0x34, 0x45, 0x56, 0x67, 0x78, 0x89,
0x12, 0x23, 0x34, 0x45, 0x56, 0x67, 0x78, 0x89,
0x12, 0x23, 0x34, 0x45, 0x56, 0x67, 0x78, 0x89,
0x12, 0x23, 0x34, 0x45, 0x56, 0x67, 0x78, 0x89,
0x12, 0x23, 0x34, 0x45, 0x56, 0x67, 0x78, 0x89,
0x12, 0x23, 0x34, 0x45, 0x56, 0x67, 0x78, 0x89,
0x12, 0x23, 0x34, 0x45, 0x56, 0x67, 0x78, 0x89,
0x12, 0x23, 0x34, 0x45, 0x56, 0x67, 0x78, 0x89,
0x12, 0x23, 0x34, 0x45, 0x56, 0x67, 0x78, 0x89,
0x12, 0x23, 0x34, 0x45, 0x56, 0x67, 0x78, 0x89,
0x12, 0x23, 0x34, 0x45, 0x56, 0x67, 0x78, 0x89,
0x12, 0x23, 0x34, 0x45, 0x56, 0x67, 0x78, 0x89,
0x12, 0x23, 0x34, 0x45, 0x56, 0x67, 0x78, 0x89,
0x12, 0x23, 0x34, 0x45, 0x56, 0x67, 0x78, 0x89,
0x12, 0x23, 0x34, 0x45, 0x56, 0x67, 0x78, 0x89,
0x12, 0x23, 0x34, 0x45, 0x56, 0x67, 0x78, 0x89,
0x12, 0x23, 0x34, 0x45, 0x56, 0x67, 0x78, 0x89,
0x12, 0x23, 0x34, 0x45, 0x56, 0x67, 0x78, 0x89,
0x12, 0x23, 0x34, 0x45, 0x56, 0x67, 0x78, 0x89,
0x12, 0x23, 0x34, 0x45, 0x56, 0x67, 0x78, 0x89,
0x12, 0x23, 0x34, 0x45, 0x56, 0x67, 0x78, 0x89,
0x12, 0x23, 0x34, 0x45, 0x56, 0x67, 0x78, 0x89,
0x12, 0x23, 0x34, 0x45, 0x56, 0x67, 0x78, 0x89,
0x12, 0x23, 0x34, 0x45, 0x56, 0x67, 0x78, 0x89,
0x12, 0x23, 0x34, 0x45, 0x56, 0x67, 0x78, 0x89,
0x12, 0x23, 0x34, 0x45, 0x56, 0x67, 0x78, 0x89,
0x12, 0x23, 0x34, 0x45, 0x56, 0x67, 0x78, 0x89,
0x12, 0x23, 0x34, 0x45, 0x56, 0x67, 0x78, 0x89,
0x12, 0x23, 0x34, 0x45, 0x56, 0x67, 0x78, 0x89,
0x12, 0x23, 0x34, 0x45, 0x56, 0x67, 0x78, 0x89,
0x12, 0x23, 0x34, 0x45, 0x56, 0x67, 0x78, 0x89,
```

```
                        0x12, 0x23, 0x34, 0x45, 0x56, 0x67, 0x78, 0x89,
                        0x12, 0x23, 0x34, 0x45, 0x56, 0x67, 0x78, 0x89,
                        0x12, 0x23, 0x34, 0x45, 0x56, 0x67, 0x78, 0x89,
                        0x12, 0x23, 0x34, 0x45, 0x56, 0x67, 0x78, 0x89,
                        0x12, 0x23, 0x34, 0x45, 0x56, 0x67, 0x78, 0x89,
                        0x12, 0x23, 0x34, 0x45
ENDFIELD(PKT_UDP_IN_2_306, 404)


/* IPGEN: processing input file "ip307.packets" */

FIELD(PKT_UDP_IN_307)
/* incoming UDP packet, ip_len == MTU */
                        0xe0, 0x2e, 0x00, 0x00,
                        /* IP proto 17 len 1500 10.11.12.14 -> 10.11.12.13 */
                        0x45, 0x00, 0x05, 0xdc, 0x11, 0xc6, 0x00, 0x00,
                        0x40, 0x11, 0x37, 0x1b, 0x0a, 0x0b, 0x0c, 0x0e,
                        0x0a, 0x0b, 0x0c, 0x0d,
                        /* UDP sport 1201 dport 9 ulen 8 */
                        0x04, 0xb1, 0x00, 0x09, 0x00, 0x08, 0xce, 0xf3,
                        /* data */
                        0x16, 0xec, 0xf3, 0x01, 0x4d, 0xf0, 0x00, 0x10,
                        0x8b, 0x67, 0xcf, 0x99, 0x50, 0x5b, 0x17, 0x9f,
                        0x8e, 0xd4, 0x98, 0x0a, 0x61, 0x03, 0xd1, 0xbc,
                        0xa7, 0x0d, 0xbe, 0x9b, 0xbf, 0xab, 0x0e, 0xd5,
                        0x98, 0x01, 0xd6, 0xe5, 0xf2, 0xd6, 0xf6, 0x7d,
                        0x3e, 0xc5, 0x16, 0x8e, 0x21, 0x2e, 0x2d, 0xaf,
                        0x02, 0xc6, 0xb9, 0x63, 0xc9, 0x8a, 0x1f, 0x70,
                        0x97, 0xde, 0x0c, 0x56, 0x89, 0x1a, 0x2b, 0x21,
                        0x1b, 0x01, 0x07, 0x0d, 0xd8, 0xfd, 0x8b, 0x16,
                        0xc2, 0xa1, 0xa4, 0xe3, 0xcf, 0xd2, 0x92, 0xd2,
                        0x98, 0x4b, 0x35, 0x61, 0xd5, 0x55, 0xd1, 0x6c,
                        0x33, 0xdd, 0xc2, 0xbc, 0xf7, 0xed, 0xde, 0x13,
                        0xef, 0xe5, 0x20, 0xc7, 0xe2, 0xab, 0xdd, 0xa4,
                        0x4d, 0x81, 0x88, 0x1c, 0x53, 0x1a, 0xee, 0xeb,
                        0x66, 0x24, 0x4c, 0x3b, 0x79, 0x1e, 0xa8, 0xac,
                        0xfb, 0x6a, 0x68, 0xf3, 0x58, 0x46, 0x06, 0x47,
                        0x2b, 0x26, 0x0e, 0x0d, 0xd2, 0xeb, 0xb2, 0x1f,
                        0x6c, 0x3a, 0x3b, 0xc0, 0x54, 0x2a, 0xab, 0xba,
                        0x4e, 0xf8, 0xf6, 0xc7, 0x16, 0x9e, 0x73, 0x11,
                        0x08, 0xdb, 0x04, 0x60, 0x22, 0x0a, 0xa7, 0x4d,
                        0x31, 0xb5, 0x5b, 0x03, 0xa0, 0x0d, 0x22, 0x0d,
                        0x47, 0x5d, 0xcd, 0x9b, 0x87, 0x78, 0x56, 0xd5,
                        0x70, 0x4c, 0x9c, 0x86, 0xea, 0x0f, 0x98, 0xf2,
                        0xeb, 0x9c, 0x53, 0x0d, 0xa7, 0xfa, 0x5a, 0xd8,
                        0xb0, 0xb5, 0xdb, 0x50, 0xc2, 0xfd, 0x5d, 0x09,
                        0x5a, 0x2a, 0xa5, 0xe2, 0xa3, 0xfb, 0xb7, 0x13,
```

0x47, 0x54, 0x9a, 0x31, 0x63, 0x32, 0x23, 0x4e,
0xce, 0x76, 0x5b, 0x75, 0x71, 0xb6, 0x4d, 0x21,
0x6b, 0x28, 0x71, 0x2e, 0x25, 0xcf, 0x37, 0x80,
0xf9, 0xdc, 0x62, 0x9c, 0xd7, 0x19, 0xb0, 0x1e,
0x6d, 0x4a, 0x4f, 0xd1, 0x7c, 0x73, 0x1f, 0x4a,
0xe9, 0x7b, 0xc0, 0x5a, 0x31, 0x0d, 0x7b, 0x9c,
0x36, 0xed, 0xca, 0x5b, 0xbc, 0x02, 0xdb, 0xb5,
0xde, 0x3d, 0x52, 0xb6, 0x57, 0x02, 0xd4, 0xc4,
0x4c, 0x24, 0x95, 0xc8, 0x97, 0xb5, 0x12, 0x80,
0x30, 0xd2, 0xdb, 0x61, 0xe0, 0x56, 0xfd, 0x16,
0x43, 0xc8, 0x71, 0xff, 0xca, 0x4d, 0xb5, 0xa8,
0x8a, 0x07, 0x5e, 0xe1, 0x09, 0x33, 0xa6, 0x55,
0x57, 0x3b, 0x1d, 0xee, 0xf0, 0x2f, 0x6e, 0x20,
0x02, 0x49, 0x81, 0xe2, 0xa0, 0x7f, 0xf8, 0xe3,
0x47, 0x69, 0xe3, 0x11, 0xb6, 0x98, 0xb9, 0x41,
0x9f, 0x18, 0x22, 0xa8, 0x4b, 0xc8, 0xfd, 0xa2,
0x04, 0x1a, 0x90, 0xf4, 0x49, 0xfe, 0x15, 0x4b,
0x48, 0x96, 0x2d, 0xe8, 0x15, 0x25, 0xcb, 0x5c,
0x8f, 0xae, 0x6d, 0x45, 0x46, 0x27, 0x86, 0xe5,
0x3f, 0xa9, 0x8d, 0x8a, 0x71, 0x8a, 0x2c, 0x75,
0xa4, 0xbc, 0x6a, 0xee, 0xba, 0x7f, 0x39, 0x02,
0x15, 0x67, 0xea, 0x2b, 0x8c, 0xb6, 0x87, 0x1b,
0x64, 0xf5, 0x61, 0xab, 0x1c, 0xe7, 0x90, 0x5b,
0x90, 0x1e, 0xe5, 0x02, 0xa8, 0x11, 0x77, 0x4d,
0xcd, 0xe1, 0x3b, 0x87, 0x60, 0x74, 0x8a, 0x76,
0xdb, 0x74, 0xa1, 0x68, 0x2a, 0x28, 0x83, 0x8f,
0x1d, 0xe4, 0x3a, 0x39, 0xcc, 0xca, 0x94, 0x5c,
0xe8, 0x79, 0x5e, 0x91, 0x8a, 0xd6, 0xde, 0x57,
0xb7, 0x19, 0xdf, 0x18, 0x8d, 0x69, 0x8e, 0x69,
0xdd, 0x2f, 0xd1, 0x08, 0x57, 0x54, 0x97, 0x75,
0x39, 0xd1, 0xae, 0x05, 0x9b, 0x43, 0x61, 0x84,
0xbc, 0xc0, 0x15, 0x47, 0x96, 0xf3, 0x9e, 0x4d,
0x0c, 0x7d, 0x65, 0x99, 0xe6, 0xf3, 0x02, 0xc4,
0x22, 0xd3, 0xcc, 0x7a, 0x28, 0x63, 0xef, 0x61,
0x34, 0x9d, 0x66, 0xcf, 0xe0, 0xc7, 0x53, 0x9d,
0x87, 0x68, 0xe4, 0x1d, 0x5b, 0x82, 0x6b, 0x67,
0x00, 0xd0, 0x01, 0xe6, 0xc4, 0x03, 0xaa, 0xe6,
0xd7, 0x76, 0x60, 0xff, 0xd9, 0x4f, 0x60, 0x0d,
0xed, 0xc6, 0xdd, 0xcd, 0x8d, 0x30, 0x6a, 0x15,
0x99, 0x4e, 0x32, 0xf4, 0xd1, 0x9d, 0x5c, 0xd1,
0x6e, 0x5d, 0xb7, 0x32, 0x60, 0x62, 0x18, 0x37,
0xd8, 0x79, 0x36, 0xb2, 0xc8, 0x96, 0xbf, 0xb5,
0x5c, 0x9c, 0x83, 0xea, 0xcd, 0xed, 0xff, 0x66,
0x3c, 0x31, 0x5a, 0x0d, 0xcf, 0xb6, 0xde, 0x3d,
0x13, 0x95, 0x6f, 0x74, 0xf7, 0x87, 0xab, 0xd0,
0x00, 0xe2, 0x82, 0xc9, 0x78, 0x41, 0x7e, 0xd5,
0xde, 0x01, 0xbf, 0xab, 0xef, 0xbe, 0x11, 0x2b,

0xef, 0x6b, 0x38, 0xbe, 0x22, 0x16, 0xfb, 0x35,
0xab, 0x6a, 0xa9, 0xa3, 0xf2, 0x55, 0x73, 0xf2,
0x37, 0xf5, 0xbb, 0xaf, 0x36, 0x3a, 0x84, 0x14,
0x3b, 0x43, 0xbf, 0x2a, 0x01, 0xd0, 0x55, 0xf1,
0x3c, 0x8d, 0xaf, 0x5e, 0xa3, 0xab, 0x93, 0x4f,
0x15, 0x3d, 0xf2, 0x07, 0x92, 0x65, 0xfa, 0xc9,
0x5a, 0xb5, 0x78, 0x90, 0xef, 0xfd, 0xa5, 0x2b,
0x40, 0x64, 0x55, 0x42, 0x35, 0xab, 0x33, 0x71,
0x38, 0xe2, 0xcf, 0xdc, 0x8d, 0x62, 0x2b, 0xa3,
0x9f, 0x1d, 0xaa, 0x31, 0x82, 0xa4, 0xfa, 0xdc,
0x5a, 0x73, 0x6c, 0x49, 0x70, 0x11, 0x74, 0xb0,
0x76, 0xca, 0xf2, 0xab, 0x75, 0x25, 0x1c, 0xad,
0x08, 0xeb, 0x89, 0x95, 0x4d, 0xb4, 0x38, 0xed,
0xd1, 0xe3, 0x1e, 0x53, 0x87, 0x19, 0x2f, 0xe1,
0x8c, 0x9c, 0x2b, 0xfc, 0xad, 0x9f, 0xac, 0x23,
0x69, 0x9f, 0xce, 0xde, 0xc4, 0xea, 0x8c, 0xcc,
0xd5, 0x15, 0x62, 0x23, 0xca, 0x9a, 0x10, 0x9b,
0x7d, 0x2e, 0xef, 0x05, 0x47, 0x1e, 0xe6, 0xd3,
0xba, 0x11, 0xcf, 0x68, 0xb1, 0x7c, 0x8b, 0x1a,
0x1b, 0x5a, 0xf9, 0xdf, 0x44, 0x85, 0xac, 0x1a,
0x9a, 0x0e, 0x3d, 0x64, 0xa8, 0x4d, 0x00, 0x26,
0x7b, 0xef, 0x2b, 0xc3, 0x0d, 0x11, 0x96, 0xc8,
0x23, 0x66, 0x30, 0xd4, 0xe2, 0xbb, 0xee, 0xfd,
0x15, 0xe7, 0xdc, 0x5a, 0x6c, 0x88, 0x74, 0x07,
0x96, 0xb1, 0x6b, 0x3f, 0xfe, 0x6b, 0x65, 0x79,
0x5a, 0x90, 0x3c, 0x68, 0xa1, 0xd3, 0x30, 0xc4,
0x39, 0x60, 0x98, 0x1b, 0x1b, 0x87, 0x18, 0x31,
0x6e, 0xf4, 0x8b, 0xdb, 0x7d, 0xff, 0xe2, 0x13,
0xb0, 0x4d, 0x52, 0xae, 0xb9, 0xb7, 0x27, 0x13,
0x47, 0x64, 0x7b, 0xe9, 0x37, 0xab, 0xad, 0x70,
0x0b, 0x46, 0x8b, 0x27, 0xcd, 0xa3, 0x58, 0x3b,
0x97, 0xe3, 0x16, 0x14, 0xe2, 0xf8, 0x28, 0x92,
0x46, 0x7a, 0x40, 0xff, 0x32, 0x67, 0x12, 0x79,
0xcb, 0x8e, 0x62, 0x02, 0x39, 0x10, 0x72, 0x45,
0x56, 0xfd, 0x6c, 0x23, 0xa0, 0xc4, 0x5e, 0x38,
0xa7, 0x75, 0x4c, 0x89, 0x6d, 0x74, 0x1b, 0xb3,
0xef, 0x5b, 0xb2, 0x21, 0xc2, 0xc5, 0x9a, 0x8e,
0x53, 0xfd, 0x90, 0x8c, 0x0d, 0x03, 0xd1, 0x63,
0x00, 0x3d, 0x86, 0xa1, 0x01, 0xe4, 0xd9, 0xa8,
0x59, 0x25, 0x31, 0xc7, 0x9a, 0x4c, 0x7a, 0x89,
0xa7, 0x2d, 0xaa, 0x6a, 0xf2, 0x44, 0xf8, 0x45,
0x41, 0x88, 0xd1, 0x4e, 0x8b, 0xa3, 0xb1, 0x8c,
0xe0, 0x37, 0x2d, 0xe2, 0x1c, 0x06, 0x8a, 0x75,
0x2b, 0xbc, 0x3c, 0xc5, 0x08, 0xb7, 0x4e, 0xb0,
0xe4, 0xf8, 0x1a, 0xd6, 0x3d, 0x12, 0x1b, 0x7e,
0x9a, 0xec, 0xcd, 0x26, 0x8f, 0x7e, 0xb2, 0x70,
0xb6, 0xdf, 0x52, 0xd2, 0xe5, 0xdc, 0x47, 0x10,

0x98, 0x84, 0xd6, 0xa1, 0x3b, 0x24, 0x51, 0x1f,
0x1d, 0x6b, 0xf5, 0x5a, 0x7d, 0x10, 0xd8, 0x17,
0xfc, 0xa5, 0x3d, 0x8c, 0x24, 0xef, 0xfc, 0xda,
0xce, 0x4e, 0xac, 0xb3, 0x2a, 0xf3, 0xc4, 0xc3,
0x77, 0x9a, 0x64, 0xb2, 0xbe, 0xb5, 0xd1, 0xdb,
0x20, 0xc6, 0x35, 0x9d, 0xd6, 0x0e, 0xb4, 0xd3,
0xb3, 0xf2, 0x5f, 0xd7, 0xe1, 0x5b, 0xb1, 0xb0,
0xa9, 0x5d, 0x63, 0xd3, 0x51, 0x27, 0x96, 0xc8,
0xc1, 0xfa, 0x7b, 0x80, 0xaf, 0x4c, 0x5b, 0xcf,
0x13, 0x91, 0x6c, 0xe9, 0x9f, 0x21, 0xbc, 0x52,
0x13, 0x1b, 0x2a, 0xf4, 0x76, 0xdb, 0xa4, 0x1f,
0x39, 0x08, 0xf3, 0x8a, 0x2f, 0x89, 0x52, 0xf1,
0x84, 0xcd, 0x71, 0x33, 0x1a, 0xcc, 0x03, 0x2d,
0x5d, 0x6f, 0x16, 0xfc, 0x90, 0xd3, 0x4f, 0xa3,
0xee, 0x79, 0x98, 0x65, 0x54, 0x3c, 0x84, 0x8d,
0x44, 0x77, 0x17, 0x74, 0x01, 0x6a, 0x65, 0x85,
0x37, 0xd6, 0xb8, 0x51, 0xa2, 0xbb, 0x7e, 0x00,
0x2b, 0x95, 0xfc, 0xbb, 0x68, 0x4b, 0x5f, 0x56,
0xc4, 0xf7, 0xbb, 0x19, 0x33, 0x40, 0xa6, 0x78,
0xb7, 0xbe, 0xec, 0xb8, 0x28, 0x51, 0x3d, 0x5f,
0x27, 0xf6, 0xb1, 0xc9, 0xb1, 0x2f, 0xc9, 0xdc,
0xc4, 0xc6, 0x98, 0x2c, 0x11, 0xf7, 0x83, 0xd6,
0xee, 0x3e, 0xef, 0x21, 0x7e, 0x95, 0x99, 0x36,
0x53, 0x85, 0xee, 0x7b, 0xd6, 0x2c, 0xdb, 0xfd,
0x22, 0x8c, 0xc7, 0xd3, 0xbb, 0x90, 0xb0, 0x80,
0x56, 0x48, 0xac, 0x68, 0x3f, 0x2f, 0x3e, 0x2d,
0x6e, 0x2d, 0x4e, 0xec, 0xc2, 0xe8, 0x22, 0x16,
0x6d, 0x11, 0x91, 0x44, 0x3d, 0x6c, 0x41, 0x5f,
0xf8, 0x08, 0x32, 0xb4, 0x99, 0xe2, 0x34, 0xef,
0x2a, 0xe0, 0x57, 0x69, 0x10, 0x95, 0x96, 0x7e,
0xc2, 0xe5, 0x6a, 0x85, 0xcd, 0x8d, 0x9b, 0x3a,
0x9e, 0x2c, 0x7e, 0xdb, 0x99, 0xc0, 0x3a, 0x91,
0xc8, 0x6c, 0x45, 0x61, 0x4f, 0x79, 0x51, 0x79,
0x5a, 0xa8, 0xe3, 0x6a, 0x3e, 0x79, 0xe8, 0x00,
0x5e, 0x52, 0x85, 0x2b, 0xdf, 0x20, 0x66, 0x7d,
0x4d, 0xe4, 0x58, 0xe6, 0xa4, 0x92, 0x77, 0x6d,
0xff, 0xbd, 0xce, 0x4e, 0x36, 0x1f, 0xc7, 0x90,
0xc8, 0xaa, 0xfa, 0x06, 0x24, 0xe2, 0x06, 0x82,
0x35, 0x8c, 0xae, 0x14, 0xac, 0x14, 0x92, 0xf9,
0xf8, 0xea, 0xdf, 0x9d, 0x7d, 0x57, 0x0a, 0x7c,
0x14, 0xd8, 0xca, 0x4a, 0xf8, 0x91, 0xdb, 0xc0,
0x3c, 0xd5, 0xc6, 0x60, 0xb8, 0xcc, 0xe2, 0xed,
0x58, 0x90, 0x01, 0x05, 0xa4, 0x93, 0xfe, 0x9d,
0x7e, 0xde, 0x3a, 0xfb, 0x35, 0x44, 0x77, 0x49,
0x1c, 0x41, 0x93, 0x14, 0xd2, 0x6e, 0xd4, 0x0e,
0x44, 0x9a, 0x6e, 0xfc, 0x67, 0x51, 0xe9, 0xbf,
0xe1, 0xea, 0xc4, 0x86, 0x7e, 0xc3, 0x23, 0xfc,

```
                        0xa1, 0x5d, 0xf7, 0xd6, 0xa1, 0x6e, 0x1f, 0xbd,
                        0xaf, 0xb2, 0xd2, 0x81, 0x21, 0xa6, 0x90, 0x65,
                        0x41, 0xfe, 0x61, 0xa8, 0x4f, 0x4a, 0x67, 0x31,
                        0x34, 0x2c, 0xb7, 0xb2, 0xef, 0xda, 0xae, 0x90,
                        0x37, 0xa5, 0x66, 0xd8, 0x13, 0x85, 0x95, 0xc2,
                        0x37, 0x67, 0x44, 0x58, 0x0e, 0xd4, 0xbd, 0x4f,
                        0xd2, 0x1e, 0xf7, 0x22, 0x68, 0x5e, 0x53, 0x9d,
                        0x8a, 0x0a, 0x4f, 0x79, 0xe4, 0xfe, 0x09, 0x1b,
                        0xa3, 0x6f, 0xf3, 0xb7, 0xf4, 0x88, 0x79, 0x2c,
                        0xf0, 0xbd, 0x84, 0xfe, 0x91, 0x42, 0x4d, 0x64,
                        0x60, 0x44, 0x86, 0xc9, 0xa2, 0xd9, 0x66, 0x2d,
                        0xe3, 0xb5, 0xa6, 0xc7, 0xb3, 0xb0, 0xe2, 0x57,
                        0x1f, 0xd5, 0x0e, 0x14, 0x5d, 0x87, 0x40, 0x4d,
                        0x45, 0xc4, 0x4b, 0xd6, 0x06, 0x98, 0x3a, 0x67,
                        0xdc, 0xc0, 0x30, 0x7f, 0x99, 0x96, 0xac, 0x7c,
                        0x4b, 0x52, 0x43, 0xff, 0x02, 0x25, 0x56, 0x22,
                        0xfa, 0x64, 0x36, 0x58, 0xeb, 0x76, 0xa5, 0x30
ENDFIELD(PKT_UDP_IN_307, 1504)


/* IPGEN: processing input file "ip308.packets" */

FIELD(PKT_UDP_IN_308)
/* incoming UDP packet, length < IP header length */
                        0x90, 0x00, 0x00, 0x00,
                        /* IP proto 17 len 256 10.11.12.14 -> 10.11.12.13 */
                        0x45, 0x00, 0x01, 0x00, 0x11, 0xc6, 0x00, 0x00,
                        0x40, 0x11, 0x3b, 0xf7, 0x0a, 0x0b, 0x0c, 0x0e,
                        0x0a, 0x0b
                        /* UDP sport 1201 dport 9 ulen 8 */
ENDFIELD(PKT_UDP_IN_308, 22)


/* IPGEN: processing input file "ip309.packets" */

FIELD(PKT_UDP_IN_309)
/* incoming UDP packet, length < length of IP header + UPD header */
                        0xd8, 0x00, 0x00, 0x00,
                        /* IP proto 17 len 27 10.11.12.14 -> 10.11.12.13 */
                        0x45, 0x00, 0x00, 0x1b, 0x11, 0xc6, 0x00, 0x00,
                        0x40, 0x11, 0x3c, 0xdc, 0x0a, 0x0b, 0x0c, 0x0e,
                        0x0a, 0x0b, 0x0c, 0x0d,
                        /* UDP sport 1201 dport 9 ulen 8 */
                        0x04, 0xb1, 0x00, 0x09, 0x00, 0x08, 0xce
ENDFIELD(PKT_UDP_IN_309, 31)
```

```
/* IPGEN: processing input file "ip310.packets" */

FIELD(PKT_UDP_IN_310)
/* incoming UDP packet, 8 octets of (fake) options */
                        0x00, 0x08, 0x00, 0x00,
                        /* IP proto 17 len 256 10.11.12.14 -> 10.11.12.13 */
                        0x47, 0x00, 0x01, 0x00, 0x11, 0xc6, 0x00, 0x00,
                        0x40, 0x11, 0xc6, 0x1c, 0x0a, 0x0b, 0x0c, 0x0e,
                        0x0a, 0x0b, 0x0c, 0x0d, 0x3a, 0xf1, 0x07, 0x41,
                        0x89, 0x41, 0xa8, 0x66,
                        /* UDP sport 1201 dport 9 ulen 8 */
                        0x04, 0xb1, 0x00, 0x09, 0x00, 0x08, 0xce, 0xf3,
                        /* data */
                        0x02, 0xd8, 0xe5, 0x9b, 0x6e, 0x91, 0x18, 0xb9,
                        0xe3, 0x5b, 0xb8, 0xe6, 0x81, 0x0e, 0x08, 0x7b,
                        0x72, 0x3e, 0xd3, 0x5e, 0xb4, 0x79, 0x8e, 0xee,
                        0x6a, 0x95, 0x2f, 0xf3, 0xd7, 0xd7, 0x59, 0xd9,
                        0xaf, 0x3e, 0x74, 0x1d, 0xcf, 0x8c, 0xd7, 0xb3,
                        0xe8, 0x8f, 0x99, 0x69, 0x9e, 0xa1, 0xe4, 0x10,
                        0xdf, 0xb8, 0x6e, 0x93, 0x31, 0xfd, 0x81, 0x9b,
                        0x92, 0xb1, 0x8e, 0x69, 0x88, 0xe8, 0x42, 0x38,
                        0x26, 0xb7, 0x55, 0xf6, 0x43, 0x2c, 0xa9, 0x2b,
                        0xbc, 0x42, 0x94, 0x5a, 0xe3, 0x79, 0x6a, 0xc2,
                        0x31, 0xd9, 0x55, 0x62, 0xd6, 0xd6, 0xfd, 0x68,
                        0x87, 0x8b, 0xd2, 0x10, 0x73, 0x14, 0x48, 0x9a,
                        0xcb, 0x9d, 0x90, 0x0f, 0xca, 0x39, 0x3a, 0x86,
                        0x7b, 0xcf, 0xe0, 0x5e, 0x48, 0x4a, 0x20, 0x79,
                        0x23, 0x75, 0xdb, 0xf9, 0x4b, 0xd8, 0x62, 0xd3,
                        0x63, 0x34, 0xe3, 0xd7, 0x48, 0x2b, 0x71, 0x14,
                        0xc8, 0x01, 0x23, 0x92, 0x3a, 0x5d, 0x18, 0xb5,
                        0x2c, 0xf8, 0x13, 0x74, 0x43, 0x33, 0xed, 0x66,
                        0xa8, 0xc8, 0x60, 0xf3, 0xa0, 0xc2, 0xc6, 0x04,
                        0xf6, 0xa9, 0xdb, 0x3e, 0xd4, 0x4c, 0x52, 0x9d,
                        0x4d, 0x75, 0x2f, 0x87, 0xd3, 0x48, 0x3c, 0xff,
                        0x40, 0x4f, 0x74, 0x83, 0x82, 0x61, 0xea, 0x2a,
                        0x2a, 0x4a, 0x1d, 0xca, 0x0c, 0xe4, 0xce, 0x02,
                        0x8d, 0xa9, 0x40, 0x62, 0xf5, 0x93, 0xff, 0x42,
                        0x08, 0x2e, 0xc9, 0xdb, 0x76, 0x05, 0xdb, 0xb7,
                        0x54, 0x4f, 0x3a, 0xd6, 0xb0, 0x24, 0x00, 0xda,
                        0x6e, 0x1e, 0xa5, 0x7a, 0x02, 0x73, 0x7c, 0x8f,
                        0x1d, 0xbd, 0xf1, 0x12
ENDFIELD(PKT_UDP_IN_310, 260)


/* IPGEN: processing input file "ip311.packets" */

FIELD(PKT_UDP_IN_311)
```

```
/* incoming UDP packet, middle fragment */
                        0x00, 0x08, 0x00, 0x00,
                        /* IP proto 17 len 256 10.11.12.14 -> 10.11.12.13 */
                        0x47, 0x00, 0x01, 0x00, 0x00, 0x00, 0x20, 0x22,
                        0x40, 0x11, 0x31, 0x9f, 0x0a, 0x0b, 0x0c, 0x0e,
                        0x0a, 0x0b, 0x0c, 0x0d, 0x50, 0xf0, 0x55, 0x58,
                        0x1f, 0x1e, 0x34, 0x95,
                        /* UDP sport 1201 dport 9 ulen 8 */
                        0x04, 0xb1, 0x00, 0x09, 0x00, 0x08, 0xce, 0xf3,
                        /* data */
                        0x24, 0x0f, 0x4c, 0x78, 0x5e, 0x87, 0x4f, 0x0e,
                        0xab, 0x4f, 0xe9, 0x1a, 0x6d, 0x8e, 0x94, 0x6f,
                        0x01, 0x11, 0xff, 0x1e, 0xce, 0xf0, 0x31, 0x1e,
                        0xe1, 0x86, 0x76, 0x00, 0xa4, 0xaa, 0x95, 0xc8,
                        0xb9, 0xe2, 0x41, 0x17, 0x69, 0x90, 0x26, 0x14,
                        0xdf, 0x0f, 0x2e, 0x4d, 0x9d, 0xc3, 0xbc, 0x9e,
                        0xd4, 0xbb, 0xbd, 0xa2, 0xac, 0xee, 0xc0, 0x8d,
                        0x74, 0x36, 0x8d, 0x18, 0xe1, 0x22, 0xe1, 0x9a,
                        0x04, 0x22, 0xb2, 0x6d, 0xb2, 0xd8, 0x82, 0x91,
                        0xe7, 0xb0, 0xde, 0x84, 0x73, 0x9b, 0x22, 0x47,
                        0x56, 0xdf, 0xe9, 0x02, 0xcd, 0xa9, 0x8f, 0x41,
                        0xe0, 0x1c, 0x5a, 0xc1, 0x3f, 0x3b, 0x5b, 0x43,
                        0x5d, 0x0d, 0xb1, 0x0f, 0xe5, 0x33, 0xa0, 0xcc,
                        0xe3, 0x7f, 0x50, 0x57, 0x1a, 0x73, 0x9e, 0x70,
                        0x52, 0x88, 0x73, 0x20, 0x31, 0x02, 0x61, 0x11,
                        0x1f, 0xbb, 0xd2, 0x5e, 0xf6, 0x2e, 0xa1, 0x53,
                        0x3b, 0x52, 0x62, 0x21, 0x85, 0x03, 0xed, 0x69,
                        0x82, 0x3e, 0xc0, 0x9c, 0xb1, 0x5e, 0x0c, 0x03,
                        0xe6, 0x7f, 0x23, 0x18, 0x82, 0x85, 0x29, 0xa1,
                        0x40, 0xfc, 0xff, 0x37, 0x2a, 0xa0, 0x8a, 0x65,
                        0xf3, 0xed, 0x86, 0x78, 0xf0, 0x74, 0xe1, 0x72,
                        0xb2, 0xa1, 0x0e, 0x63, 0x00, 0x1a, 0x66, 0xe6,
                        0x9a, 0x8a, 0xfe, 0x1c, 0x0f, 0x28, 0xbd, 0x4f,
                        0x24, 0xbc, 0x86, 0x4e, 0x5c, 0x11, 0xb3, 0x4f,
                        0xfe, 0x3a, 0xc8, 0xee, 0xae, 0xa9, 0x60, 0x60,
                        0x4b, 0x6e, 0xc3, 0x4b, 0x88, 0x29, 0x31, 0x22,
                        0xb3, 0x30, 0x3e, 0xc2, 0x58, 0xfb, 0x12, 0x7c,
                        0xb7, 0x98, 0xca, 0x14
ENDFIELD(PKT_UDP_IN_311, 260)


/* IPGEN: processing input file "ip312.packets" */

FIELD(PKT_UDP_IN_312)
/* incoming UDP packet, last fragment */
                        0x00, 0x08, 0x00, 0x00,
                        /* IP proto 17 len 256 10.11.12.14 -> 10.11.12.13 */
```

```
                        0x47, 0x00, 0x01, 0x00, 0x00, 0x00, 0x00, 0x78,
                        0x40, 0x11, 0xf1, 0x8e, 0x0a, 0x0b, 0x0c, 0x0e,
                        0x0a, 0x0b, 0x0c, 0x0d, 0xa9, 0x7d, 0x63, 0xa7,
                        0xb7, 0x2b, 0x95, 0x65,
                        /* UDP sport 1201 dport 9 ulen 8 */
                        0x04, 0xb1, 0x00, 0x09, 0x00, 0x08, 0xce, 0xf3,
                        /* data */
                        0xd5, 0xf5, 0xc5, 0x20, 0x63, 0x88, 0x6b, 0xec,
                        0xb2, 0x9c, 0x0e, 0x65, 0xcc, 0x4d, 0x28, 0x24,
                        0x48, 0x3a, 0xa0, 0x00, 0xd2, 0x6a, 0x14, 0x7c,
                        0xe8, 0x77, 0x23, 0x9f, 0xa3, 0xb9, 0x05, 0x78,
                        0xae, 0xca, 0x98, 0x12, 0x53, 0x03, 0xfe, 0x05,
                        0x9f, 0x0c, 0x6a, 0x6c, 0x59, 0x92, 0x90, 0xa2,
                        0xcc, 0x31, 0xa2, 0x9f, 0x9b, 0xb6, 0x1b, 0x83,
                        0x2d, 0x3e, 0x23, 0xd0, 0xf7, 0x28, 0x48, 0xa6,
                        0xf2, 0xe0, 0xb8, 0x45, 0xe3, 0xb6, 0x4a, 0x83,
                        0xc2, 0xb5, 0xef, 0x1c, 0x47, 0x7f, 0xbe, 0x14,
                        0xb0, 0x60, 0xb3, 0x4c, 0x16, 0xce, 0xcf, 0x43,
                        0x0c, 0xf2, 0x14, 0x04, 0x1a, 0x5c, 0xaa, 0x0d,
                        0x3d, 0x62, 0x52, 0x20, 0x18, 0x9d, 0xa3, 0xda,
                        0x52, 0x92, 0xf6, 0x99, 0x12, 0xb4, 0xad, 0xc2,
                        0x14, 0x60, 0x0e, 0x2a, 0x2e, 0xde, 0x6e, 0x3b,
                        0xd0, 0x82, 0x3f, 0xeb, 0xde, 0xe9, 0xf8, 0x1b,
                        0x4b, 0x4a, 0x3c, 0x63, 0xe7, 0xdf, 0x3d, 0x39,
                        0x72, 0x34, 0xd3, 0x84, 0xe8, 0x80, 0x46, 0xfd,
                        0xe1, 0x55, 0x27, 0x0f, 0x33, 0x95, 0x4a, 0x03,
                        0x17, 0x89, 0xee, 0xf6, 0x72, 0xe6, 0x11, 0xbd,
                        0x31, 0x4d, 0x20, 0x18, 0x2d, 0x5e, 0x52, 0x9f,
                        0x92, 0x25, 0x23, 0x7a, 0xa5, 0x69, 0x77, 0x86,
                        0xbe, 0x9f, 0x96, 0xf1, 0x34, 0xe0, 0xf5, 0x4c,
                        0x6a, 0xe3, 0x42, 0xdc, 0xca, 0x53, 0x9a, 0xfb,
                        0xa1, 0xba, 0x13, 0xce, 0x18, 0x65, 0x6d, 0xaa,
                        0x8a, 0x90, 0x25, 0x30, 0xf9, 0x9c, 0xb6, 0xb8,
                        0x3b, 0x4c, 0xa9, 0x70, 0x2d, 0x9e, 0xbc, 0x97,
                        0x82, 0xfe, 0x73, 0x4c
ENDFIELD(PKT_UDP_IN_312, 260)

FIELD(PKT_UDP_IN_314)
/* incoming UDP packet, first fragment */
                        0x00, 0x08, 0x00, 0x00,
                        /* IP proto 17 len 256 10.11.12.14 -> 10.11.12.13 */
                        0x47, 0x00, 0x01, 0x00, 0x00, 0x01, 0x20, 0x00,
                        0x40, 0x11, 0x09, 0x81, 0x0a, 0x0b, 0x0c, 0x0e,
                        0x0a, 0x0b, 0x0c, 0x0d, 0x51, 0x0d, 0x47, 0xf2,
                        0xc8, 0x5a, 0xc0, 0xe0,
                        /* UDP sport 1201 dport 9 ulen 8 */
                        0x04, 0xb1, 0x00, 0x09, 0x00, 0x08, 0xce, 0xf3,
```

```
                        /* data */
                        0xc0, 0x2d, 0x8b, 0x4a, 0xbd, 0xb0, 0x7a, 0xb7,
                        0x4c, 0x31, 0x6f, 0x88, 0x7d, 0x18, 0xf8, 0xaa,
                        0xb7, 0xb4, 0x41, 0x39, 0xb2, 0xb5, 0x85, 0x03,
                        0xc2, 0xcc, 0xf6, 0x8a, 0x26, 0xb6, 0x6b, 0xe6,
                        0xe4, 0xf6, 0x31, 0xa1, 0xa6, 0xab, 0x58, 0xf2,
                        0xdc, 0xc7, 0x7a, 0x5a, 0xe0, 0x72, 0x04, 0x97,
                        0x26, 0x46, 0xd0, 0xd8, 0xfb, 0x55, 0xdc, 0xbd,
                        0x21, 0xd2, 0x48, 0x47, 0x88, 0xb3, 0x2e, 0x6c,
                        0xa9, 0x5f, 0x0e, 0x4f, 0x0a, 0x66, 0x41, 0xe7,
                        0x2e, 0xbc, 0x41, 0x0e, 0x2e, 0x45, 0xa5, 0x55,
                        0x8b, 0x75, 0x2d, 0x86, 0xca, 0x09, 0x44, 0xeb,
                        0xdb, 0x8c, 0x32, 0x64, 0x3f, 0x60, 0xd0, 0xe8,
                        0xbf, 0xde, 0x37, 0xca, 0x45, 0x78, 0xb1, 0x73,
                        0x34, 0xf2, 0x81, 0x63, 0x37, 0x26, 0xb8, 0xc3,
                        0x9b, 0xe5, 0x49, 0x65, 0xef, 0x8d, 0x50, 0xca,
                        0x19, 0x82, 0x2e, 0x58, 0xe3, 0xff, 0x40, 0xa2,
                        0xdd, 0x77, 0x6c, 0x22, 0xf0, 0x1d, 0x95, 0x24,
                        0x0f, 0x16, 0x87, 0x47, 0x3c, 0x3f, 0x0a, 0xd7,
                        0x25, 0x53, 0x3c, 0x14, 0xe1, 0x8c, 0xde, 0xfa,
                        0x0f, 0x0d, 0x53, 0xf2, 0x0c, 0x93, 0x94, 0xe9,
                        0x0b, 0x01, 0x0c, 0xfb, 0x1e, 0xa1, 0x1f, 0x2e,
                        0xb8, 0xa7, 0x75, 0xf4, 0xe6, 0x7f, 0xcc, 0x0b,
                        0xd2, 0x08, 0x1f, 0xb3, 0x95, 0xfe, 0xae, 0xa4,
                        0x0b, 0x01, 0x96, 0x17, 0x94, 0x2a, 0x00, 0x9f,
                        0x2b, 0x0c, 0x9a, 0x4a, 0xae, 0xba, 0x78, 0x66,
                        0x61, 0xed, 0x5a, 0x47, 0x6c, 0x26, 0x53, 0x3e,
                        0x2f, 0x72, 0xf2, 0xc4, 0x70, 0xa0, 0x68, 0x7b,
                        0xa1, 0xfe, 0x92, 0x35
ENDFIELD(PKT_UDP_IN_314, 260)

/* IPGEN: processing input file "ip313.packets" */

FIELD(PKT_IP_IN_FIRST_FRAGM)
/* incoming UDP packet, first fragment */
                        0xa0, 0x01, 0x00, 0x00,
                        /* IP proto 17 len 52 10.11.12.14 -> 10.11.12.13 */
                        0x45, 0x00, 0x00, 0x34, 0x00, 0x00, 0x20, 0x00,
                        0x40, 0x11, 0x2e, 0x89, 0x0a, 0x0b, 0x0c, 0x0e,
                        0x0a, 0x0b, 0x0c, 0x0d,
                        /* Start segment 1 */
                        /* 28, UDP sport 1201 dport 9 ulen 8 */
                        0x04, 0xb1, 0x00, 0x09, 0x00, 0x08, 0xce, 0xf3,
                        0xc0, 0x2d, 0x8b, 0x4a, 0xbd, 0xb0, 0x7a, 0xb7,
                        0x4c, 0x31, 0x6f, 0x88, 0x7d, 0x18, 0xf8, 0xaa,
                        0xb7, 0xb4, 0x41, 0x39,  0xb2, 0xb5, 0x85, 0x03
 ENDFIELD(PKT_IP_IN_FIRST_FRAGM, 56)
```

```
FIELD(PKT_IP_IN_SECOND_FRAGM)
/* incoming UDP packet, first fragment */
                        0xa0, 0x01, 0x00, 0x00,
                        /* IP proto 17 len 52 10.11.12.14 -> 10.11.12.13 */
                        0x45, 0x00, 0x00, 0x34, 0x00, 0x00, 0x20, 0x04,
                        0x40, 0x11, 0x2e, 0x85, 0x0a, 0x0b, 0x0c, 0x0e,
                        0x0a, 0x0b, 0x0c, 0x0d,
                        /* Segment 2 */
                        0x04, 0xb1, 0x00, 0x09, 0x00, 0x08, 0xce, 0xf3,
                        0xc0, 0x2d, 0x8b, 0x4a, 0xbd, 0xb0, 0x7a, 0xb7,
                        0x4c, 0x31, 0x6f, 0x88, 0x7d, 0x18, 0xf8, 0xaa,
                        0xb7, 0xb4, 0x41, 0x39,  0xb2, 0xb5, 0x85, 0x03
 ENDFIELD(PKT_IP_IN_SECOND_FRAGM, 56)

FIELD(PKT_IP_IN_LAST_FRAGM)
/* incoming UDP packet, last fragment */
                        0xa0, 0x01, 0x00, 0x00,
                        0x45, 0x00, 0x00, 0x34, 0x00, 0x00, 0x00, 0x08,
                        0x40, 0x11, 0x4e, 0x81, 0x0a, 0x0b, 0x0c, 0x0e,
                        0x0a, 0x0b, 0x0c, 0x0d,
                        0x04, 0xb1, 0x00, 0x09, 0x00, 0x08, 0xce, 0xf3,
                        0xc0, 0x2d, 0x8b, 0x4a, 0xbd, 0xb0, 0x7a, 0xb7,
                        0x4c, 0x31, 0x6f, 0x88, 0x7d, 0x18, 0xf8, 0xaa,
                        0xb7, 0xb4, 0x41, 0x39,  0xb2, 0xb5, 0x85, 0x03
ENDFIELD(PKT_IP_IN_LAST_FRAGM, 56)

FIELD(PKT_IP_OUT_FRAGM)
/* Fragment after assemble  */
                        0xa0, 0x03, 0x00, 0x00,
                        0x45, 0x00, 0x00, 0x74, 0x00, 0x00, 0x00, 0x00,
                        0x40, 0x11, 0x4e, 0x49, 0x0a, 0x0b, 0x0c, 0x0e,
                        0x0a, 0x0b, 0x0c, 0x0d,
                        /* Segment 1*/
                        0x04, 0xb1, 0x00, 0x09, 0x00, 0x08, 0xce, 0xf3,
                        0xc0, 0x2d, 0x8b, 0x4a, 0xbd, 0xb0, 0x7a, 0xb7,
                        0x4c, 0x31, 0x6f, 0x88, 0x7d, 0x18, 0xf8, 0xaa,
                        0xb7, 0xb4, 0x41, 0x39,  0xb2, 0xb5, 0x85, 0x03,
                        /* Segment 2 */
                        0x04, 0xb1, 0x00, 0x09, 0x00, 0x08, 0xce, 0xf3,
                        0xc0, 0x2d, 0x8b, 0x4a, 0xbd, 0xb0, 0x7a, 0xb7,
                        0x4c, 0x31, 0x6f, 0x88, 0x7d, 0x18, 0xf8, 0xaa,
                        0xb7, 0xb4, 0x41, 0x39,  0xb2, 0xb5, 0x85, 0x03,
                        /* Segment 3 */
                        0x04, 0xb1, 0x00, 0x09, 0x00, 0x08, 0xce, 0xf3,
                        0xc0, 0x2d, 0x8b, 0x4a, 0xbd, 0xb0, 0x7a, 0xb7,
                        0x4c, 0x31, 0x6f, 0x88, 0x7d, 0x18, 0xf8, 0xaa,
```

```
                        0xb7, 0xb4, 0x41, 0x39,  0xb2, 0xb5, 0x85, 0x03
ENDFIELD(PKT_IP_OUT_FRAGM, 120)

FIELD(PKT_ICMP_REASS_TIMER_317)
/* Reassemble-timer not correct  */
                        0xc0, 0x01, 0x00, 0x00,
                        /* IP proto 1 len 56 10.11.12.13 -> 10.11.12.14 */
                        0x45, 0x00, 0x00, 0x38, 0x00, 0x00, 0x00, 0x00,
                        0x40, 0x01, 0x4e, 0x95, 0x0a, 0x0b, 0x0c, 0x0d,
                        0x0a, 0x0b, 0x0c, 0x0e,
                        /* ICMP type 11 code 1 */
                        0x0b, 0x01, 0x21, 0x49, 0x00, 0x00, 0x00, 0x00,
                        /* data */
                        0x45, 0x00, 0x00, 0x34, 0x00, 0x00, 0x20, 0x00,
                        0x40, 0x11, 0x2e, 0x89, 0x0a, 0x0b, 0x0c, 0x0e,
                        0x0a, 0x0b, 0x0c, 0x0d,
                        /* 28, UDP sport 1201 dport 9 ulen 8  */
                        0x04, 0xb1, 0x00, 0x09, 0x00, 0x08, 0xce, 0xf3
ENDFIELD(PKT_ICMP_REASS_TIMER_317, 60)

/* IPGEN: processing input file "ip400.packets" */

FIELD(PKT_UDP_NODATA_OUT_400)
/* outgoing UDP packet with no data bytes UDP -> IP */
                        0xe0, 0x00, 0x00, 0x00,
                        /* IP proto 17 len 0 10.11.12.13 -> 10.11.12.14 */
                        0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
                        0x00, 0x11, 0x00, 0x00, 0x0a, 0x0b, 0x0c, 0x0d,
                        0x0a, 0x0b, 0x0c, 0x0e,
                        /* UDP sport 1201 dport 9 ulen 8 */
                        0x04, 0xb1, 0x00, 0x09, 0x00, 0x08, 0xce, 0xf3
ENDFIELD(PKT_UDP_NODATA_OUT_400, 32)


/* IPGEN: processing input file "ip402.packets" */

FIELD(PKT_UDP_NODATA_IN_402)
/* incoming UDP packet with no data bytes */
                        0xe0, 0x00, 0x00, 0x00,
                        /* IP proto 17 len 28 10.11.12.14 -> 10.11.12.13 */
                        0x45, 0x00, 0x00, 0x1c, 0x11, 0xc6, 0x00, 0x00,
                        0x40, 0x11, 0x3c, 0xdb, 0x0a, 0x0b, 0x0c, 0x0e,
                        0x0a, 0x0b, 0x0c, 0x0d,
                        /* UDP sport 1201 dport 9 ulen 8 */
                        0x04, 0xb1, 0x00, 0x09, 0x00, 0x08, 0xce, 0xf3
ENDFIELD(PKT_UDP_NODATA_IN_402, 32)
```

```
/* IPGEN: processing input file "ip404.packets" */

FIELD(PKT_UDP_NODATA_OUT_404)
/* outgoing UDP packet with no data bytes UDP -> IP */
                        0xe0, 0x00, 0x00, 0x00,
                        /* IP proto 17 len 0 10.11.12.13 -> 10.11.12.14 */
                        0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
                        0x00, 0x11, 0x00, 0x00, 0x0a, 0x0b, 0x0c, 0x0d,
                        0x0a, 0x0b, 0x0c, 0x0e,
                        /* UDP sport 1201 dport 9 ulen 8 */
                        0x04, 0xb1, 0x00, 0x09, 0x00, 0x08, 0xce, 0xf3
ENDFIELD(PKT_UDP_NODATA_OUT_404, 32)


/* IPGEN: processing input file "ip406.packets" */

FIELD(PKT_UDP_NODATA_IN_406)
/* incoming UDP packet with no data bytes */
                        0xe0, 0x00, 0x00, 0x00,
                        /* IP proto 17 len 28 10.11.12.14 -> 10.11.12.13 */
                        0x45, 0x00, 0x00, 0x1c, 0x11, 0xc6, 0x00, 0x00,
                        0x40, 0x11, 0x3c, 0xdb, 0x0a, 0x0b, 0x0c, 0x0e,
                        0x0a, 0x0b, 0x0c, 0x0d,
                        /* UDP sport 1201 dport 9 ulen 8 */
                        0x04, 0xb1, 0x00, 0x09, 0x00, 0x08, 0xce, 0xf3
ENDFIELD(PKT_UDP_NODATA_IN_406, 32)

/* ICMP packet with wrong destination address */
FIELD(ICMP_WRONG_DEST)
        0xa0, 0x02, 0x00, 0x00,
        0x45,0x00,0x00,0x54,0x5c,0x00,0x00,0x00,0x80,0x01,0x4f,0x49,0xc6,0xa8,0x01,0x0a,
        0xc6,0xa8,0x01,0x05,0x00,0x00,0xd8,0xb8,0x69,0x09,0x00,0x00,0x39,0xcb,0x3e,0x1f,
        0x00,0x04,0x5b,0x4c,0x08,0x09,0x0a,0x0b,0x0c,0x0d,0x0e,0x0f,0x10,0x11,0x12,0x13,
        0x14,0x15,0x16,0x17,0x18,0x19,0x1a,0x1b,0x1c,0x1d,0x1e,0x1f,0x20,0x21,0x22,0x23,
        0x24,0x25,0x26,0x27,0x28,0x29,0x2a,0x2b,0x2c,0x2d,0x2e,0x2f,0x30,0x31,0x32,0x33,
        0x34,0x35,0x36,0x37
ENDFIELD(ICMP_WRONG_DEST, 88)
/* ICMP-Message*/
FIELD(IP_BRODCAST_SRC_ADDR)
        0xa0, 0x02, 0x00, 0x00,
        0x45,0x00,0x00,0x54,0x1c,0x38,0x00,0x00,0x40,0x01,0x96,0xc4,0xff,0xff,0xff,0xff,
        0xc6,0xa8,0x01,0x05,0x08,0x00,0x96,0xbe,0xb5,0x0d,0x02,0x2e,0x39,0xcb,0x66,0xc0,
        0x00,0x00,0x1e,0x77,0x08,0x09,0x0a,0x0b,0x0c,0x0d,0x0e,0x0f,0x10,0x11,0x12,0x13,
        0x14,0x15,0x16,0x17,0x18,0x19,0x1a,0x1b,0x1c,0x1d,0x1e,0x1f,0x20,0x21,0x22,0x23,
        0x24,0x25,0x26,0x27,0x28,0x29,0x2a,0x2b,0x2c,0x2d,0x2e,0x2f,0x30,0x31,0x32,0x33,
        0x34,0x35,0x36,0x37
```

```
ENDFIELD(IP_BRODCAST_SRC_ADDR, 88)

/* ICMP-Message */
FIELD(IP_BRODCAST_DST_ADDR)
        0xa0, 0x02, 0x00, 0x00,
        0x45,0x00,0x00,0x54,0x1c,0x38,0x00,0x00,0x40,0x01,0x96,0xc4,0xc6,0xa8,0x01,0x05,
        0xff,0xff,0xff,0xff, 0x08,0x00,0x96,0xbe,0xb5,0x0d,0x02,0x2e,0x39,0xcb,0x66,0xc0,
        0x00,0x00,0x1e,0x77,0x08,0x09,0x0a,0x0b,0x0c,0x0d,0x0e,0x0f,0x10,0x11,0x12,0x13,
        0x14,0x15,0x16,0x17,0x18,0x19,0x1a,0x1b,0x1c,0x1d,0x1e,0x1f,0x20,0x21,0x22,0x23,
        0x24,0x25,0x26,0x27,0x28,0x29,0x2a,0x2b,0x2c,0x2d,0x2e,0x2f,0x30,0x31,0x32,0x33,
        0x34,0x35,0x36,0x37
ENDFIELD(IP_BRODCAST_DST_ADDR, 88)

/* UDP and IP - Datagram */
FIELD(UDP_IP_BRODCAST_DST_ADDR)
        0x70, 0x02, 0x00, 0x00,
        0x45,0x00,0x00,0x4e,0x12,0x58,0x00,0x00,0x40,0x11,0xa0,0x9a,0xc6,0xa8,0x01,0x05,
        0xff,0xff,0xff,0xff,0x00,0x89,0x00,0x89,0x00,0x3a,0x53,0x16,0x7a,0x2e,0x01,0x10,
        0x00,0x01,0x00,0x00,0x00,0x00,0x00,0x00,0x20,0x45,0x47,0x45,0x4d,0x45,0x42,0x46,
        0x45,0x45,0x49,0x46,0x46,0x46,0x44,0x43,0x41,0x43,0x41,0x43,0x41,0x43,0x41,0x43,
        0x41,0x43,0x41,0x43,0x41,0x43,0x41,0x42,0x4e,0x00,0x00,0x20,0x00,0x01
ENDFIELD(UDP_IP_BRODCAST_DST_ADDR, 82)

/* IPGEN: processing input file "ip500.packets" */

FIELD(PKT_UDP_NODATA_UNREACH_500)
/* incoming UDP packet with wrong dest address */
                0xe0, 0x00, 0x00, 0x00,
                /* IP proto 17 len 28 10.11.12.14 -> 10.11.12.15 */
                0x45, 0x00, 0x00, 0x1c, 0x11, 0xc6, 0x00, 0x00,
                0x40, 0x11, 0x3c, 0xd9, 0x0a, 0x0b, 0x0c, 0x0e,
                0x0a, 0x0b, 0x0c, 0x0f,
                /* UDP sport 1201 dport 9 ulen 8 */
                0x04, 0xb1, 0x00, 0x09, 0x00, 0x08, 0xce, 0xf1
ENDFIELD(PKT_UDP_NODATA_UNREACH_500, 32)

FIELD(PKT_ICMP_UNREACH_500)
/* ICMP unreachable response for PKT_UDP_NODATA_UNREACH */
                0xc0, 0x01, 0x00, 0x00,
                /* IP proto 1 len 56 10.11.12.13 -> 10.11.12.14 */
                0x45, 0x00, 0x00, 0x38, 0x00, 0x00, 0x00, 0x00,
                0x40, 0x01, 0x4e, 0x95, 0x0a, 0x0b, 0x0c, 0x0d,
                0x0a, 0x0b, 0x0c, 0x0e,
                /* ICMP type 3 code 1 */
                0x03, 0x01, 0x29, 0x4b, 0x00, 0x00, 0x00, 0x00,
                /* data */
                0x45, 0x00, 0x00, 0x1c, 0x11, 0xc6, 0x00, 0x00,
```

```
                        0x40, 0x11, 0x3c, 0xd9, 0x0a, 0x0b, 0x0c, 0x0e,
                        0x0a, 0x0b, 0x0c, 0x0f, 0x04, 0xb1, 0x00, 0x09,
                        0x00, 0x08, 0xce, 0xf1
ENDFIELD(PKT_ICMP_UNREACH_500, 60)


/* IPGEN: processing input file "ip501.packets" */

FIELD(ICMP_ECHO_REQUEST_501)
                        0xa0, 0x02, 0x00, 0x00,
                        /* IP proto 1 len 84 172.20.48.48 -> 172.20.78.9 */
                        0x45, 0x00, 0x00, 0x54, 0x8b, 0xab, 0x00, 0x00,
                        0xff, 0x01, 0x59, 0x9b, 0xac, 0x14, 0x30, 0x30,
                        0xac, 0x14, 0x4e, 0x09,
                        /* ICMP type 8 code 0 */
                        0x08, 0x00, 0xdb, 0x07, 0x1a, 0x34, 0x00, 0x11,
                        /* data */
                        0x42, 0xac, 0x3f, 0x39, 0x93, 0xca, 0x02, 0x00,
                        0x08, 0x09, 0x0a, 0x0b, 0x0c, 0x0d, 0x0e, 0x0f,
                        0x10, 0x11, 0x12, 0x13, 0x14, 0x15, 0x16, 0x17,
                        0x18, 0x19, 0x1a, 0x1b, 0x1c, 0x1d, 0x1e, 0x1f,
                        0x20, 0x21, 0x22, 0x23, 0x24, 0x25, 0x26, 0x27,
                        0x28, 0x29, 0x2a, 0x2b, 0x2c, 0x2d, 0x2e, 0x2f,
                        0x30, 0x31, 0x32, 0x33, 0x34, 0x35, 0x36, 0x37
ENDFIELD(ICMP_ECHO_REQUEST_501, 88)

FIELD(ICMP_ECHO_REPLY_501)
                        0xa0, 0x02, 0x00, 0x00,
                        /* IP proto 1 len 84 172.20.78.9 -> 172.20.48.48 */
                        0x45, 0x00, 0x00, 0x54, 0x00, 0x00, 0x00, 0x00,
                        0x40, 0x01, 0xa4, 0x47, 0xac, 0x14, 0x4e, 0x09,
                        0xac, 0x14, 0x30, 0x30,
                        /* ICMP type 0 code 0 */
                        0x00, 0x00, 0xe3, 0x07, 0x1a, 0x34, 0x00, 0x11,
                        /* data */
                        0x42, 0xac, 0x3f, 0x39, 0x93, 0xca, 0x02, 0x00,
                        0x08, 0x09, 0x0a, 0x0b, 0x0c, 0x0d, 0x0e, 0x0f,
                        0x10, 0x11, 0x12, 0x13, 0x14, 0x15, 0x16, 0x17,
                        0x18, 0x19, 0x1a, 0x1b, 0x1c, 0x1d, 0x1e, 0x1f,
                        0x20, 0x21, 0x22, 0x23, 0x24, 0x25, 0x26, 0x27,
                        0x28, 0x29, 0x2a, 0x2b, 0x2c, 0x2d, 0x2e, 0x2f,
                        0x30, 0x31, 0x32, 0x33, 0x34, 0x35, 0x36, 0x37
ENDFIELD(ICMP_ECHO_REPLY_501, 88)


/* IPGEN: processing input file "ip502.packets" */
```

```
FIELD(PKT_UDP_NODATA_OUT_1_502)
/* outgoing UDP packet with no data bytes UDP -> IP */
                        0xe0, 0x00, 0x00, 0x00,
                        /* IP proto 17 len 0 10.11.12.13 -> 10.11.12.14 */
                        0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
                        0x00, 0x11, 0x00, 0x00, 0x0a, 0x0b, 0x0c, 0x0d,
                        0x0a, 0x0b, 0x0c, 0x0e,
                        /* UDP sport 1201 dport 9 ulen 8 */
                        0x04, 0xb1, 0x00, 0x09, 0x00, 0x08, 0xce, 0xf3
ENDFIELD(PKT_UDP_NODATA_OUT_1_502, 32)

FIELD(PKT_UDP_NODATA_OUT_2_502)
/* outgoing UDP packet with no data bytes IP -> PPP */
                        0xe0, 0x00, 0x00, 0x00,
                        /* IP proto 17 len 28 10.11.12.14 -> 10.11.12.13 */
                        0x45, 0x00, 0x00, 0x1c, 0x00, 0x00, 0x00, 0x00,
                        0x40, 0x11, 0x4e, 0xa1, 0x0a, 0x0b, 0x0c, 0x0d,
                        0x0a, 0x0b, 0x0c, 0x0e,
                        /* UDP sport 1201 dport 9 ulen 8 */
                        0x04, 0xb1, 0x00, 0x09, 0x00, 0x08, 0xce, 0xf3
ENDFIELD(PKT_UDP_NODATA_OUT_2_502, 32)

FIELD(PKT_UDP_NODATA_OUT_3_502)
/* Same as above, but with different ID */
                        0xe0, 0x00, 0x00, 0x00,
                        /* IP proto 17 len 28 10.11.12.14 -> 10.11.12.13 */
                        0x45, 0x00, 0x00, 0x1c, 0x00, 0x01, 0x00, 0x00,
                        0x40, 0x11, 0x4e, 0xa0, 0x0a, 0x0b, 0x0c, 0x0d,
                        0x0a, 0x0b, 0x0c, 0x0e,
                        /* UDP sport 1201 dport 9 ulen 8 */
                        0x04, 0xb1, 0x00, 0x09, 0x00, 0x08, 0xce, 0xf3
ENDFIELD(PKT_UDP_NODATA_OUT_3_502, 32)

LONG   IP_NUM_PROT  1

LONG   UDP_LINK_ID          5
LONG   PPP_LINK_ID          3

DECLARATION(IP_UDP_PROTO_ARR)
FIELD(IP_UDP_PROTO)
                        0x02, 0x00, 0x11, 0x00,
                        0x55,0x44,0x50,0x00,0x00,0x00
ENDFIELD(IP_UDP_PROTO, 10)

FIELD(NAME_PPP)
                        0x50,0x50,0x50,0x00,0x00,0x00
ENDFIELD(NAME_PPP,6)
```

```
BEGIN_STRUCT_ARRAY(IP_UDP_PROTO_ARR,1)
IP_UDP_PROTO
ENDARRAY

LONG   OWN_IP_ADDRESS      0x0a0b0c0d    /* 10.11.12.13 */
LONG   PEER_IP_ADDRESS     0x0a0b0c0e    /* 10.11.12.14 */

LONG   TEST_SRC_IP_ADDRESS          0x0a0b0c0e
LONG   TEST_PEER_IP_ADDRESS         0x0a0b0c0d
LONG       TEST_SRC_IP_ADDRESS_1    0xc6a80105
LONG       TEST_SRC_IP_ADDRESS_2    0xac144e09

/*     Actually the MTU size is >= 1500, but as the maximum memory chunk length is 1500,
       we have to limit this for the test cases.
*/
SHORT  FRAGM_MTU_SIZE      52
SHORT  NORMAL_MTU_SIZE     400
SHORT          BIG_MTU_SIZE          1500
SHORT  ACI_TUI             3425          /* All TUIs are completely hypothetical */

BYTE   PPP_OP_ACK          0
BYTE   UDP_OP_ACK          0
BYTE   IP_OP_ACK           0

BYTE   ZERO_BYTE           0
SHORT  ZERO_SHORT          0

BYTE   IP_NOT_READY 0x0f

BYTE   UDP_P_ID       0x5
BYTE   IP_P_ID        0x21
BYTE   PPP_P_ID       0x3

/* Parameters for sending DTI2 primitives */

BEGIN_PSTRUCT ("st_lines", DTI_PARA_ST_LINES_SASB_ON)
       SET_COMP ("st_flow",          DTI_FLOW_ON)
       SET_COMP ("st_line_sa",             DTI_SA_ON)
       SET_COMP ("st_line_sb",             DTI_SB_ON)
       SET_COMP ("st_break_len",    DTI_BREAK_OFF)
ENDSTRUCT
BEGIN_PSTRUCT ("st_lines", DTI_PARA_ST_LINES_SASB_OFF)
       SET_COMP ("st_flow",          DTI_FLOW_ON)
       SET_COMP ("st_line_sa",             DTI_SA_OFF)
       SET_COMP ("st_line_sb",             DTI_SB_OFF)
```

```
        SET_COMP ("st_break_len",    DTI_BREAK_OFF)
ENDSTRUCT
BEGIN_PSTRUCT ("st_lines", DTI_PARA_ST_LINES_SA_OFF)
        SET_COMP ("st_flow",         DTI_FLOW_ON)
        SET_COMP ("st_line_sa",              DTI_SA_OFF)
        SET_COMP ("st_line_sb",              DTI_SB_ON)
        SET_COMP ("st_break_len",    DTI_BREAK_OFF)
ENDSTRUCT
BEGIN_PSTRUCT ("st_lines", DTI_PARA_ST_LINES_SB_OFF)
        SET_COMP ("st_flow",         DTI_FLOW_ON)
        SET_COMP ("st_line_sa",              DTI_SA_ON)
        SET_COMP ("st_line_sb",              DTI_SB_OFF)
        SET_COMP ("st_break_len",    DTI_BREAK_OFF)
ENDSTRUCT

BEGIN_PSTRUCT ("parameters", DTI_PARAMETER_UDP)
        SET_COMP ("p_id",    UDP_P_ID)
        SET_COMP ("st_lines",        DTI_PARA_ST_LINES_SASB_ON)
ENDSTRUCT


BEGIN_PSTRUCT ("parameters", DTI_PARAMETER_IP)
        SET_COMP ("p_id",    IP_P_ID)
        SET_COMP ("st_lines",        DTI_PARA_ST_LINES_SASB_ON)
ENDSTRUCT

BEGIN_PSTRUCT ("parameters", DTI_PARAMETER_PPP)
        SET_COMP ("p_id",    PPP_P_ID)
        SET_COMP ("st_lines",        DTI_PARA_ST_LINES_SASB_ON)
ENDSTRUCT
```

3 TEST CASES

3.1 Routing (internal)

3.1.1 IP001: Setup the Routing for the IP test

Description:

Routings for the IP tests are set.

Preamble:

None

```
     ACI                          IP                          PPP
      |                           |                           |
COMMAND (TAP RESET)
COMMAND (SMI RESET)
COMMAND (UDP RESET)
COMMAND (IP RESET)
COMMAND (PPP RESET)
      |                           |                           |
COMMAND (TAP REDIRECT CLEAR)
COMMAND (SMI REDIRECT CLEAR)
COMMAND (UDP REDIRECT CLEAR)
COMMAND (IP REDIRECT CLEAR)
COMMAND (PPP REDIRECT CLEAR)
      |                           |                           |
COMMAND (IP REDIRECT UDP TAP)
COMMAND (IP REDIRECT MMI TAP)
COMMAND (IP REDIRECT SMI TAP)
COMMAND (IP REDIRECT PPP TAP)
      |                           |                           |
COMMAND (TAP REDIRECT TAP IP)
      |                           |                           |
      |                           |                           |
```

Parametrization:

Primitive	Parameter	Value

History:

19 June 2000	NI	initial

3.2 Activation, Deactivation, and Configuration (IP100A - IP1xx)

3.2.1 IP100: Activation of IP

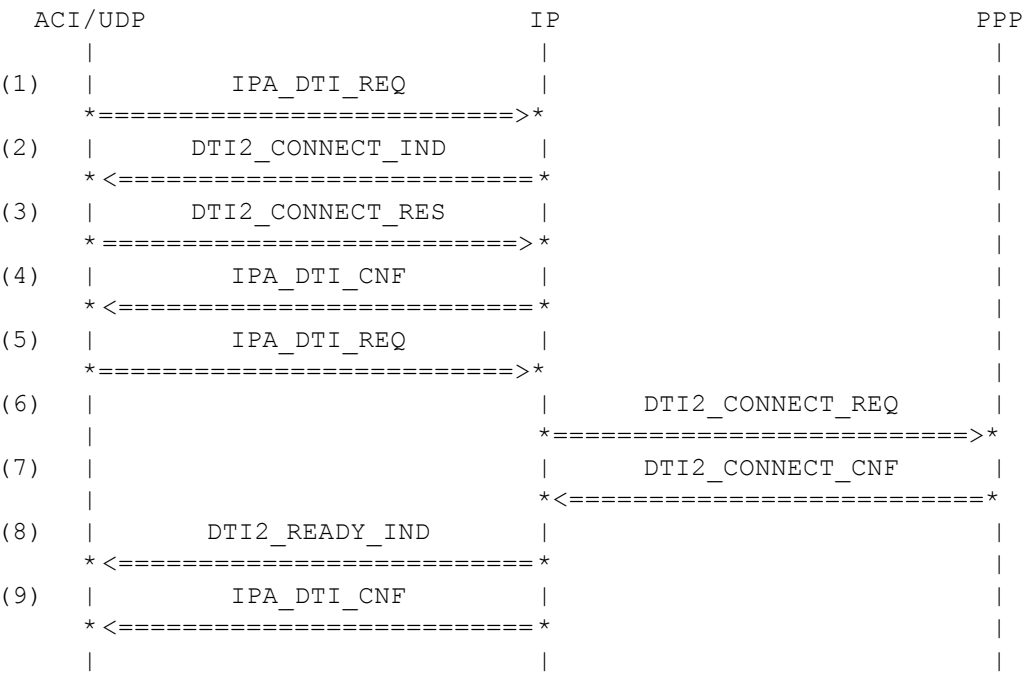
Description:

The IP entity receives the IPA_ACTIVATE_REQ and is switched to the activated state. At this point every related transport layer must be known.
IP informs UDP that it is ready to receive data. (Occurs with each transport layer)
IP informs ACI that the entity is activated.

Variants: <A>..<F>

Preamble:

<A>IP001
<B>IP400
<C>IP401
<D>IP402
<E>IP403
<F>IP112B



Parametrization:

Primitive	Parameter	Value
(1) IPA_DTI_REQ	dti_conn	IPA_CONNECT_DTI
	entity_name	HL_NAME
	link_id	UDP_LINK_ID
	dti_direction	IPA_DTI_TO_HIGHER_LAYER

(2) DTI2_CONNECT_IND	link_id	UDP_LINK_ID
	version	DTI_VERSION_10
(3) DTI2_CONNECT_RES	link_id	UDP_LINK_ID
	version	DTI_VERSION_10
(4) IPA_DTI_CNF	dti_conn	IPA_CONNECT_DTI
	link_id	UDP_LINK_ID
(5) IPA_DTI_REQ	dti_conn	IPA_CONNECT_DTI
	entity_name	LL_NAME
	link_id	PPP_LINK_ID
	dti_direction	IPA_DTI_TO_LOWER_LAYER
(6) DTI2_CONNECT_REQ	link_id	PPP_LINK_ID
	version	DTI_VERSION_10
(7) DTI2_CONNECT_CNF	link_id	PPP_LINK_ID
	version	DTI_VERSION_10
(8) DTI2_READY_IND	link_id	UDP_LINK_ID
(9) IPA_DTI_CNF	dti_conn	IPA_CONNECT_DTI
	link_id	PPP_LINK_ID

History:

19 June 2000	NI	initial
10 July 2000	xof	IPA_ACTIVATE_REQ fixed
05 December 2002	KJF	new IPA SAP

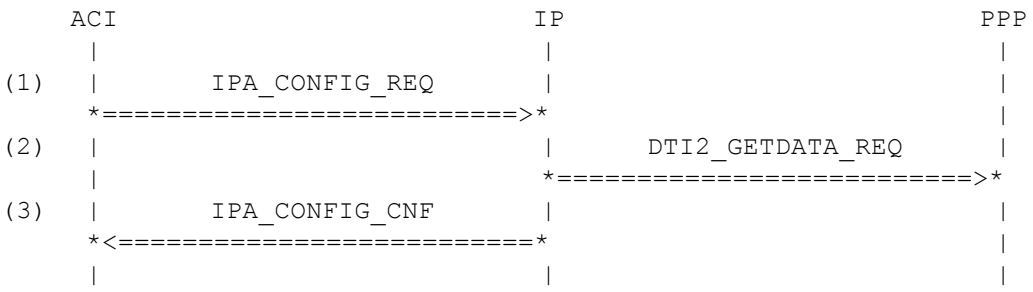
3.2.2 IP101: Configuration of IP, Interface goes up

Description:

Upon Reception of IPA_CONFIG_REQ the IP entity sends a DTI2_GETDATA_REQ to PPP and confirms to ACI with IPA_CONFIG_CNF.
It is now able to receive data from PPP or UDP and to relay it if appropriate.

Preamble:

IP100A



Parametrization:

Primitive	Parameter	Value
(1) IPA_CONFIG_REQ	ip	OWN_IP_ADDRESS
	peer_ip	PEER_IP_ADDRESS
	mtu	NORMAL_MTU_SIZE
	cmd	IPA_CONFIG_UP
(2) DTI2_GETDATA_REQ	link_id	PPP_LINK_ID
(3) IPA_CONFIG_CNF	ack_flag	IPA_CONFIG_ACK
	all_down	IPA_ALLDOWN_FALSE

History:

19 June 2000	NI	initial
09 July 2000	xof	bug fix fixed
05 December 2002	KJF	new IPA SAP

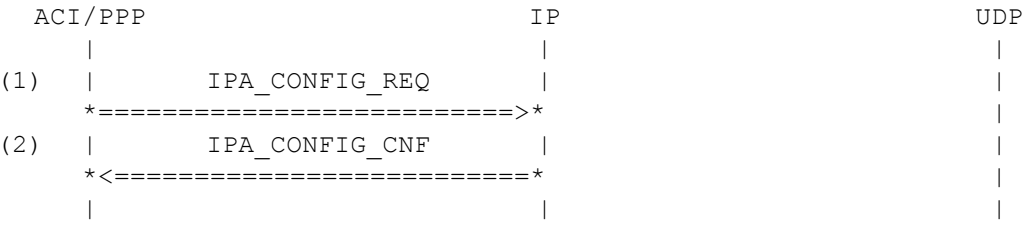
3.2.3 IP102: Disconnection by IPA_CONFIG_REQ

Description:

The UDP entity receives IPA_CONFIG_REQ(UDPA_CONFIG_DOWN).
IP confirms its deactivation and switches to deactivated state.

Preamble:

IP101



Parametrization:

Primitive	Parameter	Value
(1) IPA_CONFIG_REQ	ip	OWN_IP_ADDRESS
	peer_ip	PEER_IP_ADDRESS
	mtu	NORMAL_MTU_SIZE
	cmd	IPA_CONFIG_DOWN
(2) IPA_CONFIG_CNF	ack_flag	IPA_CONFIG_ACK
	all_down	IPA_ALLDOWN_TRUE

History:

19 June 2000	NI	initial
09 July 2000	xof	bug fix
05 December 2002	KJF	new IPA SAP

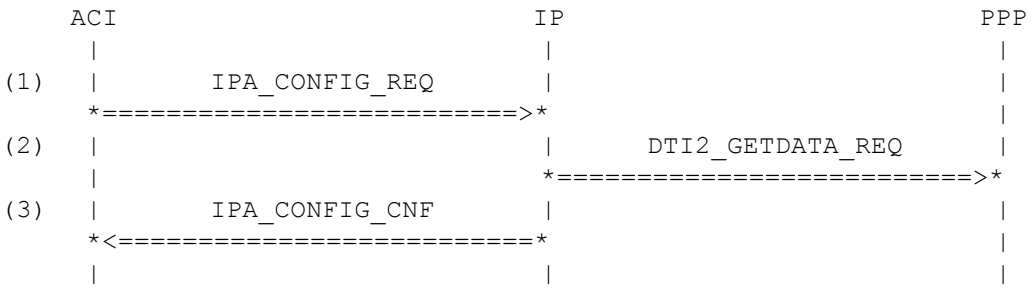
3.2.4 IP104: Configuration of IP, Interface goes up - less MTU for testing fragmenting

Description:

Upon Reception of IPA_CONFIG_REQ the IP entity sends a DTI2_GETDATA_REQ to PPP and confirms to ACI with IPA_CONFIG_CNF.
It is now able to receive data from PPP or UDP and to relay it if appropriate.

Preamble:

IP100A



Parametrization:

Primitive	Parameter	Value
(1) IPA_CONFIG_REQ	ip	PEER_IP_ADDRESS
	peer_ip	OWN_IP_ADDRESS
	mtu	FRAGM_MTU_SIZE
	cmd	IPA_CONFIG_UP
(2) DTI2_GETDATA_REQ	link_id	PPP_LINK_ID
(3) IPA_CONFIG_CNF	ack_flag	IPA_CONFIG_ACK
	all_down	IPA_ALLDOWN_FALSE

History:

30 August 2000	OFL	initial

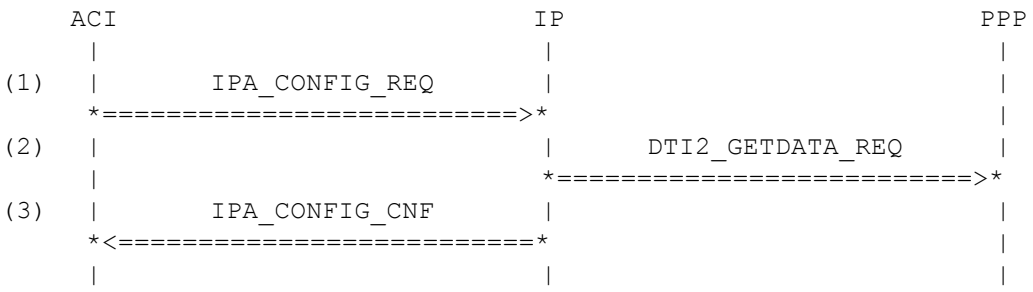
3.2.5 IP105: Configuration of IP, Interface goes up with other another source address

Description:

Upon Reception of IPA_CONFIG_REQ the IP entity sends a DTI2_GETDATA_REQ to PPP and confirms to ACI with IPA_CONFIG_CNF.
It is now able to receive data from PPP or UDP and to relay it if appropriate.

Preamble:

IP100A



Parametrization:

Primitive	Parameter	Value
(1) IPA_CONFIG_REQ	ip	OWN_IP_ADDRESS
	peer_ip	PEER_IP_ADDRESS
	mtu	NORMAL_MTU_SIZE
	cmd	IPA_CONFIG_UP
(2) DTI2_GETDATA_REQ	link_id	PPP_LINK_ID
(3) IPA_CONFIG_CNF	ack_flag	IPA_CONFIG_ACK
	all_down	IPA_ALLDOWN_FALSE

History:

30 August 2000	OFL	initial

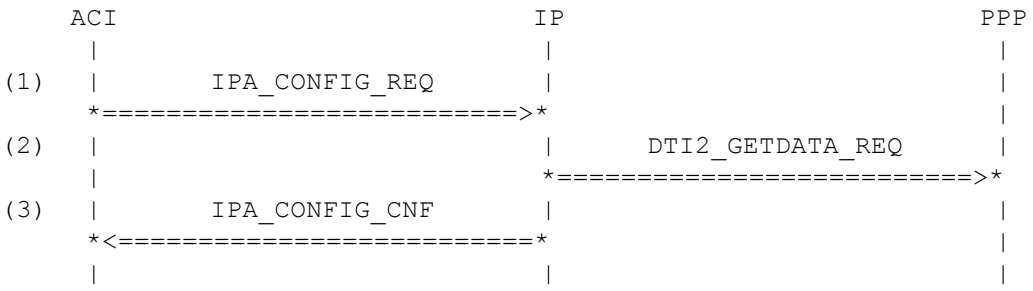
3.2.6 IP106: Configuration of IP, Interface goes up with other destination address

Description:

Upon Reception of IPA_CONFIG_REQ the IP entity sends a DTI2_GETDATA_REQ to PPP and confirms to ACI with IPA_CONFIG_CNF. It is now able to receive data from PPP or UDP and to relay it if appropriate.

Preamble:

IP100A



Parametrization:

Primitive	Parameter	Value
(1) IPA_CONFIG_REQ	ip	TEST_SRC_IP_ADDRESS
	peer_ip	TEST_PEER_IP_ADDRESS
	mtu	BIG_MTU_SIZE
	cmd	IPA_CONFIG_UP
(2) DTI2_GETDATA_REQ	link_id	PPP_LINK_ID
(3) IPA_CONFIG_CNF	ack_flag	IPA_CONFIG_ACK
	all_down	IPA_ALLDOWN_FALSE

History:

26 September 2000	OFL	initial

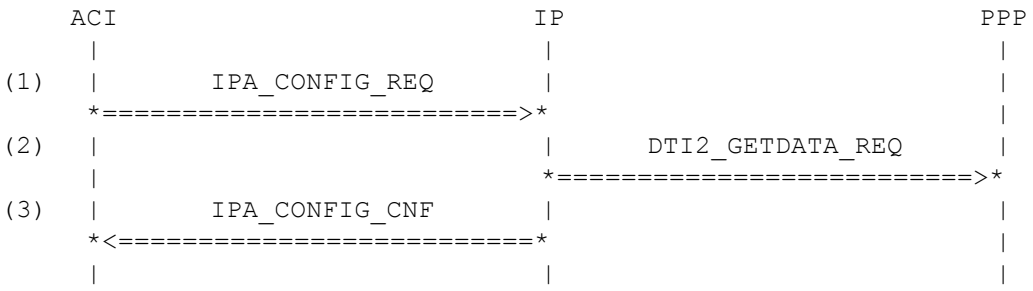
3.2.7 IP107: Configuration of IP, Interface goes up with a big MTU-Size

Description:

Upon Reception of IPA_CONFIG_REQ the IP entity sends a DTI2_GETDATA_REQ to PPP and confirms to ACI with IPA_CONFIG_CNF.
It is now able to receive data from PPP or UDP and to relay it if appropriate.

Preamble:

IP100A



Parametrization:

Primitive	Parameter	Value
(1) IPA_CONFIG_REQ	ip	TEST_SRC_IP_ADDRESS_1
	peer_ip	TEST_PEER_IP_ADDRESS
	mtu	BIG_MTU_SIZE
	cmd	IPA_CONFIG_UP
(2) DTI2_GETDATA_REQ	link_id	PPP_LINK_ID
(3) IPA_CONFIG_CNF	ack_flag	IPA_CONFIG_ACK
	all_down	IPA_ALLDOWN_FALSE

History:

30 August 2000	OFL	initial

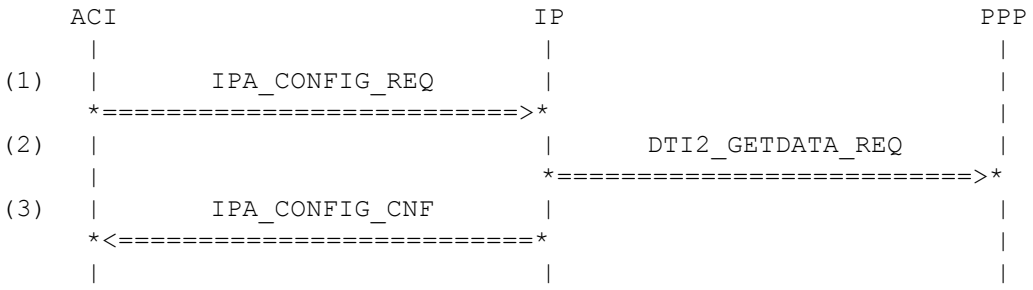
3.2.8 IP108: Configuration of IP, Interface goes up with other destination address

Description:

Upon Reception of IPA_CONFIG_REQ the IP entity sends a DTI2_GETDATA_REQ to PPP and confirms to ACI with IPA_CONFIG_CNF. It is now able to receive data from PPP or UDP and to relay it if appropriate.

Preamble:

IP100A



Parametrization:

Primitive	Parameter	Value
(1) IPA_CONFIG_REQ	ip	TEST_SRC_IP_ADDRESS_2
	peer_ip	TEST_PEER_IP_ADDRESS
	mtu	BIG_MTU_SIZE
	cmd	IPA_CONFIG_UP
(2) DTI2_GETDATA_REQ	link_id	PPP_LINK_ID
(3) IPA_CONFIG_CNF	ack_flag	IPA_CONFIG_ACK
	all_down	IPA_ALLDOWN_FALSE

History:

26 September 2000	OFL	initial

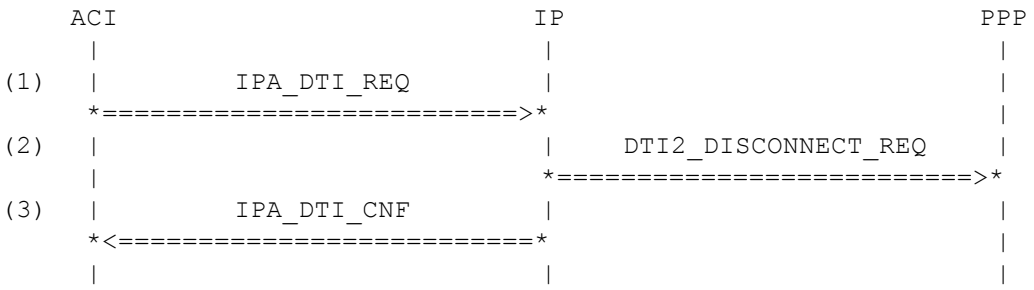
3.2.9 IP109: Deactivation by IP_DTI_REQ (lower layer)

Description:

The IP entity receives the IPA_DTI_REQ(IP_DISCONNECT_DTI) primitive for the lower layer.

Preamble:

IP102



Parametrization:

Primitive	Parameter	Value
(1) IPA_DTI_REQ	dti_conn	IPA_DISCONNECT_DTI
	entity_name	LL_NAME
	link_id	PPP_LINK_ID
	dti_direction	IPA_DTI_TO_LOWER_LAYER
(2) DTI2_DISCONNECT_REQ	link_id	PPP_LINK_ID
	cause	DTI_CAUSE_NORMAL_CLOSE
(3) IPA_DTI_CNF	dti_conn	IPA_DISCONNECT_DTI
	link_id	PPP_LINK_ID

History:

28 June 2000	NI	initial
25 Oct 2002	KJF	new IPA SAP

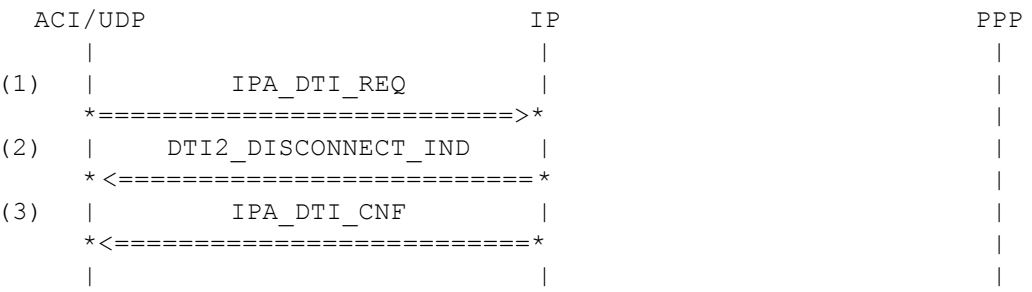
3.2.10 IP110: Deactivation by IPA_DTI_REQ (higher layer)

Description:

The IP entity receives the IPA_DTI_REQ(IPA_DISCONNECT_DTI) primitive for the higher layer.

Preamble:

IP109



Parametrization:

Primitive	Parameter	Value
(4) IPA_DTI_REQ	dti_conn	IPA_DISCONNECT_DTI
	entity_name	HL_NAME
	link_id	UDP_LINK_ID
	dti_direction	IPA_DTI_TO_HIGHER_LAYER
(5) DTI2_DISCONNECT_IND	link_id	UDP_LINK_ID
	cause	DTI_CAUSE_NORMAL_CLOSE
(6) IPA_DTI_CNF	dti_conn	IPA_DISCONNECT_DTI
	link_id	UDP_LINK_ID

History:

28 June 2000	NI	initial
25 Oct 2002	KJF	new IPA SAP

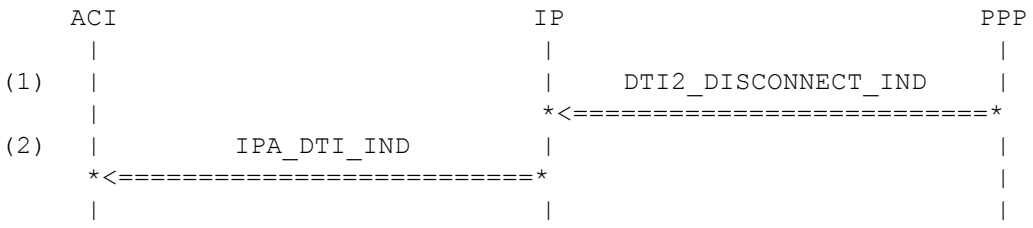
3.2.11 IP111: Deactivation after reception of DTI2_DISCONNECT_IND

Description:

The IP entity receives a DTI2_DISCONNECT_IND
IP switches to deactivated state.

Preamble:

<A>IP101
<B>IP326



Parametrization:

Primitive	Parameter	Value
(1) DTI2_DISCONNECT_IND	link_id	PPP_LINK_ID
	cause	DTI_CAUSE_NORMAL_CLOSE
(2) IPA_DTI_IND	link_id	PPP_LINK_ID

History:

13 Nov 2001	TVO	initial
30-Oct-2002	KJF	new IPA SAP

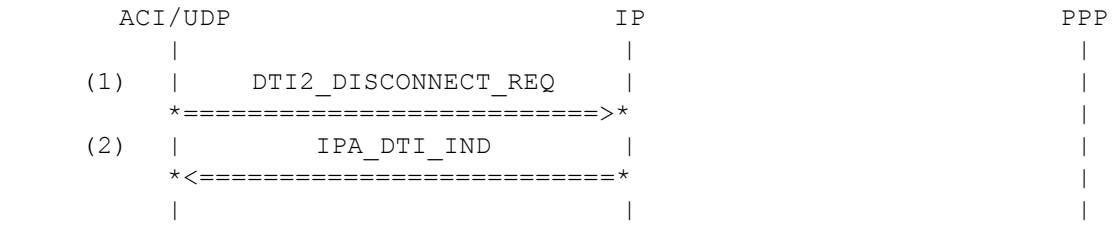
3.2.12 IP112: Deactivation after reception of DTI2_DISCONNECT_REQ

Description:

The UDP entity receives a DTI2_DISCONNECT_REQ.
UDP confirms its deactivation and switches to deactivated state.

Preamble:

<A>IP101
<B>IP111B



Parametrization:

Primitive	Parameter	Value
(1) DTI2_DISCONNECT_REQ	link_id	UDP_LINK_ID
	cause	DTI_CAUSE_NORMAL_CLOSE
(2) IPA_DTI_IND	link_id	UDP_LINK_ID

History:

13 Nov 2001	TVO	initial
25 Oct 2002	KJF	new IPA SAP

3.3 IP Address Request (IP200)

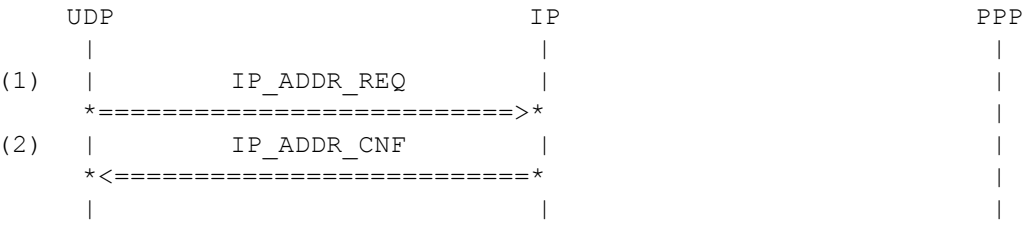
3.3.1 IP200: Reception of IP Address Request in Configured State

Description:

The IP entity receives an IP_ADDR_REQ
and determines the outgoing interface for the specified destination IP address.
IP returns an IP_ADDR_CNF with the IP address of the outgoing interface.

Preamble:

IP101



Parametrization:

Primitive	Parameter	Value
(1) IP_ADDR_REQ	dst_addr	PEER_IP_ADDRESS
	trans_prot	UDP_PROTOCOL
(1) IP_ADDR_CNF	src_addr	OWN_IP_ADDRESS
	err	IP_ADDR_NOERROR
	trans_prot	UDP_PROTOCOL

History:

19 June 2000	NI	initial

3.4 Packet Relaying (IP300 - IP3xx)

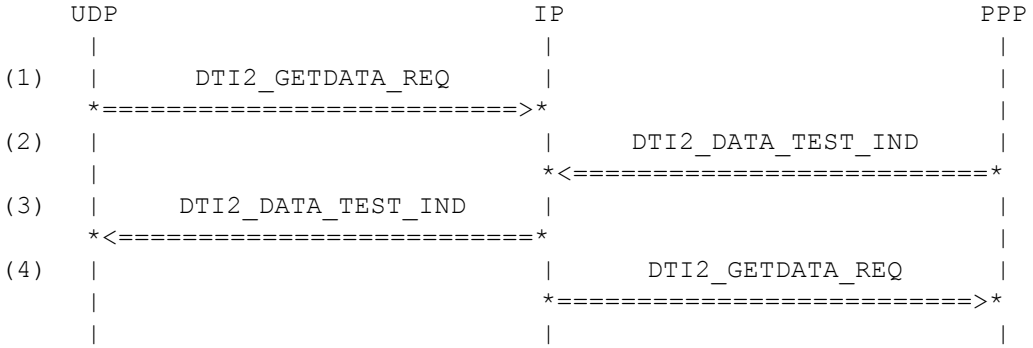
3.4.1 IP300: Packet PPP -> IP -> UDP

Description:

(1) The IP entity receives a valid UDP packet from PPP.
(2) After appropriate processing, IP relays the packet to UDP.
(3) IP answers to PPP with DTI2_GETDATA_REQ.

Preamble:

IP101



Parametrization:

Primitive	Parameter	Value
(1) DTI2_GETDATA_REQ	link_id	UDP_LINK_ID
(2) DTI2_DATA_TEST_IND	link_id	PPP_LINK_ID
	parameters	DTI_PARAMETER_PPP
	sdu	PKT_UDP_NODATA_IN_300
(3) DTI2_DATA_TEST_IND	link_id	UDP_LINK_ID
	parameters	DTI_PARAMETER_IP
	sdu	PKT_UDP_NODATA_IN_300
(4) DTI2_GETDATA_REQ	link_id	PPP_LINK_ID

History:

26 June 2000	NI	bug fix
09 July 2000	XOF	Filled in DTI_TEST primitives
13 July 2000	NI	moved DTI2_GETDATA_REQ to front
25 Oct 2002	KJF	new IPA SAP

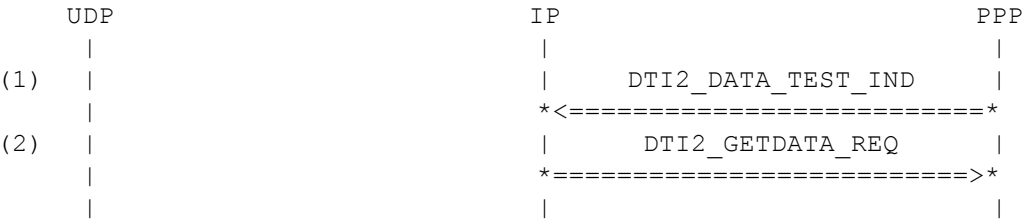
3.4.2 IP301: Packet PPP -> IP -> UDP with Checksum error

Description:

The IP entity receives a UDP packet with checksum error from PPP.
IP answers to PPP with DTI2_GETDATA_REQ.

Preamble:

IP101



Parametrization:

Primitive	Parameter	Value
(1) DTI2_DATA_TEST_IND	link_id	PPP_LINK_ID
	parameters	DTI_PARAMETER_IP
	sdu	PKT_UDP_NODATA_IN_301
(2) DTI2_GETDATA_REQ	link_id	PPP_LINK_ID

History:

19 June 2000	NI	initial
09 July 2000	xof	fix bugs

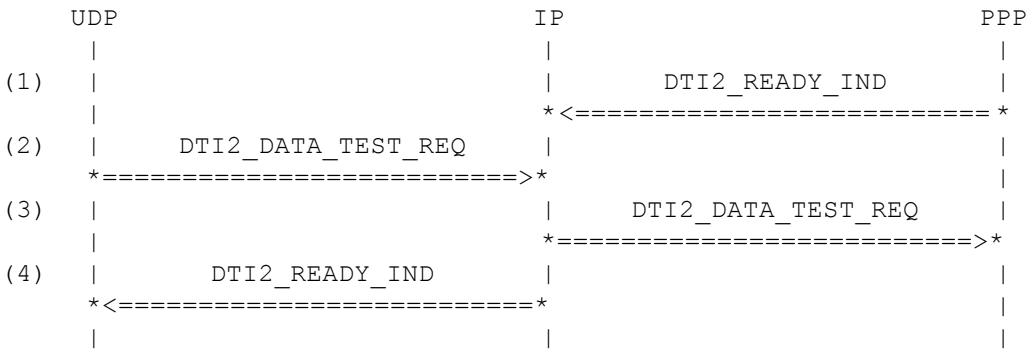
3.4.3 IP302: Packet UDP -> IP -> PPP

Description:

The IP entity receives a valid packet from UDP.
After appropriate processing, IP relays the packet to PPP.
IP answers to UDP with DTI2_READY_IND.

Preamble:

IP105



Parametrization:

Primitive	Parameter	Value
(1) DTI2_READY_IND	link_id	PPP_LINK_ID
(2) DTI2_DATA_TEST_REQ	link_id	UDP_LINK_ID
	parameters	DTI_PARAMETER_UDP
	sdu	PKT_UDP_NODATA_OUT_1_302
(3) DTI2_DATA_TEST_REQ	link_id	PPP_LINK_ID
	parameters	DTI_PARAMETER_IP
	sdu	PKT_UDP_NODATA_OUT_2_302
(4) DTI2_READY_IND	link_id	UDP_LINK_ID

History:

19 June 2000	NI	initial
09 July 2000	xof	bug fix
13 July 2000	NI	moved DTI2_READY_IND to front

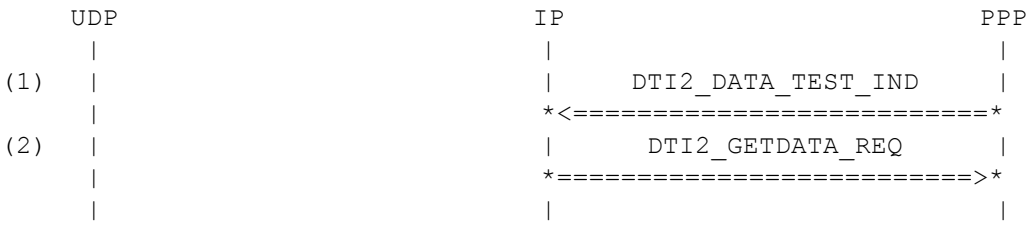
3.4.4 IP303: Packet PPP -> IP -> UDP, IP length > packet length

Description:

The IP entity receives a UDP packet from PPP with IP length field > packet length, which is thrown away.
IP answers to PPP with DTI2_GETDATA_REQ.

Preamble:

IP101



Parametrization:

Primitive	Parameter	Value
(1) DTI2_DATA_TEST_IND	link_id	PPP_LINK_ID
	parameters	DTI_PARAMETER_PPP
	sdu	PKT_UDP_NODATA_IN_303
(2) DTI2_GETDATA_REQ	link_id	PPP_LINK_ID

History:

26 June 2000	NI	initial

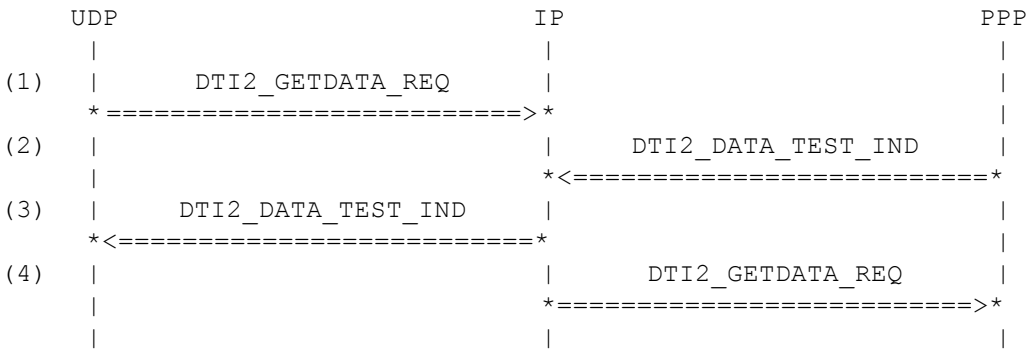
3.4.5 IP304: Packet PPP -> IP -> UDP, IP length < packet length

Description:

The IP entity receives a UDP packet from PPP with IP length less than field packet size. IP truncates the packet and relays it to UDP.
IP answers to PPP with DTI2_GETDATA_REQ.

Preamble:

IP101



Parametrization:

Primitive	Parameter	Value
(1) DTI2_GETDATA_REQ	link_id	UDP_LINK_ID
(2) DTI2_DATA_TEST_IND	link_id	PPP_LINK_ID
	parameters	DTI_PARAMETER_PPP
	sdu	PKT_UDP_IN_304
(3) DTI2_DATA_TEST_IND	link_id	UDP_LINK_ID
	parameters	DTI_PARAMETER_IP
	sdu	PKT_UDP_IN_2_304
(4) DTI2_GETDATA_REQ	link_id	PPP_LINK_ID

History:

26 June 2000	NI	initial
09 July 2000	xof	bug fix
13 July 2000	NI	moved DTI2_GETDATA_REQ to front

3.4.6 IP305: Packet PPP -> IP -> UDP, IP length (== packet length) > MTU

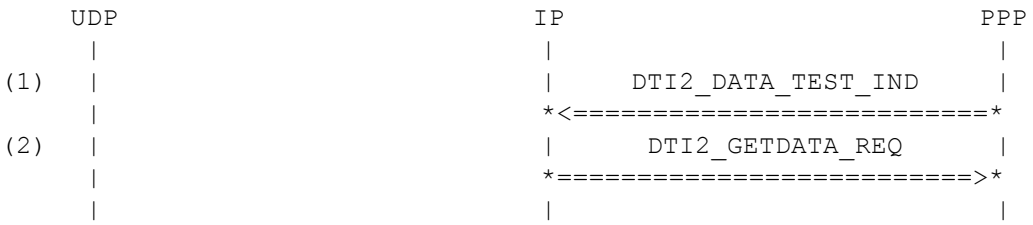
Description:

The IP entity receives a UDP packet from PPP longer than the MTU size, which is thrown away.

IP answers to PPP with DTI2_GETDATA_REQ.

Preamble:

IP101



Parametrization:

Primitive	Parameter	Value
(1) DTI2_DATA_TEST_IND	link_id	PPP_LINK_ID
	parameters	DTI_PARAMETER_PPP
	sdu	PKT_UDP_IN_305
(2) DTI2_GETDATA_REQ	link_id	PPP_LINK_ID

History:

26 June 2000	NI	initial

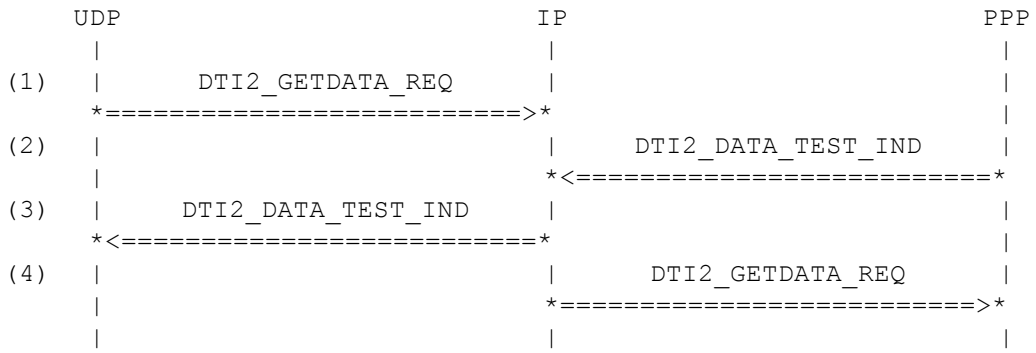
3.4.7 IP306: Packet PPP -> IP -> UDP, IP length == MTU, packet length > MTU

Description:

The IP entity receives a UDP packet from PPP which is greater than the MTU size. But the IP length field is not.
IP truncates the packet and relays it to UDP.

Preamble:

IP101



Parametrization:

Primitive	Parameter	Value
(1) DTI2_GETDATA_REQ	link_id	UDP_LINK_ID
(2) DTI2_DATA_TEST_IND	link_id	PPP_LINK_ID
	parameters	DTI_PARAMETER_PPP
	sdu	PKT_UDP_IN_306
(3) DTI2_DATA_TEST_IND	link_id	UDP_LINK_ID
	parameters	DTI_PARAMETER_IP
	sdu	PKT_UDP_IN_2_306
(4) DTI2_GETDATA_REQ	link_id	PPP_LINK_ID

History:

26 June 2000	NI	initial
11 July 2000	xof	Put in DTI2_GETDATA_REQ

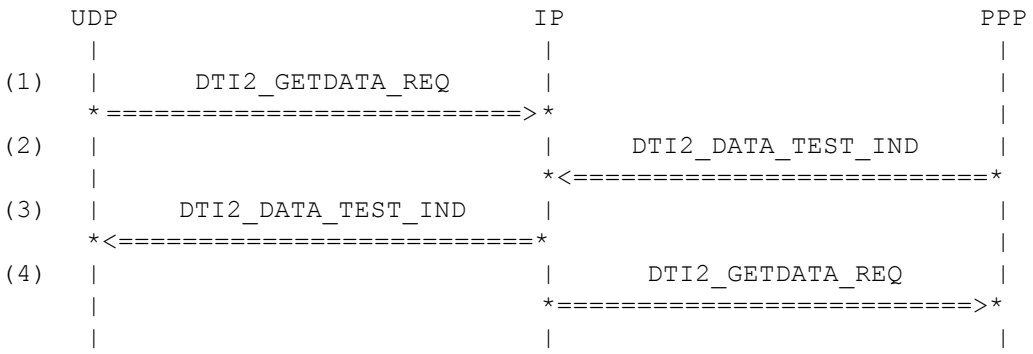
3.4.8 IP307: Packet PPP -> IP -> UDP, IP length == packet length == MTU

Description:

(1) The IP entity receives a valid UDP packet from PPP with a length of MTU size.
(2) After appropriate processing, IP relays the packet to UDP.
(3) IP answers to PPP with DTI2_GETDATA_REQ.

Preamble:

IP101



Parametrization:

Primitive	Parameter	Value
(1) DTI2_GETDATA_REQ	link_id	UDP_LINK_ID
(2) DTI2_DATA_TEST_IND	link_id	PPP_LINK_ID
	parameters	DTI_PARAMETER_PPP
	sdu	PKT_UDP_IN_2_306
(3) DTI2_DATA_TEST_IND	link_id	UDP_LINK_ID
	parameters	DTI_PARAMETER_IP
	sdu	PKT_UDP_IN_2_306
(4) DTI2_GETDATA_REQ	link_id	PPP_LINK_ID

History:

26 June 2000	NI	initial
11 July 2000	xof	Put in DTI2_GETDATA_REQ

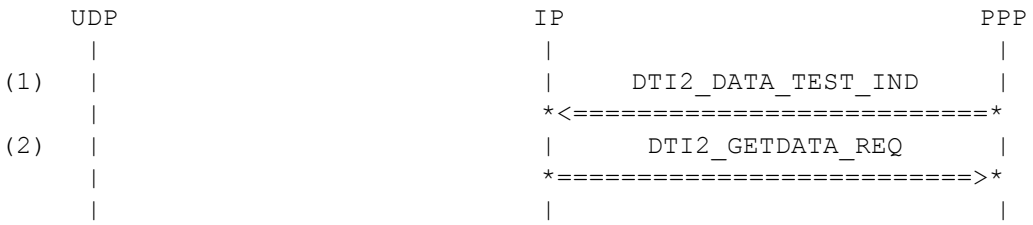
3.4.9 IP308: Packet PPP -> IP -> UDP, length < IP header length

Description:

The IP entity receives a UDP packet from PPP that is shorter than an IP header, which is thrown away.
IP answers to PPP with DTI2_GETDATA_REQ.

Preamble:

IP101



Parametrization:

Primitive	Parameter	Value
(1) DTI2_DATA_TEST_IND	link_id	PPP_LINK_ID
	parameters	DTI_PARAMETER_PPP
	sdu	PKT_UDP_IN_308
(2) DTI2_GETDATA_REQ	link_id	PPP_LINK_ID

History:

27 June 2000	NI	initial

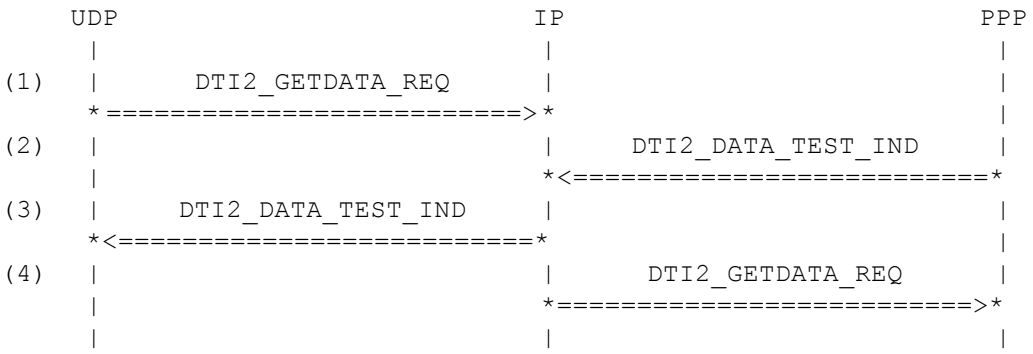
3.4.10 IP310: Packet PPP -> IP -> UDP, 8 octets of (fake) options

Description:

The IP entity receives a UDP packet from PPP with IP options.
The IP options are invalid, but they are ignored anyway.
After appropriate processing, IP relays the packet to UDP.
IP answers to PPP with DTI2_GETDATA_REQ.

Preamble:

IP101



Parametrization:

Primitive	Parameter	Value
(1) DTI2_GETDATA_REQ	link_id	UDP_LINK_ID
(2) DTI2_DATA_TEST_IND	link_id	PPP_LINK_ID
	parameters	DTI_PARAMETER_PPP
	sdu	PKT_UDP_IN_310
(3) DTI2_DATA_TEST_IND	link_id	UDP_LINK_ID
	parameters	DTI_PARAMETER_IP
	sdu	PKT_UDP_IN_310
(4) DTI2_GETDATA_REQ	link_id	PPP_LINK_ID

History:

27 June 2000	NI	initial
11 July 2000	xof	Put in DTI2_GETDATA_REQ

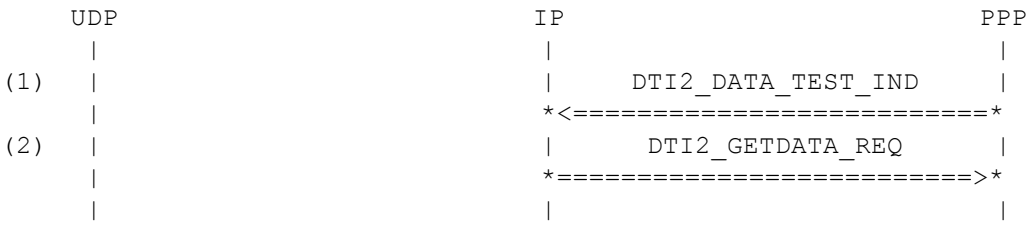
3.4.11 IP311: Fragmenting PPP -> IP -> UDP, middle fragment

Description:

The IP entity receives a middle fragment packet from PPP, which is thrown away, as the implementation at the moment does not support IP fragmentation.
IP answers to PPP with DTI2_GETDATA_REQ.

Preamble:

IP101



Parametrization:

Primitive	Parameter	Value
(1) DTI2_DATA_TEST_IND	link_id	PPP_LINK_ID
	parameters	DTI_PARAMETER_PPP
	sdu	PKT_UDP_IN_311
(2) DTI2_GETDATA_REQ	link_id	PPP_LINK_ID

History:

27 June 2000	NI	initial
09 July 2000	XOF	Filled in DTI_TEST primitive
25 Oct 2002	KJF	new IPA SAP

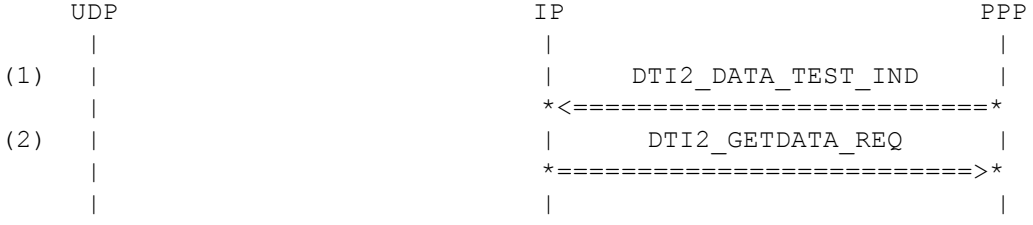
3.4.12 IP312: Fragmenting PPP -> IP -> UDP, last fragment

Description:

(1) The IP entity receives a last fragment packet from PPP.
(2) IP answers to PPP with DTI2_GETDATA_REQ.

Preamble:

IP101



Parametrization:

Primitive	Parameter	Value
(1) DTI2_DATA_TEST_IND	link_id	PPP_LINK_ID
	parameters	DTI_PARAMETER_PPP
	sdu	PKT_UDP_IN_312
(2) DTI2_GETDATA_REQ	link_id	PPP_LINK_ID

History:

27 June 2000	NI	initial
09 July 2000	XOF	Filled in DTI_TEST primitives

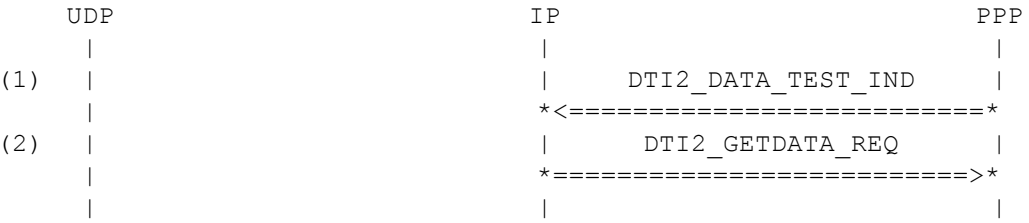
3.4.13 IP313: Fragmenting PPP -> IP -> UDP, first fragment

Description:

(1) The IP entity receives a first fragment packet from PPP.
(2) IP answers to PPP with DTI2_GETDATA_REQ.

Preamble:

IP101



Parametrization:

Primitive	Parameter	Value
(1) DTI2_DATA_TEST_IND	link_id	PPP_LINK_ID
	parameters	DTI_PARAMETER_PPP
	sdu	PKT_IP_IN_FIRST_FRAGM
(2) DTI2_GETDATA_REQ	link_id	PPP_LINK_ID

History:

22 August 2000	xof	initial

3.4.14 IP314: Fragmenting PPP -> IP -> UDP, to fragments with not the same id - drop the second packet

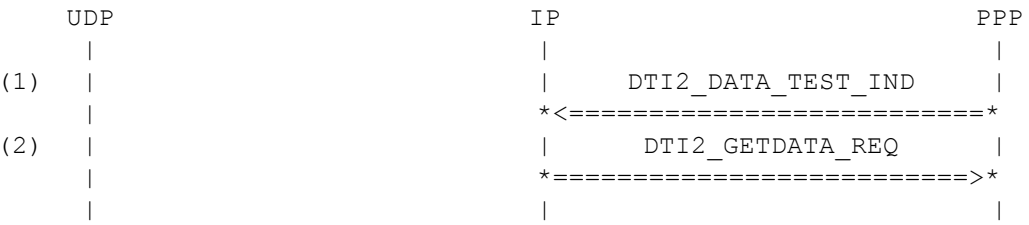
Description:

(3) The IP entity receives two fragments with not the same id. Ip drop the second packet.

(4) Send ready for new packet.

Preamble:

IP313



Parametrization:

Primitive	Parameter	Value
(1) DTI2_DATA_TEST_IND	link_id	PPP_LINK_ID
	parameters	DTI_PARAMETER_PPP
	sdu	PKT_UDP_IN_314
(2) DTI2_GETDATA_REQ	link_id	PPP_LINK_ID

History:

22 August 2000	xof	initial

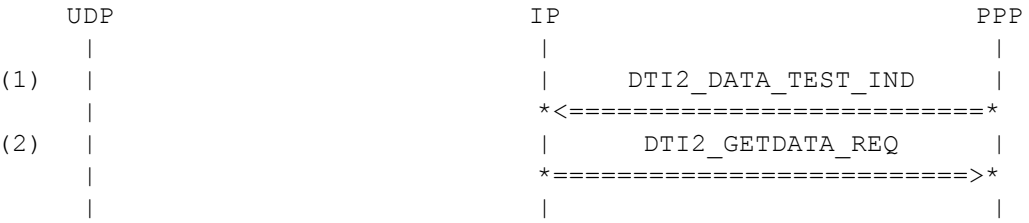
3.4.15 IP315: Fragmenting PPP -> IP -> UDP, middle fragment with correct ID

Description:

The IP entity receives the second fragments.
Send ready indication.

Preamble:

IP313



Parametrization:

Primitive	Parameter	Value
(1) DTI2_DATA_TEST_IND	link_id	PPP_LINK_ID
	parameters	DTI_PARAMETER_PPP
	sdu	PKT_IP_IN_SECOND_FRAGM
(2) DTI2_GETDATA_REQ	link_id	PPP_LINK_ID

History:

22 August 2000	xof	initial

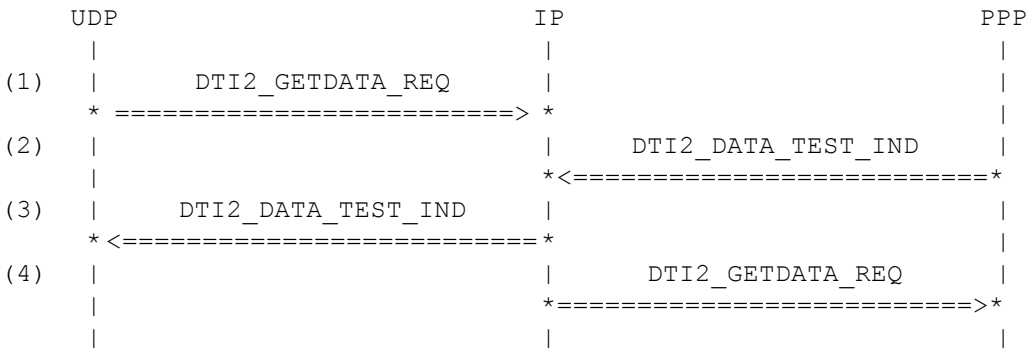
3.4.16 IP316: Fragmenting PPP -> IP -> UDP, last fragment with correct id - success with fragmenting

Description:

The IP entity get DTI2_GETDATA_REQ
IP get the second fragment
IP send the correct packet to higher entity
IP send a DTI2_GETDATA_REQ – and indicate success

Preamble:

IP315



Parametrization:

Primitive	Parameter	Value
(1) DTI2_GETDATA_REQ	link_id	UDP_LINK_ID
(2) DTI2_DATA_TEST_IND	link_id	PPP_LINK_ID
	parameters	DTI_PARAMETER_PPP
	sdu	PKT_IP_IN_LAST_FRAGM
(3) DTI2_DATA_TEST_IND	link_id	UDP_LINK_ID
	parameters	DTI_PARAMETER_IP
	sdu	PKT_IP_OUT_FRAGM
(4) DTI2_GETDATA_REQ	link_id	PPP_LINK_ID

History:

22 August 2000	xof	initial

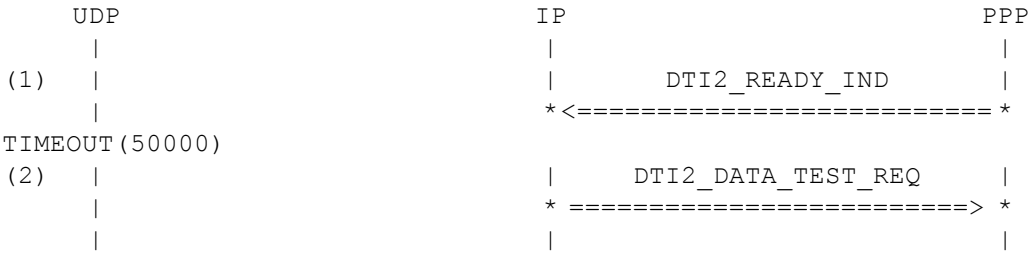
3.4.17 IP317: Fragmenting PPP -> IP -> UDP, not receiving the last fragment - reassembly timeout

Description:

IP do not get the last fragment – send a ICMP message with error indication and a DTI2_GETDATA_REQ. The correct timer value is 30 – 60 seconds.

Preamble:

IP315



Parametrization:

Primitive	Parameter	Value
(1) DTI2_READY_IND	link_id	PPP_LINK_ID
(2) DTI2_DATA_TEST_REQ	link_id	PPP_LINK_ID
	parameters	DTI_PARAMETER_IP
	sdu	PKT_ICMP_REASS_TIMER_317
(3) DTI2_GETDATA_REQ	link_id	PPP_LINK_ID

History:

22 August 2000	xof	initial

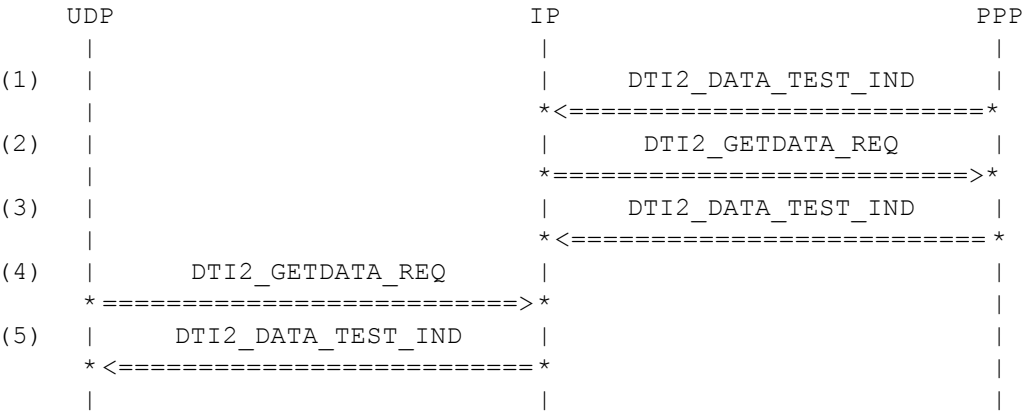
3.4.18 IP318: Fragmenting PPP -> IP -> UDP, first, last, then middle fragment with correct ID

Description: IP receives last fragment after first fragment before middle fragment. IP sort the fragments, build

the second fragment and send it to higher entity.

Preamble:

IP313



Parametrization:

Primitive	Parameter	Value
(1) DTI2_DATA_TEST_IND	link_id	PPP_LINK_ID
	parameters	DTI_PARAMETER_PPP
	sdu	PKT_IP_IN_LAST_FRAGM
(2) DTI2_GETDATA_REQ	link_id	PPP_LINK_ID
(3) DTI2_DATA_TEST_IND	link_id	PPP_LINK_ID
	parameters	DTI_PARAMETER_PPP
	sdu	PKT_IP_IN_SECOND_FRAGM
(4) DTI2_GETDATA_REQ	link_id	UDP_LINK_ID
(5) DTI2_DATA_TEST_IND	link_id	UDP_LINK_ID
	parameters	DTI_PARAMETER_IP
	sdu	PKT_IP_OUT_FRAGM

History:

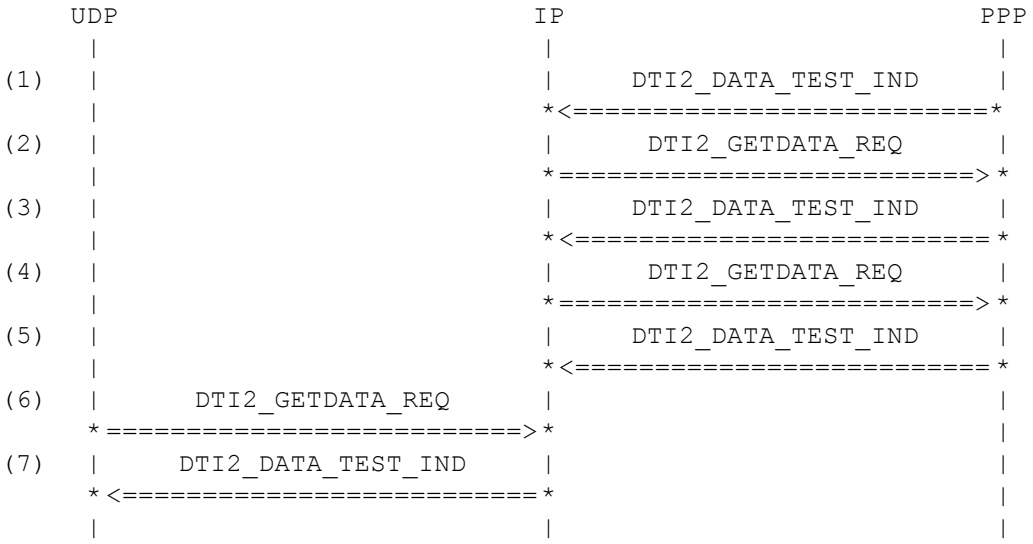
22 August 2000	xof	initial

3.4.19 IP319: Fragmenting PPP -> IP –> UDP, first, twice, last, then middle fragment with correct ID

Description: IP throw the second last fragment away and send the correct packet to higher entity.

Preamble:

IP313



Parametrization:

Primitive	Parameter	Value
(1) DTI2_DATA_TEST_IND	link_id	PPP_LINK_ID
	parameters	DTI_PARAMETER_PPP
	sdu	PKT_IP_IN_LAST_FRAGM
(2) DTI2_GETDATA_REQ	link_id	PPP_LINK_ID
(3) DTI2_DATA_TEST_IND	link_id	PPP_LINK_ID
	parameters	DTI_PARAMETER_PPP
	sdu	PKT_IP_IN_LAST_FRAGM
(4) DTI2_GETDATA_REQ	link_id	PPP_LINK_ID
(5) DTI2_DATA_TEST_IND	link_id	PPP_LINK_ID
	parameters	DTI_PARAMETER_PPP
	sdu	PKT_IP_IN_SECOND_FRAGM
(6) DTI2_GETDATA_REQ	link_id	UDP_LINK_ID
(7) DTI2_DATA_TEST_IND	link_id	UDP_LINK_ID
	parameters	DTI_PARAMETER_IP
	sdu	PKT_IP_OUT_FRAGM

History:

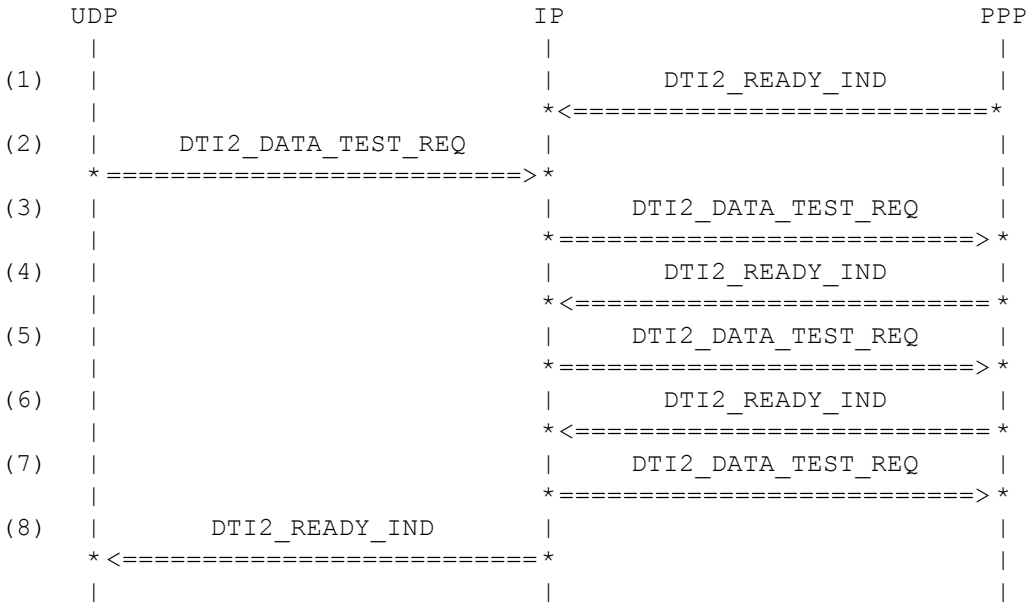
22 August 2000	xof	initial

3.4.20 IP320: Fragmenting UDP -> IP -> PPP packet

Description: IP becomes a packet from higher layer which is bigger than MTU. IP build fragments.

Preamble:

IP104



Parametrization:

Primitive	Parameter	Value
(1) DTI2_READY_IND	link_id	PPP_LINK_ID
(2) DTI2_DATA_TEST_REQ	link_id	UDP_LINK_ID
	parameters	DTI_PARAMETER_UDP
	sdu	PKT_IP_OUT_FRAGM
(3) DTI2_DATA_TEST_REQ	link_id	PPP_LINK_ID
	parameters	DTI_PARAMETER_IP
	sdu	PKT_IP_IN_FIRST_FRAGM
(4) DTI2_READY_IND	link_id	PPP_LINK_ID
(5) DTI2_DATA_TEST_REQ	link_id	PPP_LINK_ID
	parameters	DTI_PARAMETER_IP
	sdu	PKT_IP_IN_SECOND_FRAGM
(6) DTI2_READY_IND	link_id	PPP_LINK_ID
(7) DTI2_DATA_TEST_REQ	link_id	PPP_LINK_ID
	parameters	DTI_PARAMETER_IP
	sdu	PKT_IP_IN_LAST_FRAGM
(8) DTI2_READY_IND	link_id	UDP_LINK_ID

History:

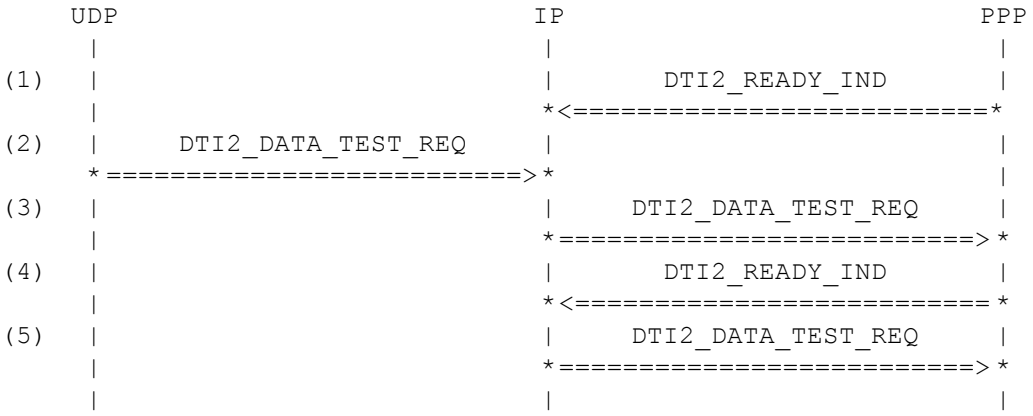
22 August 2000	xof	initial

3.4.21 IP321: Fragmenting UDP -> IP -> PPP, only to fragments

Description: IP becomes a packet from higher layer which is bigger than MTU. IP build fragments.

Preamble:

IP320



Parametrization:

Primitive	Parameter	Value
(1) DTI2_READY_IND	link_id	PPP_LINK_ID
(2) DTI2_DATA_TEST_REQ	link_id	UDP_LINK_ID
	parameters	DTI_PARAMETER_UDP
	sdu	PKT_IP_OUT_FRAGM
(3) DTI2_DATA_TEST_REQ	link_id	PPP_LINK_ID
	parameters	DTI_PARAMETER_IP
	sdu	PKT_IP_IN_FIRST_FRAGM
(4) DTI2_READY_IND	link_id	PPP_LINK_ID
(5) DTI2_DATA_TEST_REQ	link_id	PPP_LINK_ID
	parameters	DTI_PARAMETER_IP
	sdu	PKT_IP_IN_SECOND_FRAGM

History:

29 September 2000	xof	initial

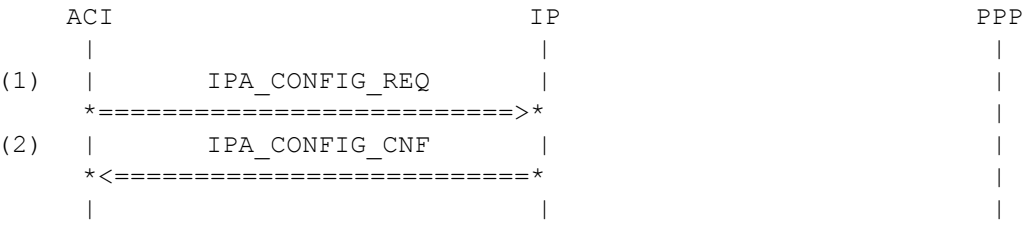
3.4.22 IP322: Configuration of IP, Interface goes down - less MTU for testing fragmenting

Description:

Upon Reception of IPA_CONFIG_REQ the IP entity sends a DTI2_GETDATA_REQ to PPP and confirms to ACI with IPA_CONFIG_CNF. Interface is down.

Preamble:

IP321



Parametrization:

Primitive	Parameter	Value
(1) IPA_CONFIG_REQ	ip	TEST_SRC_IP_ADDRESS
	peer_ip	TEST_PEER_IP_ADDRESS
	mtu	FRAGM_MTU_SIZE
	cmd	IPA_CONFIG_DOWN
(2) IPA_CONFIG_CNF	ack_flag	IPA_CONFIG_ACK
	all_down	IPA_ALLDOWN_TRUE

History:

29 September 2000	OFL	initial

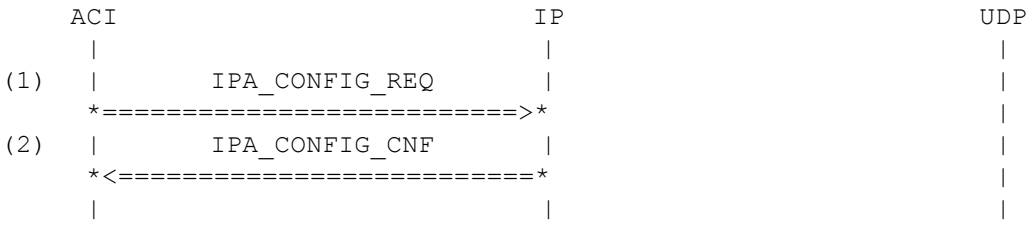
3.4.23 IP323: Configuration of IP, Interface goes up - less MTU for testing fragmenting

Description:

Upon Reception of IPA_CONFIG_REQ the IP entity sends a DTI2_GETDATA_REQ to PPP and confirms to ACI with IPA_CONFIG_CNF. It is now able to receive data from PPP or UDP and to relay it if appropriate.

Preamble:

IP322



Parametrization:

Primitive	Parameter	Value
(1) IPA_CONFIG_REQ	ip	TEST_SRC_IP_ADDRESS
	peer_ip	TEST_PEER_IP_ADDRESS
	mtu	FRAGM_MTU_SIZE
	cmd	IPA_CONFIG_UP
(2) IPA_CONFIG_CNF	ack_flag	IPA_CONFIG_ACK
	all_down	IPA_ALLDOWN_FALSE

History:

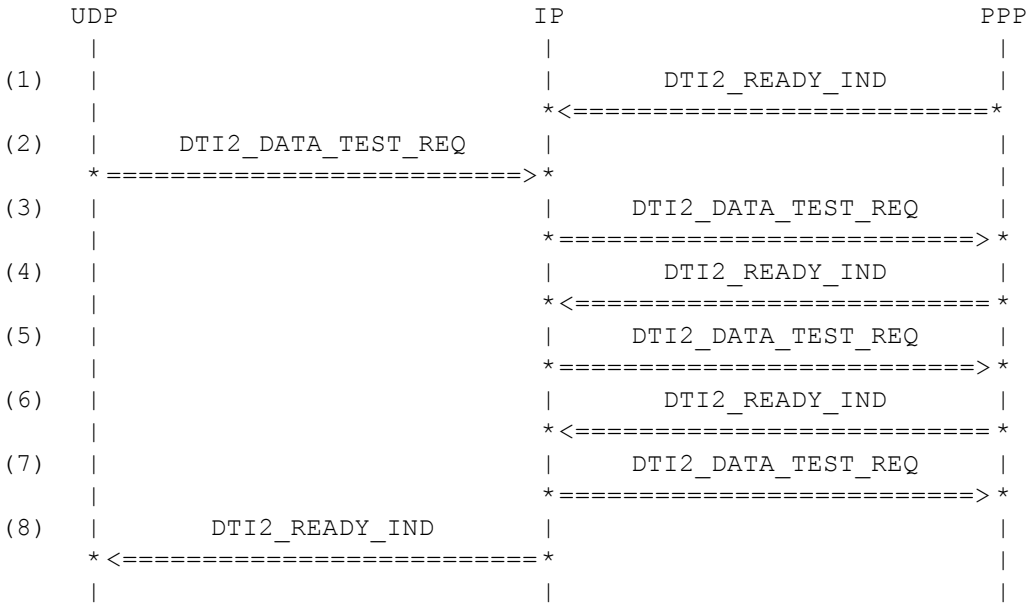
30 August 2000	OFL	initial

3.4.24 IP324: Fragmenting UDP -> IP -> PPP, receiving packet

Description: IP becomes a packet from higher layer which is bigger than MTU. IP build fragments.

Preamble:

IP323



Parametrization:

Primitive	Parameter	Value
(1) DTI2_READY_IND	link_id	PPP_LINK_ID
(2) DTI2_DATA_TEST_REQ	link_id	UDP_LINK_ID
	parameters	DTI_PARAMETER_UDP
	sdu	PKT_IP_OUT_FRAGM
(3) DTI2_DATA_TEST_REQ	link_id	PPP_LINK_ID
	parameters	DTI_PARAMETER_IP
	sdu	PKT_IP_IN_FIRST_FRAGM
(4) DTI2_READY_IND	link_id	PPP_LINK_ID
(5) DTI2_DATA_TEST_REQ	link_id	PPP_LINK_ID
	parameters	DTI_PARAMETER_IP
	sdu	PKT_IP_IN_SECOND_FRAGM
(6) DTI2_READY_IND	link_id	PPP_LINK_ID
(7) DTI2_DATA_TEST_REQ	link_id	PPP_LINK_ID
	parameters	DTI_PARAMETER_IP
	sdu	PKT_IP_IN_LAST_FRAGM
(8) DTI2_READY_IND	link_id	UDP_LINK_ID

History:

22 August 2000	xof	initial

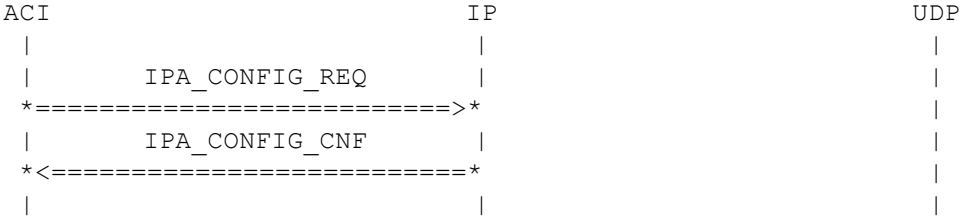
3.4.25 IP325: Fragmenting PPP -> IP -> UDP, receiving to of 3 fragments before deactivation of IP

Description:

The IP entity receives the IPA_DEACTIVATE_REQ and is switched to the deactivated state. IP informs ACI that the entity is deactivated.

Preamble:

IP315



Parametrization:

Primitive	Parameter	Value
(1) IPA_CONFIG_REQ	ip	OWN_IP_ADDRESS
	peer_ip	PEER_IP_ADDRESS
	mtu	NORMAL_MTU_SIZE
	cmd	IPA_CONFIG_DOWN
(2) IPA_CONFIG_CNF	ack_flag	IPA_CONFIG_ACK
	all_down	IPA_ALLDOWN_TRUE

History:

29 September 2000	xof	initial

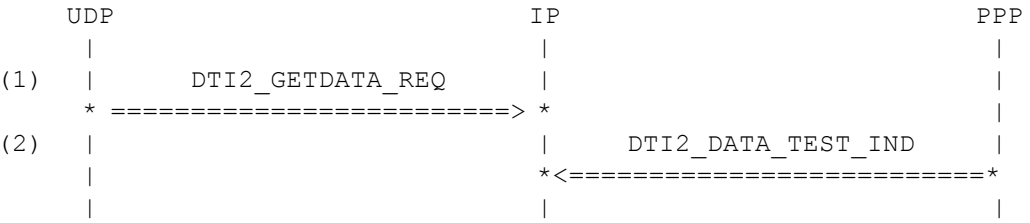
3.4.26 IP326: Fragmenting PPP -> IP -> UDP, after deactivation receiving last fragment - drop all fragments

Description:

The IP entity get DTI2_GETDATA_REQ
IP get the second fragment

Preamble:

IP325



Parametrization:

Primitive	Parameter	Value
(1) DTI2_GETDATA_REQ	link_id	UDP_LINK_ID
(2) DTI2_DATA_TEST_IND	link_id	PPP_LINK_ID
	parameters	DTI_PARAMETER_PPP
	sdu	PKT_IP_IN_LAST_FRAGM

History:

27 September 2000	xof	initial

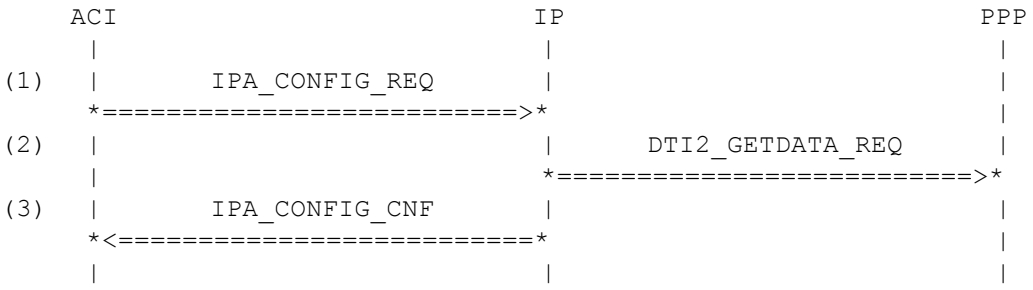
3.4.27 IP328: Configuration of IP, Interface goes up

Description:

Upon Reception of IPA_CONFIG_REQ the IP entity sends a DTI2_GETDATA_REQ to PPP and confirms to ACI with IPA_CONFIG_CNF.
It is now able to receive data from PPP or UDP and to relay it if appropriate.

Preamble:

IP100F



Parametrization:

Primitive	Parameter	Value
(1) IPA_CONFIG_REQ	ip	OWN_IP_ADDRESS
	peer_ip	PEER_IP_ADDRESS
	mtu	NORMAL_MTU_SIZE
	cmd	IPA_CONFIG_UP
(2) DTI2_GETDATA_REQ	link_id	PPP_LINK_ID
(3) IPA_CONFIG_CNF	ack_flag	IPA_CONFIG_ACK
	all_down	IPA_ALLDOWN_FALSE

History:

29 September 2000 xof init

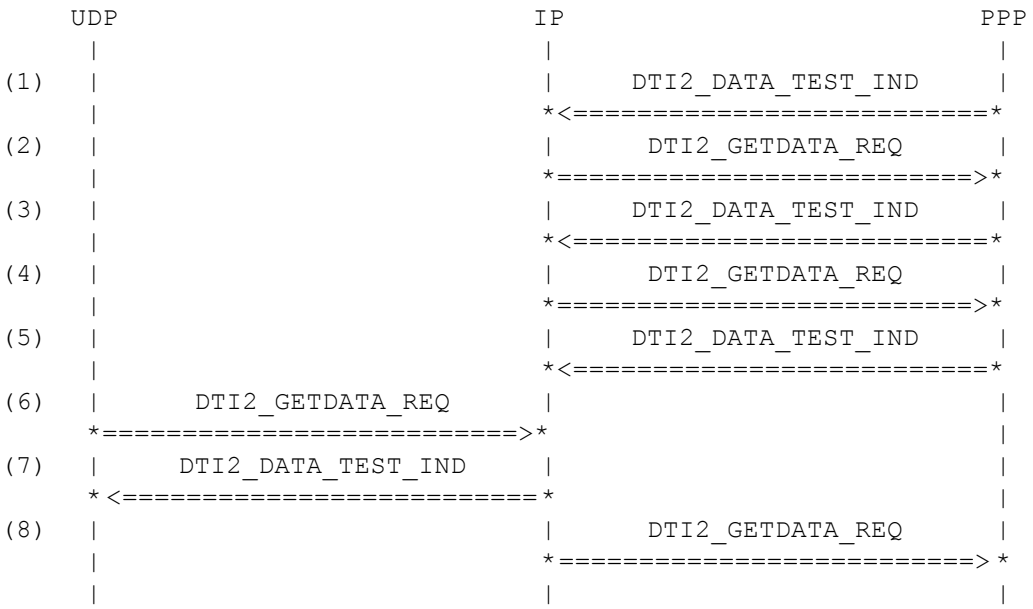
3.4.28 IP329: Fragmenting PPP -> IP -> UDP, 3 fragments

Description:

The IP entity receives tree fragments from PPP and build the complete packet to UDP .

Preamble:

IP328



Parametrization:

Primitive	Parameter	Value
(1) DTI2_DATA_TEST_IND	link_id	PPP_LINK_ID
	parameters	DTI_PARAMETER_PPP
	sdu	PKT_IP_IN_FIRST_FRAGM
(2) DTI2_GETDATA_REQ	link_id	PPP_LINK_ID
(3) DTI2_DATA_TEST_IND	link_id	PPP_LINK_ID
	parameters	DTI_PARAMETER_PPP
	sdu	PKT_IP_IN_SECOND_FRAGM
(4) DTI2_GETDATA_REQ	link_id	PPP_LINK_ID
(5) DTI2_DATA_TEST_IND	link_id	PPP_LINK_ID
	parameters	DTI_PARAMETER_PPP
	sdu	PKT_IP_IN_LAST_FRAGM
(6) DTI2_GETDATA_REQ	link_id	UDP_LINK_ID
(7) DTI2_DATA_TEST_IND	link_id	UDP_LINK_ID
	parameters	DTI_PARAMETER_IP
	sdu	PKT_IP_OUT_FRAGM

(8) DTI2_GETDATA_REQ	link_id	PPP_LINK_ID

History:

22 August 2000	xof	initial

3.4.29 IP330: Packet UDP -> IP -> PPP with broadcast address - drop packet

Description:

The IP entity receives a packet from UDP with broadcast address.
After appropriate processing, IP drop the packet.
IP answers to UDP with DTI2_READY_IND.

Preamble:

IP107

Parametrization:

Primitive	Parameter	Value
(1) DTI2_READY_IND	link_id	PPP_LINK_ID
(2) DTI2_DATA_TEST_REQ	link_id	UDP_LINK_ID
	parameters	DTI_PARAMETER_UDP
	sdu	IP_BRODCAST_DST_ADDR
(3) DTI2_READY_IND	link_id	UDP_LINK_ID

History:

27 September 2000	xof	start

3.5 Deactivated and/or Non-configured State (IP400 - IP4xx)

Most cases in this section don't yield a visible response from the IP entity, as the inputs are either ignored or only change internal state.
In order to check the correct (null) response in the test environment, the activation or configuration sequence (IP100A or IP101) is appended.
If this sequence completes without error, the input has been correctly ignored.

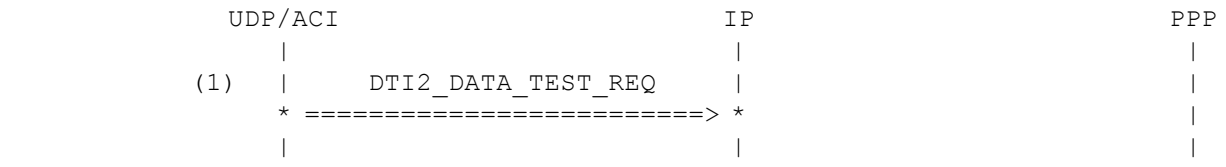
3.5.1 IP400: Reception of DTI2_DATA_TEST_REQ in Deactivated State

Description:

The IP entity receives DTI2_DATA_TEST_REQ from the transport layer and ignores it. Following activation proceeds normally.

Preamble:

IP001



Parametrization:

Primitive	Parameter	Value
(1) DTI2_DATA_TEST_REQ	link_id	UDP_LINK_ID
	parameters	DTI_PARAMETER_UDP
	sdu	PKT_UDP_NODATA_OUT_400

History:

19 June 2000	NI	initial
11 July 2000	xof	Fix bugs

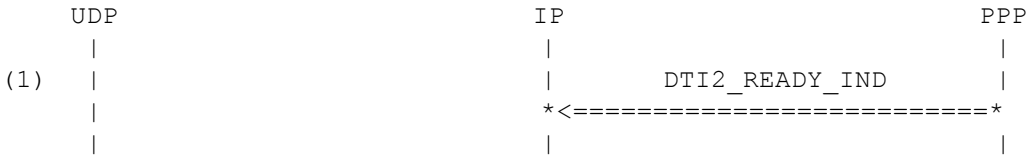
3.5.2 IP401: Reception of DTI2_READY_IND in Deactivated State

Description:

The IP entity receives a DTI2_READY_IND from an interface layer and stores it. Following activation completes normally.

Preamble:

IP001



Parametrization:

Primitive	Parameter	Value
(1) DTI2_READY_IND	link_id	PPP_LINK_ID

History:

19 June 2000	NI	initial
11 July 2000	xof	Fix bugs

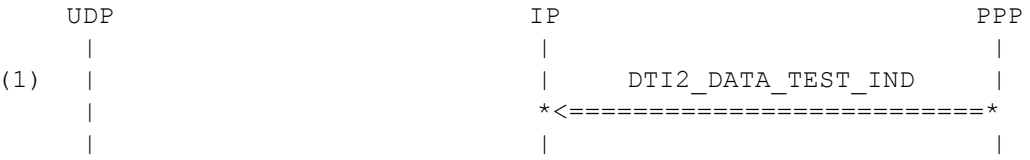
3.5.3 IP402: Reception of DTI2_DATA_TEST_IND in Deactivated State

Description:

The IP entity receives a DTI2_DATA_TEST_IND from the interface layer and ignores it. Following activation completes normally.

Preamble:

IP001



Parametrization:

Primitive	Parameter	Value
(1) DTI2_DATA_TEST_IND	link_id	PPP_LINK_ID
	parameters	DTI_PARAMETER_PPP
	sdu	PKT_UDP_NODATA_IN_402

History:

19 June 2000	NI	initial
11 July 2000	xof	Fix bugs

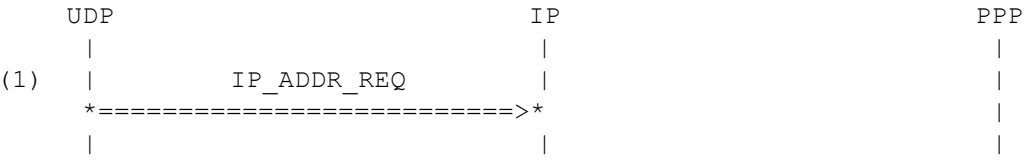
3.5.4 IP403: Reception of IP_ADDR_REQ in Deactivated State

Description:

The IP entity receives an IP_ADDR_REQ from the transport layer. IP returns an IP_ADDR_CNF with the error code IP_NOT_READY.

Preamble:

IP001



Parametrization:

Primitive	Parameter	Value
(1) IP_ADDR_REQ	dst_addr	PEER_IP_ADDRESS
	trans_prot	UDP_PROTOCOL

History:

19 June 2000	NI	initial
13 July 2000	xof	IP_NOT_READY changed to IP_ADDR_NOROUTE
5 December 2002	KJF	new IPA SAP

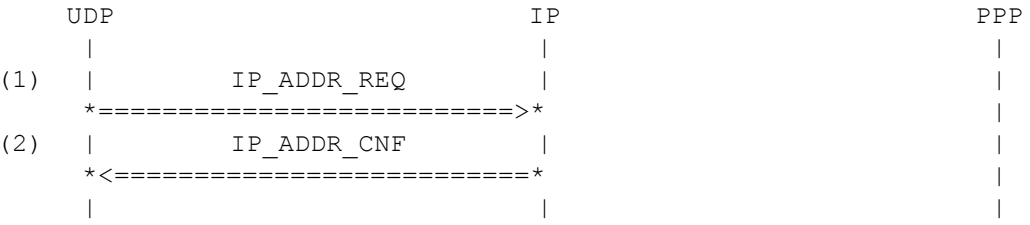
3.5.5 IP404: Reception of IP_ADDR_REQ in Non-configured State

Description:

The IP entity receives an IP_ADDR_REQ from the transport layer.
IP returns an IP_ADDR_CNF with the error code IP_ADDR_NOROUTE.

Preamble:

IP100A



Parametrization:

Primitive	Parameter	Value
(1) IP_ADDR_REQ	dst_addr	PEER_IP_ADDRESS
	trans_prot	UDP_PROTOCOL
(2) IP_ADDR_CNF	src_addr	OWN_IP_ADDRESS
	err	IP_ADDR_NOROUTE
	trans_prot	UDP_PROTOCOL

History:

19 June 2000	NI	initial
11 July 2000	xof	fix bug

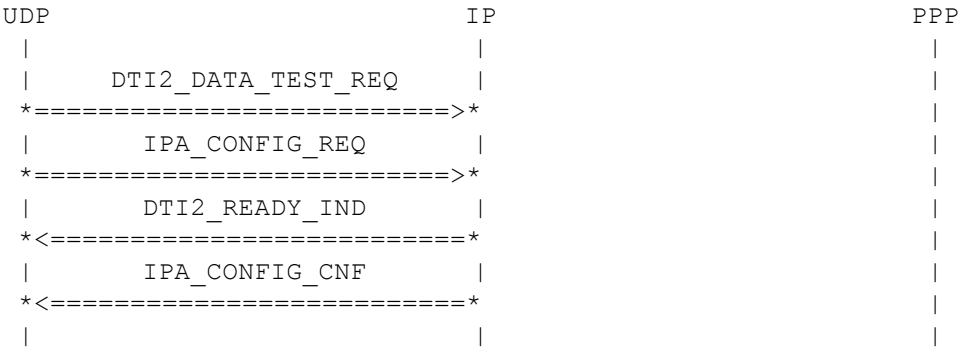
3.5.6 IP405: Reception of DTI2_DATA_TEST_REQ in Non-configured State

Description:

The IP entity receives a DTI2_DATA_TEST_REQ from the transport layer and ignores it. Following configuration completes normally.

Preamble:

IP100A



Parametrization:

Primitive	Parameter	Value
(1) DTI2_DATA_TEST_REQ	link_id	UDP_LINK_ID
	parameters	DTI_PARAMETER_UDP
	sdu	PKT_UDP_NODATA_OUT_404
(2) IPA_CONFIG_REQ	ip	OWN_IP_ADDRESS
	peer_ip	PEER_IP_ADDRESS
	mtu	NORMAL_MTU_SIZE
	cmd	IPA_CONFIG_UP
(3) DTI2_READY_IND	link_id	UDP_LINK_ID
(4) IPA_CONFIG_CNF	ack_flag	IPA_CONFIG_ACK
	all_down	IPA_ALLDOWN_FALSE

History:

19 June 2000	NI	initial
11 July 2000	xof	fix bug

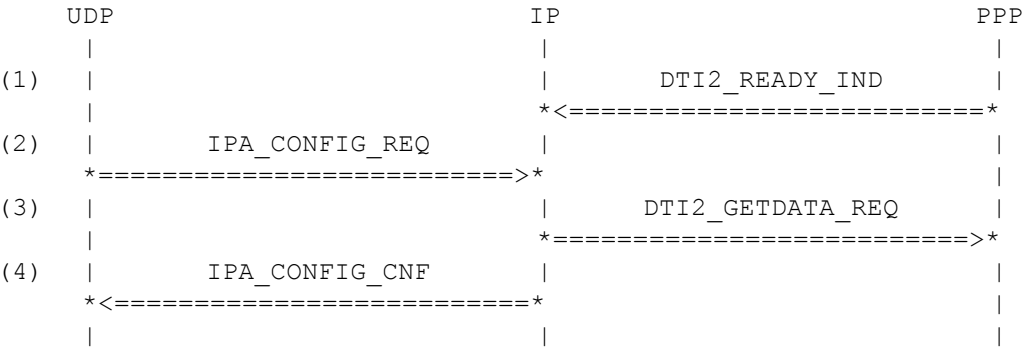
3.5.7 IP406: Reception of DTI2_READY_IND in Non-configured State

Description:

(1) The IP entity receives a DTI2_READY_IND from an interface layer and stores it.
(2) Following configuration completes normally.

Preamble:

IP100A



Parametrization:

Primitive	Parameter	Value
(1) DTI2_READY_IND	link_id	PPP_LINK_ID
(2) IPA_CONFIG_REQ	ip	OWN_IP_ADDRESS
	peer_ip	PEER_IP_ADDRESS
	mtu	NORMAL_MTU_SIZE
	cmd	IPA_CONFIG_UP
(3) DTI2_GETDATA_REQ	link_id	PPP_LINK_ID
(4) IPA_CONFIG_CNF	ack_flag	IPA_CONFIG_ACK
	all_down	IPA_ALLDOWN_FALSE

History:

19 June 2000	NI	initial
11 July 2000	xof	fix bug

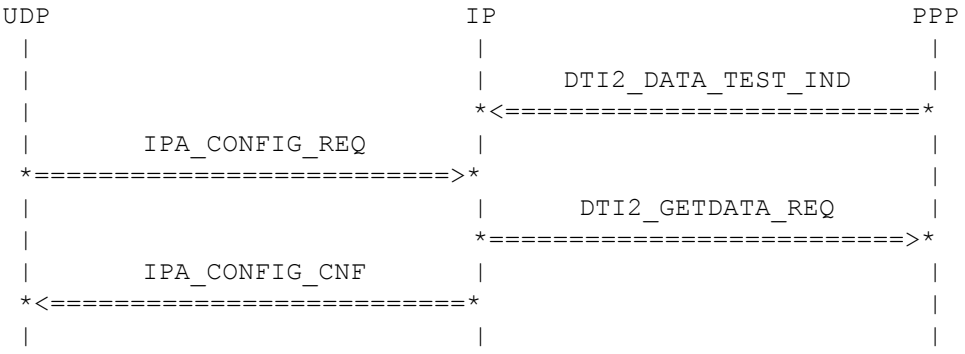
3.5.8 IP407: Reception of DTI2_DATA_TEST_IND in Non-configured State

Description:

The IP entity receives a DTI2_DATA_TEST_IND from the interface layer and ignores it. Following configuration completes normally.

Preamble:

IP100A



Parametrization:

Primitive	Parameter	Value
(1) DTI2_DATA_TEST_IND	link_id	PPP_LINK_ID
	parameters	DTI_PARAMETER_PPP
	sdu	PKT_UDP_NODATA_IN_406
(2) IPA_CONFIG_REQ	ip	OWN_IP_ADDRESS
	peer_ip	PEER_IP_ADDRESS
	mtu	NORMAL_MTU_SIZE
	cmd	IPA_CONFIG_UP
(3) DTI2_GETDATA_REQ	link_id	PPP_LINK_ID
(4) IPA_CONFIG_CNF	ack_flag	IPA_CONFIG_ACK
	all_down	IPA_ALLDOWN_FALSE

History:

19 June 2000	NI	initial
11 July 2000	xof	fix bug

3.6 ICMP Messages (IP500 - IP5xx)

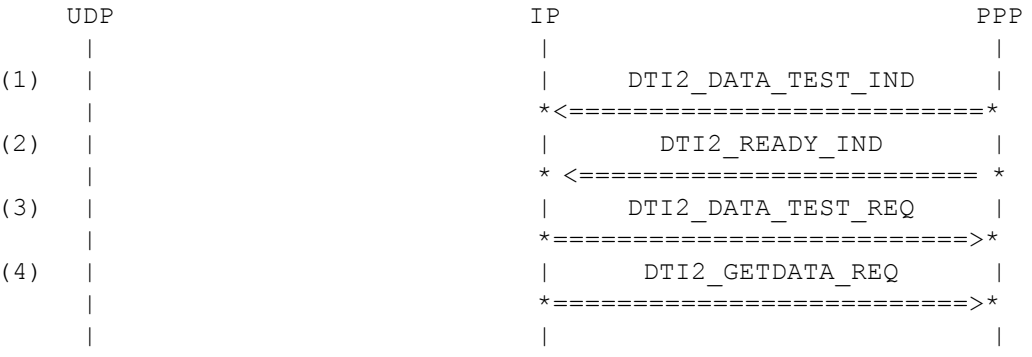
3.6.1 IP500: Incoming Packet for a Different Destination IP Address

Description:

The IP entity receives a DTI2_DATA_TEST_IND from the interface layer with a destination address that does not belong to one of its interfaces. The IP entity sends an ICMP destination unreachable message with code Host Unreachable back to the originating IP address as a DTI DATA_REQ to the interface layer.
The IP entity sends a DTI2_GETDATA_REQ to the interface layer

Preamble:

IP101



Parametrization:

Primitive	Parameter	Value
(1) DTI2_DATA_TEST_IND	link_id	PPP_LINK_ID
	parameters	DTI_PARAMETER_PPP
	sdu	PKT_UDP_NODATA_UNREACH_500
(2) DTI2_READY_IND	link_id	PPP_LINK_ID
(3) DTI2_DATA_TEST_REQ	link_id	PPP_LINK_ID
	parameters	DTI_PARAMETER_IP
	sdu	PKT_ICMP_UNREACH_500
(4) DTI2_GETDATA_REQ	link_id	PPP_LINK_ID

History:

19 June 2000	NI	initial
12 July 2000	xof	Put in the DTI2_READY_IND

3.6.2 IP501: ICMP Echo request and reply

Description:

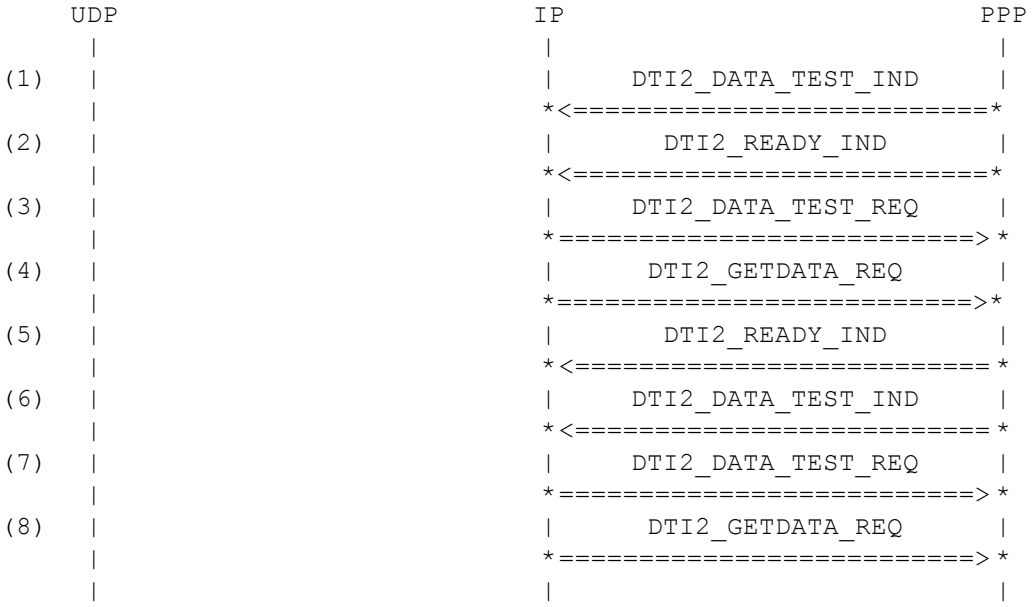
The IP entity receives a DTI2_DATA_TEST_IND from the interface layer with an ICMP echo request.

The IP entity sends an ICMP echo reply message back.

The IP entity sends a DTI2_GETDATA_REQ to the interface layer

Preamble:

IP108



Parametrization:

Primitive	Parameter	Value
(1) DTI2_DATA_TEST_IND	link_id	PPP_LINK_ID
	parameters	DTI_PARAMETER_PPP
	sdu	ICMP_ECHO_REQUEST_501
(2) DTI2_READY_IND	link_id	PPP_LINK_ID
(3) DTI2_DATA_TEST_REQ	link_id	PPP_LINK_ID
	parameters	DTI_PARAMETER_IP
	sdu	ICMP_ECHO_REPLY_501
(4) DTI2_GETDATA_REQ	link_id	PPP_LINK_ID
(5) DTI2_READY_IND	link_id	PPP_LINK_ID
(6) DTI2_DATA_TEST_IND	link_id	PPP_LINK_ID
	parameters	DTI_PARAMETER_PPP
	sdu	ICMP_ECHO_REQUEST_501

(7) DTI2_DATA_TEST_REQ	link_id	PPP_LINK_ID
	parameters	DTI_PARAMETER_IP
	sdu	ICMP_ECHO_REPLY_501
(8) DTI2_GETDATA_REQ	link_id	PPP_LINK_ID

History:

19 June 2000	NI	initial	
12 July 2000	xof		Put in the DTI2_READY_IND

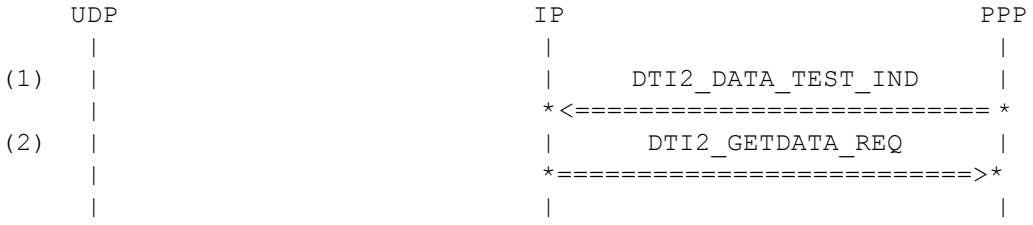
3.6.3 IP503: ICMP Packet with wrong destination address

Description:

The IP entity receives a DTI2_DATA_TEST_IND from the interface layer with a destination address
that does not belong to one of its interfaces.
IP drop the packet entity and sends a DTI2_GETDATA_REQ to PPP

Preamble:

IP101



Parametrization:

Primitive	Parameter	Value
(1) DTI2_DATA_TEST_IND	link_id	PPP_LINK_ID
	parameters	DTI_PARAMETER_PPP
	sdu	ICMP_WRONG_DEST
(2) DTI2_GETDATA_REQ	link_id	PPP_LINK_ID

History:

21 September 2000	xof	initial

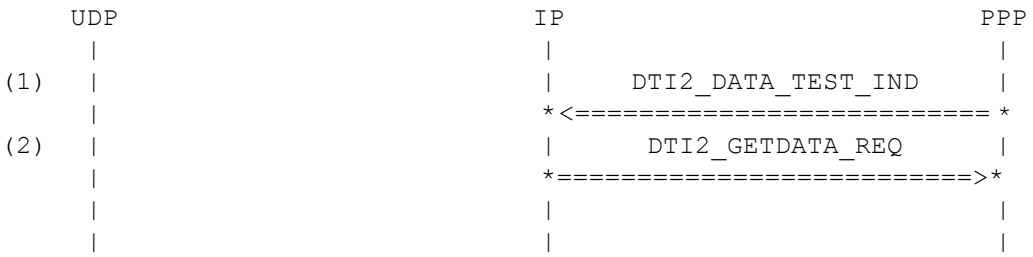
3.6.4 IP504: IP datagram with destination broadcast address

Description:

The IP entity receives a DTI2_DATA_TEST_IND from the interface layer with a broadcast destination address.
IP drop the packet entity and sends a DTI2_GETDATA_REQ to PPP. Send no ICMP message.

Preamble:

IP101



Parametrization:

Primitive	Parameter	Value
(1) DTI2_DATA_TEST_IND	link_id	PPP_LINK_ID
	parameters	DTI_PARAMETER_PPP
	sdu	UDP_IP_BRODCAST_DST_ADDR
(2) DTI2_GETDATA_REQ	link_id	PPP_LINK_ID

History:

21 September	2000	xof	initial

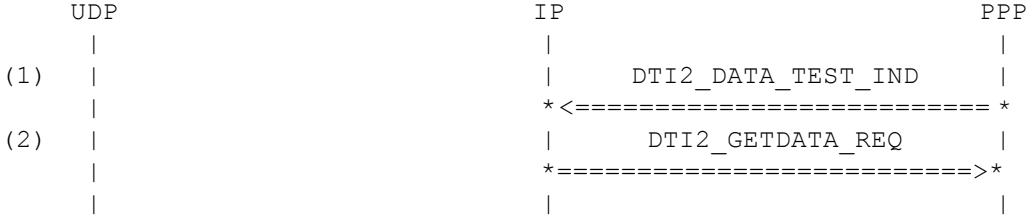
3.6.5 IP505: IP datagram with wrong destination address and broadcast source-address

Description:

The IP entity receives a DTI2_DATA_TEST_IND from the interface layer with wrong destination address and with a broadcast source address.

Preamble:

IP106



Parametrization:

Primitive	Parameter	Value
(1) DTI2_DATA_TEST_IND	link_id	PPP_LINK_ID
	parameters	DTI_PARAMETER_PPP
	sdu	IP_BRODCAST_SRC_ADDR
(2) DTI2_GETDATA_REQ	link_id	PPP_LINK_ID

History:

21 September 2000	xof	initial

3.7 Fault tolerance (IP600 - IP6xx)

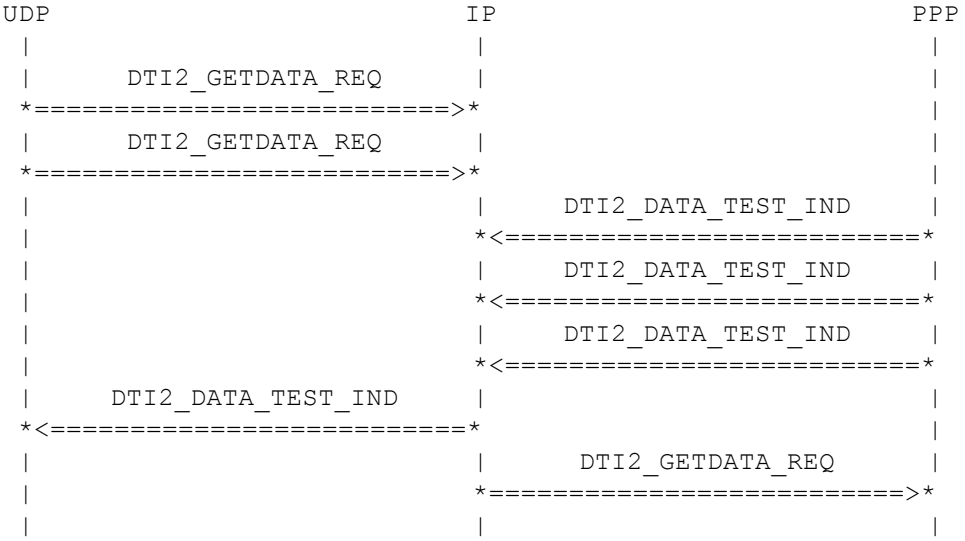
3.7.1 IP600: IP receives tree data packets from PPP

Description:

IP receives two DTI2_GETDATA_REQ and three DTI2_DATA_TEST_IND packets from PPP. IP have different send state for the tree packets.
The first packet is send.
The middle packet is overwritten from the last packet and can be send.

Preamble:

IP101



Parametrization:

Primitive	Parameter	Value
(1) DTI2_GETDATA_REQ	link_id	UDP_LINK_ID
(2) DTI2_GETDATA_REQ	link_id	UDP_LINK_ID
(3) DTI2_DATA_TEST_IND	link_id	PPP_LINK_ID
	parameters	DTI_PARAMETER_PPP
	sdu	PKT_UDP_NODATA_IN_300
(4) DTI2_DATA_TEST_IND	link_id	PPP_LINK_ID
	parameters	DTI_PARAMETER_PPP
	sdu	PKT_UDP_NODATA_IN_300
(5) DTI2_DATA_TEST_IND	link_id	PPP_LINK_ID
	parameters	DTI_PARAMETER_PPP
	sdu	PKT_UDP_NODATA_IN_300

(6) DTI2_DATA_TEST_IND	link_id	UDP_LINK_ID
	parameters	DTI_PARAMETER_IP
	sdu	PKT_UDP_NODATA_IN_300
(7) DTI2_GETDATA_REQ	link_id	PPP_LINK_ID

History:

18 September 2000	xof	start

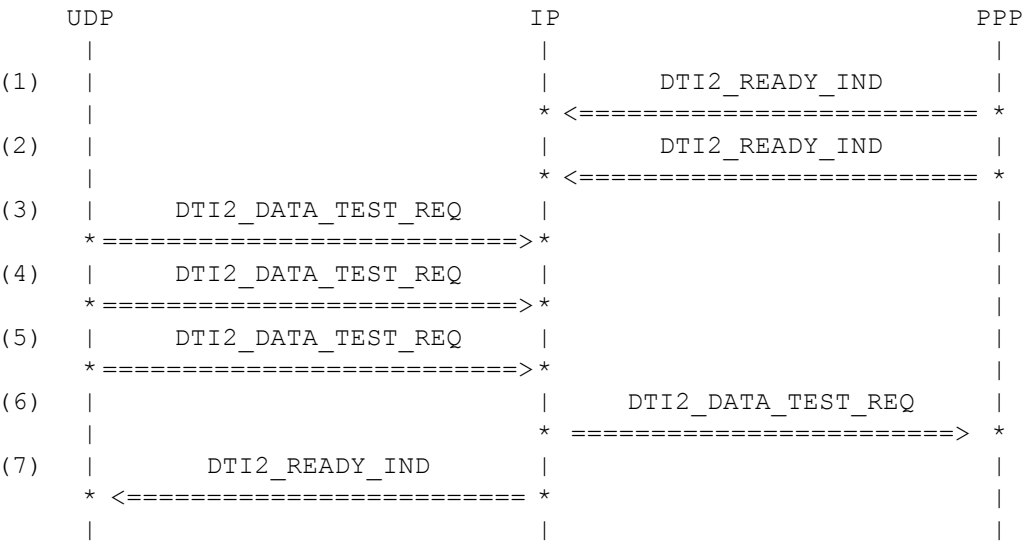
3.7.2 IP601: IP receives tree data packets from UDP

Description:

IP receives two DTI2_READY_IND and three DTI2_DATA_TEST_REQ packets from UDP.
IP have different send state for the tree packets.
The first packet is send.
The middle packet is overwritten from the last packet.

Preamble:

IP105



Parametrization:

Primitive	Parameter	Value
(1) DTI2_READY_IND	link_id	PPP_LINK_ID
(2) DTI2_READY_IND	link_id	PPP_LINK_ID
(3) DTI2_DATA_TEST_REQ	link_id	UDP_LINK_ID
	parameters	DTI_PARAMETER_UDP
	sdu	PKT_UDP_NODATA_OUT_1_302
(4) DTI2_DATA_TEST_REQ	link_id	UDP_LINK_ID
	parameters	DTI_PARAMETER_UDP
	sdu	PKT_UDP_NODATA_OUT_1_302
(5) DTI2_DATA_TEST_REQ	link_id	UDP_LINK_ID
	parameters	DTI_PARAMETER_UDP
	sdu	PKT_UDP_NODATA_OUT_1_302
(6) DTI2_DATA_TEST_REQ	link_id	PPP_LINK_ID
	parameters	DTI_PARAMETER_IP
	sdu	PKT_UDP_NODATA_OUT_2_302

(7)	DTI2_READY_IND	link_id	UDP_LINK_ID

History:

18 September 2000	xof	start

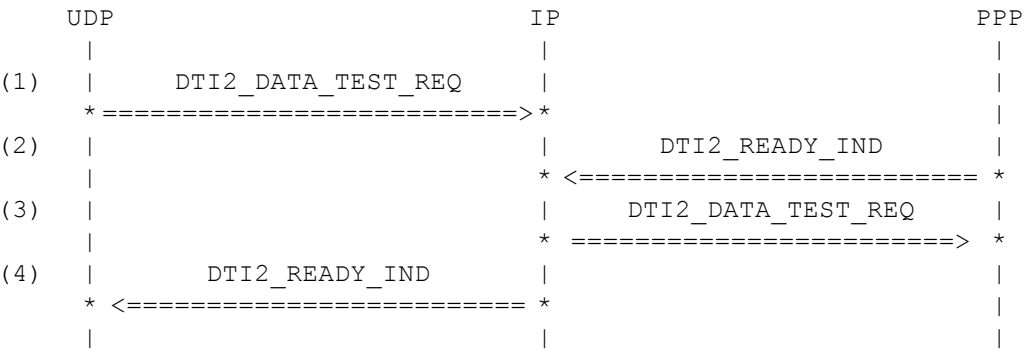
3.7.3 IP602: IP receives tree data packets from UDP

Description:

IP receives DTI2_READY_IND after DTI2_DATA_TEST_REQ packet. IP send the packet normaly.

Preamble:

IP105



Parametrization:

Primitive	Parameter	Value
(1) DTI2_DATA_TEST_REQ	link_id	UDP_LINK_ID
	parameters	DTI_PARAMETER_UDP
	sdu	PKT_UDP_NODATA_OUT_1_302
(2) DTI2_READY_IND	link_id	PPP_LINK_ID
(3) DTI2_DATA_TEST_REQ	link_id	PPP_LINK_ID
	parameters	DTI_PARAMETER_IP
	sdu	PKT_UDP_NODATA_OUT_2_302
(4) DTI2_READY_IND	link_id	UDP_LINK_ID

History:

18 September 2000	xof	start

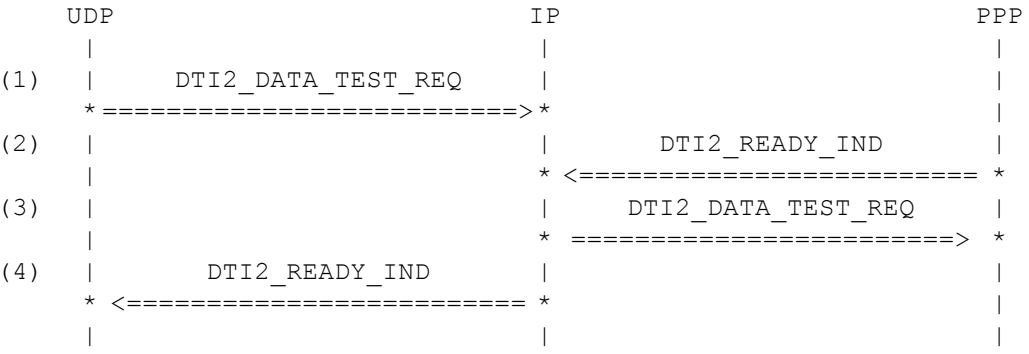
3.7.4 IP603: IP get DTI2_READY_IND after DTI2_DATA_TEST_REQ

Description:

IP receives DTI2_GETDATA_REQ after DTI2_DATA_TEST_REQ from UDP.
IP fills in header information and sends the packet to PPP.

Preamble:

IP105



Parametrization:

Primitive	Parameter	Value
(1) DTI2_DATA_TEST_REQ	link_id	UDP_LINK_ID
	parameters	DTI_PARAMETER_UDP
	sdu	PKT_UDP_NODATA_OUT_1_302
(2) DTI2_READY_IND	link_id	PPP_LINK_ID
(3) DTI2_DATA_TEST_REQ	link_id	PPP_LINK_ID
	parameters	DTI_PARAMETER_IP
	sdu	PKT_UDP_NODATA_OUT_2_302
(4) DTI2_READY_IND	link_id	UDP_LINK_ID

History:

18 September 2000 xof initial

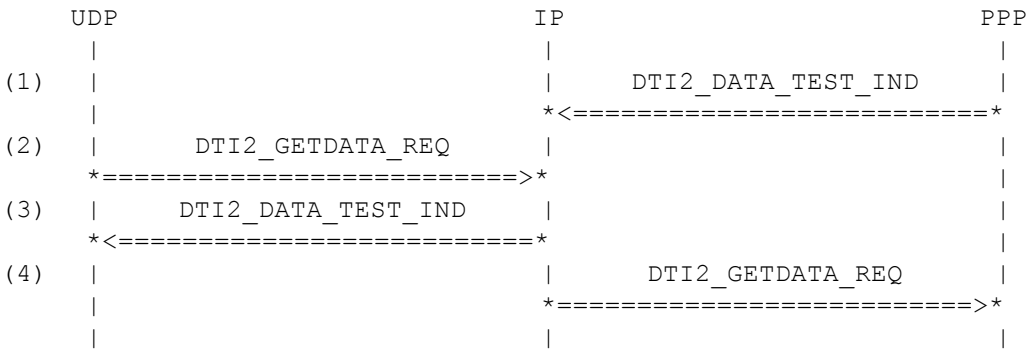
3.7.5 IP604: Packet PPP -> IP -> UDP, with DTI2_GETDATA_REQ after DTI2_DATA_TEST_IND

Description:

The IP entity receives a valid UDP packet from PPP.
IP receives DTI2_GETDATA_REQ.
After appropriate processing, IP relays the packet to UDP.
IP answers to PPP with DTI2_GETDATA_REQ.

Preamble:

IP101



Parametrization:

Primitive	Parameter	Value
(1) DTI2_DATA_TEST_IND	link_id	PPP_LINK_ID
	parameters	DTI_PARAMETER_PPP
	sdu	PKT_UDP_NODATA_IN_300
(2) DTI2_GETDATA_REQ	link_id	UDP_LINK_ID
(3) DTI2_DATA_TEST_IND	link_id	UDP_LINK_ID
	parameters	DTI_PARAMETER_IP
	sdu	PKT_UDP_NODATA_IN_300
(4) DTI2_GETDATA_REQ	link_id	PPP_LINK_ID

History:

18 September 2000 xof initial