

DIN EN 14908-1

ICS 97.120; 35.240.99; 91.140.01

Entwurf

Einsprüche bis 2012-11-24
Vorgesehen mit
DIN EN 14908-1:2007-11
als Ersatz für
die 2007-11 zurückgezogene
Norm
DIN EN 14908-1:2006-01

**Firmenneutrale Datenkommunikation für die Gebäudeautomation und
Gebäudemanagement –
Gebäudedatennetzprotokoll –
Teil 1: Datenprotokollschichtenmodell;
Englische Fassung FprEN 14908-1:2012**

Open Data Communication in Building Automation, Controls and Building Management –
Control Network Protocol –
Part 1: Protocol Stack;
English version FprEN 14908-1:2012

Réseau ouvert de communication de données pour l'automatisation, la régulation et la
gestion techniques du bâtiment –
Protocole de réseau pour le bâtiment –
Partie 1: Niveaux du protocole;
Version anglaise FpEN 14908-1:2012

Anwendungswarnvermerk

Dieser Norm-Entwurf mit Erscheinungsdatum 2012-09-24 wird der Öffentlichkeit zur Prüfung und
Stellungnahme vorgelegt.

Weil die beabsichtigte Norm von der vorliegenden Fassung abweichen kann, ist die Anwendung dieses
Entwurfes besonders zu vereinbaren.

Stellungnahmen werden erbeten

- vorzugsweise als Datei per E-Mail an nhrs@din.de in Form einer Tabelle. Die Vorlage dieser Tabelle kann
im Internet unter www.din.de/stellungnahme oder für Stellungnahmen zu Norm-Entwürfen der DKE unter
www.dke.de/stellungnahme abgerufen werden;
- oder online im Norm-Entwurfs-Portal des DIN unter www.entwuerfe.din.de, sofern dort wiedergegeben;
- oder in Papierform an den Normenausschuss Heiz- und Raumluftechnik (NHRs) im DIN, 10772 Berlin
(Hausanschrift: Burggrafenstr. 6, 10787 Berlin).

Die Empfänger dieses Norm-Entwurfs werden gebeten, mit ihren Kommentaren jegliche relevanten
Patentrechte, die sie kennen, mitzuteilen und unterstützende Dokumentationen zur Verfügung zu stellen.

Gesamtumfang 427 Seiten

Normenausschuss Heiz- und Raumluftechnik (NHRs) im DIN
DKE Deutsche Kommission Elektrotechnik Elektronik Informationstechnik im DIN und VDE

Nationales Vorwort

Dieses Dokument (FprEN 14908-1:2012) wurde vom Technischen Komitee CEN/TC 247 „Building environment design“ erarbeitet, dessen Sekretariat vom SNV (Schweiz) gehalten wird.

Das zuständige deutsche Gremium ist der Arbeitsausschuss NA 041-03-68 AA „Gebäudeautomation und -kommunikation (SpA CEN/TC 247/WG 4)“ im Normenausschuss Heiz- und Raumlufttechnik (NHRS) im DIN.

Der englische Originaltext dieser Europäischen Norm wurde unverändert übernommen. Das Präsidium des DIN hat mit seinem Beschluss 1/2004 festgelegt, dass von dem in den Regeln der europäischen Normungsarbeit von CEN/CENELEC verankerten Grundsatz, wonach Europäische Normen in den drei offiziellen Sprachen Deutsch, Englisch und Französisch veröffentlicht werden, in begründeten Ausnahmefällen abgewichen und auf die deutsche Sprachfassung verzichtet werden kann. Die Genehmigung dafür hat die DIN-Geschäftsleitung entsprechend ihren in Anlage 1 zu dem DIN-Rundschreiben A5/2004 festgelegten Kriterien für die vorliegende Norm auf Antrag des NHRS als Ergebnis einer Einzelfallentscheidung erteilt, zumal bereits dieser Norm zugrunde liegende Papiere überwiegend in englischer Sprache auch von den deutschen Marktteilnehmern angewendet werden.

Da sich die Anwender der vorliegenden Norm der englischen Sprache als Fachsprache bedienen, wird die englische Fassung der EN 14908-1 veröffentlicht.

Die Europäische Norm EN 14908-1 ist Teil einer Serie von Normen, die sich mit dem Kommunikationssystem „Control Network Protocol“ befassen, das auf der LonWorks-Technologie basiert. In dieser Serie werden sowohl Hardware als auch Software spezifiziert, die für die Funktion des Kommunikationssystems notwendig sind. Des Weiteren wird es Teile geben, in der Funktionen, Profile und Konformitätstests beschrieben werden.

In EN 14908-1 wird der so genannte „Protocol Stack“ definiert. Dieser „Protokoll-Stapel“ beschreibt das Kommunikationssystem anhand des ISO/OSI Basis-Referenzmodells ISO 7498, das eine Kommunikation in bis zu sieben Schichten aufteilt mit jeweils einem eigenen Protokoll. Im Falle des Control Network Protocol werden alle sieben Schichten und die Schnittstellen zwischen den Schichten spezifiziert.

Dabei sind die äußeren Schnittstellen zur Bitübertragungsschicht (Physical Layer) und zur Anwendungsschicht (Application Layer) von besonderer Bedeutung, weil dies in der Regel die Schnittstellen sind, an die ein Anwender sein Gerät, seine Software usw. anbinden muss. Da es für das Building Network Protocol verschiedene Medien für die Übertragung gibt, wie z. B. verdrehte Zweidrahtleitung (Twisted Pair) und Powerline-Kommunikation, gibt es in dieser Normenreihe dafür eigene Teile, die sich an den allgemeinen Teil der Spezifikation Schicht 1 anpassen.

Um die Spezifikation für diejenigen verständlicher zu machen, die das Control Network Protocol implementieren wollen, ist im Anhang A eine Referenz-Implementierung aufgeführt, die beispielhaft einige Teile der Spezifikation in Form von Programm-Teilen beschreibt.

DIN EN 14908 besteht aus den folgenden Teilen unter dem Haupttitel *Firmenneutrale Datenkommunikation für die Gebäudeautomation und Gebäudemanagement*:

- Teil 1: Datenprotokollschichtenmodell
- Teil 2: Kommunikation über paarig verdrehte Leitungen
- Teil 3: Kommunikation über die Stromversorgungsleitungen
- Teil 4: Kommunikation mittels Internet Protokoll (IP)
- Teil 5: Implementierungsrichtlinie

Änderungen

Gegenüber DIN EN 14908-1:2007-11 wurden folgende Änderungen vorgenommen:

- a) EN 14908-5:2009 wurde in die normativen Verweisungen aufgenommen;
- b) der normative Anhang A wurde zum besseren Verständnis überarbeitet. Die Referenz-Implementierung dieser Norm zeigt detailliert welcher Teil normativ und hardwareunabhängig, welcher normativ und hardwareabhängig und welcher aufgrund von Hardwareabhängigkeit nicht normativ ist. Diese Informationen unterstützen die Entwicklung eines Datenprotokollschichtenmodells und das Verständnis für die spezifizierten Kommunikationsdienste.

Nationaler Anhang NA (informativ)

Begriffe

Im Folgenden wird EN 14908-1:2012, Abschnitt 3 ins Deutsche übersetzt. Die Benennung der folgenden Begriffe und Abkürzungen ist identisch mit der Benennung in der englischen Fassung.

3 Begriffe

Für die Anwendung dieses Dokuments gelten die folgenden Begriffe.

Für die Anwendung dieser Europäischen Norm enthält der folgende Unterabschnitt die wesentlichen Begriffe, die in dieser Europäischen Norm gebraucht werden. Die meisten von ihnen sind allgemein gebräuchlich und die Benennungen besitzen sowohl im allgemeinen Kontext als auch im Normkontext dieselbe Bedeutung. Dennoch bestehen für einige Benennungen feine Unterschiede. Beispielsweise unternehmen Bridges eine selektive Weiterleitung basierend auf der Zieladresse der Schicht 2. Es gibt keine Adressen der Schicht 2 in dem Protokoll dieser Norm, sodass Bridges alle Pakete weiterleiten, solange die Domainadresse im Paket mit einer Domain übereinstimmt, welche zur Bridge gehört. Router führen im Allgemeinen eine Netzwerk-Adressenänderung durch, sodass zwei Protokolle mit derselben Transportschicht aber unterschiedlichen Vermittlungsschichten verbunden werden können, um ein einzelnes logisches Netzwerk zu bilden. Router dieser Norm können eine Netzwerkadressenänderung vornehmen, sie untersuchen aber in der Regel nur die Netzwerkadressenfelder und leiten Pakete selektiv auf Grundlage der Vermittlungsschicht-Adressenfelder weiter.

3.1

Kanal

(en: channel)

physikalische Bandbreiteneinheit, die einen oder mehrere Kommunikationsknoten verbindet.

Anmerkung 1 zum Begriff: Siehe Anhang E für weiterführende Erläuterungen der Beziehung zwischen einem Kanal und einem Subnetz.

3.2

physikalischer Repeater

(en: physical repeater)

Gerät, welches das ankommende Signal der Bitübertragungsschicht an einem Kanal aufbereitet und es auf einem anderen Kanal weitersendet.

3.3

Store-and-forward-Repeater

(en: store-and-forward repeater)

Gerät, das Datenpakete speichert und sie anschließend auf einem zweiten Kanal weiterleitet.

3.4

Bridge

(en: bridge)

Gerät, das zwei Kanäle (x und y) miteinander verbindet; es leitet alle Pakete von x an y und umgekehrt weiter, solange die Pakete ihren Ursprung an einer der Domains besitzen, zu der die Bridge gehört.

3.5

Konfiguration

(en: configuration)

vom Gerät verwendete nicht flüchtige Daten zum Anpassen seiner Funktion. Es gibt Konfigurationsdaten für die ordnungsgemäße Funktion des Protokolls in jedem Gerät, und optionale für die Anwendungsfunktion. Die Netzwerkkonfigurationsdaten, die in jedem Gerät gespeichert sind, verfügen über eine den Daten entsprechende Prüfsumme. Beispiele der Netzwerkkonfigurationsdaten sind die Knoten-Adressen, Kommunikations-Medienparameter wie z. B. Prioritätensetzung usw. Daten der Anwendungskonfiguration sind anwendungs-spezifisch.

3.6

Domain

(en: domain)

virtuelles Netzwerk, das die Netzwerkeinheit für Steuerung und Verwaltung darstellt. Gruppenadressen und Subnetzadressen (siehe nachstehend) werden vom Administrator vergeben, der für die Domain verantwortlich ist, und sie besitzen nur im Kontext dieser Domain eine Bedeutung.

3.7

flexible Domain

(en: flexible domain)

verwendet in Verbindung mit der Unique_Node_ID und der Broadcast-Adressierung. Ein Knoten antwortet einer an die Unique_Node_ID gerichteten Nachricht, wenn die Adresse übereinstimmt, unabhängig von der Domain, an welche die Nachricht gesendet wurde. Umso zu antworten, dass der Sender der Nachricht die Antwort empfängt, muss die Antwort an die Domain gesendet werden, von der sie empfangen wurde. Außerdem muss sich an diese Domain für die Dauer der Transaktion erinnert werden, damit ein doppeltes Erkennen von Wiederholungen möglich ist. Dieser vorübergehende Eintrag der Domain an einem Knoten wird als flexible Domain bezeichnet. Wie viele flexible Domaineinträge ein Knoten unterstützt, hängt von der Umsetzung ab. Mindestens ein Eintrag ist jedoch erforderlich.

3.8

Subnetz

Teilnetz

(en: subnet)

Reihe von Knoten, die über dasselbe Sicherungsschicht-Protokoll zugänglich sind; Routing-Abstraktion für einen Kanal; in dieser Norm sind Subnetze auf maximal 127 Knoten beschränkt.

3.9

Knoten

(en: node)

Abstraktion für einen physikalischen Knoten, der den höchsten Grad der Adresslösbarkeit in einem Netzwerk besitzt. Ein Knoten wird innerhalb eines Subnetzes durch seine (logische) Knoten-ID gekennzeichnet (adressiert). Ein physikalischer Knoten kann zu mehr als einem Subnetz gehören; wenn dies der Fall ist, wird ihm eine (logische) Knoten-Nummer für jedes Subnetz, zu dem er gehört, zugewiesen. Ein physikalischer Knoten kann zu höchstens zwei Subnetzen gehören; diese Subnetze müssen in unterschiedlichen Domains liegen. Ein Knoten kann auch innerhalb eines Netzwerks mithilfe seiner Unique_Node_ID absolut gekennzeichnet werden.

3.10

Gruppe

(en: group)

eindeutig identifizierbare Reihe von Knoten innerhalb einer Domain. Innerhalb dieser Reihe werden einzelne Elemente durch ihre Nummer gekennzeichnet. Gruppen ermöglichen die Einer-an-Alle-Kommunikation und sind dafür vorgesehen, die funktionale Adressierung zu unterstützen.

3.11

Router

(en: router)

Gerät, das Datenpakete durch selektive Weiterleitung von Subnetz zu Subnetz an ihre jeweiligen Ziele leitet; ein Router verbindet stets zwei (Gruppen von) Subnetze(n); Router können Vermittlungsschicht-Adressfelder ändern. Router können auf einen von vier Modi eingestellt werden: Repeater-Modus, Bridge-Modus, Lern-Modus und Konfigurations-Modus. Im Repeater-Modus werden die Pakete weitergeleitet, sofern sie ohne Fehler empfangen wurden. Im Bridge-Modus werden die Pakete weitergeleitet, sofern sie ohne Fehler empfangen wurden und einer Domain entsprechen, der der Router angehört. Router im Lern-Modus erlernen die Topologie, indem der Paketverkehr untersucht wird, während Router, die im Konfigurations-Modus eingestellt sind, die Netzwerktopologie in ihren Speichern abgelegt haben und deren Routing-Entscheidungen ausschließlich von den Inhalten ihrer Konfigurationstabellen abhängen.

3.12

(Anwendungs-)Gateway

(en: (application) gateway)

verbindet Netzwerke an ihren höchsten Protokollschichten miteinander (oft zwei unterschiedliche Protokolle). Zwei Domains können auch durch ein Anwendungs-Gateway verbunden werden.

3.13

Beta1

(en: Beta1)

Periode, die unmittelbar an das Ende eines Paketzklus anschließt. Ein Knoten, der eine Übertragung anstrebt, überwacht den Zustand des Kanals und, wenn er keine Übertragung während der Beta1-Periode erkennt, hält er den Kanal für nicht belegt.

3.14

Beta2

(en: Beta2)

mit Zufallszahlen arbeitender Slot. Ein übertragungswilliger Knoten erzeugt eine willkürliche Verzögerungszeit T. Diese Verzögerungszeit ist eine ganze Zahl der mit Zufallszahlen arbeitenden Slots mit der Dauer Beta2.

3.15

Netzwerkvariable

(en: network variable)

Variable in einem Anwendungsprogramm, deren Wert automatisch über das Netzwerk verbreitet wird, sobald ihr ein neuer Wert zugewiesen wird.

3.16

Standardnetzwerkvariablen

SNVTs

(en: Standard Network Variable Types)

Variablen mit vereinbarter Semantik. Diese Variablen werden von allen Anwendungen auf die gleiche Weise interpretiert und bilden die Grundlage für die Interoperabilität. Die Definition von spezifischen SNVTs liegt außerhalb des Anwendungsbereichs dieser Europäischen Norm.

3.17

Nachricht der manuellen Dienste-Anforderung

(en: manual service request message)

Nachricht des Netzwerkmanagements, welche die Unique_Node_ID eines Knotens enthält. Sie wird von einem Netzwerkmanagement-Gerät verwendet, das diese Nachricht empfängt, um den Knoten zu installieren und zu konfigurieren. Sie kann durch die Anwendung oder durch den Systemcode erzeugt werden. Sie kann von einem externen Hardwareereignis ausgelöst werden, z. B. Auslösen einer Dienste-Anforderung.

3.18

Transaktion

(en: transaction)

Folge von Nachrichten, die miteinander in Beziehung stehen. Beispielsweise eine Anforderung und die Antworten auf die Anforderung sind alle Teil einer einzelnen Transaktion. Eine Transaktion ist erfolgreich, wenn alle erwarteten Nachrichten von jedem an der Transaktion beteiligten Knoten mindestens einmal empfangen werden. Eine Transaktion scheitert nach dieser Europäischen Norm, wenn eine der erwarteten Nachrichten bei der Transaktion nicht empfangen wird. Wiederholungen von Nachrichten bei einer Transaktion werden verwendet, um die Erfolgswahrscheinlichkeit einer Transaktion bei Auftreten von vorübergehenden Fehlern zu erhöhen.

— Leerseite —

October 2012

ICS 35.240.99; 91.140.01; 97.120

Will supersede EN 14908-1:2005

English Version

Open Data Communication in Building Automation, Controls and Building Management - Control Network Protocol - Part 1: Protocol Stack

Firmenneutrale Datenkommunikation für die
Gebäudeautomation und Gebäudemanagement -
Gebäudedatennetzprotokoll - Teil 1:
Datenprotokollschichtenmodell

This draft European Standard is submitted to CEN members for unique acceptance procedure. It has been drawn up by the Technical Committee CEN/TC 247.

If this draft becomes a European Standard, CEN members are bound to comply with the CEN/CENELEC Internal Regulations which stipulate the conditions for giving this European Standard the status of a national standard without any alteration.

This draft European Standard was established by CEN in three official versions (English, French, German). A version in any other language made by translation under the responsibility of a CEN member into its own language and notified to the CEN-CENELEC Management Centre has the same status as the official versions.

CEN members are the national standards bodies of Austria, Belgium, Bulgaria, Croatia, Cyprus, Czech Republic, Denmark, Estonia, Finland, Former Yugoslav Republic of Macedonia, France, Germany, Greece, Hungary, Iceland, Ireland, Italy, Latvia, Lithuania, Luxembourg, Malta, Netherlands, Norway, Poland, Portugal, Romania, Slovakia, Slovenia, Spain, Sweden, Switzerland, Turkey and United Kingdom.

Recipients of this draft are invited to submit, with their comments, notification of any relevant patent rights of which they are aware and to provide supporting documentation.

Warning : This document is not a European Standard. It is distributed for review and comments. It is subject to change without notice and shall not be referred to as a European Standard.



EUROPEAN COMMITTEE FOR STANDARDIZATION
COMITÉ EUROPÉEN DE NORMALISATION
EUROPÄISCHES KOMITEE FÜR NORMUNG

Management Centre: Avenue Marnix 17, B-1000 Brussels

Contents

	Page
Foreword	7
Introduction.....	8
1 Scope	9
2 Normative references	9
3 Terms and definitions	9
4 Symbols and abbreviations.....	11
4.1 Symbols and graphical representations	11
4.2 Abbreviations.....	12
5 Overview of protocol layering.....	13
6 MAC sublayer.....	15
6.1 General	15
6.2 Service provided.....	15
6.3 Interface to the link layer	15
6.4 Interface to the physical layer.....	16
6.5 MPDU format.....	17
6.6 Predictive p -persistent CSMA — overview description	17
6.7 Idle channel detection.....	18
6.8 Randomising	19
6.9 Backlog estimation.....	19
6.10 Optional priority.....	20
6.11 Optional collision detection	21
6.12 Beta1, Beta2 and Preamble Timings	21
7 Link layer.....	23
7.1 Assumptions	23
7.2 Service provided.....	24
7.3 CRC	24
7.4 Transmit algorithm	25
7.5 Receive Algorithm	25
8 Network layer	25
8.1 Assumptions	25
8.2 Service provided.....	27
8.3 Service interface.....	27
8.4 Internal structuring of the network layer	28
8.5 NPDU format	28
8.6 Address recognition.....	28
8.7 Routers	29
8.8 Routing algorithm.....	30
8.9 Learning algorithm — subnets	30
9 Transaction control sublayer	30
9.1 Assumptions	30
9.2 Service provided.....	31
9.3 Service interface.....	31
9.4 State variables	31
9.5 Transaction control algorithm	32
10 Transport layer	32
10.1 Assumptions	32
10.2 Service provided.....	32

10.3	Service interface.....	33
10.4	TPDU types and formats.....	33
10.5	Protocol diagram.....	35
10.6	Transport protocol state variables.....	35
10.7	Send algorithm.....	35
10.8	Receive algorithm.....	36
10.9	Receive transaction record pool size and configuration engineering.....	36
10.9.1	General.....	36
10.9.2	Number of retries.....	36
10.9.3	Transport layer timers.....	37
11	Session layer.....	38
11.1	Assumptions.....	38
11.2	Service Provided.....	38
11.3	Service interface.....	39
11.4	Internal structure of the session layer.....	39
11.5	SPDU types and formats.....	40
11.6	Protocol timing diagrams.....	41
11.7	Request-response state variables.....	43
11.8	Request-response protocol — client part.....	44
11.9	Request-response protocol — server part.....	44
11.10	Request-response protocol timers.....	44
11.11	Authentication protocol.....	45
11.12	Encryption algorithm.....	45
11.13	Retries and the role of the checksum function.....	45
11.14	Random Number Generation.....	46
11.15	Using Authentication.....	46
12	Presentation/application layer.....	46
12.1	Assumptions.....	46
12.2	Service provided.....	46
12.3	Service interface.....	47
12.4	APDU types and formats.....	48
12.5	Protocol diagrams.....	49
12.6	Application protocol state variables.....	50
12.7	Request - response messaging in offline state.....	50
12.8	Network variables.....	51
12.8.1	General.....	51
12.8.2	Network variable processing.....	51
12.9	Error notification to the application program.....	52
12.9.1	General.....	52
12.9.2	Error notification for messages.....	52
12.9.3	Error notification for network variables.....	52
13	Network management & diagnostics.....	52
13.1	Assumptions.....	52
13.2	Services provided.....	53
13.3	Network management and diagnostics application structure.....	53
13.4	Node states.....	53
13.5	Using the network management services.....	54
13.5.1	General.....	54
13.5.2	Addressing considerations.....	54
13.5.3	Making network configuration changes.....	55
13.5.4	Downloading an Application Program.....	55
13.5.5	Error handling conditions (informative).....	55
13.6	Using router network management commands.....	58
13.7	NMPDU formats and types.....	59
13.7.1	General.....	59
13.7.2	Query ID.....	59
13.7.3	Respond to query.....	60
13.7.4	Update domain.....	60

13.7.5	Leave domain.....	60
13.7.6	Update key	60
13.7.7	Update address.....	61
13.7.8	Query address	61
13.7.9	Query network variable configuration.....	61
13.7.10	Update group address	61
13.7.11	Query domain	61
13.7.12	Update network variable configuration.....	61
13.7.13	Set node mode.....	62
13.7.14	Read memory.....	62
13.7.15	Write memory.....	62
13.7.16	Checksum recalculate.....	62
13.7.17	Install	63
13.7.18	Memory refresh.....	77
13.7.19	Query SI.....	77
13.7.20	Network variable value fetch.....	78
13.7.21	Manual service request message	78
13.7.22	Network management escape code	78
13.7.23	Router mode	79
13.7.24	Router clear group or subnet table	79
13.7.25	Router group or subnet table download	79
13.7.26	Router group forward.....	79
13.7.27	Router subnet forward	79
13.7.28	Router Do Not forward group.....	79
13.7.29	Router Do Not forward subnet	79
13.7.30	Router group or subnet table report	79
13.7.31	Router status	80
13.7.32	Router half escape code	80
13.8	DPDU types and formats	80
13.8.1	General	80
13.8.2	Query status.....	80
13.8.3	Proxy status	84
13.8.4	Clear status	84
13.8.5	Query transceiver status	84
Annex A	(normative) Reference implementation.....	85
A.1	General	85
A.2	Predictive CSMA algorithm	85
A.3	LPDU transmit algorithm	140
A.4	LPDU receive algorithm	142
A.5	Routing algorithm.....	143
A.6	Learning algorithm	144
A.7	Transaction control algorithm	144
A.8	Network layer algorithm.....	151
A.9	TPDU and SPDU send algorithm with authentication	167
A.10	Application Layer	222
A.11	Network Management Commands.....	277
A.12	Configuration data structures.....	314
A.13	Include files for the reference implementation	333
A.14	Application protocol state variables and address recognition Structures	362
A.15	Query-id data structures.....	365
A.16	Respond to query data structure	365
A.17	Update somain data structures.....	366
A.18	Leave domain data structures	366
A.19	Update key data structures	366
A.20	Update address data structures	366
A.21	Query address data structures	367
A.22	Query NV Cnfg data structures.....	368
A.23	Update group address data structures	368
A.24	Query domain data structures	368

A.25	Update network variable configuration data structures.....	369
A.26	Set node mode data structures.....	369
A.27	Read memory data structures.....	369
A.28	Write memory data structures	370
A.29	Checksum recalculate data structures	370
A.30	Install command data structures	370
A.31	Memory refresh data structures	379
A.32	Query SI data structures.....	379
A.33	NV fetch data structures	379
A.34	Manual service request message ddata structures	379
A.35	Product query data structures	380
A.36	Router mode data structures	380
A.37	Router table clear group or subnet table data structures.....	380
A.38	Router group or subnet download data structures	380
A.39	Router group forward data structures	381
A.40	Router subnet forward data structures.....	381
A.41	Router group No-Forward data structures	381
A.42	Router subnet No-Forward data structures.....	381
A.43	Group / subnet table report data structures.....	382
A.44	Router status data structures	382
A.45	Query status data structures	382
A.46	Proxy status data structures.....	383
A.47	Clear status data structures.....	383
A.48	Query transceiver status data structures	383
Annex B	(normative) Additional Data Structures	384
B.1	General	384
B.1.1	System image	384
B.1.2	Application image.....	384
B.1.3	Network image	385
B.2	Read-only structures.....	385
B.2.1	Fixed read-only data structures.....	385
B.2.2	Read-only structure field descriptions.....	386
B.3	Domain table	389
B.3.1	Domain table field descriptions	390
B.4	Address table	390
B.4.1	Declaration of group address format	391
B.4.2	Group address field descriptions	391
B.4.3	Declaration of subnet/node address format.....	391
B.4.4	Subnet/node address field descriptions	392
B.4.5	Declaration of broadcast address format	392
B.4.6	Broadcast address field descriptions	392
B.4.7	Declaration of turnaround address format	392
B.4.8	Turnaround address field descriptions.....	393
B.4.9	Declaration of protocol processor's address format	393
B.4.10	Protocol processor address field descriptions.....	393
B.4.11	Timer field descriptions	393
B.5	Network variable tables - informative.....	394
B.5.1	Network variable configuration table field descriptions - informative	395
B.5.2	Network variable alias table field descriptions - informative	396
B.5.3	Network variable fixed table field descriptions - informative	396
B.6	Self-Identification structures.....	396
B.6.1	SI Structure field descriptions	397
B.6.2	NV descriptor table field descriptions.....	397
B.6.3	SNVT table extension records	398
B.6.4	SNVT alias field descriptions	399
B.6.5	Version 2 SI data.....	399
B.7	Configuration structure	402
B.7.1	General	402
B.7.2	Configuration structure field descriptions.....	403

B.8	Statistics relative structure	404
Annex C (informative) Behavioral characteristics		406
C.1	Channel capacity and throughput	406
C.2	Network metrics	407
C.3	Transaction metrics	408
C.4	Boundary conditions — power-up	409
C.5	Boundary conditions — high load	409
Annex D (normative) PDU summary		410
Annex E (normative) Naming and addressing		412
E.1	Address types and formats	412
E.2	Domains	412
E.3	Subnets and nodes	413
E.4	Groups	413
E.5	Unique_Node_ID and node address assignment	414
E.6	NPDU addressing	414
Annex F (normative) List of patents that pertain to this European Standard		417
Bibliography		419

Foreword

This document (FprEN 14908-1:2012) has been prepared by Technical Committee CEN/TC 247 “Building Automation, Controls and Building Management”, the secretariat of which is held by SNV.

This document is currently submitted to the Unique Acceptance Procedure.

This document will supersede EN 14908-1:2005.

This publication is copyright under the Berne Convention and the Universal Copyright Convention. No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form or by means, electronic, mechanical, photocopying, recording, or otherwise, without the permission of the European Committee for Standardization (CEN), the European Committee for Electrotechnical Standardization (CENELEC), their National Standards Bodies and their Licensees to reproduce this European Standard in full and including this copyright notice for the purposes of European standardisation.

This European Standard is part of a series of standards for open data transmission in building automation, control and in building management systems. The content of this European Standard covers the data communications used for management, automation/control and field functions.

The following is a list of technical changes since the previous edition:

- EN 14908-5 has been added to the normative references;
- the normative Annex A has been re-worked for a better understanding. The reference implementation of the standard shows in detail which part is normative and hardware independent, which one is normative but hardware dependent and which one is not normative because it is hardware dependent. This information supports the development of a protocol stack and the understanding of the specified communication services.

EN 14908-1 is part of a series of European Standards under the general title *Control Network Protocol (CNP)*, which comprises the following parts:

Part 1: *Protocol stack*;

Part 2: *Twisted pair communication*;

Part 3: *Power line channel specification*;

Part 4: *IP communication*;

Part 5: *Implementation*;

Part 6: *Application elements*.

Introduction

This European Standard has been prepared to provide mechanisms through which various vendors of building automation, control, and building management systems may exchange information in a standardised way. It defines communication capabilities.

This European Standard will be used by all involved in design, manufacture, engineering, installation and commissioning activities.

This European Standard has been made in response to the essential requirements of the Constructive Products Directive.

1 Scope

This European Standard applies to a communication protocol for networked control systems in commercial Building Automation, Controls and Building Management. The protocol provides peer-to-peer communication for networked control and is suitable for implementing both peer-to-peer and master-slave control strategies. This specification describes services in layers 2 to 7. In the layer 2 (data link layer) specification, it also describes the MAC sub-layer interface to the physical layer. The physical layer provides a choice of transmission media. The interface described in this specification supports multiple transmission media at the physical layer. In the layer 7 specification, it includes a description of the types of messages used by applications to exchange application and network management data.

2 Normative references

The following documents, in whole or in part, are normatively referenced in this document and are indispensable for its application. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

EN 14908-5, *Open Data Communication in Building Automation, Controls and Building Management Implementation Guideline — Control Network Protocol — Part 5: Implementation*

3 Terms and definitions

For the purposes of this document, the following terms and definitions apply.

For the purposes of this European Standard, the following subclause introduces the basic terminology employed throughout this European Standard. Most of it is commonly used and the terms have the same meaning in both the general and the standard context. However, for some terms, there are subtle differences. For example, in general, bridges do selective forwarding based on the layer 2 destination address. There are no layer 2 addresses in this standard protocol, so bridges forward all packets, as long as the domain address in the packet matches a domain of which the bridge is a member. Routers, in general, perform network address modification so that two protocols with the same transport layer but different network layers can be connected to form a single logical network. Routers of this standard may perform network address modification, but typically, they only examine the network address fields and selectively forward packets based on the network layer address fields.

3.1

channel

physical unit of bandwidth linking one or more communication nodes.

Note 1 to entry: Refer to Annex E for further explanation of the relationship between a channel and a subnet.

3.2

physical repeater

device that reconditions the incoming physical layer signal on one channel and retransmits it onto another channel

3.3

store-and-forward repeater

device that stores and then reproduces data packets onto a second channel

3.4

bridge

device that connects two channels (x and y); forwards all packets from x to y and vice versa, as long as the packets originate on one of the domain(s) that the bridge belongs to

3.5

configuration

non-volatile information used by the device to customise its operation. There is configuration data for the correct operation of the protocol in each device, and optionally, for application operation. The network configuration data stored in each device has a checksum associated with the data. Examples of network configuration data are node addresses, communication media parameters such as priority settings, etc. Application configuration information is application specific

3.6

domain

virtual network that is the network unit of management and administration. Group and subnet (see below) addresses are assigned by the administrator responsible for the domain, and they have meaning only in the context of that domain

3.7

flexible domain

used in conjunction with Unique_Node_ID and broadcast addressing. A node responds to a Unique_Node_ID-addressed message if the address matches, regardless of the domain on which the message was sent. To respond so that the sender receives it, the response shall be sent on the domain in which it was received. Furthermore, this domain shall be remembered for the duration of the transaction so that duplicate detection of any retries is possible. This transitory domain entry at a node is called the flexible domain. How many flexible domain entries a node supports depends on the implementation. However, a minimum of 1 is required

3.8

subnet

set of nodes accessible through the same link layer protocol; a routing abstraction for a channel; in this standard subnets are limited to a maximum of 127 nodes

3.9

node

abstraction for a physical node that represents the highest degree of address resolvability on a network. A node is identified (addressed) within a subnet by its (logical) node identifier. A physical node may belong to more than one subnet; when it does, it is assigned one (logical) node number for each subnet to which it belongs. A physical node may belong to at most two subnets; these subnets shall be in different domains. A node may also be identified (absolutely) within a network by its Unique_Node_ID

3.10

group

uniquely identifiable set of nodes within a domain. Within this set, individual members are identified by their member number. Groups facilitate one-to-many communication and are intended to support functional addressing

3.11

router

device that routes data packets to their respective destinations by selectively forwarding from subnet to subnet; a router always connects two (sets of) subnets; routers may modify network layer address fields. Routers may be set to one of four modes: repeater mode, bridge mode, learning mode, and configured mode. In repeater mode, packets are forwarded if they are received with no errors. In bridge mode, packets are forwarded if they are received with no errors and match a domain that the router is a member of. Routers in learning mode learn the topology by examining packet traffic, while routers that are set to configured mode have the network topology stored in their memory and make their routing decisions solely upon the contents of their configured tables

3.12

(application) gateway

interconnects networks at their highest protocol layers (often two different protocols). Two domains can also be connected through an application gateway

3.13**Beta1**

period immediately following the end of a packet cycle. A node attempting to transmit monitors the state of the channel, and if it detects no transmission during the Beta1 period, it determines the channel to be idle

3.14**Beta2**

randomising slot. A node wishing to transmit generates a random delay T . This delay is an integer number of randomising slots of duration Beta2

3.15**network variable**

variable in an application program whose value is automatically propagated over the network whenever a new value is assigned to it

3.16**Standard Network Variable Types (SNVTs)**

variables with agreed-upon semantics. These variables are interpreted by all applications in the same way, and are the basis for interoperability. Definition of specific SNVTs is beyond the scope of this European Standard

3.17**manual service request message**

network management message containing a node's Unique_Node_ID. Used by a network management device that receives this message to install and configure the node. May be generated by application or system code. May be triggered by external hardware event, e.g., driving a "manual service request" input low

3.18**transaction**

sequence of messages that are correlated together. For example, a request and the responses to the request are all part of a single transaction. A transaction succeeds when all the expected messages from every node involved in the transaction are received at least once. A transaction fails in this European Standard if any of the expected messages within the transaction are not received. Retries of messages within a transaction are used to increase the probability of success of a transaction in the presence of transient errors

4 Symbols and abbreviations

4.1 Symbols and graphical representations

Figure 1 shows the basic topology of networks based on this protocol and the symbolic representations used in this European Standard.

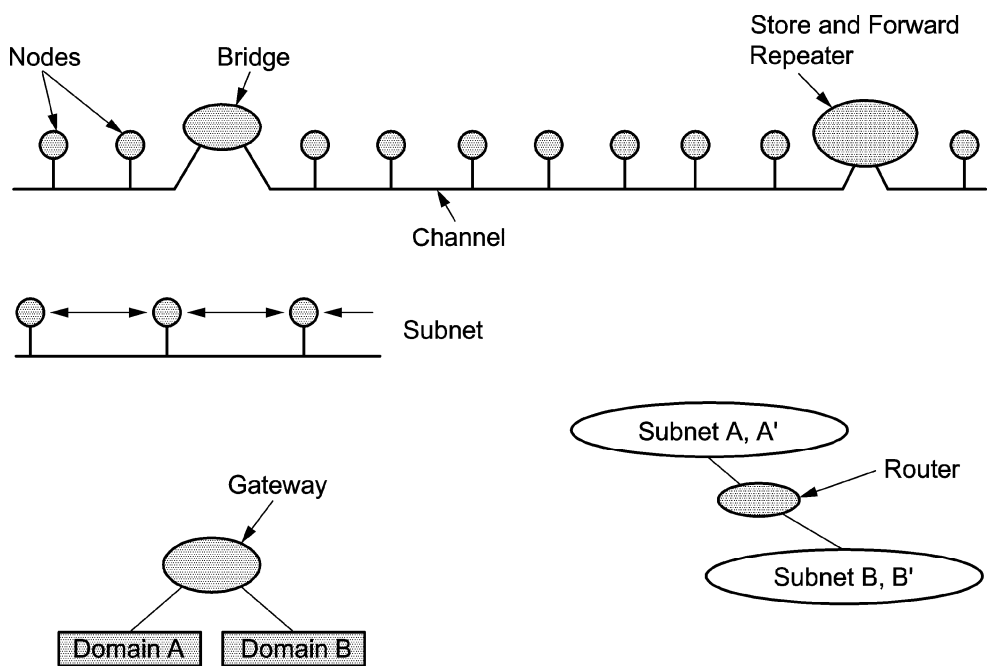


Figure 1 — Network topology & symbols

The layering of this protocol is described using standard OSI terminology, as shown in Figure 2.

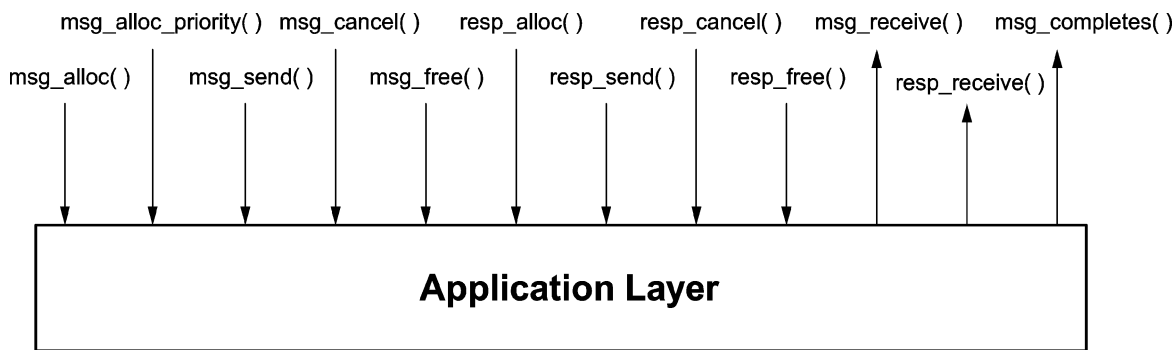


Figure 2 — Protocol terminology

4.2 Abbreviations

— CNP Control Network Protocol

The Protocol Data Unit (PDU) abbreviations used throughout this Standard are:

- PPDU Physical Protocol Data Unit, or frame
- MPDU MAC Protocol Data Unit, or frame
- LPDU Link Protocol Data Unit, or frame
- NPDU Network Protocol Data Unit, or packet
- TPDU Transport Protocol Data Unit, or a message/ack

- SPDU Session Protocol Data Unit, or request/response
- NMPDU Network Management Protocol Data Unit
- DPDU Diagnostic Protocol Data Unit
- APDU Application Protocol Data Unit
- FSM Finite State Machine (diagram)

Annex D (PDU Summary) contains the details of these PDUs.

5 Overview of protocol layering

The protocol specified by this Standard consists of the layers shown in Figure 3. Each layer is described below.

Multiple physical layer protocols and data encoding methods are allowed in systems based on this European Standard. Each encoding scheme is medium-dependent.

The *MAC* (Medium Access Control) sublayer employs a collision avoidance algorithm called Predictive *p*-persistent CSMA (Carrier Sense, Multiple Access). For a number of reasons, including simplicity and compatibility with the multicast protocol, the link *layer* supports a simple connectionless service. Its functions are limited to framing, frame encoding, and error detection, with no error recovery by re-transmission.

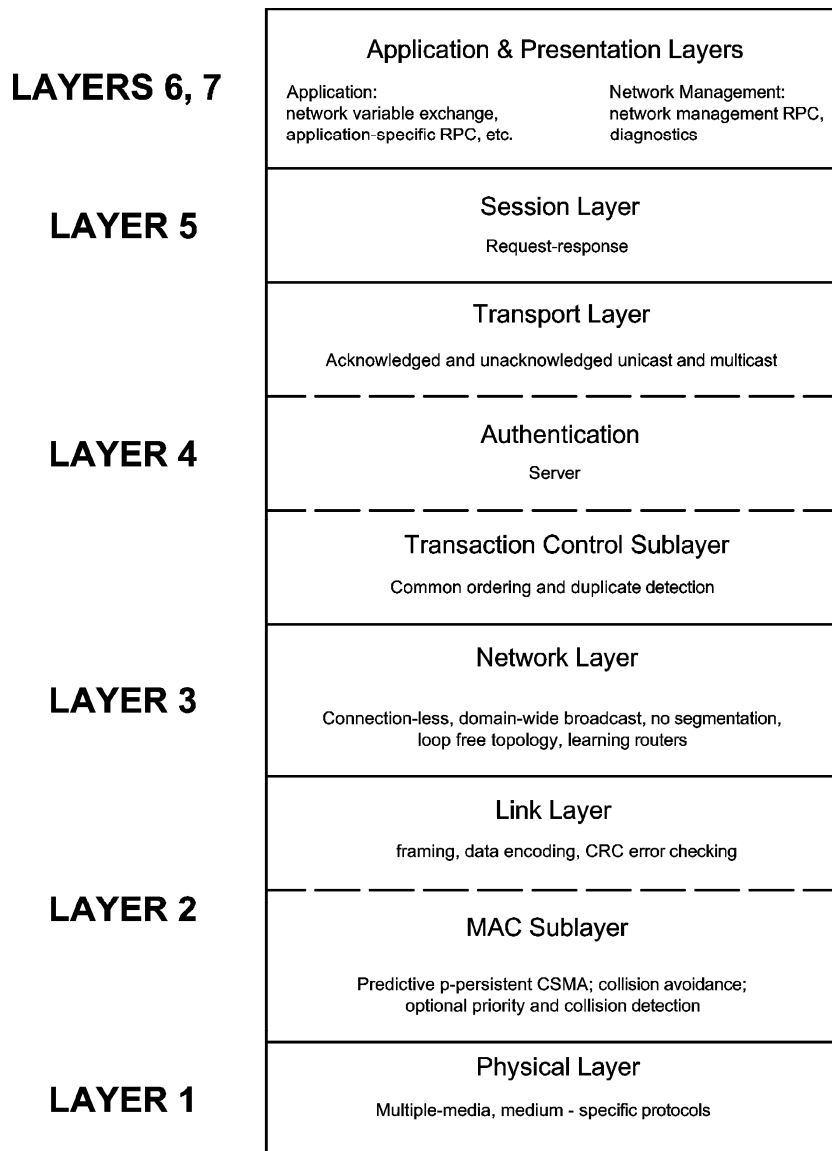


Figure 3 — Protocol layering

The *Network* layer handles packet delivery within a single domain, with no provisions for inter-domain communication. The Network service is connection-less, unacknowledged, and supports neither segmentation nor re-assembly of messages. The routing algorithms employed by the network layer to learn the topology assumes a tree-like network topology; routers with configured tables may operate on topologies with physical loops, as long as the communication paths are logically tree-like. In this topology, a packet may never appear more than once at the router on the side on which the packet originated. The unicast routing algorithm uses learning for minimal over-head and no additional routing traffic. Use of configured routing tables is supported for both unicast and group addresses, although in many applications a simple flooding of group addressed messages is sufficient.

The heart of the protocol hierarchy is the *Transport* and *Session* layers. A common *Transaction Control Sublayer* handles transaction ordering and duplicate detection for both. The *Transport* layer is connection-less and provides reliable message delivery to both single and multiple destinations. Authentication of the message sender's identity is included as a transport layer service, for use when the security of sender authentication is required. The authentication server requires only the Transaction Control Sublayer to accomplish its function. Thus Transport and Session layer messages may be authenticated using all of the addressing modes other than broadcast.

The session layer provides a simple Request-Response mechanism for access to remote servers. This mechanism provides a platform upon which application specific remote procedure calls can be built. The network management protocol, for example, depends upon the Request-Response mechanism in the Session layer.

A transport layer acknowledged message expects indication of message delivery from remote destination(s). A session layer request message expects indication that application-specific remote task(s) have been completed. A given message uses only one or the other type of service, but not both.

This specification includes the *Presentation Layer* and the lowest level of the *Application Layer*. These layers provide services for sending and receiving application messages including network variables, and other types of messages such as network management and diagnostic messages and foreign frames (see Clause 13). For a network variable update, the APDU header provides information on how to interpret the APDU. This application-independent interpretation of the data allows data to be shared among nodes without prior arrangement.

6 MAC sublayer

6.1 General

In this European Standard the following Media Access Control sublayer is defined. If there is a need for other MAC sublayers they are defined in additional parts of this European Standard.

6.2 Service provided

The Media Access Control (MAC) sublayer facilitates media access with optional priority and optional collision detection/collision resolution. It uses a protocol called Predictive *p*-persistent CSMA (Carrier Sense, Multiple Access), that has some resemblance to the *p*-persistent CSMA protocol family.

Predictive *p*-persistent CSMA is a *collision avoidance* technique that randomises channel access using knowledge of the expected channel load. A node wishing to transmit always accesses the channel with a random delay in the range (0..w). To avoid throughput degradation under high load, the size of the randomising window, w, is a function of estimated channel backlog BL:

$$w = (BL \times W_{\text{base}}) - 1, \quad (1)$$

where

W_{base} is the base window size. W_{base} is measured in time. Its duration, derived from Beta2 (see 6.8), equals 16 Beta2 slots.

6.3 Interface to the link layer

The MAC sublayer is closely coupled to the Link layer, described in Clause 7. With the MAC sublayer being responsible for media access, the Link layer deals with all other layer 2 issues, including framing and error detection. For explanatory purposes, the interface between the two layers is described in the form shown in Figure 4.

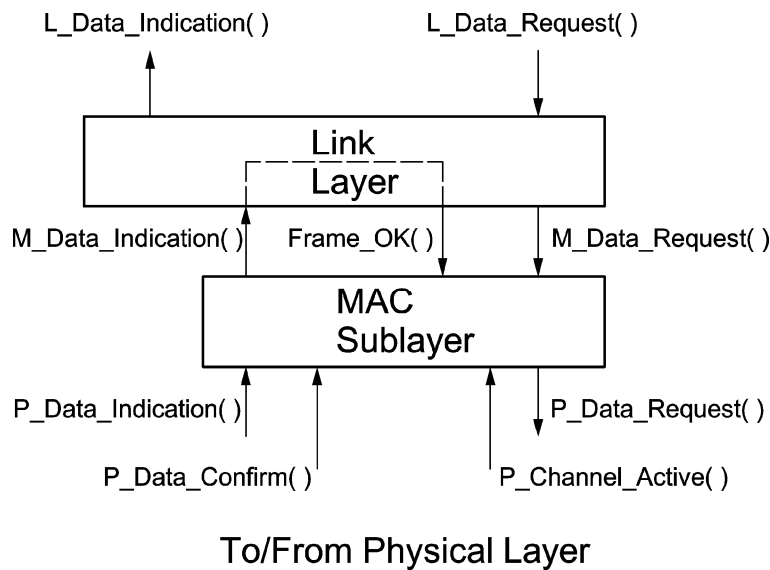


Figure 4 — Interface between the MAC and the link layers

Although the service interface primitives are defined using a syntax similar to programming language procedure calls, no implementation technique is implied. Frame reception is handled entirely by the Link layer, that notifies the MAC sublayer about the backlog increment via the `Frame_OK()` primitive.

The following service interface primitives facilitate the interface between the Link and the MAC layers:

`M_Data_Request (Priority, delta_BL, ALT_Path, LPDU)`

This primitive is used by the Link layer to pass an outbound LPDU/MPDU to the MAC sublayer. Priority defines the priority with which the frame is to be transmitted; delta_BL is the backlog increment expected as a result of delivering this MPDU. ALT_Path is a binary flag indicating whether the LPDU is to be transmitted on the primary or alternate channel, baud rate, etc. See 6.5 for how ALT_Path is set.

`Frame_OK (delta_BL)`

On receiving a frame and verifying that its CRC is correct, the Link layer invokes this primitive to notify the MAC sublayer about the backlog increment associated with the frame just received.

`M_Data_Indication()`

The MAC sublayer provides this indication to the link layer once per incoming LPDU/MPDU.

6.4 Interface to the physical layer

The physical layer handles the actual transmission and reception of binary data. Multiple physical layer protocols are supported by the control network protocol. The bit error rate presented to the link layer shall be equal to or better than $1 \text{ in } 10^{-4}$. For compatibility with the higher layers, all physical protocols shall support the defined service interface (see figure 4):

`P_Data_Indication (Frame)`

Physical layer provides this indication to the MAC sublayer and the link layer once per in-coming LPDU/MPDU.

`P_Data_Request (Frame)`

The MAC sublayer uses this primitive to pass the Frame, the encoded LPDU/MPDU, to the physical layer for immediate transmission. The bit transmission order is defined in Annex D.

P_Data_Confirm (Status)

The physical layer returns Status as to whether the frame was transmitted. Status has three possible values: success—indicating the frame was transmitted, request_denied—indicating that activity was detected on the line prior to transmission, and collision—indicating that transmission began, but a collision was detected. Whether or not the transmission is aborted depends on when the collision is detected (see 6.11).

P_Channel_Active ()

The physical layer uses this primitive to pass the status of the channel to the MAC sublayer. This is an indication of activity, not necessarily of valid data.

6.5 MPDU format

The combined MPDU/LPDU format is shown in Figure 5. (Annex D contains the details of the NPDU frame).

LPDU/MPDU

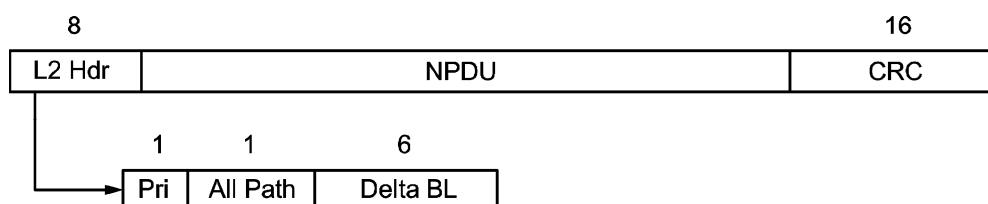


Figure 5 — MPDU/LPDU format

The MAC sublayer uses the L2Hdr field, that has the following syntax and semantics:

Pri 1-bit field specifying the priority of this MPDU: 0 = Normal, 1 = High.

Alt_Path 1-bit field specifying the channel to use. This is a provision for transceivers that have the ability to transmit on two different channels and receive on either one without the need to instruct the transceiver to explicitly receive on a specific channel. The transport layer sets this bit for the last two attempts (for acknowledged and request/response services), unless requested to specify the alternate path for every transmission. For any packet received that has the alt_path bit set and that requires an acknowledgement, response, challenge, or reply, the alt_path bit shall be set in the corresponding acknowledgement, response, challenge, or reply.

Delta_BL 6-bit unsigned field (≥ 0); specifies channel backlog increment to be generated as a result of deliver-ing this MPDU.

6.6 Predictive *p*-persistent CSMA — overview description

Like CSMA, Predictive *p*-persistent CSMA senses the medium before transmitting. A node attempting to transmit monitors the state of the channel (see Figure 6), and determines the channel to be idle if it detects no transmission during the Beta1 period. Nodes without a packet to transmit during this Beta1 period shall remain in synchronisation for the duration of the priority slots (see 6.11), and at least W_{base} randomising slots. This maintenance of synchronisation allows a packet that arrives in the output queue of the MAC sublayer after the end of the Beta1 time to be transmitted in a valid slot according to the other nodes with a packet to transmit.

Next, the node generates a random delay T (transmit) from the interval $(0..(BL \times W_{base})-1)$, where W_{base} is the number of randomising slots within the basic randomising window and BL is an estimate of the current channel backlog. T (transmit) is defined as an integer number of randomising slots of duration $Beta2$ (see 6.8 and 6.9). If the channel is idle when the delay expires, the node transmits; otherwise, the node receives the incoming packet, and then repeats the MAC algorithm. In Figure 6, D_{mean} is the average randomisation delay between packets, and, since the random delay T is uniformly distributed, D_{mean} is given as $(W_{base}-1)/2$ for small values of BL .

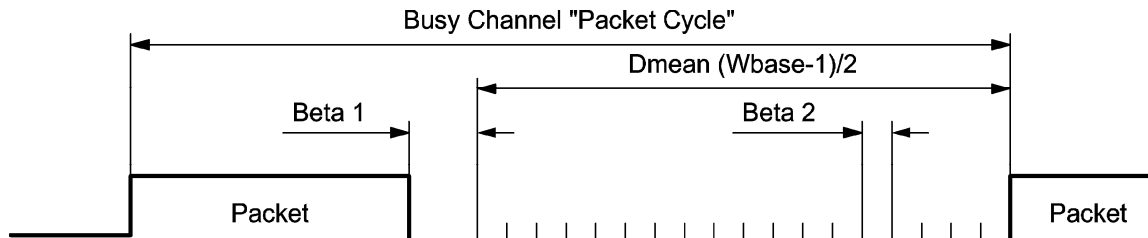


Figure 6 — Predictive p -persistent CSMA concepts and parameters

By adjusting the size of the randomising window as a function of the predicted load, the algorithm keeps the collision rate constant and independent of the load. Provided that the estimated backlog is greater than or equal to the real back-log, the following holds:

$$\text{Collision Rate} = \text{Error Pkt Cycles} / \text{Error Free Pkt Cycles} \leq 1 / 2W_{base}$$

A base window size of 16 is used. This implies that there are an average of 8 randomising slots of width $Beta2$ and one slot of width $Beta1$ between each packet. In addition, the width of the $Beta2$ period is crucial to efficient utilisation of the channel.

The algorithm for Predictive CSMA is in A.2.

6.7 Idle channel detection

The idle channel condition is asserted whenever the following two conditions are met:

- 1) The current channel state reported by the physical layer via the `P_Data_Indication ()` primitive is low; and
- 2) No transition has been detected during the last period of $Beta1$. Note that the MAC sublayer may be configured to ignore transitions during a portion of the $Beta1$ period. This portion of time that transitions are ignored (the channel is assumed to be idle during this time even in the presence of transitions) is called the indeterminate time (see 6.12 for the details).

The length of the $Beta1$ period is defined by the following constraint:

$$Beta1 > 1 \text{ bit time} + (2 \times Tau_p + Tau_m) \quad (2)$$

The first term assumes a data encoding method that guarantees a transition and/or carrier during every bit time. If encoding methods are used that do not meet this constraint, then the first term shall be adjusted to be the longest time that the channel may appear idle without being idle, i.e., the longest run in legal data transmission without a transition and/or carrier asserted on the medium. The second term takes care of propagation and turnaround delays:

Tau_p is the physical propagation delay defined by the media length;

Tau_m is the detection and turnaround delay within the MAC sublayer; this is the period from the time the idle channel condition is detected, to the point when the first output transition appears on the output. On media where there is a carrier, this time shall include the time between turning on the carrier, and it being asserted as a valid carrier on the medium.

6.8 Randomising

At the beginning of the randomising period, a node wishing to transmit generates a random delay T (transmit) from the interval $(0.. (\text{BL} \times W_{\text{base}})-1)$. The node then waits for this period, while continuing to monitor channel status; if the channel is still idle when the delay expires, the node transmits.

The transmit delay T (transmit) is an integer number of randomising slots of duration $\text{Beta}2$; the length of the randomising slot shall meet the following constraint:

$$\text{Beta}2 > 2 \times \text{Tau}_p + \text{Tau}_m \quad (3)$$

Parameters Tau_p and Tau_m are defined in 6.7.

6.9 Backlog estimation

The predictive aspect of the MAC algorithm is based on backlog estimation. Each node maintains an estimate of the current channel backlog BL , that is incremented as a result of sending or receiving an MPDU and decrements periodically — once every packet cycle. The increment to the backlog is encoded into the link layer header, and represents the number of messages that the packet shall cause to be generated upon reception. The backlog is initially set to one. After sending or receiving a packet with a non-zero backlog increment, the node's backlog estimation is incremented by the backlog increment. The maximum backlog value is 63. If the backlog exceeds 63, then the backlog overflow statistic is incremented (see 13.7.14 and B.8).

The backlog decrements under one of the following conditions:

- On waiting to transmit: If W_{base} randomising slots go by without channel activity.
- On receive: If a packet is received with a backlog increment of '0'.
- On transmit: If a packet is transmitted with a backlog increment of '0'.
- On idle: If a packet cycle time expires without channel activity.

The packet cycle timer is reset to its initial value whenever the backlog is changed. It is started (begins counting down at its current value) whenever the MAC layer becomes idle. An idle MAC layer is defined as:

- 1) not receiving;
- 2) not transmitting;
- 3) not waiting to transmit;
- 4) not timing $\text{Beta}1$;
- 5) not waiting for priority slots; and
- 6) not waiting for the first W_{base} randomising window to complete.

On transition from idle to either transmit or receive, the packet cycle timer is halted.

The backlog always has a value ≥ 1 . The algorithm post-increments rather than pre-increments the backlog by the amount associated with the MPDU being transmitted, because the number of expected responses is of no importance until after transmitting the MPDU.

6.10 Optional priority

On a channel by channel basis, the protocol supports optional priority. Priority slots, if any, follow immediately after the Beta1 period that follows the transmission of a packet (Figure 7). The number of priority slots per channel ranges from 0 to 127. Priority slots are typically not contended for, but rather are uniquely assigned to nodes on the channel. Nodes that have been assigned a priority slot do not have to use it with every message; the node decides on a message by message basis whether or not to use the assigned priority slot. This determination is made by examining the priority bit within the LPDU header (Figure 5).

It is possible to assign all the nodes on the channel the same priority slot. An example of an architecture where this makes sense would be one where there is a background of peer-to-peer activity, but a single master that cycles around doing something to each node (such as network management, polling, etc.). By giving each node the same slot, and having it used only for this purpose, these transactions (from the single master to multiple slaves) would tend to be completed ahead of the background traffic.

An application may decide that a message is high priority and attempt to send it as such. If the node does not have a priority slot assigned to it, the message shall go out in the usual way, except that the priority bit in the layer 2 header shall be set. If, subsequently, the packet passes through a router that has a priority slot on its destination channel, the packet shall be sent using the priority of the router. If the router does not have a priority slot, the message shall be for-warded in the usual way, with the priority bit in the layer 2 header remaining set.

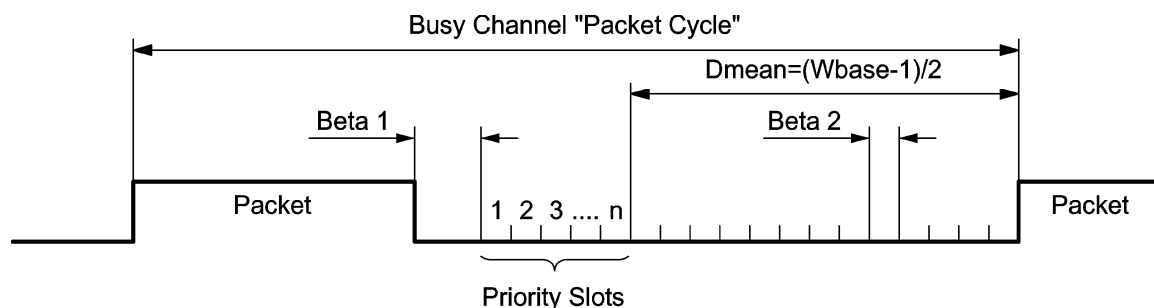


Figure 7 — Allocation of priority slots within the Busy Channel Packet Cycle

The protocol provides no synchronisation among the nodes. Therefore, if the channel has been idle for longer than the randomising period ($\text{Beta1} + \text{number of priority slots} + D_{\text{mean}}$ above), access to the link is random without regard to priority. Once the link returns to the busy state, access to the link shall be in priority order.

If a priority message requires:

- acknowledgement (acknowledged messaging);
- response (request/response messaging);
- challenge (in response to an authenticated acknowledged message or authenticated request); or
- reply (in response to an authentication challenge);

the responding node shall attempt to send a priority acknowledgement/response/challenge/reply by setting the priority bit in the layer 2 header. If a high priority message is generated within a node, it is sent prior to any queued packets of normal priority. Multiple high priority packets are sent in FIFO

order. If the application attempts to send a high priority message while its node is sending a packet, the packet in progress completes first.

If a channel has priority slots and a node has multiple messages queued within it, it shall not send them in consecutive packet cycles. In the case where a node has a packet to send, and it has sent a packet in the previous packet cycle, the node does not use its normal medium access algorithm in the current cycle. Instead it attempts to access the medium using a slightly modified version of the non-priority MAC algorithm. The modified version of the algorithm is that the transmitter node waits out the Beta1 time after the priority packet it just transmitted, then it waits for all the priority slots to go by, then it waits for one window of randomising slots to go by (16 Beta2 slots). At this point it randomises its access to the network using the current channel backlog estimate. If the node is not successful in the current packet cycle, it may use its priority slot in the subsequent packet cycle.

6.11 Optional collision detection

The physical layer may optionally notify the MAC sublayer of a detected collision. The MAC sublayer can be configured to check for collision detection notification any time during packet transmission.

Collision resolution may be implemented by means of collision detection. A bit pattern unique to each node on the channel may be inserted before the preamble field to resolve collisions. Such a physical layer shall define a sufficiently long preamble after the arbitration pattern so that transmitting nodes that lose the arbitration for a packet cycle can turn around and receive the incoming packet from the node that just won the arbitration.

Collision detection, if used, shall only be enabled upon transmission of a packet. When a collision is detected in the MAC sublayer, the backlog is incremented. Receivers do not detect collisions, rather they see them as packets that are too short, or have incorrect CRCs or sometimes they are seen as just spurious transitions for which bit synchronisation cannot be achieved. When a collision is detected, the backlog is incremented, and the collision statistic is incremented (see 13.7.14 and B.8).

The MAC sublayer attempts to re-transmit the packet upon notification of a collision. It obeys the following rules:

- 1) If a collision is detected on two successive attempts to transmit a priority packet in the node's priority slot, the next attempt to transmit the priority packet shall not use the configured priority slot, but rather shall be in a slot picked according to the non-priority MAC algorithm. The priority bit in the layer 2 header of the packet shall remain set in this case.
- 2) Whenever a collision is detected by a transmitting node, that transmitting node increments its estimate of the channel backlog by 1.
- 3) Whenever a collision is detected on 256 successive attempts to transmit a packet, the packet is discarded.

6.12 Beta1, Beta2 and Preamble Timings

A node may be implemented to support one or more media. If implemented to support multiple media, its preamble length and Beta1 and Beta2 times shall be configurable to support the relevant physical communication media. Beta2 is the duration in time of a media access time-slot (either randomising or priority slot). The MAC layer timers for Beta1 are provided for both the transmitter in the previous packet cycle as well as the receiver. This is to correct for differences in the time base for the end of the preceding packet between transmitters and receivers. Note that in this family of physical layer standards the timings are with reference to the timings observable on the medium itself, while these timings are in reference to the protocol processor's reference. For example, imagine a transceiver that added bits of error correction code to the actual packet. From the transmitting protocol processor's reference the end of the packet would be sent before these bits were added to the packet, while the receiving protocol processor's time reference would show packet end at the end of the packet including the error correction bits. Thus, the need for the ability to specify a difference in the timing depending upon whether a node is a transmitter or a receiver for a given packet. In addition there are

two communication configurations (comm_type in B.7) with different formula for calculating Beta1 based on the type.

A node shall be able to generate one of five base time periods to generate Beta1 and Beta2 slots of the right duration.

The formulae for Beta1, Beta2, and the preamble are shown below.

CT is a time base used to generate Beta1 and Beta2 slots and the preamble. Depending on the media type(s) supported, each node shall support one or more of the following values for CT: 600 ns, 1,2 µs, 2,4 µs, 4,8 µs, 9,6 µs.

v, a tuning parameter, is in the range 0 to 255 inclusive. There are 3 tuning parameters. These are represented as v₁, v₂ and v₃ in the formula below.

f(x) is a function that returns a time delay to compensate for other slower nodes on a channel that may generate actual values of CT below the nominal for a channel.

$$\text{Beta2} = \text{CT} \times (40 + 20 \times v_1) \quad (4)$$

There are four formulae for computing Beta1. The formula to use depends on a bit field in the configuration structure, comm_type, listed in B.7, and whether the timing is done immediately following the receipt or the transmission of a packet. Comm_type can have one of two values: 1 or 2. The selection of the comm_type determines the behaviour of the MAC sublayer in the following ways:

Comm_type = 1:

- 1) An indeterminate time is defined during the Beta 1 period in which all transitions on the channel are ignored. This period starts following the end of any packet (transmitted or received). Its duration is defined below. Following the indeterminate time and before the first Beta2 slot, the MAC sublayer repeatedly waits for a period of duration of one Beta2 slot to pass with no transitions on the channel before proceeding (note that if a valid preamble is detected during this period, an attempt shall be made to receive the incoming packet). This Beta2 slot duration accounts for the Beta2 slot duration component of the total Beta1 time.
- 2) The transmitted length of the preamble is under the control of the MAC sublayer.
- 3) The MAC sublayer ignores collisions occurring during the first 25 % of the transmitted preamble. It optionally (according to the cd_tail field described in B.7) ignores collisions reported following the transmission of the CRC but prior to the end of transmission.
- 4) If a collision is detected during preamble transmission, the MAC sublayer can terminate the packet if so configured. Collisions detected after the preamble has been sent do not terminate transmission.

Comm_type = 2:

- 1) No indeterminate time is defined at the MAC sublayer. If there is an indeterminate time, it shall be enforced by the physical layer.
- 2) The MAC sublayer is configured to either control the transmitted length of the preamble or to have the transmitted length of the preamble controlled by the physical layer.
- 3) The MAC sublayer shall always terminate the packet upon notification of a collision.
- 4) The MAC sublayer reconfigures the physical layer prior to transmission, if necessary, to inform it of a change in transmission path selection (alternate path versus no alternate path).

The formulae for Beta1 for the four cases of transmitting, receiving, comm_type = 1 and comm_type = 2 are as follows:

$$\begin{aligned} \text{Beta1(after transmission)} &= \text{CT} \times (583 + f(\text{xmit_interpacket})) + \text{Beta2} \{ \text{for comm_type} = 1 \} \\ \text{Beta1(after reception)} &= \text{CT} \times (565 + f(\text{recv_interpacket})) + \text{Beta2} \{ \text{for comm_type} = 1 \} \\ \text{Beta1(after transmission)} &= \text{CT} \times (624 + f(\text{xmit_interpacket})) + \text{Beta2} \{ \text{for comm_type} = 2 \} \\ \text{Beta1(after reception)} &= \text{CT} \times 602 + f(\text{recv_interpacket}) + \text{Beta2} \{ \text{for comm_type} = 2 \} \end{aligned}$$

(5 to 8)

$$f(v) = (v < 128) ? (41 \times v_2) : 145 \times (v_2 - 128) \quad (9)$$

Parameters xmit_interpacket and recv_interpacket have ranges from 0 to 255 and are parameters stored in the Configuration Structure (B.7).

The two formulae for computing the indeterminate time for the MAC sublayer are as follows:

$$\begin{aligned} \text{IDT(after transmission)} &= \text{CT} * (313 + f(\text{xmit_interpacket})) \{ \text{for comm_type} = 1 \} \\ \text{IDT(after reception)} &= \text{CT} * (295 + f(\text{recv_interpacket})) \{ \text{for comm_type} = 1 \} \end{aligned}$$

(10 to 11)

CT, f(x), xmit_interpacket and recv_interpacket are as defined for the Beta1 calculations. Note that in all cases the indeterminate time is constrained to be less than the total Beta1 time.

Preamble length is either controlled by the protocol processor or by the transceiver. When the protocol processor controls the preamble, the formula for all possible values of preamble length is shown below. For this formula the variable v has a range of from 0 to 253 inclusive. CT is as defined above.

$$\text{Preamble} = \text{CT} \times (219 + 32 \times v_3) \quad (12)$$

When the transceiver controls the preamble length (an option available when comm_type = 2), the minimum preamble length allowed is 181 µs. This is the value corresponding to a protocol processor with a 600 ns timer value. This value scales with the time base, CT. When the transceiver controls the preamble length, the maximum preamble length is determined by the transceiver.

Beta1, Beta2 and preamble timings for each transceiver type compatible to this standard are noted in the relevant physical layer specification.

7 Link layer

7.1 Assumptions

The Link layer assumes that CRC errors due to both collisions and transmission errors occur with some probability P_e , and that P_e is small enough so that link level error recovery is not needed.

The above assumption means that successful end-to-end communication is only possible when the sum of error probabilities along the communication path is much less than one.

$$\text{SUM}(P_e) \ll 1$$

In networks where P_e is constant, the maximum communication distance D (network or group diameter) shall be:

$$D \ll 1/P_e$$

7.2 Service provided

The Link layer provides subnet-wide, ordered, unacknowledged LPDU delivery with error detection but no error recovery. A corrupted frame is discarded as soon as its CRC check fails.

7.3 CRC

The CRC is computed over the entire NPDU including the L2Hdr field. The CRC is generated using the polynomial:

$$X^{16} + X^{12} + X^5 + 1 \text{ (the CCITT CRC-16 standard).}$$

For each NPDU, the CRC register is initialised to all ones. Figure 8 shows an example of the time behaviour of the CRC shift register that implements the above polynomial. The register bits are represented by the columns labelled bit 0 to bit 15. The data, over which the CRC is computed, appears in the column labelled `crc_in`. The register is initialised to all ones. The data is then presented serially to the register starting with bit7 of byte 1. Each row of the table shows the register state at a particular bit time. The register continues to input data and compute the CRC as long as the compute mode is true (`mode compute = 1`). The compute mode going false signals the end of data and the end of the CRC computation.

Following data input, the sixteen (16) bits of CRC data are shifted out serially as shown by the column labelled `nrz_out`.

When the link layer receives a packet with a CRC error or a packet that is less than 8 bytes in length, a transmission error statistic is incremented (see 13.8.2), and the packet cycle timer is started but not reset first, unlike after the reception of a good packet. If a packet is received but is too long for the input buffer, or there are no input buffers, the missed packet statistic is incremented (see 13.8.2).

For a normative description of the CRC-16 computation, refer to the pseudocode example in A.3.

comment	mode	compute	crc_in	bit	bit	bit	bit	bit	bit	bit	bit	bit	bit	bit	bit	bit	bit	bit	nrz_out
initialize				1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
byte1 bit7	1		0	1	1	1	1	1	0	1	1	1	1	1	1	0	1	1	0
6	1		1	0	1	1	1	1	1	0	1	1	1	1	1	1	0	1	1
5	1		1	0	0	1	1	1	1	1	0	1	1	1	1	1	1	0	1
4	1		1	0	0	0	1	1	1	1	1	1	0	1	1	1	1	1	0
3	1		1	1	0	0	0	1	0	1	1	1	0	1	1	0	1	1	1
2	1		0	1	1	0	0	0	0	0	1	1	1	0	1	0	0	1	0
1	1		0	1	1	1	0	0	1	0	0	1	1	1	0	0	0	1	0
0	1		1	0	1	1	1	0	0	1	0	0	1	1	1	0	0	0	1
byte2 bit7	1		1	1	0	1	1	1	1	0	1	0	0	1	1	0	0	0	1
6	1		0	0	1	0	1	1	1	1	0	1	0	0	1	1	0	0	0
5	1		0	0	0	1	0	1	1	1	1	0	1	0	0	1	1	0	0
4	1		1	1	0	0	1	0	0	1	1		0	1	0	1	1	1	0
3	1		1	1	1	0	0	1	1	0	1	1	1	0	1	1	1	1	1
2	1		0	1	1	1	0	0	1	0	1	1	1	0	1	0	1	1	0
1	1		0	1	1	1	1	0	1	0	1	0	1	1	1	1	0	1	0
0	1		0	1	1	1	1	1	1	1	0	1	0	1	1	0	1	0	0
byte3 bit7	1		1	0	1	1	1	1	1	1	1	0	1	0	1	1	0	1	0
6	1		1	1	0	1	1	1	1	0	1	1	0	1	0	1	0	1	1
5	1		1	0	1	0	1	1	1	0	1	1	1	0	1	0	0	1	0
4	1		0	0	0	1	0	1	1	1	0	1	1	1	0	1	0	0	0
3	1		0	1	0	0	1	0	0	1	1	0	1	1	1	1	0	0	0
2	1		0	0	1	0	0	1	0	0	1	1	0	1	1	1	1	0	0
1	1		0	0	0	1	0	0	1	0	0	1	1	0	1	1	1	1	0
0	1		0	1	0	0	1	0	1	1	0	0	1	1	0	0	1	1	0
crc bit15	0			0	1	0	0	1	0	1	1	0	0	1	1	0	0	1	0
14	0			0	0	1	0	0	1	0	1	1	0	0	1	1	0	0	0
13	0			0	0	0	1	0	0	1	0	1	1	0	0	1	1	0	0
12	0			0	0	0	0	1	0	0	1	0	1	1	0	0	1	1	0
11	0			0	0	0	0	0	1	0	0	1	0	1	1	0	0	1	1
10	0			0	0	0	0	0	0	1	0	0	1	0	1	1	0	0	1
9	0			0	0	0	0	0	0	0	0	1	0	0	1	0	1	1	0
8	0			0	0	0	0	0	0	0	0	0	1	0	0	1	0	1	0
7	0			0	0	0	0	0	0	0	0	0	0	1	0	0	1	0	1
6	0			0	0	0	0	0	0	0	0	0	0	1	0	0	1	0	0
5	0			0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	0
4	0			0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1
3	0			0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
2	0			0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
1	0			0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
0	0			0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 8 — CRC register state behaviour example

7.4 Transmit algorithm

The pseudocode Transmit algorithm is in A.3.

7.5 Receive Algorithm

A valid frame starts with the channel active state, and terminates with the channel idle state. Upon receipt, valid frames are processed as defined below; invalid frames are discarded. The pseudocode Receive algorithm is in A.4.

8 Network layer

8.1 Assumptions

This protocol supports a variety of topologies in order that the requirements from many application areas can be met. Within a single channel, the topology can be a bus, a ring, a star, or “free” (see Figure 9). Free topology is defined as a total wire specification with no other rules, and a single termination placed anywhere on the network. Thus, the set of all free topologies includes a ring, a star, a bus and virtually any other combination of these constructs.

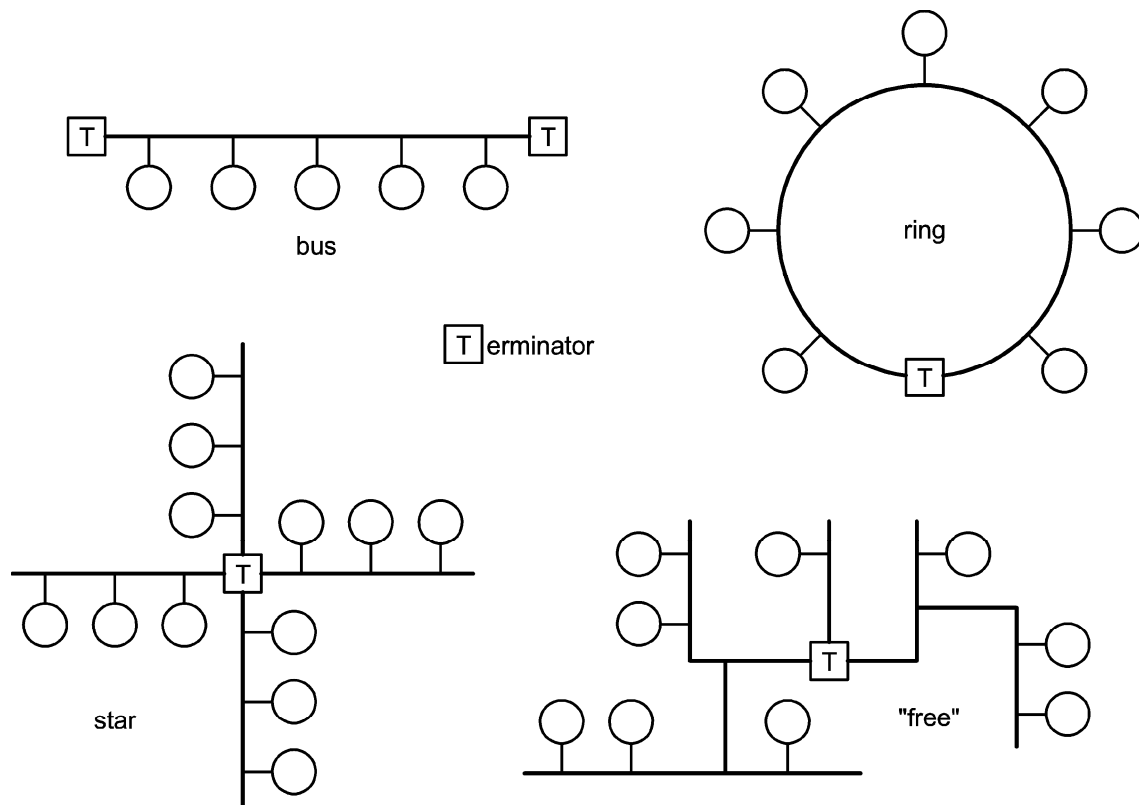


Figure 9 — Single channel topologies

The protocol supports physical layer repeaters as well as store-and-forward repeaters to repeat packets from one channel to another. The protocol also supports bridges to repeat all packets on the bridge's domain(s) from one channel to another. Additionally, both learning and configured routers are supported to segment traffic and thus increase total system performance.

In networks where there is a possibility of more than one path from one node to another, there is a danger of packets looping indefinitely. In these networks, configured routers shall be used to impose a logical tree structured topology on top of the physical looping topology.

If there is a desire to use repeaters, bridges, and learning routers, then control of the topology shall be maintained so that no loops exist. To avoid routing loops within a domain, domain topology shall be tree-structured as shown in Figure 10. Store-and-forward repeaters may only be used if they connect two different channels together; store-and-forward repeaters that repeat on the same channel are not supported. This restriction is necessary for an order-preserving network.

If any node receives a packet with a protocol version number other than a version number it supports, the packet is discarded.

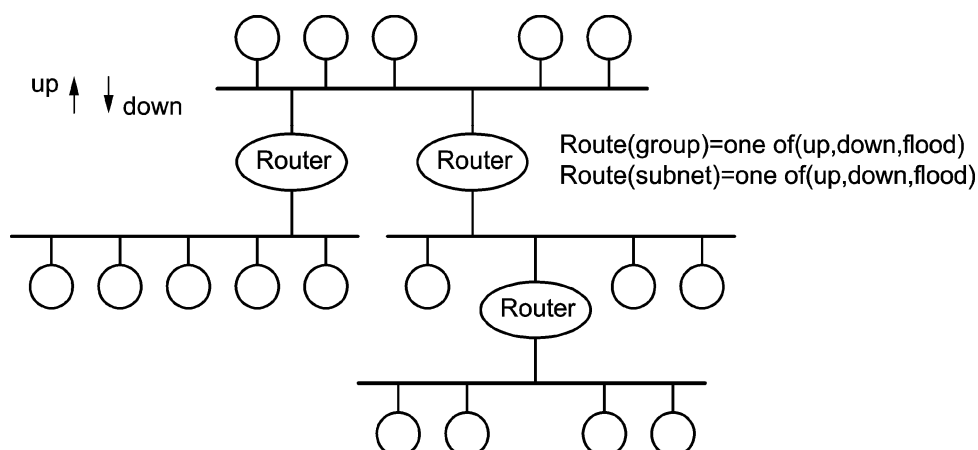


Figure 10 — Typical tree-like domain topology

8.2 Service provided

This network layer provides a connection-less network service facilitating domain wide packet delivery with the following attributes:

- *Unacknowledged Unicast, Multicast, and Broadcast.* Depending on its destination address, the packet submitted is delivered to one node, multiple nodes, or all nodes within the domain (or optionally all nodes on a specified subnet within the domain). This delivery occurs with some probability $p \leq 1$;
- *Lossiness.* The network layer supports no re-transmissions or acknowledgments. The probability of delivery decreases with the number of channels traversed;
- *Order Preserving.* Loop-free topology (accomplished logically with configured routers or physically via topological control) coupled with the absence of single channel store and forward repeaters provides natural ordering;
- *No Segmentation.* No message segmentation and/or message re-assembly are performed anywhere within the network layer.

When learning routers are used, the routers discover the topology by examining the network layer address fields in the packets. The learning algorithm imposes no additional traffic overhead on the network. It assumes that the domain is loop free, and it learns about the location of subnets by observing the source addresses of NPDUs being routed. NPDUs addressed to groups are routed by flooding, with the NPDU being propagated through the entire domain.

The network layer suppresses outgoing messages if the node is not in the hard offline state (see 13.4) or configured state (e.g. unconfigured), except for manual service request messages, responses, and ACKs. These excepted messages can be sent in any state.

8.3 Service interface

The Network service interface consists of the Send_Packet () service request and the Rcv_Packet () indication, as shown in Figure 11. Again this interface is provided for explanation purposes. Actual implementations may, for example, combine this layer with the transport layer and only expose the transport layer interface.

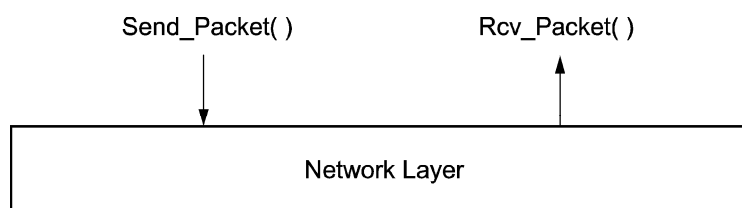


Figure 11 — Network service interface

The syntax of these two interface primitives is:

`Send_Packet (address_pair, pduType, PDU, priority, delta_BL, Alt_path)`

`Rcv_Packet (address_pair, pduType, PDU, priority)`

8.4 Internal structuring of the network layer

The network layer performs two functions—address recognition and routing—as shown in Figure 12.

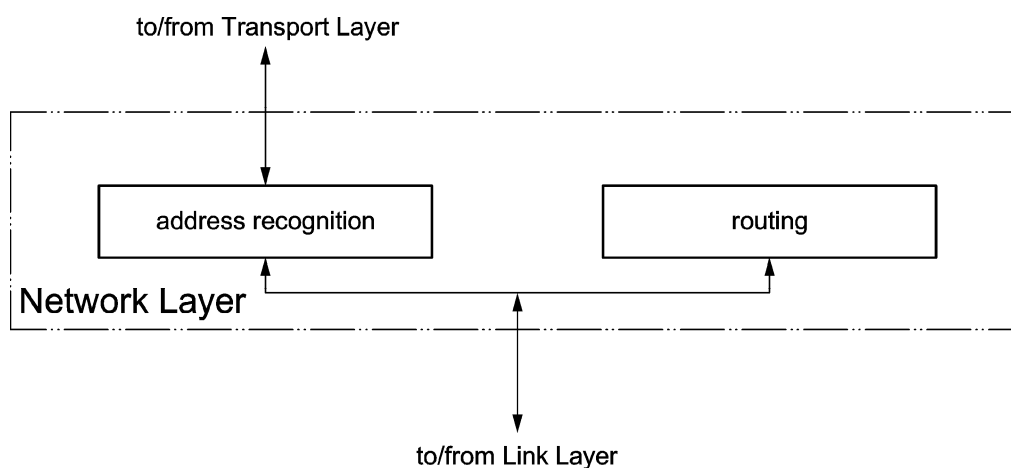


Figure 12 — Network layer—internal structure

8.5 NPDU format

NPDU format is shown in Figure 13. An NPDU carries and encapsulates either a TPDU, SPDU, AuthPDU or APDU. There are no NPDUs defined for internal network layer use. The numbers above each field in Figure 13 specify the field size in bits. The symbolic field values used in the figure are assigned in the order shown, as enumerated ranges (0, 1, 2, 3, ..., n). For example, a value of 3 in the PDU_Fmt field signifies that the enclosed PDU is an APDU. For additional details, including the bit/byte transmission order, refer to Annex D.

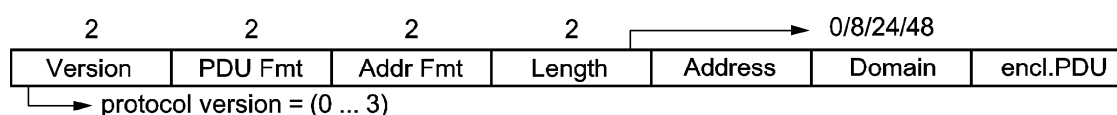


Figure 13 — NPDU format

8.6 Address recognition

Each node compliant to this European Standard performs address recognition. Address recognition at a node depends on the node's state. In the Unconfigured state, all broadcast messages and

messages that match a node's Unique_Node_ID are received. In the Configured state, messages shall either match the Unique_Node_ID or match the domain and be either broadcast, group (i.e., multicast) or unicast messages. To match the domain both the domain length and the domain address shall match. For broadcast messages, the subnet shall match unless the broadcast message is sent domain-wide. For a group message, the node shall be a member of the group, i.e., one of the group addresses shall match. For a unicast message, subnet as well as node numbers shall match. The match of the node number is a comparison of the entire byte even though the node number is defined as seven bits.

In this subclause the information that shall be kept by every node is identified without specifying the address-matching algorithm, that is implementation specific. For an example of an actual implementation of these data structures, refer to A.14.

Any packet received by a node with a protocol version number not supported by the node shall be discarded.

A packet whose source subnet/node address matches that in the matching domain is discarded.

When not in the configured or the hard-offline state, only broadcast (domain-wide or subnet) or matching Unique_Node_ID messages are received. The receive transaction source address is saved based on the actual domain in which the message was received (the flexible domain). Any responses are returned on that domain with a subnet/node number of 0x00/0x80. Either the implementation shall support one flexible domain shared among all receive transactions or the implementation shall support one flexible domain per receive transaction.

8.7 Routers

A router performs the routing function for a specific domain. The routing function is independent of the network layer address recognition function. Thus, a router also has its own network addresses and can be addressed like any other node for purposes of network management and diagnostics. A router connects two sets of subnets. A router is a logical rather than physical entity; more than one router may be housed within a single physical package. Routers have four modes that may be set with network management messages: repeater, bridge, learning, and configured. Routers with their mode set to learning, automatically learn the subnet topology of the network. Routers with their mode set to configured have their routing tables statically configured with network management messages to enforce a specific topology and to provide information on group address topology. Similarly, a router may be set to be a bridge or repeater using the network management messages to set its mode. The same subnets may exist on both sides of a bridge or repeater.

A router uses three routing functions: ROUTE_{uc}(), ROUTE_{mc}(), and ROUTE_{bc}() to forward NPDUs. The first function specifies how to forward NPDUs addressed to subnets, the second how to forward NPDUs addressed to groups, and the third how to forward NPDUs when they are broadcast. The functions are tables, where each entry has the following form:

```
ROUTEuc (DestSubnet) = one of (forward, discard)
ROUTEmc (DestGroup) = one of (forward, discard)
ROUTEbc (DestSubnet) = one of (forward, discard)
```

The entry for an address X specifies whether an NPDu addressed to X should be forwarded or discarded. These functions are called from the side of the router on which the packet was received—that is, the algorithm in A.5 executes independently on each side of the router and makes all routing decisions for packets arriving at the side on which it runs. Each side of the router shall have a different set of tables that have the information as to whether the packet should be passed across the router or not. For configured routers, these tables are initialised during the network configuration of the router and shall be updated if the topology changes. Learning routers discover the contents of their subnet tables by observing the source addresses of the packets (see A.6), but cannot learn the topology of a group. Therefore, learning routers always pass group messages to the other side.

8.8 Routing algorithm

The pseudocode for the routing algorithm is in A.5.

8.9 Learning algorithm — subnets

The subnet routing table defining the routing function $ROUTE_{uc}()$ is created by the algorithm in A.6. Upon initialisation, forwarding is used for all subnet addresses. The algorithm subsequently learns about the location of subnets by observing the source addresses of NPDUs being routed. Again, this algorithm executes on each side of the router independently.

9 Transaction control sublayer

9.1 Assumptions

The transaction sequencing protocol described in this clause uses 4-bit transaction numbers that are allocated by the sender and used by the receiver to detect duplicate packets. This sublayer provides end to end protocol integrity by allowing the receiver to only act upon a packet once, even if several retries for the initial packet are correctly received by the receiver. It is assumed that the network is either order preserving or, if it is not, that packets are delayed approximately uniformly in the multiple paths that they traverse from source to destination. The difference in packet propagation time when there are multiple paths from a given source to a destination shall be less than the time it takes for the source to complete one transaction (via an acknowledgement on the shortest path) and begin a second transaction to the same destination. Stale acknowledgements and responses are detected as duplicates, but stale packets that initiate a transaction and arrive after a subsequent transaction has started will not be detected as duplicates.

The transaction control sublayer may pass a transaction to the higher layers for processing. In some implementations, it may be possible for the upper layers to never respond to the transaction control sublayer. If such a situation exists, the transaction control sublayer can defend itself from such poorly behaved upper layers. During the time that the transaction control sublayer is waiting for a response from the upper layers, the transaction record can be locked and not be de-allocated even if the receive transaction timer expires. Such a transaction is only de-allocated when the upper layers finally respond, or when there are no more transaction records available and one is needed. Then, a locked transaction record may be used for the new transaction. Use of this mechanism is implementation dependent.

Generally, the number of concurrent outgoing transactions is restricted to a single priority and a single non-priority transaction. This restriction is because acknowledgements/responses from either a subnet/node addressed message/request or a Unique Node ID addressed message/request are indistinguishable. Implementations that restrict the use of one of these addressing modes may support multiple, concurrent transactions. Receive transactions are shared among the transport and session layers. Transactions at either layer can perform duplicate detection.

The transaction timer, receive timer, and retry count interact to establish the reliability of the duplicate detection mechanism. It is assumed that the receive timer is set to be long enough to cover the configured number of retries, yet short enough so that the transaction ID does not wrap around causing a new transaction to be falsely detected as a duplicate transaction. In an implementation, it is required to have a transaction space for all priority transactions and a second transaction space for all non-priority transactions. If the sender wishes to send a packet to a given destination and the transaction ID for the last message sent to that destination is identical to the current transaction ID, the transaction ID is incremented. An optional enhancement is to have a separate priority and non-priority transaction space for every destination address. In this second case, the sender remembers the last transaction ID used per destination address. In any case, an implementation shall defend against the situation that acknowledgements/responses cannot be differentiated from each other when using subnet/node addressing or Unique Node ID addressing. Other implementations are permitted as long

as the duplicate detection mechanism remains robust in all of the error cases and all of the addressing formats in use by the node.

9.2 Service provided

The transaction control sublayer is responsible for the common functions related to transaction ordering, sequencing, and duplicate detection. It provides the following services:

- Outgoing Sequencing. To guarantee ordering among outgoing transport messages and session layer requests, the transaction control sublayer controls the allocation of send transaction numbers. It limits the number of concurrent transactions to any destination to ≤ 1 priority and ≤ 1 non-priority transactions;
- Incoming Sequencing and Duplicate Detection. The transaction control sublayer provides duplicate detection.

9.3 Service interface

Access to transaction control services is facilitated by the interface depicted in Figure 14.

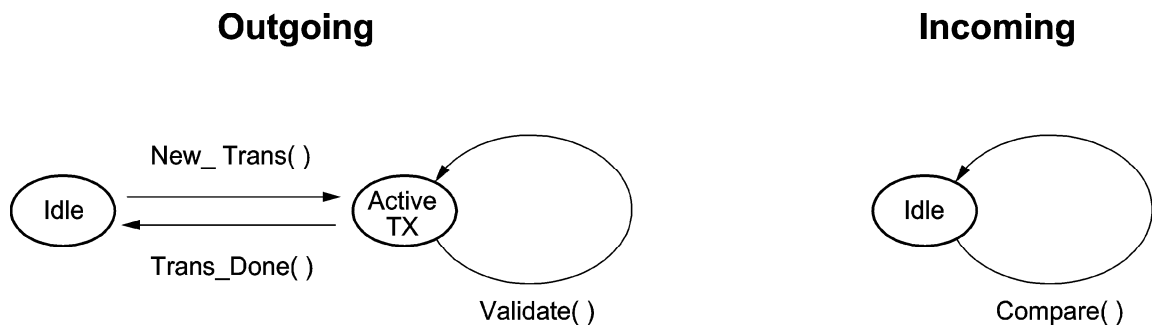


Figure 14 — Transaction control service interface

The syntax and semantics of the interface primitives are:

```

New_Trans (Priority, Dest_addr) -> (Trans_No)
    is used to obtain a transaction number for a new outgoing transaction

Validate (Priority, Trans_No, Dest_addr) -> (result)
    where result = one of (current, not_current), verifies that Trans_No is in
    the transmit window

Trans_Done (Priority, Trans_No, Dest_addr)
    notification of an outgoing transaction completion

Compare (T1, T2) -> (result)
    where result is one of (new, duplicate ); defines the relationship of T1
    relative to T2, where both T1 and T2 are receive transaction numbers

```

9.4 State variables

To support the allocation of transaction numbers, the transaction control sublayer uses a number of destination records shown below.

```

Transaction_CTRL_Record = record
    PriTX                True or False
    Trans_No:            (0..15); initial value = 0;
    In_Progress:         boolean;
end;

```

9.5 Transaction control algorithm

The pseudocode algorithm in A.7 is a version of the transaction control algorithm that provides only two transaction spaces – one for all priority transactions and one for all non-priority transactions. In any implementation, the first transaction ID provided by New_Trans() after a reset shall be 0 for every transaction space provided by the implementation. New_Trans() shall then increment the transaction ID within the space from 1 to 15 and continue again with 1 so that the transaction ID 0 is used exclusively by the first transaction per transaction space after a reset.

In general, any transaction control algorithm shall follow the following basic operation:

- When a node has a packet to send using the transaction control algorithm, the sender node picks a transaction ID and sends the packet.
- Upon receipt of that packet, the receiver node searches its currently active receive transactions for a transaction that matches the source and destination address of the packet, the priority attribute, and the transaction ID.
- If there is no match a new record is allocated.
- If everything but the transaction ID matches and the existing transaction is not locked, then the preceding transaction is assumed to have completed successfully and the receive transaction record may be reused with the new transaction ID inserted into it. Processing of the incoming packet cannot proceed until a receive transaction is allocated for it.
- Once the receiver has allocated the transaction record, it starts a receive transaction timer for that record. The value of the receive transaction timer is 8 s for a Unique_ID addressed message, the value of the non-group receive timer for a subnet/node addressed message, and, for group addressed messages, the receive transaction timer configured for the group. The record will be de-allocated upon the expiration of this timer unless the transaction record becomes locked (see 9.1).
- Subsequent packets that arrive at the receiver node that match the transaction ID, source address, destination address and priority attribute of an existing transaction will be considered duplicates by the receiver node.
- Packets detected as duplicates are responded to according to the protocol service requested, but are either not delivered to the application, or if they are delivered to the application, they are delivered along with the notification that they are duplicate packets.

10 Transport layer

10.1 Assumptions

The Transport protocol makes no assumptions apart from relying on the transaction control sublayer for correct TPDU sequencing and duplicate detection.

10.2 Service provided

The transport layer provides the following services:

- Reliable Multicast and Unicast. The transport protocol supports both multicast within a group, and multicast to a group with the sender not being a member of the group. If the service requires acknowledgements or responses, then multicast to a group that the sender is not a member of requires the transport layer to be told that the group size is the actual group size plus one. All reliable services have the following attributes:
- (a) reliable delivery with best effort determined by the number of retries;
- (b) assuming the layer 4 timers are set correctly, duplicate detection is provided in all cases except when the sender or receiver just reset. In this case, the reset node shall start with

transaction number 0. This may result in a transaction in progress between the two nodes not to be acted upon at all, but acknowledged/responded to by the receiver;

- (c) partial ordering—ordering is preserved but a message is not delivered when delivery fails within the specified number of retries; and,
- (d) immediate re-synchronisation—following a network partitioning, the very first message is delivered.
- Unacknowledged-Repeated Multicast and Unicast. These services differ from the reliable ones described above only in that no acknowledgement is expected, and that the message is sent repeatedly until the number of repetitions is equal to the retry count. When using this service, the limit of 64 members in a group does not apply—the only limit on the number of members in a group addressed via this service is the number of nodes in a domain.

Groups in this European Standard are symmetric in that every member of the group can both send and receive.

10.3 Service interface

The service interface provided to the session and application layers has the form shown in Figure 15.

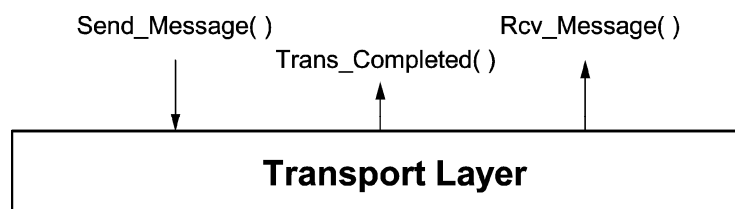


Figure 15 — Transport interface to upper layers

The syntax and semantics of the transport layer interface are:

```

Send_Message (Address, APDU, priority, ServiceType, alt_path) -> (TID)

Trans_Completed (TID, Result)

Rcv_Message (APDU)
    
```

TID, above, is a unique identifier for the transaction.

10.4 TPDU types and formats

TPDU syntax is shown in Figure 16; the number above each field specifies the field size in bits. The symbolic field values shown in the picture are mapped onto numeric ranges (0, 1, 2, 3...) in the order shown. Additional details, such as the bit/byte transmission order, are defined in Annex D. Timers and counters used to transmit retries, count retries, and time out transaction IDs, are described in B.4.11.

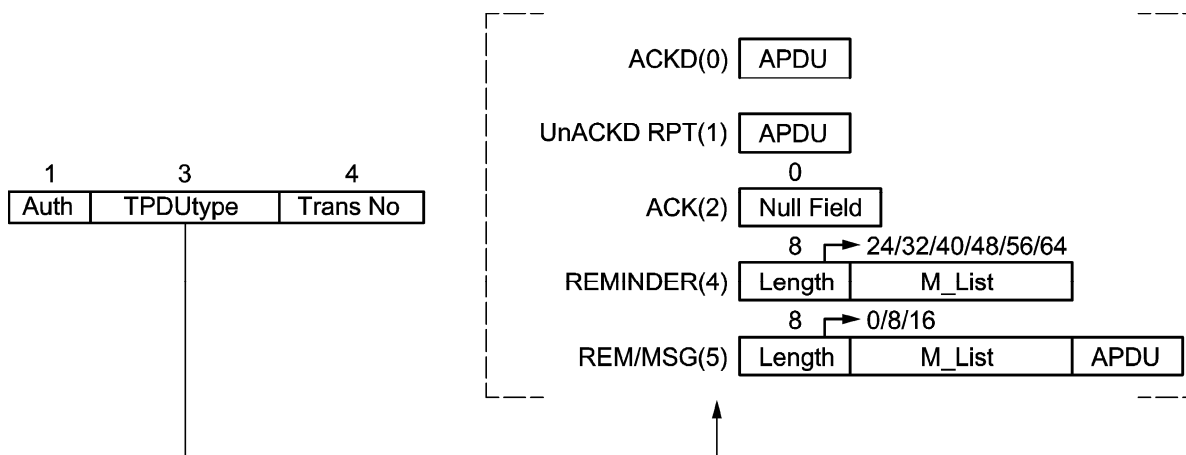


Figure 16 — TPDUs and formats

The *Acknowledged Message* (ACKD) TPDUs (see Figure 34) is used for the first transmission of a message. It is used with addressing formats #1, #2a, and to a limited extent #3. Unlike the Unacknowledged message TPDUs, it shall be acknowledged by all addressed recipients, thus the transport layer provides the number of acknowledgements expected to the link layer for encoding into the link layer header. This TPDUs is used for acknowledged initial TPDUs (both unicast and multicast) as well as unicast reminders. Multicast reminders are covered below. Refer to Figure 36 or Figure 34 to review the addressing formats.

The *Unacknowledged-repeated Message* (UnACKD_RPT) TPDUs is identical to its acknowledged counterpart with one exception: upon receipt, no acknowledgements are returned to the sender. This TPDUs is used with no modifications for the unacknowledged-repeated service. For this type of message the transport layer provides a backlog increment equal to the number of retries on the first message and a backlog increment of zero on all subsequent transmissions within the transaction. Simple unacknowledged messages are only sent once, have no TPDUs header, and thus have no duplicate detection.

The *Message-Reminder* (REM/MSG) TPDUs facilitates selective soliciting of acknowledgements for multicast transactions. REM/MSG type 5 is used when the highest numbered group member from which the sender has received an acknowledgement is < 16; this TPDUs contains both the member list (M_List []), that is an array of bits, and the APDU. The Length field specifies (in bytes) the size of the M_List field. A value of 0 in the bit field M_List [X] indicates that member X's acknowledgement has not been received by the sender, whereas a value of 1 indicates that the acknowledgement has been received. Bit locations within a byte are assigned right to left with bit number zero on the right. When Length = 0, the M_List[] field is absent and the meaning is "all members should acknowledge." The backlog increment for this message is the number of acknowledgements that have not yet been received.

Type 4 is a plain *reminder*, without an APDU (see Figure 16). It is used in cases where the highest numbered member that has acknowledged the message is ≥ 16. Acknowledgements are solicited using the TPDUs pair (REMINDER, ACKD) and this pair is logically equivalent to a single type 5 REM/MSG TPDUs used when the members needing to acknowledge may be encoded within the type 5 format. (This separate solution is provided for large groups because of the need to limit maximum TPDUs size.) This message always has a backlog increment of zero. The ACKD message that immediately follows its transmission has its backlog increment set to the number of acknowledgements that are yet to be received.

The *Acknowledgement* (ACK) TPDUs is null. It uses addressing format #2a (unicast acknowledgement) or #2b (group acknowledgement). The Trans_No field conveys the transaction being acknowledged. Acknowledgements always inherit the priority and alt_path attributes of the original message.

Any TPDU that requires an acknowledgement can be flagged as an authenticated packet by setting the Auth bit to '1'.

Codes 3, 6, and 7 are reserved. Packets with these codes are discarded.

10.5 Protocol diagram

The diagram in Figure 17 augments—but does not replace—the protocol description in 10.7 and 10.8.

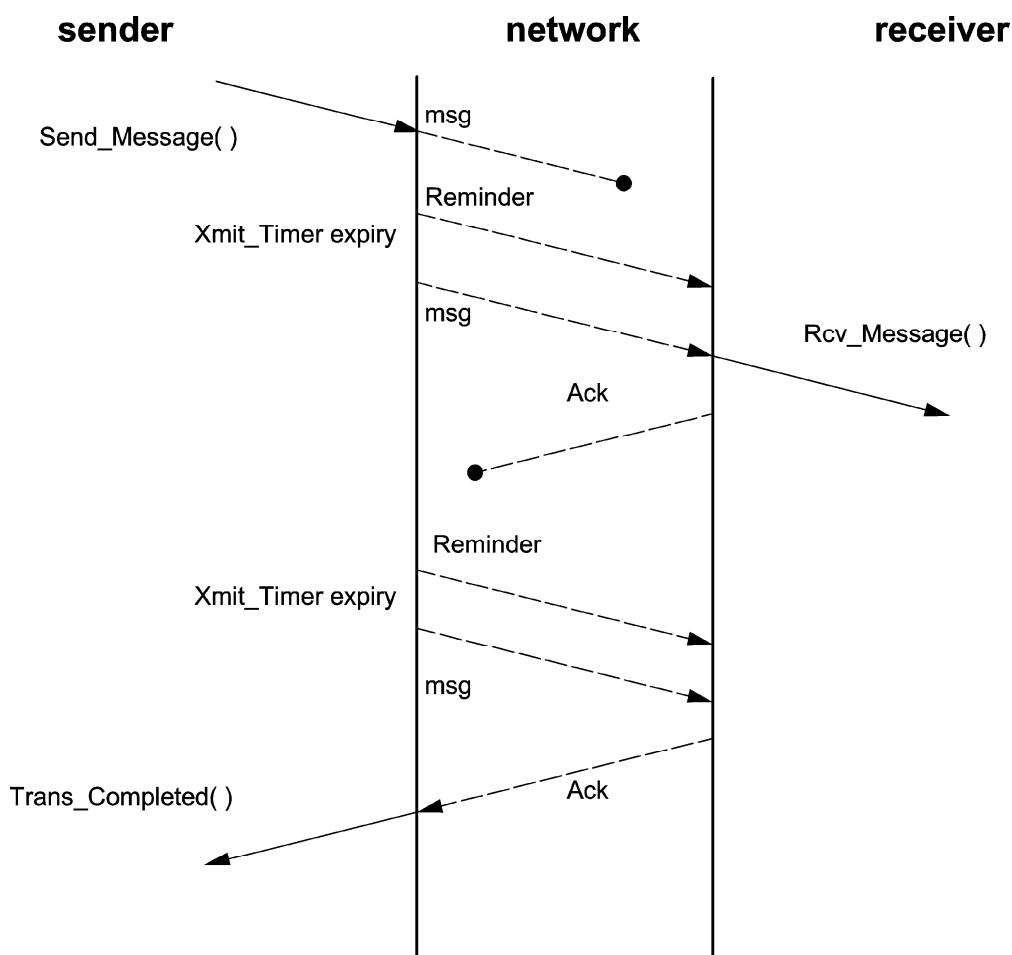


Figure 17 — Transport protocol diagram for multicast message with a loss of both the message and the ACK TPDUs

10.6 Transport protocol state variables

The transport protocol consists of two independent functions—Send and Receive. To support the send function, the transport layer keeps one Transmit record per transaction in progress. A shared pool of Receive records facilitates message reception. The Transmit and Receive data structures are described in A.8 and A.9.

10.7 Send algorithm

The simplified transmit FSM is shown in Figure 18, with full details in the algorithm in A.9. The algorithm resets the re-transmission timer, Xmit_Timer, whenever an acknowledgement or challenge is received, as opposed to once per Expiration period. The alternate path bit is set in the packet on the last two attempts by this layer notifying the lower layers of the alternate path attribute for each attempt.

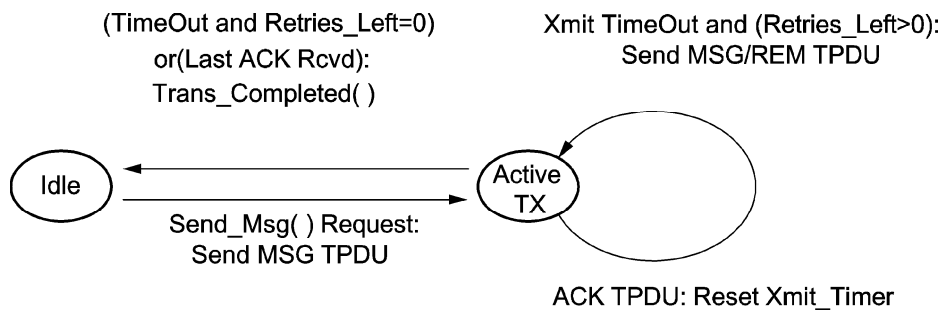


Figure 18 — Transport protocol—Send FSM

10.8 Receive algorithm

Message reception uses the timer-based mechanism. Figure 19 shows the receive FSM for a single transaction, while the algorithm in A.9 specifies full details.

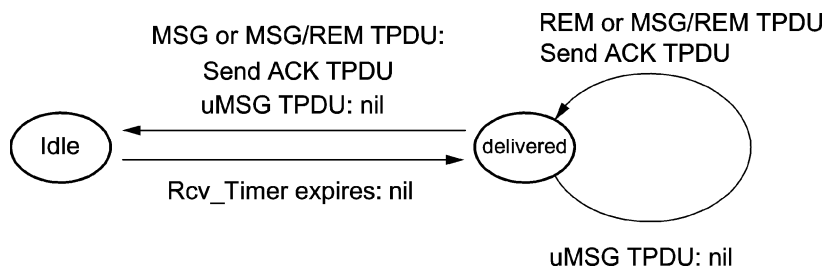


Figure 19 — Transport protocol—Receive FSM

10.9 Receive transaction record pool size and configuration engineering

10.9.1 General

Space for the Receive Transaction Record (RTR) pool defined in 9.5 and A.9, may be fixed for an implementation, or may be allocated at the time the application program is linked with the protocol code. The size of the RTR pool limits the number of *concurrent receive* operations on a node ("concurrent" means concurrent within the context of Rcv_Timer) and should be engineered according to the number of concurrent transactions expected within the receive timer interval. If an attempt is made to allocate a receive transaction record and none are available, the receive transaction full statistic is incremented (see 13.8.2).

10.9.2 Number of retries

The number of retries should be large enough to ensure that message delivery is successfully completed with acceptable probability, e.g., $\geq 99\%$. However, the upper bound of the number of retries in this European Standard is 15.

If the delivery failure probability of a single attempt is p , then the probability that message delivery within a group of n succeeds in $\leq k$ attempts is shown in the following formulae. (It is assumed that the single attempt probability p is the same for all destinations.)

$$P \{ \text{no retry} \} = (1-p)^n$$

$$P \{ \leq k \text{ retries} \} = \sum_{i=0}^k \binom{k+1}{i} p^i (1-p)^{(k-i+1)} (1-p^{(k-i+1)})^{(n-1)}$$

$$k \geq 0, n \geq 2$$

(13)

NOTE For a single channel $p = (1/2w + p_e)$, where w is the size of the MAC layer randomising window and p_e is the probability of packet loss because of a transmission error.

The above probabilities are graphed in Figure 20. Given a group size and the error probability p , the required number of retries for, say 99,5 % probability of success, can be read off the graph.

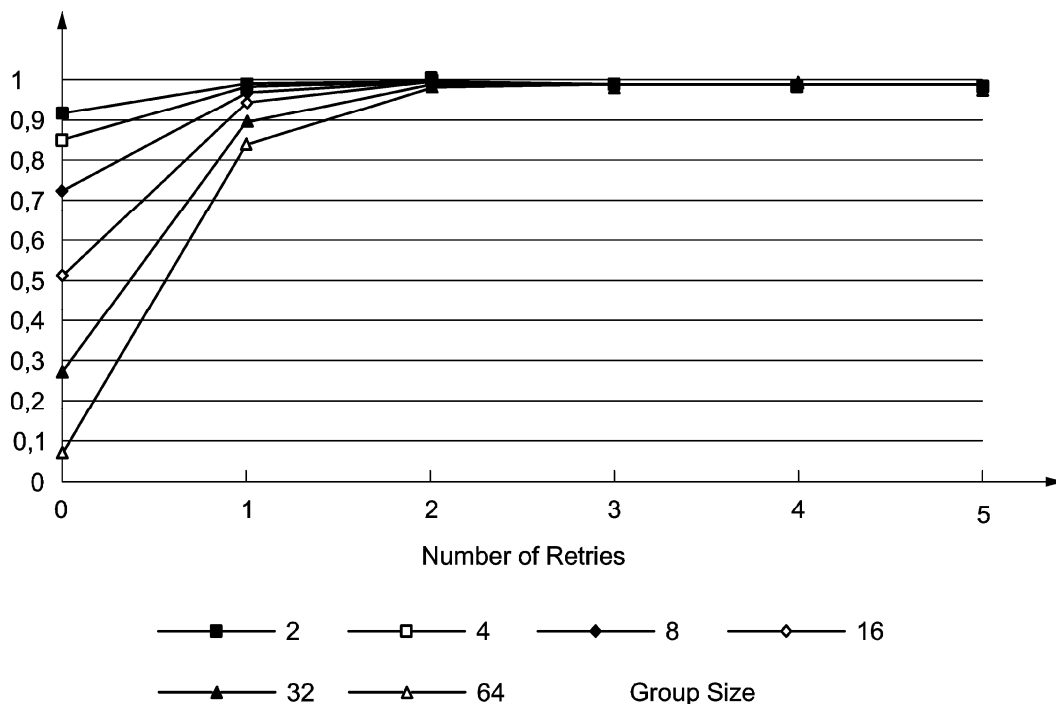


Figure 20 — Probability of transaction completion in k Retries

Pragmatic considerations should be respected in any real system design. For example, there is no need for a large number of retries with transactions that are repeated periodically. Additionally, when the acknowledgements to a multicast message would take up a significant percentage of the bandwidth of the channel, it is more likely that a message will be properly delivered using the “unacknowledged-repeated” service rather than the acknowledged multicast service. In practice, it is expected that a Retry_Count in the (2.to.5) range will cover most situations.

10.9.3 Transport layer timers

There are three timers used by the transport layer protocol. They are the:

- Xmit_Timer the layer 4 retransmission timer
- Repeat_Interval_Timer the UNACKD_RPT interval timer
- Rcv_Timer the receive record timer (see A.9 and 10.9)

These timers are active for every transaction. On the sender's side, either the Repeat_Interval_Timer or the Xmit_Timer is active, while on the receive side, the Rcv_Timer is active. Xmit_Timer is reset on

every ACK TPDU or Authentication challenge reception, while Rcv_Timer is reset whenever a MSG or MSG/REM TPDU that has a new transaction number is received for this destination. The Repeat_Interval_Timer determines the interval between the UNACKD_RPT packets sent by the sender. The recommended methodology for calculating the timer values is shown in Figure 21. These recommendations are for unicast transactions across a single channel. For multi-channel networks, the calculation depends upon the speed of the router and the number of buffers. Multicast transactions need somewhat longer receive transaction values since multicast transactions take longer to complete even when there are no retries needed. Since buffering and clock rates are adjustable, the system designer shall make some measurements to set the timers and retry counts correctly within each node.

Retry Count = 2 to 5	(see 10.9.2 and Figure 20)
Xmit_Timer $\geq 3 \times$ packet cycle time + margin	(margin = best case tx completion time)
Rcv_Timer $\geq$ Xmit_Timer * (Retry_Count + 2)	
where:	
“3 * packet cycle time” assumes that the average station on the channel must wait two packet cycles of delay to access the network. Then, following network access, the transmission time of the average packet (also the packet cycle time) is added to the timer value. Finally, the time it takes the receiver to process the packet and send the acknowledgement is added. This is a function of the clock speeds of the associated nodes.	

Figure 21 — Methodology for calculating timer values

All messages using the Unique_Node_ID use an 8-second receive timer, regardless of the configured value of the non-group receive timer. All non-group messages use the non-group receive timer.

11 Session layer

11.1 Assumptions

The session layer makes no assumptions apart from relying on the transaction control sublayer for correct SPDU sequencing and duplicate detection.

11.2 Service Provided

The session layer provides a single service:

- Request-Response. This service facilitates application communication similar to a remote procedure call. In particular, it allows a client to make a request to a remote server and receive a response to this request.
- *Non-idempotent* transactions are to be executed “*at most once*” (i.e., exactly once or not at all). Non-idempotent transactions are those where the action depends on a prior state, such as “open the valve an additional 10 %.” The protocol considers a transaction to be non-idempotent if and only if its response length is ≤ 1 byte.
- Requests with response length > 1 byte are considered *idempotent*; such requests may be executed “*several times*” (i.e., zero or more times). Idempotent transactions are those where the action may be repeated any number of times, and the effect is the same. An example of an idempotent command is “read the first 10 table values”.
- The distinction between idempotent and non-idempotent transactions is based upon the size of the response, in order to limit the amount of storage required for transaction records within a node. If a transaction has a response length > 1 byte but may not be executed more than once, it is the responsibility of the application to save the response and send it again. The

session layer facilitates this in that notification that the request is a duplicate is provided to the application layer by the session layer.

- A request-response transaction fails unless the server response is generated within the limit imposed by the Request-Response timers and re-transmissions (see 10.8).

11.3 Service interface

The service interface to the application layer has the form shown in Figure 22.

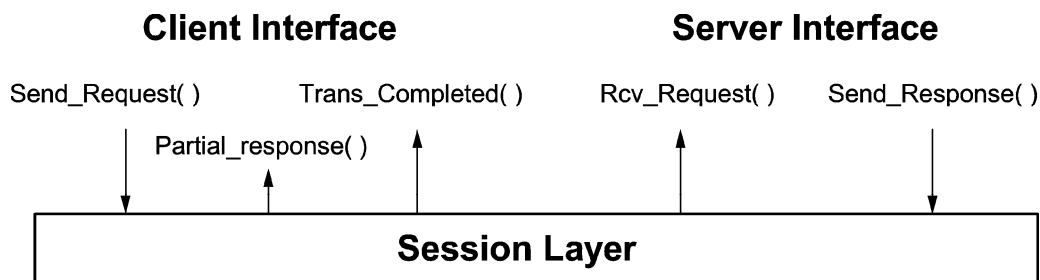


Figure 22 — Session layer interface to application layer

The syntax of the service interface primitives is given below. TID is a unique identifier for the transaction. See A.7 for implementation details. Duplicate is a Boolean that, when true, indicates that the client is retrying a previously executed request. Result is a Boolean; true and false denotes success and failure respectively.

Send_Request (Address, APDU, priority) -> (TID) {client}
 Partial_Response (TID, APDU, priority)
 Trans_Completed (TID, Result)
 Rcv_Request (TID, APDU, duplicate, priority) {server}
 Send_Response (TID, APDU, priority)

11.4 Internal structure of the session layer

The session layer is internally structured as shown in Figure 23.

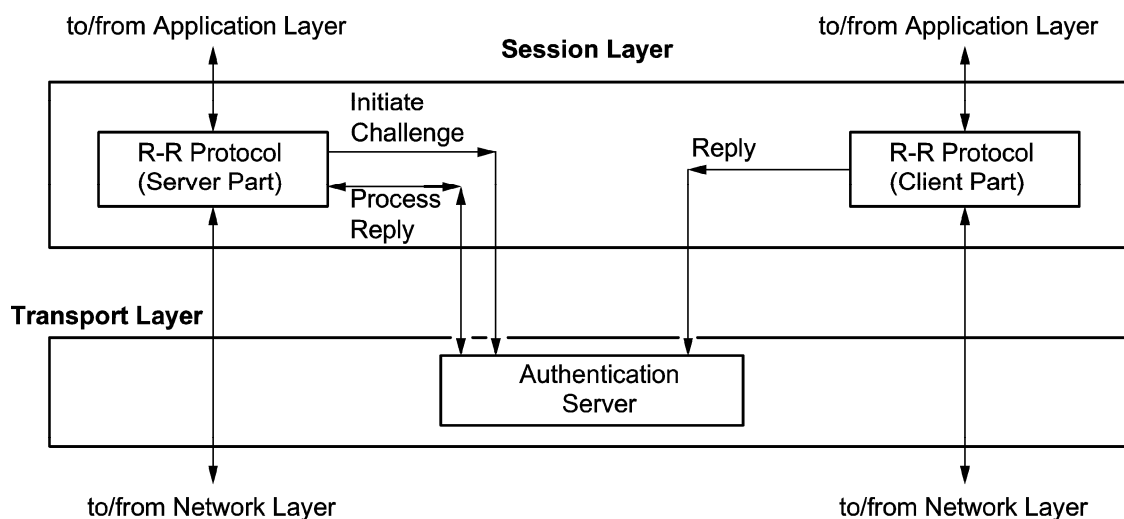


Figure 23 — Session layer—internal structuring

The Request-Response protocol accesses the Authentication server via the following three calls (details in 11.11). The transport layer also accesses the Authentication server with the same three calls.

Initiate_Challenge(RR,PDU) -> (null) { challenger }

Reply(XR, PDU)-> (null) { challengee }

Process_Request(RR,PDU) -> (pass/fail); { challenger }

11.5 SPDU types and formats

SPDU formats are shown in Figure 24, where the number above each field specifies the field width in bits. The symbolic values shown in the picture are mapped onto numeric ranges (0, 1, 2, 3, ...) in the order shown.

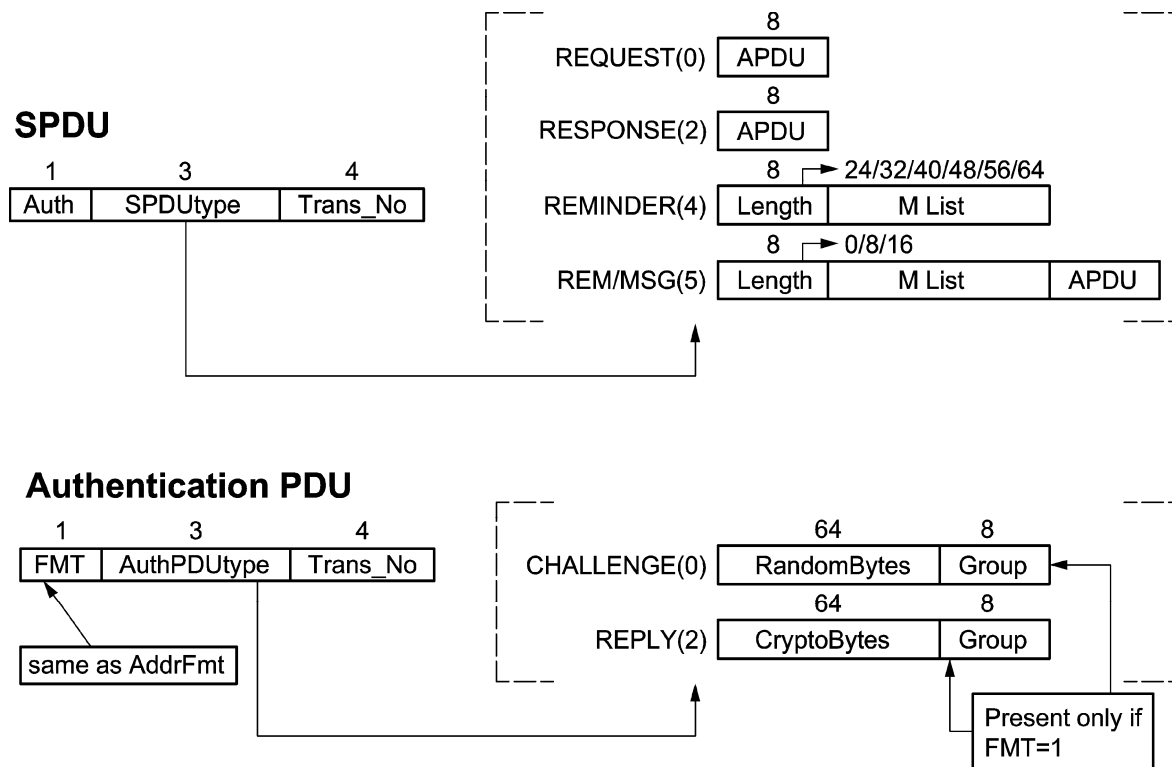


Figure 24 — SPDU types and formats

The Request-Response protocol uses three basic PDU types: Request (REQUEST), Response (RESPONSE), and combined Request-Reminder (REM/MSG). The syntax (packet layout) and semantics (packet processing) of these three basic SPDU types correspond closely to that of ACKD, ACK, and REM/MSG TPDU. Note that although the session layer makes use of the transport layer features internally, no transport layer encoding exists separately in a packet with an SPDU. For instance, a packet may have either a TPDU or an SPDU.

The *Request* (REQUEST) SPDU is used with the first transmission of the request. It employs addressing formats #1, #2a, and to a limited extent #3 and #0 (see Figure 34). Address format #0 is used by a special network management command to broadcast a request searching for any nodes that have not been configured. Address format #3 is then used to configure those nodes that respond to this special network management command. The session layer provides a backlog increment to the lower layers equal to the number of responses expected.

The *Request-Reminder* (REM/MSG) SPDU facilitates selective soliciting of responses. REM/MSG type 5 is used in groups where the highest member number needing to acknowledge is < 16 ; this SPDU contains both the member list (M_List []), that is an array of bits, and the APDU (i.e., the request itself). The Length field specifies (in bytes) the size of the M_List field. A value of 0 in M_List [X] indicates that member X's response has not been received by the requester, whereas a value of 1 indicates that the response has been received. Bit locations are assigned right to left within each byte with bit 0 being the rightmost bit in the byte. The backlog increment encoded with the REM/MSG is equal to the number of remaining responses outstanding.

Type 4 is a plain reminder, without a request (see Figure 24). It is used where the highest member number needing to acknowledge the reminder is ≥ 16 ; in this case, responses are solicited using the SPDU pair (REMINDER-type 4, REQUEST-type 0) and this pair is logically equivalent to a single type 4 REM/MSG SPDU used in small groups. (A separate solution is provided for large groups because of the need to limit maximum SPDU size.) Finally, when Length = 0 the M_List[] field is absent and the meaning is "all members should acknowledge." The backlog increment for the REMINDER – type 4 message is always zero, while the REQUEST that immediately follows it contains the backlog increment for the number of responses that are still outstanding.

The *Response* (RESPONSE) SPDU uses addressing format #2a (unicast acknowledgement) or #2b (group acknowledgement). The Trans_No field conveys the transaction being acknowledged. The length of the APDU implicitly defines the type of transaction: if the response can be stored in a single byte, the transaction is treated as non-idempotent. Otherwise, the transaction is treated as idempotent. Responses inherit the priority and alt_path attributes of the Request packet.

Authenticated SPDUs (Auth bit set to '1') identify requests that are to be authenticated by the recipient. In all other respects, they are identical to the SPDUs that are not authenticated.

Authentication. The authentication server is a single server contained in Layer 4 and available to the transport and session layer protocols. It provides a one-way authentication service. It is the client's responsibility to initiate authenticated transactions when required. Setting the Auth bit in the SPDU or TPDU does this. When a TPDU or SPDU is received with the Auth bit set, the server shall challenge using the "challenge" AuthPDU. The FMT field in the authentication PDU shall be set to the same value as in the address format field of the NPDU header in the message being challenged. The client then computes a transformation based upon the server's challenge, the original APDU sent by the client, and the client's authentication key. The result of this transformation is sent to the server using the "reply" AuthPDU. The FMT field in the reply shall be set to the same value as the FMT field in the corresponding challenge AuthPDU. When the server receives the reply, its contents are compared to the transformation computed by the server. If they match, the transaction is authenticated. In all cases the SPDU/TPDU is passed to the application layer for processing along with notification as to whether the authentication failed or succeeded. Note that if the application layer on the server node has no requirement for authentication of the particular transaction, it may choose to honour the request even if the authentication failed. If the application layer chooses not to honour the request, it simply discards the APDU without further processing.

11.6 Protocol timing diagrams

The protocol timing diagrams in Figure 25 and Figure 26 are intended to provide an intuitive feeling for the session layer protocols and to augment (but not to replace) the protocol specification in 11.7 to 11.13. Note again that for needed acknowledgements from group member numbers < 16 , the REM/MSG SPDU is used and is functionally equivalent to the (REMINDER, REQUEST) pair.

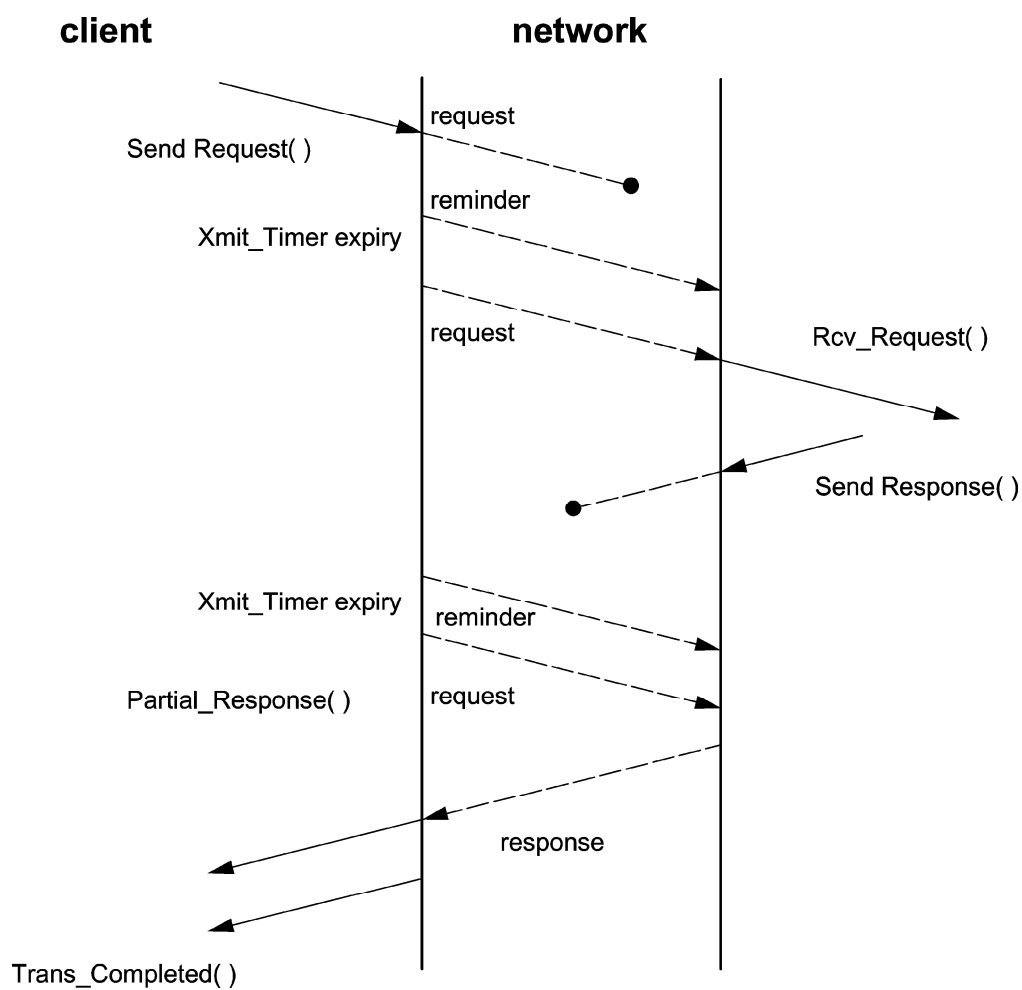


Figure 25 — Non-Idempotent request with multiple SPDU losses