



DLMS/COSEM transport layer for IP networks

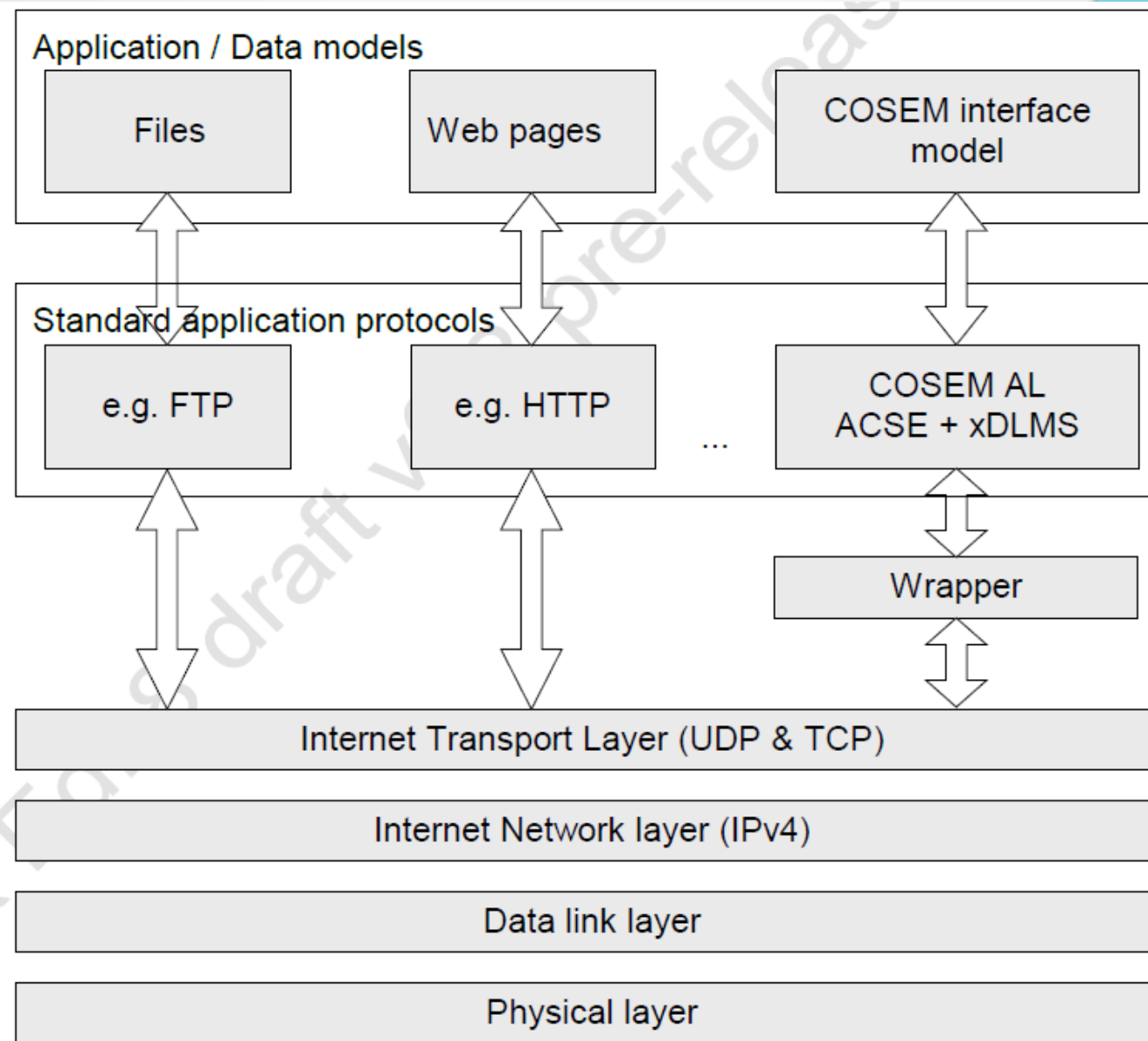
■ SCOPE

- Specifies a connection-less and a connection oriented transport layer (TL) for DLMS/COSEM communication profiles used on IP networks.
- These TLs provide OSI-style services to the service user DLMS/COSEM AL.
- The connection-less TL is based on the Internet Standard User Datagram Protocol (UDP). The connection-oriented TL is based on the Internet Standard Transmission Control Protocol(TCP).
- Although the major part of the DLMS/COSEM TLs is the UDP and TCP, it also includes an additional sublayer, called wrapper.

OVERVIEW

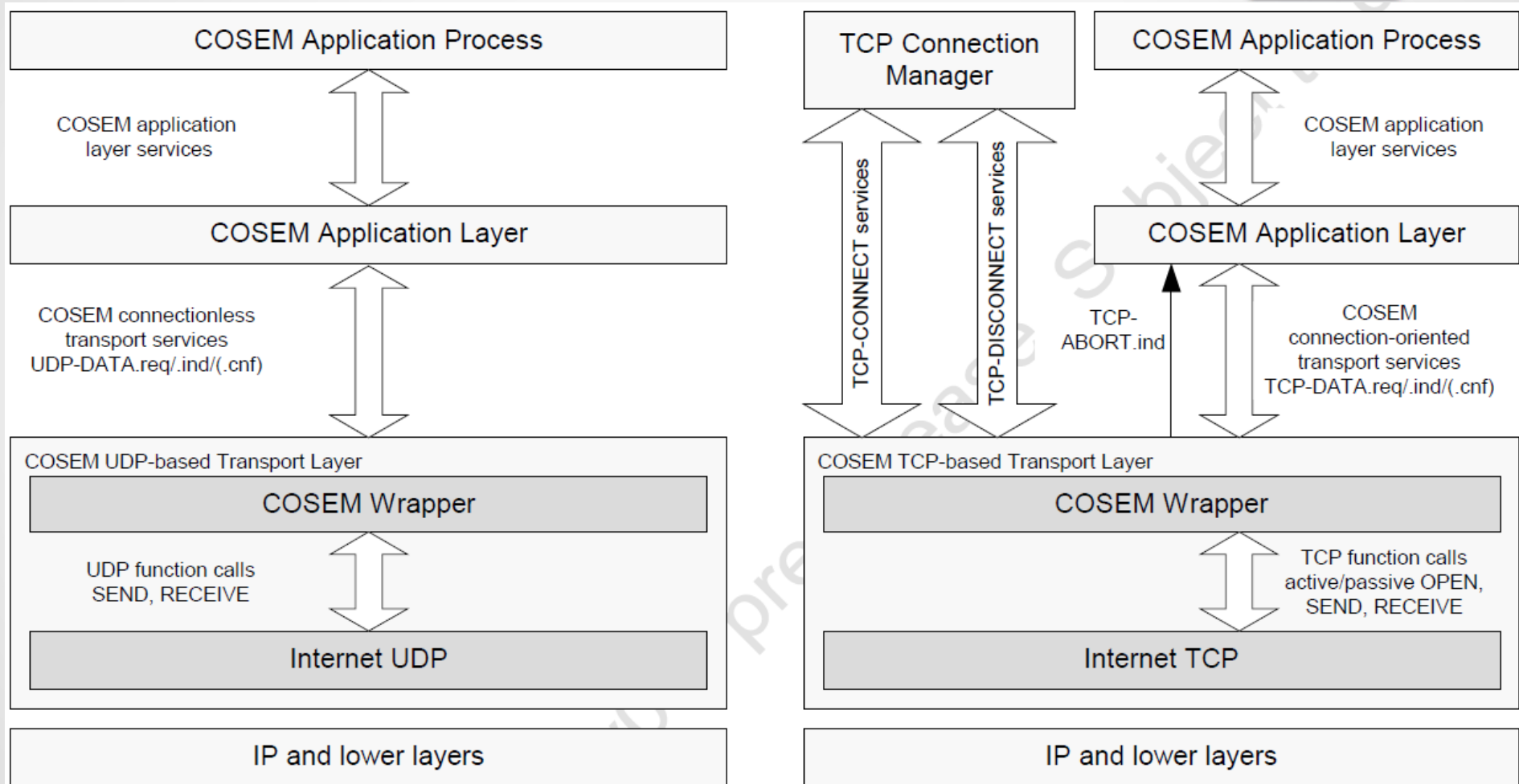
- In the DLMS/COSEM_on_IP profiles, the DLMS/COSEM AL uses the services of one of these TLs, which use then the services of the Internet Protocol (IP) network layer to communicate with other nodes connected to the abstract IP network.
 - When used in these profiles, the DLMS/COSEM AL can be considered as another Internet standard application protocol (like the well-known HTTP, FTP or SNMP) and it may co-exist with other Internet application protocols, as it is shown in Fig.
 - For DLMS/COSEM, the following port numbers have been registered by the IANA.
<http://www.iana.org/assignments/port-numbers>
-
- + dlms/cosem 4059/TCP DLMS/COSEM
 - + dlms/cosem 4059/UDP DLMS/COSEM

DLMS/COSEM as a standard Internet application protocol



- The DLMS/COSEM TLs consist of a wrapper sublayer and the UDP or TCP TL.
- The wrapper sublayer is a lightweight, nearly state-less entity:
- Main function is to adapt the OSI-style service set, provided by the DLMS/COSEM TL, to UDP or TCP function calls and vice versa. In addition, the wrapper sublayer has the following functions:
 - + It provides an additional addressing capability (wPort) on top of the UDP/TCP port
 - + It provides information about the length of the data transported . This feature helps the sender to send and the receiver to recognize the reception of a complete APDU, which may be sent and received in multiple TCP packets.

Transport layers of the DLMS/COSEM_on_IP profile



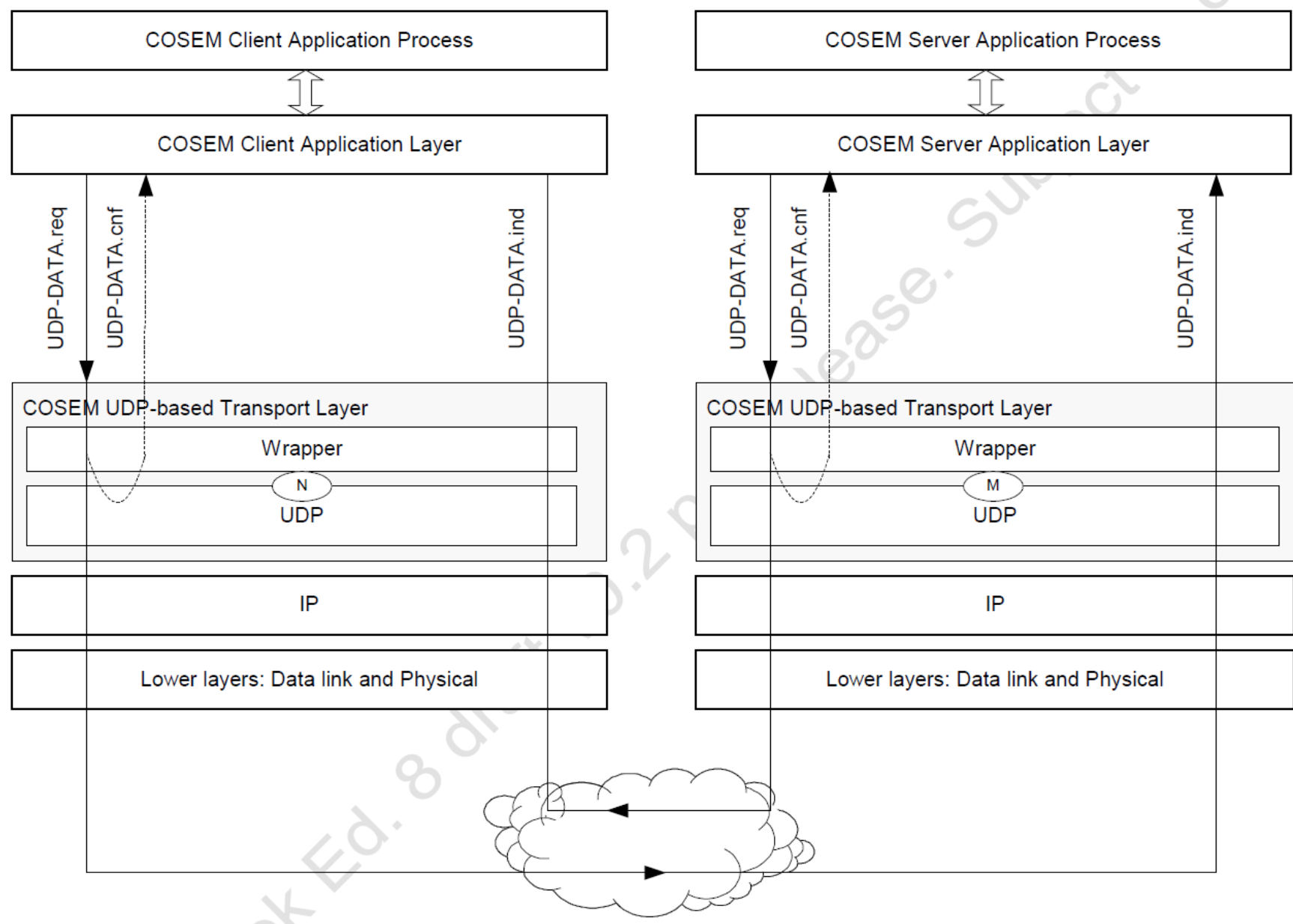
■ The DLMS/COSEM connection-less, UDP-based transport layer

- UDP provides a procedure for application programs to send messages to other programs with a minimum of protocol mechanism.
- It is transaction oriented, and delivery and duplicate protection are not guaranteed.
- UDP is simple, it adds a minimum of overhead, and it is efficient and easy to use.
- Another advantage of UDP is that being connection-less, it is easily capable of multi- and broadcasting.
- UDP basically provides an upper interface to the IP layer, with an additional identification capability, the UDP port number. This allows distinguishing between APs, hosted in the same physical device and identified by its IP address.

■ Service specification for the DLMS/COSEM UDP-based transport layer

- The DLMS/COSEM UDP-based TL provides only a data transfer service: the connection-less UDP-DATA service. Consequently, the service specification for this service is the same for both the client and server TLs, as it is shown in Figure.
- The .request and .indication service primitives are mandatory. The implementation of the local .confirm service primitive is optional.
- The xDLMS APDU pre-fixed with the wrapper header shall fit in a single UDP datagram.

Services of the DLMS/COSEM connection-less, UDP-based transport layer



■ The UDP-DATA service

UDP-DATA.request :

Function :

This primitive is the service request primitive for the connection-less mode data transfer service.

Semantics of the service primitive :

UDP-DATA.request

(
 Local_wPort,
 Remote_wPort,
 Local_UDP_Port,
 Remote_UDP_Port,
 Local_IP_Address,
 Remote_IP_Address,
 Data_Length,
 Data
)

Use:

- The UDP-DATA.request primitive is invoked by either the client or the server DLMS/COSEM AL to request sending an APDU to a single peer AL, or, in the case of multi- or broadcasting, to multiple peer ALs.
- The reception of this service primitive shall cause the wrapper sublayer to pre-fix the wrapper header to the APDU received, and then to call the SEND() function of the UDP sublayer with the properly formed WPDU, as DATA.
- The UDP sublayer shall transmit the WPDU to the peer wrapper sublayer as described in STD0006

UDP-DATA.indication

Function :

This primitive is the service indication primitive for the connection-less mode data transfer service.

Semantics of the service primitive :

UDP-DATA.indication

(
Local_wPort,
Remote_wPort,
Local_UDP_Port,
Remote_UDP_Port,
Local_IP_Address,
Remote_IP_Address,
Data_Length,
Data
)

Use :

- The UDP-DATA.indication primitive is generated by the DLMS/COSEM UDP based TL to indicate to the service user DLMS/COSEM AL that an APDU from the peer layer entity has been received.
- The primitive is generated following the reception of an UDP Datagram by the UDP sublayer, if both the Local_UDP_Port and Local_wPort parameters of the message received contain valid port numbers, meaning that there is a DLMS/COSEM AE in the receiving device bound to the given port numbers.
- Otherwise, the message received shall simply be discarded.

UDP-DATA.confirm

Function :

This primitive is the optional service confirm primitive for connection-less mode data transfer service.

Semantics of the service primitive :

UDP-DATA.confirm

```
(  
  Local_wPort,  
  Remote_wPort,  
  Local_UDP_Port,  
  Remote_UDP_Port,  
  Local_IP_Address,  
  Remote_IP_Address,  
  Result  
)
```

Use :

- The UDP-DATA.confirm primitive is optional. If implemented, it is generated by the DLMS/COSEM TL to confirm to the service user DLMS/COSEM AL the result of the previous UDP-DATA.request. It is locally generated and indicates only whether the Data in the .request primitive could be sent or not.
-
- An UDP-DATA.confirm with Result == OK means only that the Data has been sent, and does not mean that the Data has been (or will be) successfully delivered to the destination.

Protocol specification for the DLMS/COSEM UDP-based transport layer

DLMS/COSEM-specific light-weight wrapper sublayer

- The wrapper sublayer is a state-less entity.

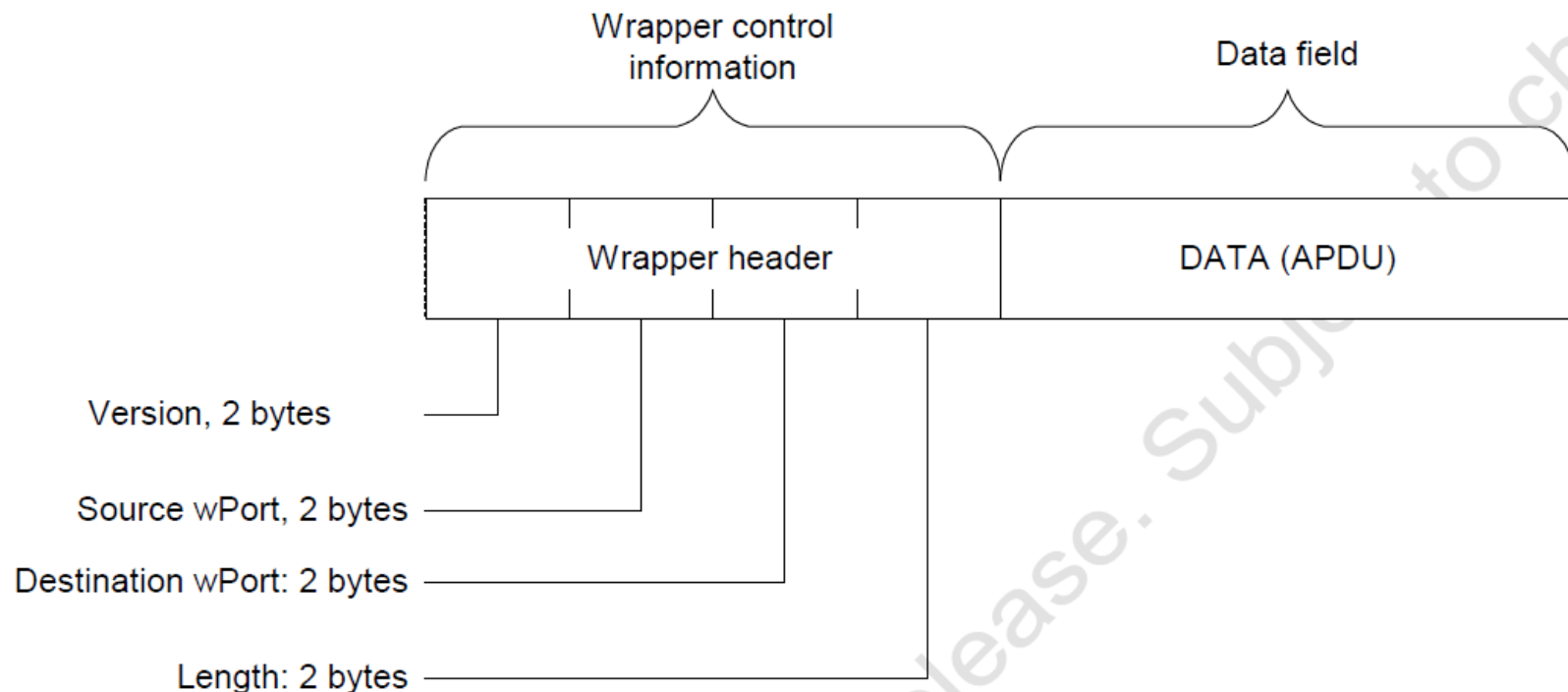
Its only roles are::

- Ensure source and destination DLMS/COSEM AE identification using the wPort numbers.
- And provide conversion between the OSI-style UDP-DATA.xxx service invocations and the SEND() and RECEIVE() interface functions provided by the standard UDP.

■ The wrapper protocol data unit (WPDU)

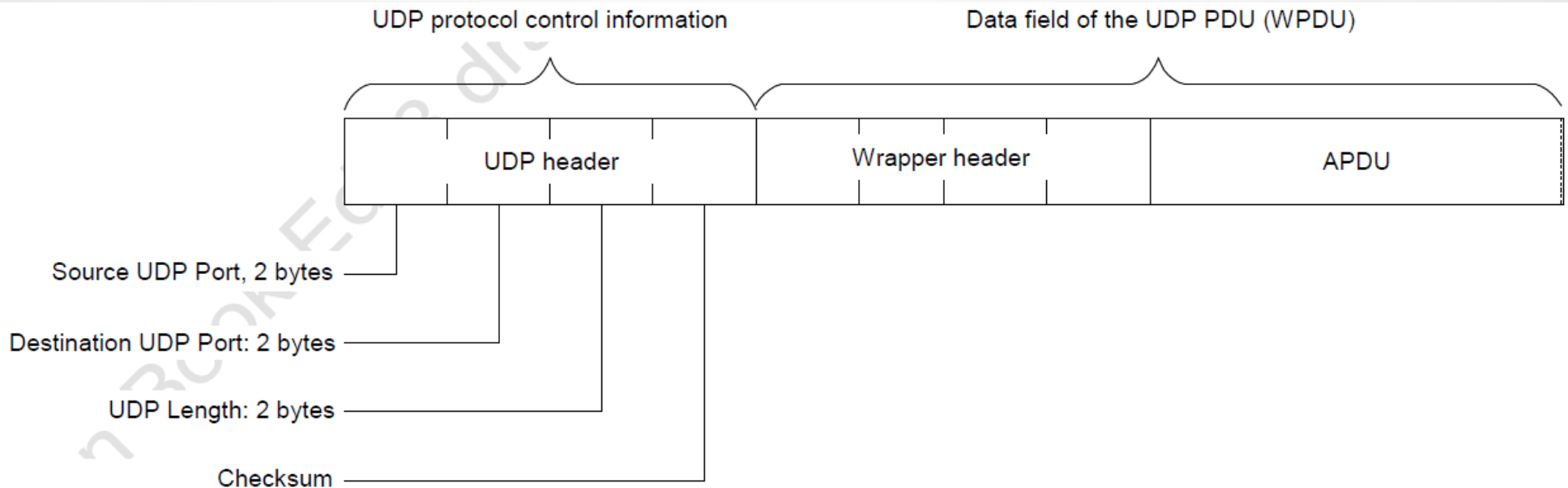
The WPDU consists of two parts:

- The wrapper header part, containing the wrapper control information.
- The data part, containing the DATA parameter – an xDLMS APDU – of the corresponding UDP-DATA.xxx service invocation.
- + The wrapper header includes four fields, see Figure. Each field is a 16 bit long unsigned integer value.



■ The DLMS/COSEM UDP-based transport layer protocol data unit

In this profile, WPDUs shall be transmitted in UDP Datagrams, specified in Internet standard STD0006. They shall encapsulate the WPDU, as shown in Figure



■ Reserved wrapper port numbers (wPorts)

Client side reserved addresses	
	Wrapper Port Number
No-station	0x0000
Client Management Process	0x0001
Public Client	0x0010
Server side reserved addresses	
	Wrapper Port Number
No-station	0x0000
Management Logical Device	0x0001
Reserved	0x0002..0x000F
All-station (Broadcast)	0x007F

■ Protocol state machine

- As the wrapper sublayer in this profile is state-less, for all other protocol related issues – protocol state machine, etc. – the governing rules are as they are specified in the Internet standard STD0006.
-
- The only supplementary rule is concerning discarding inappropriate messages: messages with an invalid destination wPort number
- There is no DLMS/COSEM AE in the receiving device bound to this wPort number – shall be discarded by the wrapper sublayer.

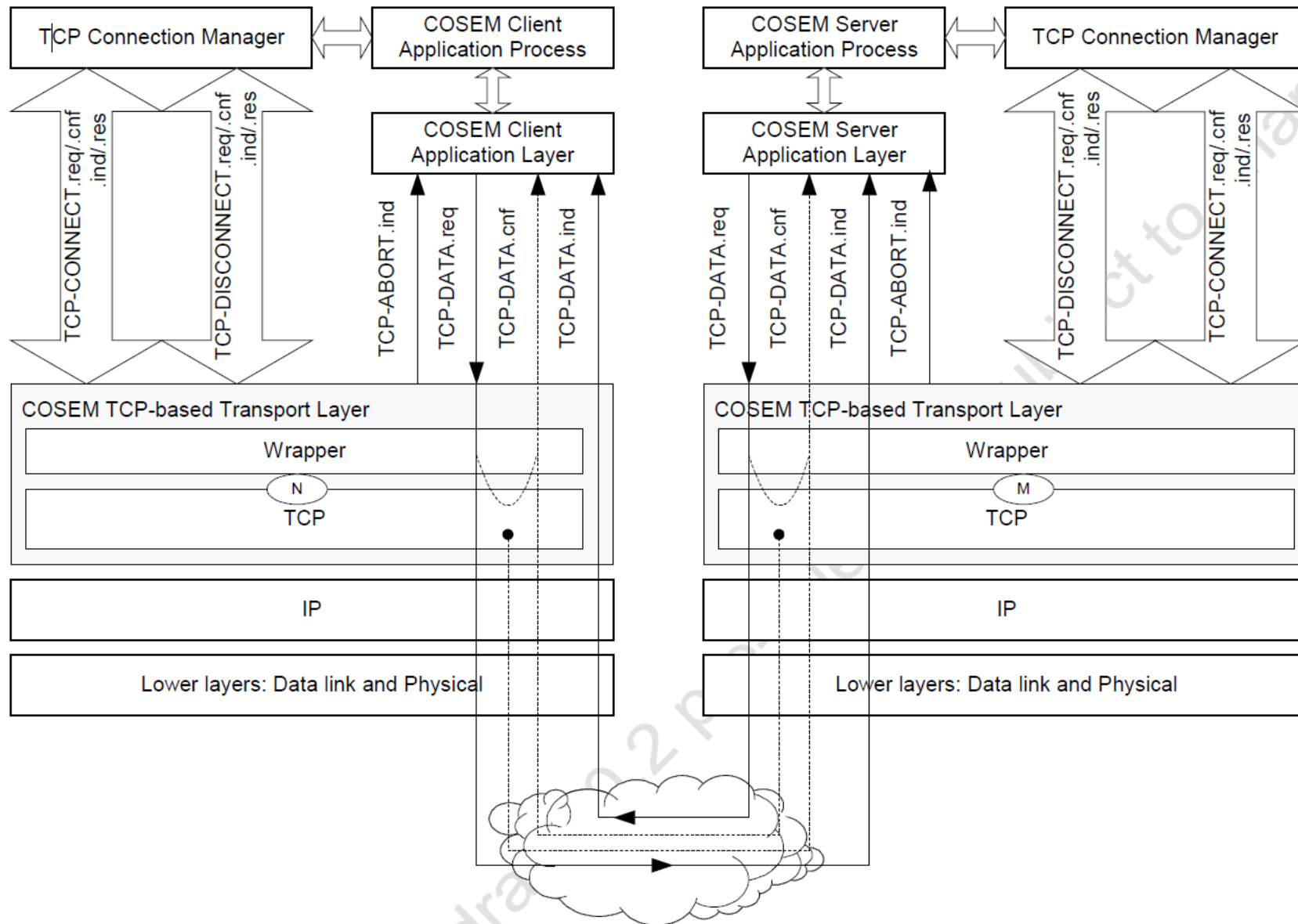
The DLMS/COSEM connection-oriented, TCP-based transport layer

- The DLMS/COSEM connection-oriented TL is based on the connection-oriented Internet transport protocol, called Transmission Control Protocol.
- TCP is an end-to-end reliable protocol.
- This reliability is ensured by a conceptual “virtual circuit”, using a method called PAR, Positive Acknowledgement with Retransmission.
- It provides acknowledged data delivery, error detection, data re-transmission after an acknowledgement time-out, etc. Therefore it deals with lost, delayed, duplicated or erroneous data packets. In addition, TCP offers an efficient flow control mechanism and full-duplex operation, too.

TCP, as a connection-oriented transfer protocol involves three phases: connection establishment, data exchange and connection release.

- Consequently, the DLMS/COSEM TCP-based TL provides OSI-style services to the service user(s) for all three phases:
- For the connection establishment phase, the TCP-CONNECT service is provided to the service user TCP connection manager process;
- For the data transfer phase, the TCP-DATA service is provided to the service user DLMS/COSEM AL;
- For the connection closing phase, the TCP-DISCONNECT service is provided to the service user TCP connection manager process;
- In addition, a TCP-ABORT service is provided to the service user DLMS/COSEM AL.

Service specification for the DLMS/COSEM TCP-based transport layer



The TCP-CONNECT service

TCP-CONNECT.request

Function

This primitive is the service request primitive for the connection establishment service.

Semantics of the service primitive

TCP-CONNECT.request

```
(  
Local_TCP_Port,  
Remote_TCP_Port,  
Remote_IP_Address  
)
```

Use:

The TCP-CONNECT.request primitive is invoked by the service user TCP connection manager process to establish a connection with the peer DLMS/COSEM TCP-based TL.

TCP-CONNECT.indication

Function

This primitive is the service indication primitive for the connection establishment service.

Semantics of the service primitive

TCP-CONNECT.indication

(
Local_TCP_Port,
Remote_TCP_Port,
Local_IP_Address,
Remote_IP_Address
)

Use

The TCP-CONNECT.indication primitive is generated by the DLMS/COSEM TCP-based TL following the reception of a TCP packet, indicating to the TCP connection manager process that a remote device is requesting a new TCP connection.

TCP-CONNECT.response

Function

This primitive is the service response primitive for the connection establishment service.

Semantics of the service primitive

TCP-CONNECT.response

(
Local_TCP_Port,
Remote_TCP_Port,
Local_IP_Address,
Remote_IP_Address,
Result
)

USE:

The TCP-CONNECT.response primitive is invoked by the TCP connection manager process to indicate to the DLMS/COSEM TCP-based TL whether the TCP connection requested previously has been accepted. The TCP connection manager cannot reject a requested connection.

TCP-CONNECT.confirm

Function

This primitive is the service confirm primitive for the connection establishment service.

Semantics of the service primitive

TCP-CONNECT.confirm

(
Local_TCP_Port,
Remote_TCP_Port,
Local_IP_Address,
Remote_IP_Address,
Result,
Reason_of_Failure
)

Use

The TCP-CONNECT.confirm primitive is generated by the DLMS/COSEM TCP-based TL to indicate to the service user TCP connection manager process the result of a TCP-CONNECT.request service invocation received previously.

TCP-DISCONNECT.request

Function

This primitive is the service request primitive for the connection termination service.

Semantics of the service primitive

TCP-DISCONNECT.request

(
Local_TCP_Port,
Remote_TCP_Port,
Local_IP_Address,
Remote_IP_Address
)

Use

The TCP-DISCONNECT.request primitive is invoked by the service user TCP connection manager process to request the disconnection of an existing TCP connection.

TCP-DISCONNECT.indication

Function

This primitive is the service indication primitive for the connection termination service.

Semantics of the service primitive

TCP-DISCONNECT.indication

(
Local_TCP_Port,
Remote_TCP_Port,
Local_IP_Address,
Remote_IP_Address,
Reason
)

Use

The TCP-DISCONNECT.indication primitive is generated by the DLMS/COSEM TCP-based TL to the service user TCP connection manager process to indicate that the peer entity has requested the disconnection of an existing TCP connection.

TCP-DISCONNECT.response

Function

This primitive is the service response primitive for the connection termination service.

TCP-DISCONNECT.response

```
(  
  Local_TCP_Port,  
  Remote_TCP_Port,  
  Local_IP_Address,  
  Remote_IP_Address,  
  Result  
)
```

Use

The TCP-DISCONNECT.response primitive is invoked by the TCP connection manager process to indicate to the DLMS/COSEM TCP-based TL whether the previously requested TCP disconnection is accepted.

The TCP connection manager process cannot reject the requested disconnection. This service primitive is invoked only if the corresponding TCP-DISCONNECT.indication service indicated a remotely initiated disconnection request (Reason == REMOTE_REQ).

TCP-DISCONNECT.confirm

Function

This primitive is the service confirm primitive for the connection termination service.

Semantics of the service primitive

TCP-DISCONNECT.confirm

(
Local_TCP_Port,
Remote_TCP_Port
Local_IP_Address,
Remote_IP_Address,
Result,
Reason_of_Failure
)

Use

The TCP-DISCONNECT.confirm primitive is invoked by the DLMS/COSEM TCP-based TL to confirm to the service user TCP connection manager the result of a previous TCP-DISCONNECT.request service invocation.

■The TCP-ABORT service

TCP-ABORT.indication

Function

This primitive is the service indication primitive for the connection termination service.

Semantics of the service primitive

TCP-ABORT.indication

(
Local_TCP_Port,
Remote_TCP_Port,
Local_IP_Address,
Remote_IP_Address,
Reason
)

Use

The TCP-ABORT.indication primitive is generated by the DLMS/COSEM TCP-based TL to indicate to the service user DLMS/COSEM AL a non-solicited disruption of the supporting TCP connection.

When this indication is received, the DLMS/COSEM AL shall release all AAs established using this TCP connection, and shall indicate this to the COSEM AP using the COSEM-ABORT.indication service primitive. .

■ The TCP-DATA service

TCP-DATA.request

Function

This primitive is the service request primitive for the connection mode data transfer service.

Semantics of the service primitive

```
TCP-DATA.request  
(  
  Local_wPort,  
  Remote_wPort,  
  Local_TCP_Port,  
  Remote_TCP_Port,  
  Local_IP_Address,  
  Remote_IP_Address,  
  Data_Length,  
  Data  
)
```

Use

The TCP-DATA.request primitive is invoked by either the client or the server DLMS/COSEM AL to request sending an APDU to a single peer application.

The reception of this primitive shall cause the wrapper sublayer to pre-fix the wrapper-specific fields (Local_wPort, Remote_wPort and the Data_Length) to the xDLMS APDU received, and then to call the SEND() function of the TCP sublayer with the properly formed WPDU, as DATA.

TCP-DATA.indication

Function

This primitive is the service request primitive for the connection mode data transfer service.

Semantics of the service primitive :

TCP-DATA.indication

(
Local_wPort,
Remote_wPort,
Local_TCP_Port,
Remote_TCP_Port,
Local_IP_Address,
Remote_IP_Address,
Data_Length,
Data
)

Use

The TCP-DATA.indication primitive is generated by the DLMS/COSEM TL to indicate to the service user DLMS/COSEM AL that an xDLMS APDU has been received from a remote device.

It is generated following the reception of a complete APDU (in one or more TCP packets) by the DLMS/COSEM TCP-based TL, if both the Local_TCP_Port and Local_wPort parameters in the TCP packet(s) carrying the APDU contain valid port numbers, meaning that there is a DLMS/COSEM AE in the receiving device bound to the given port numbers. Otherwise, the message received shall simply be discarded.

TCP-DATA.confirm

Function

This primitive is the optional service confirm primitive for the connection mode data transfer service.

Semantics of the service primitive

TCP-DATA.confirm

(
Local_wPort,
Remote_wPort,
Local_TCP_Port,
Remote_TCP_Port,
Local_IP_Address,
Remote_IP_Address,
Confirmation_Type,
Result
)

Use

The TCP-DATA.confirm primitive is optional. If implemented, it is generated by the DLMS/COSEM TL to confirm to the service user DLMS/COSEM AL the result of the execution of the previous .request

Protocol specification for the DLMS/COSEM TCP-based transport layer

The DLMS/COSEM CO, TCP-based TL includes the Internet standard TCP layer as specified in STD0007, and the DLMS/COSEM-specific wrapper sublayer.

- In the TCP-based TL the wrapper sublayer is more complex than in the UDP-based TL.
- Its main role is also to ensure source and destination DLMS/COSEM AE identification using the wPort numbers, and to convert OSI-style TCP-DATA service primitives to and from the SEND() and RECEIVE() interface functions provided by the standard TCP.
- On the other hand, the wrapper sublayer in the TCP-based TL has also the task to help the service user DLMS/COSEM ALs to exchange complete APDUs.
- TCP is a “streaming” protocol.
- It is the responsibility the wrapper sublayer to know how much data had to be sent / received, to keep track how much has been actually sent / received, and repeat the operation until the complete APDU is transmitted.
- The wrapper sublayer in the TCP-based DLMS/COSEM TL is not a state-less entity:

The wrapper protocol data unit (WPDU)

- The wrapper protocol data unit is as it is specified in UDP TL.

■Reserved wrapper port numbers

- Reserved wPort Numbers are specified in UDP TL Table

Unit

0	4	8	10	16	24	31
Source TCP Port				Destination TCP Port		
Sequence Number						
Acknowledgement Number						
Data Offset	Reserved		Flags	Window		
Checksum				Urgent Pointer		
TCP Options					Padding	
Data (part of the WPDU)						
...Data (part of the WPDU)...						

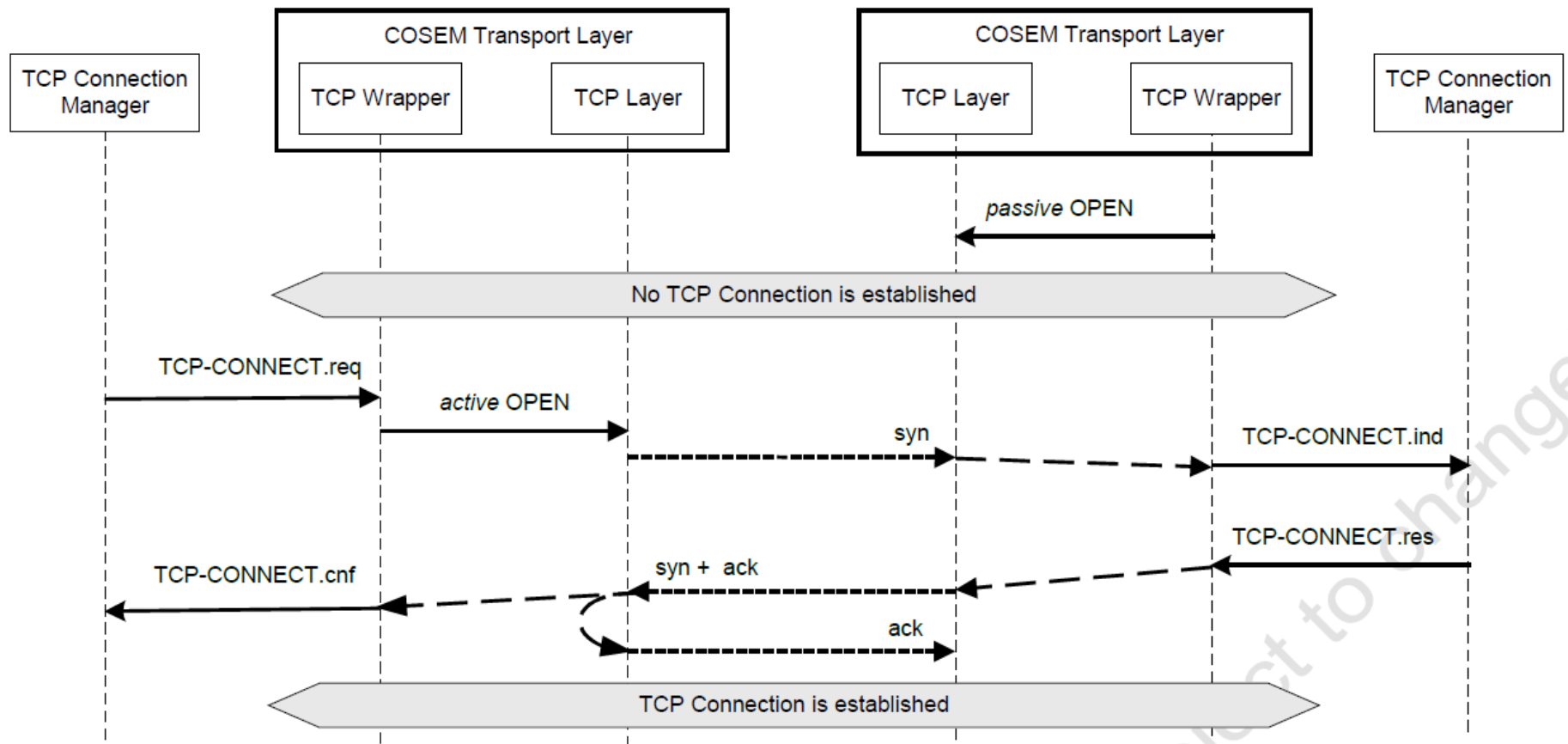
Control Bits: As mentioned, TCP does not use a separate format for control messages. Instead, certain bits are set to indicate the communication of control information. The six bits are:

Subfield Name	Size (bytes)	Description
URG	1/8 (1 bit)	Urgent Bit: When set to 1, indicates that the priority data transfer feature has been invoked for this segment, and that the <i>Urgent Pointer</i> field is valid.
ACK	1/8 (1 bit)	Acknowledgment Bit: When set to 1, indicates that this segment is carrying an acknowledgment, and the value of the <i>Acknowledgment Number</i> field is valid and carrying the next sequence expected from the destination of this segment.
PSH	1/8 (1 bit)	Push Bit: The sender of this segment is using the TCP push feature , requesting that the data in this segment be immediately pushed to the application on the receiving device.
RST	1/8 (1 bit)	Reset Bit: The sender has encountered a problem and wants to reset the connection .
SYN	1/8 (1 bit)	Synchronize Bit: This segment is a request to synchronize sequence numbers and establish a connection ; the <i>Sequence Number</i> field contains the initial sequence number (ISN) of the sender of the segment.
FIN	1/8 (1 bit)	Finish Bit: The sender of the segment is requesting that the connection be closed .

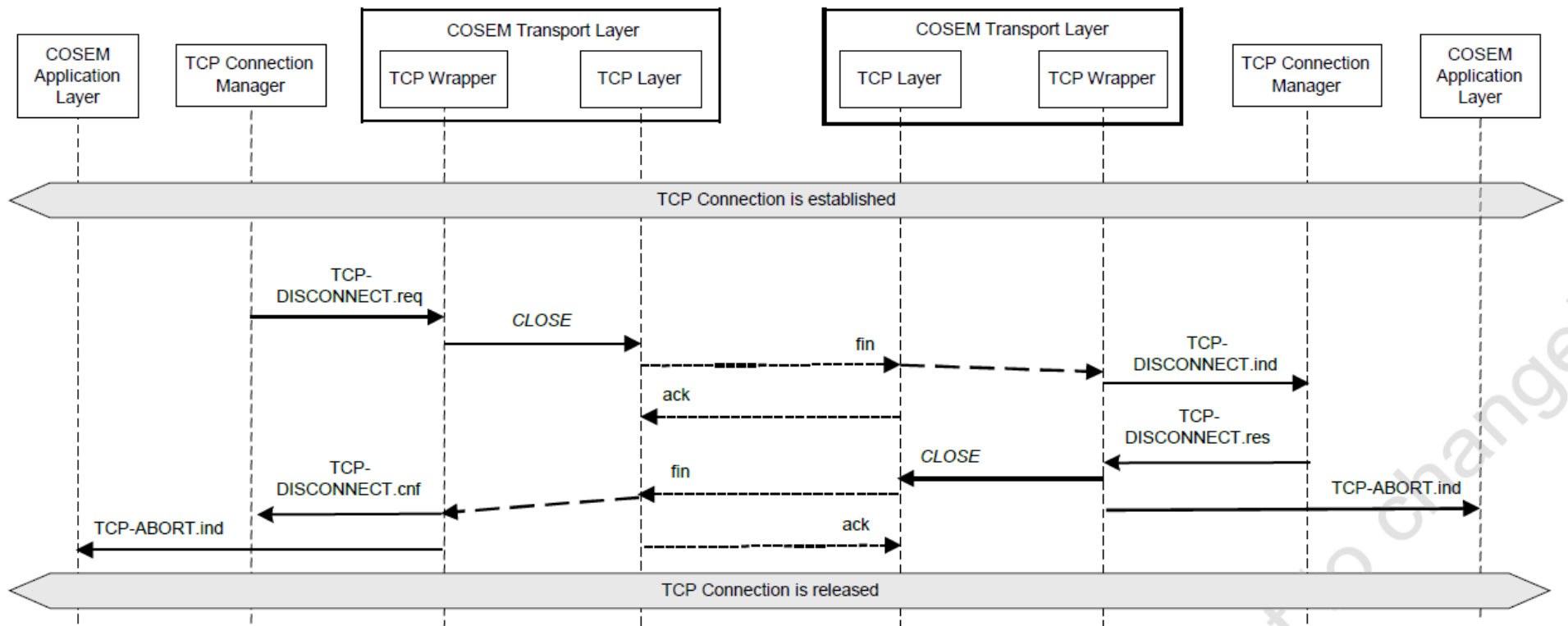
Options: TCP includes a generic mechanism for including one or more sets of optional data in a TCP segment. Each of the options can be either one byte in length or variable in length. The first byte is the *Option-Kind* subfield, and its value specifies the type of option, which in turn indicates whether the option is just a single byte or multiple bytes. Options that are many bytes consist of three fields:

Subfield Name	Size (bytes)	Description
<i>Option-Kind</i>	1	Option-Kind: Specifies the option type.
<i>Option-Length</i>	1	Option-Length: The length of the entire option in bytes, including the <i>Option-Kind</i> and <i>Option-Length</i> fields.
<i>Option-Data</i>	Variable	Option-Data: The option data itself. In at least one oddball case, this field is omitted (making <i>Option-Length</i> equal to 2).

Definition of the procedures TCP connection establishment



TCP disconnection

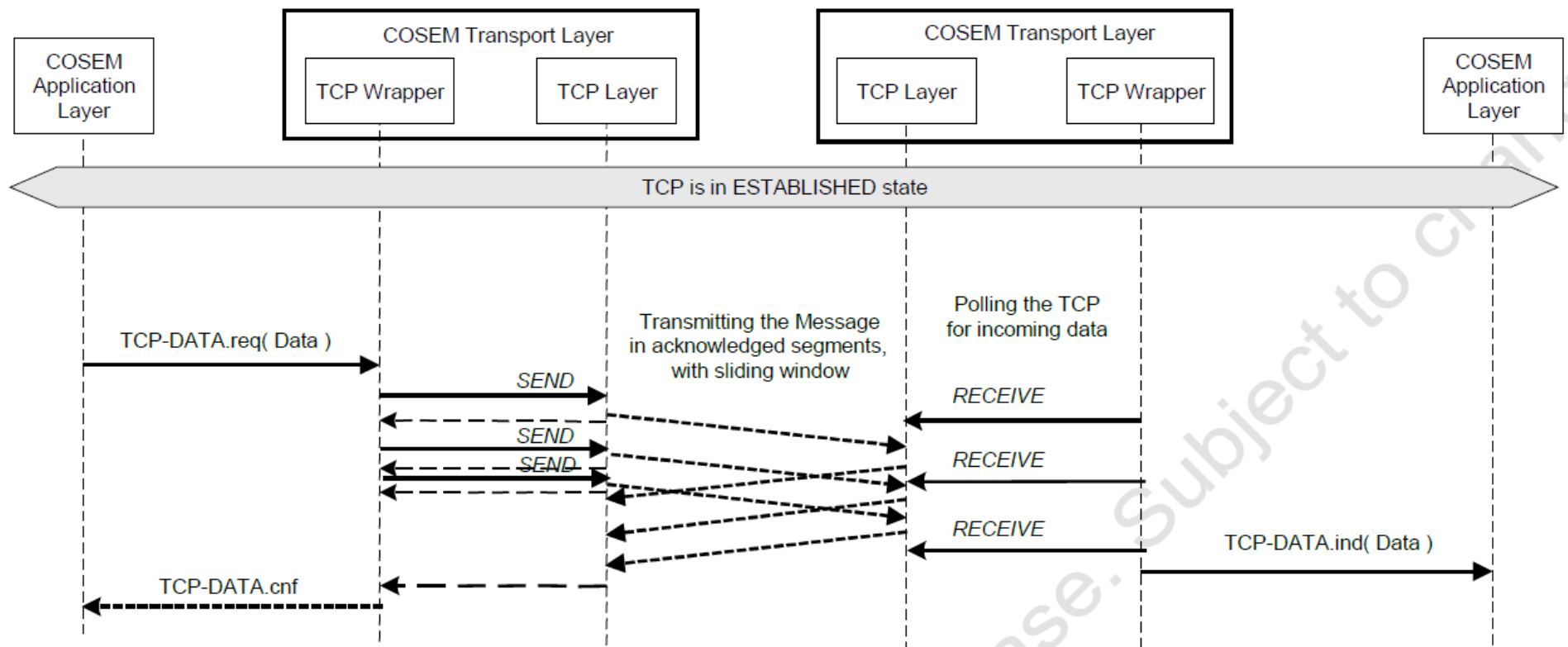


TCP connection abort

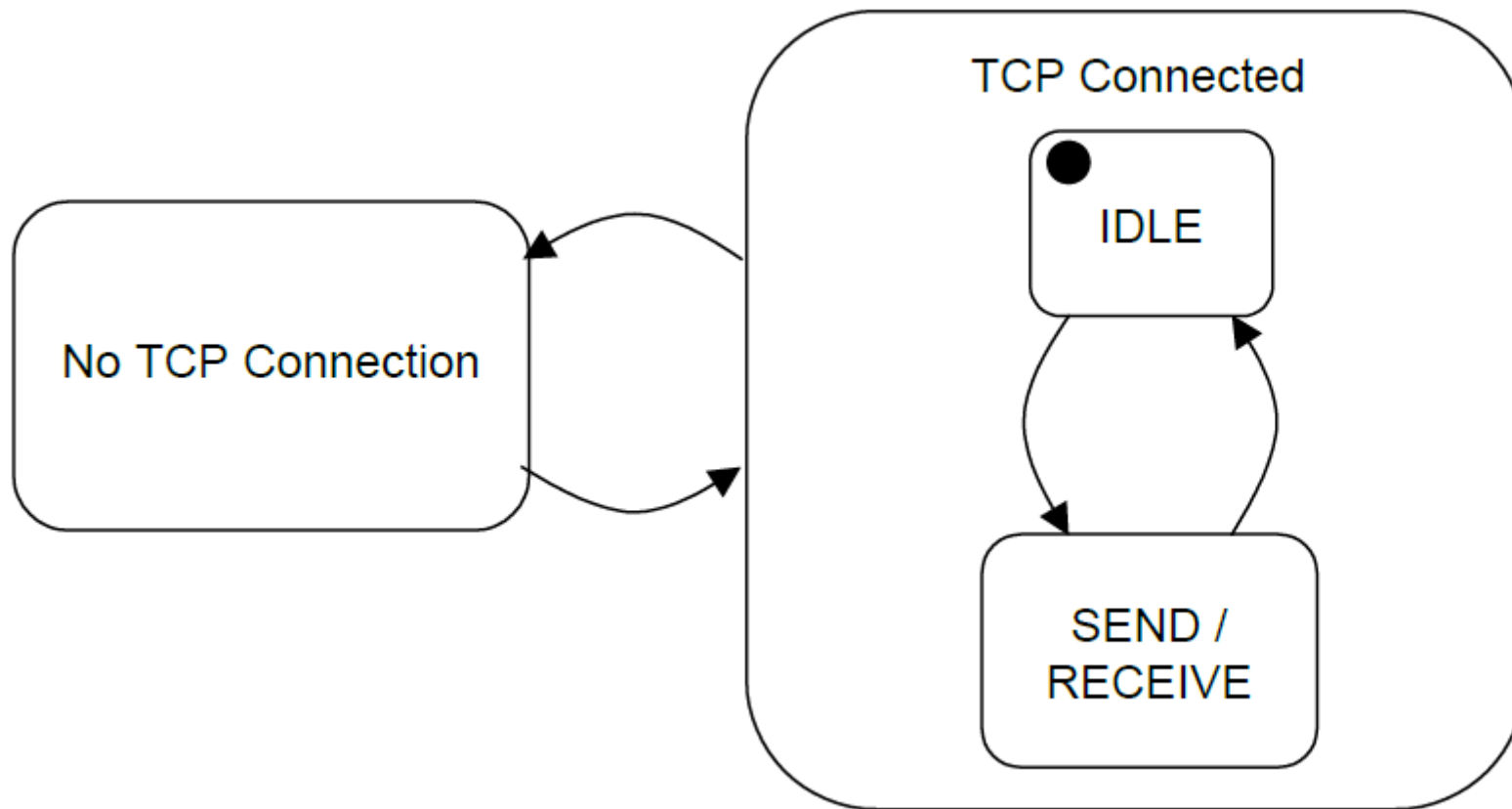
The DLMS/COSEM TCP-based TL indicates the disruption or disconnection of the supporting TCP connection to the DLMS/COSEM AL with the help of the TCP-ABORT.indication primitive.

- This is the only TCP connection management service provided to the DLMS/COSEM AL.
- The service is invoked either when the TCP connection is disconnected by the TCP connection manager process or when the TCP disconnection occurs in a non-solicited manner, for example the TCP sublayer is detecting a non-resolvable error or the physical connection is shut down.
- The purpose of this service is to inform the DLMS/COSEM AL about the disruption of the TCP connection, so that it could release all existing AAs.

Data transfer using the COSEM TCP-based transport layer



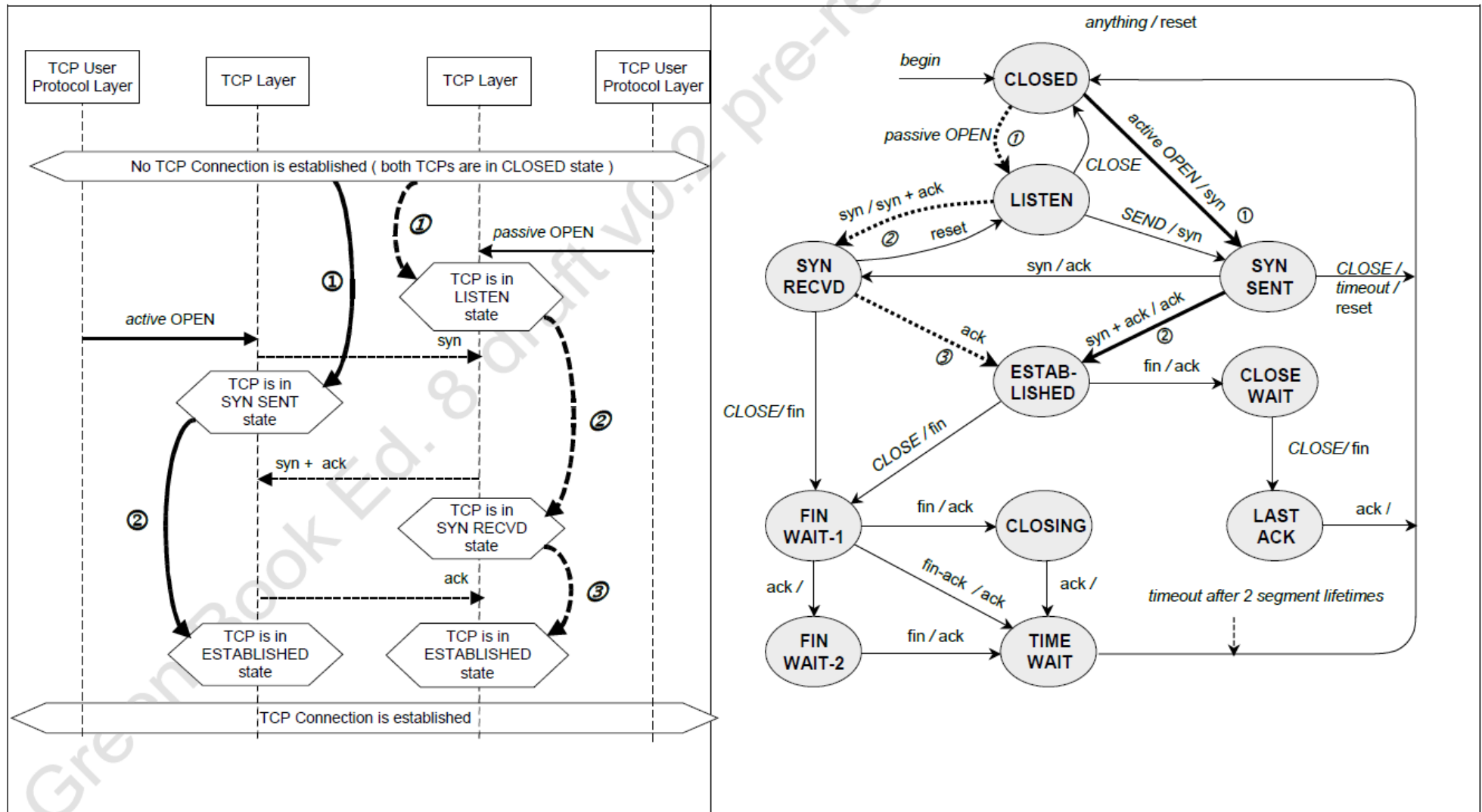
High-level state transition diagram of the wrapper sublayer



Converting OSI-style TL services to and from RFC-style TCP function calls

- As specified in STD0007, a TCP connection is established by calling the OPEN function. This 2658 function can be called in *active* or *passive* manner.
- According to the TCP connection state diagram (Figure 34) a *passive* OPEN takes the caller device to the LISTEN state, waiting for a connection request from any remote TCP and port.
- An *active* OPEN call makes the TCP to establish the connection to a remote TCP.
- The establishment of a TCP Connection is performed by using the so-called “Three-way handshake” procedure. This is initiated by one TCP calling an *active* OPEN and responded by another TCP, the one, which has already been called a *passive* OPEN and consequently is in the LISTEN state
- The message sequence – and the state transitions corresponding to that message exchange – for this “three-way handshake” procedure are shown in Figure
- This process, consisting of three messages, establishes the TCP connection and “synchronizes” the initial sequence numbers at both sides. This mechanism has been carefully designed to guarantee, that both sides are ready to transmit data and know that the other side is ready to transmit as well.
- Note that the procedure also works if two TCPs simultaneously initiate the procedure.

MSC and state transitions for establishing a transport layer and TCP connection

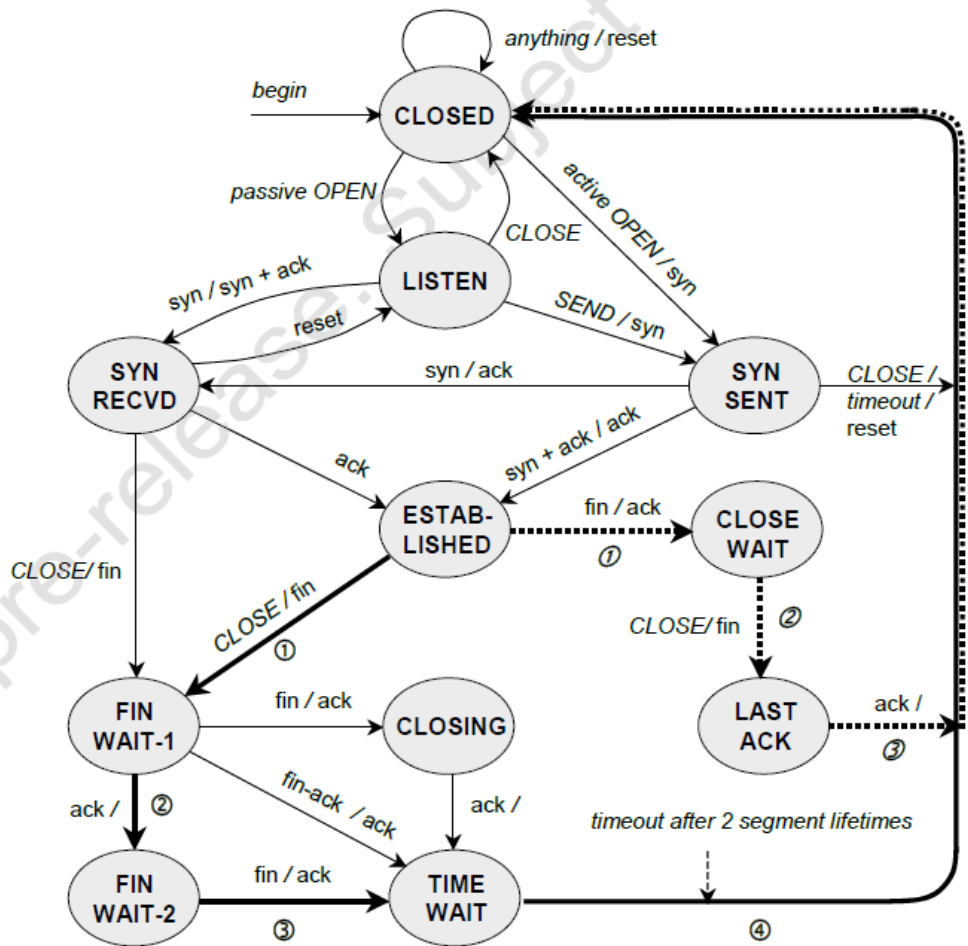
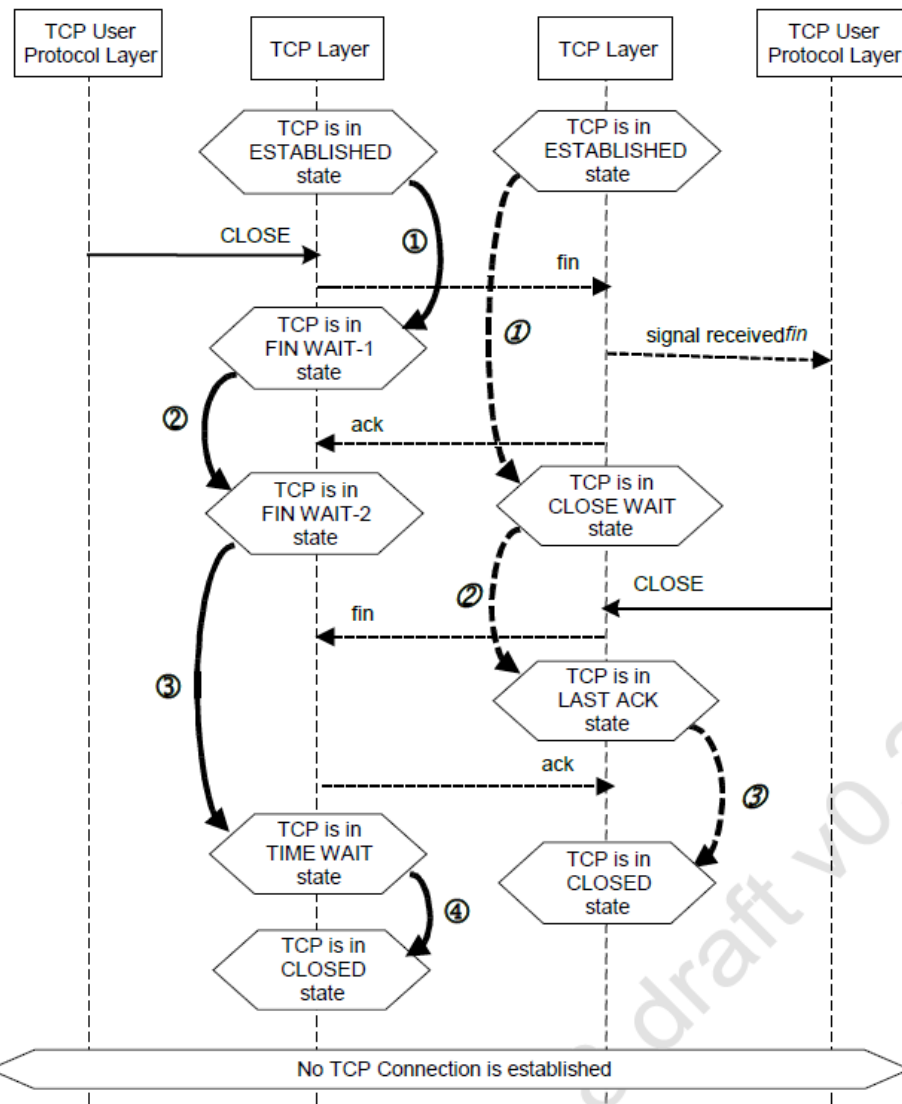


NOTE In the case of the COSEM transport layer, the TCP user protocol layer is the wrapper sublayer.

Closing a transport layer and a TCP connection

- Closing a TCP connection is done by calling the CLOSE function, generally when there is no more data to be sent.
- Upon the invocation of the TCP-DISCONNECT.request service primitive by the TCP connection manager process, the wrapper sublayer invokes the CLOSE function of the TCP sublayer.
- However, as the TCP connection is full duplex, the other side may still have data to send. Therefore, after calling the CLOSE function, the TCP-based transport later may continue to receive data and send it to the DLMS/COSEM AL, until it is told that the other side has CLOSED, too.
- At this point it generates the COSEM-ABORT.indication primitive, and all AAs are released.
- The message sequence chart and the state transitions corresponding to a successful TCP connection release are shown in Figure
- NOTE In the case of the DLMS/COSEM TL, the TCP user protocol layer is the wrapper sublayer.

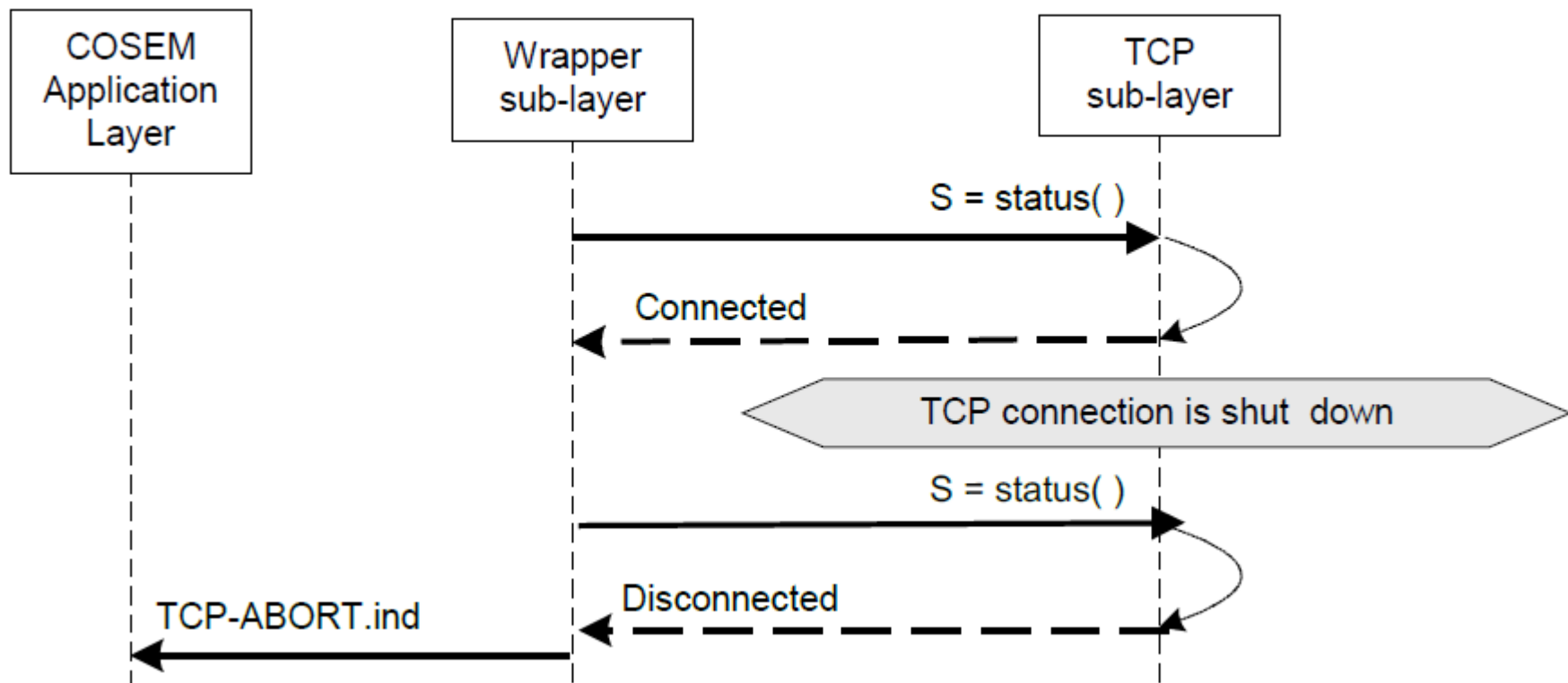
MSC and state transitions for closing a transport layer and TCP connection



Polling the TCP sublayer for TCP abort indication

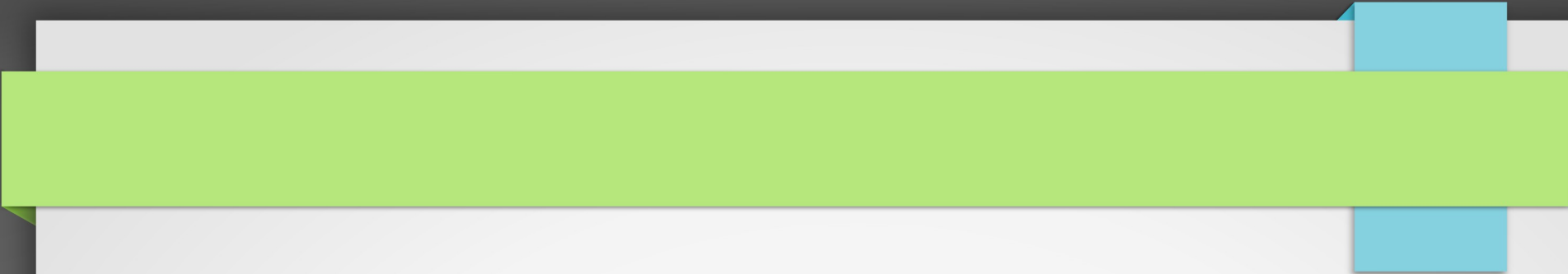
TCP connection abort

STD0007 does not specify a standard function to indicate an unexpected abort at TCP level. However, it can be detected by the TCP user entity by polling the status of the TCP with the STATUS() function, as shown in Figure

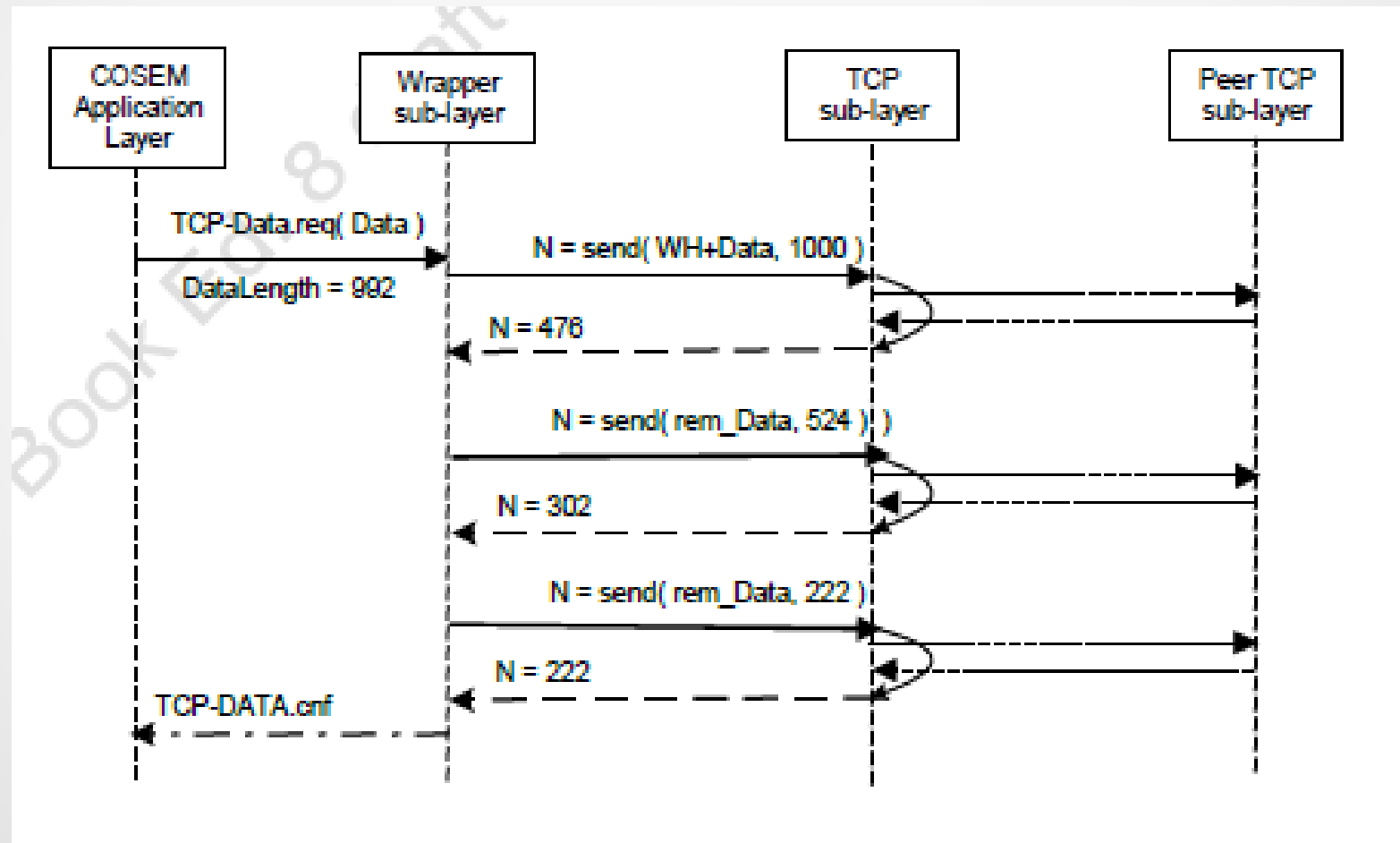


Data transfer using the TCP-DATA service

- To send an APDU to the peer, the DLMS/COSEM AL simply invokes the TCP-DATA.request primitive of the DLMS/COSEM TCP-based TL.
- Also, when a complete APDU is received, this is indicated to the COSEM AL with the help of the TCP-DATA.indication primitive. Thus, for the AL the TL behaves as if it would transport the whole APDU in one piece.
- However, as TCP is a streaming protocol, not preserving data boundaries, nothing ensures that an APDU is actually transmitted in one TCP packet. As already mentioned in the DLMS/COSEM TCP-based TL it is the responsibility of the wrapper sublayer to “hide” the streaming nature of the TCP sublayer.
- The following example illustrates how the wrapper sublayer accomplishes this task. Let's suppose that an AL entity wants to send an APDU containing 992 bytes via the DLMS/COSEM TCP-based TL. It invokes the TCP-DATA.request service with this APDU as the DATA service parameter as shown in Figure .

- 
- Upon the reception of this service invocation, the wrapper sublayer constructs the WPDU:
 - it pre-fixes the APDU with the wrapper header (WH), including the local and remote wPort numbers and the APDU length.
 - It calls then the SEND() function of the TCP sublayer, requesting to send the WPDU, which is now 1000 bytes long: 8 bytes of wrapper header plus 992 bytes of APDU.
 - The SEND() function returns with the number of bytes sent or an error (a negative value). Let's suppose that no error occurs and the SEND() function successfully returns with the value 476.
 - This number is the number of bytes sent. This also illustrates the meaning of the “streaming” nature of the TCP: in fact, the SEND() function returns with success even if the number of bytes sent is less than the number of bytes requested to be sent.

Sending an APDU in three TCP packets



- From the value returned, the wrapper knows that not the whole WPDU has been sent. It calls the SEND() function again, with the remaining part of the WPDU and so on, until the complete WPDU is sent.
- As already mentioned in depending on the implementation, the successful return of the SEND() function may even not mean that something has been really sent to the network.
- It may mean only that the protocol implementation took and buffered the data. It may happen that the protocol implementation delays the transmission to comply with protocol conventions or network traffic related algorithms.
- On the receiving side, it is also the responsibility of the wrapper sublayer to assemble the complete APDU before invoking the TCP-DATA.indication primitive.
- This is possible by using the length bytes of the WPDU header. The wrapper repeats RECEIVE() calls until the number of bytes, indicated in the WPDU header is received.
- All these SEND() and RECEIVE() calls are internal to the DLMS/COSEM TL. The service user DLMS/COSEM AL simply uses the TCP-DATA services, and observes a reliable data transfer service preserving the data boundaries of the APDUs

Receiving the message in several packets

