

## 一、overview

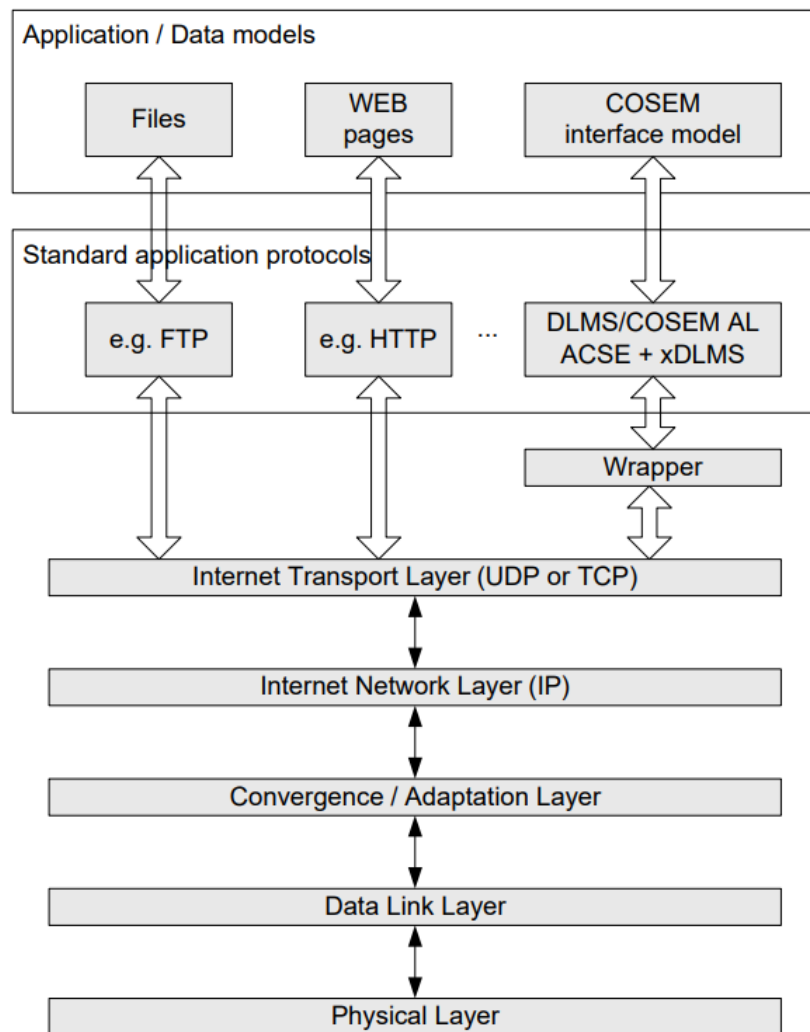
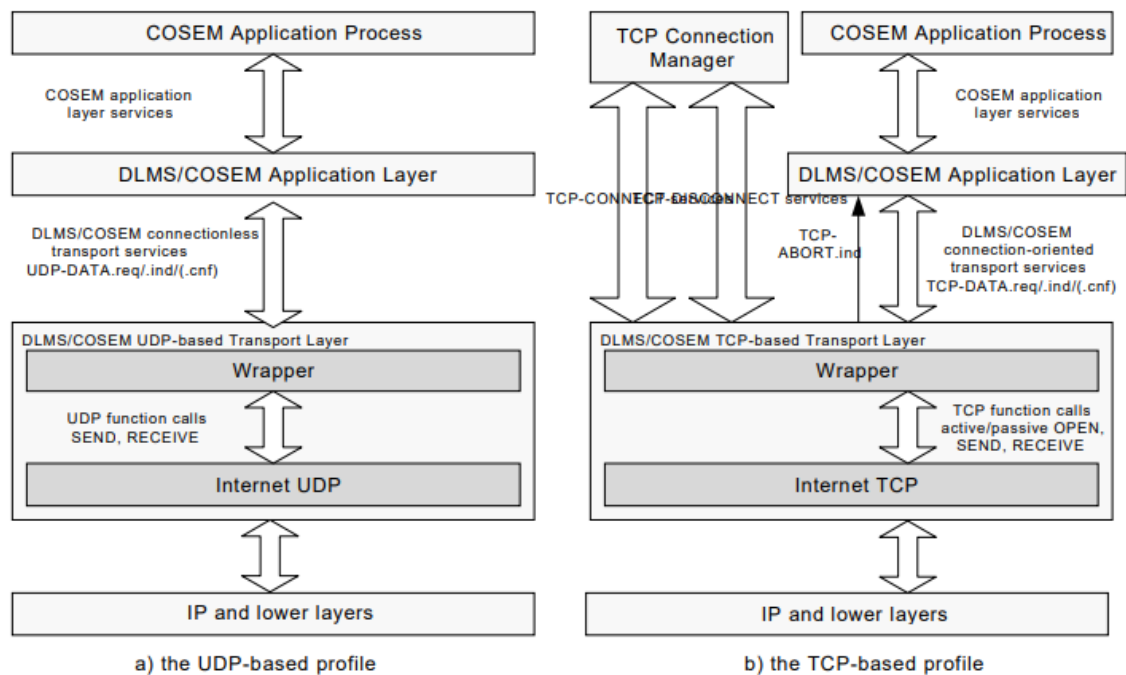


Figure 26 – DLMS/COSEM as a standard Internet application protocol

## 二、TCP、UDP区别

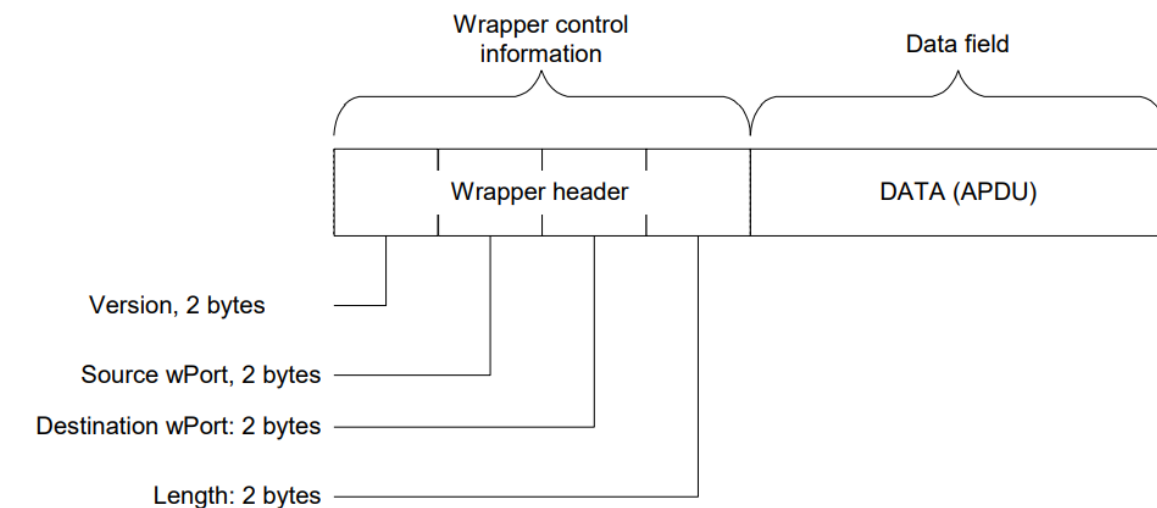


**Figure 27 – Transport layers of the DLMS/COSEM\_on\_IP profile**

区别：

	TCP	UDP
连接方面	面向连接	无连接，即发送数据之前不需要建立连接
安全方面	提供可靠的服务，通过TCP连接传送的数据，无差错，不丢失，不重复，且按序到达	尽最大努力交付，即不保证可靠交付
传输效率	传输效率相对较低	传输效率高，适用于对高速传输和实时性有较高的通信或广播通信
连接对象数量	连接只能是点到点，一对一的	支持一对一，一对多和多对多的交互通信

### 三、WPDU



NOTE The maximum length of the APDU should be eight bytes less than the maximum length of the UDP datagram.

- **Version:** carries the version of the wrapper. Its value is controlled by the DLMS UA. The current value is 0x0001. Note, that in later versions the wrapper header may have a different structure;
- **Source wPort:** carries the wPort number identifying the sending DLMS/COSEM AE;
- **Destination wPort:** carries the wPort number identifying the receiving DLMS/COSEM AE;
- **Data length:** indicates the length of the DATA field of the WPDU (the xDLMS APDU transported).

	Destination wPort	Source wPort
p2P通信	固定为1	客户端id
DCU透传链表	sapId (大于1)	客户端id
DCU连接	固定为1	客户端id

## 四、UDP-PDU

- 报文结构图

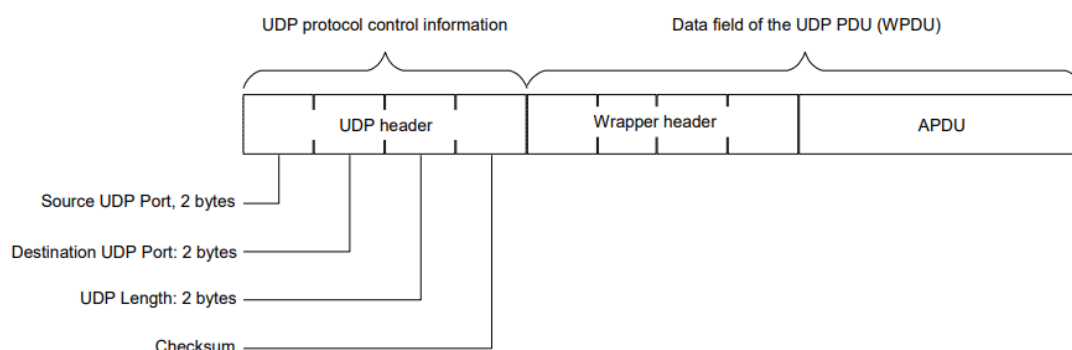


Figure 30 – The DLMS/COSEM connection-less, UDP-based transport layer PDU (UDP-PDU)

注：UDP项目使用较少，暂无实际报文，后续遇到再补。

## 五、TCP-PDU

### • 报文结构图

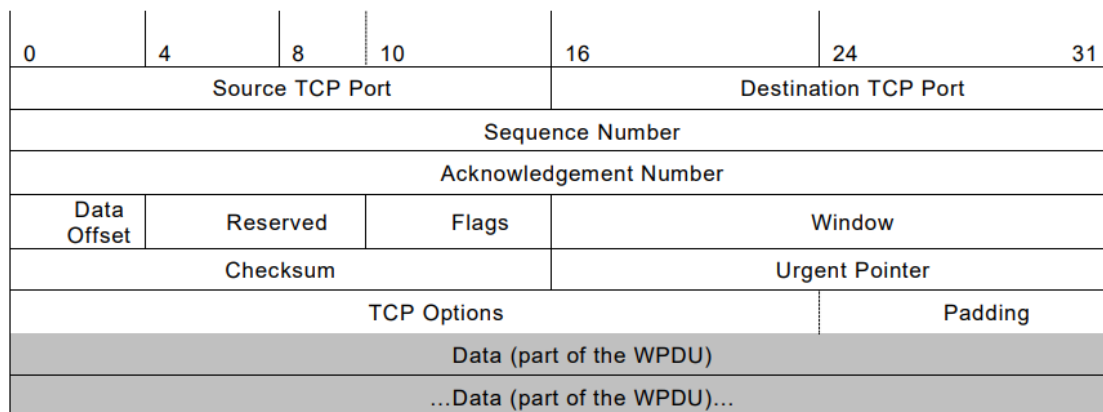


Figure 32 – The TCP packet format

```
> Frame 5674: 130 bytes on wire (1040 bits), 130 bytes captured (1040 bits) on interface \Device\NPF_{62609AF1-9BB0-42F3-8323-3765B0942E9C}, id 0
> Ethernet II, Src: e0:be:03:2d:34:c1 (e0:be:03:2d:34:c1), Dst: Shenzhen_50:ff:9f (00:1c:55:50:ff:9f)
> Internet Protocol Version 4, Src: 192.168.140.20, Dst: 192.168.140.235
> Transmission Control Protocol, Src Port: 54477, Dst Port: 4059, Seq: 1, Ack: 1, Len: 76
▼ Data (76 bytes)
  Data: 00010001000100446042a109060760857405080101a60a04...
  [Length: 76]
```

### • TCP连接

三次握手：

第一次握手：

客户端将TCP报文标志位SYN置为1，随机产生一个序号值seq=J，保存在TCP首部的序列号(Sequence Number)字段里，指明客户端打算连接的服务器的端口，并将该数据包发送给服务器端，发送完毕后，客户端进入SYN\_SENT状态，等待服务器端确认。

第二次握手：

服务器端收到数据包后由标志位SYN=1知道客户端请求建立连接，服务器端将TCP报文标志位SYN和ACK都置为1，ack=J+1，随机产生一个序号值seq=K，并将该数据包发送给客户端以确认连接请求，服务器端进入SYN\_RCVD状态。

第三次握手：

客户端收到确认后，检查ack是否为J+1，ACK是否为1，如果正确则将标志位ACK置为1，ack=K+1，并将该数据包发送给服务器端，服务器端检查ack是否为K+1，ACK是否为1，如果正确则连接建立成功，客户端和服务器端进入ESTABLISHED状态，完成三次握手，随后客户端与服务器端之间可以开始传输数据了。

5670	2022-02-17 15:02:05.588970	192.168.140.20	192.168.140.235	TCP	66 54477 → 4059 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1
5671	2022-02-17 15:02:05.590300	192.168.140.235	192.168.140.20	TCP	66 4059 → 54477 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460 SACK_PERM=1 WS=64
5672	2022-02-17 15:02:05.590464	192.168.140.20	192.168.140.235	TCP	54 54477 → 4059 [ACK] Seq=1 Ack=1 Win=131328 Len=0
5674	2022-02-17 15:02:05.653377	192.168.140.20	192.168.140.235	TCP	130 54477 → 4059 [PSH, ACK] Seq=1 Ack=1 Win=131328 Len=76
5675	2022-02-17 15:02:05.654696	192.168.140.235	192.168.140.20	TCP	60 4059 → 54477 [ACK] Seq=1 Ack=77 Win=64192 Len=0
5677	2022-02-17 15:02:05.725688	192.168.140.235	192.168.140.20	TCP	105 4059 → 54477 [PSH, ACK] Seq=1 Ack=77 Win=64192 Len=51
5679	2022-02-17 15:02:05.766155	192.168.140.20	192.168.140.235	TCP	54 54477 → 4059 [ACK] Seq=77 Ack=52 Win=131328 Len=0
6156	2022-02-17 15:02:31.247588	192.168.140.20	192.168.140.235	CLASSI..	75 Message: Binding Request
6157	2022-02-17 15:02:31.248899	192.168.140.235	192.168.140.20	TCP	60 4059 → 54477 [ACK] Seq=52 Ack=98 Win=64192 Len=0
6158	2022-02-17 15:02:31.280581	192.168.140.235	192.168.140.20	TCP	197 4059 → 54477 [PSH, ACK] Seq=52 Ack=98 Win=64192 Len=143
6160	2022-02-17 15:02:31.321852	192.168.140.20	192.168.140.235	TCP	54 54477 → 4059 [ACK] Seq=98 Ack=195 Win=131072 Len=0
6185	2022-02-17 15:02:32.483405	192.168.140.20	192.168.140.235	CLASSI..	75 Message: Binding Request
6186	2022-02-17 15:02:32.484721	192.168.140.235	192.168.140.20	TCP	60 4059 → 54477 [ACK] Seq=195 Ack=119 Win=64192 Len=0
6187	2022-02-17 15:02:32.490743	192.168.140.235	192.168.140.20	TCP	197 4059 → 54477 [PSH, ACK] Seq=195 Ack=119 Win=64192 Len=143
6188	2022-02-17 15:02:32.533562	192.168.140.20	192.168.140.235	TCP	54 54477 → 4059 [ACK] Seq=119 Ack=338 Win=130816 Len=0
7915	2022-02-17 15:03:44.097646	192.168.140.20	192.168.140.235	TCP	85 54477 → 4059 [PSH, ACK] Seq=119 Ack=338 Win=130816 Len=31
7916	2022-02-17 15:03:44.098983	192.168.140.235	192.168.140.20	TCP	60 4059 → 54477 [ACK] Seq=338 Ack=150 Win=64192 Len=0
7917	2022-02-17 15:03:44.100472	192.168.140.235	192.168.140.20	TCP	85 4059 → 54477 [PSH, ACK] Seq=338 Ack=150 Win=64192 Len=31
7919	2022-02-17 15:03:44.141222	192.168.140.20	192.168.140.235	TCP	54 54477 → 4059 [ACK] Seq=150 Ack=369 Win=130816 Len=0

四次挥手：

第一次挥手：Client端发起挥手请求，向Server端发送标志位是FIN报文段，设置序列号seq，此时，Client端进入FIN\_WAIT\_1状态，这表示Client端没有数据要发送给Server端了。

第二次挥手：Server端收到了Client端发送的FIN报文段，向Client端返回一个标志位是ACK的报文段，ack设为seq+1，Client端进入FIN\_WAIT\_2状态，Server端告诉Client端，我确认并同意你的关闭请求。

第三次挥手：Server端向Client端发送标志位是FIN的报文段，请求关闭连接，同时Client端进入LAST\_ACK状态。

第四次挥手：Client端收到Server端发送的FIN报文段，向Server端发送标志位是ACK的报文段，然后Client端进入TIME\_WAIT状态。Server端收到Client端的ACK报文段以后，就关闭连接。此时，Client端等待2MSL的时间后依然没有收到回复，则证明Server端已正常关闭，那好，Client端也可以关闭连接了。

因为开启了延时ack机制，导致收到第一个fin之后，发送ack的条件不能满足立即发送ack的条件，导致ack的发送被延时了，在延时的过程中，应用如果确认没数据要发，并且也要关闭此连接的情况下，会触发发送fin，这个fin就会和之前的ack合并被发出

66639	2022-02-17 15:39:31.343766	192.168.140.235	192.168.140.20	TCP	105 4059 → 62580 [PSH, ACK] Seq=1 Ack=77 Win=64192 Len=51
66640	2022-02-17 15:39:31.385019	192.168.140.20	192.168.140.235	TCP	54 62580 → 4059 [ACK] Seq=77 Ack=52 Win=131328 Len=0
66705	2022-02-17 15:39:34.546273	192.168.140.20	192.168.140.235	TCP	85 62580 → 4059 [PSH, ACK] Seq=77 Ack=52 Win=131328 Len=31
66706	2022-02-17 15:39:34.547602	192.168.140.235	192.168.140.20	TCP	60 4059 → 62580 [ACK] Seq=52 Ack=108 Win=64192 Len=0
66707	2022-02-17 15:39:34.548904	192.168.140.235	192.168.140.20	TCP	85 4059 → 62580 [PSH, ACK] Seq=52 Ack=108 Win=64192 Len=31
66709	2022-02-17 15:39:34.588466	192.168.140.20	192.168.140.235	TCP	54 62580 → 4059 [ACK] Seq=108 Ack=83 Win=131072 Len=0
66711	2022-02-17 15:39:34.651373	192.168.140.20	192.168.140.235	TCP	54 62580 → 4059 [FIN, ACK] Seq=108 Ack=83 Win=131072 Len=0
66712	2022-02-17 15:39:34.653122	192.168.140.235	192.168.140.20	TCP	60 4059 → 62580 [FIN, ACK] Seq=83 Ack=109 Win=64192 Len=0
66713	2022-02-17 15:39:34.653151	192.168.140.20	192.168.140.235	TCP	54 62580 → 4059 [ACK] Seq=109 Ack=84 Win=131072 Len=0