Master
ENGINEERING (MSE)

DeSEm
Layered Communication Architecture

Dominique Gabioud
Michael Clausen
Thomas Sterren
Medard Rieder

HES-SO 2021/22

Table of Content

Communication systems are complex!

Address a given application within a station…

Share a radio channel with other stations…

Exchange cryptographic keys…

Find a route in a meshed network…

Encode/decode bits in/from a radio wave…

Manage bit errors and packet/frame losses…

Get dynamically a node address…

Subscribe to values measured by a network sensor…

Decode audio…

Agree on meaning of register identifiers

Humans know how to build complex systems!

- Solution: decompose the complex system in smaller, manageable subsystems
  - Subsystems form a hierarchy
  - Limited interaction between subsystems
    - Through well-known interfaces

Hierarchical decomposition of a complex system: an example

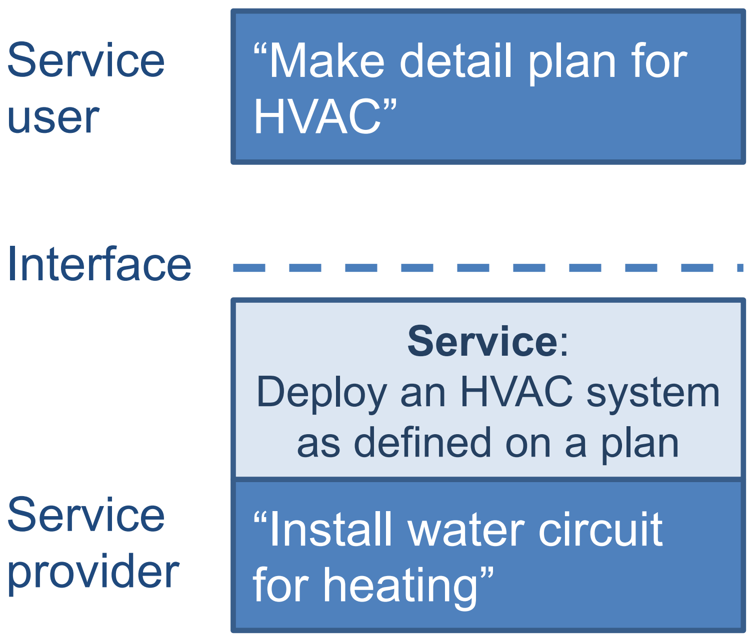
"Build a new office building with an open space for 80 persons and individual offices for 10"	"The boss"
"Select an appropriate land"	"An economist & a geographer"
"Build a reduced-size model and draw overall plans"	"An architect"
"Elaborate a concept for HVAC" …	"A heating engineer"
"Make detail plan for HVAC" …	"A HVAC designer"
"Install water pipes for heating" …	"A plumber"

Abstract

Concrete

Services

A “lower layer” makes available services to an “upper layer”



The service abstracts some activity for the service user

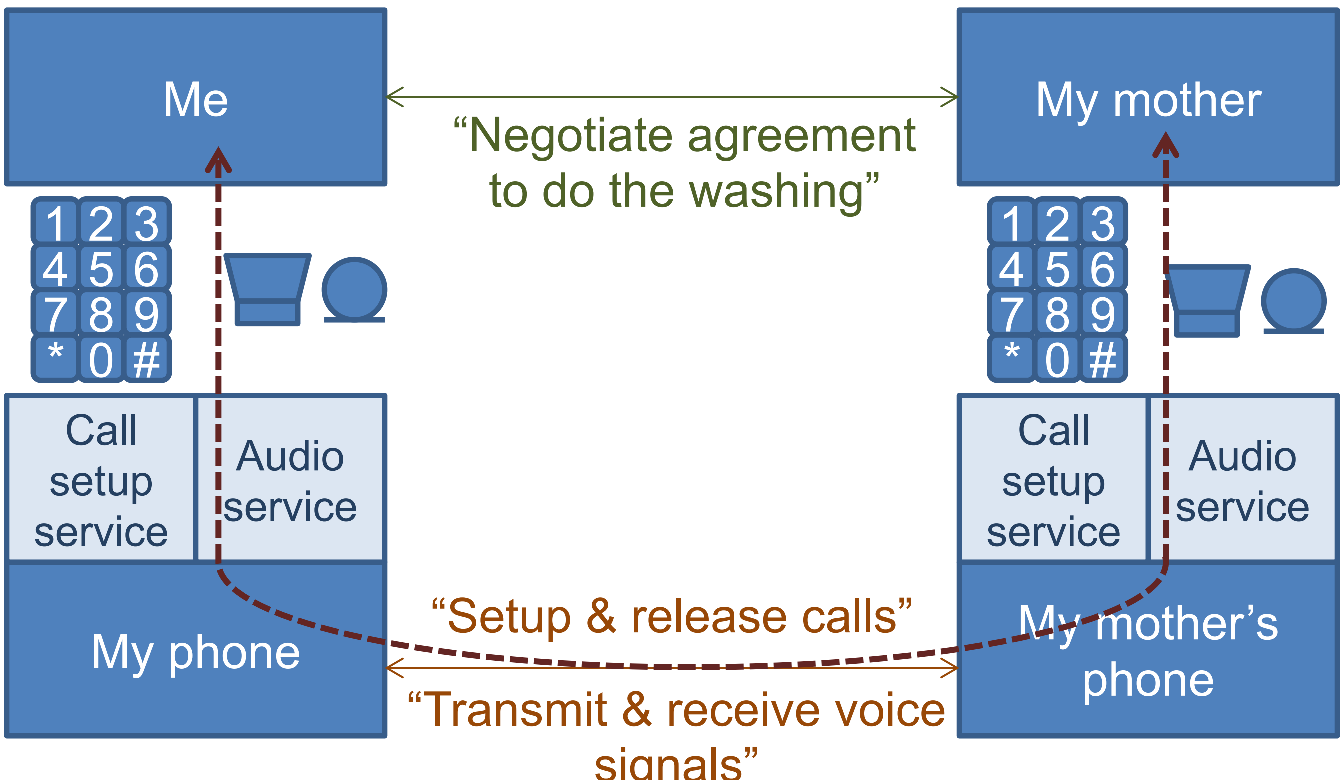
The service can be used through a specific interface

The interface must be agreed upon by the service provider and the service user

Specificity of Communication Systems

"Virtual" communication link between peer parties

At each layer, two peer communicating parties



"Real" communication link between a person and her/his telephone

"Real" communication link only between peer parties at the lowest layer

Interoperability & standardisation

- By definition, communication involves interworking of different appliances
  - Usually from several manufacturers
- Interoperability requires respect of common rules
- Common rules are typically defined in standards
- Standard may be defined:
  - by official bodies like ISO, IETF, IEEE…
  - by ad hoc bodies like the Buetooth SIG

ISO: International Organization for Standardization

ISO standardisation for communication

- In the 1970's computer communication was a "mess"!
  - Proprietary cabling systems, file formats, network services
- Two initiatives started to promote interoperability
  - TCP/IP initiative led by US universities
    - Universities financed by the US Department of Defence (DoD)
  - OSI initiative led by ISO and the computer / telecom industry
- TCP/IP initiative:
  - Focus on the development of interoperable solutions
- OSI initiative:
  - Focus on the specification of interoperable solutions

OSI: Open Systems Interconnection

Purposes of the OSI model

- Purpose #1:
  - Define layers for open communication systems and assign functions to them
- Purpose #2:
  - Elaborate a framework for the definition of a layered communication architecture

ISO vision:

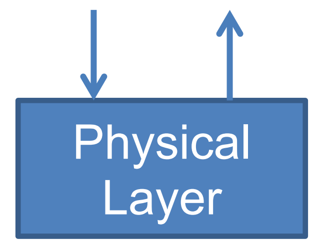
Provide tools to promote interoperability without limiting innovation

Physical & Data Link layers

OSI model layers

- **Physical** layer

Bits

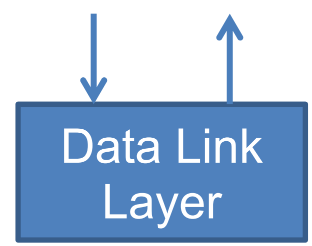


Some physical signal representing bits

Transmit & receive bits

- **Data Link** layer

Frames



Bits

Framing, broadcast channel access control, local addressing, local error control

Exchange frames with local station

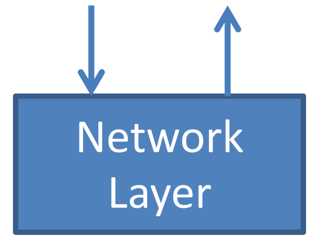
Network & Transport layers

OSI model layers

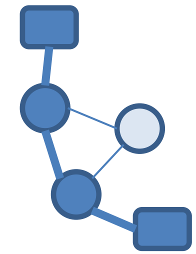
- Network layer

Exchange packets over a network

Packets



Frames

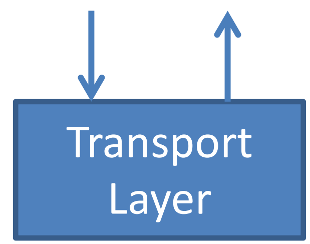


Finding a route in a meshed network

- Transport layer

Provide a reliable communication channel between two applications

Packets, streams



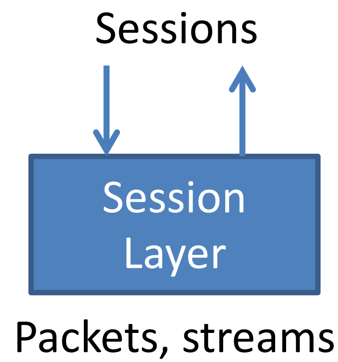
Packets

Internal application sub-addressing, end-to-end error control

Session & Presentation layers

OSI model layers

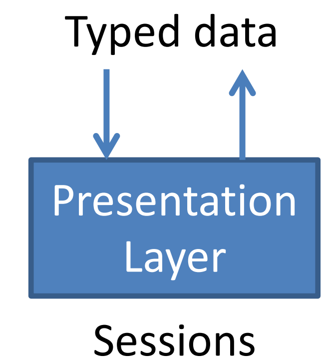
- **Session** layer



Maintain service, possibly over several Transport channels

Keep track of current activity, to be able to resume it on a another stream

- **Presentation** layer



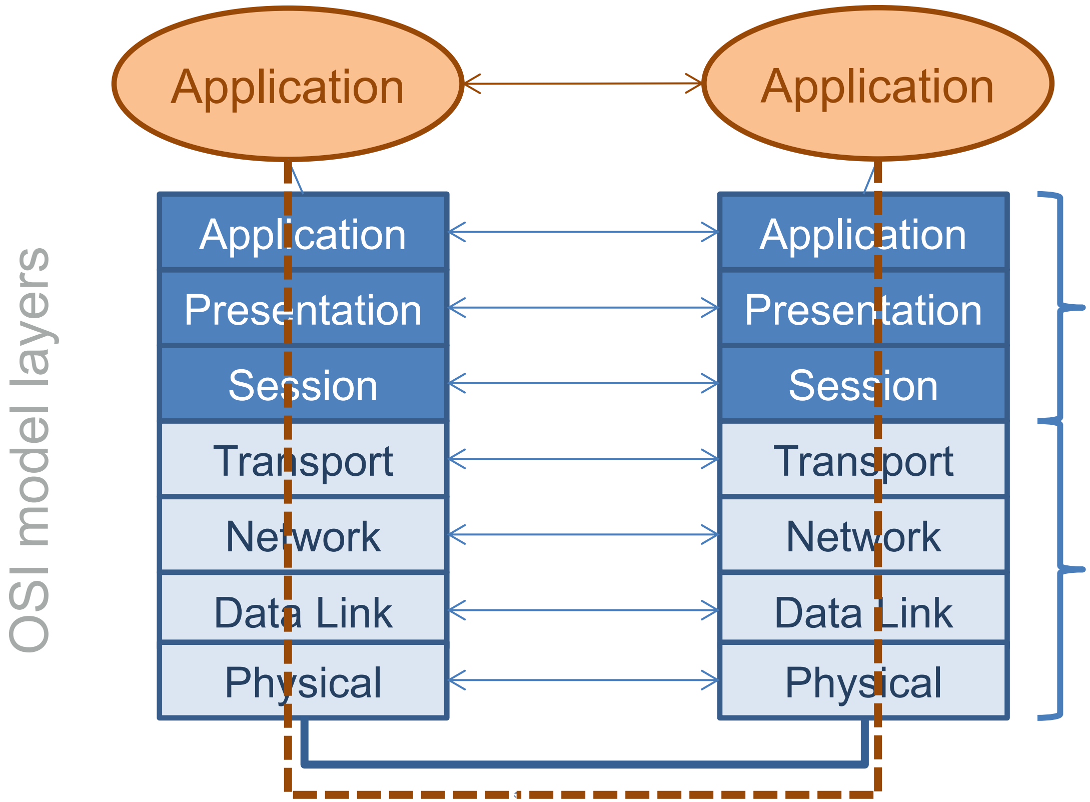
Represent typed data in an agreed format

Negotiate common format, perform required translation, implement ciphering & authentication

Application Layer

- Application layer
  - Set of specific services
    - File transfer, file sharing, document retrieval, remote access to local inputs & outputs, remote terminal, instant messaging…

Overview of OSI Layers

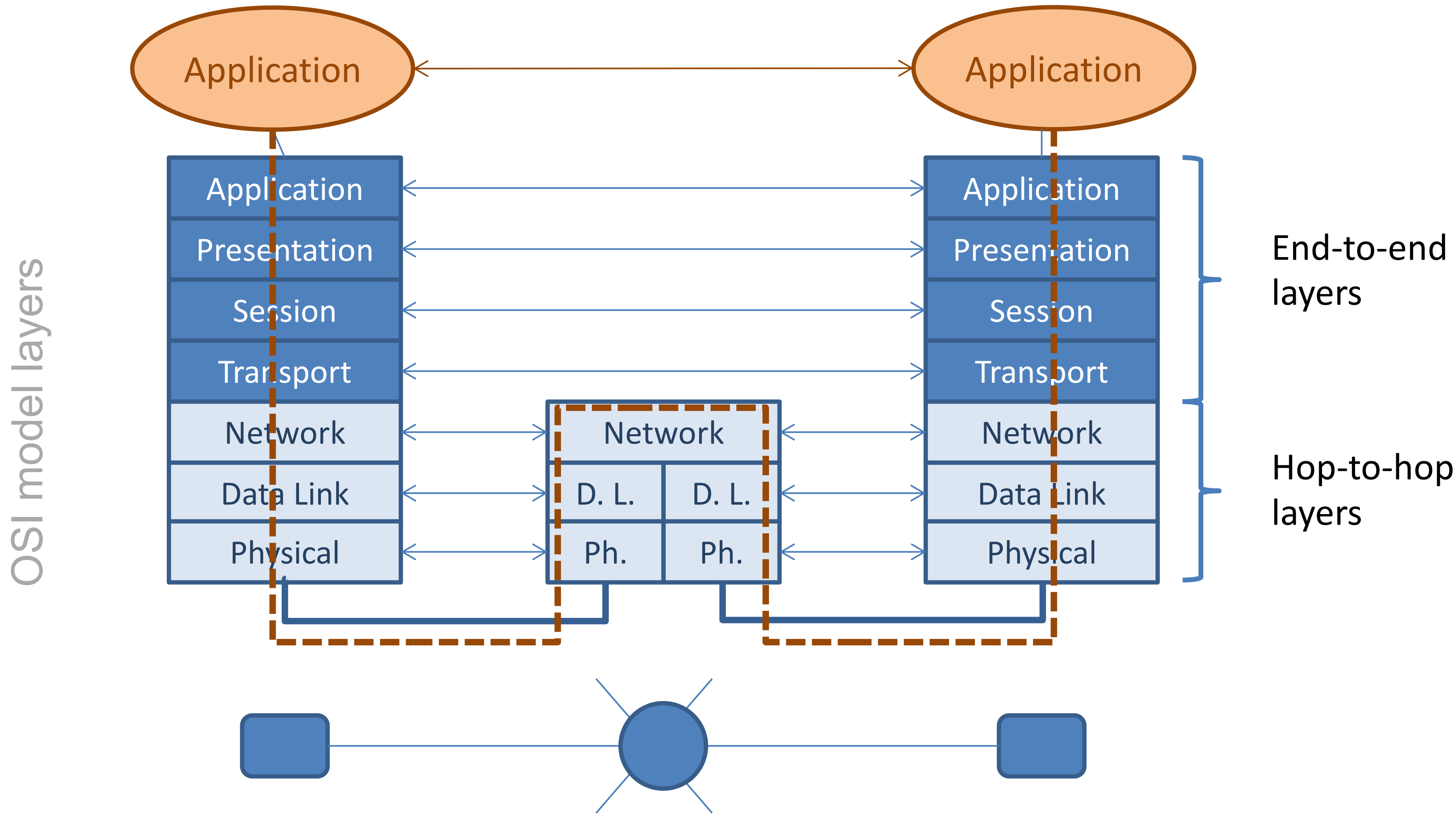


Operation handling & data formatting

Reliable application-to-application raw data transport

Abstract "horizontal" communication
Real "vertical" communication

OSI layers & communication networks



Concluding remarks

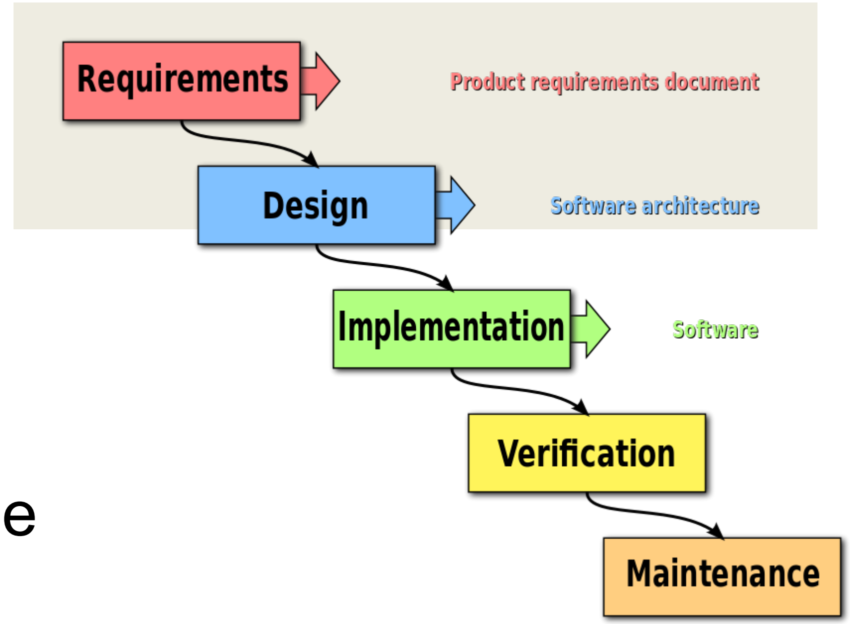
- The intent of ISO was to “write in stone” the role of each layer
  - Interoperability would then be simplified because a given role is always implemented at the same layer
- The number of layers and their role is strongly influenced by the context and the technology
  - OSI intent was never concretised
  - The OSI model is not given by law of physics but is the result of a compromise between experts
- Today, the OSI layers are a kind of scale allowing to quickly position a communication layer

What's a protocol?

- The behaviour of each layer in a communication architecture is specified by a "law" called a protocol
  - The protocol covers:
    - The services that are provided to the layer above
    - The peer-to-peer (horizontal) communication
    - The relationship between services and peer-to-peer communication
      - Dynamic behaviour typically specified in a state machine
- The OSI model does not specify protocols
      - Actually, ISO defined also OSI compatible protocols, but they have never been widely used…
    - ISO vision: protocols would evolve with technology…
- The protocol definition does not address implementation issues

What we are going to do

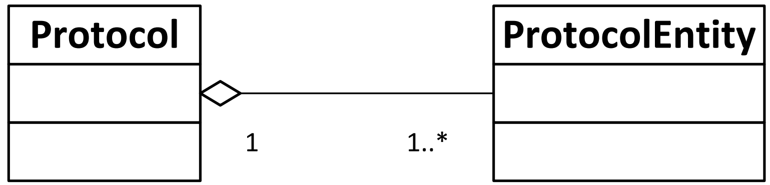
- Study the OSI model framework for protocol definition
  - Framework as "language" for requirements
- Address some patterns for the design phase



Waterfall model by Peter Kemp / Paul Smith – Adapted from Paul Smith's work

- *In DeSEm, you'll have the opportunity to perform the whole process based on a very simple communication architecture*

Protocol vs. protocol entity

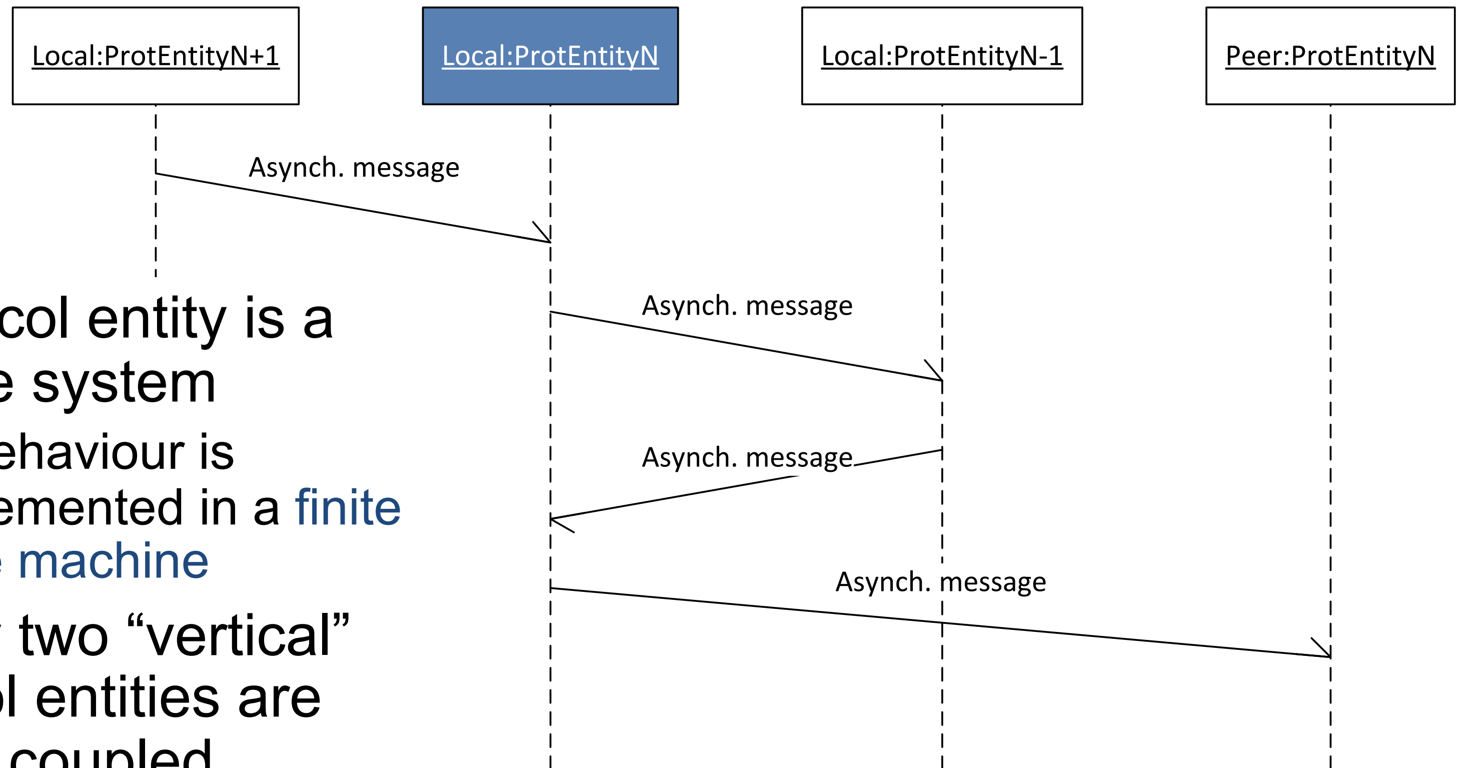


OSI model operation

- In a given station, a protocol can be implemented in one or more reactive protocol entities
  - Examples:
    - One protocol entity for data transfer and one protocol entity for management
      - Each entity executes a different finite state machine
    - One protocol entity per connection, parallel connections supported
      - In this case, protocol entities are created dynamically and have the same life duration as their corresponding connections
      - Each entity executes an instance of the same state machine

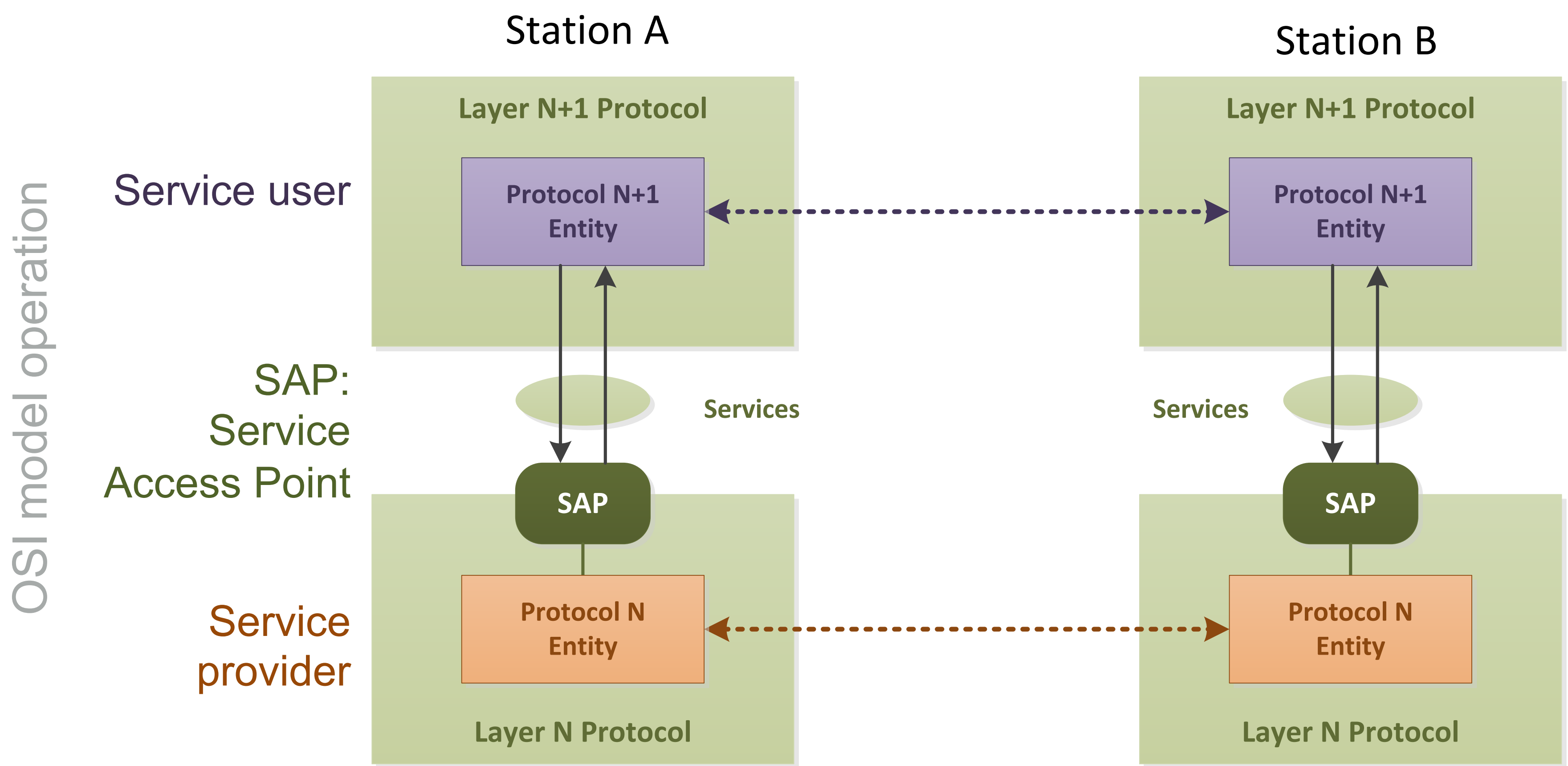
Protocol: a specification

Protocol entity as a Finite State Machine



- A protocol entity is a reactive system
  - Its behaviour is implemented in a finite state machine
- Usually two “vertical” protocol entities are loosely coupled
  - Asynchronous communication

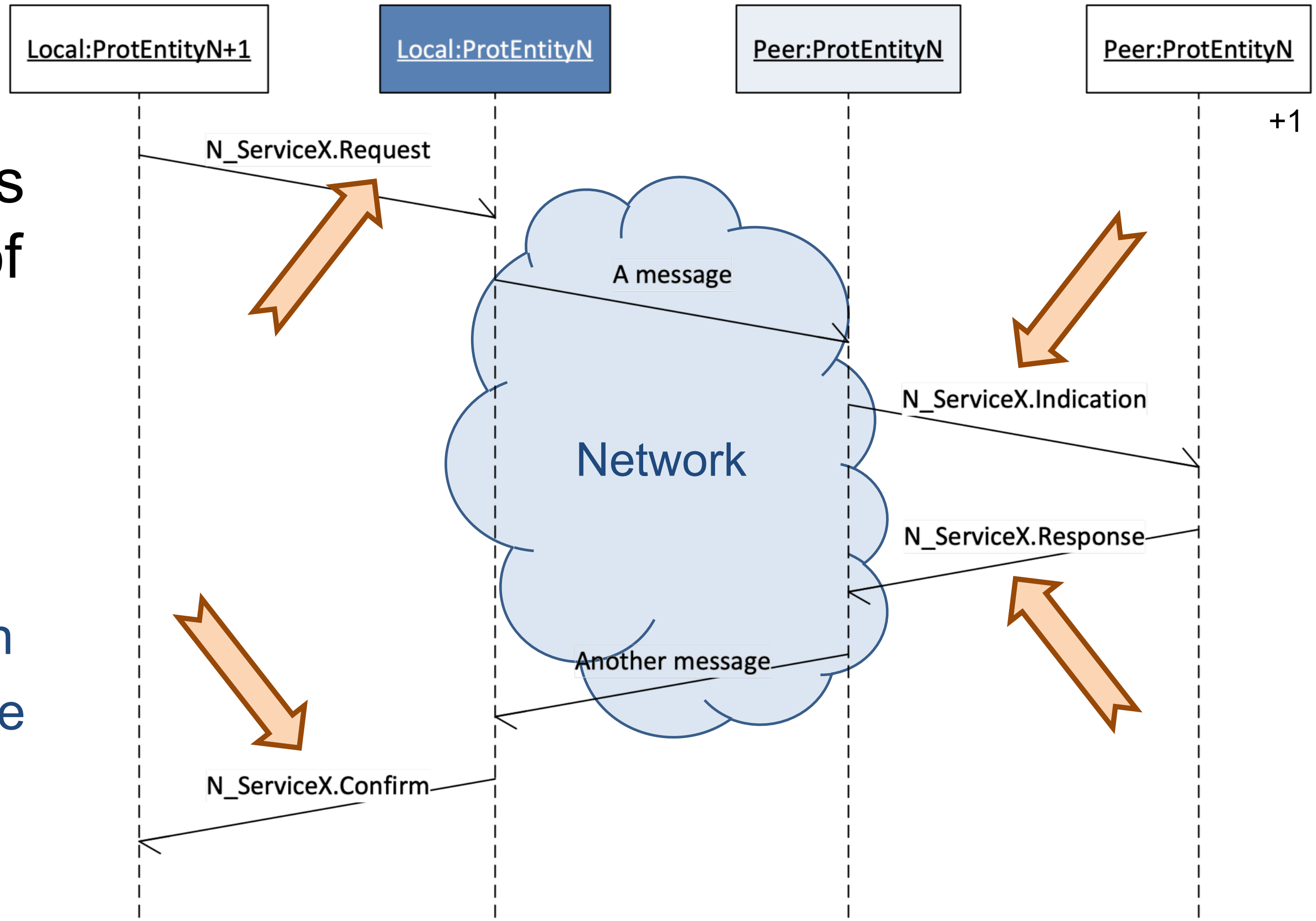
Services in the OSI model



Service & service primitives

OSI model operation

- A service is made up of a set of service primitives
  - Request
  - Indication
  - Response
  - Confirm



Service primitives

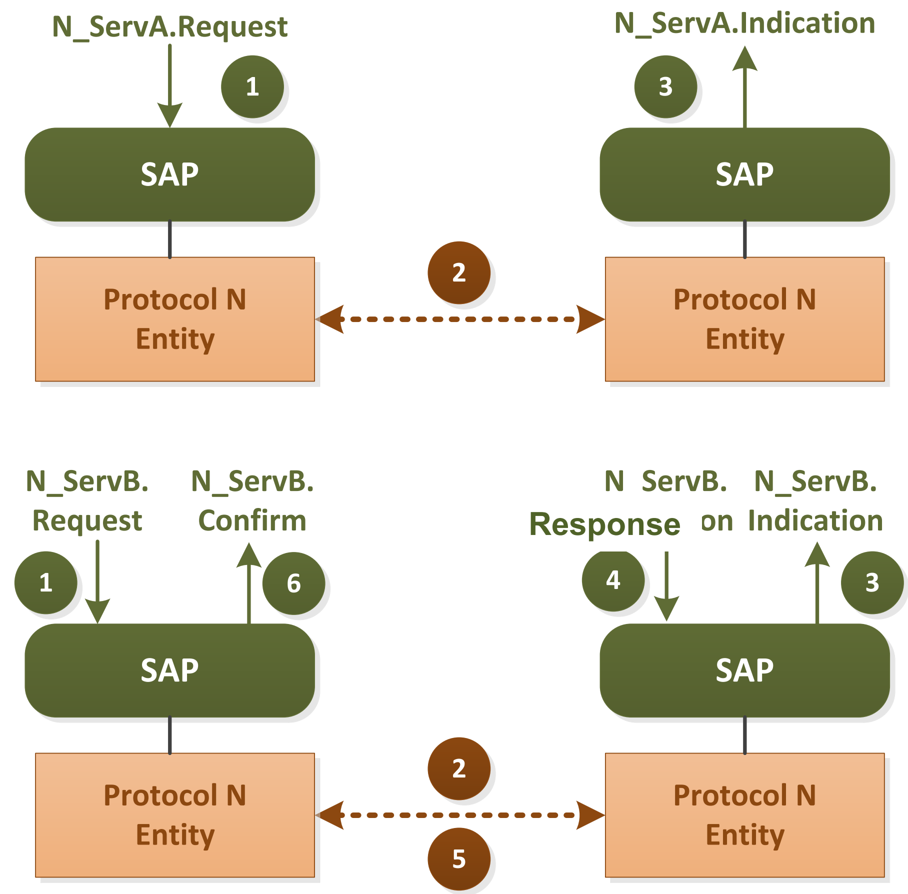
- Request
  - Primitive used by a service user to request a service
- Indication
  - Primitive used by a service provider to indicate to a service user that a peer service user has requested a service
- Response
  - Primitive used by a service user to respond to an indication primitive
- Confirm
  - Primitive used by a service provider to report the result of a previous service request primitive

Confirmed & unconfirmed services

OSI model operation

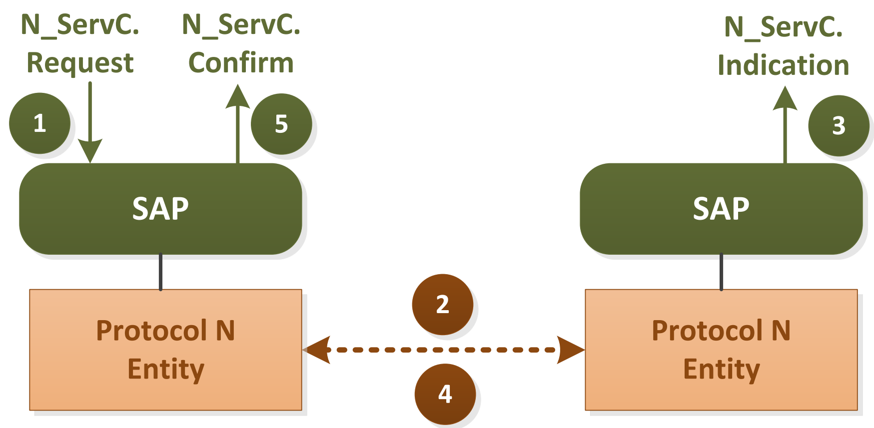
"**ServA**" is an
unconfirmed service

"**ServB**" is a
service confirmed by peer

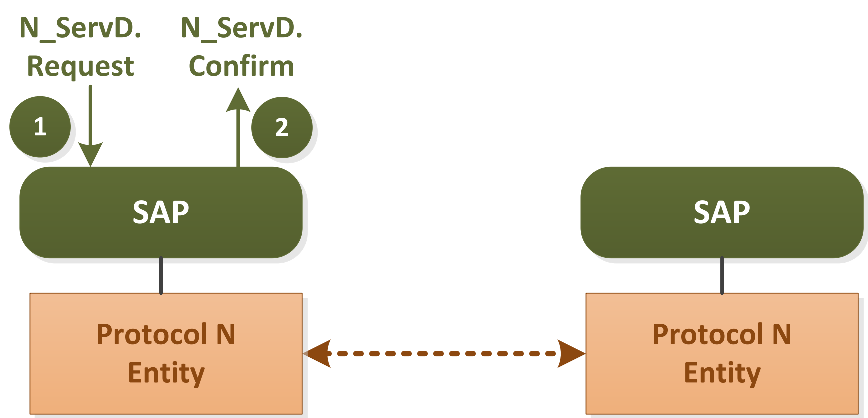


Confirmed & unconfirmed services

"**ServC**" is a locally confirmed service (based on some feedback provided by the Protocol N peer Entity)

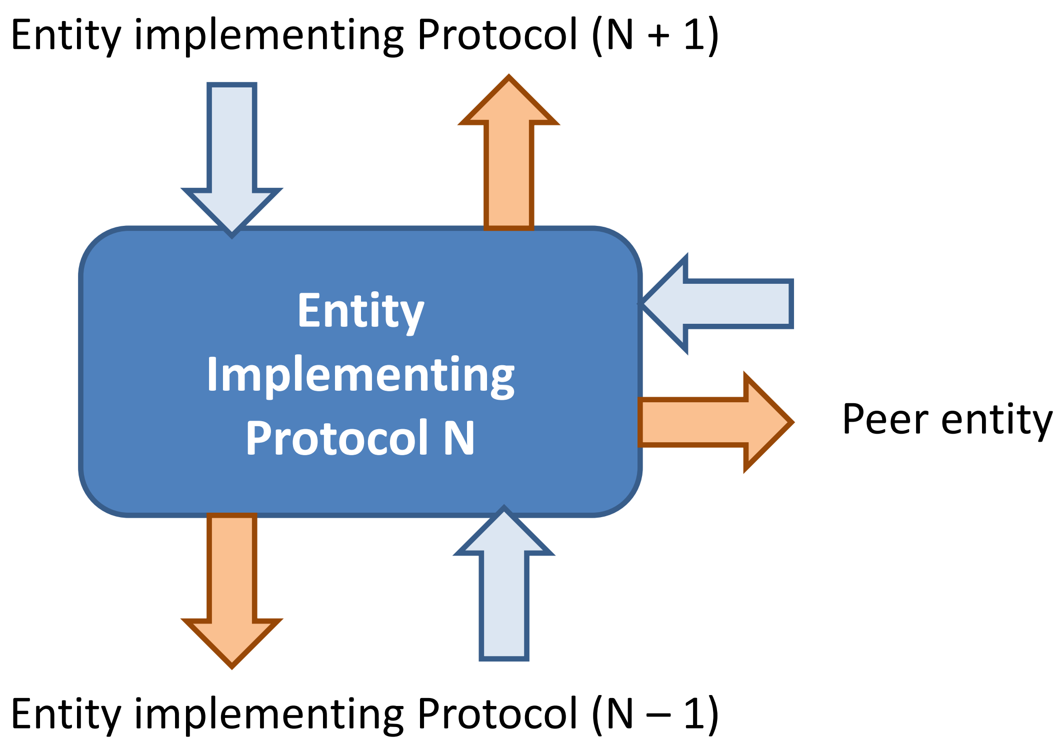


"**ServD**" is a locally confirmed service (the service modifies some internal parameter of the Protocol N Entity)

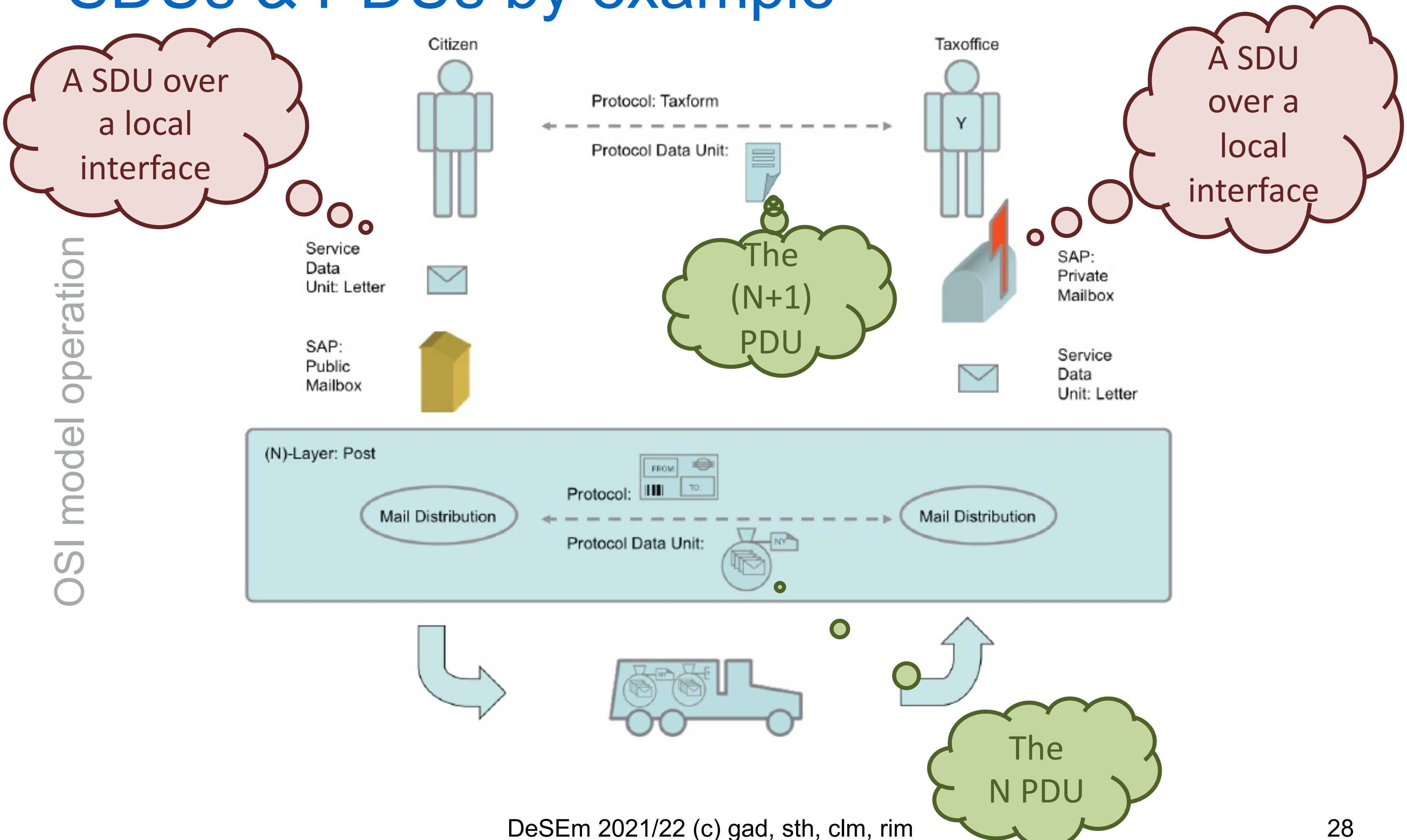


Protocol entity as a reactive system

- A protocol entity
  - Hardware and/or software reactive module implementing the behaviour defined by a protocol



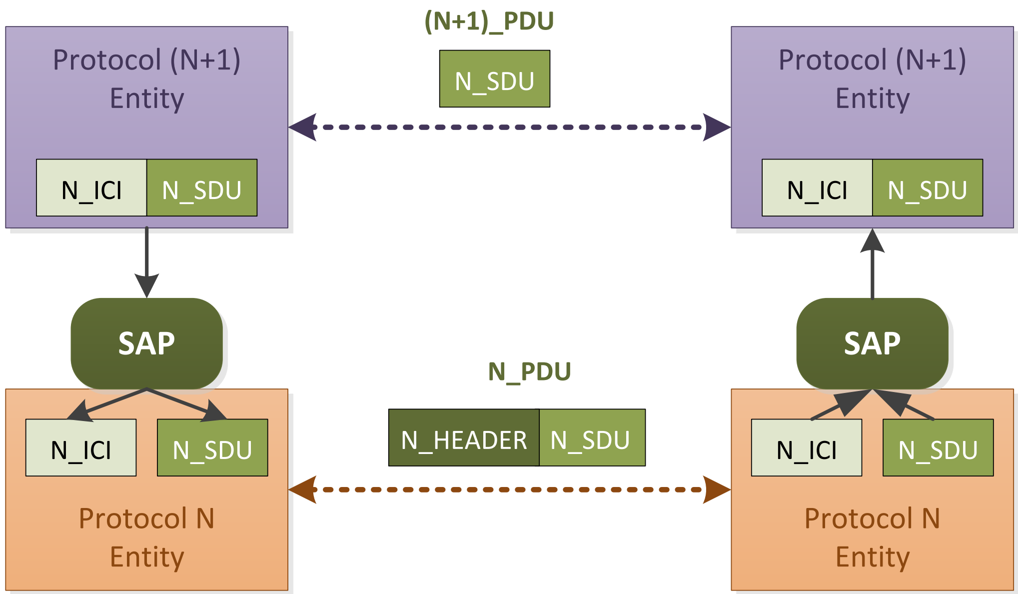
SDUs & PDUs by example



OSI model operation

SDUs & PDUs: Functioning

OSI model operation

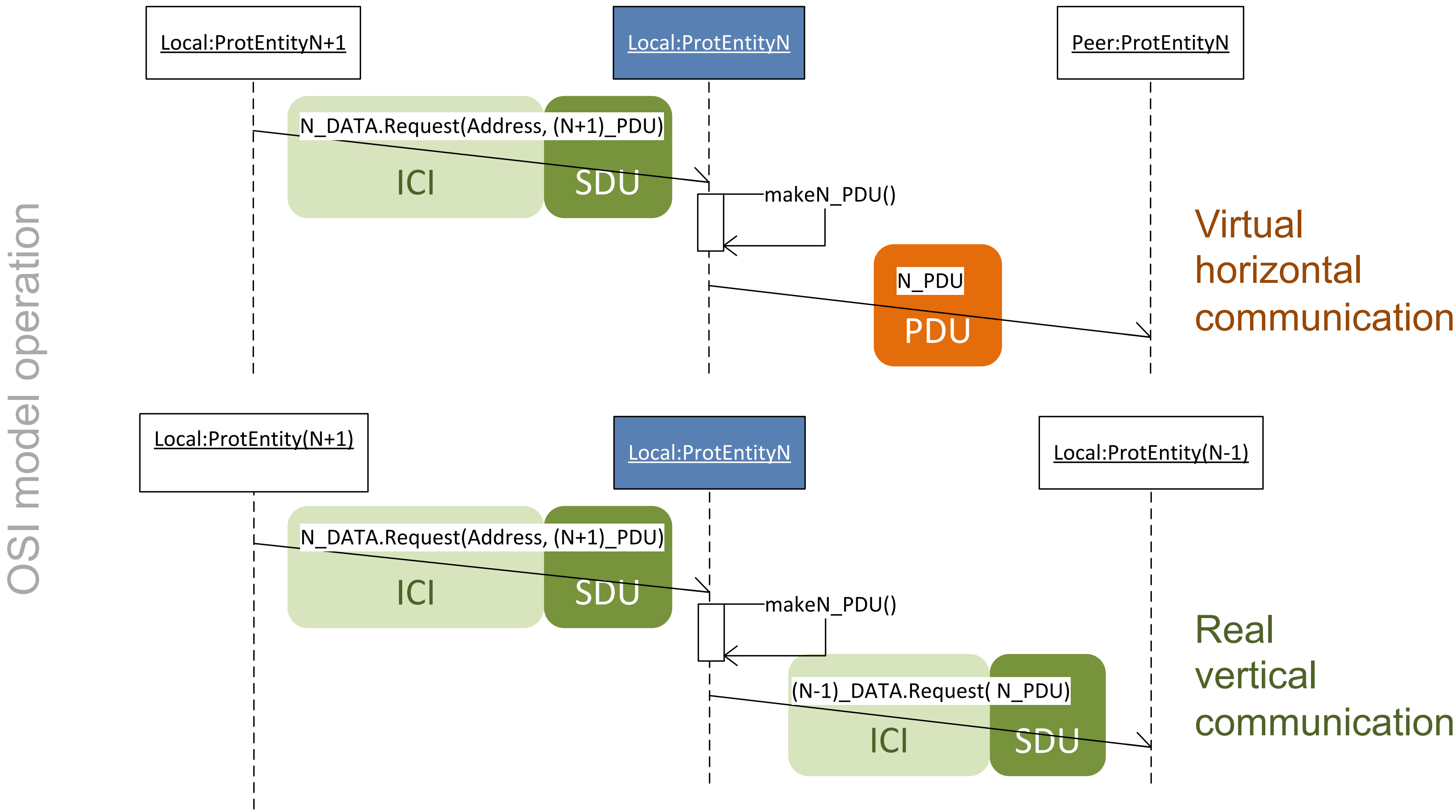


IDU	Interface Data Unit
ICI	Interface Control Information
PDU	Protocol Data Unit
SDU	Service Data Unit

PDU(N) = Header(N) + PDU(N + 1)
= Header(N) + SDU(N)
= SDU(N - 1)

SDU(N) = PDU(N + 1)

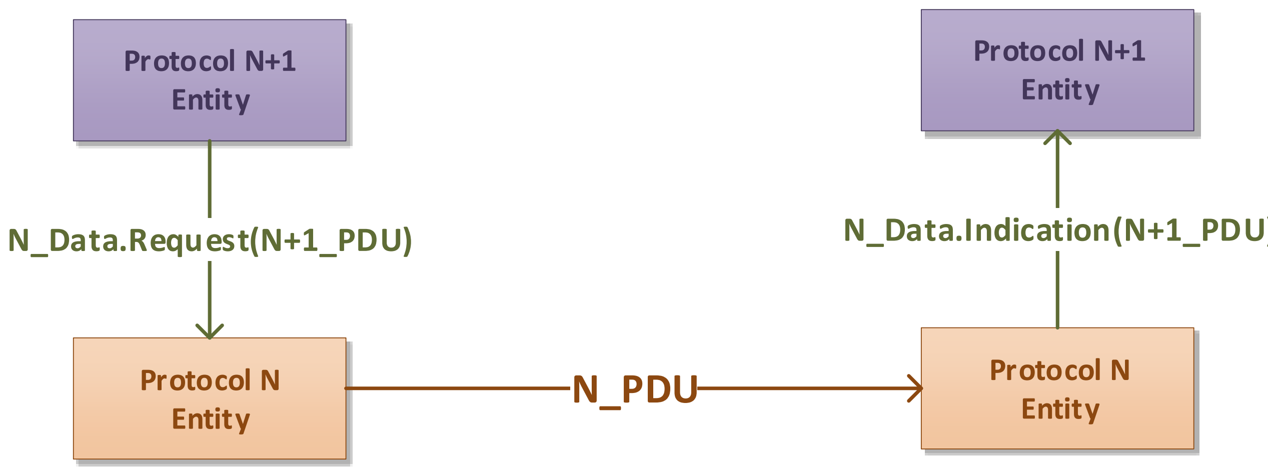
SDUs & PDUs : Interaction



Real communication vs. virtual communication in the OSI model

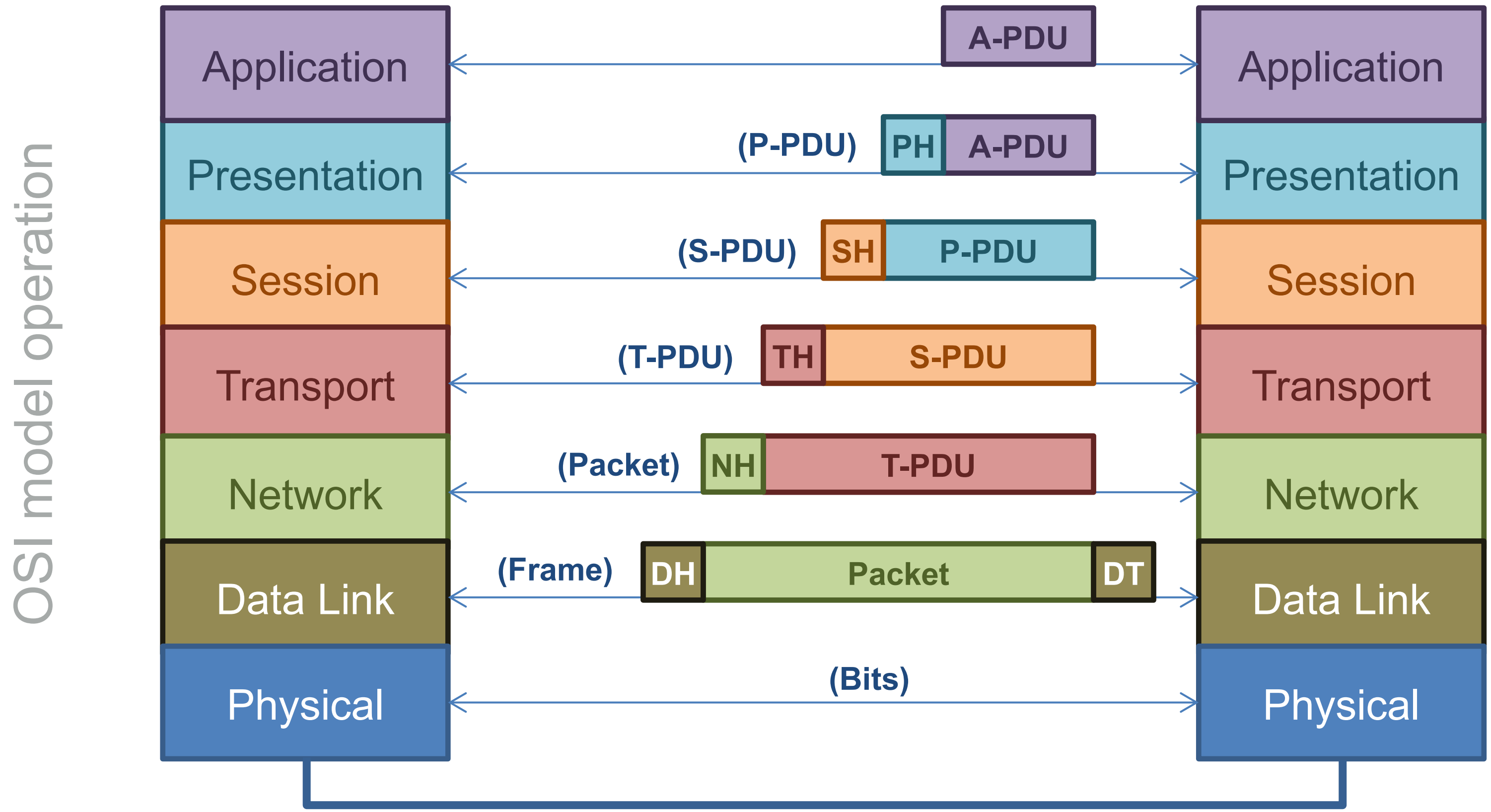
OSI model operation

What we have in practice: the Data service takes care of the transport of an upper layer PDU



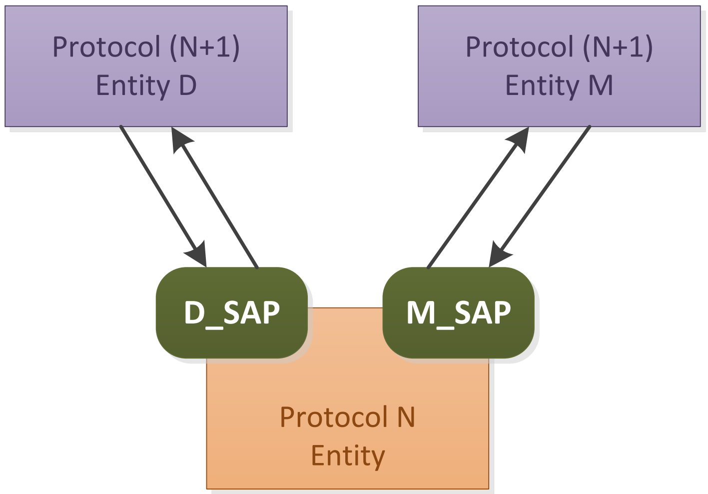
From a logical standpoint we consider that the N+1 protocol entity receives directly the PDU from tis peer entity

Recursion!

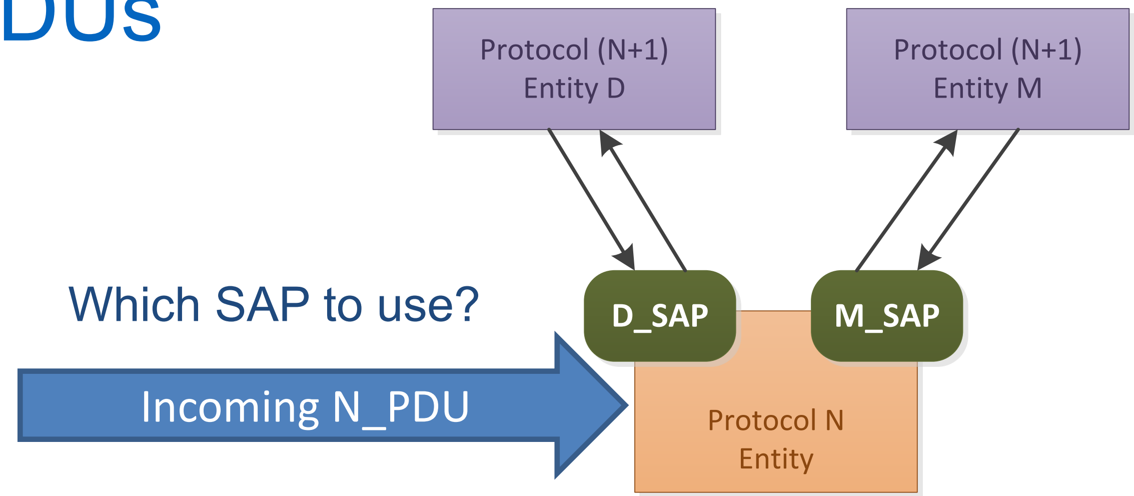

OSI model operation
Application
Presentation
Session
Transport
Network
Data Link
Physical
A-PDU
(P-PDU)
PH
A-PDU
(S-PDU)
SH
P-PDU
(T-PDU)
TH
S-PDU
(Packet)
NH
T-PDU
(Frame)
DH
Packet
DT
(Bits)
Application
Presentation
Session
Transport
Network
Data Link
Physical

SAPs & Protocol entities

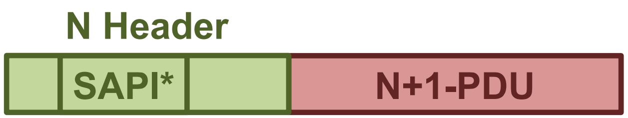
- A Service Access Point (SAP) is a conceptual location at which a protocol entity at layer N+1 can request a service of a protocol entity at layer N
- There is one SAP per pairs of “vertical” communicating entities



SAPIs & PDUs



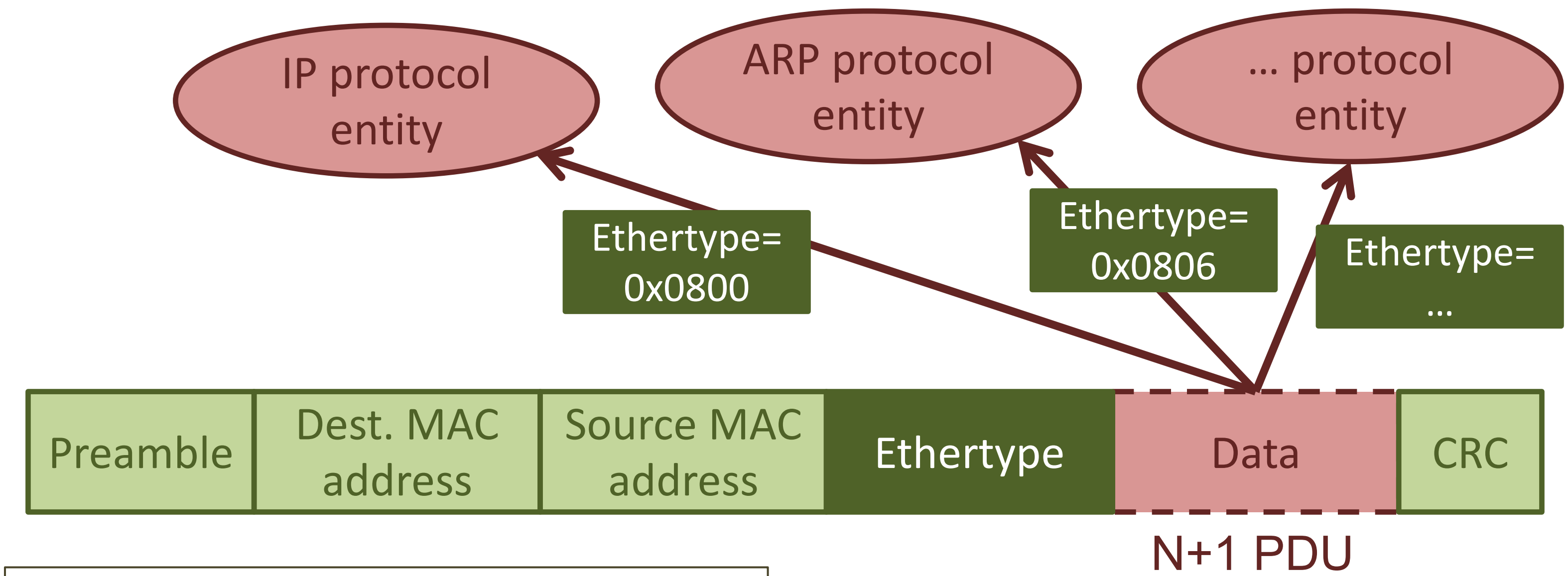
- The N-PDU contains a reference to a SAP
- This reference is called SAPI: SAP Identifier (usually a number)
- The SAPI is transmitted in a dedicated field of the N_PDU



* Either source SAPI & destination SAPI or single SAPI for both protocol N entities

SAPs & SAPIs: an Example

OSI model operation



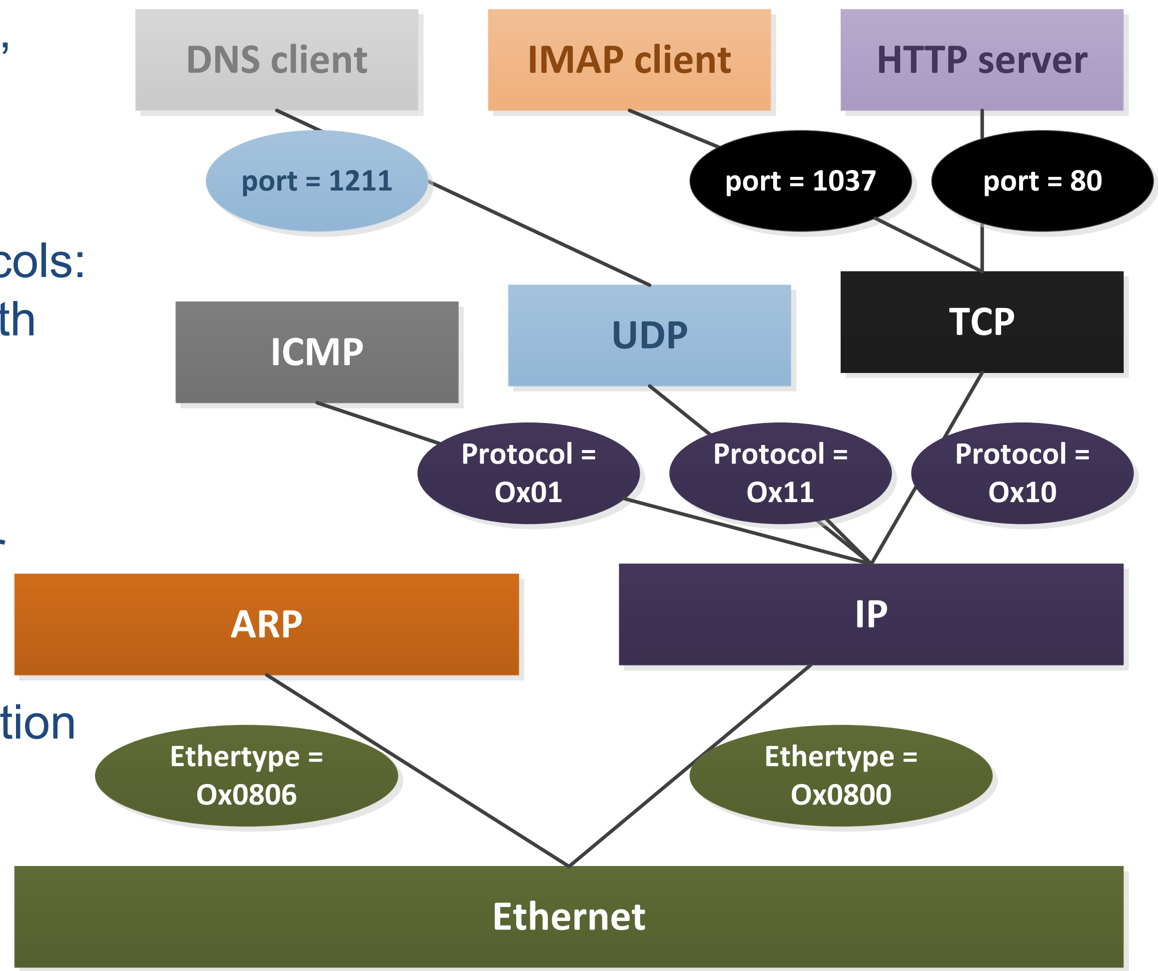
Same Ethertype SAPI used for both communicating protocol entities
Central registry of Ethernet SAPIs to avoid "collisions"

N+1 Protocol entity is identified by the SAPI Ethertype.
Receiving Ethernet protocol entity uses this information to forward the Ethernet PDU to the appropriate upper layer protocol entity

SAPs & SAPIs in TCP/IP

OSI model operation

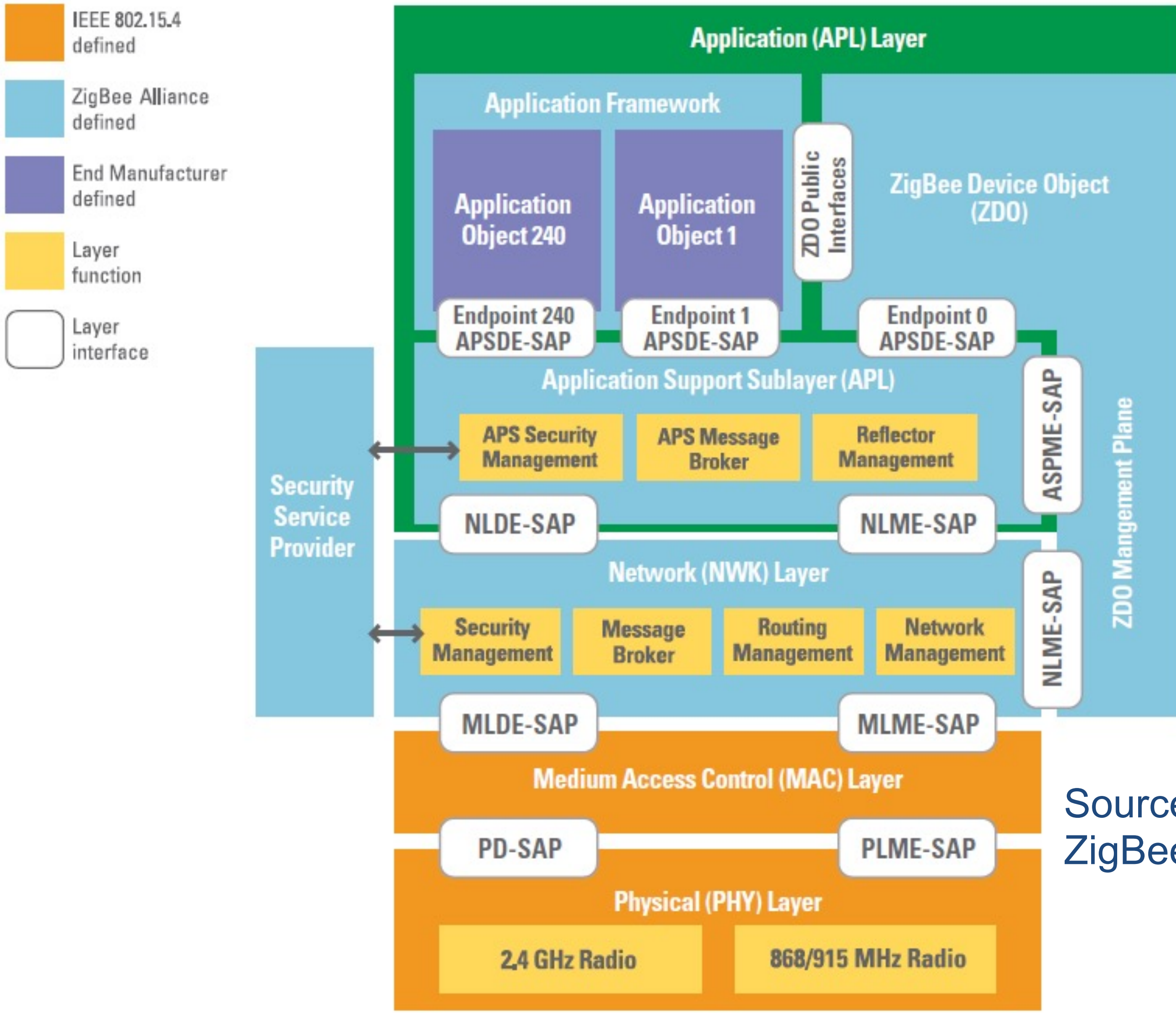
- Ethertypes, IP Protocols, TCP/UDP ports are examples of SAPIs
- Ethertypes and IP protocols: Unique SAPI field for both entities
- TCP/UDP ports:
  - Source port number is a source SAPI
  - Destination port number is a destination SAPI
  - TCP/UDP PDUs contains source & destination port numbers / SAPIs



OSI model vs. TCP/IP

- OSI introduced concept of services, interface, protocols.
  - These were eventually force-fitted to TCP later
    - In TCP model, not in TCP implementation
    - It is not easy to replace protocols in TCP
- In OSI, reference model was done before protocols
  - In TCP, protocols were done before the model
- OSI: Standardize first, build later
  - TCP: Build first, standardize later
  - OSI took too long to standardize; TCP/IP was already in wide use by the time
- OSI became too complex
- TCP/IP is not general but was designed ad hoc

Example: Zigbee



Source:
ZigBee Alliance

Example: IEEE802.15.4 Physical layer entities & SAPs

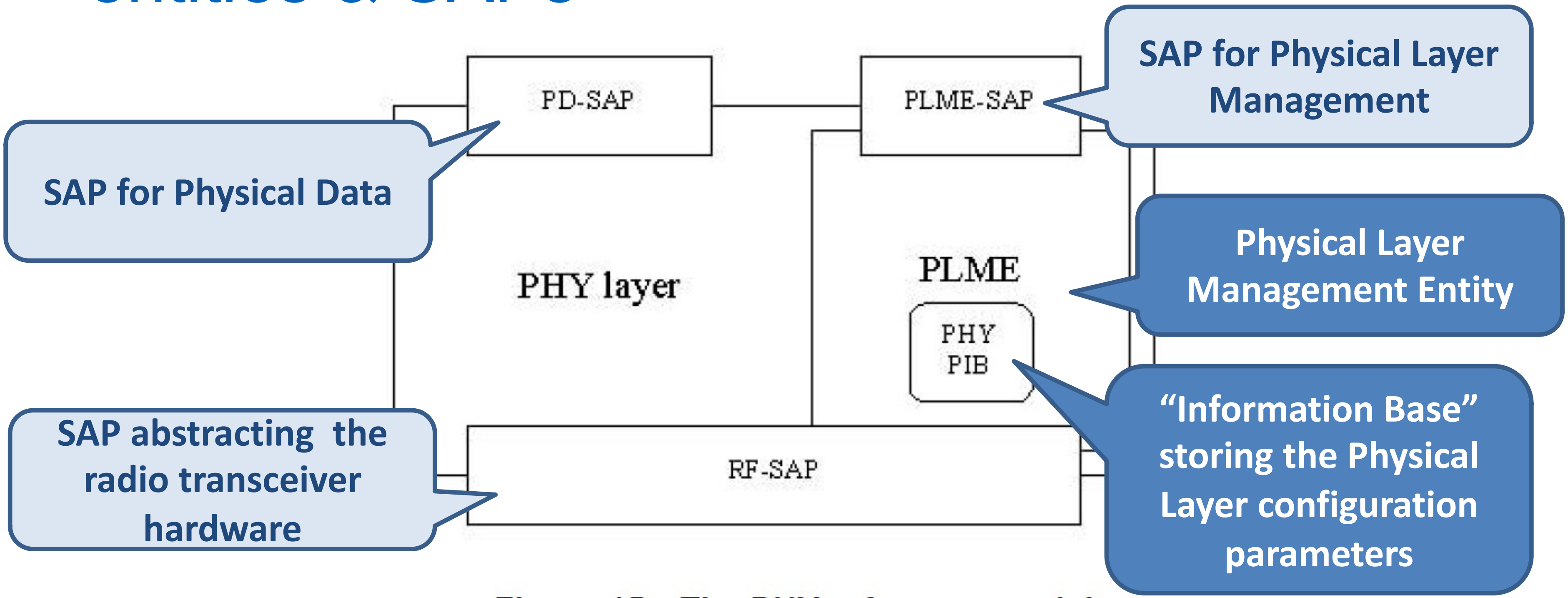


Figure 15—The PHY reference model

The PHY provides two services, accessed through two SAPs: the PHY data service, accessed through the PHY data SAP (PD-SAP), and the PHY management service, accessed through the PLME's SAP (PLME-SAP).

The PD-SAP service and its primitives

PD-SAP primitive	Request	Confirm	Indication
PD-DATA	6.2.1.1	6.2.1.2	6.2.1.3

This is a locally confirmed service

Table 4—PD-DATA.request parameters

Name	Type	Valid range	
psduLength	Unsigned Integer	≤ *aMaxPHYPacketSize*	The number of octets contained in the PSDU to be transmitted by the PHY entity.
psdu	Set of octets	—	The set of octets forming the PSDU to be transmitted by the PHY entity.

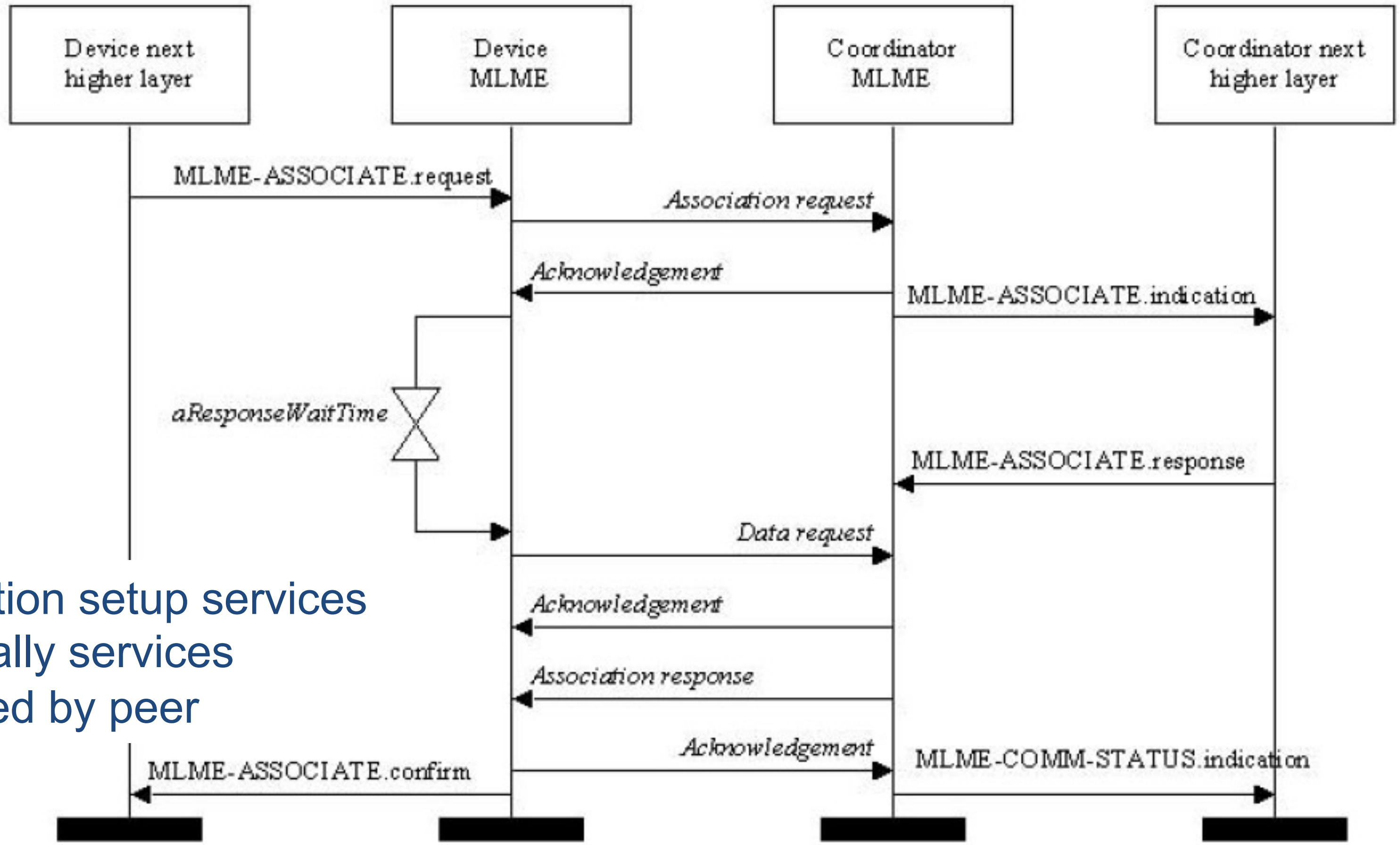
Table 5—PD-DATA.confirm parameters

Name	Type	Valid range	Description
status	Enumeration	SUCCESS, RX_ON, or TRX_OFF	The result of the request to transmit a packet.

The PLME-SAP

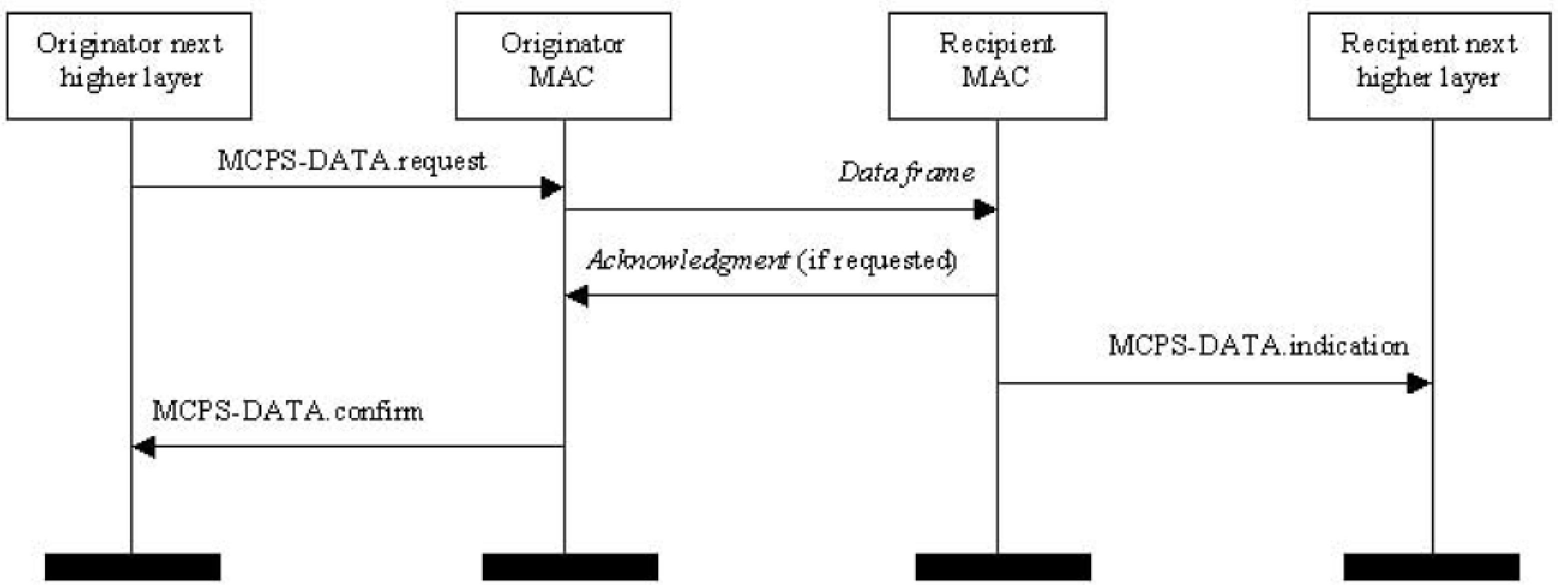
- The PLME-SAP supports management services
- Example of functions implemented by management services:
  - Select a radio channel and a transmit power
  - Turn RX radio on or off
  - Check whether the radio channel is busy
- All PLME-SAP services are locally confirmed services

Example of a service confirmed by peer

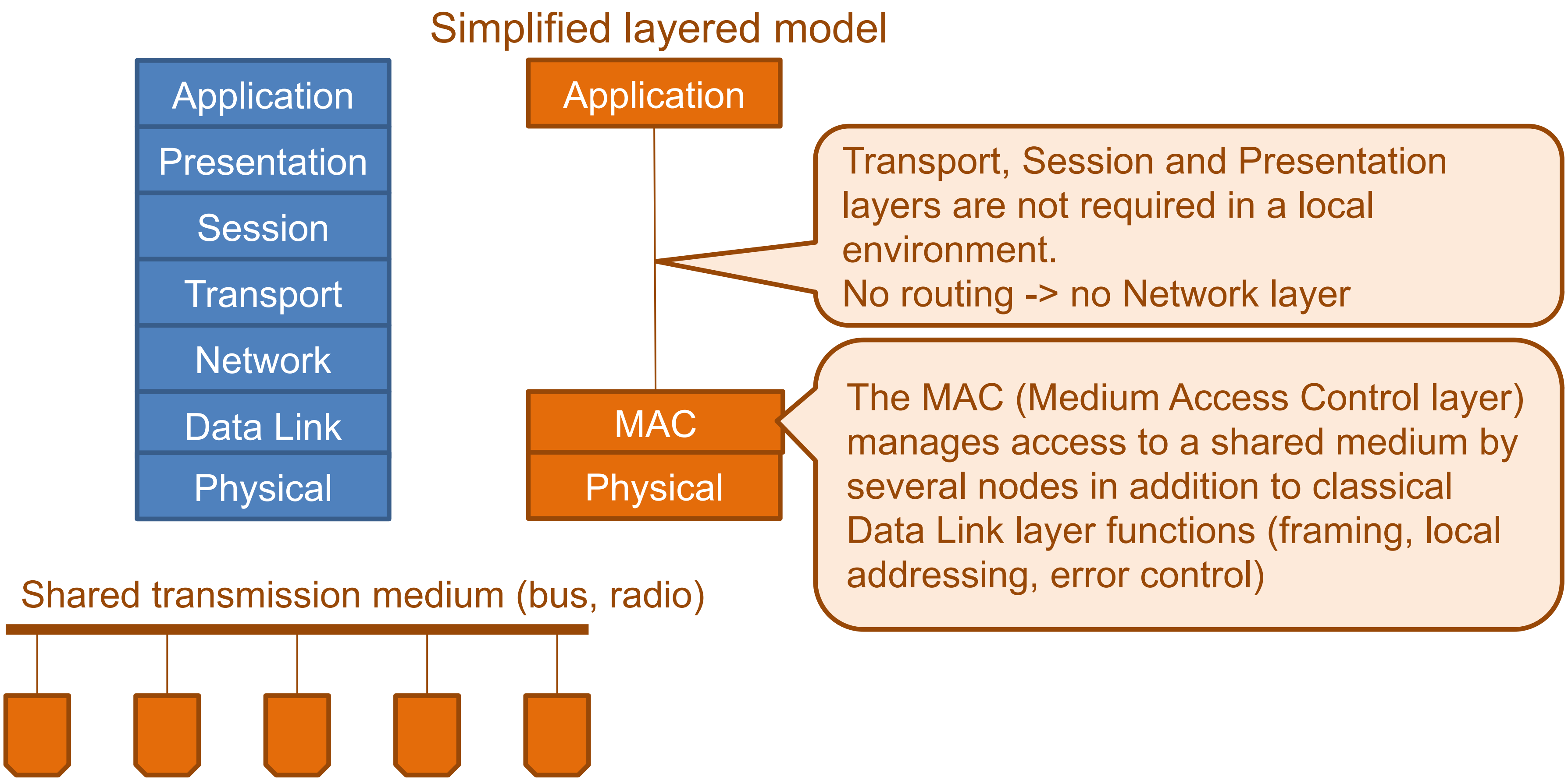


Connection setup services are usually services confirmed by peer

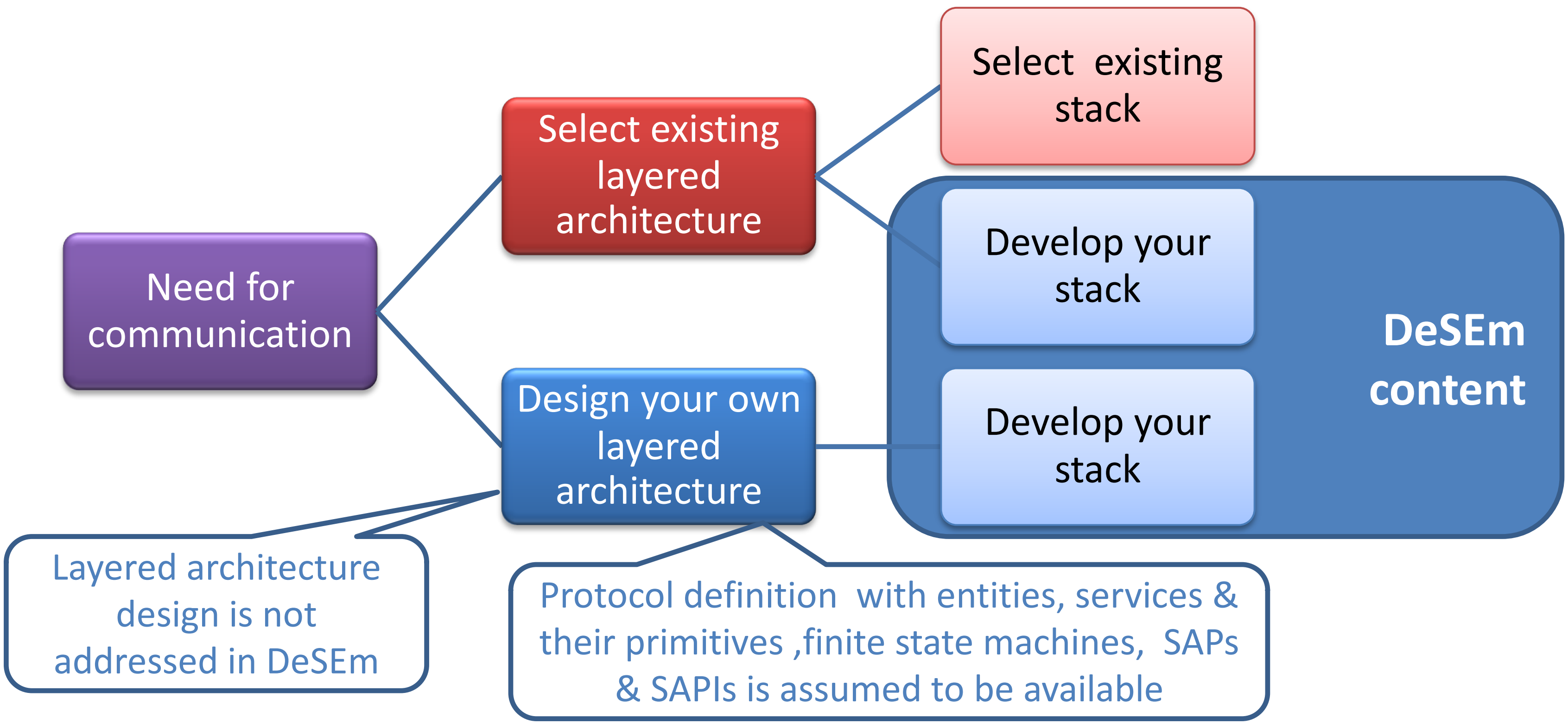
Example of a locally confirmed service



A simplified layered architecture

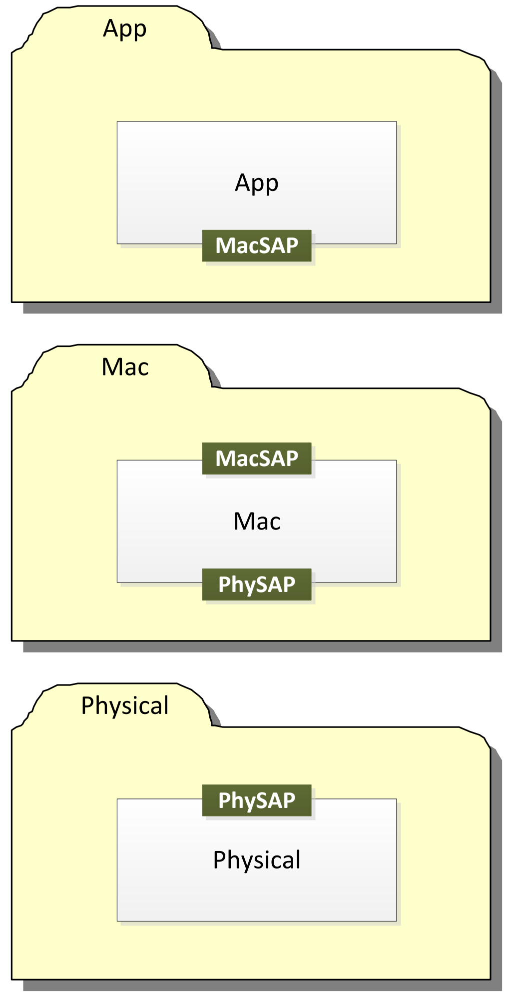


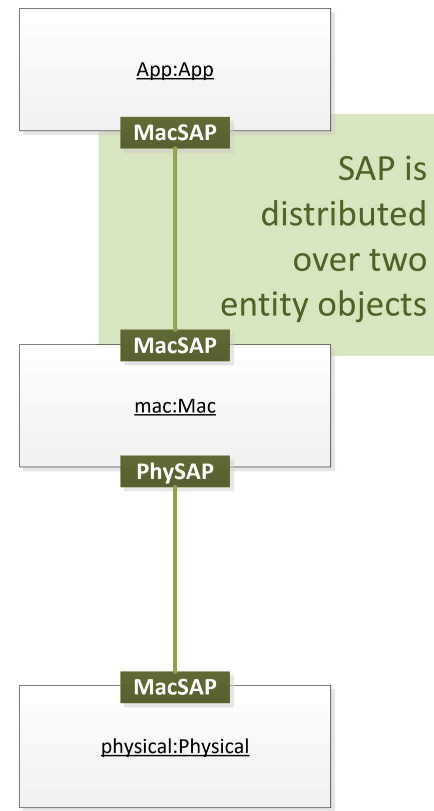
DeSEm approach



Single process strategy

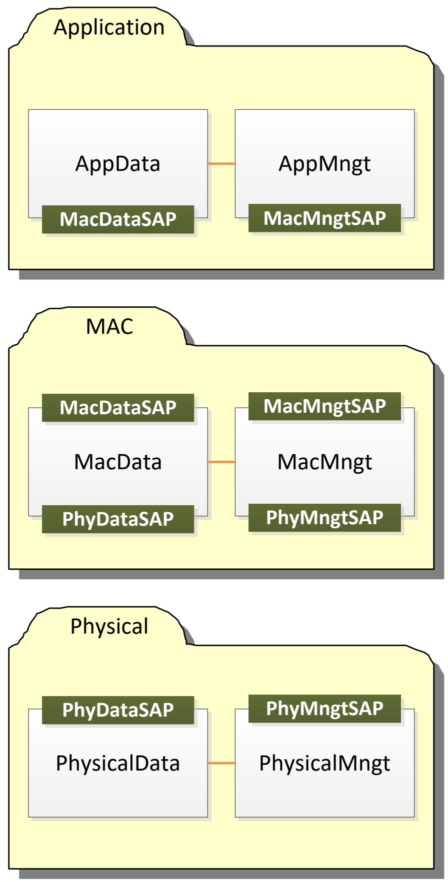
- One process per protocol entity
- A process deals with two SAPs
  - Except higher layer process and lower layer process

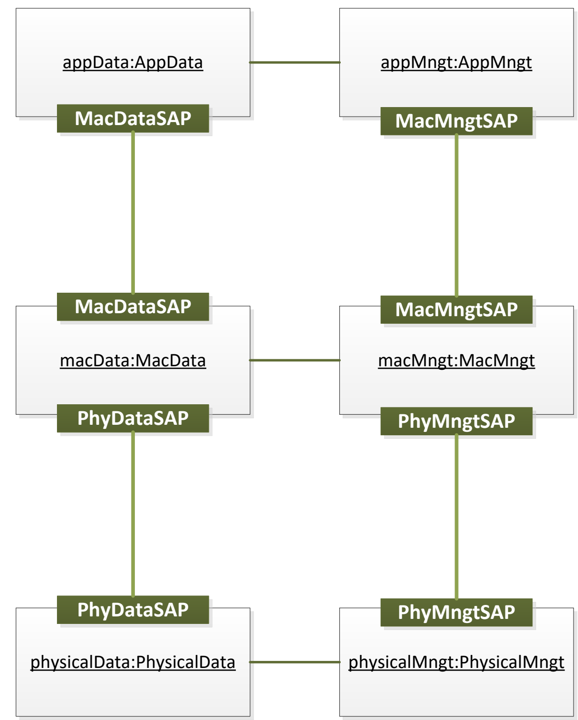




Data and management processes strategy

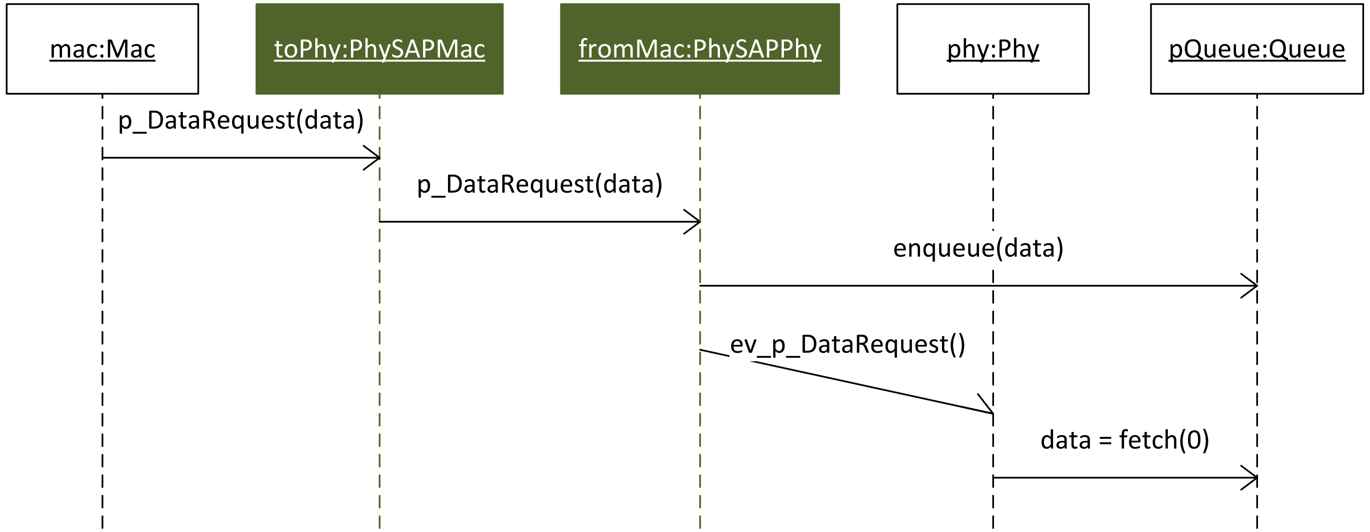
- Goal:
  - Separate data and management flows



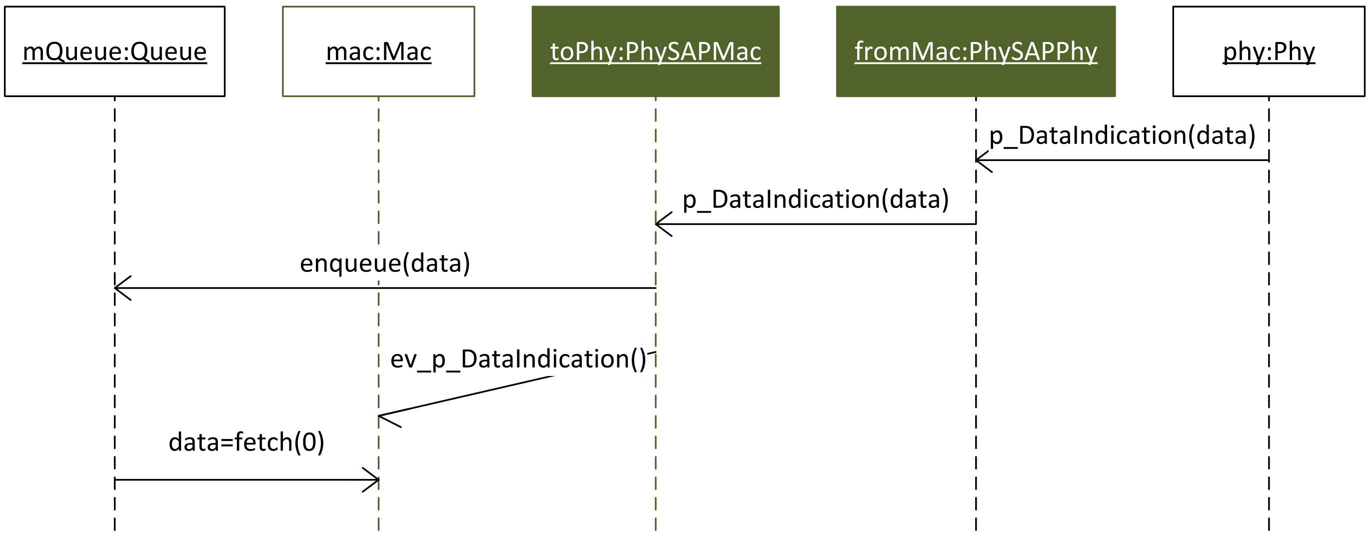


Layer decoupling

- toPhy abstracts link to phy for mac
- fromMac abstracts link to mac for phy
- Asynchronicity is managed on phy using the synchronous message ev_p_Request() and the queue pQueue



Decoupling (2)

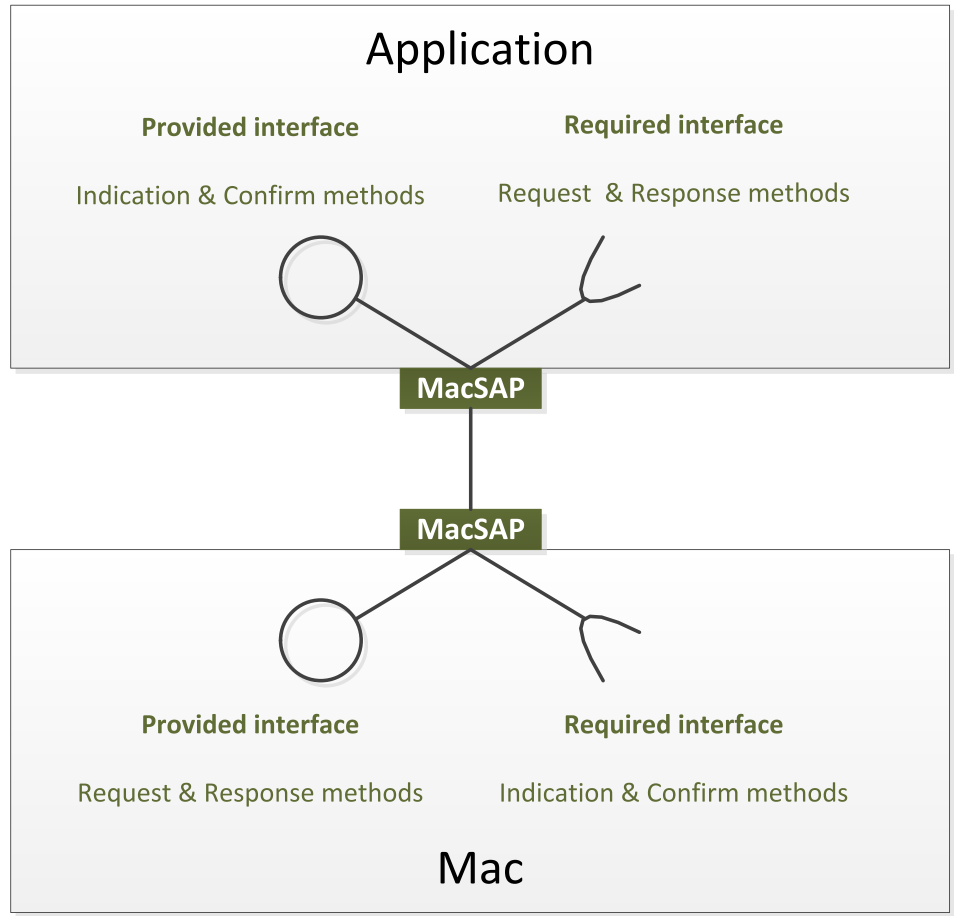


A first UML view of service primitives

OSI model operation

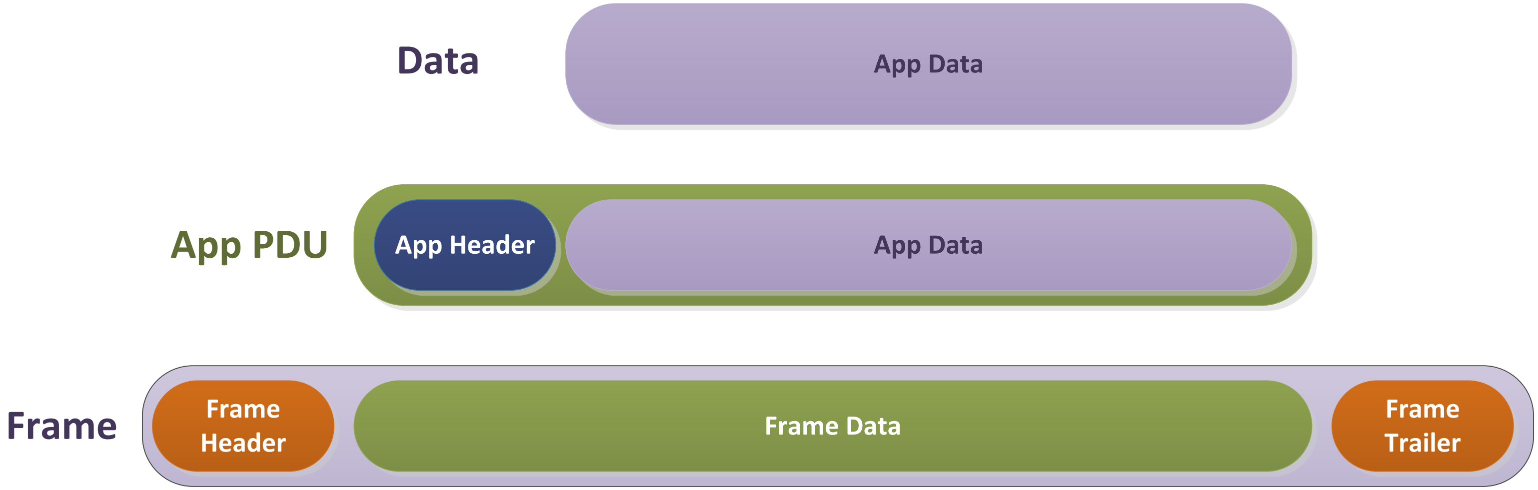
- “Higher” class
  - i.e. class whose instance is an upper layer protocol entity

  Implements:
  - Indication methods
    - Methods corresponding to service primitives of type “Indication”
  - Confirm methods
- “Lower” class implements:
  - Request methods
    - Methods corresponding to service primitives of type “Request”
  - Response methods



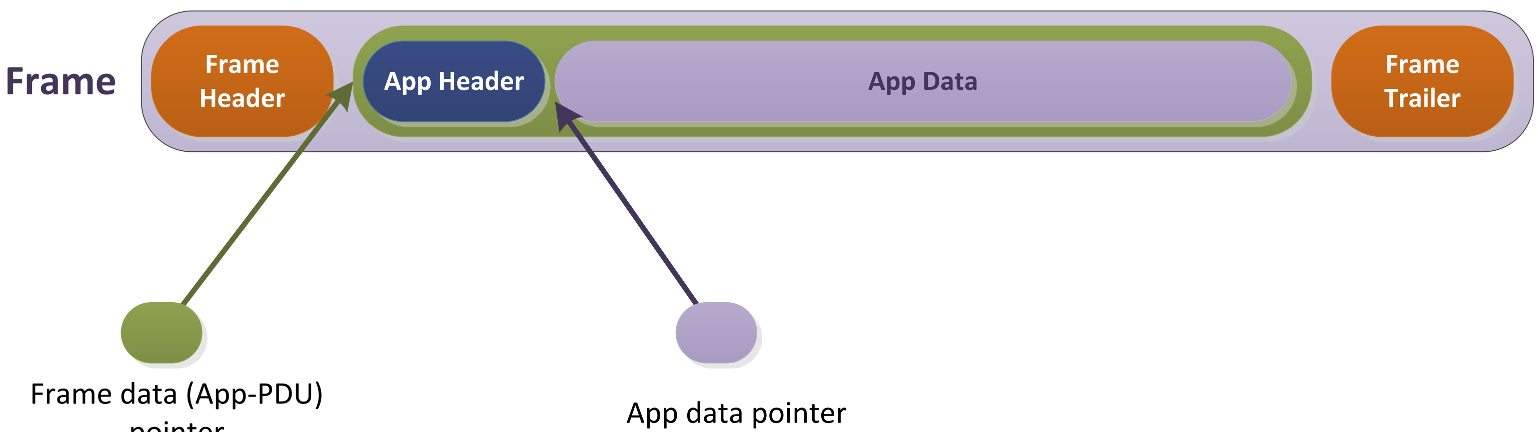
Memory strategy: copy of data

- Each layer has its own data objects
- Simple approach, close to model
- Tedious and memory consuming operation



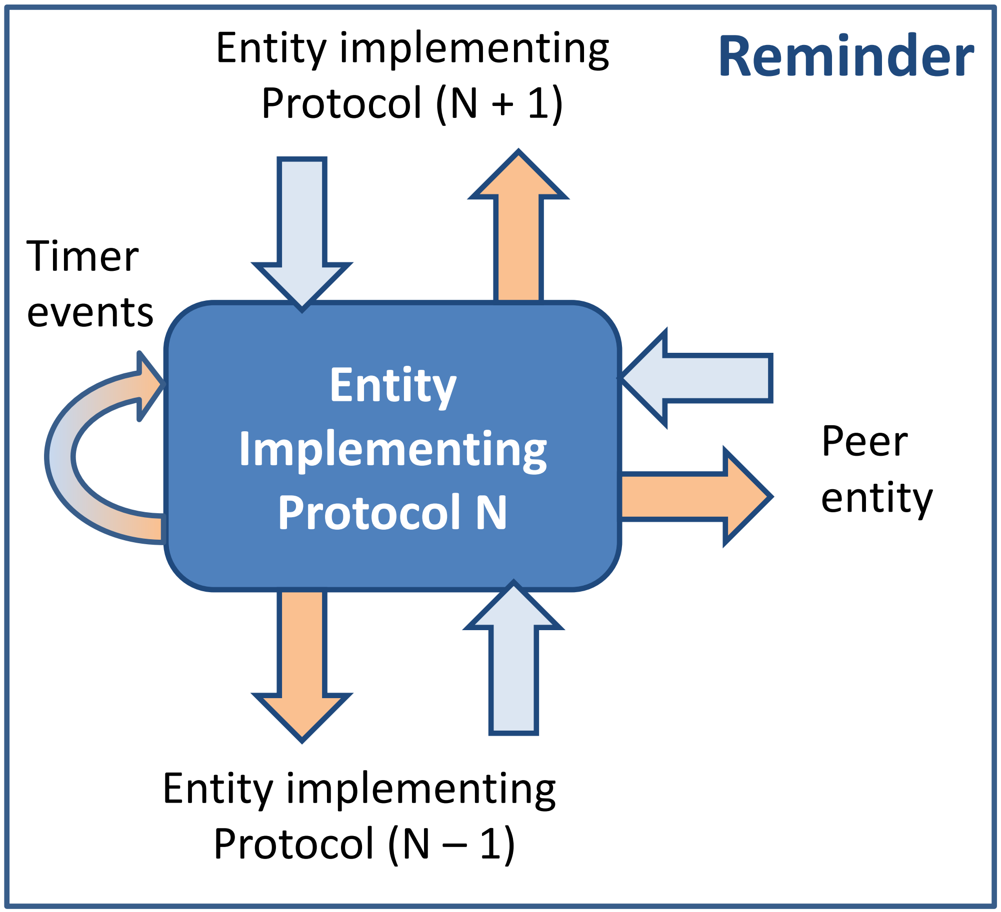
Memory strategy: reference to data

- Upper layer (outgoing data) and lower layer (incoming data) entities reserve memory for a full frame
  - An entity should free the reserved memory once the PDU is not used anymore
    - Usually but not always the "opposite" layer entity
- Offsets for the different PDUs are stored together with the frame itself
- More complex handling, but reduced memory requirements / processing time



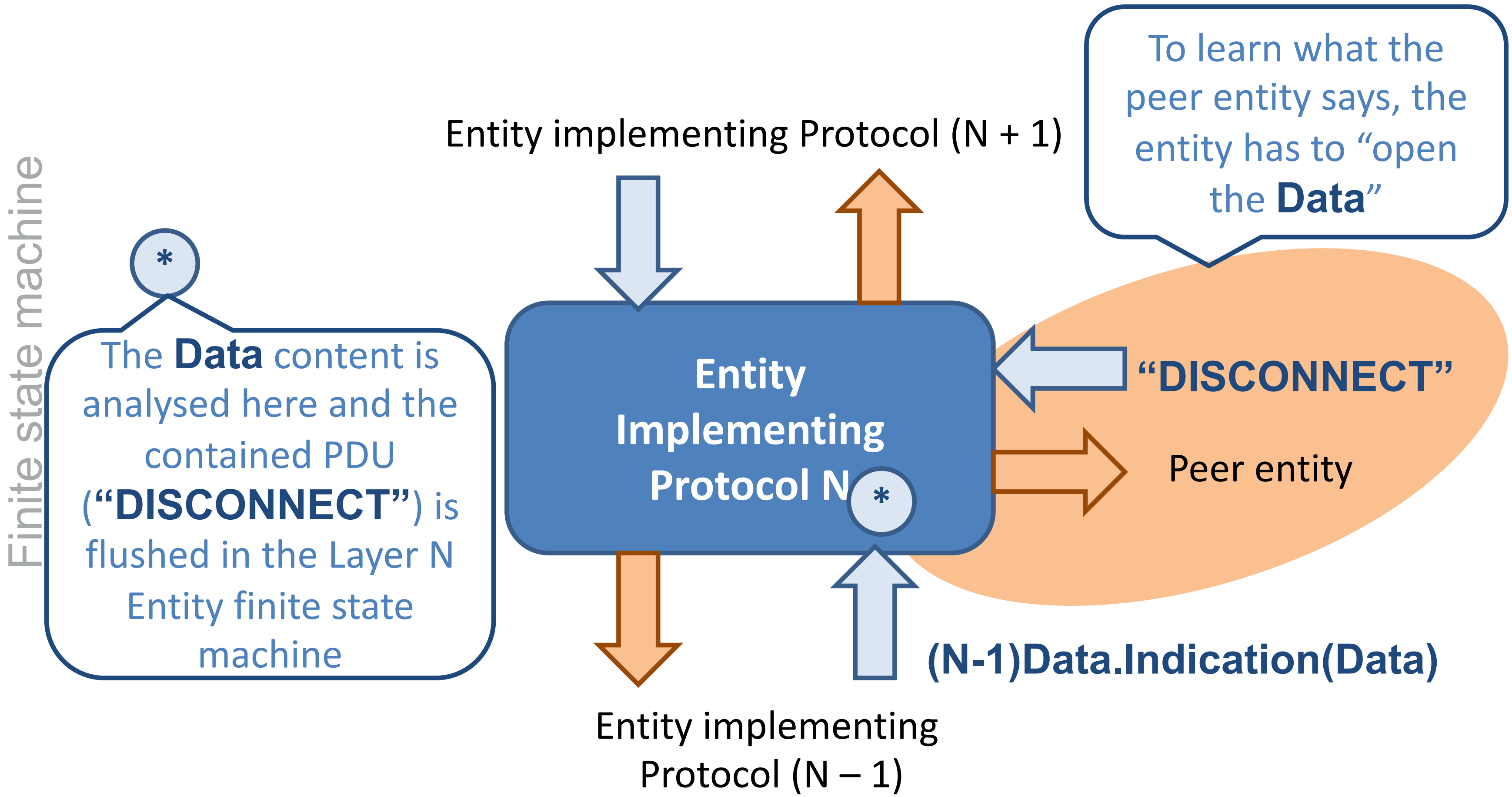
Protocol entity as a finite state machine

Finite state machine



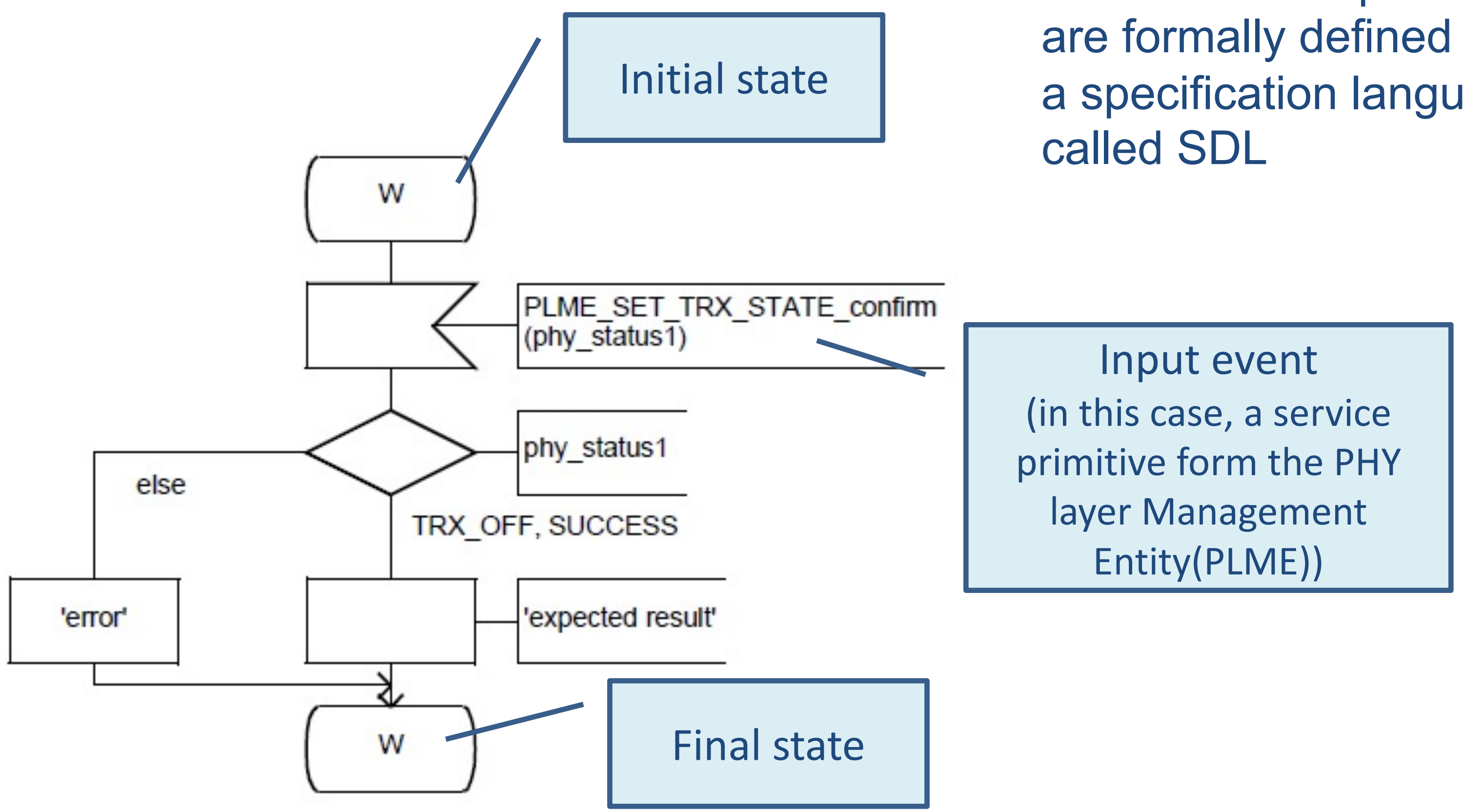
- Protocol definition contains a finite state machine definition
  - Sometimes in a very primitive form
  - Sometimes implicitly
- Timer timeouts are events generated by the entity on itself

Protocol entity as a finite state machine



An example based on IEEE 802.15.4

Finite state machine



- MAC and PHY protocols are formally defined using a specification language called SDL