Master ENGINEERING (MSE)

Hes·so VALAIS WALLIS

DeSEm
Layered Communication Architecture

Dominique Gabioud
Michael Clausen
Thomas Sterren
Medard Rieder

HES-SO 2021/22

Table of Content

- Introduction to layered architectures
- The OSI model
- OSI model layers
- OSI model operation
- Example based on IEEE802.15.4
- Implementation strategies

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”

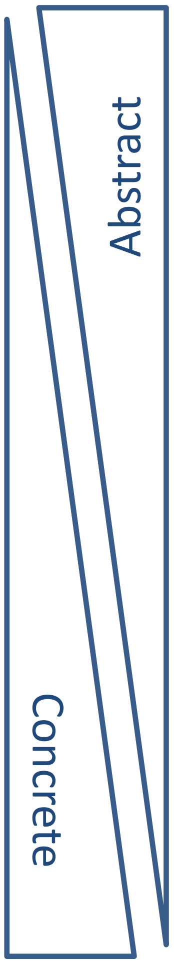
“Select an appropriate land” — “An economist & a geographer”

“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”



Services

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

Service user — “Make detail plan for HVAC”

Interface

Service: Deploy an HVAC system as defined on a plan

Service provider — “Install water circuit for heating”

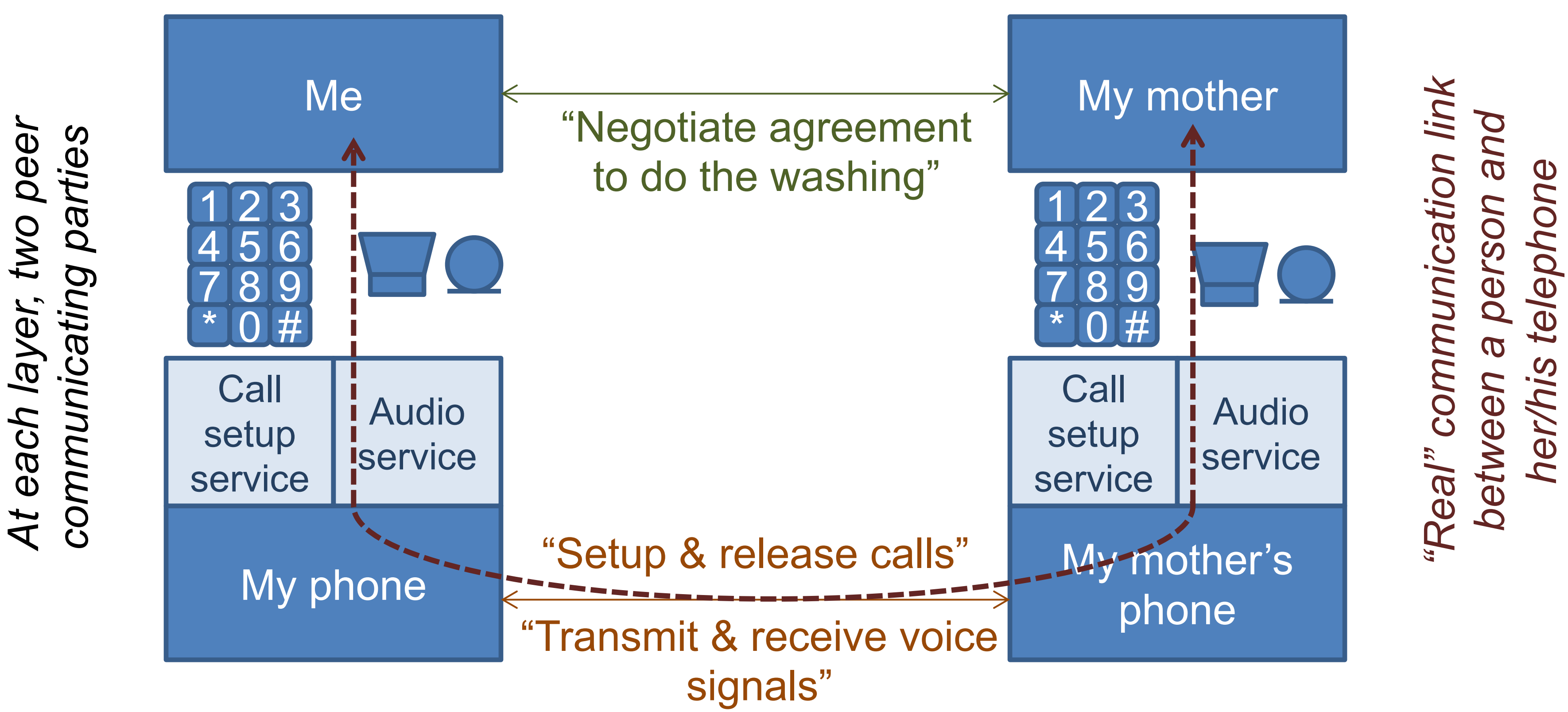
The service abstracts some activity for the service user

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



"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:

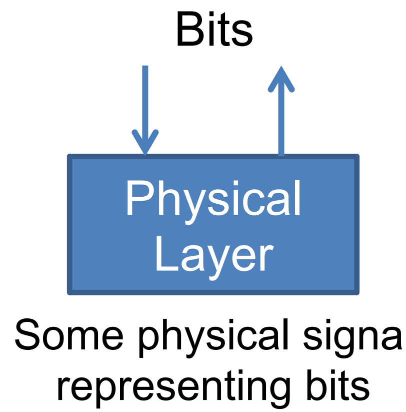
Provide tools to promote interoperability without limiting innovation

Physical & Data Link layers

OSI model layers

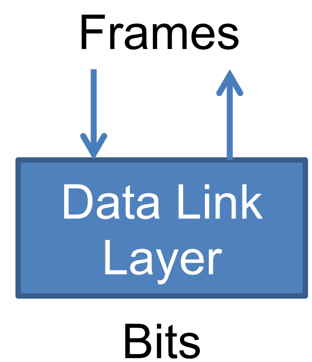
- Physical layer

  Transmit & receive bits



- Data Link layer

  Exchange frames with local station



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

Network & Transport layers

OSI model layers

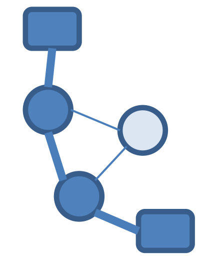
- Network layer

Exchange packets over a network

Packets

Network Layer

Frames



Finding a route in a meshed network

- Transport layer

Provide a reliable communication channel between two applications

Packets, streams

Transport Layer

Packets

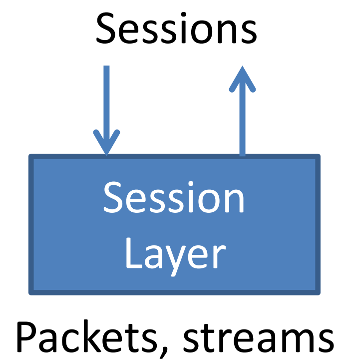
Internal application sub-addressing, end-to-end error control

Session & Presentation layers

OSI model layers

- Session layer

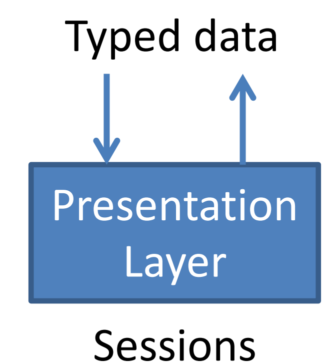
Maintain service, possibly over several Transport channels



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

- Presentation layer

Represent typed data in an agreed format

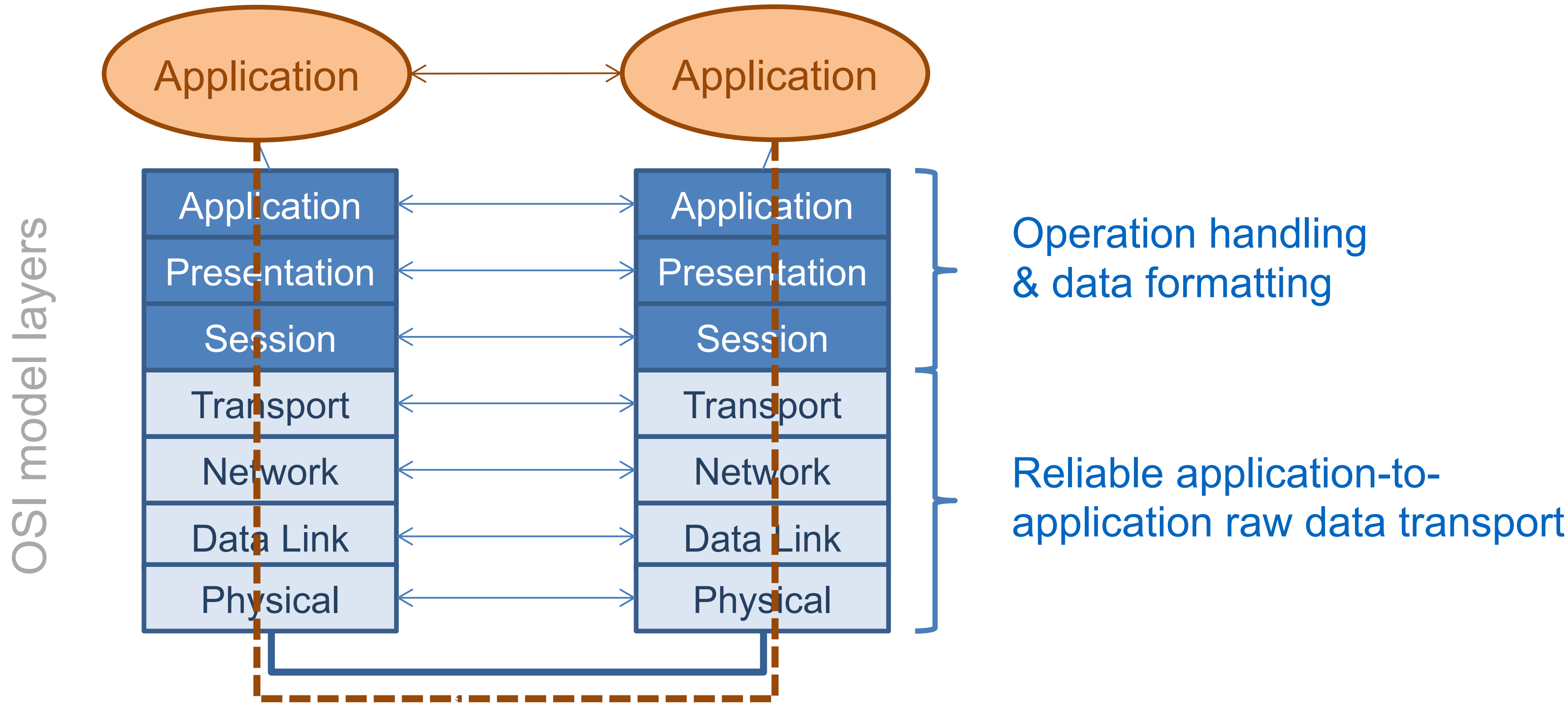


Negotiate common format, perform required translation, implement ciphering & authentication

Application Layer

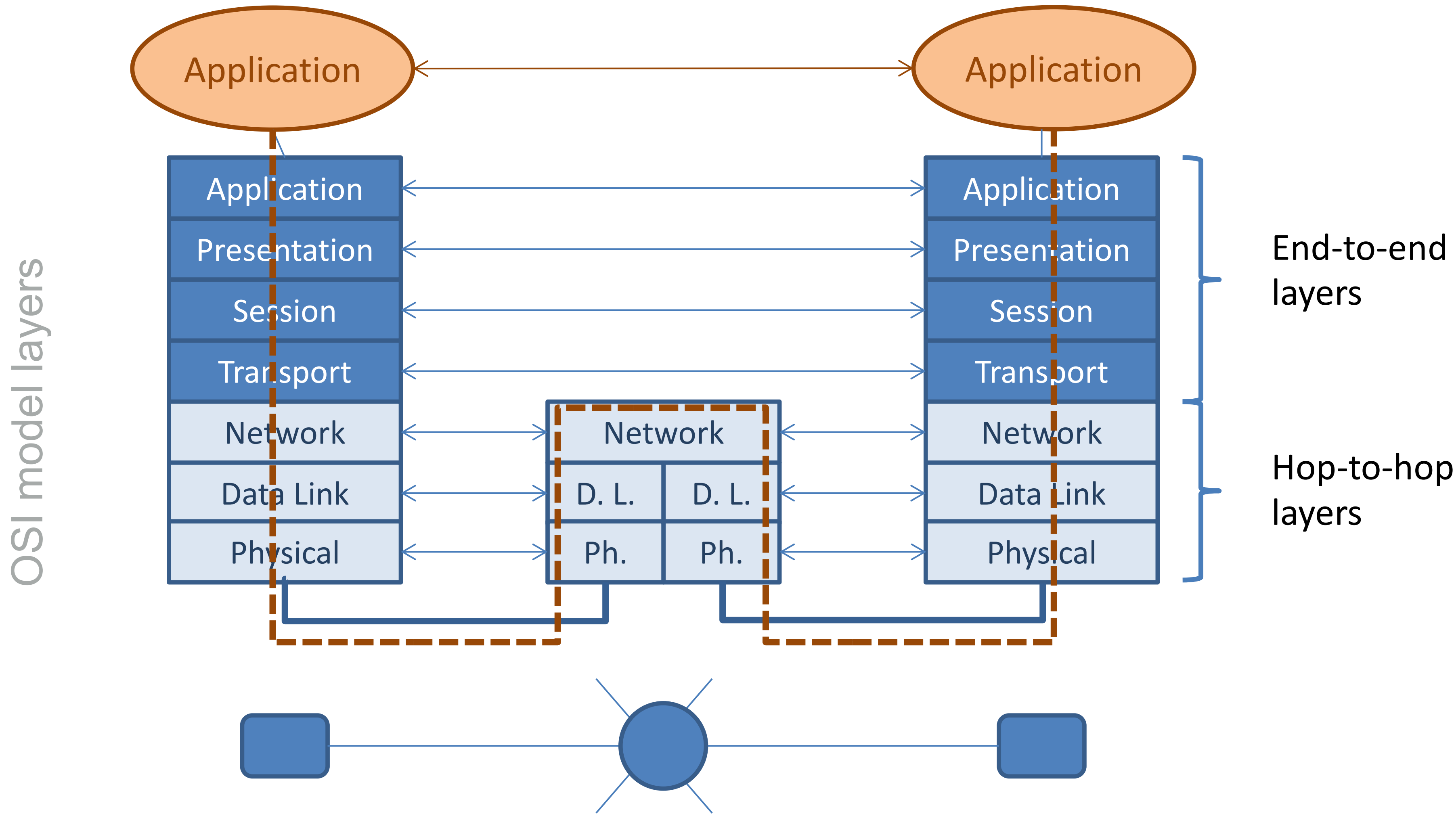
- Application layer
  - Set of specific services
    - File transfer, file sharing, document retrieval, remote access to local inputs & outputs, remote terminal, instant messaging…

Overview of OSI Layers



Abstract "horizontal" communication
Real "vertical" communication

OSI layers & communication networks



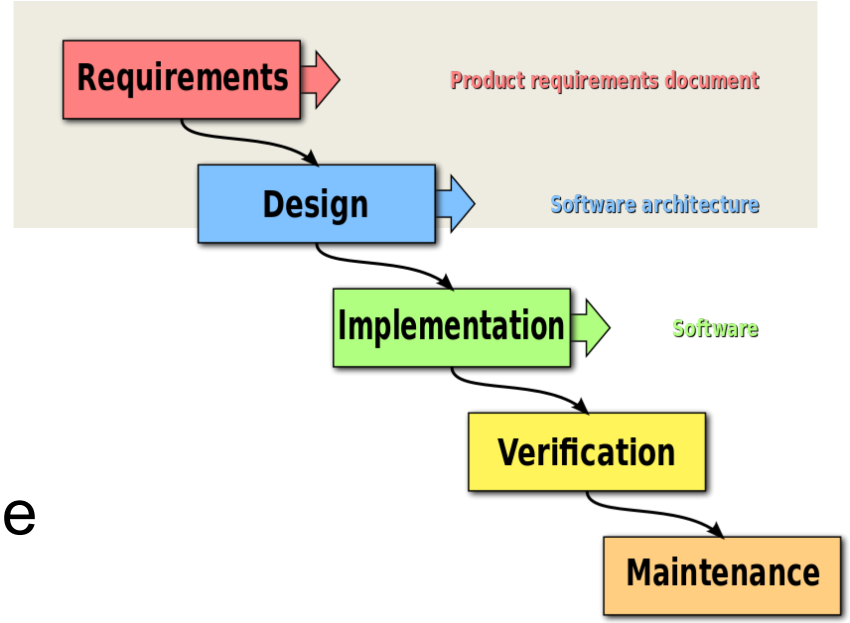
Concluding remarks

- 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

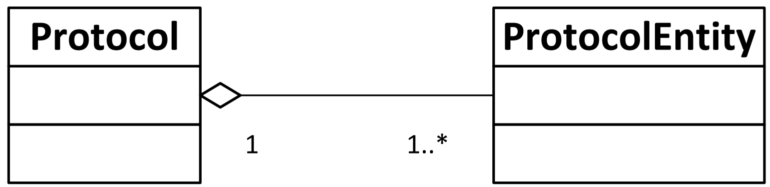


[Waterfall model by Peter Kemp / Paul Smith – Adapted from Paul Smith's work]

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

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

Protocol vs. protocol entity

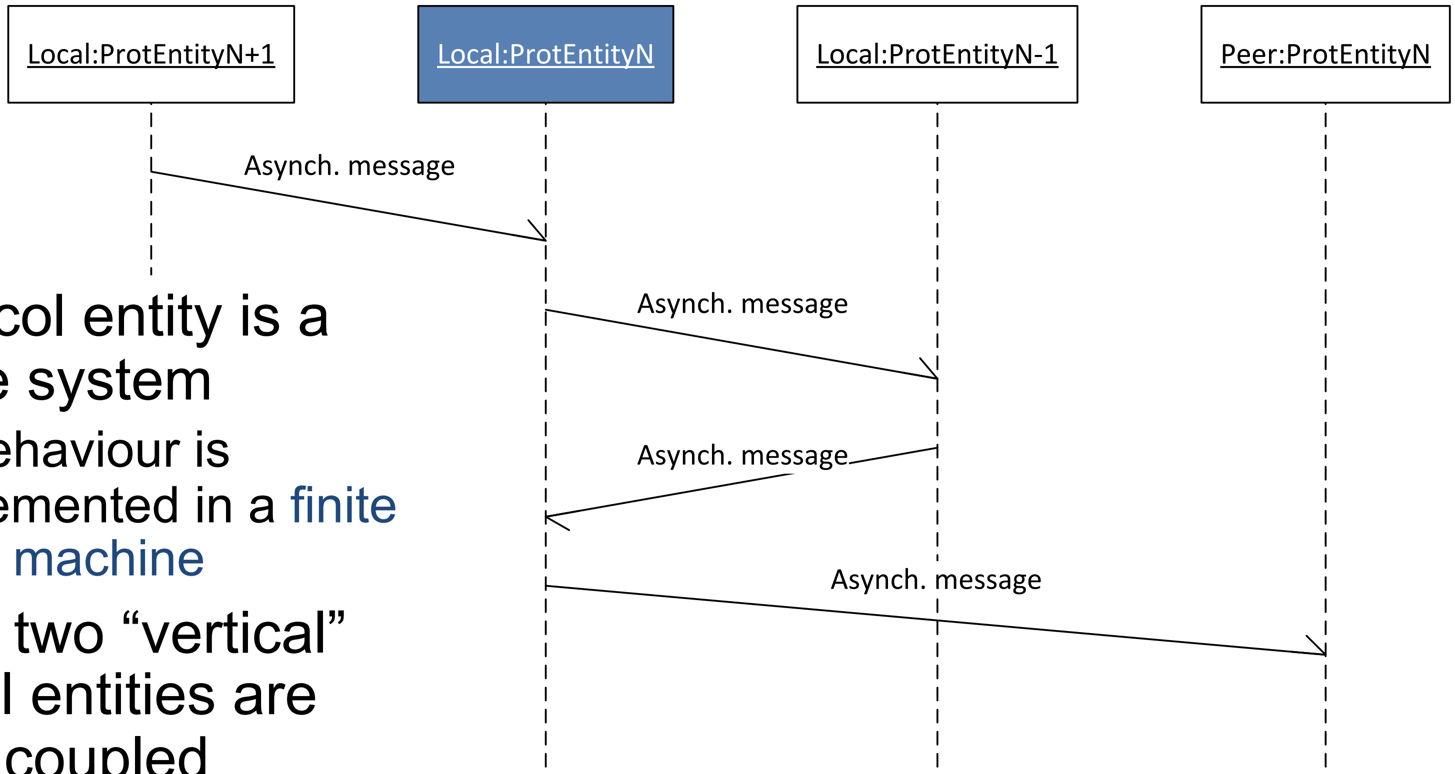


OSI model operation

- 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

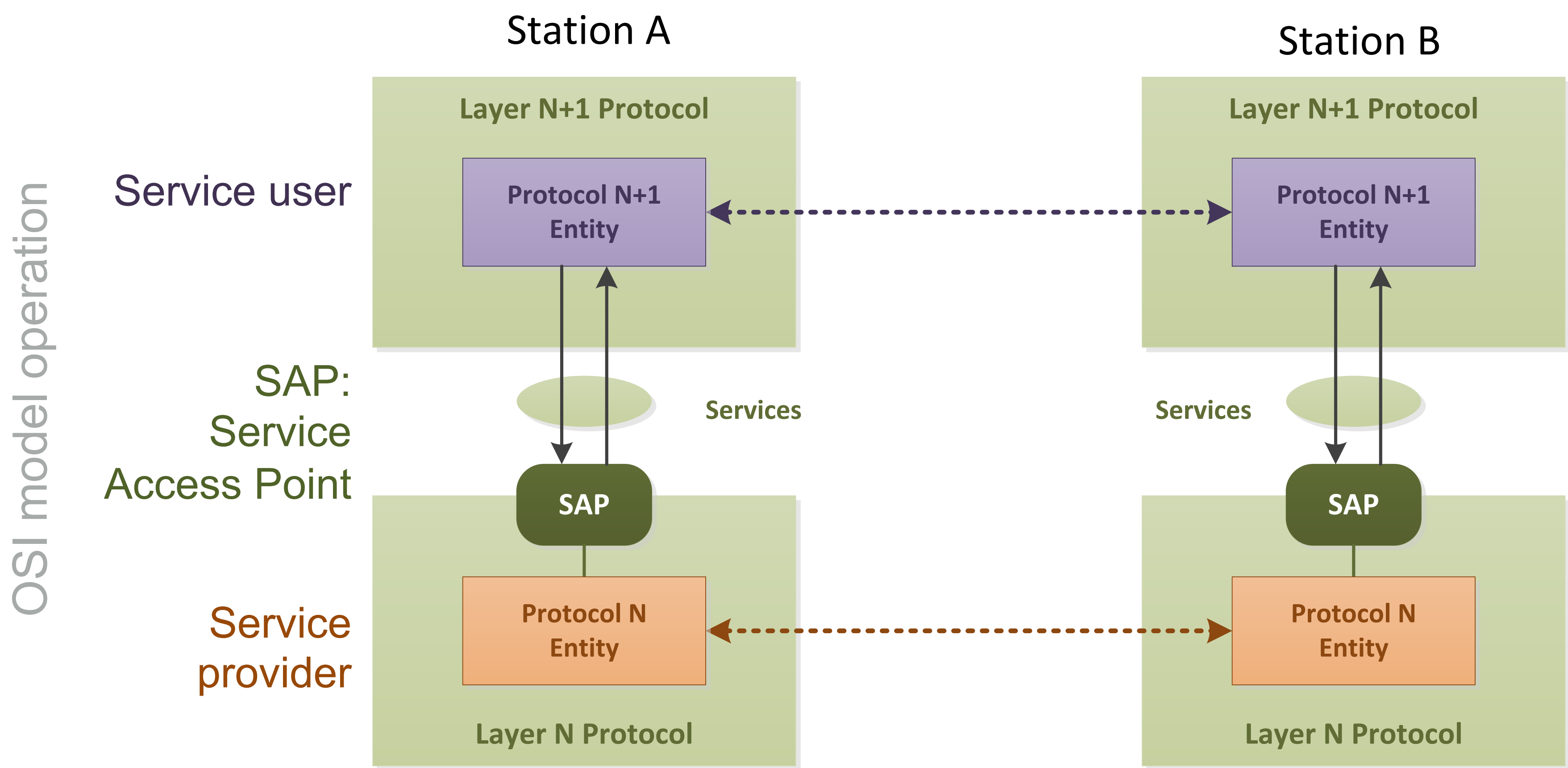
Protocol entity as a Finite State Machine



OSI model operation

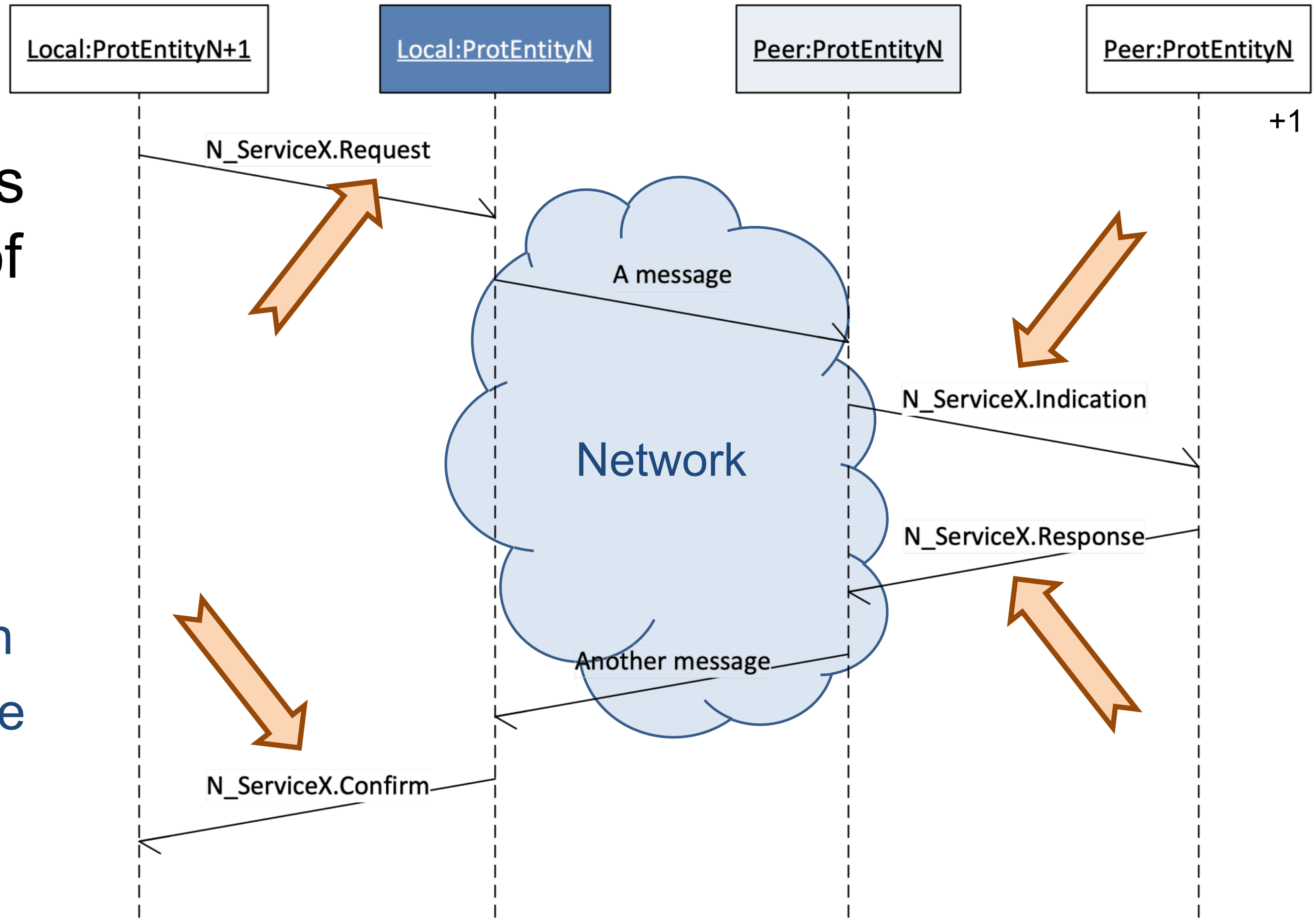
- A protocol entity is a reactive system
  - Its behaviour is implemented in a finite state machine
- Usually two "vertical" protocol entities are loosely coupled
  - Asynchronous communication

Services in the OSI model



Service & service primitives

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



Service primitives

OSI model operation

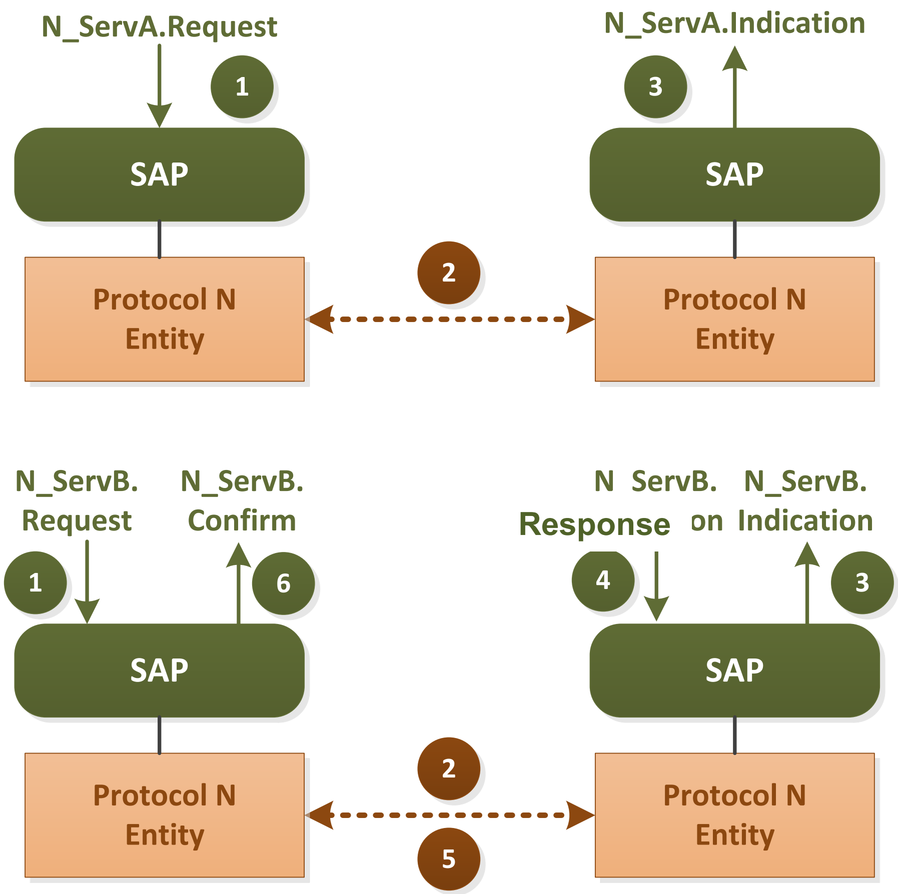
- Request
  - Primitive used by a service user to request a service
- Indication
  - Primitive used by a service provider to indicate to a service user that a peer service user has requested a service
- Response
  - Primitive used by a service user to respond to an indication primitive
- Confirm
  -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

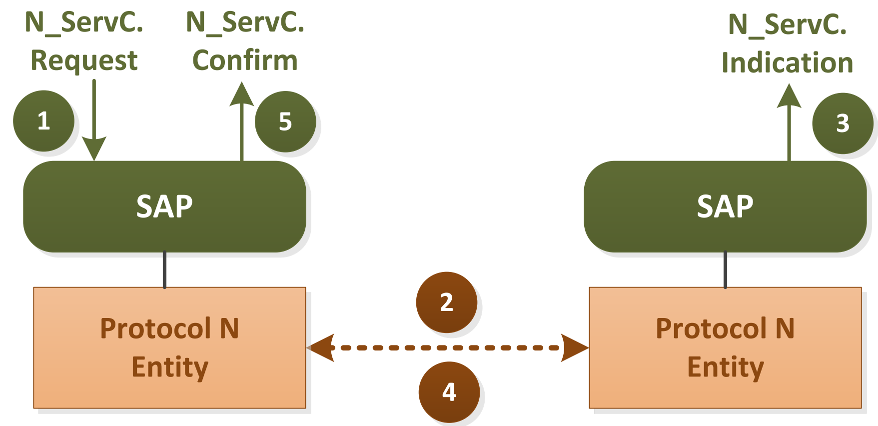
"**ServB**" is a service confirmed by peer



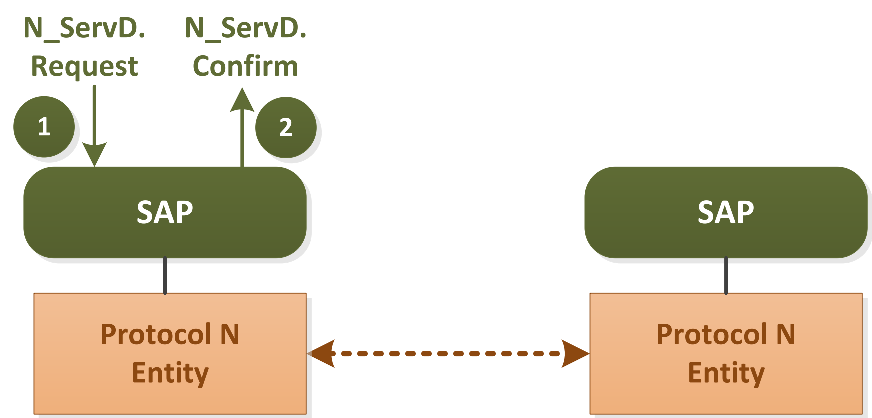
Confirmed & unconfirmed services

OSI model operation

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

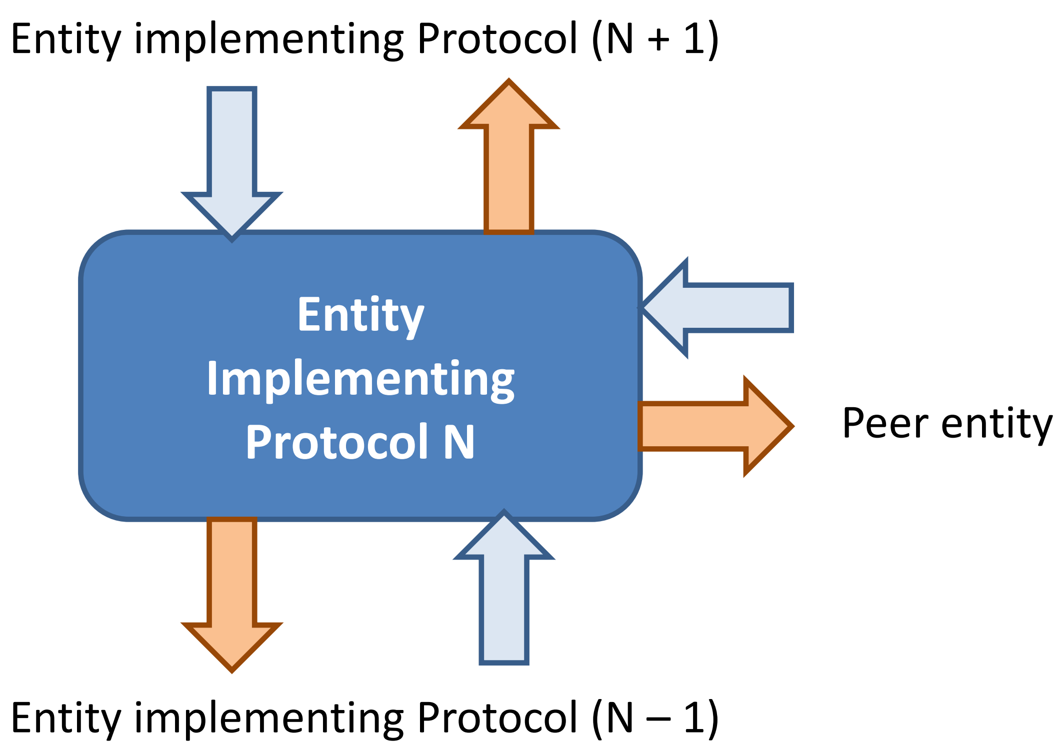


"**ServD**" is a locally confirmed service (the service modifies some internal parameter of the Protocol N Entity)

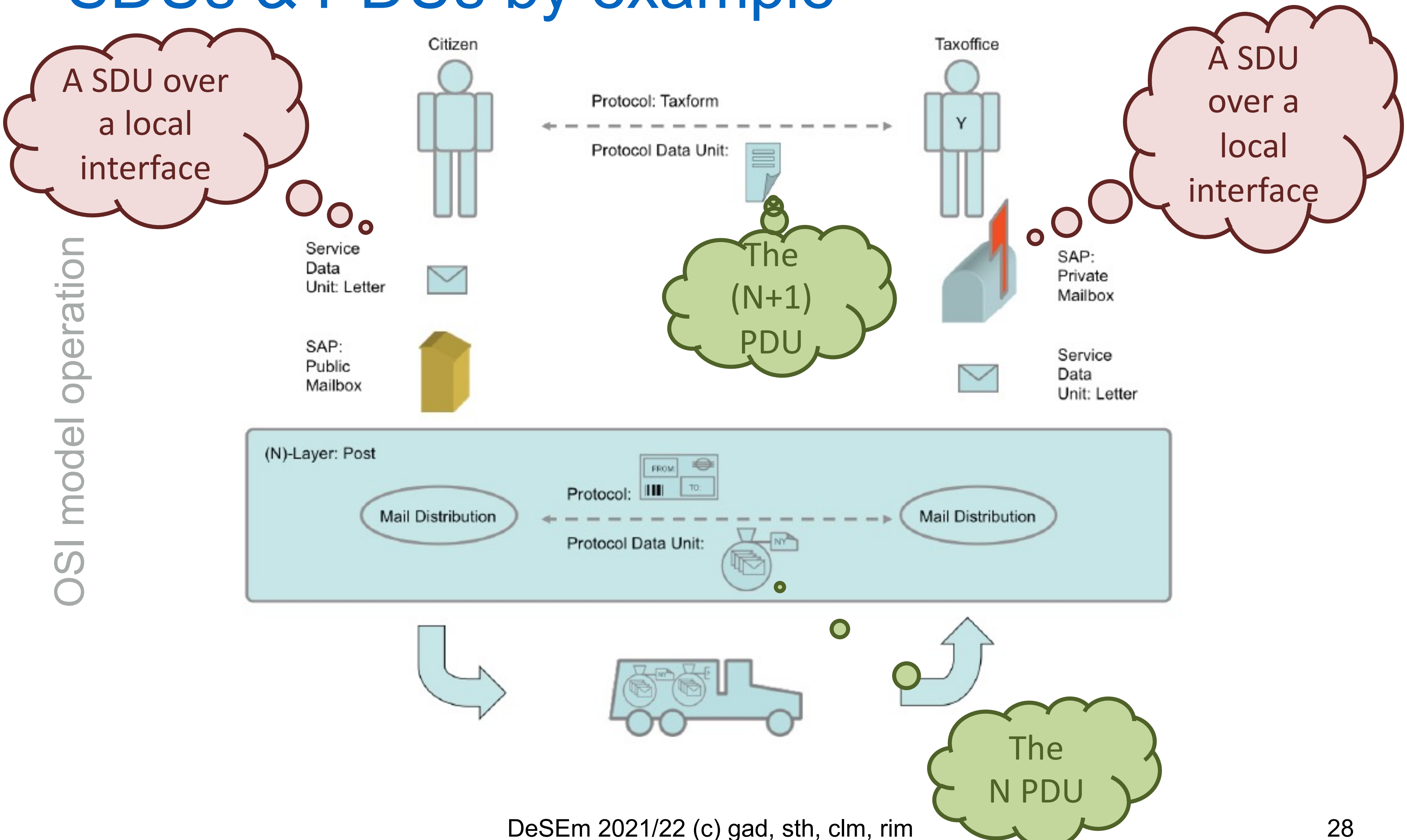


Protocol entity as a reactive system

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

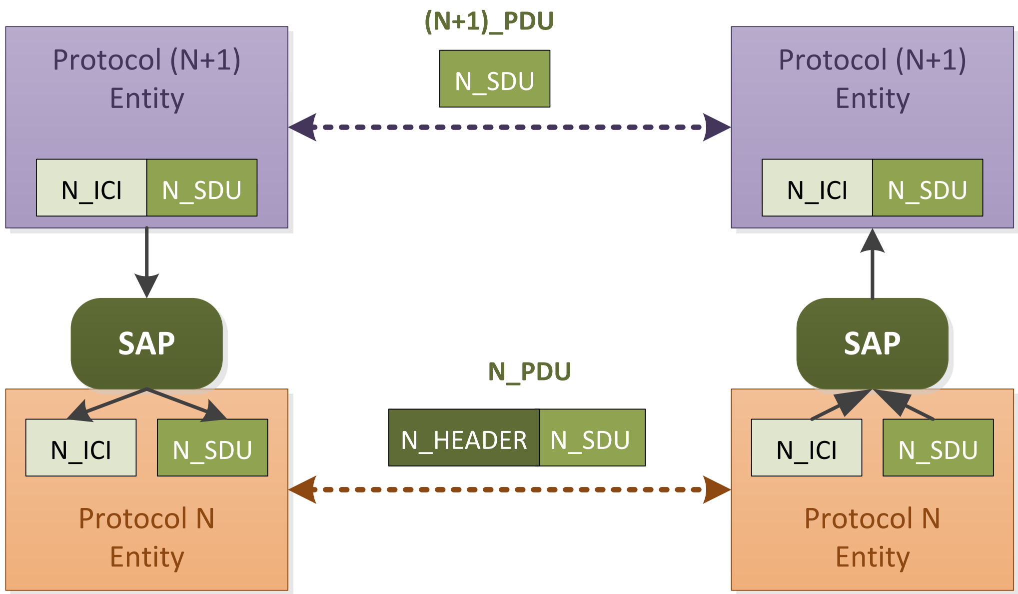


SDUs & PDUs by example



SDUs & PDUs: Functioning

OSI model operation



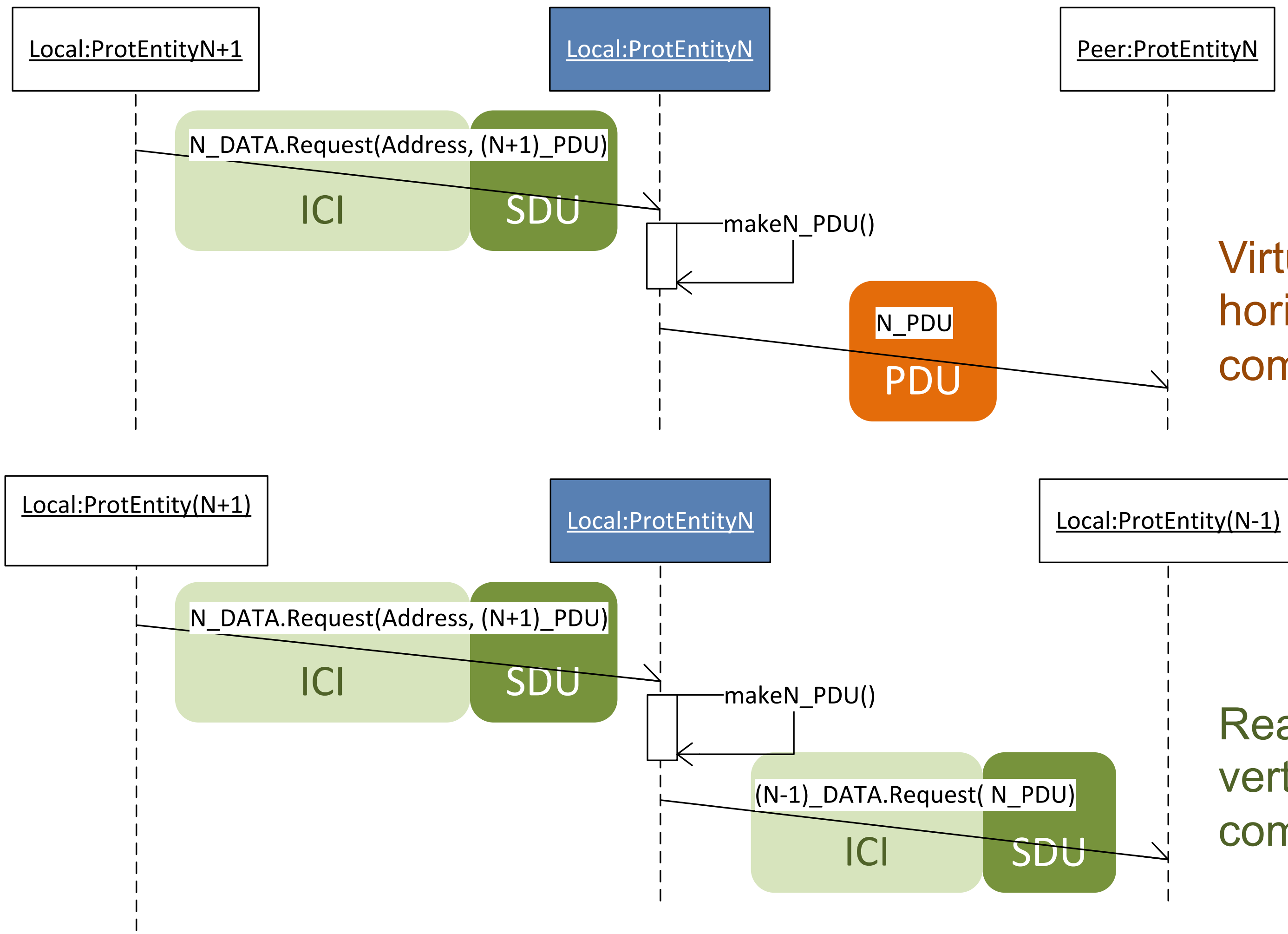
IDU Interface Data Unit
ICI Interface Control Information
PDU Protocol Data Unit
SDU Service Data Unit

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

SDU(N) = PDU(N + 1)

SDUs & PDUs : Interaction

OSI model operation



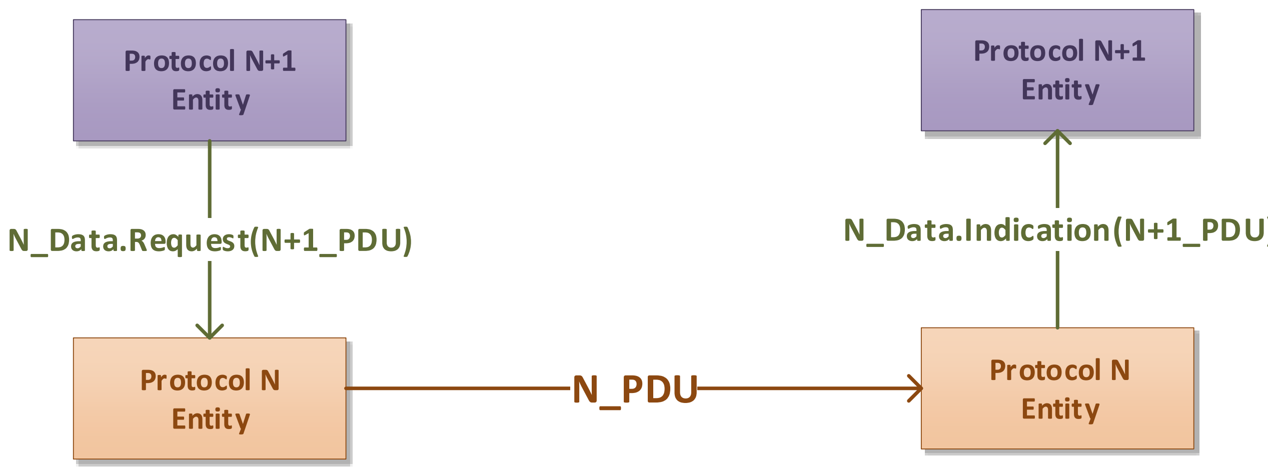
Virtual horizontal communication

Real vertical communication

Real communication vs. virtual communication in the OSI model

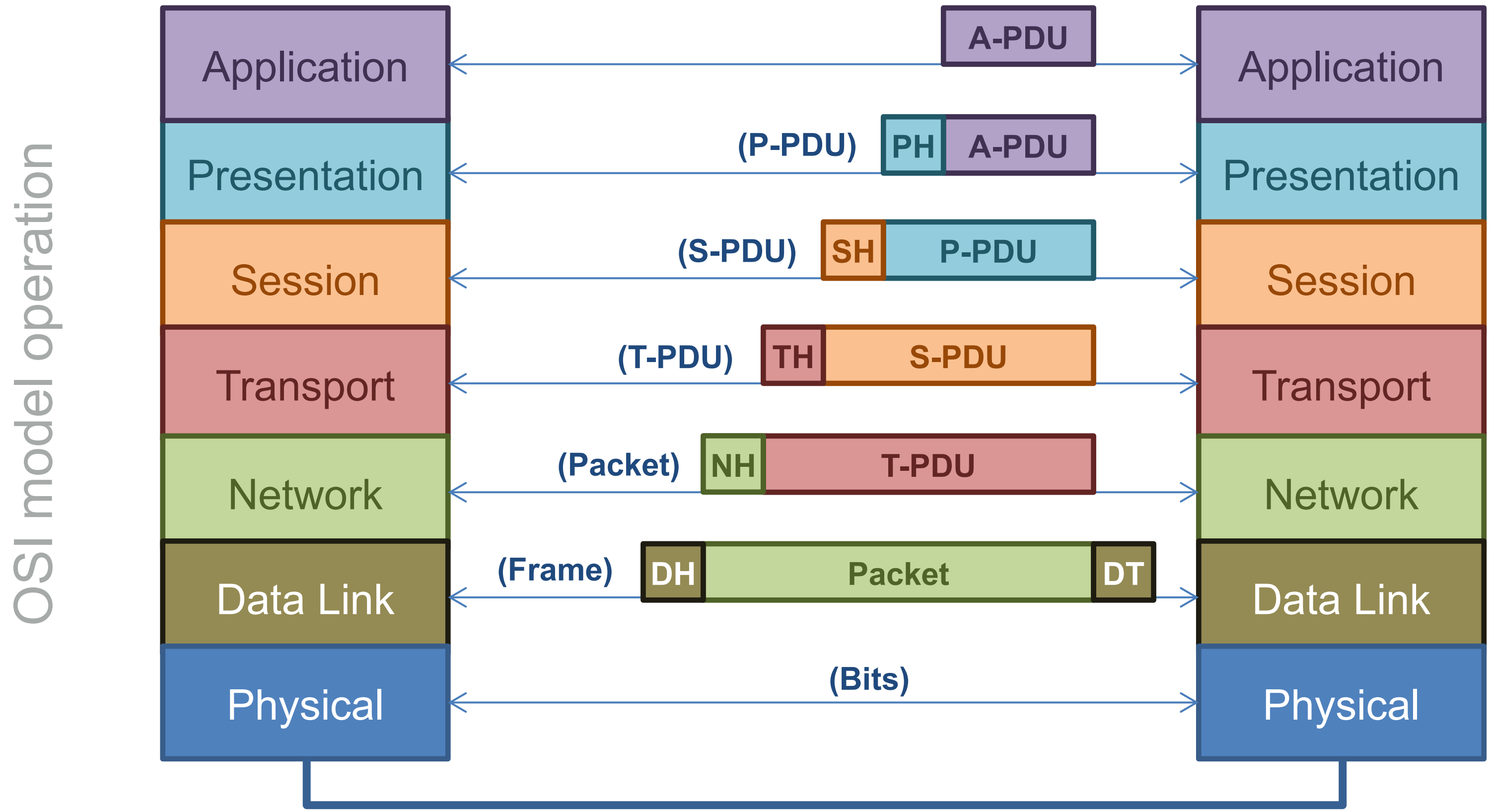
OSI model operation

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



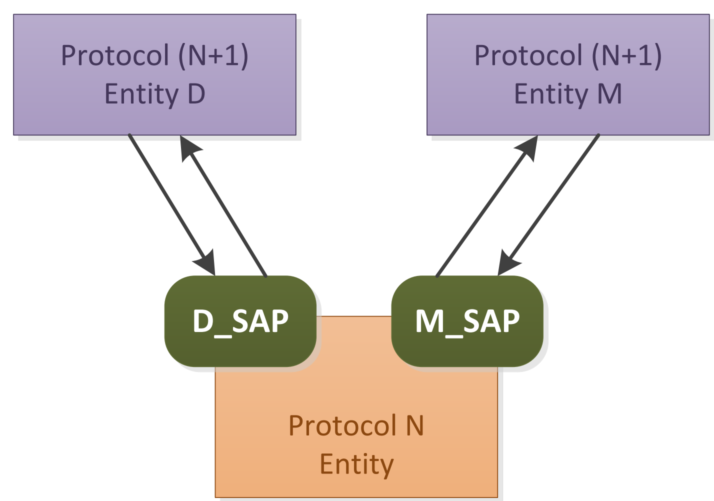
From a logical standpoint we consider that the N+1 protocol entity receives directly the PDU from tis peer entity

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

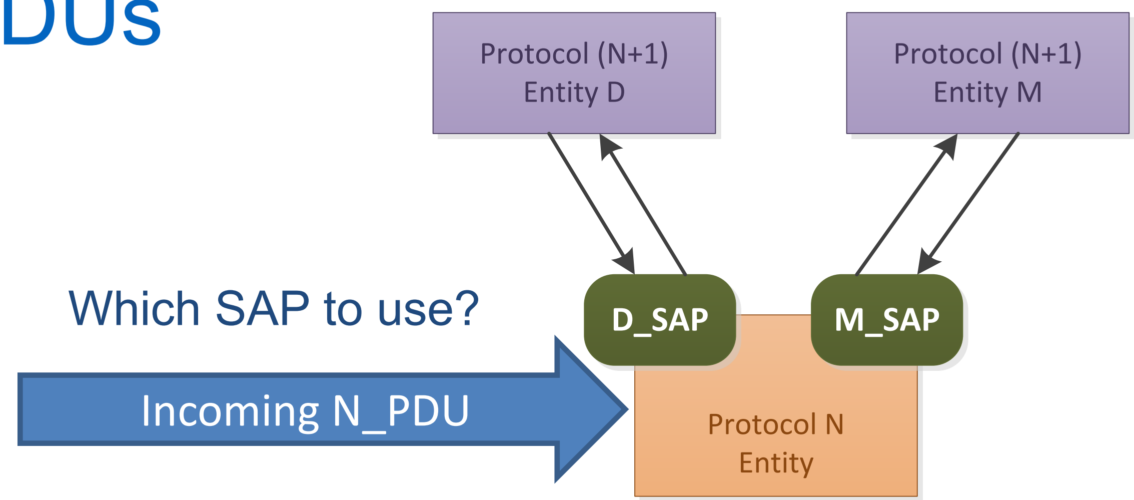
SAPs & Protocol entities

- 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

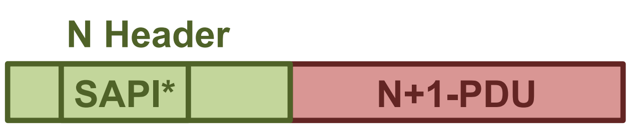


OSI model operation

SAPIs & PDUs

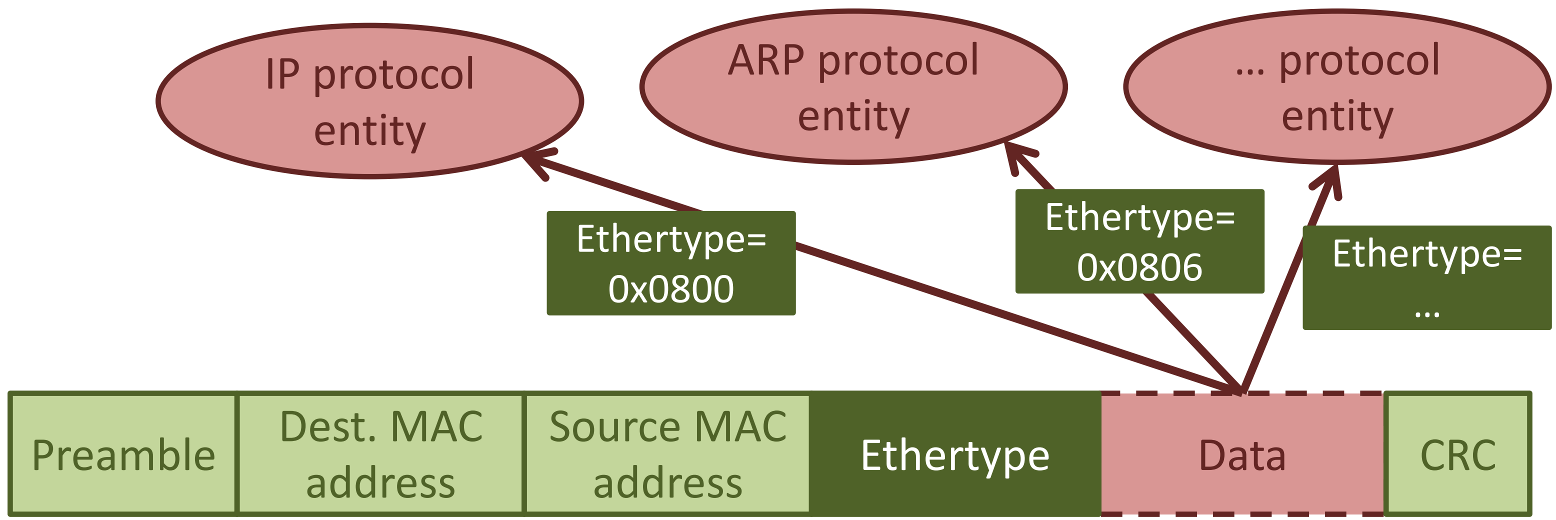


- The N-PDU contains a reference to a SAP
- This reference is called SAPI: SAP Identifier (usually a number)
- The SAPI is transmitted in a dedicated field of the N_PDU



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

SAPs & SAPIs: an Example



N+1 PDU

OSI model operation

Same Ethertype SAPI used for both communicating protocol entities

Central registry of Ethernet SAPIs to avoid “collisions”

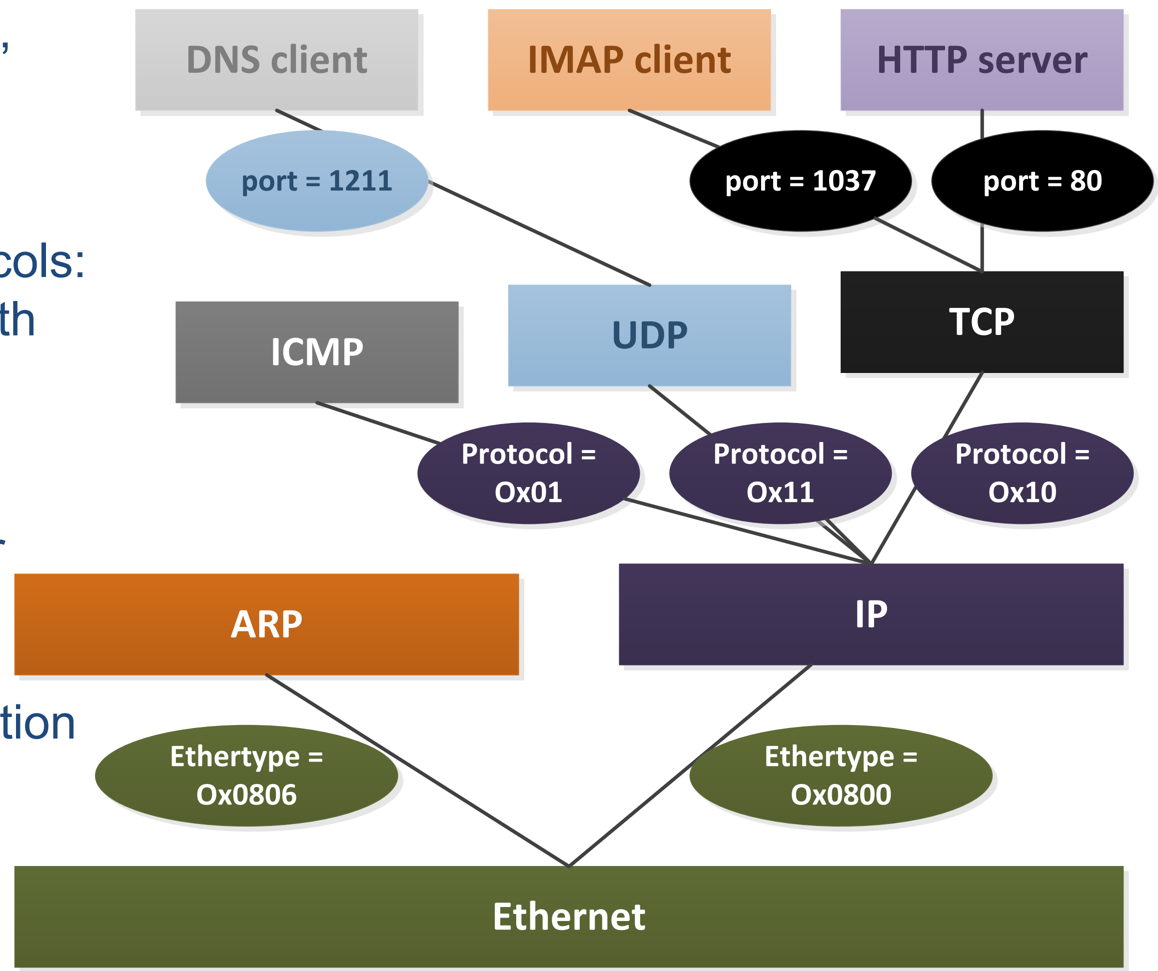
N+1 Protocol entity is identified by the SAPI Ethertype.

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

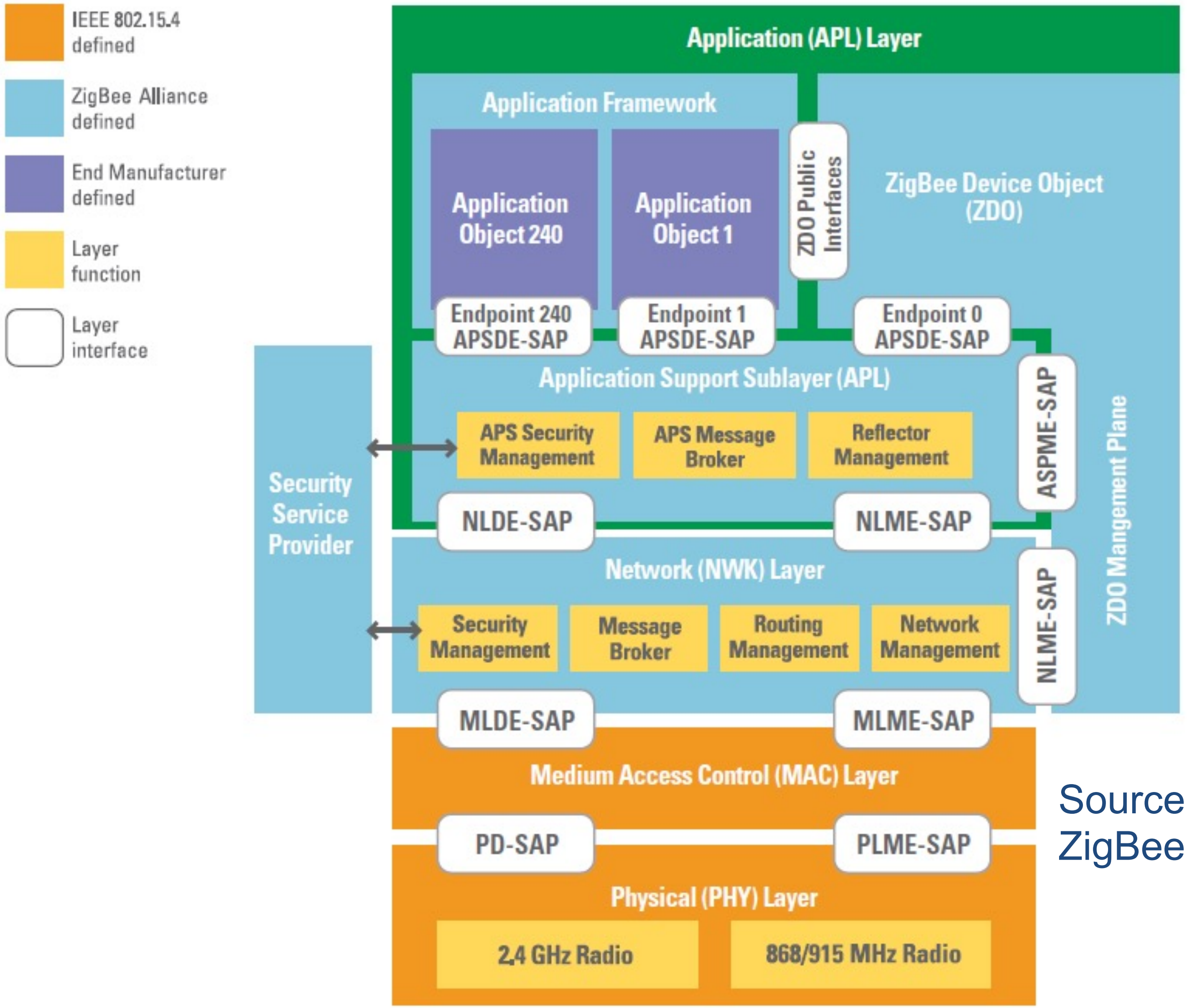
- Ethertypes, IP Protocols, TCP/UDP ports are examples of SAPIs
- Ethertypes and IP protocols: Unique SAPI field for both entities
- TCP/UDP ports:
  - Source port number is a source SAPI
  - Destination port number is a destination SAPI
  - TCP/UDP PDUs contains source & destination port numbers / SAPIs



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

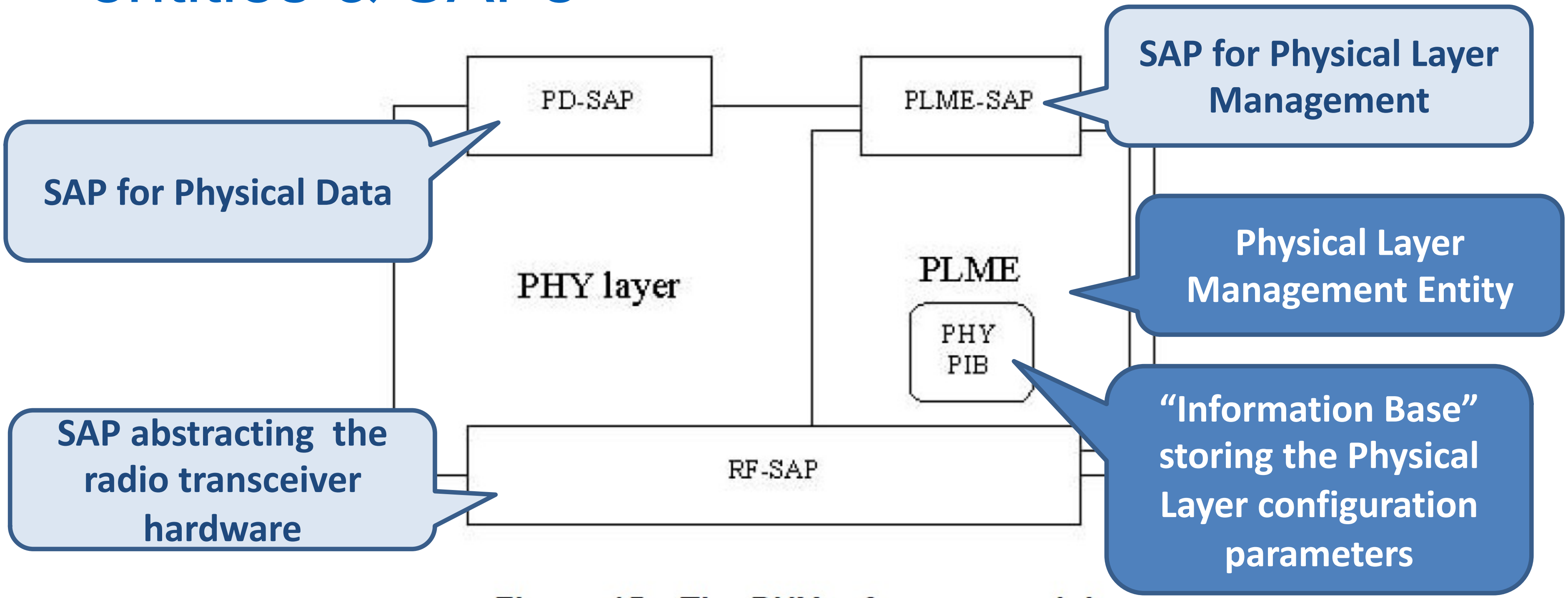


Figure 15—The PHY reference model

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.

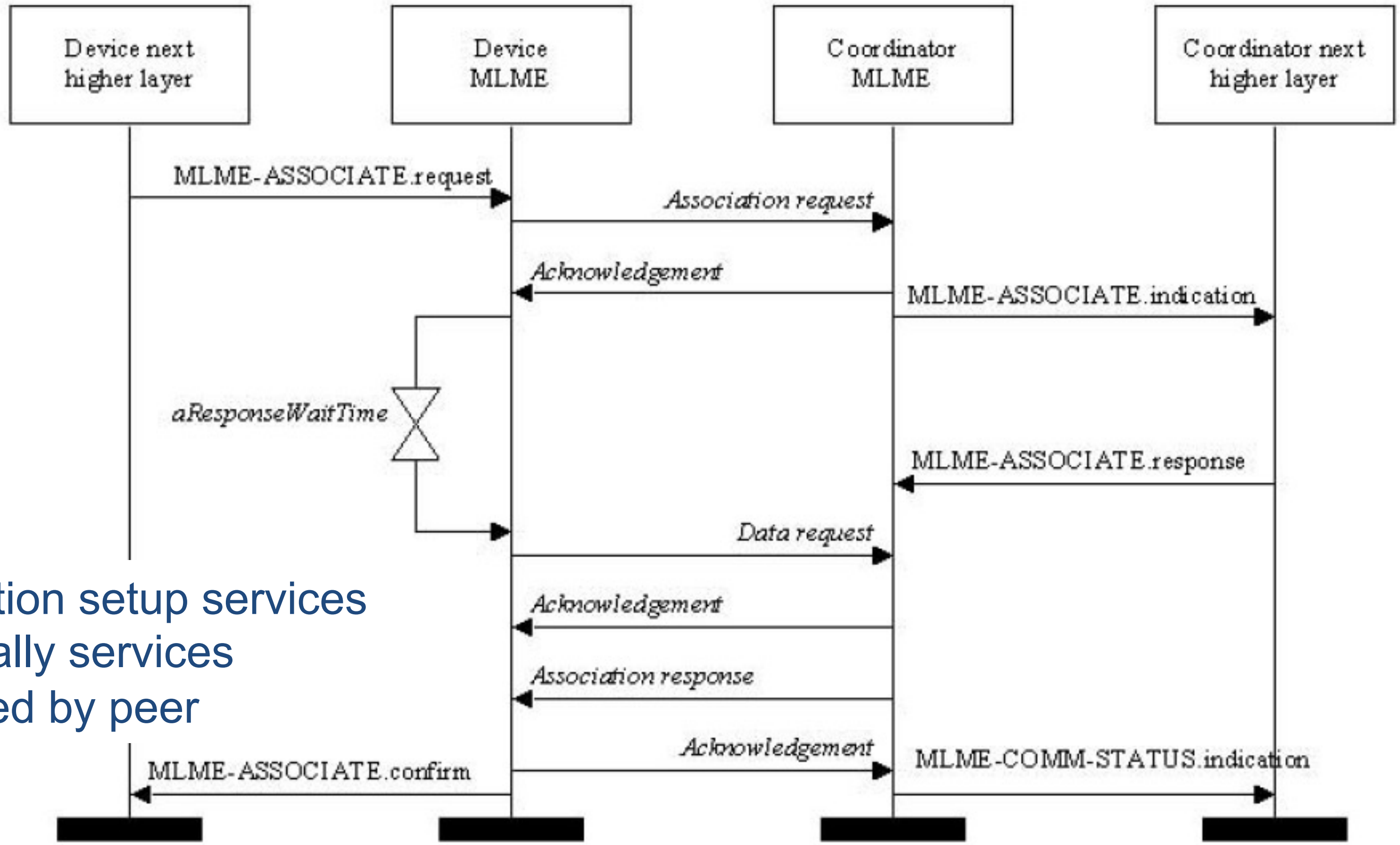
Table 5—PD-DATA.confirm parameters

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

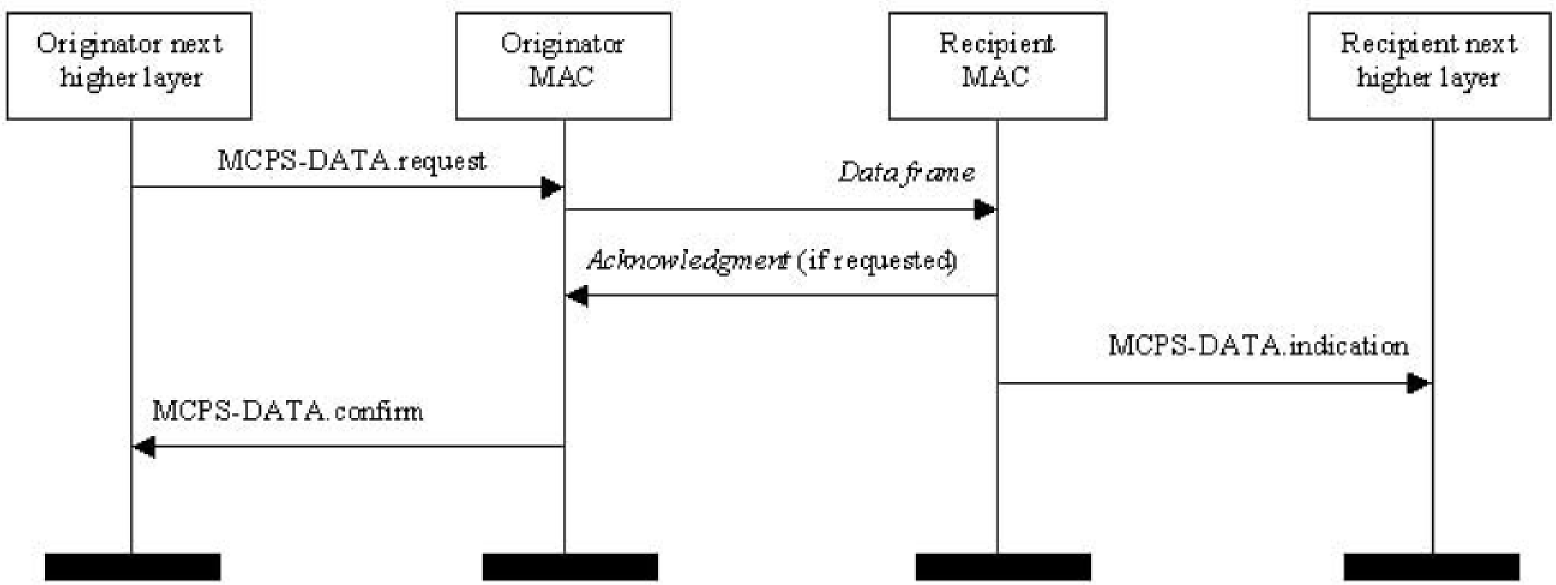
- The PLME-SAP supports management services
- Example of functions implemented by management services:
  - Select a radio channel and a transmit power
  - Turn RX radio on or off
  - Check whether the radio channel is busy
- All PLME-SAP services are locally confirmed services

Example of a service confirmed by peer

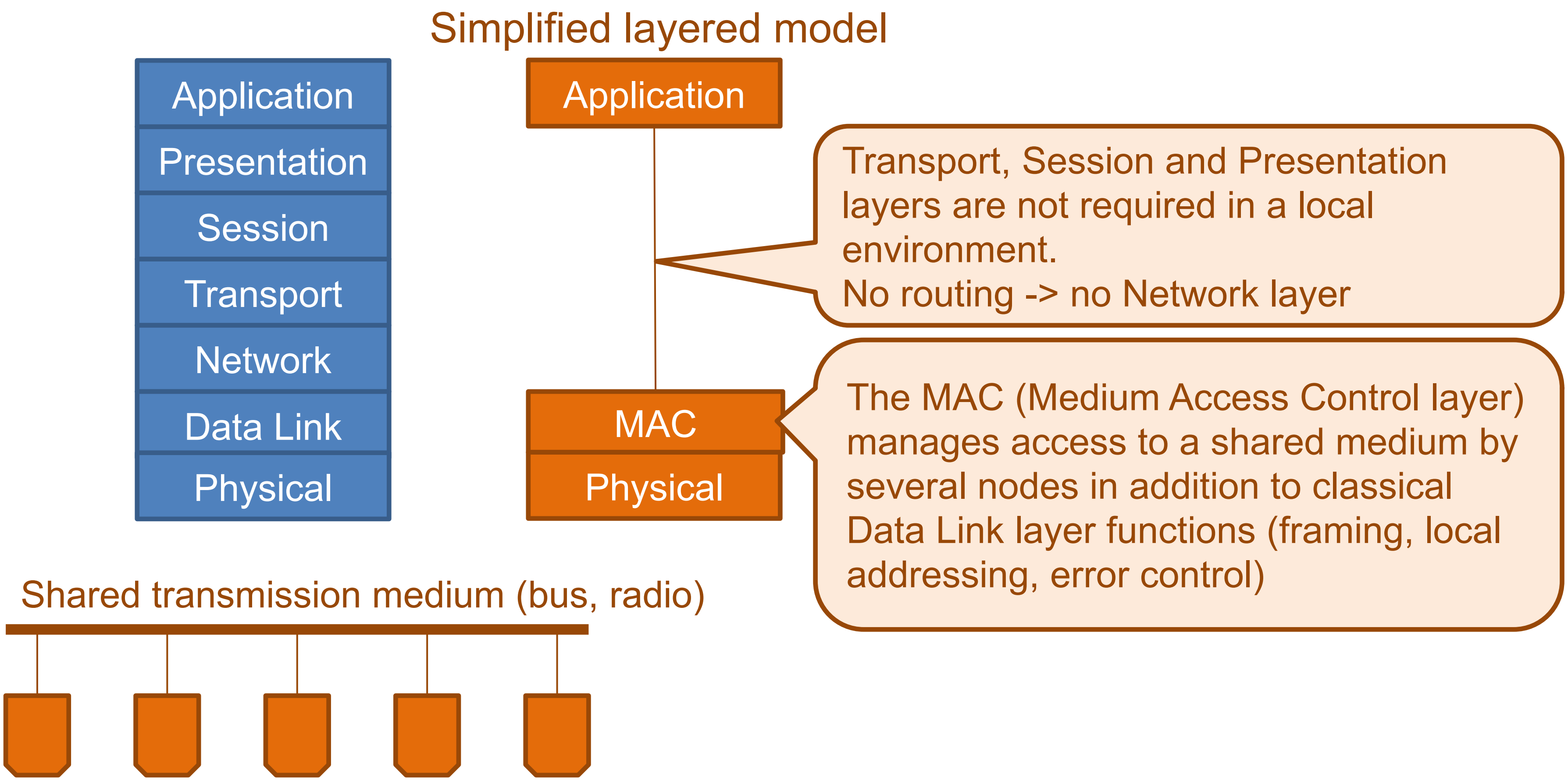


Connection setup services are usually services confirmed by peer

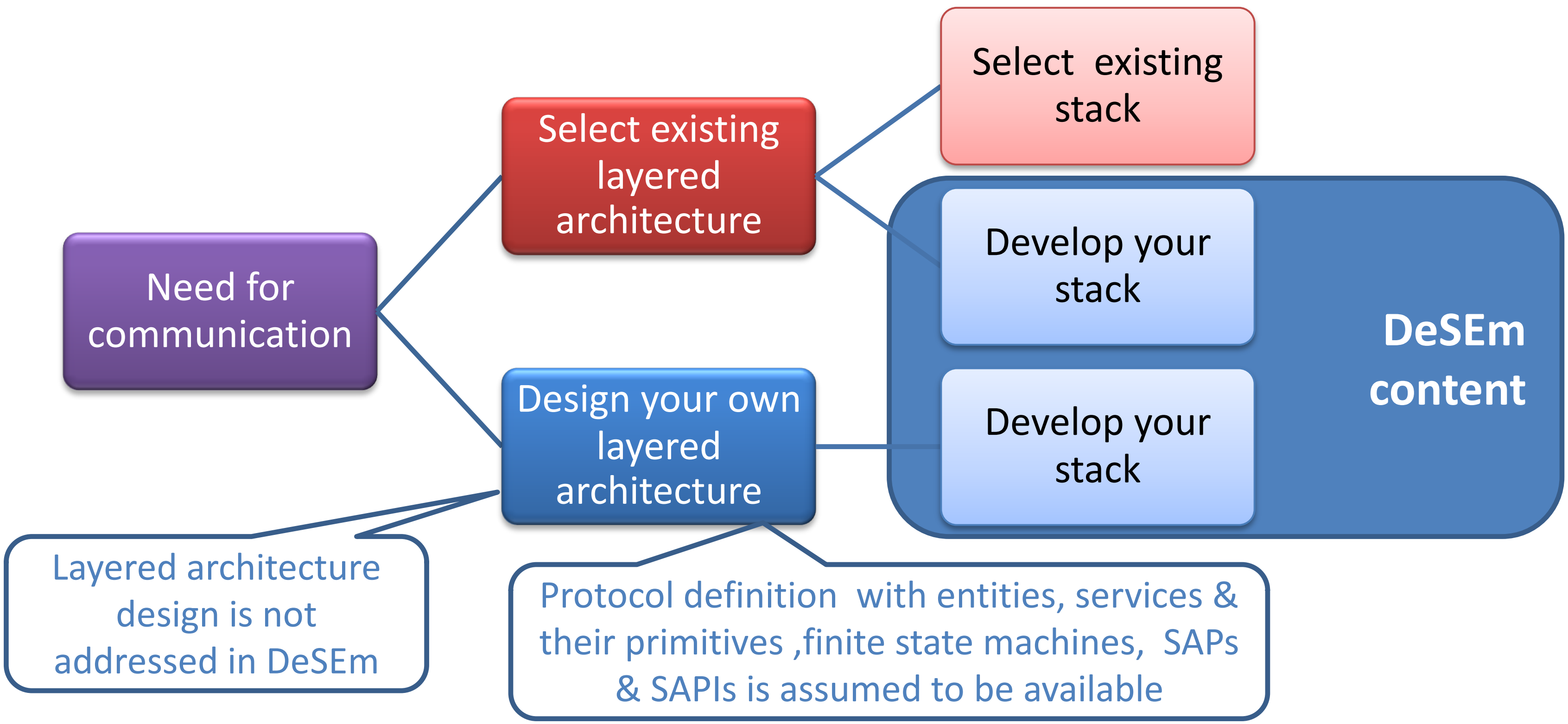
Example of a locally confirmed service



A simplified layered architecture

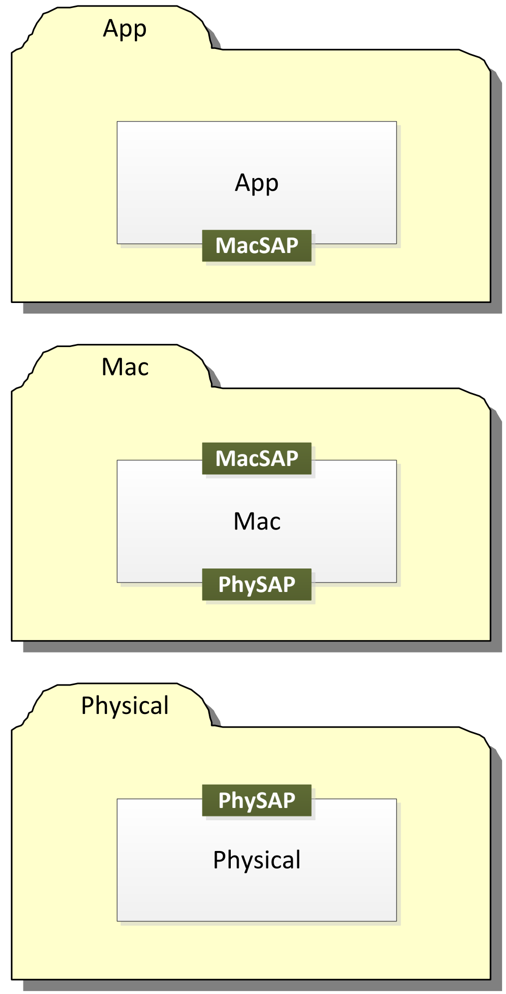


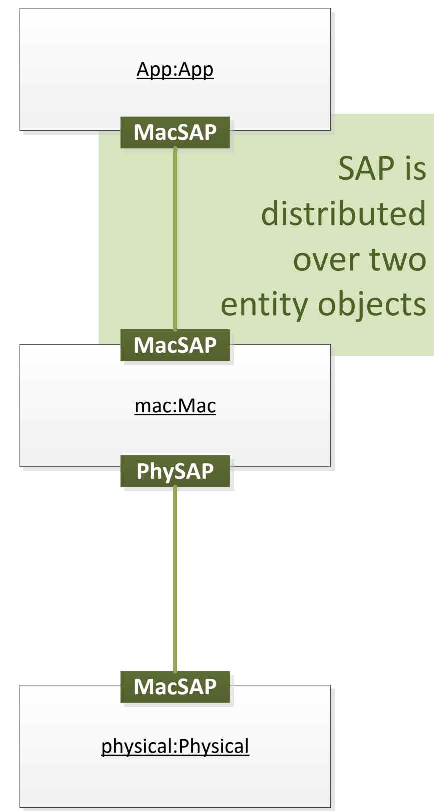
DeSEm approach



Single process strategy

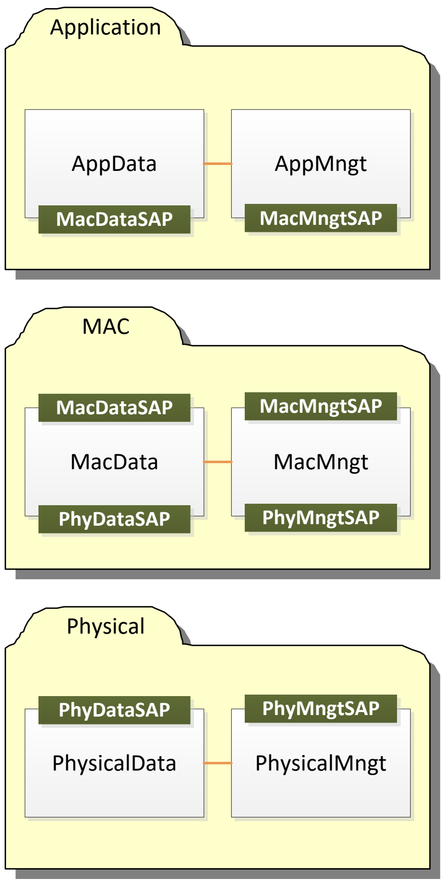
- One process per protocol entity
- A process deals with two SAPs
  - Except higher layer process and lower layer process

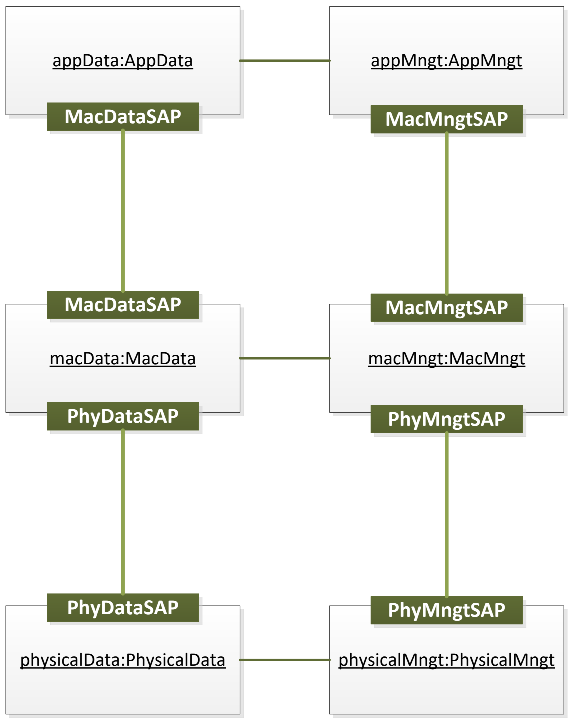




Data and management processes strategy

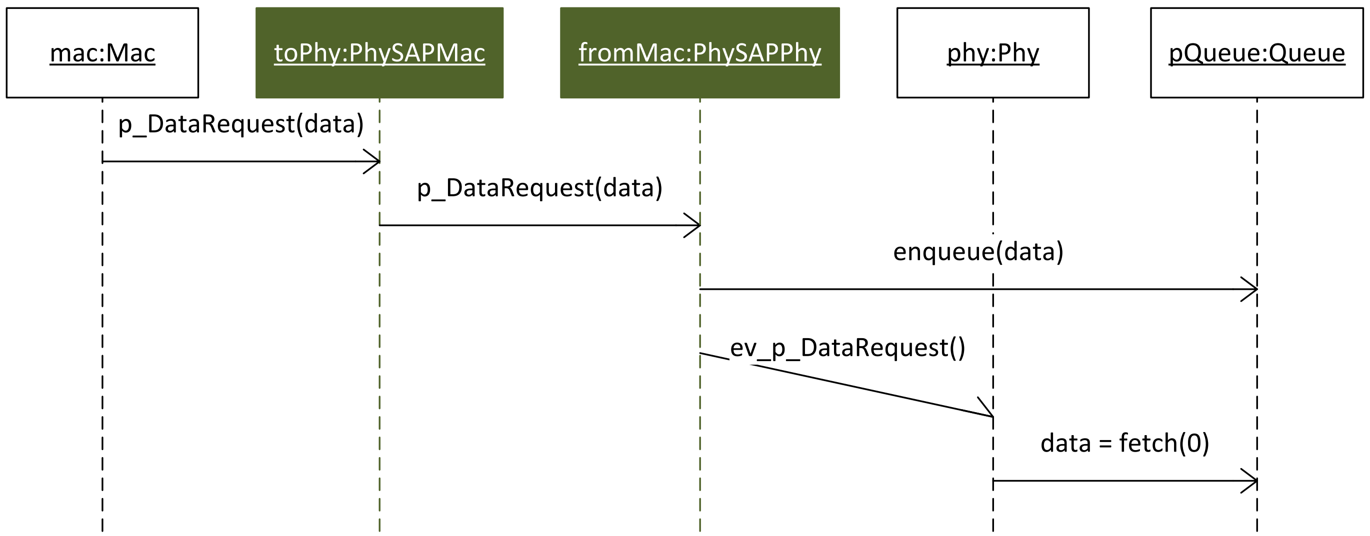
- Goal:
  - Separate data and management flows



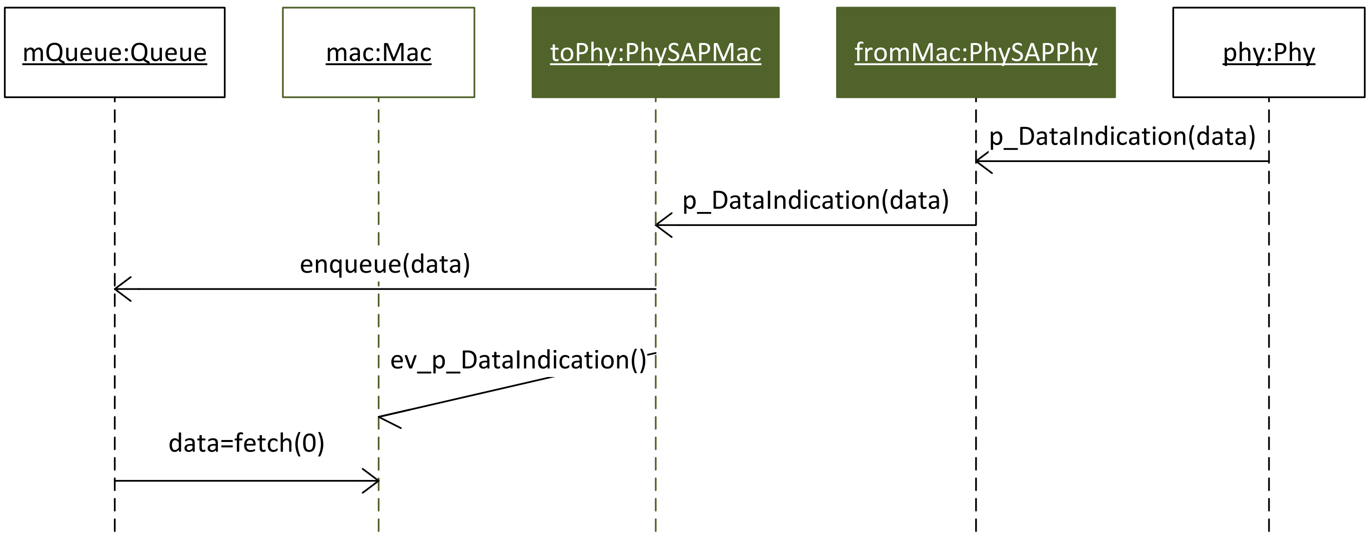


Layer decoupling

- 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)

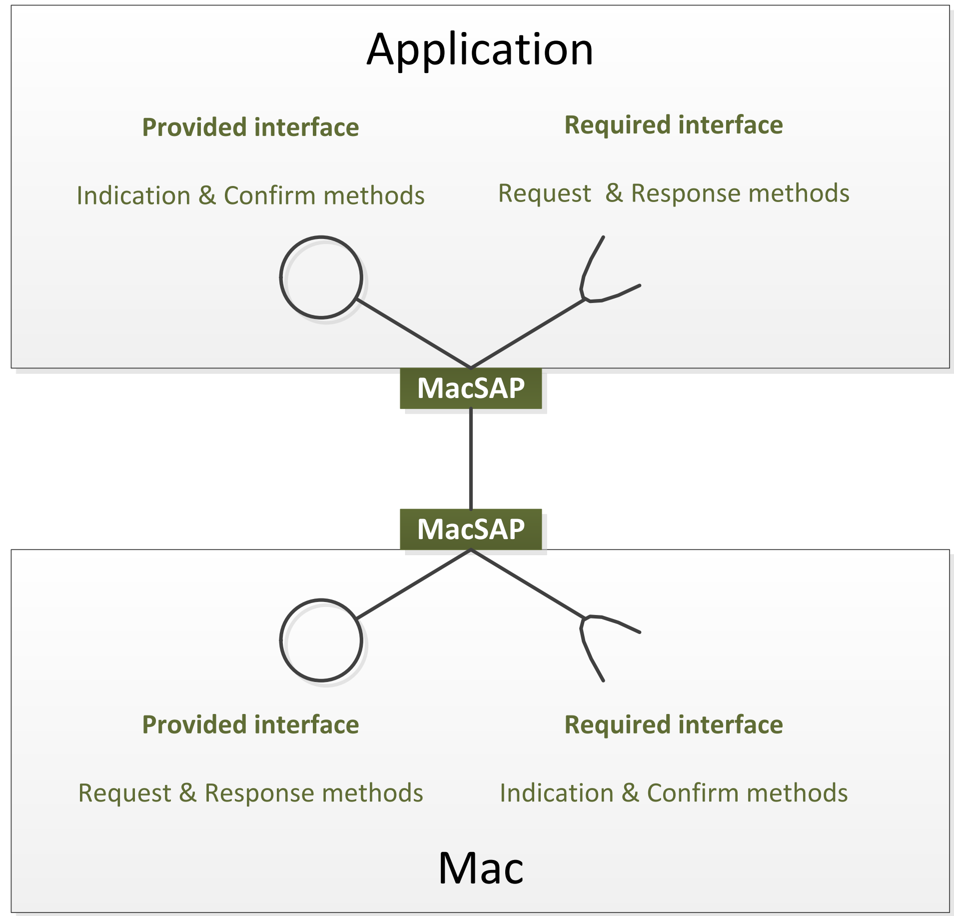


A first UML view of service primitives

OSI model operation

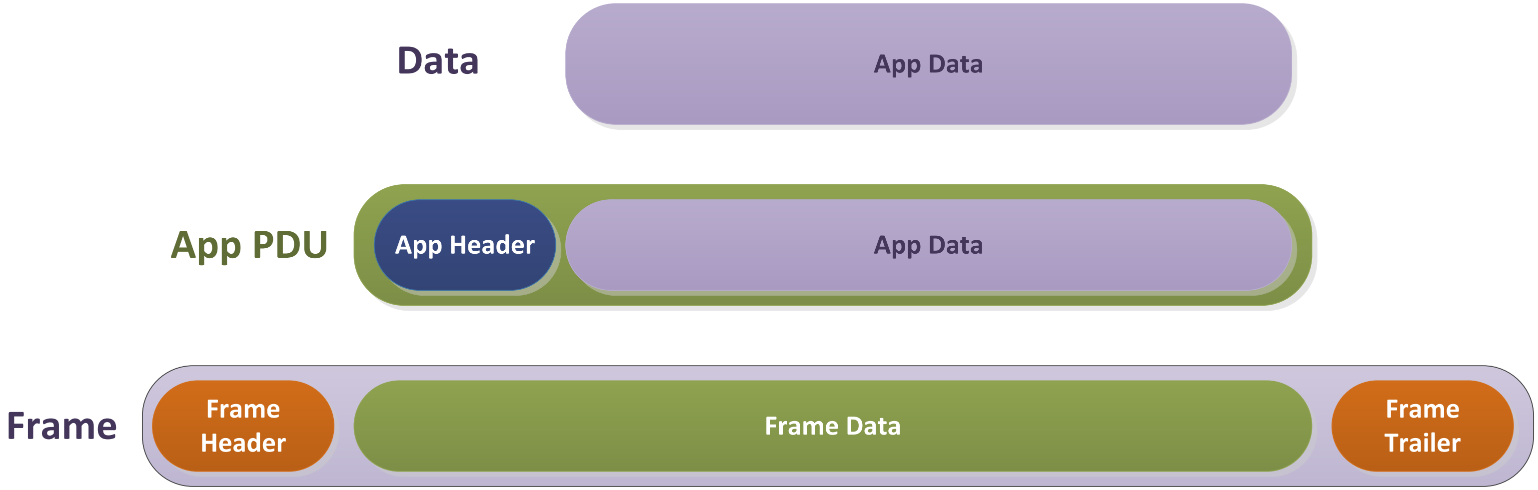
- "Higher" class
  - i.e. class whose instance is an upper layer protocol entity

  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



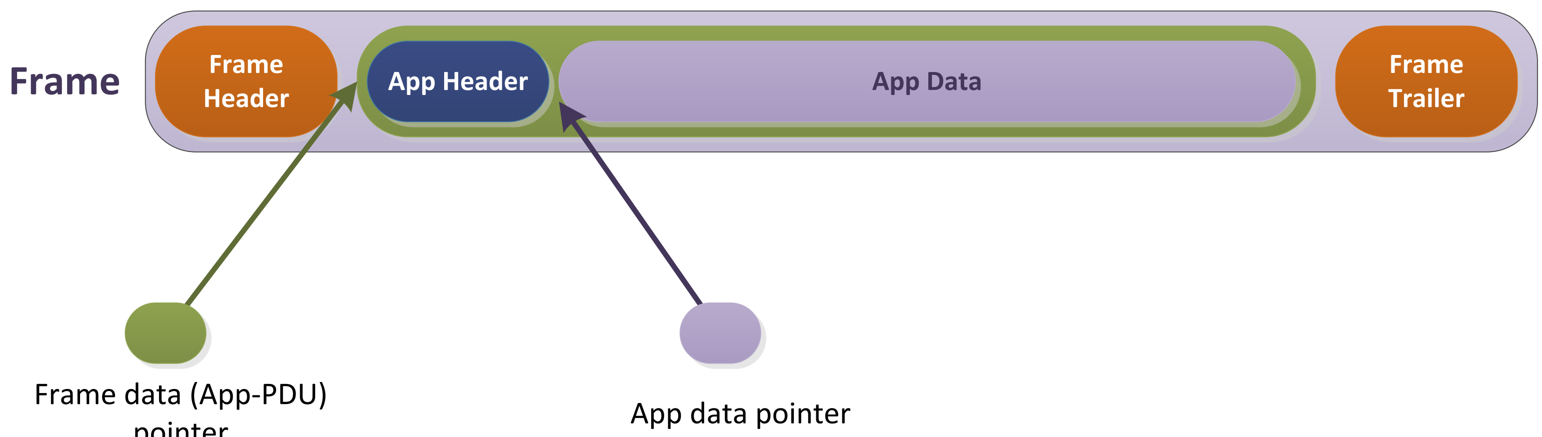
Memory strategy: copy of data

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



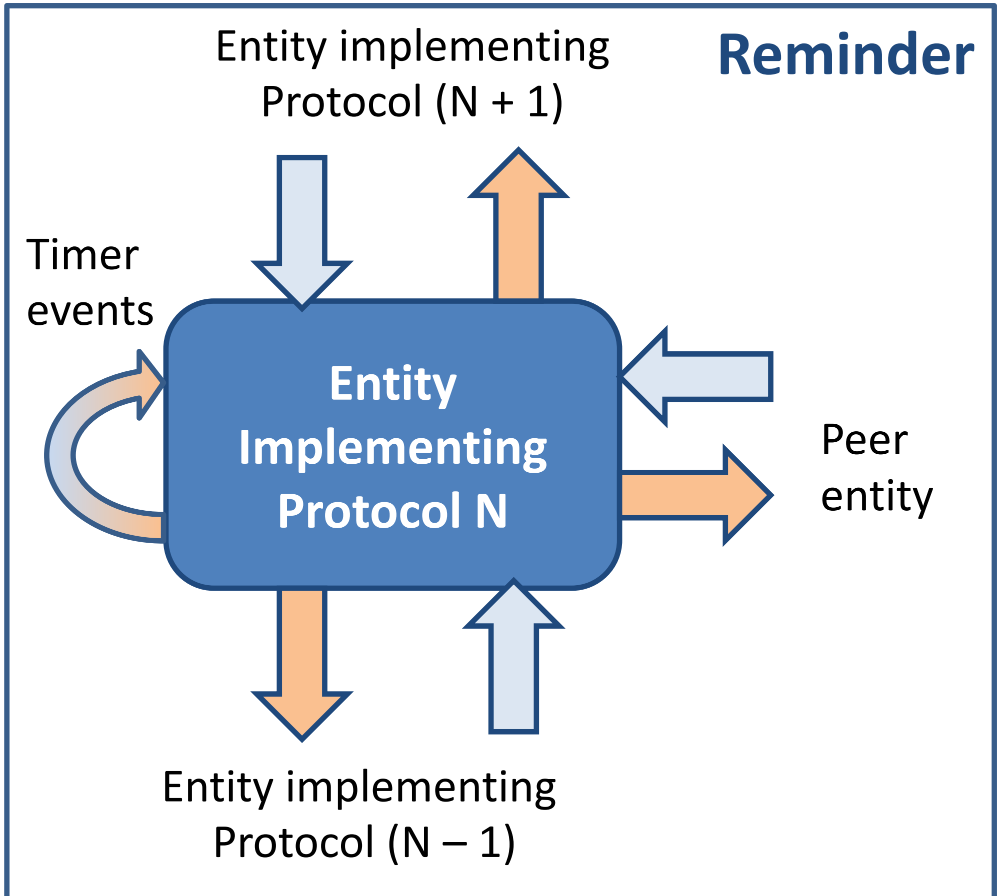
Memory strategy: reference to data

- 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



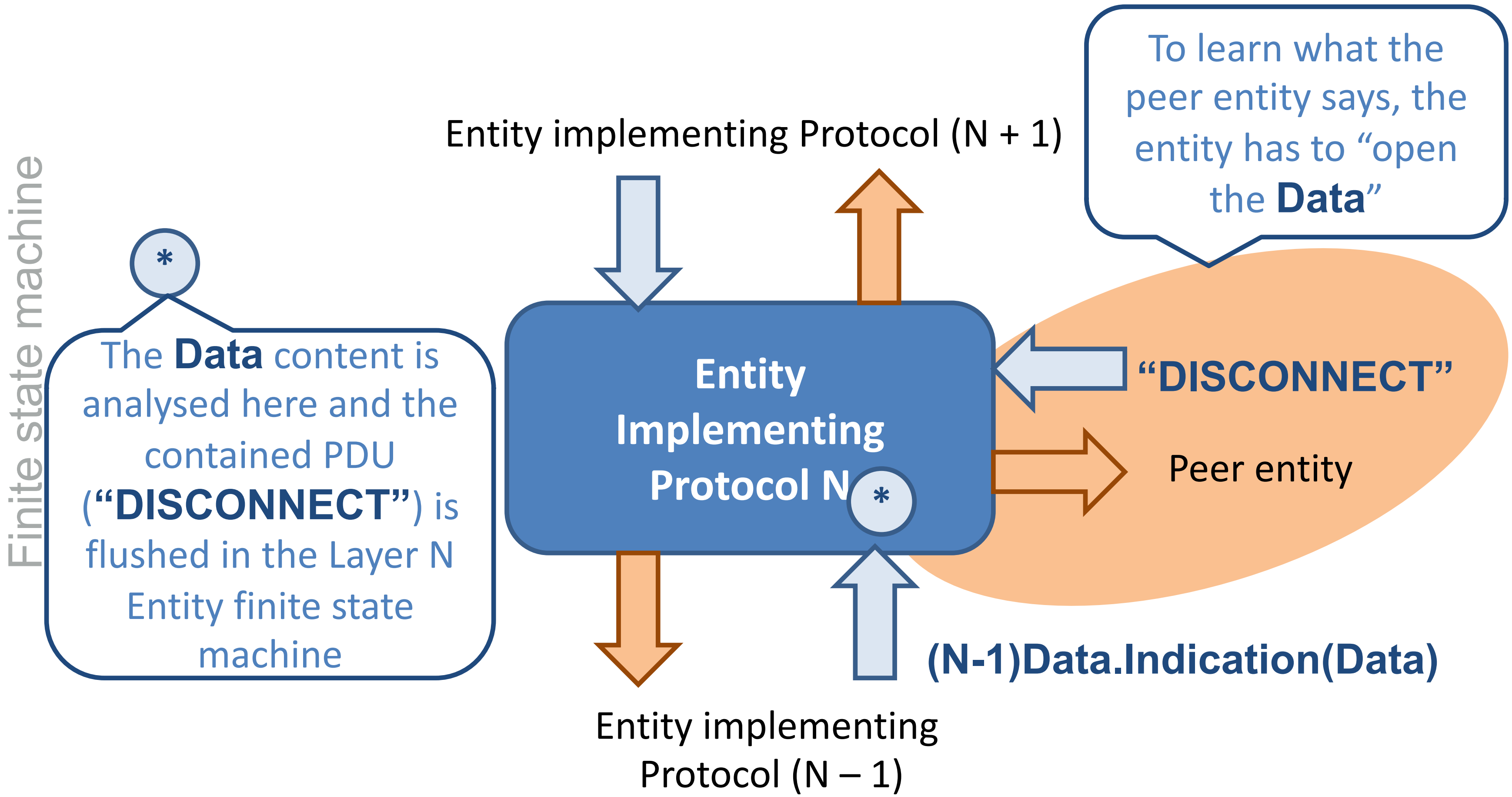
Protocol entity as a finite state machine

Finite state machine



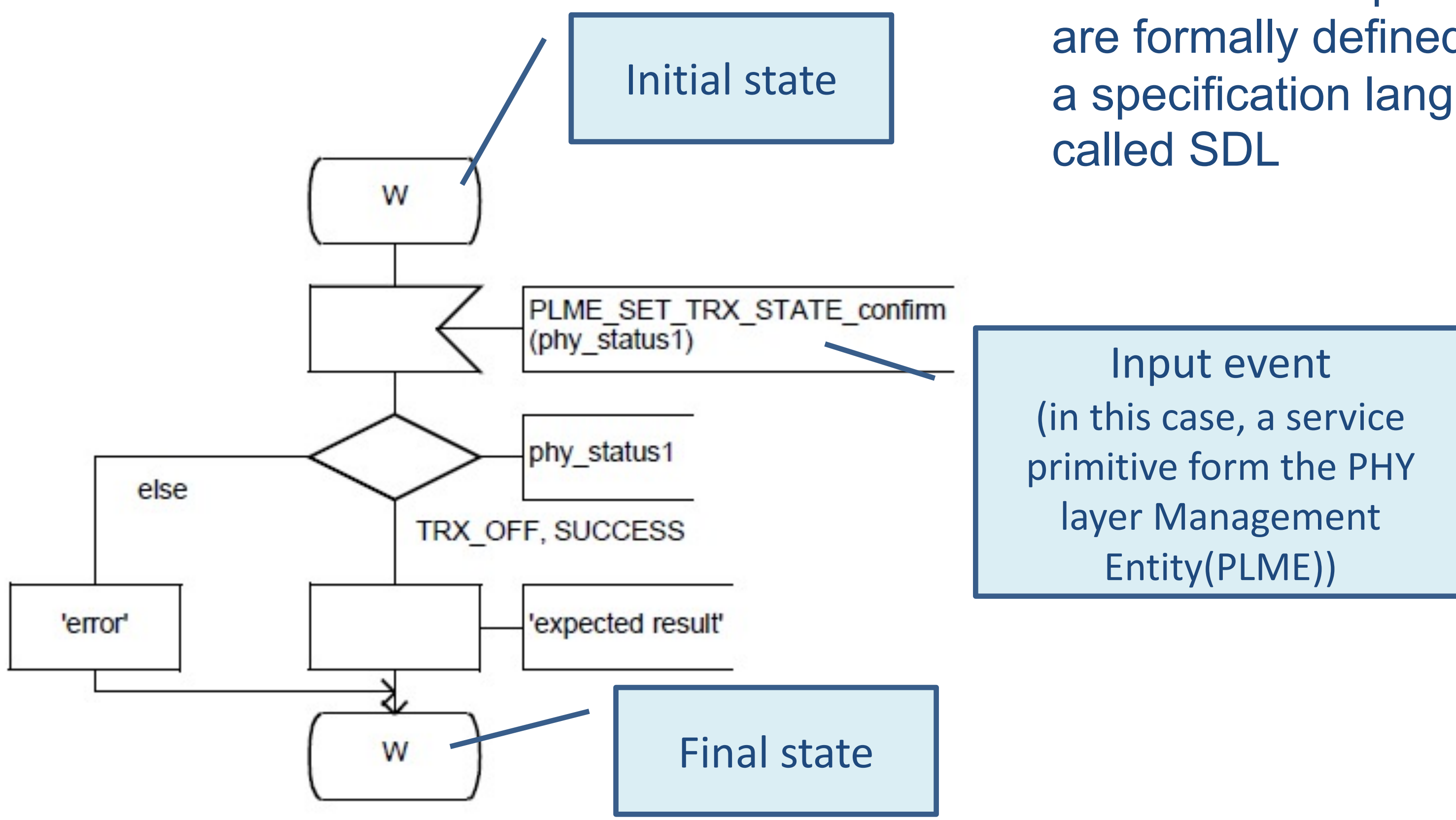
- Protocol definition contains a finite state machine definition
  - Sometimes in a very primitive form
  - Sometimes implicitly
- Timer timeouts are events generated by the entity on itself

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

Master
ENGINEERING (MSE)

DeSEm
Design and Specification of the DeseNET Protocol

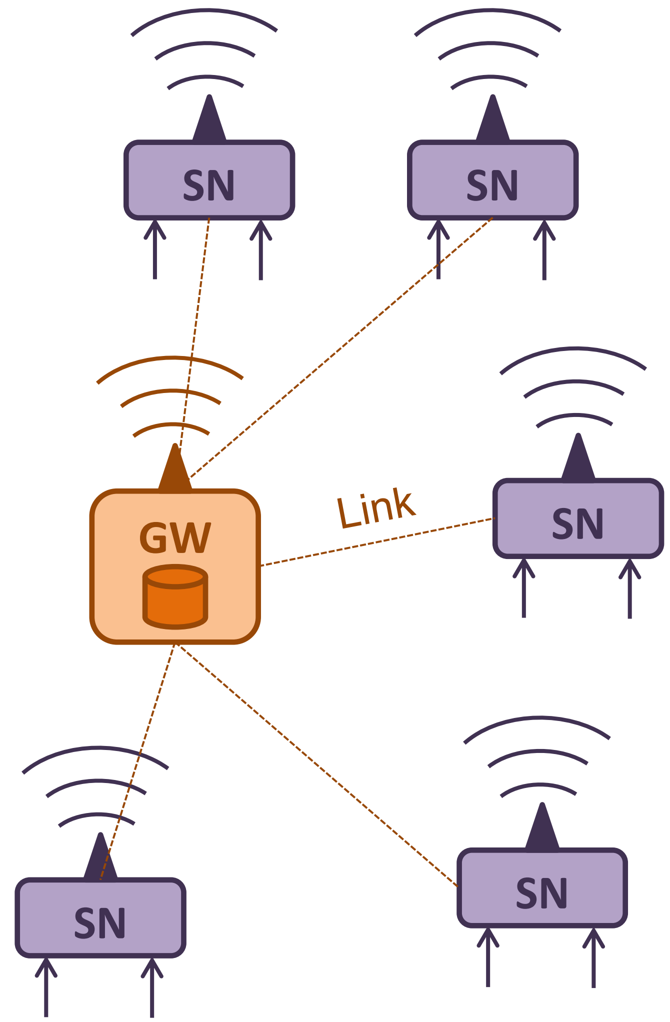
Dominique Gabioud
Michael Clausen
Thomas Sterren
Medard Rieder

HES-SO 2020/21

Table of Content

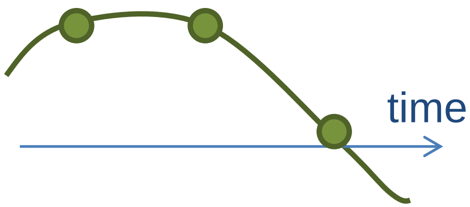
Context

- Wireless sensor network
  - GateWay (GW)
    - Role: store sensor data
      - Local storage capacity or connected to “cloud”
  - Sensor Nodes (SN)
- Star topology
  - Only GW – SN communication
- Battery powered sensor nodes
  - DeseNET protocol should enable low power implementation strategies
- Not a Plug and Play protocol
  - “Keep it simple and stupid!”
    - Manual configuration and/or implicit assumptions about data formats



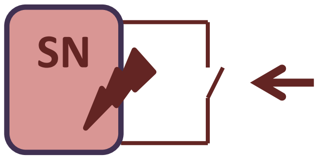
What kind of data do SNs acquire?

- Sampled Values
  - Discrete samples of a continuous signal
    - Digitised by an ADC
- Events
  - Events occur not regularly over time
    - Usually rather infrequently
  - Events can be:
    - Change of state for a binary input,
    - Continuous values above/below threshold,
    - …



Sampled value service

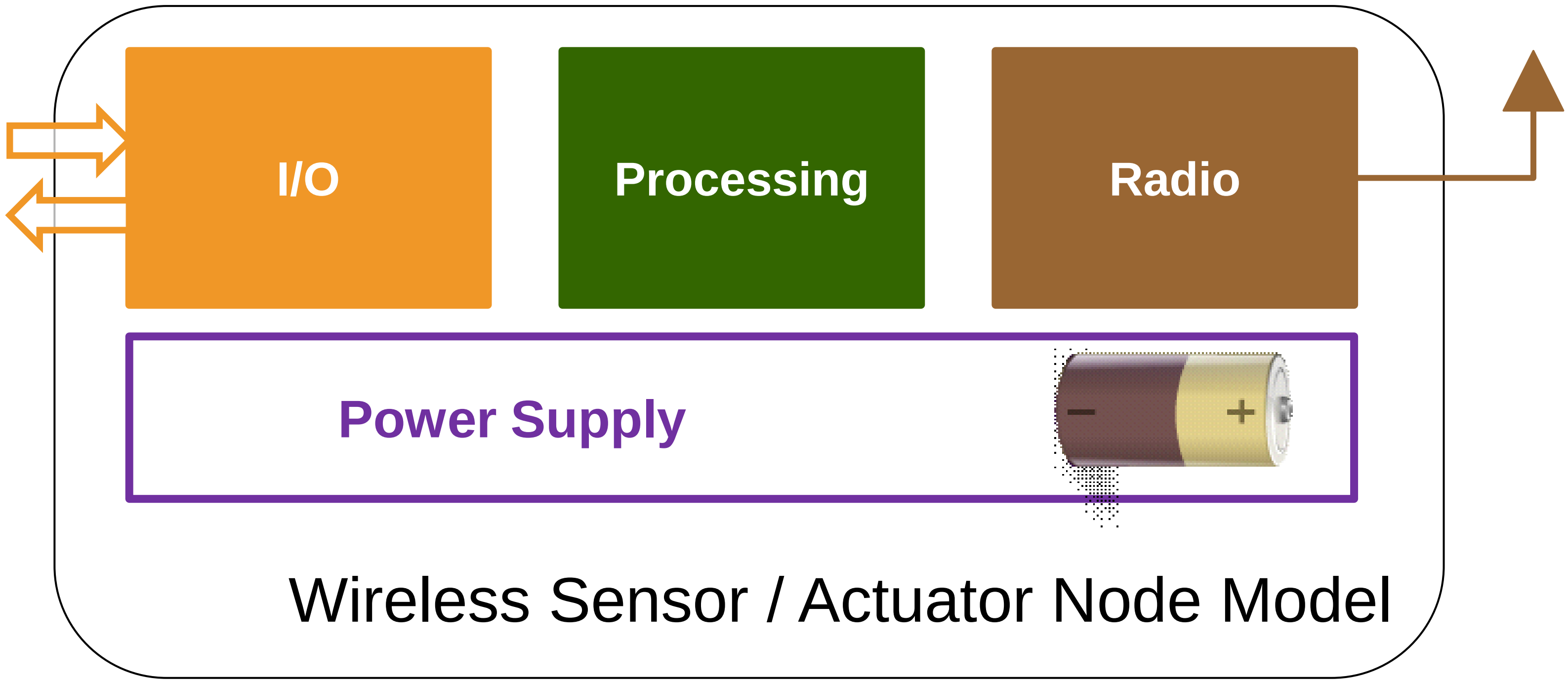
It is required that sampling is performed synchronously on all nodes, to provide a good "picture" of a process .
Frequency and phase synchronicity required!



Event service

Protocol stack should implement **a sampled value service** and an **event service**

Wireless & energy: Model



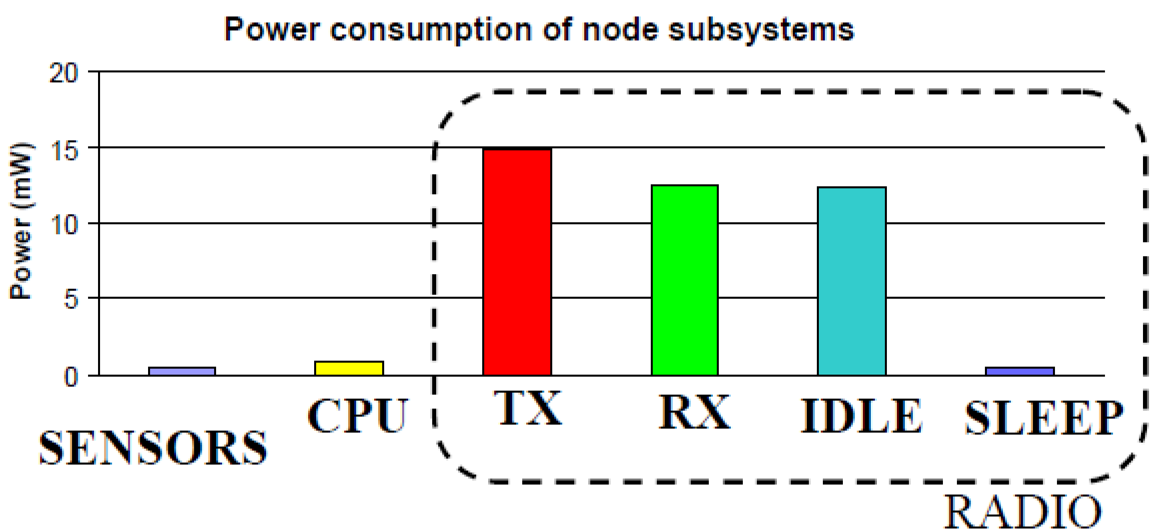
The wireless radio is often the most energy consuming unit of a node

Wireless & energy: Transceiver state

- Radio transceiver states:
  - **Transmit (TX)** : The radio is sending frames over the air to peer nodes
  - **Receive (RX)** : The radio receives frames from peer nodes
  - **Idle**: The radio receiver is turned on, but no frame is incoming
  - **Sleep**: The radio is turned off
- Radio transceiver control:
  - ON / OFF command
    - OFF: **Sleep** state
    - ON: Default state is **Idle**
      In case of incoming frame, the transceiver goes automatically in **Receive** state

      Upon frame transmission request, the transceiver goes in **Transmit** state and comes back to **Idle** state after the end of transmission

Wireless & energy: Distribution of consumption



From Tsiatis et al. 2002

The **Transmit**, **Receive** and **Idle** states consume almost the same energy

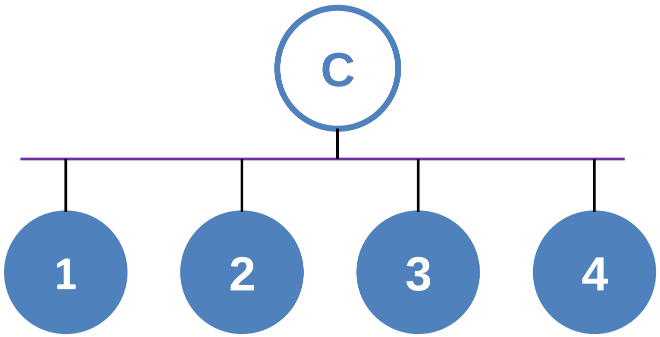
In the **Sleep** state, the consumption is 3 orders of magnitude lower

MAC & Energy

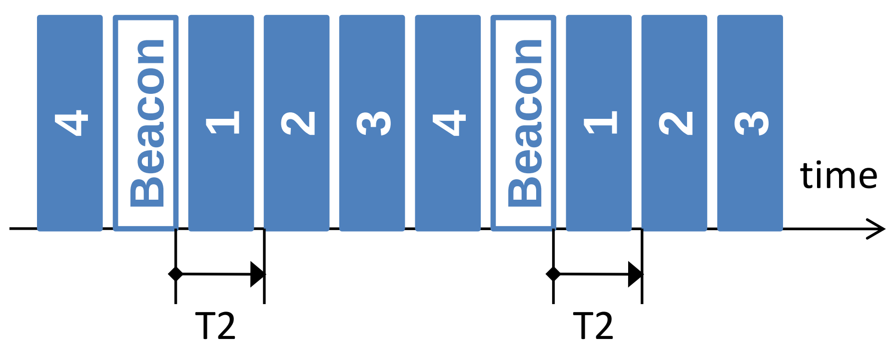
- MAC (Medium Access Control) layer role
  - Organise access to a shared communication channel by several nodes
  - Wired bus or wireless channel
- MAC & energy efficiency
  - An energy efficient MAC must control the radio transceiver states for minimum energy consumption
    - While still fulfilling the expected requirements
- Major sources of energy wastes:
  - Collision: Frames transmitted simultaneously by several nodes are corrupted.
    - Retransmission increases energy consumption
  - Overhearing: A node picks up packet destined to other nodes
  - Overhead: Sending and receiving control frames or control fields in data frames
  - Idle listening: Radio receiver turned on but no incoming frames

Principle of TDMA MAC

TDMA:
Time Division Multiple Access



- A node with coordinator role broadcasts periodic beacons
- Each node has its reserved transmission time slot
  - Fixed beacon to time slot delay
- Each node can read time slots
  - Time slot determined by beacon delay time
- Typical use:
  - Continuous bit rate “streaming applications”
  - **DECT** (Digital Enhanced Cordless Telecommunications)



TDMA MAC & energy

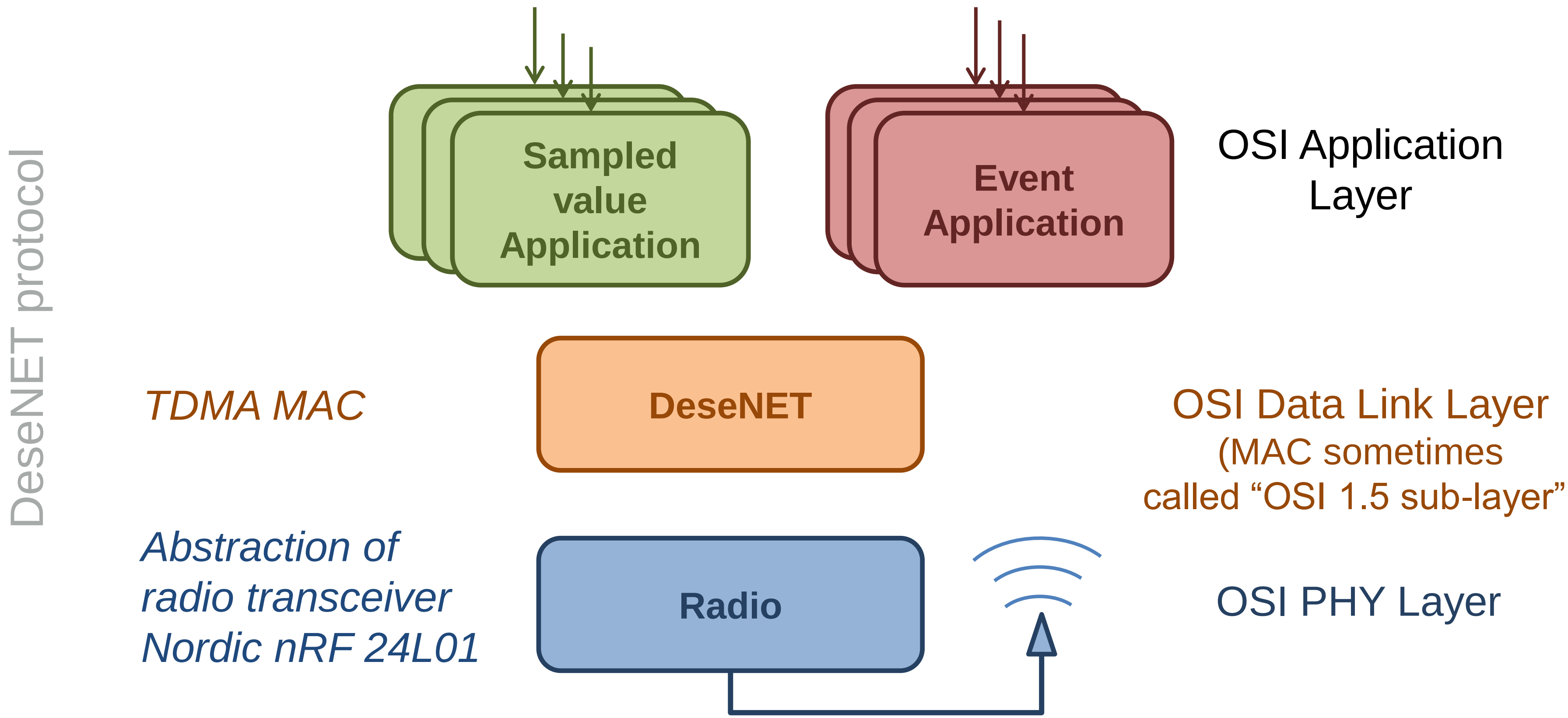
- Source of energy wastes
  - Collision: ++
    - No collision
  - Overhearing: ++
    - Non-existent, at least for nodes without actuator
  - Overhead: ++
    - Control frames kept to minimum
  - Idle listening: ++
    - Non-existent, at least for nodes without actuators

Performance assessment (best grade: "++")

TDMA: An overall excellent behaviour regarding energy

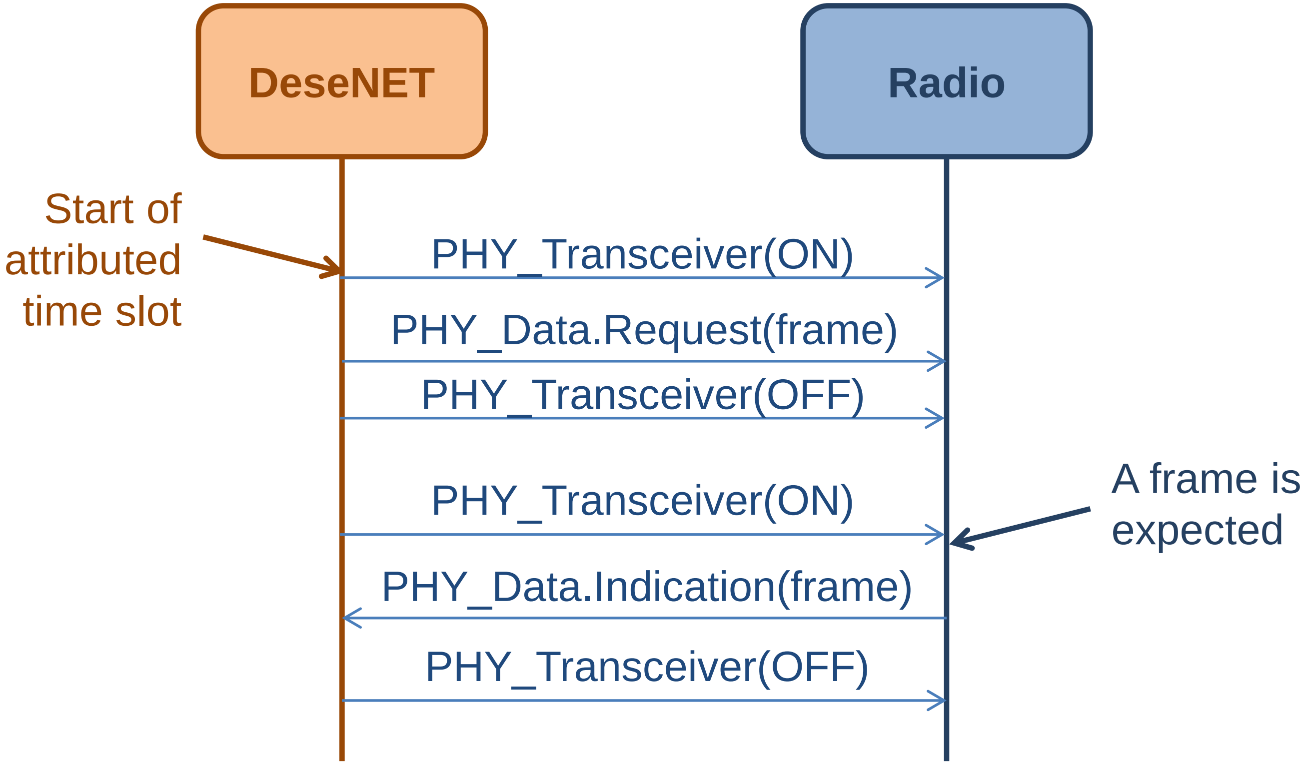
Any limitation for TDMA MAC?

DeseNET layer architecture



Radio & DeseNET layers

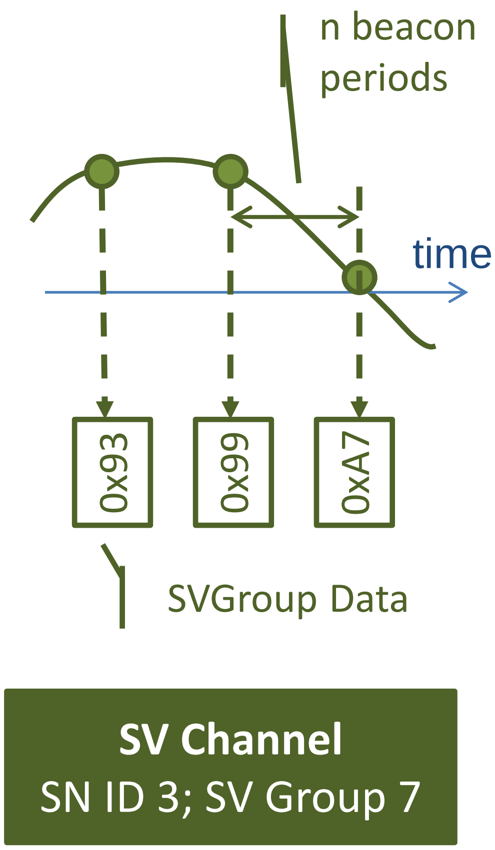
DeseNET protocol



The Sampled Value service

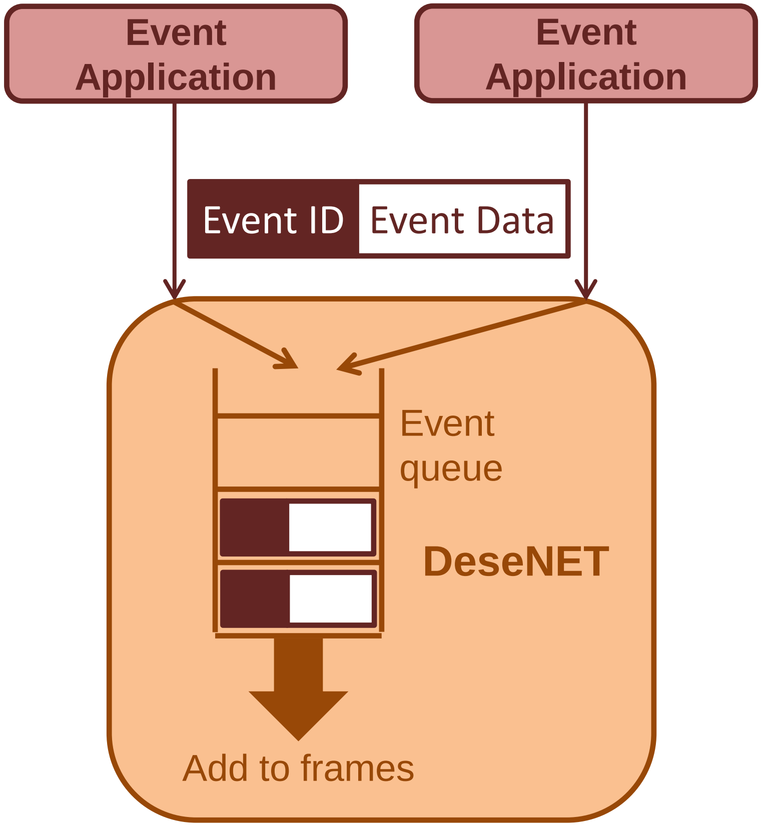
DeseNET protocol

- Chunk of data (SV Group Data) collected periodically by local sensors
  - A SV Group Data block is typically obtained through sampling (ADC) of a continuous signal
  - A SV Group Data block may contain several multiplexed sampled signals or any other periodically generated data block
    - Encoding of the SV Group Data is outside the scope of DeseNET
- Sequence of SV Group Data from the same source build a SV Channel
  - A Sampled Value Channel is identified by:
    - the SN ID of the originating SN, and
    - a so-called SV Group parameter, which identifies the local SV Group Data source
  - A SV Group identifies similar sources over all SNs
    - Example: SV Group7 -> Accelerometer sensor samples
- SV Group Data are transmitted on request of the Gateway:
  - Beacons carry Sampled Value transmission request for a given SV Group
  - A request for a given SV Group is transmitted in every n beacons (n =1, 2, 3…)
    - n may be different for each SV group



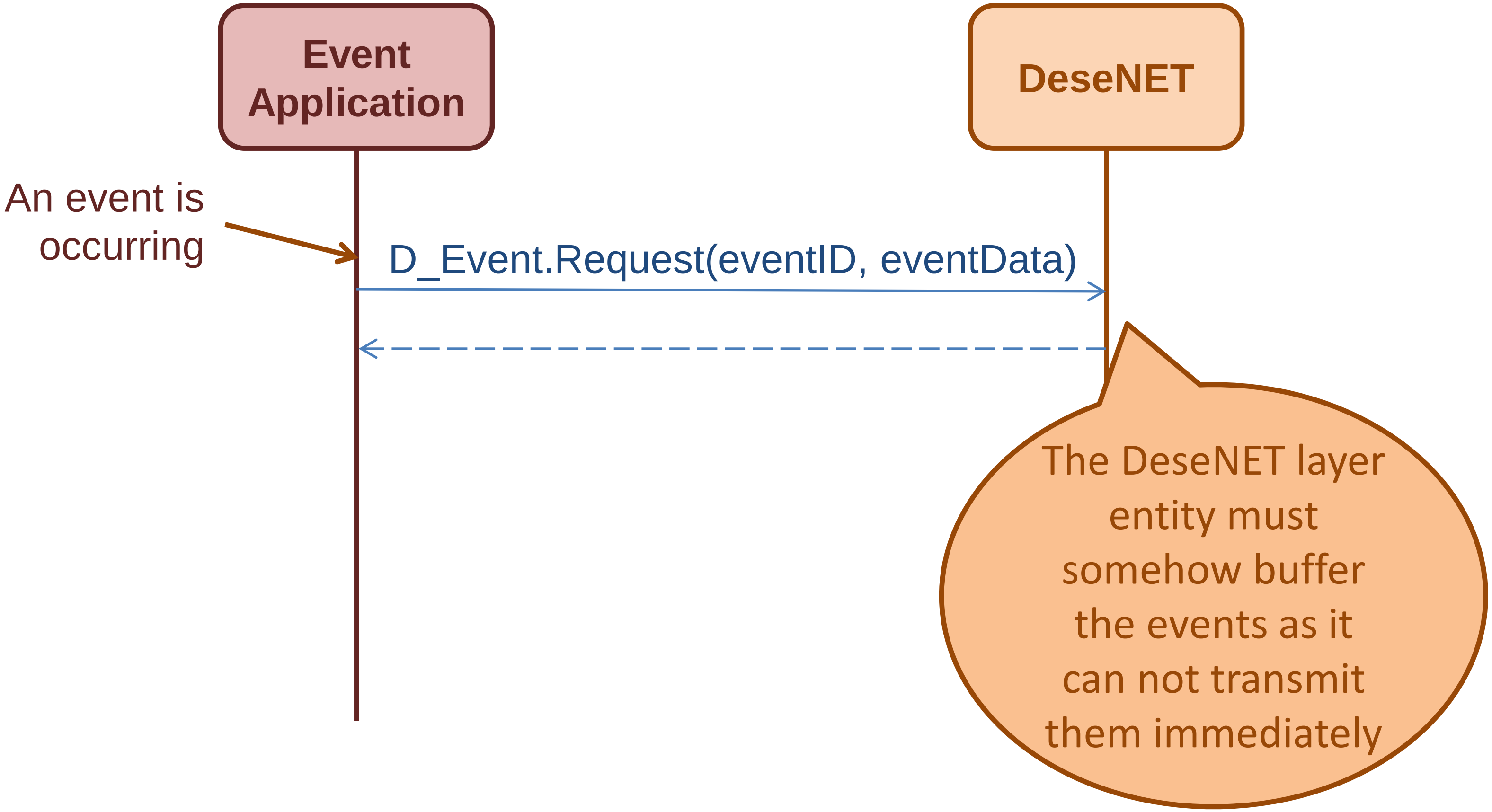
The Event service

- An event is characterised by:
  - an Event ID, and
  - associated Event Data
- There is no restriction on the pace of occurrence of event. Hence, DeseNET:
  - queues events generated by local applications
  - empties the event queue as fast as possible. Extracted events are sent using the DeseNET protocol



DeseNET & Event Application layers

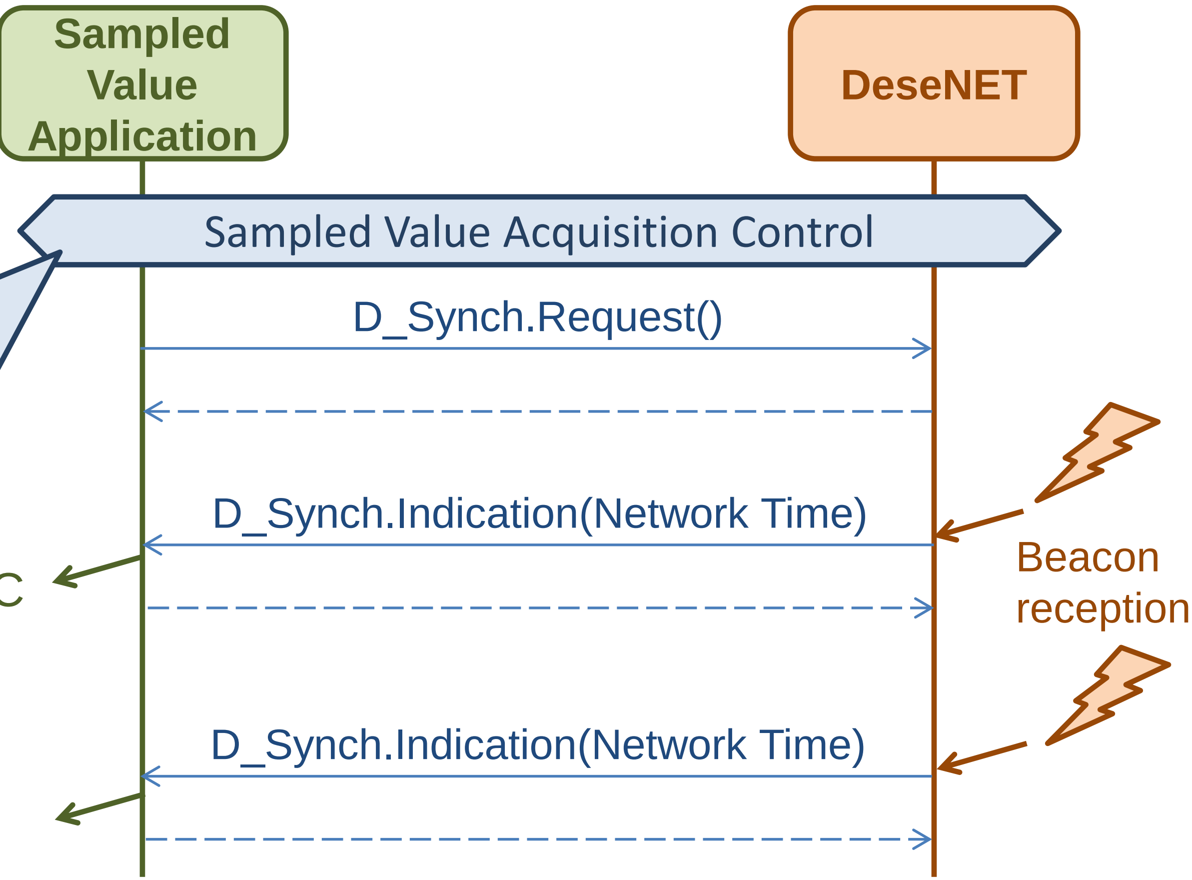
DeseNET protocol



DeseNET & SV Application layers: Synchronising ADC

DeseNET protocol

Roles:
- To perform ADC exactly once per TDMA cycle
- To (phase and frequency synchronise ADC over several sensor nodes



Read ADC

The DeseNET entity needs to keep track of the SV Applications to synch…

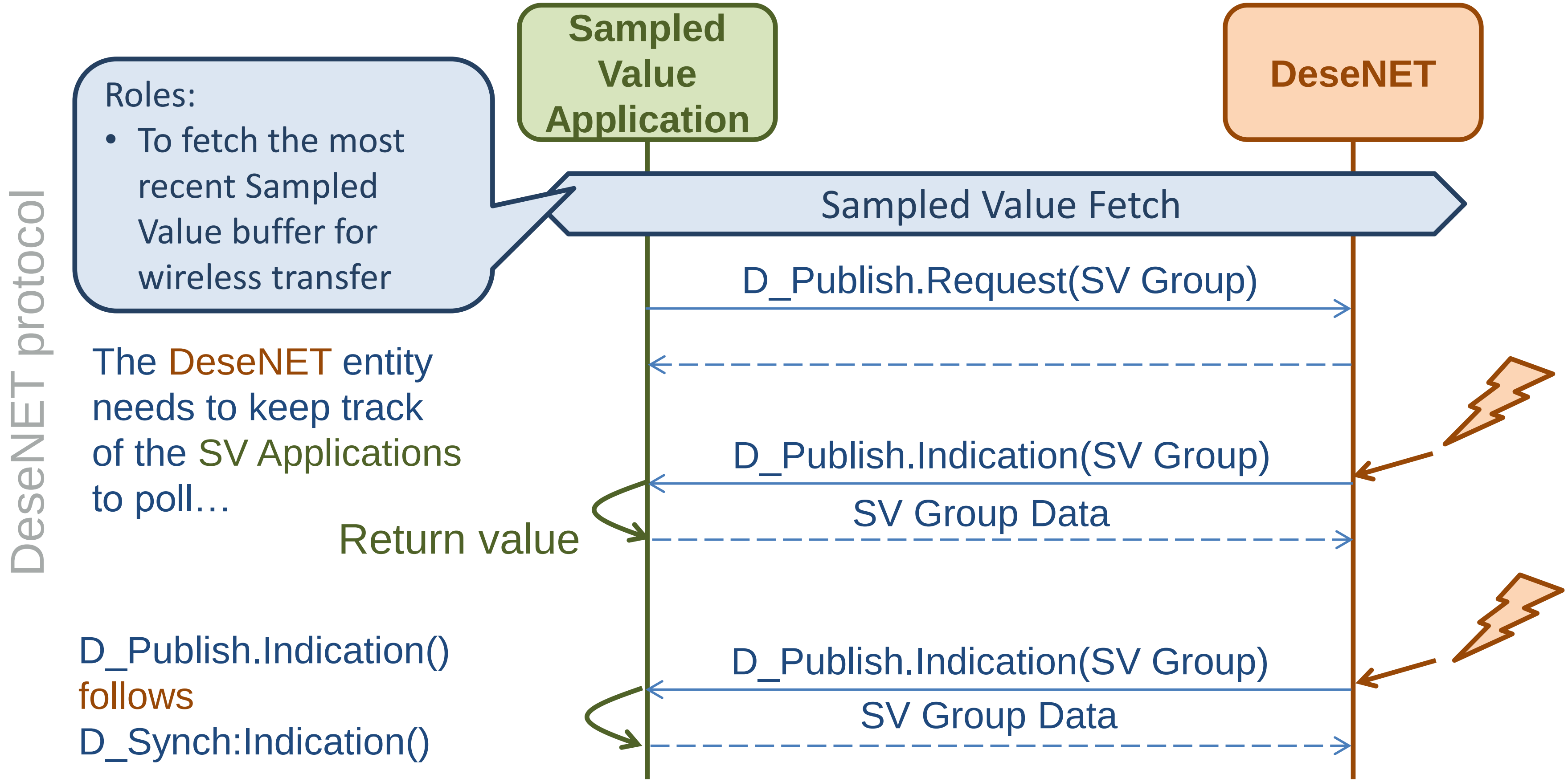
DeseNET & SV Application layers: Synchronising ADC

- DeseNET enables a Gateway to trigger ADC reading on a per cycle basis
  - The Gateway may ask to read & transmit a selected subset of the SV Groups at each cycle

- Synchronisation service is optional
    - ADC read could be performed cyclically for example

But it's a good practice to read samples from ADC at the TDMA transmission rate

DeseNET & SV Application layers: Getting the sampled values

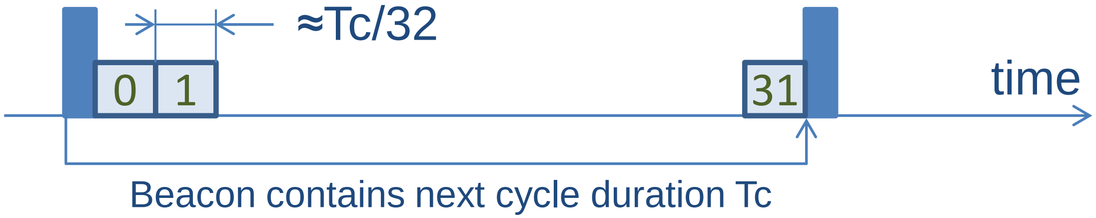


The DeseNET entity needs to keep track of the SV Applications to poll…

D_Publish.Indication() follows D_Synch:Indication()

TDMA implementation

- DeseNET features:
  - 32 time slots numbered 0… 31
  - A variable period Tc, indicated by the Gateway in each Beacon
    - Hence the time position in cycle and the slot duration can be calculated
- The slot number (“the address”) is statically configured in each sensor node



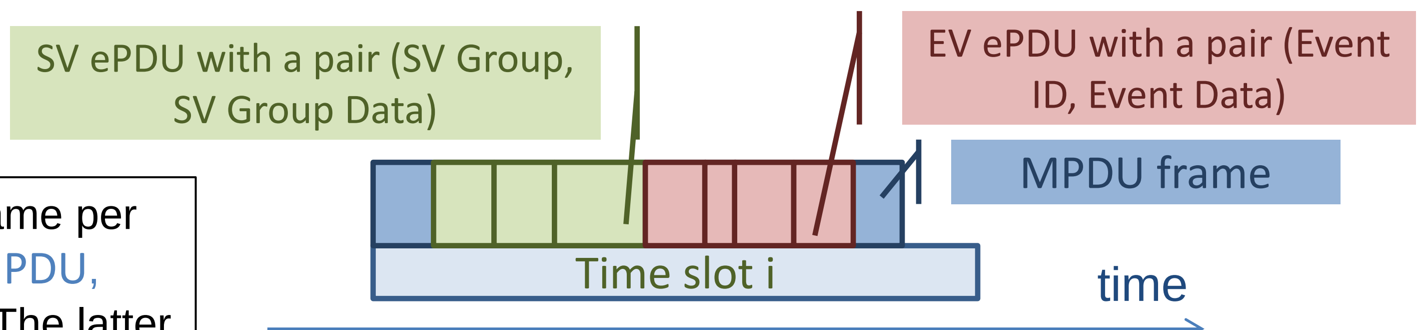
Overview of DeseNET frame format

DeseNET frame format

- Within a cycle period Tc, a sensor node can have gathered
  - Sampled values: A constant number of pairs (SV Group, SV Group Data)
    - Unless new subscriptions occurred
  - Events: A variable number of pairs (Event ID, Event Data)
- All these data have to be send in the attributed time slot
  - DeseNET chose the option to send a unique frame per time slot
    - There is one and only one receiver (the Gateway)
    - Less overhead with one frame

DeseNET frame is called MPDU (Multiple PDU)

There is one frame per time slot (the MPDU, Multiple PDU). The latter contains several ePDUs (embedded PDUs)



A simple, conservative, suboptimal algorithm is to be implemented to keep the MPDU frame duration shorter than the time slot

ePDU format

- An SV ePDU contains mainly:
  - The SV Group and the SV Group Data
- An EV ePDU contains mainly
  - The Event ID and the Event Data
- SV Groups and Event IDs must be managed at DeseNET level
  - But their management is outside the scope of the DeseNET protocol
  - For the DeseNET protocol:
    - SV Groups and Event IDs are just numbers
    - SV Group Data and Event Data are just byte arrays

Master
ENGINEERING (MSE)

Hes·so VALAIS WALLIS

DeSEm
Laboratory - Development of the DeseNet Protocol Stack

Dominique Gabioud

Michael Clausen

Thomas Sterren

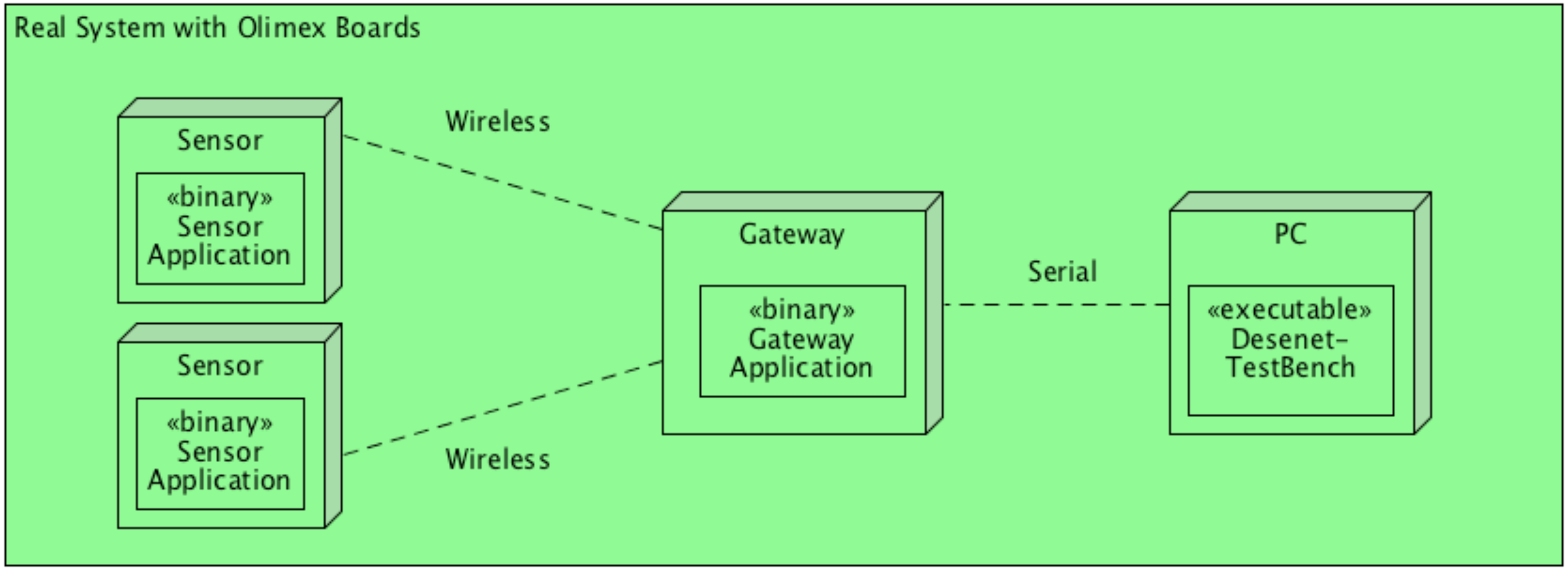
Medard Rieder

HES-SO 2021/22

Table of Content

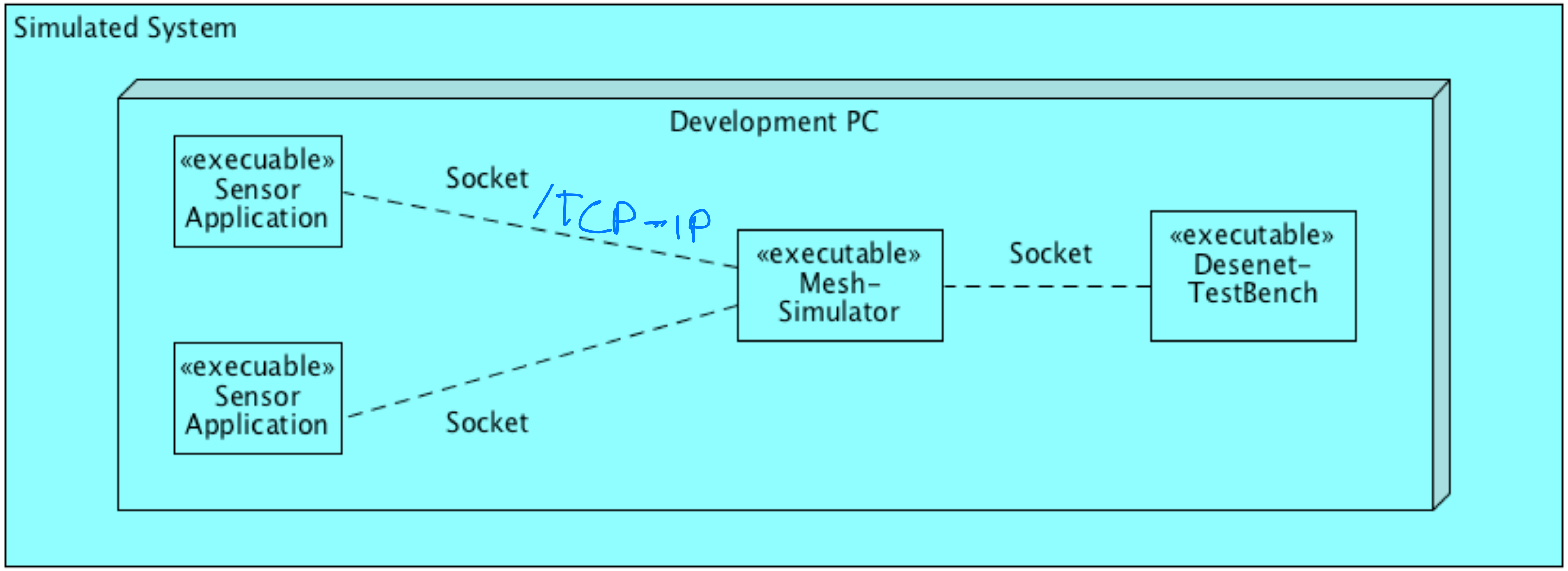
System with real Nodes

System Overview

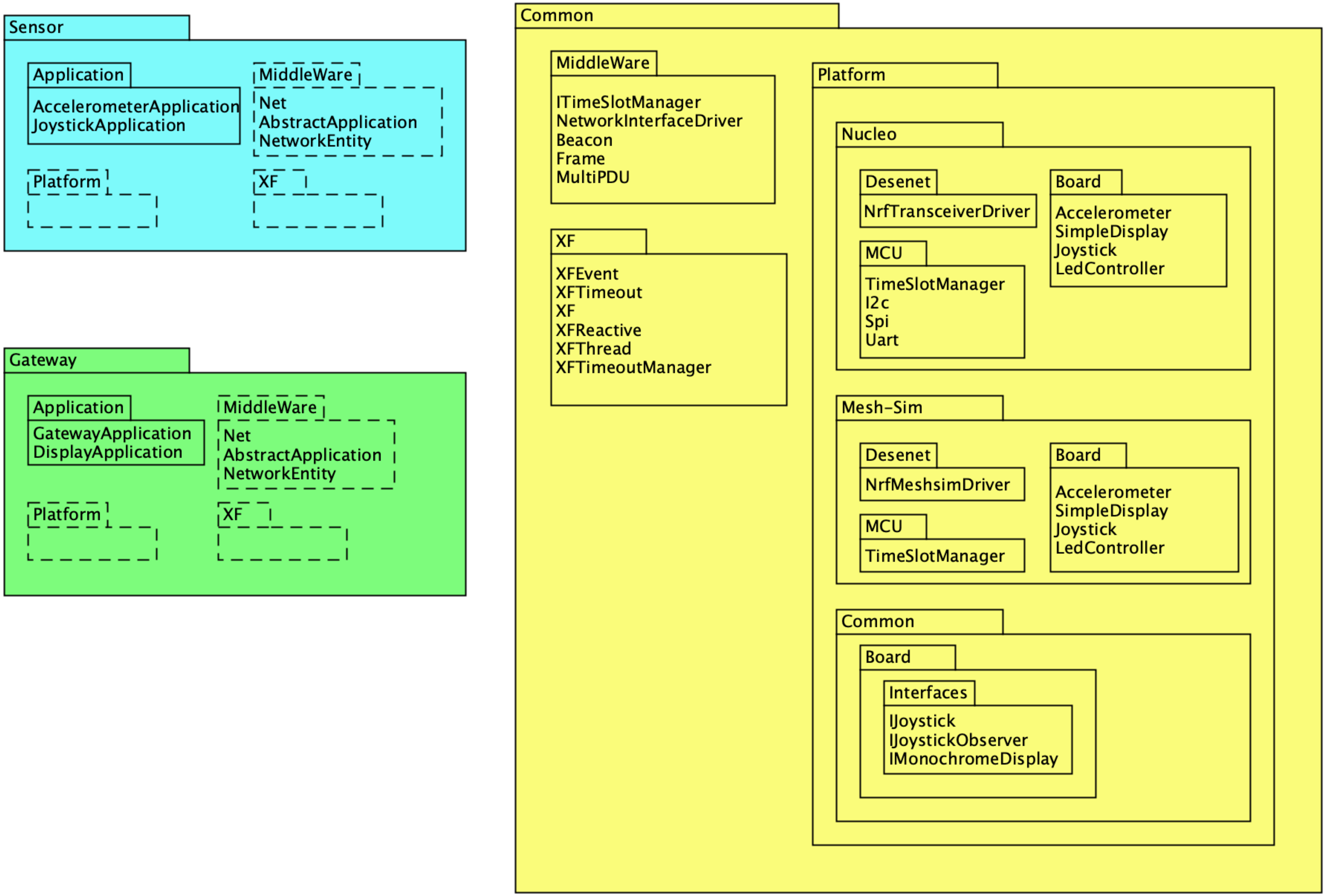


System with virtual Nodes

System Overview

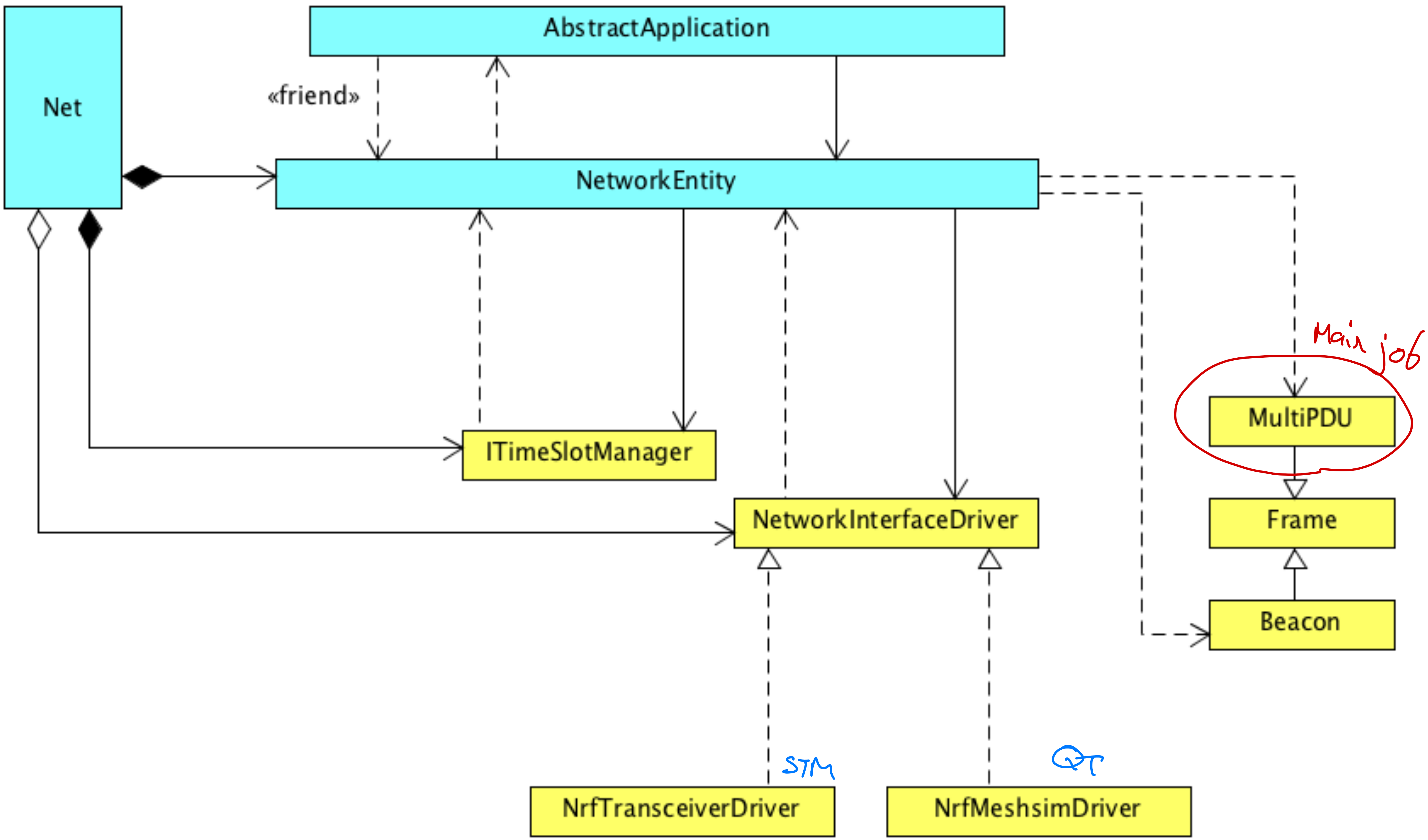


Architecture

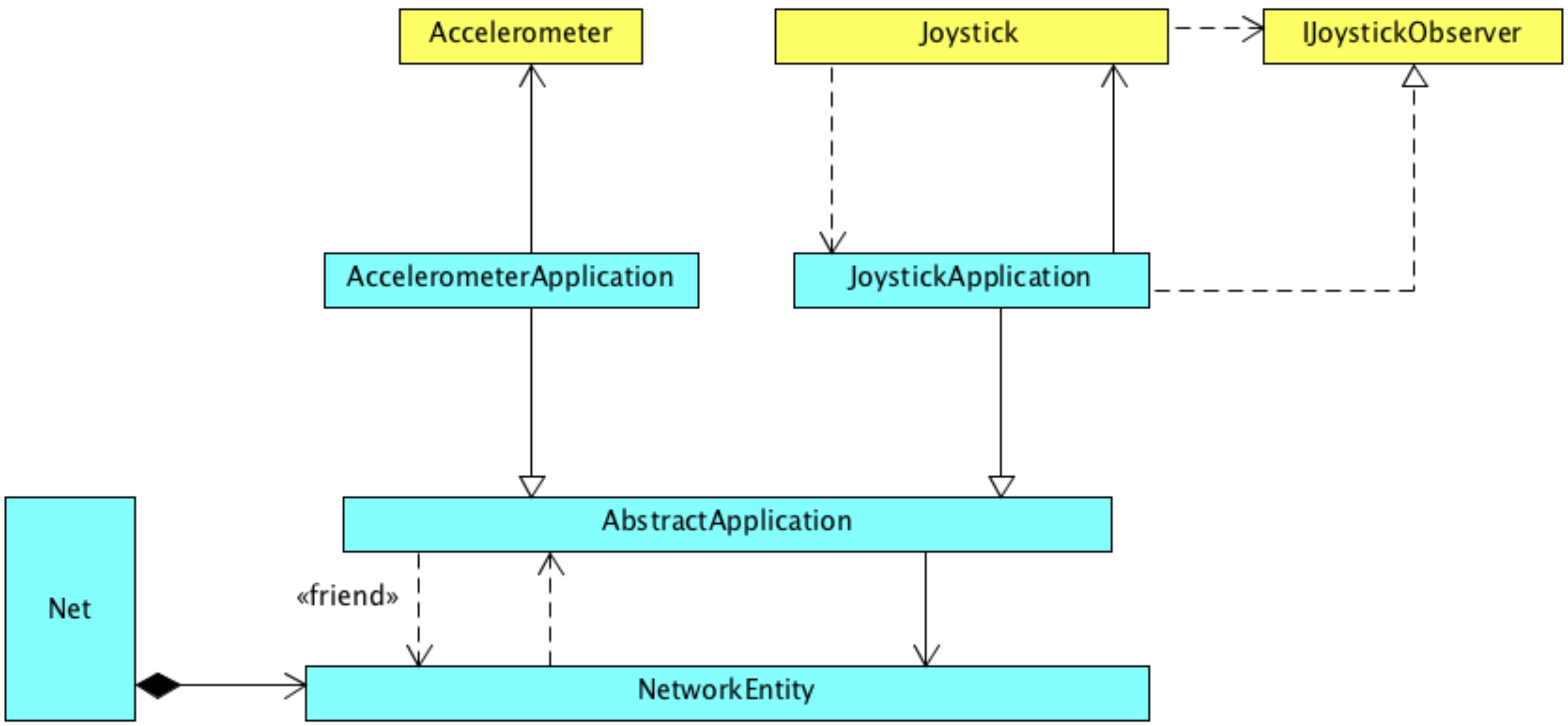


Protocol Elements - Classe

Protocol Design



Application Elements

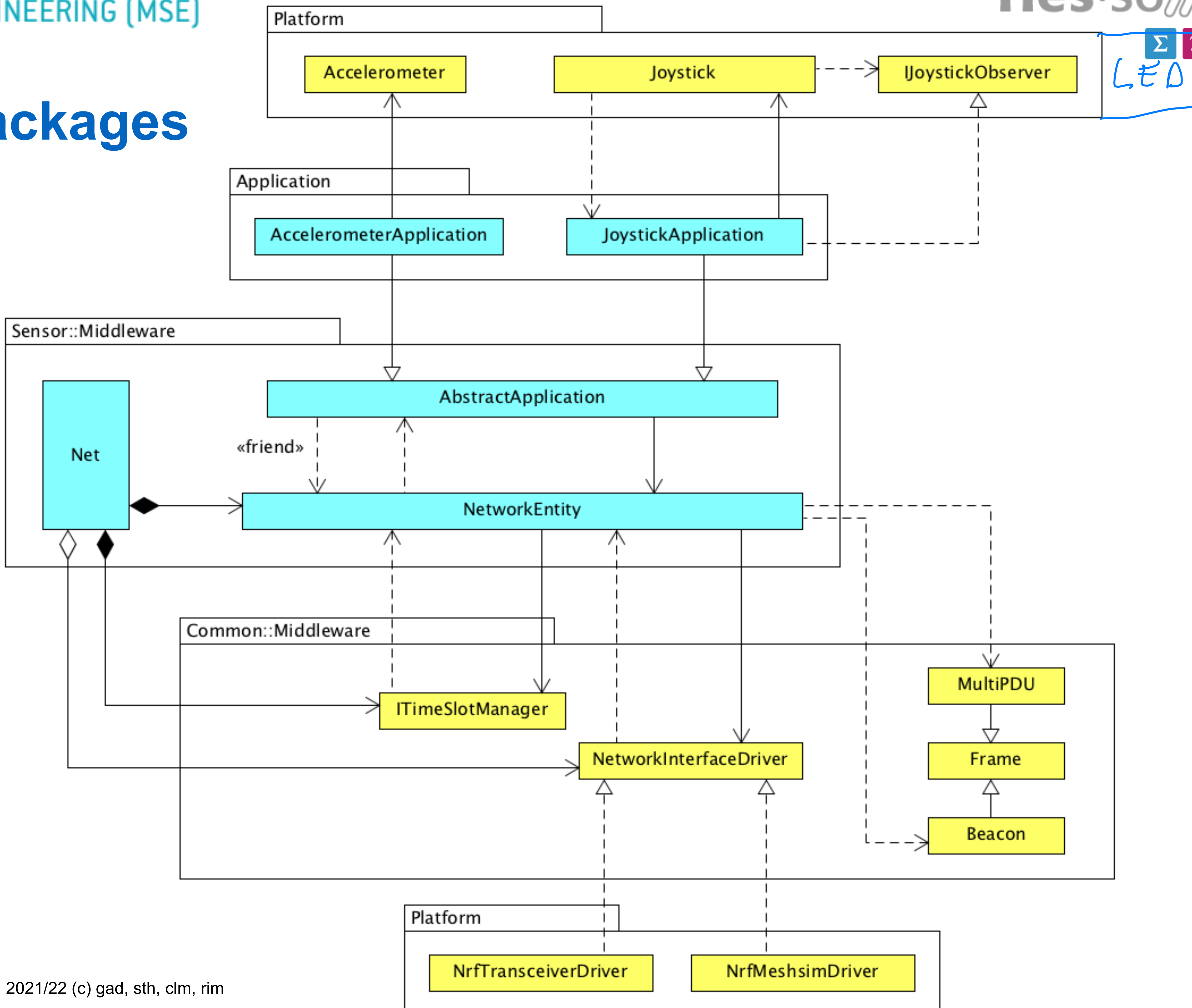


SAP

AbstractApplication
svSyncRequest():void # svPublishRequest(group:SvGroup):bool # evPublishRequest(id:EvId, evData:SharedByteBuffer):void # evSubscribeRequest(id:EvId):void – svSyncIndication(time:NetworkTime):void – svPublishIndication(group:SvGroup, svData:SharedByteBuffer):SharedByteBuffer::sizeType

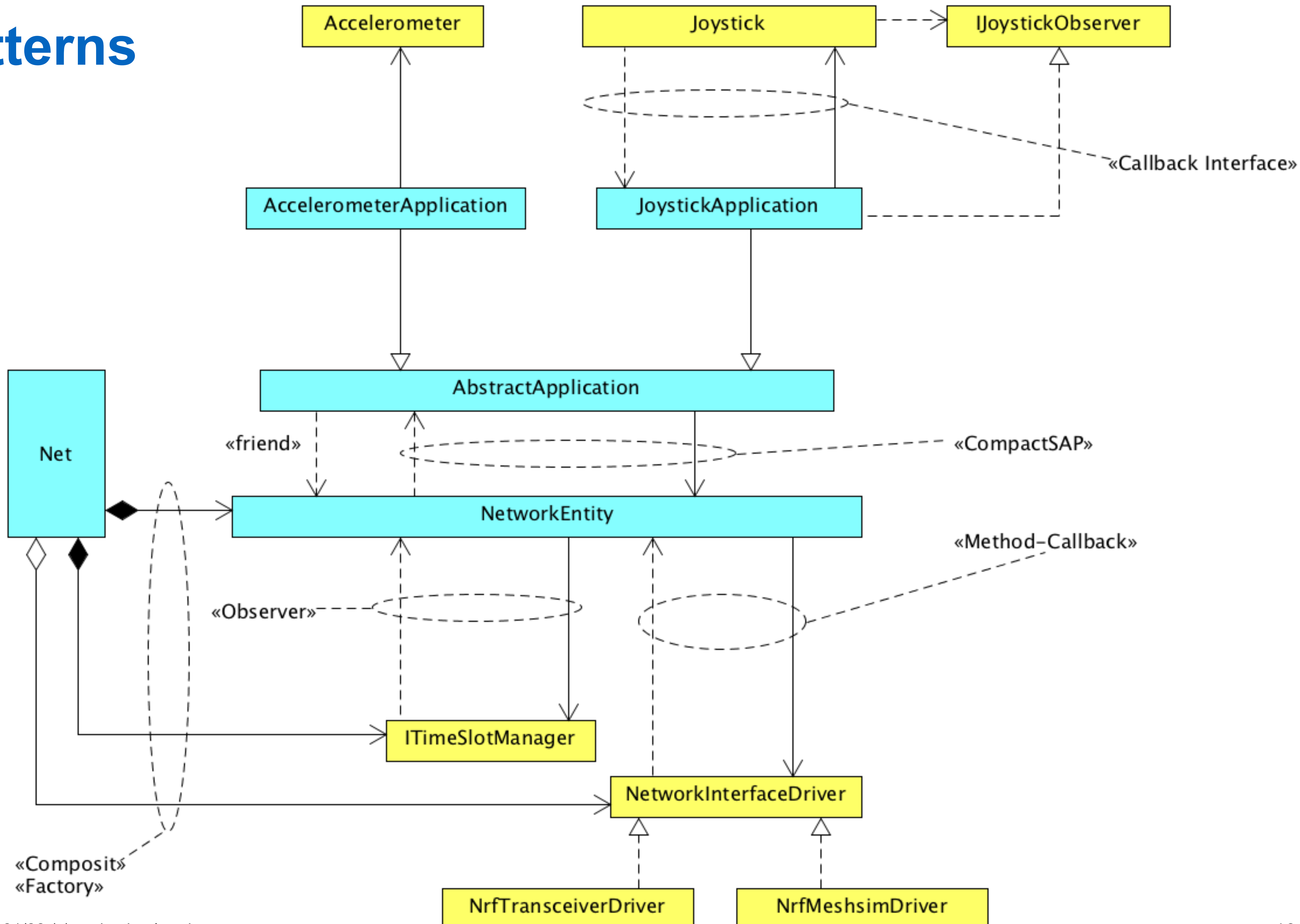
Packages

Protocol Design



Patterns

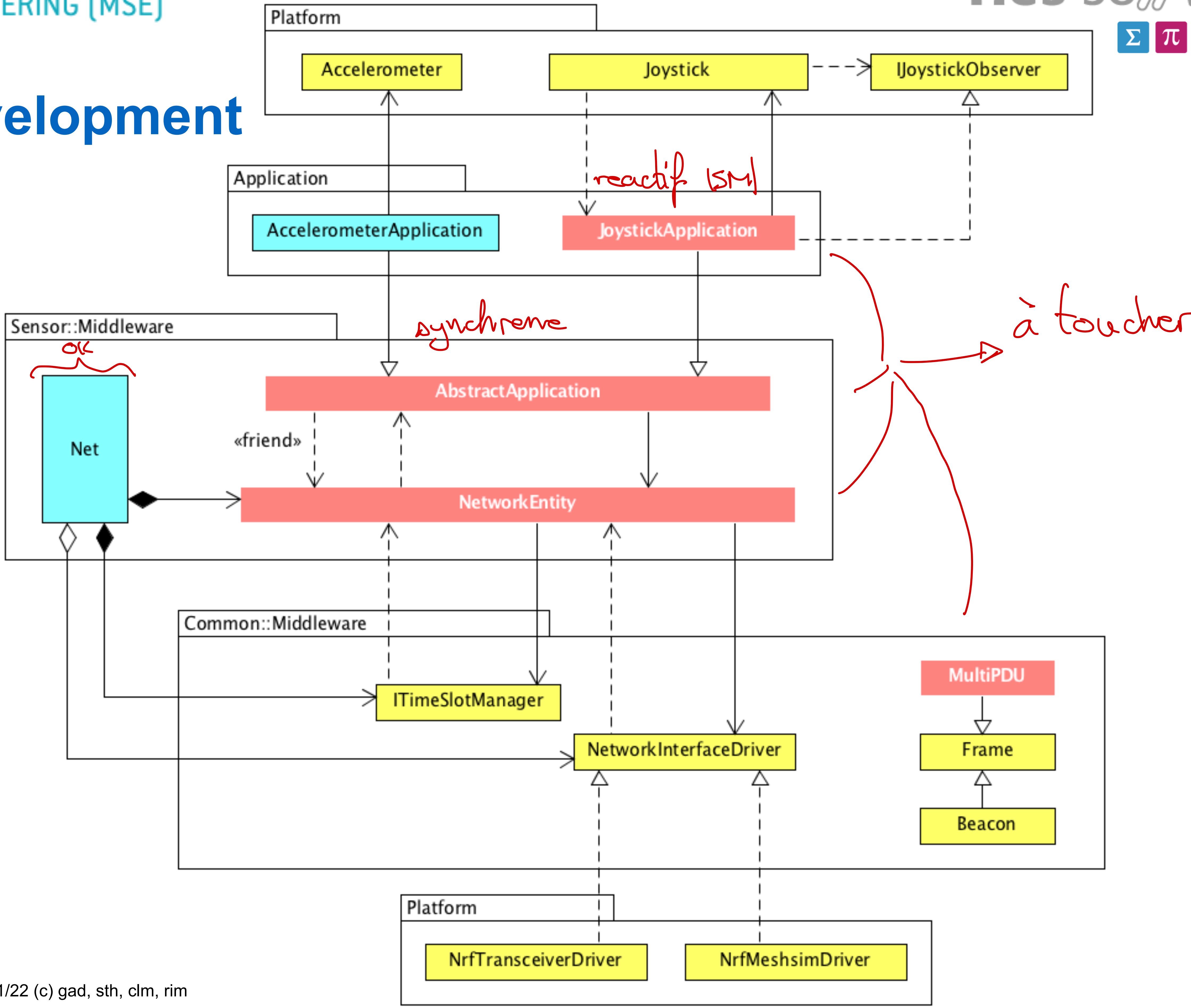
Protocol Design

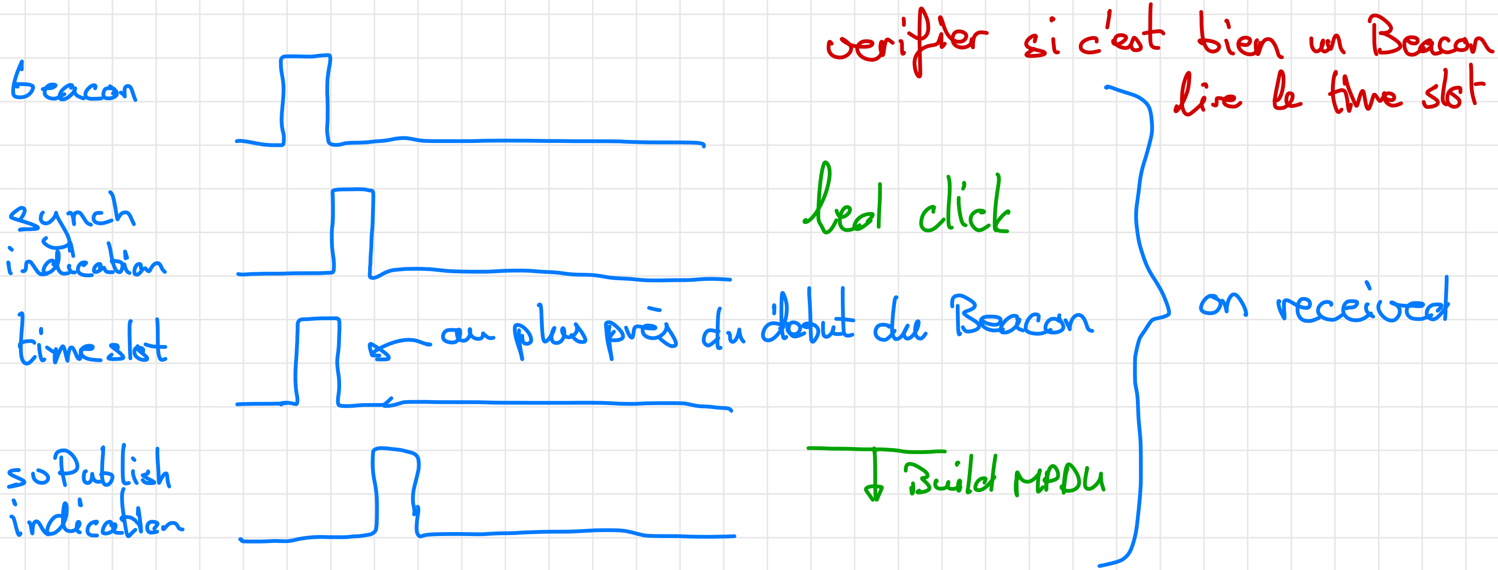


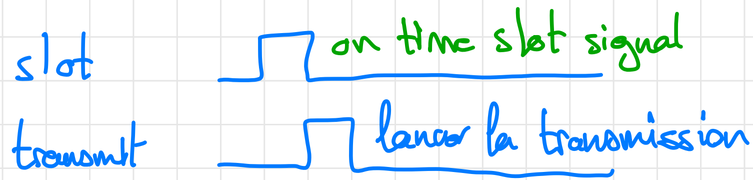
Master ENGINEERING (MSE)

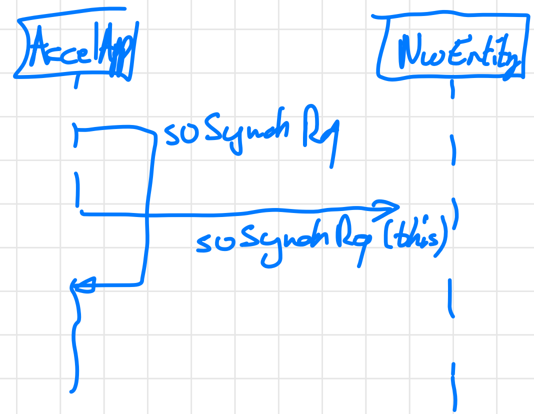
Development

Task Setting




verifier si c'est bien un Beacon
lire le time slot
beacon
synch indication
timeslot
soPublish indication
led click
au plus près du début du Beacon
Build MPDU
on received


slot
on time slot signal
transmit
lancer la transmission

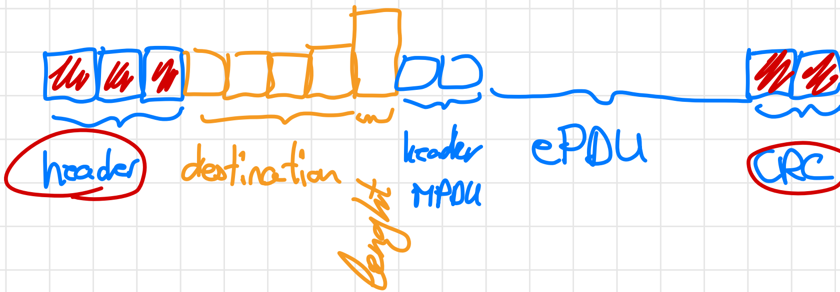

AccelApp
NwEntity
soSynchReq
soSynchReq(this)

Desenet - Thoughts

namespace desenet
#define
struct

mpdu

Frame



→ Not my Business!

1. Sensor ID
2. set type
3. set EPDUcnt
4.

dans cette ordre

reset MPDU → length
→ sensor ID
→ frame type

dans networkentity

↳ attribu MPDU nom

struct

union {
unsigned : 3
: 4
: 1
ordre important

EPDU
→ uint8

Development Process

Task Setting

- Study the preset project and the structure of it
- Understand classes, their relations and patterns used (using these slides)
- Develop the simulated solution using the Mesh-Sim environment
  - Step 1: Receive beacons
  - Step 2: Implement notification of the applications on reception of the beacon
  - Step 3: Implement MultiPDU class
  - Step 4: Collect sampled values, put them into MultiPDU and send it a the right slot
  - Step 5: Implement the Joystick application
- Test the simulated solution using the Mesh-Sim environment
  - Define and describe test cases
  - Test and document each test case
  - Generate an error / a todo list

Development Process continued

- „Port“ your simulated solution to the Nucleo target
  - Step 1: Rebuild for Nucleo target
  - Step 2: Flash run and debug it
- Test your Nucleo solution against the Gateway that will be present in class room
  - Define and describe test cases
  - Test and document each test case
  - Generate an error / a todo list
- Documentation
  - During all the development, create UML diagrams as possible or necessary. Use class and sequence diagrams as well as state charts.
  - Comment well your code !!!
- Delivery
  - Eclipse project without compiled code. UML diagrams in PDF format. Pack everything into a ZIP archive with the name „FirstnameLastname.zip“

How we will grade you

- This is how we will generate marks:
- No delivery at all : 1.0
- DeseNET protocol not working: 2.5
- DeseNET protocol and joystick application working on simulator (4.0)
- DeseNET protocol and joystick application tested on simulator and tests documented 5.0
- DeseNET protocol and joystick application working on target: 5.5
- DeseNET protocol and joystick application tested on target and tests documented 6.0
- No code documentation (-1.0)
- Bad or insufficient code documentation (-0.5)
- No model documentation (-1.0)
- Bad or insufficient model documentation (-0.5)
- No test documentation (-1.0)
- Bad or insufficient test documentation (-0.5)
- No pattern usage (-0.5)
- Copy from other : For involved persons maximum mark is 3.0

Plan Your Time well

Task Setting

9	22.11.21	Desem protocol entity design	Desem protocol design	Desem protocol design
10	29.11.21	Desem protocol implementation & test		
11	06.12.21			
12	13.12.21			
13	20.12.21			
14	10.01.22	Reserve week (Desem protocol implementation & test)		
15	17.01.22	Prepare Exams		
16	24.01.22	Exam		

Communication Theory	Communication Lab / Exercise	
		Project
Software Engineering Theory	Software Engineering Lab / Exercise	