Master
ENGINEERING (MSE)

Hes·so// VALAIS WALLIS

DeSEm
Laboratory - Development of the DeseNet Protocol Stack

Dominique Gabioud

Michael Clausen

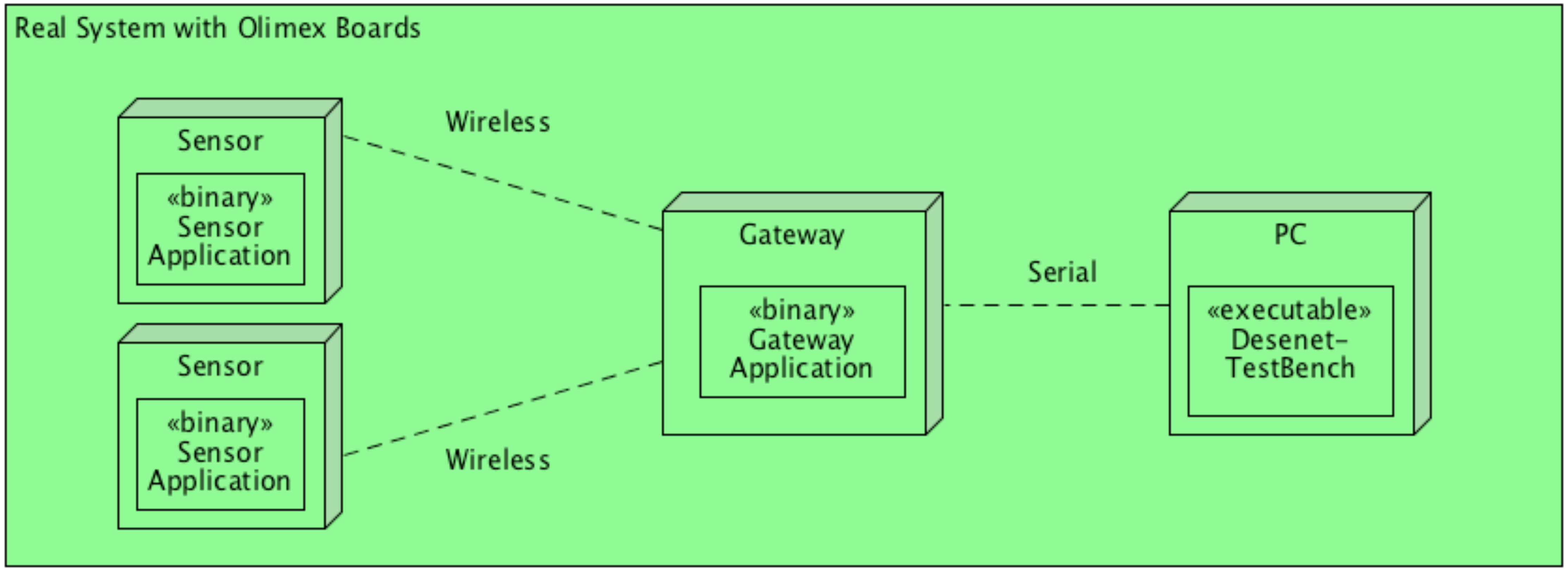
Thomas Sterren

Medard Rieder

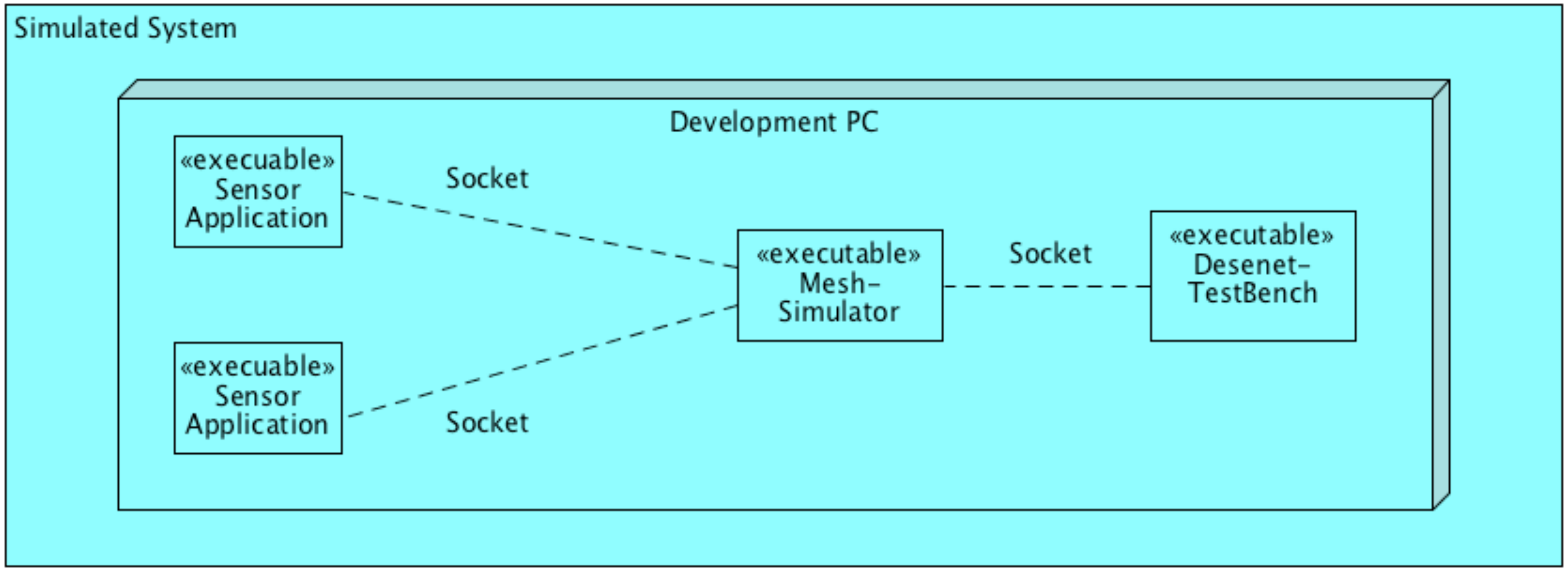
HES-SO 2023/24

Table of Content

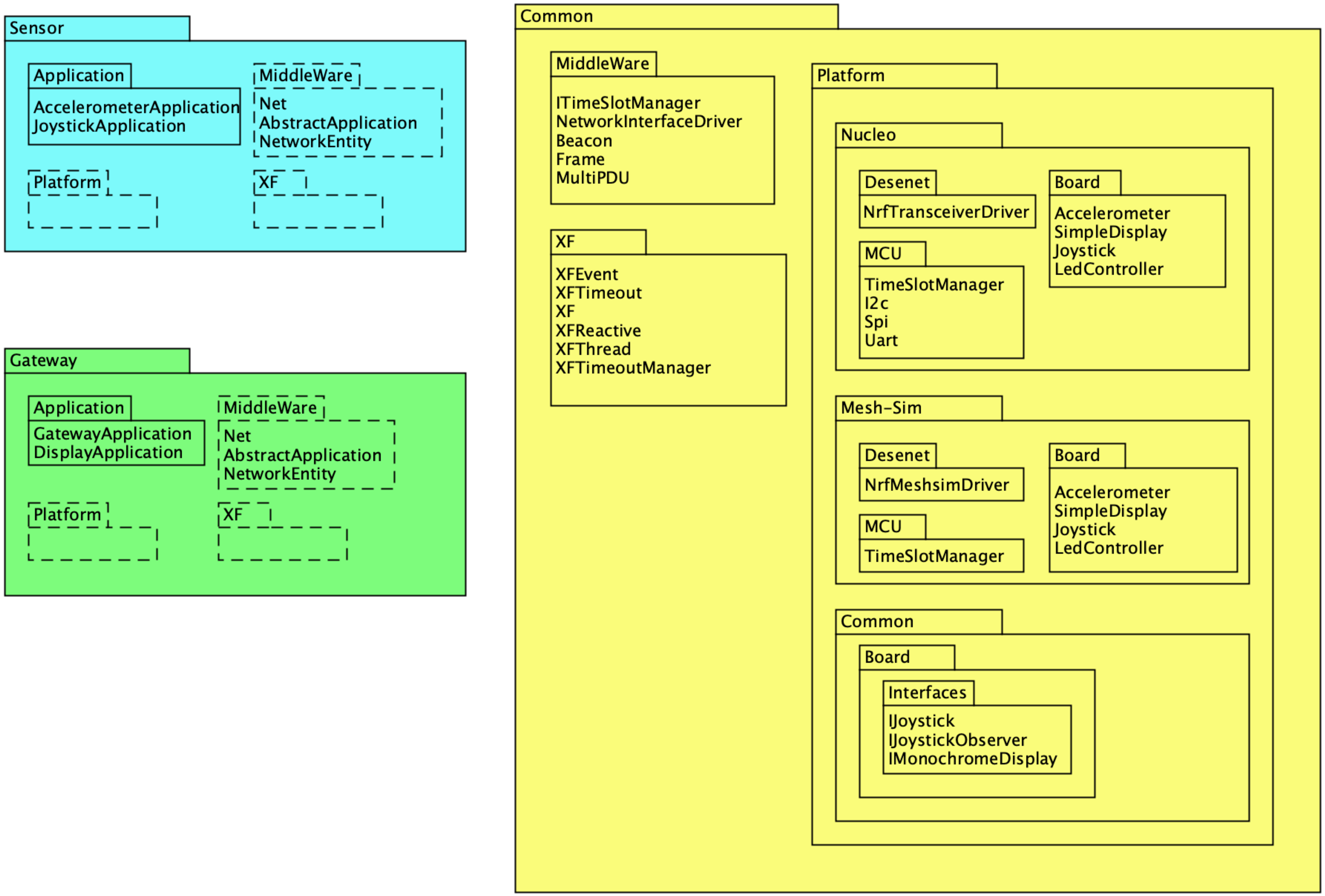
System with real Nodes


Real System with Olimex Boards
Sensor
«binary» Sensor Application
Sensor
«binary» Sensor Application
Wireless
Wireless
Gateway
«binary» Gateway Application
Serial
PC
«executable» Desenet-TestBench

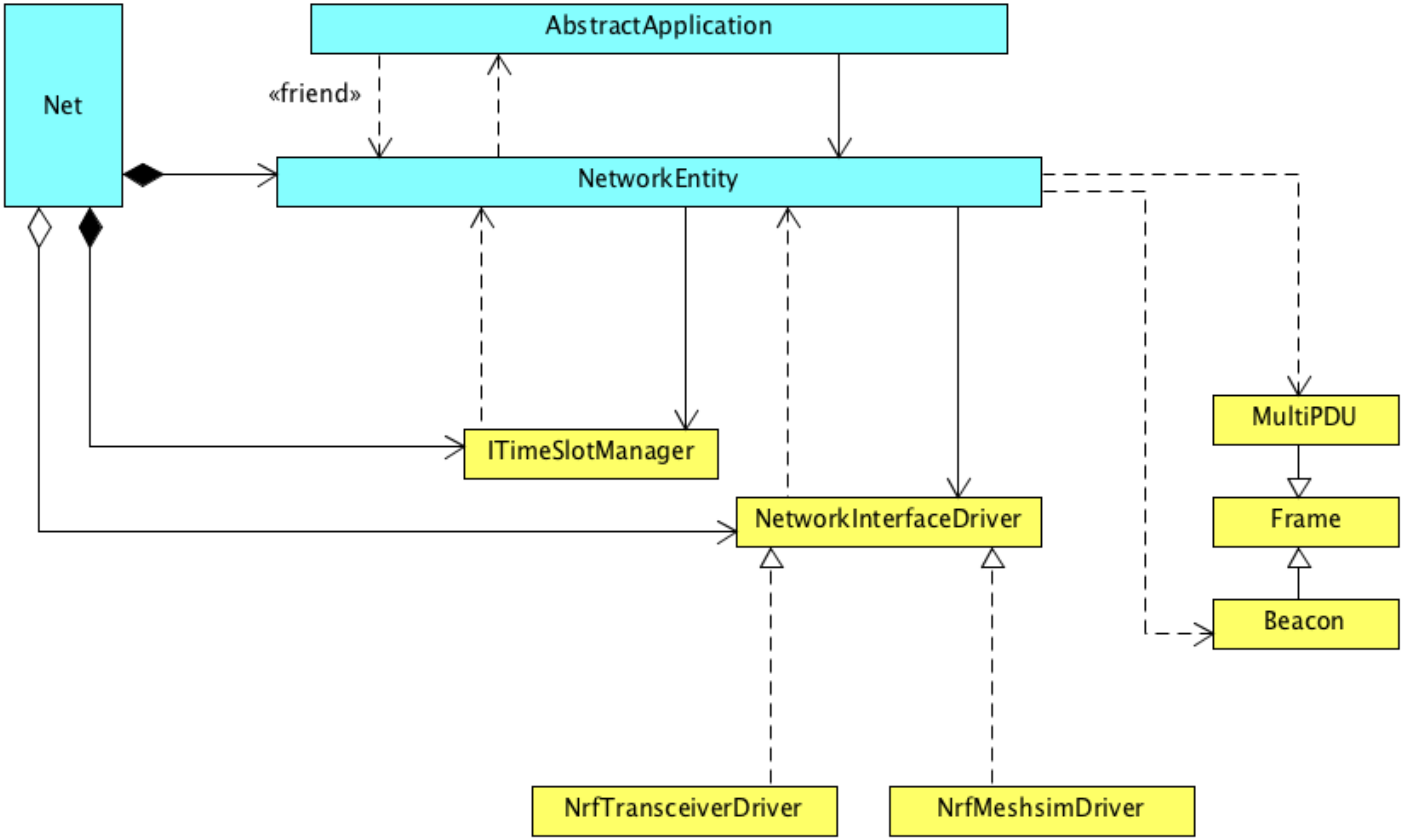
System with virtual Nodes



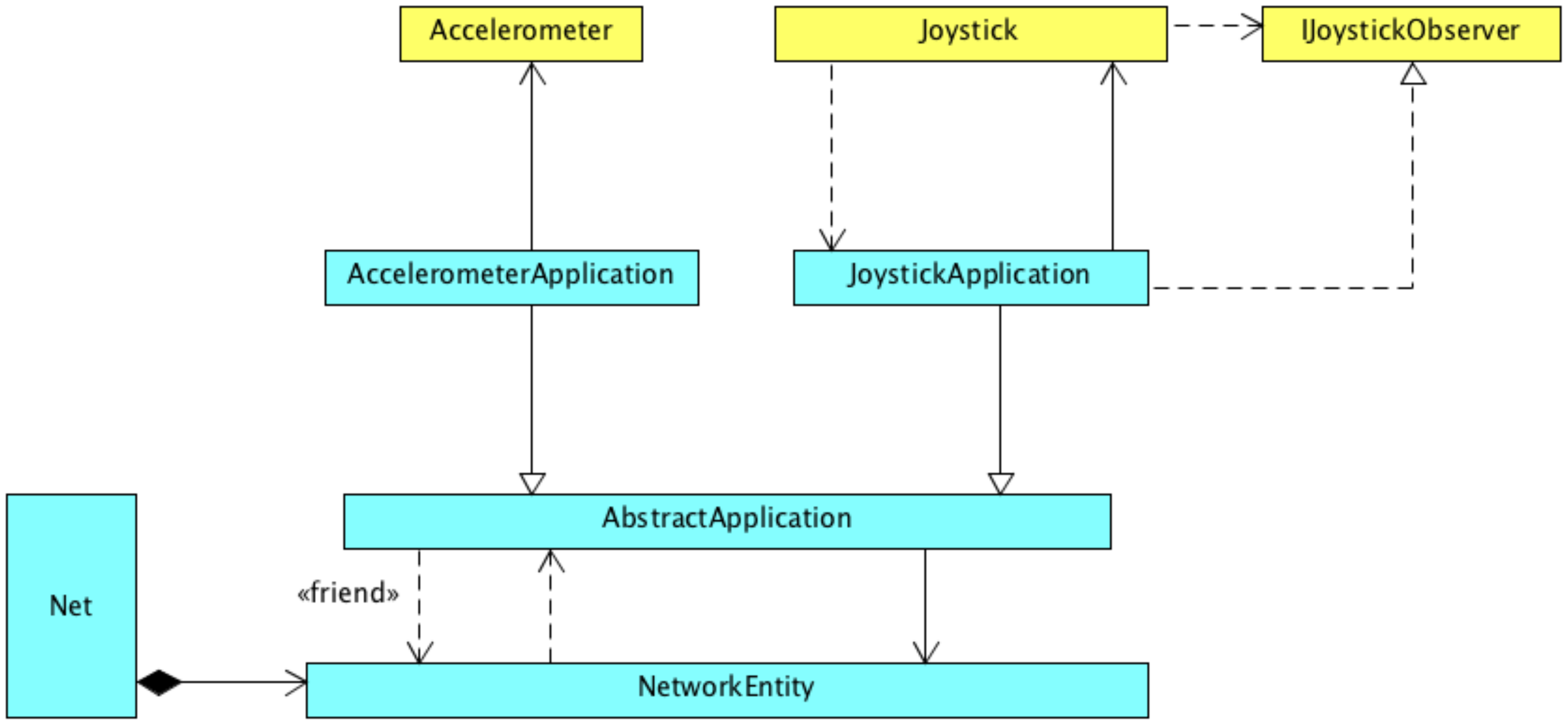
Architecture



Protocol Elements



Application Elements



SAP

networkEntity ist ein singleton -> durch NetworkEntity::Instance() gibt uns die Referenz zu dem objekt

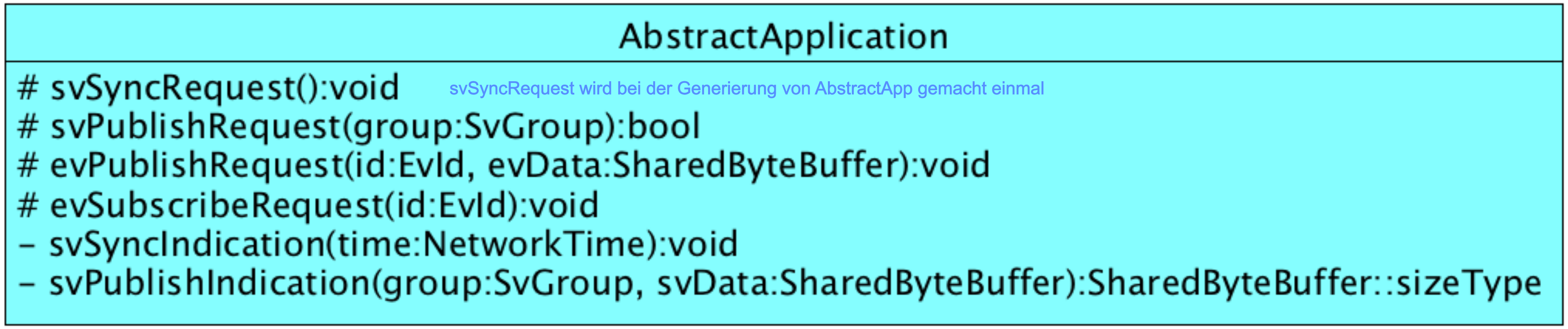
Dasselbe bei Klasse Net

Net::Instance().entity() gibt uns auch die referenz

networkEntity::onReceive() wird aufgerufen wenn ein Beacon kommt

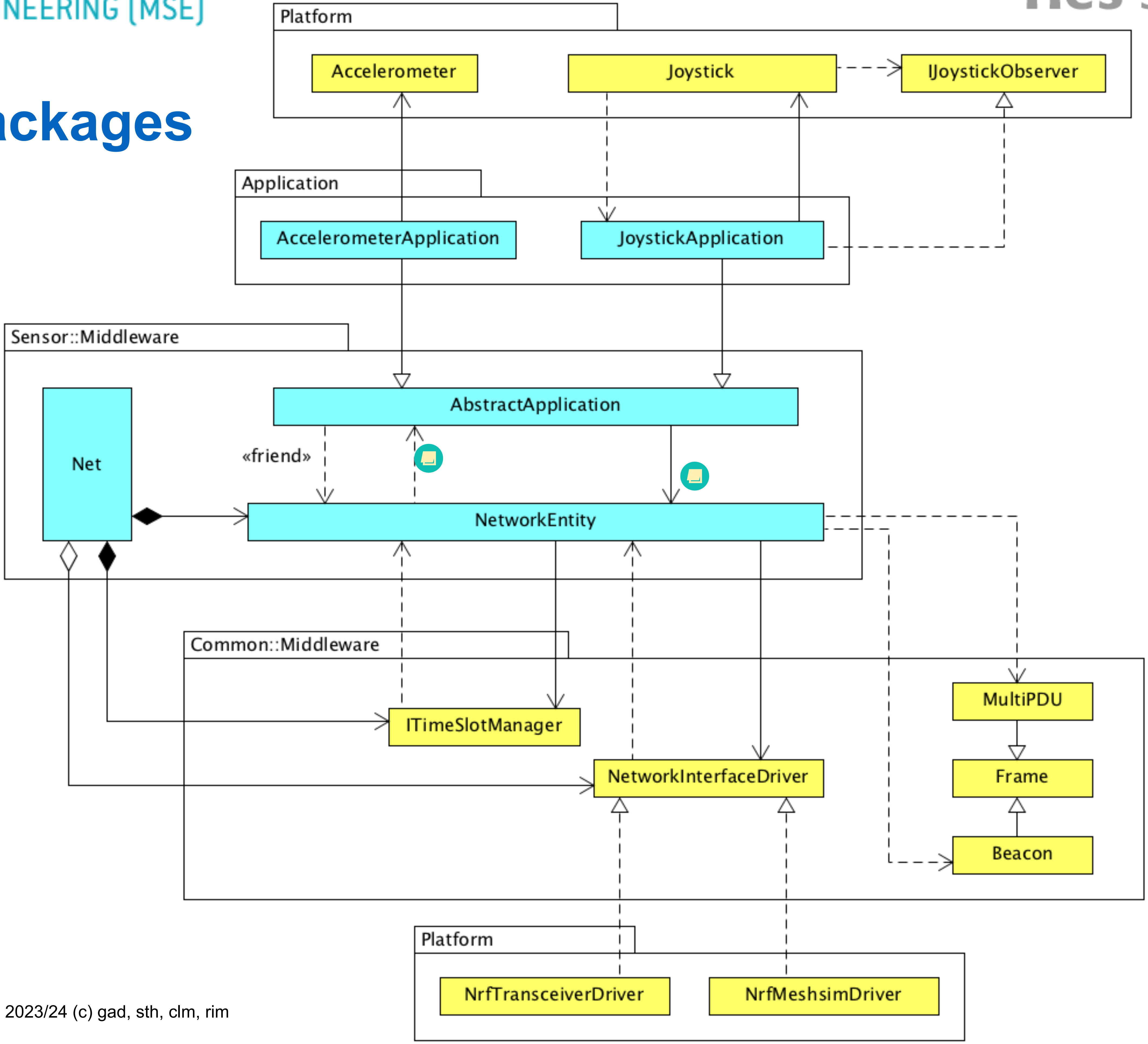
Klasse ledController hat eine Methode die led anlässt oder ausschaltet

Protocol Design


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

svSyncRequest wird bei der Generierung von AbstractApp gemacht einmal

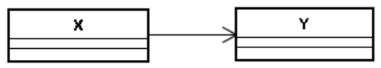
Packages


Platform
Accelerometer
Joystick
IJoystickObserver
Application
AccelerometerApplication
JoystickApplication
Sensor::Middleware
Net
AbstractApplication
«friend»
NetworkEntity
Common::Middleware
ITimeSlotManager
NetworkInterfaceDriver
MultiPDU
Frame
Beacon
Platform
NrfTransceiverDriver
NrfMeshsimDriver

Association

One object is aware of another; it contains a pointer or reference to another object.

Representaion



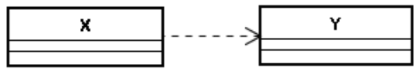
C++ Example

```
Class X {

  X(Y *y) : y_ptr(y) {}

  void SetY(Y *y) { y_ptr = y;   }

  void f()        { y_ptr->Foo();}
  ----
  Y *y_ptr; // pointer
};
```

Dependency

One class depends on another if the independent class is a parameter variable or local variable of a method of the dependent class

Representaion



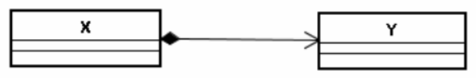
C++ Example

```
class X {
 ...
 void f1(Y y)  {…;  y.Foo();       }
 void f2(Y *y) {…;  y->Foo();      }
 void f3(Y &y) {…;  y.Foo();       }
 void f4()     {   Y y; y.Foo();  …}
 void f5()     {…; Y::StaticFoo(); }
 ...
};
```

Composition

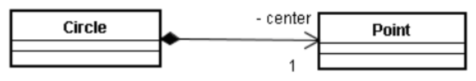
Composition is the stronger form of aggregation. Composition can occur when a class is a collection or container of other classes, but where the contained classes have a strong life cycle dependency on the container—essentially, if the container is destroyed, its contents are also destroyed

Representation

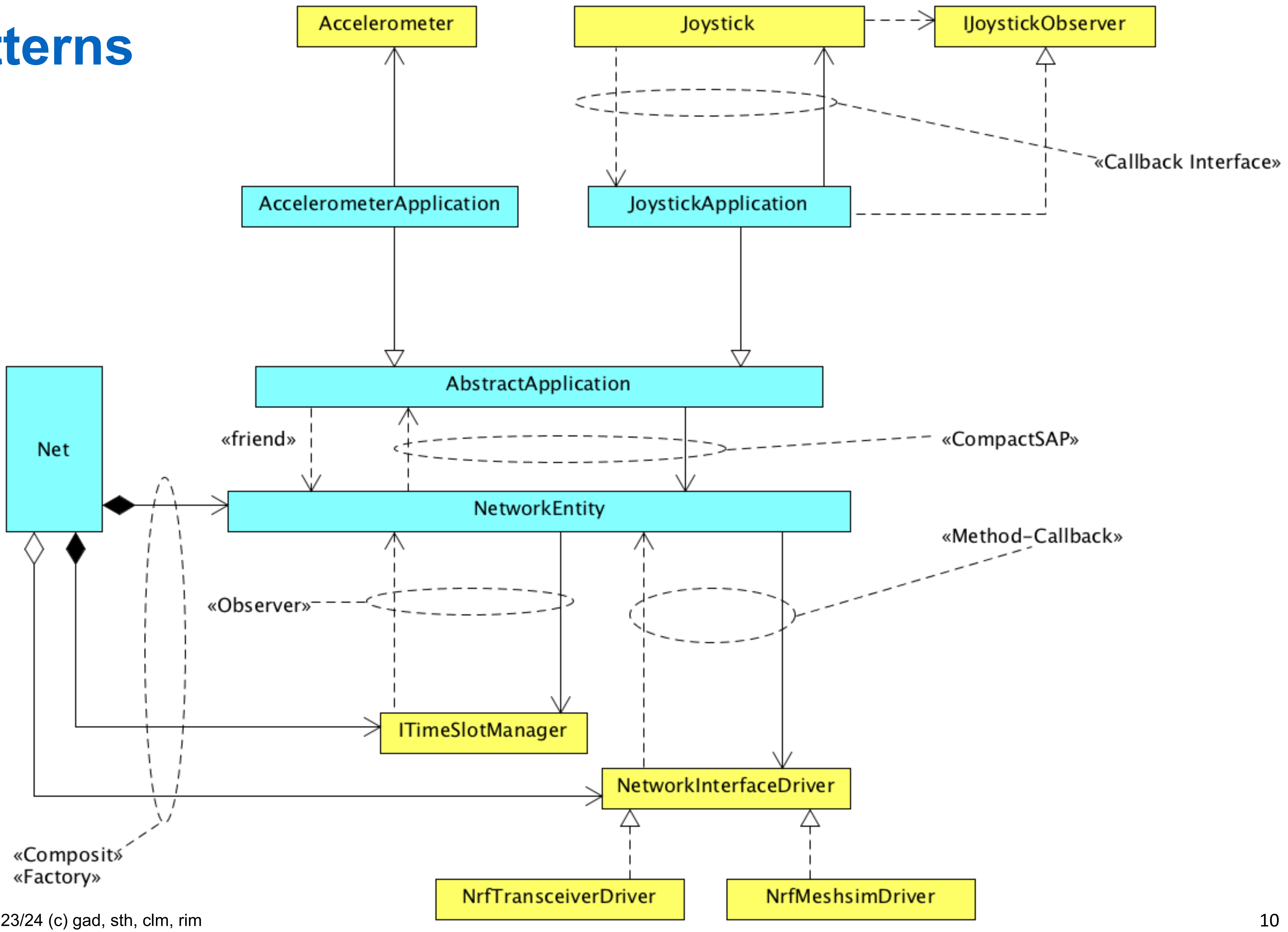


C++ Example

```
class Circle
{
private:
      ...
    Point center;
....
};
```

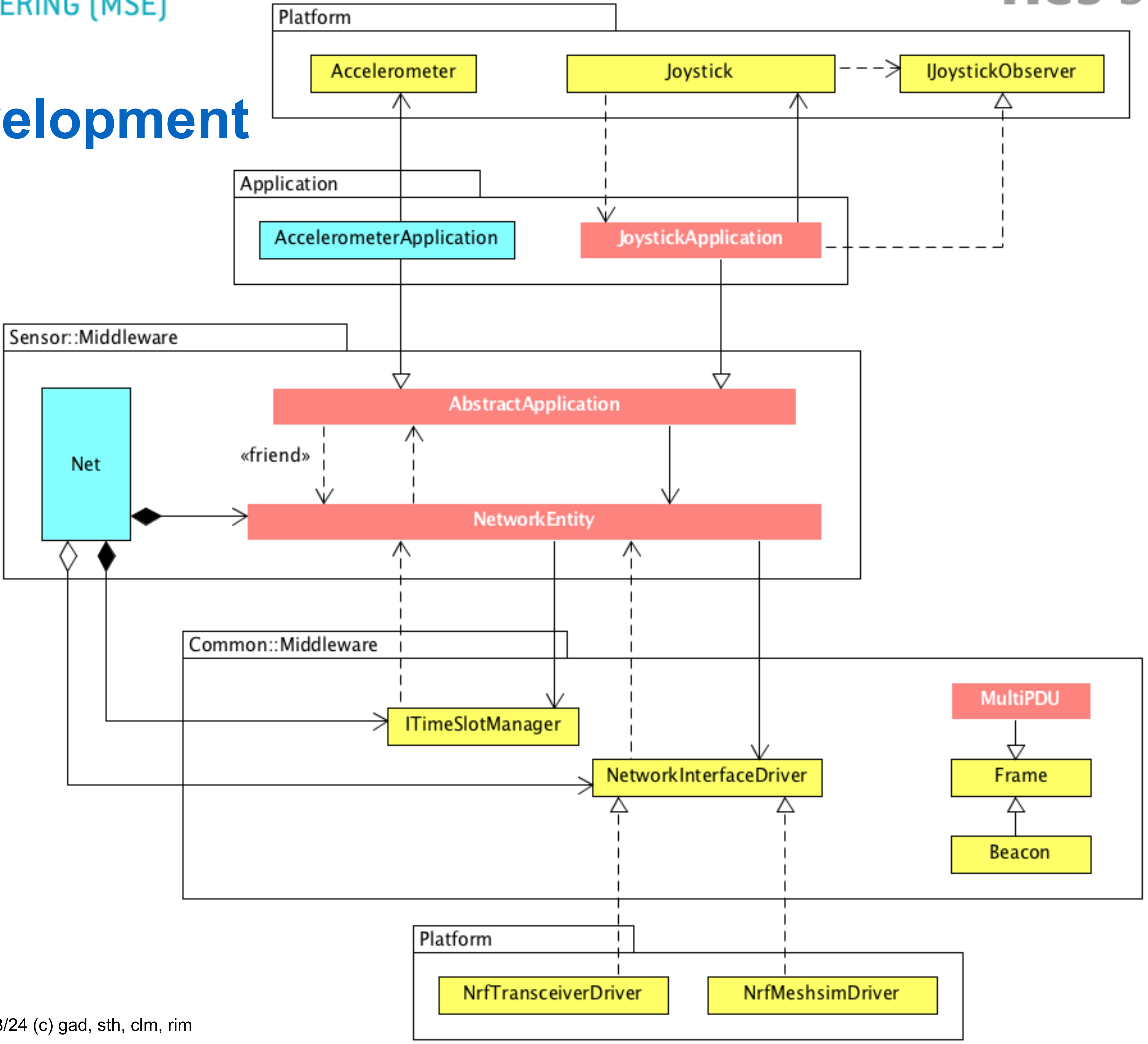


Patterns


Accelerometer
Joystick
IJoystickObserver
«Callback Interface»
AccelerometerApplication
JoystickApplication
AbstractApplication
Net
«friend»
«CompactSAP»
NetworkEntity
«Method-Callback»
«Observer»
ITimeSlotManager
NetworkInterfaceDriver
«Composit»
«Factory»
NrfTransceiverDriver
NrfMeshsimDriver

Development

Task Setting



Development Process

- 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

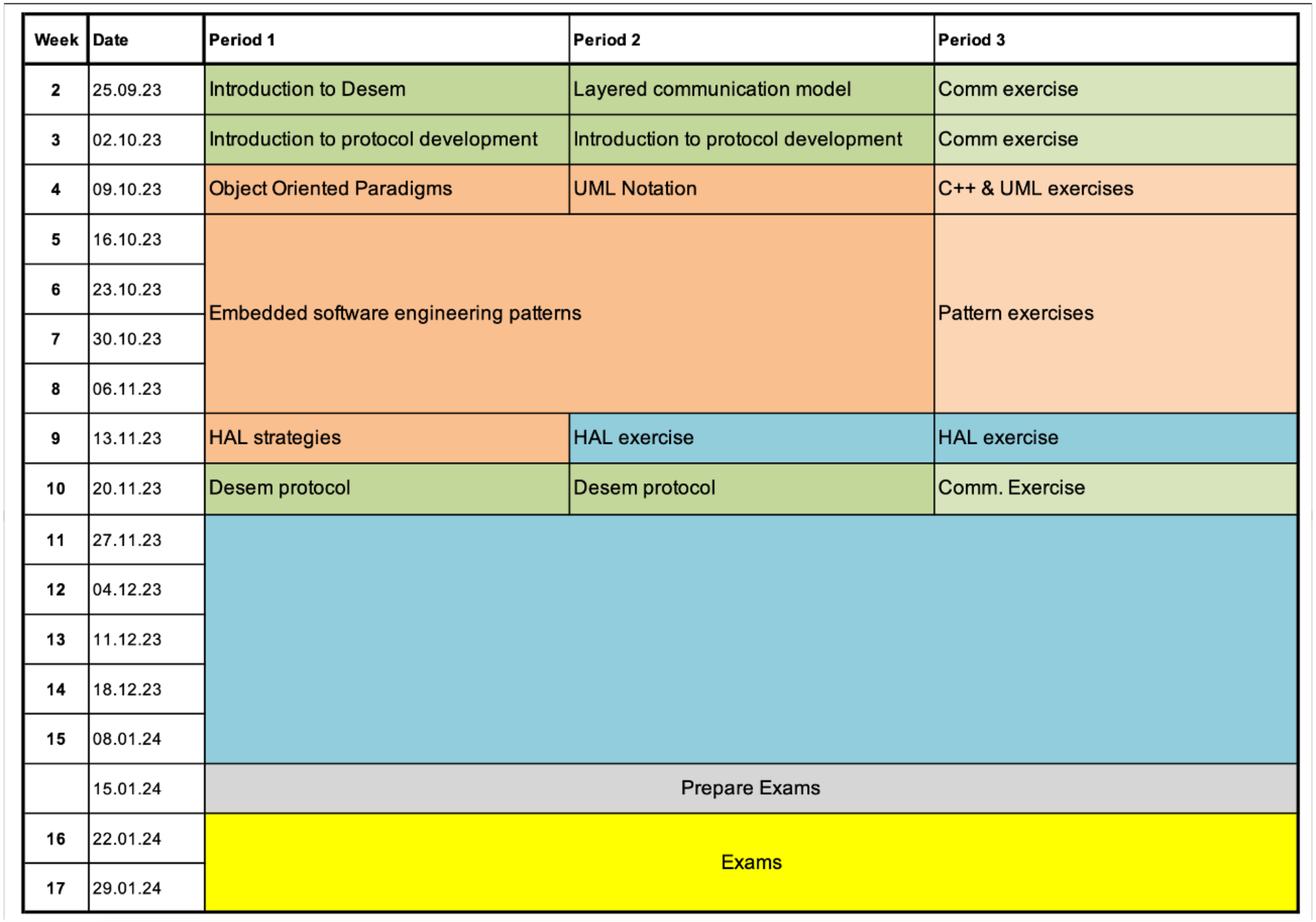
Development Process continued

- „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



Week	Date	Period 1	Period 2	Period 3
2	25.09.23	Introduction to Desem	Layered communication model	Comm exercise
3	02.10.23	Introduction to protocol development	Introduction to protocol development	Comm exercise
4	09.10.23	Object Oriented Paradigms	UML Notation	C++ & UML exercises
5	16.10.23	Embedded software engineering patterns	Embedded software engineering patterns	Pattern exercises
6	23.10.23	Embedded software engineering patterns	Embedded software engineering patterns	Pattern exercises
7	30.10.23	Embedded software engineering patterns	Embedded software engineering patterns	Pattern exercises
8	06.11.23	Embedded software engineering patterns	Embedded software engineering patterns	Pattern exercises
9	13.11.23	HAL strategies	HAL exercise	HAL exercise
10	20.11.23	Desem protocol	Desem protocol	Comm. Exercise
11	27.11.23			
12	04.12.23			
13	11.12.23			
14	18.12.23			
15	08.01.24			
	15.01.24	Prepare Exams	Prepare Exams	Prepare Exams
16	22.01.24	Exams	Exams	Exams
17	29.01.24	Exams	Exams	Exams

UM SIM ZU STARTEN:

Zuerst Meshsim öffnen. Danach open file -> in 7_Project-> .msim öffnen

Danach testbench öffnen und send continuous beacon drücken.

danach auf qt das Programm runnen und den neuen node mit dem gateway (alias testbench) verbinden.