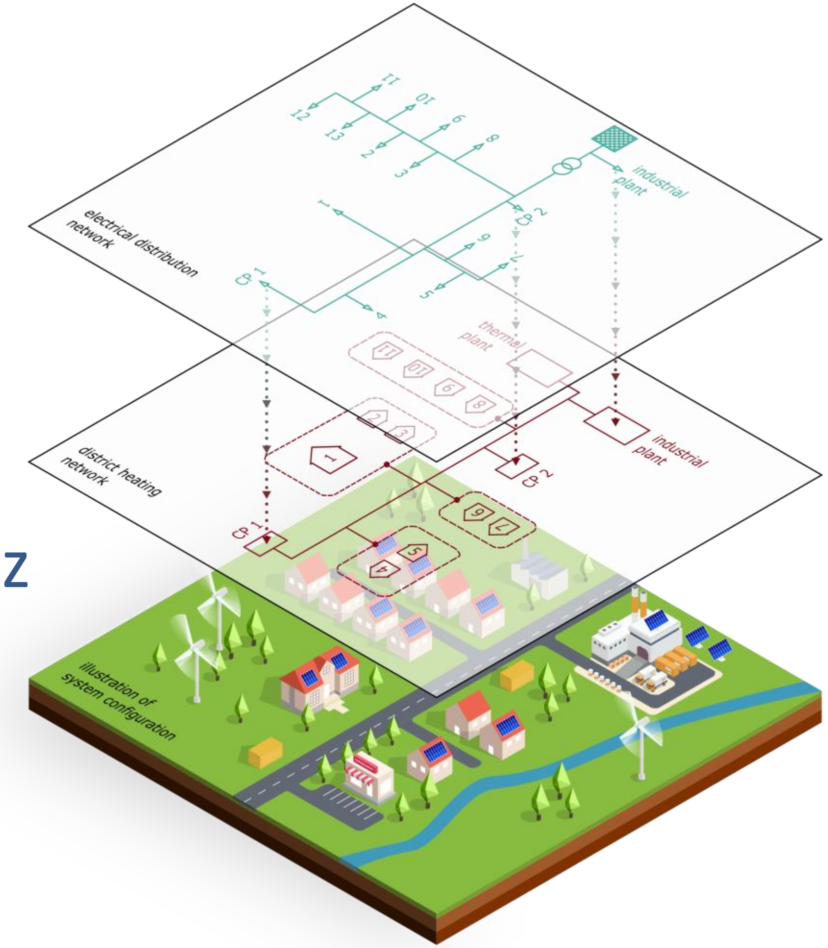
Connecting European
Smart Grid Infrastructures

Modelling and simulation of integrated energy systems

Edmund Widl, Peter Palensky, Jan Sören Schwarz

ERIGrid 2.0 / SINERGY / RESili8 Online Training Lecture Part 1



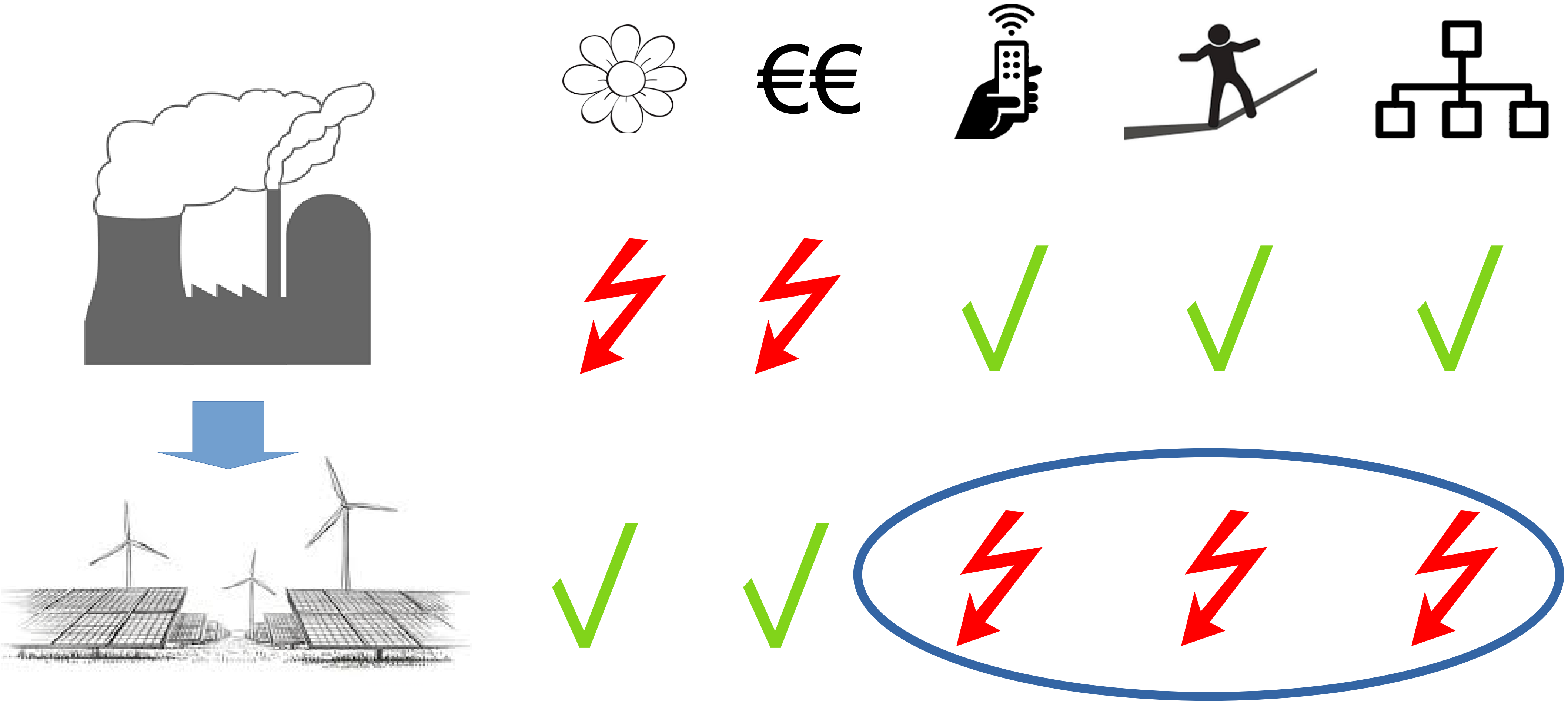
EU H2020 Programme GA No. 870620

Outline

- Part A: **Introduction to co-simulation**
  - → What is co-simulation?
  - → Why is co-simulation relevant for the assessment of future energy systems?
- Part B: **Co-simulation with mosaik**
  - → Basics of using mosaik for co-simulation
- Part C: **Example multi-energy network application**
  - → Assessment of a coupled thermo-electrical network

Part A: Introduction to co-simulation

Decarbonization of our Energy System



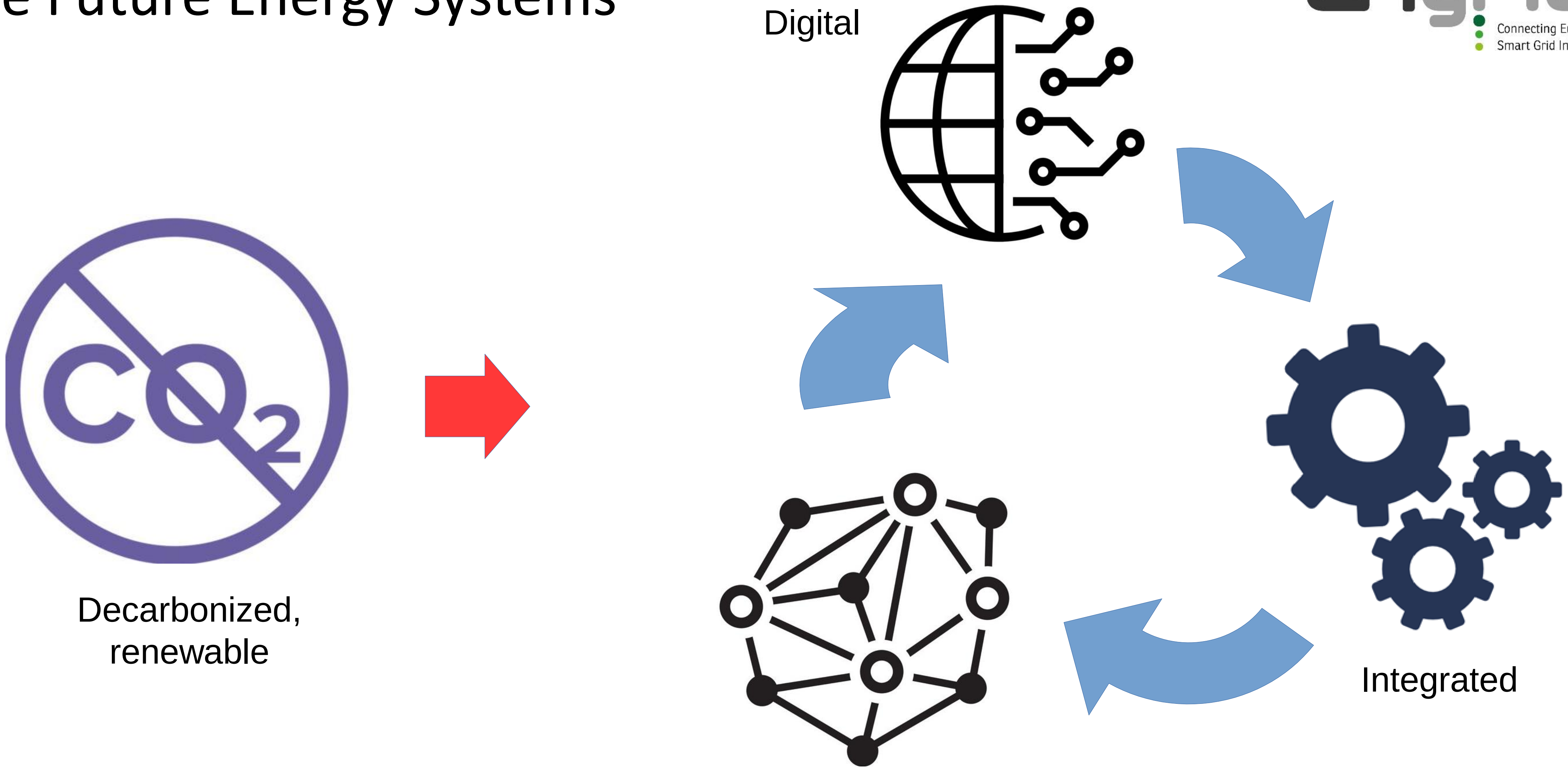
control,
operations

stability

roles,
stakeholders, relations

EU H2020 Programme GA No. 870620

The Future Energy Systems



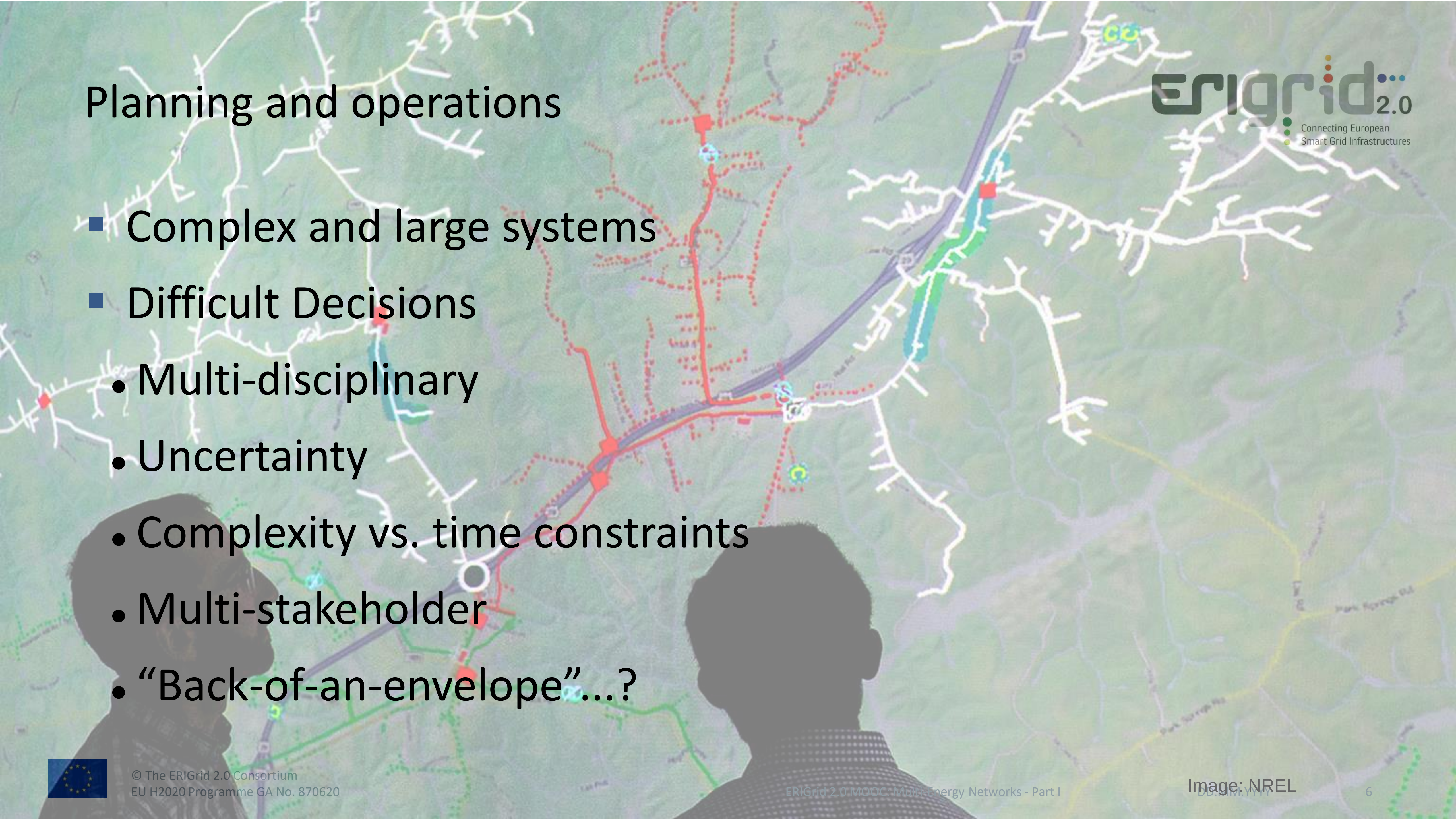
Decarbonized, renewable

Digital

Integrated

Distributed

Planning and operations

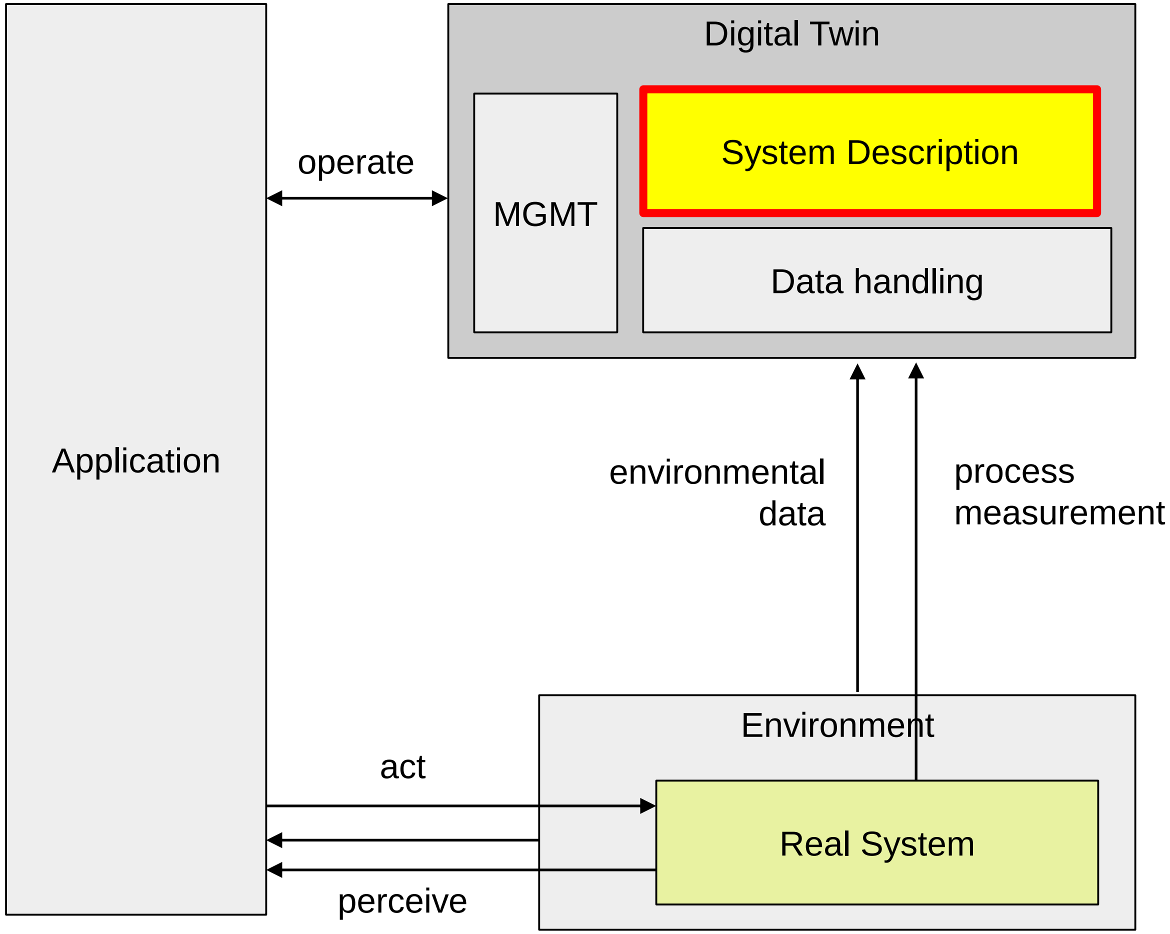


- Complex and large systems
- Difficult Decisions
  - Multi-disciplinary
  - Uncertainty
  - Complexity vs. time constraints
  - Multi-stakeholder
  - “Back-of-an-envelope”...?

Image: NREL

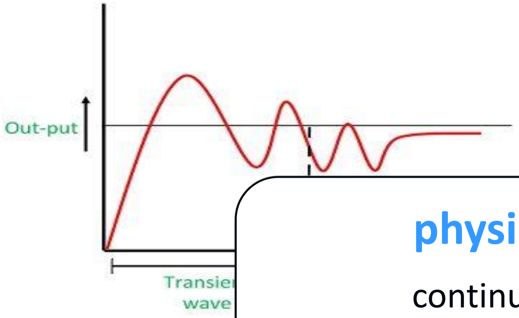
Support by numerical models

- During design phase
  - Dimensioning
  - Stability
  - Interoperability check
- Support of operations
  - Sanity/safety check
  - Digital Twins
- Post-mortem forensics



MGMT: Management

Describing the future Energy Network(s)



physical world

continuous models

energy generation, transport, distribution, consumption, etc.

roles/behavior

game theory models

agents acting on behalf of a customer, market players, etc.

cyber-physical energy system

information technology

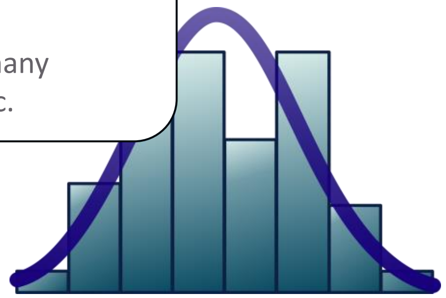
discrete models

controllers, communication infrastructure, software, etc.

aggregate/stochastic

statistical models

weather, macro-view of many individual elements, etc.



Future Energy System is...

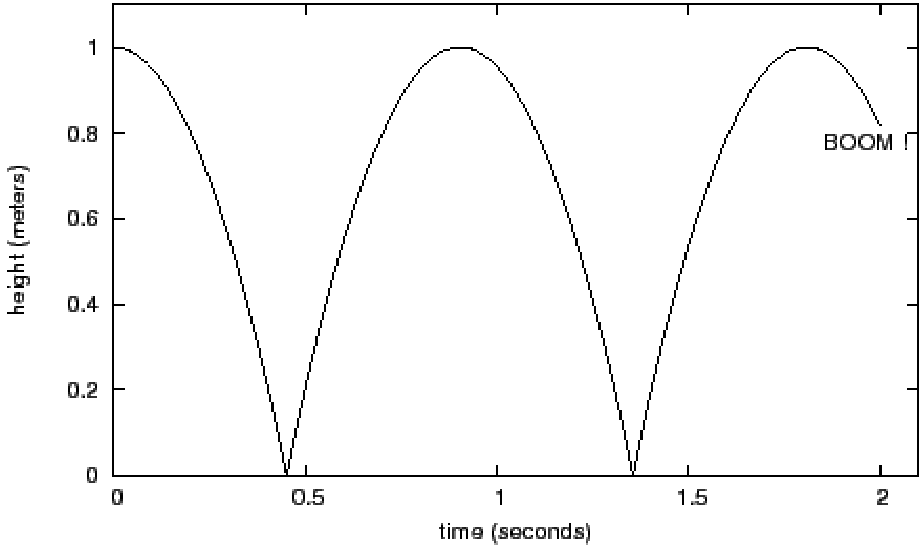
- **Cyber-physical** (discrete+continuous)
- **Multi-physical** (heat, power, gas,…)
- **Multi-timescale** (power electronics, hydraulics,…)
- **Complex** (hidden states, emerging behavior)
- **Probabilistic** (rare high impact events)

→ how to model…?

Good question, Einstein ...

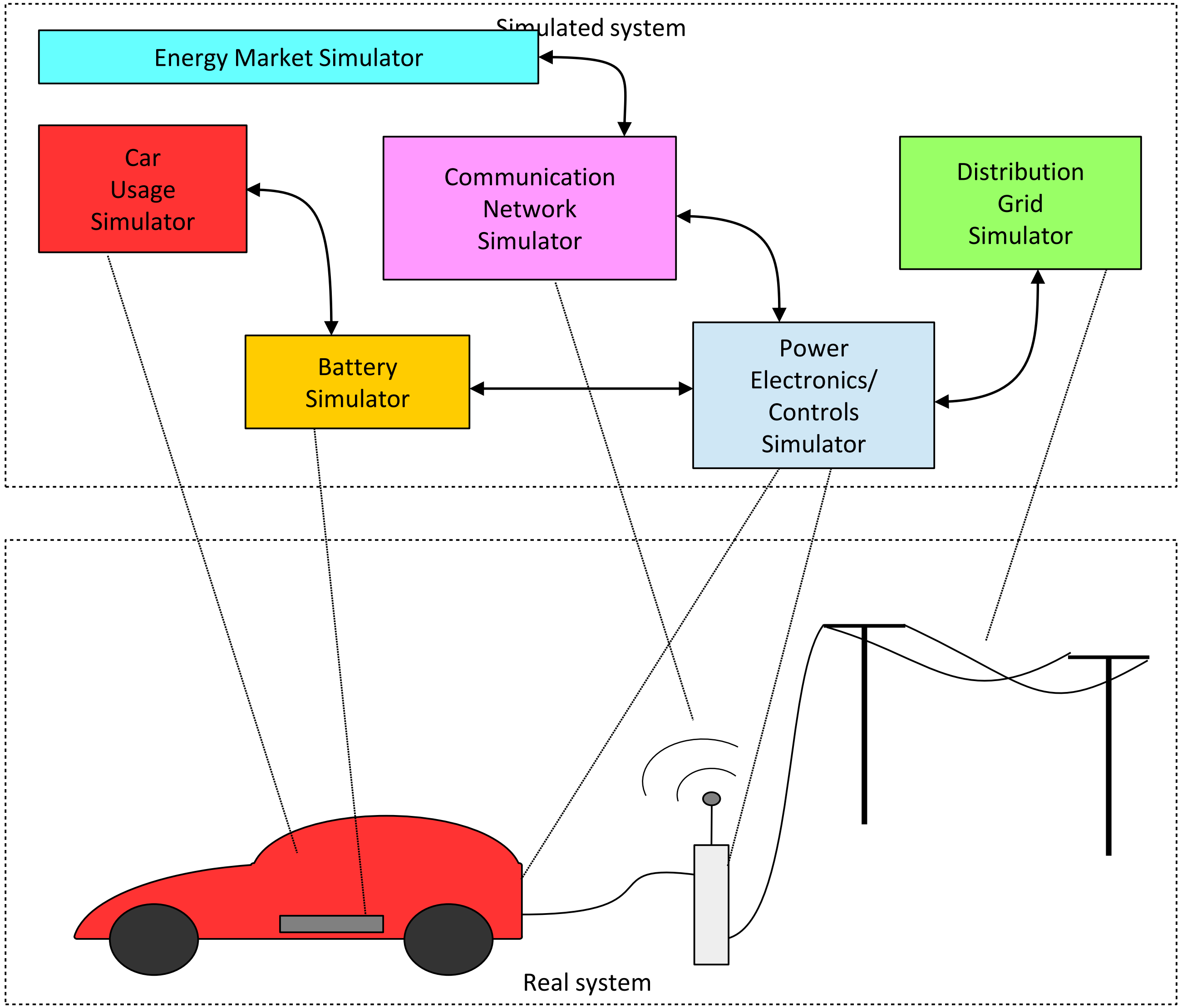
Options to model/simulate

- (1) Squeeze all submodels into one tool, language, solver, method
  - n-1 submodels in wrong language
  - Tedious, Error prone
  - Brutal simplifications
- (2) Universal tool
  - Universal language, solver, etc.
  - Performance problems
- (3) Combine specialized languages, solvers, tools?



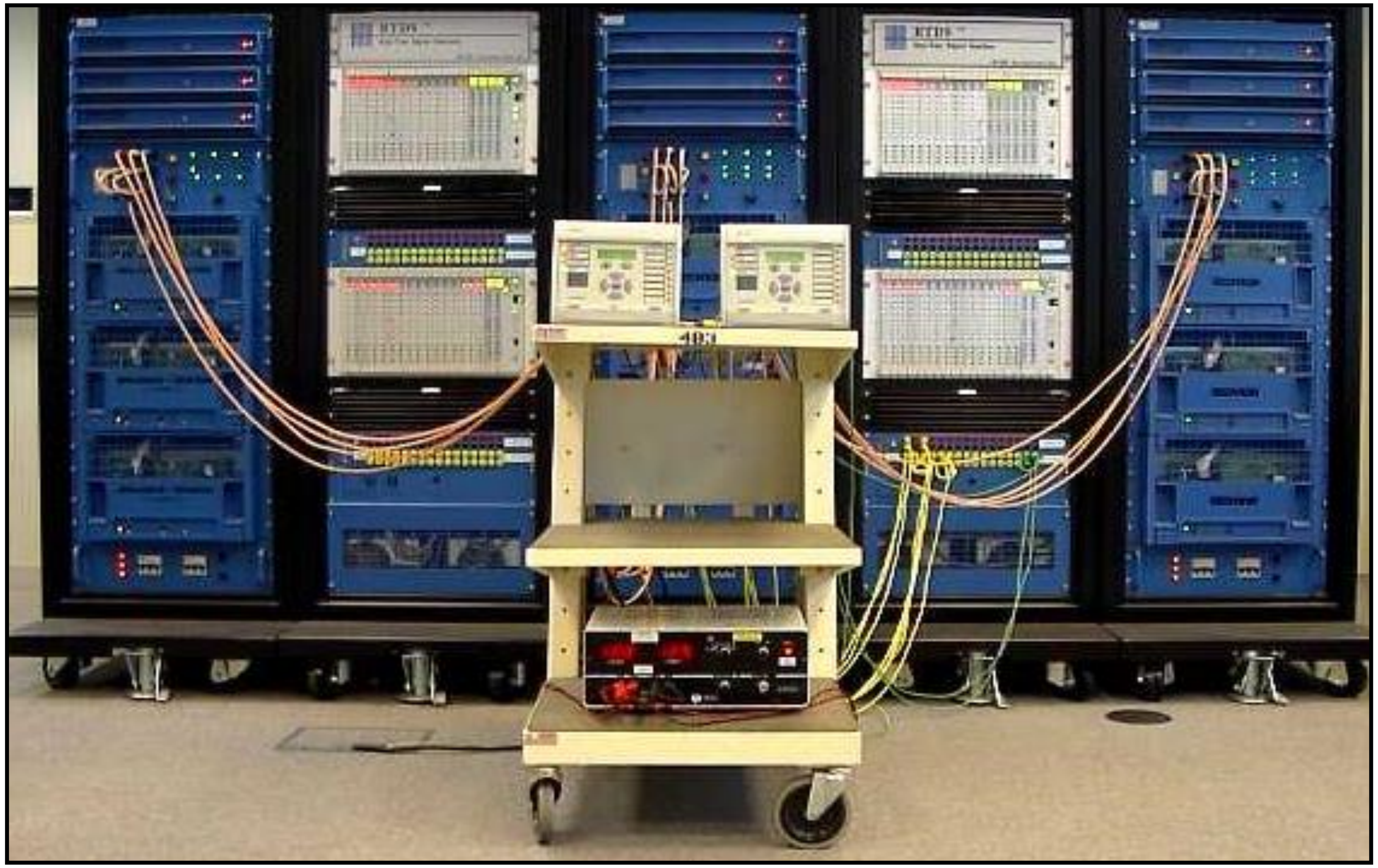
Connecting models/tools!

- **Co**mbine numerical models and run solvers **co**ncurrently
- Solve **co**llaboratively
- Multi-disciplinary, **co**nnected problems and teams possible
- “**Co**-Simulation”



Real-time Co-Simulation

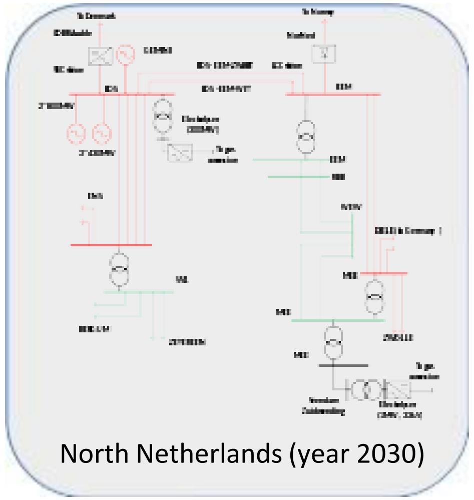
- Run in real-time (not faster, not slower)
- Implicit synchronization
- Coupled (physical) variables
- In the loop
  - Controller
  - Power HW
  - People
  - Software



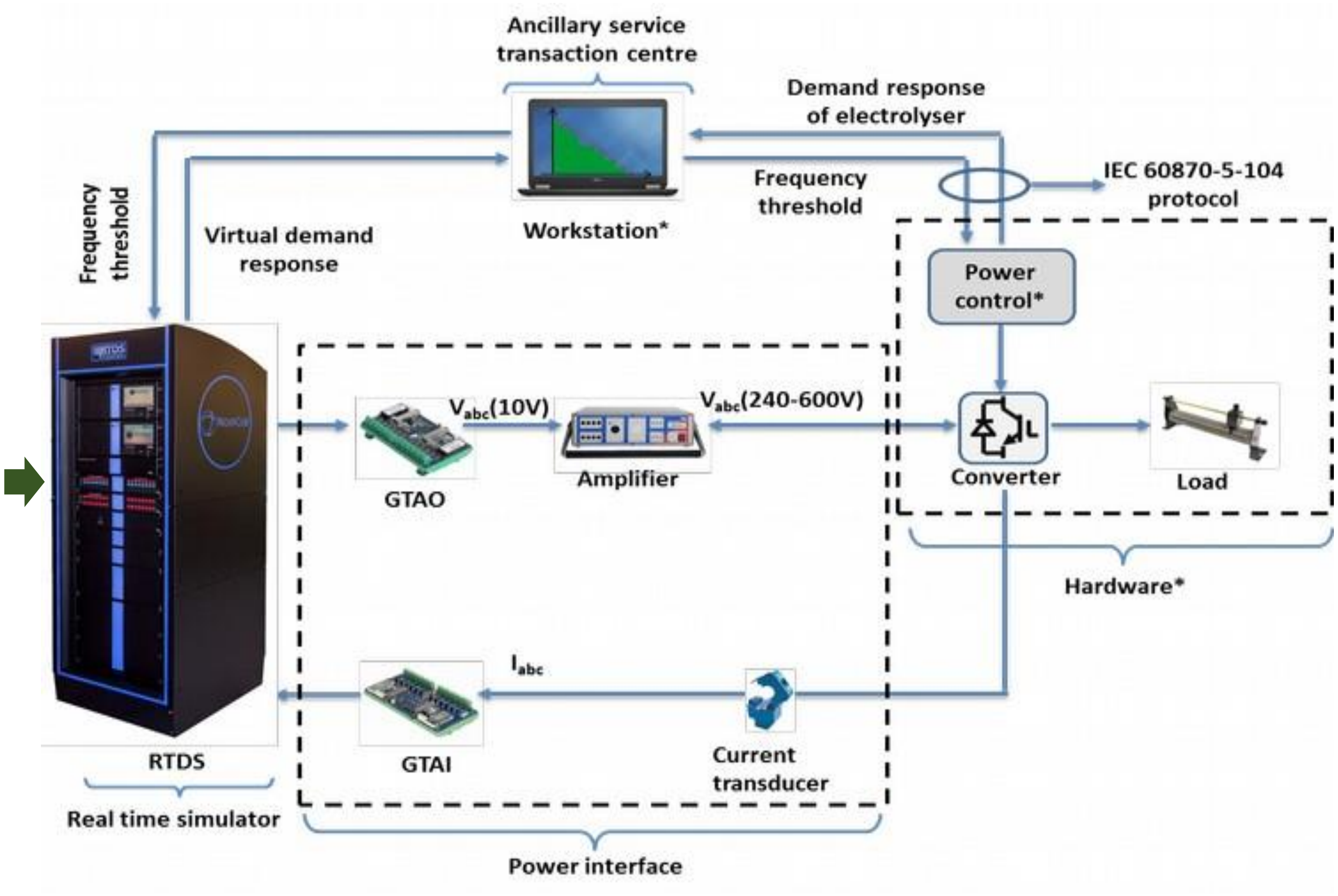
Source: RTDS Technologies, “Real time digital simulation: Modelling renewable energy applications,” Feb. 2018.

RT: Ancillary Services from Hydrolyzers

EMT system model



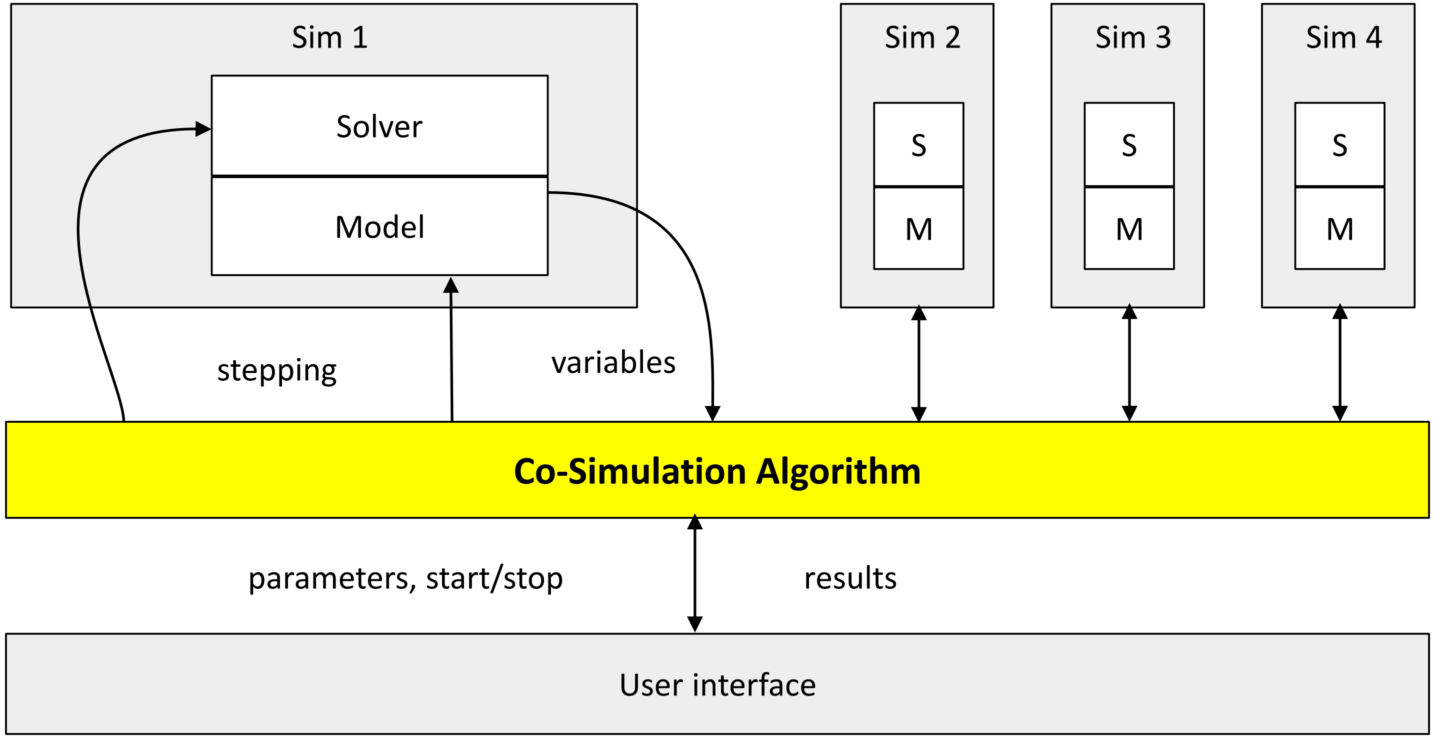
North Netherlands (year 2030)



EU H2020 Programme GA No. 870620

Non-real time: Co-Simulation Algorithm

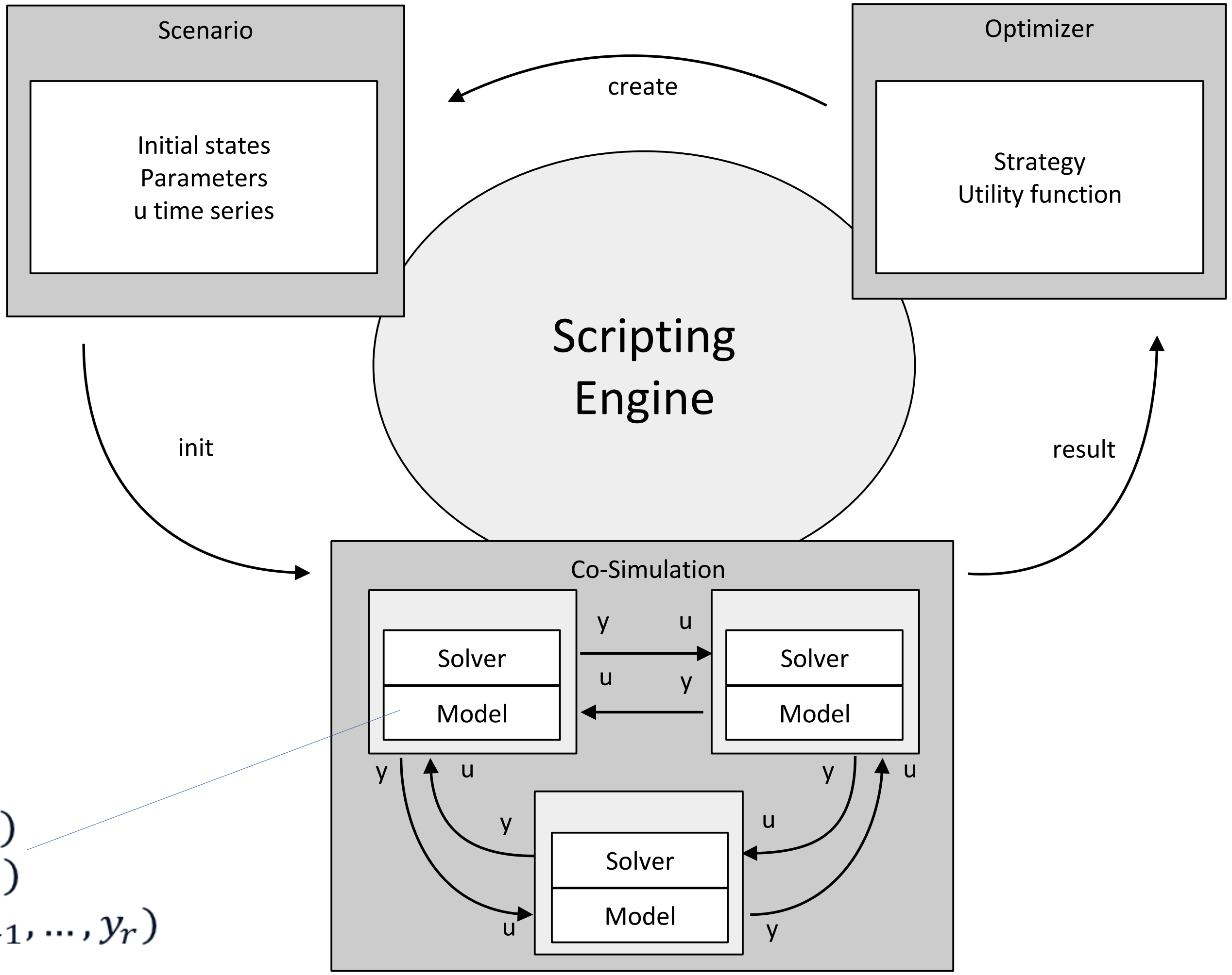
- Initialize simulators
- Exchange variables
- Sync time stepping
- User interaction



EU H2020 Programme GA No. 870620

Co(upled) Simulation Workflow

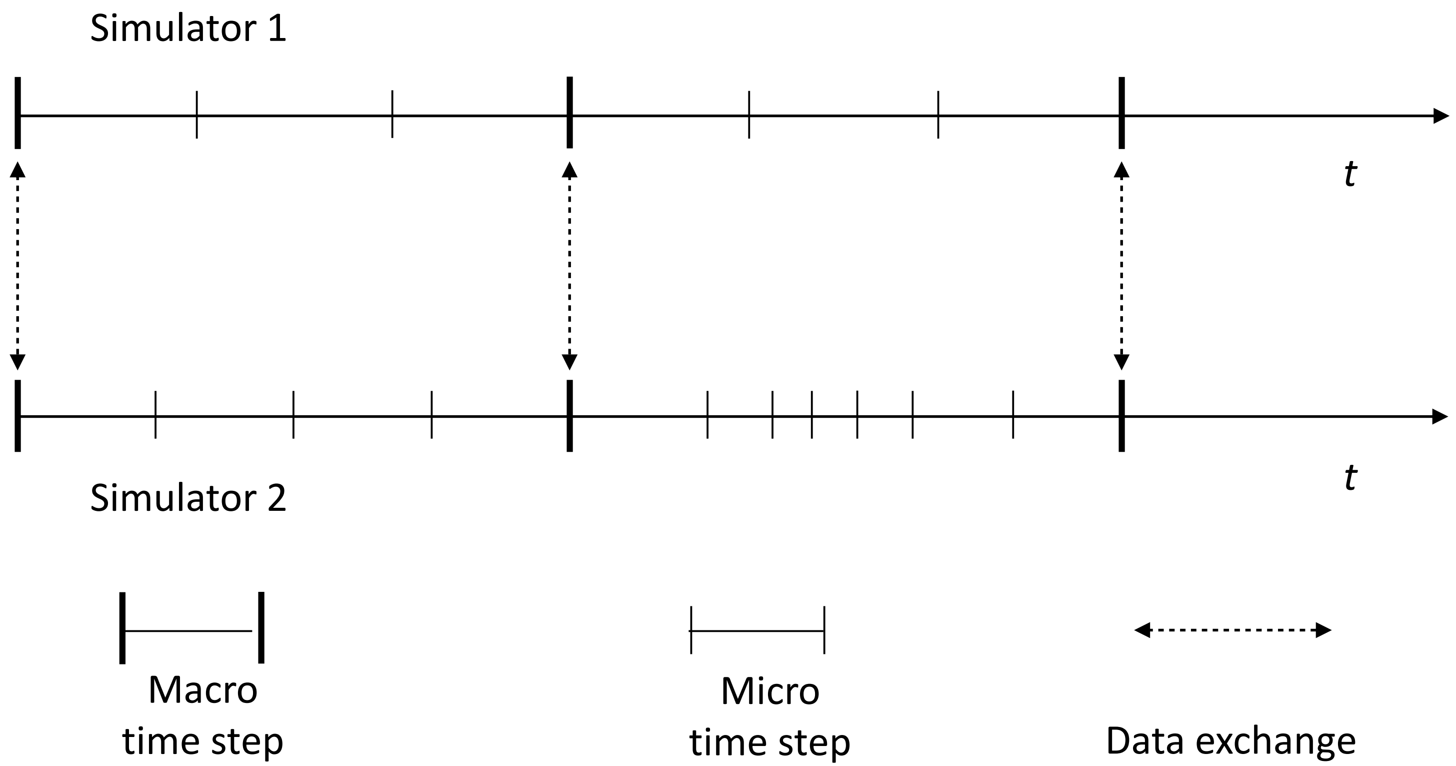
- Multiple simulators
- Multiple models
  - Coupled DAEs
- How to link/sync?
- Scenario Handling?
- Interfaces?



$$\dot{x}_i = f_i(t, x_i, u_i)$$
$$y_i = g_i(t, x_i, u_i)$$
$$u_i = c_i(y_1, \ldots, y_{i-1}, y_{i+1}, \ldots, y_r)$$
$$i = 1, \ldots, r$$

Synchronization

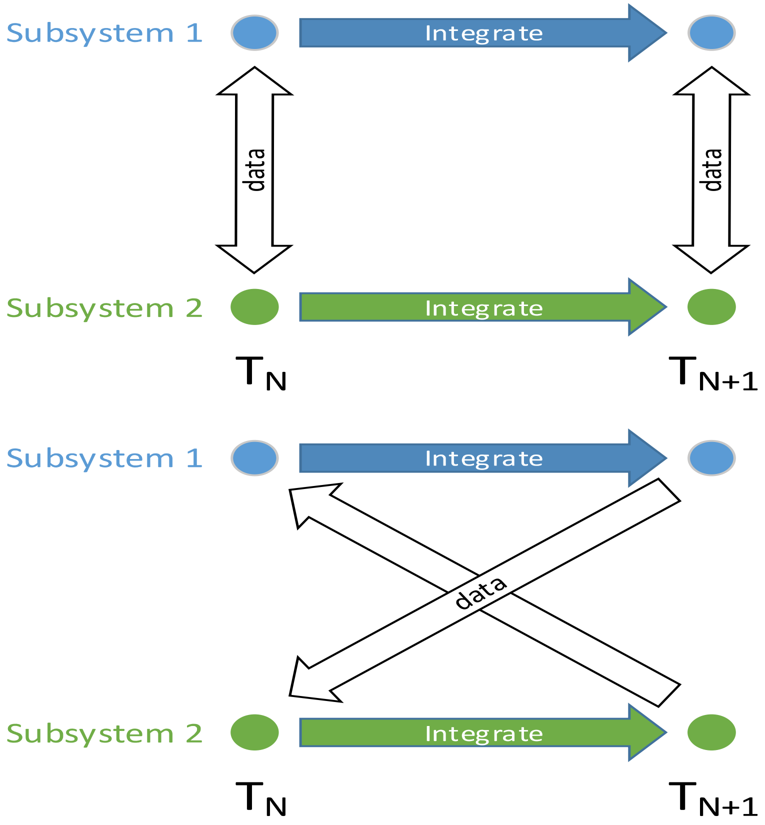
- Sync points = Macro steps
- Exchange variables



EU H2020 Programme GA No. 870620

Coupling principles

- Explicit coupling exchanges data at every external step once
- Implicit coupling iterates each external step until the system converges
- # external steps determines error



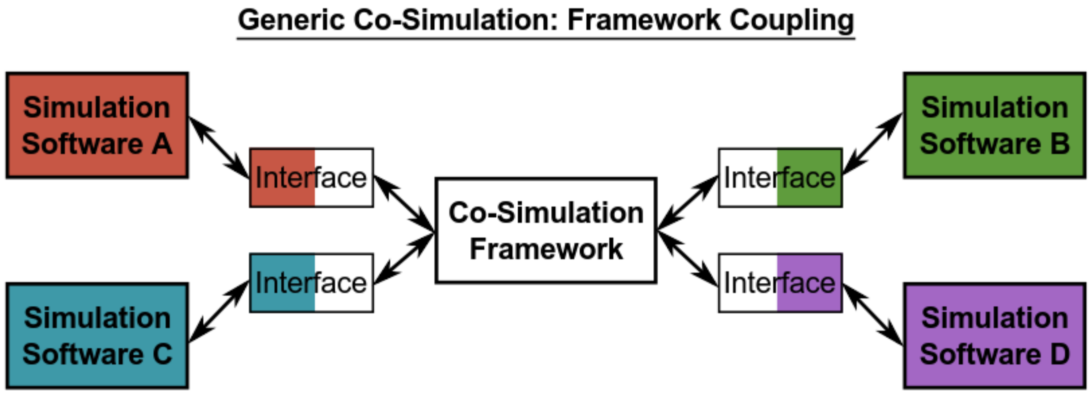
EU H2020 Programme GA No. 870620

Co-Simulation Bottom Line

- Coupled, coordinated simulators
- Multi-everything possible...
- RT co-sim vs. non-RT co-sim
- Scenario handling?
- Performance?

Part B: Co-simulation with mosaik

Co-simulation framework mosaik



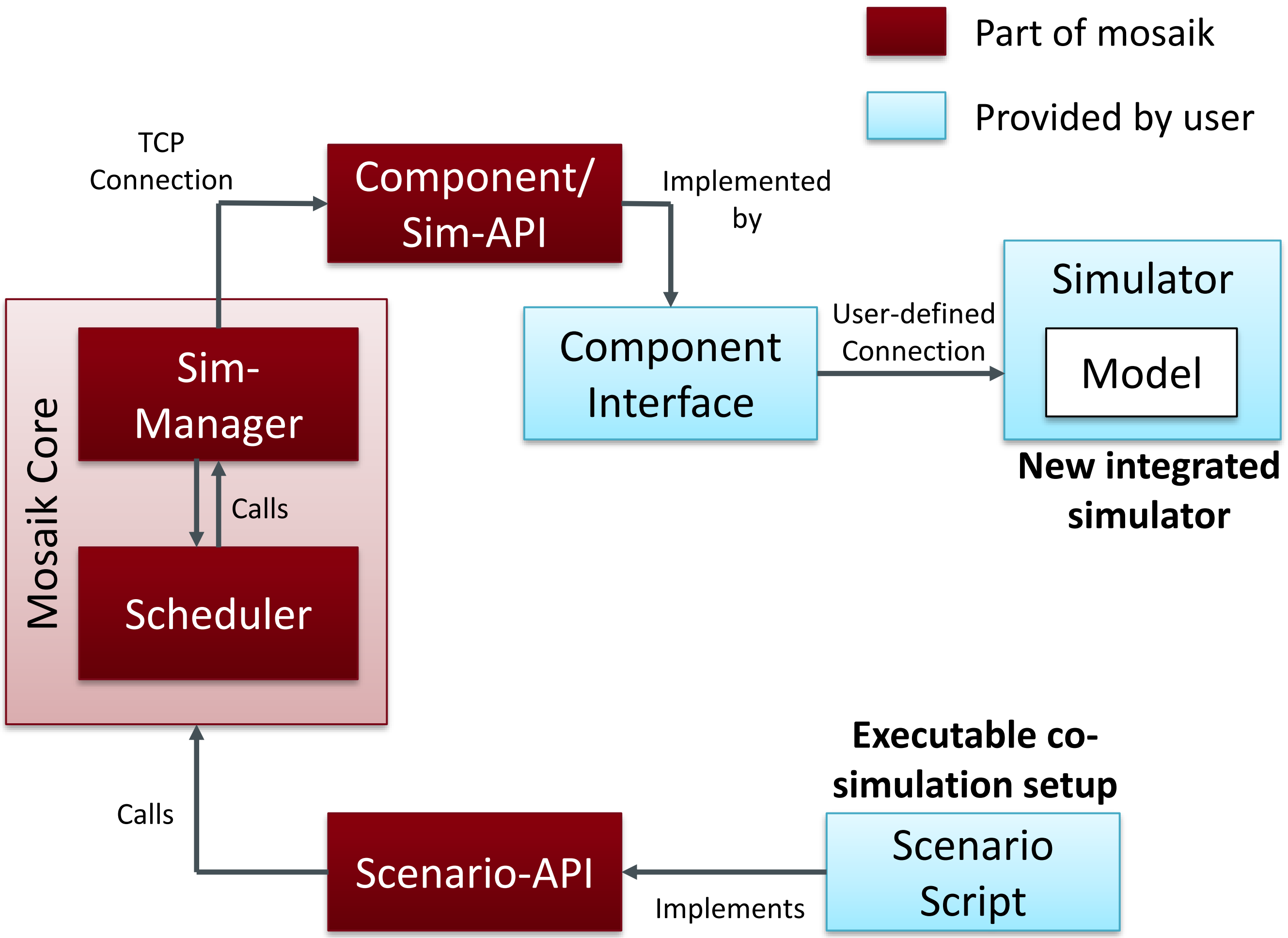
- Main features
  - Integration and re-use of heterogeneous simulation components
  - Specification of simulation scenarios
  - Coordination of data exchange and scheduling
  - Discrete time and discrete event simulation
- Open source (LGPL): gitlab.com/mosaik
- Documentation: mosaik.readthedocs.io

EU H2020 Programme GA No. 870620

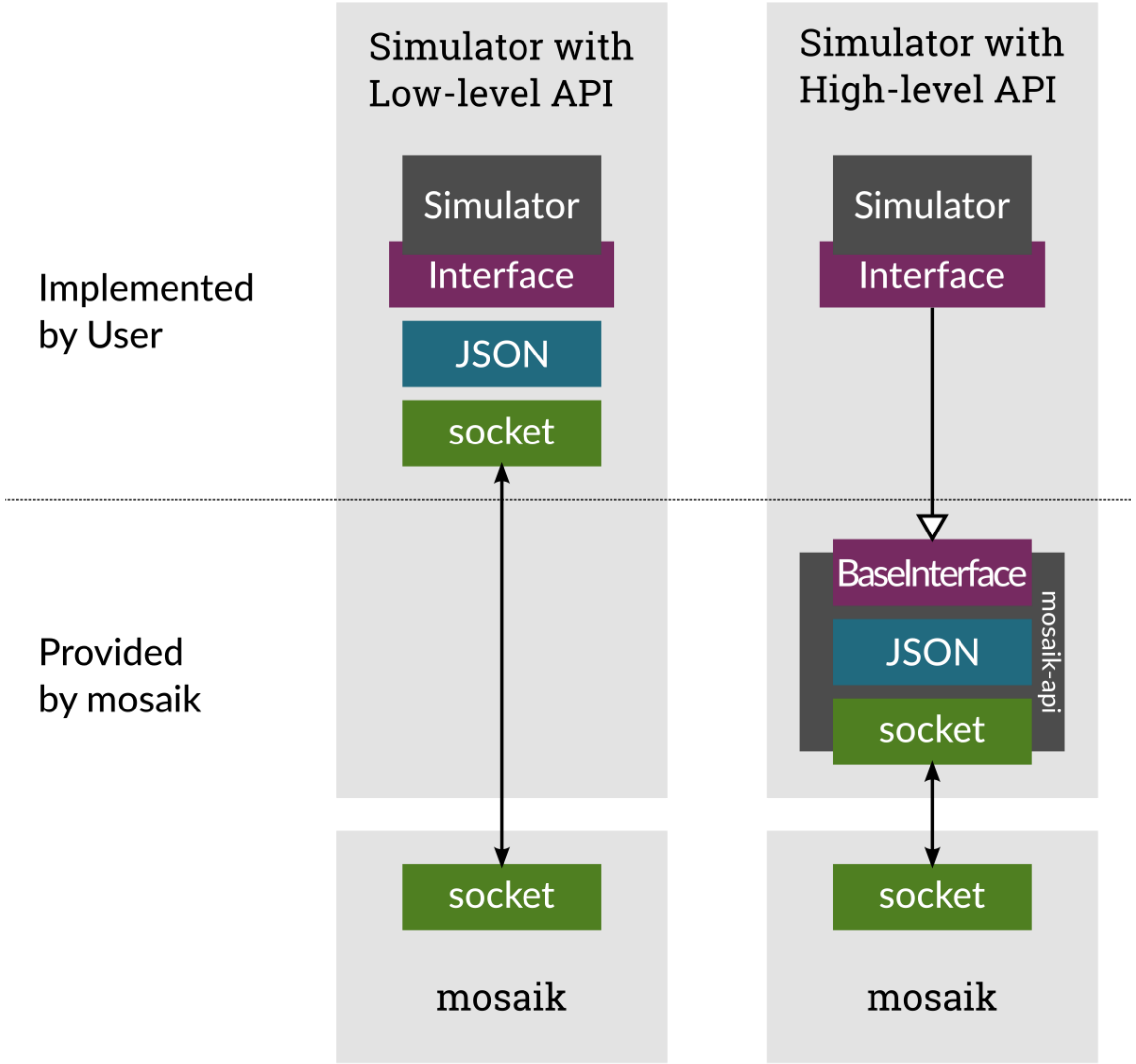
Mosaik ecosystem

- Simulation models
- Interfaces for simulation tools (e.g., pandapower, PowerFactory)
- Wrappers for programming languages (e.g., Java)
- Wrappers for standard interface (e.g., fmi or OPC UA)
- Visualization and data storage (e.g., HDF5, Influx+Grafana)

Mosaik architecture



Component API of mosaik



Implemented
by User

Provided
by mosaik

Simulator with
Low-level API

Simulator
Interface
JSON
socket

socket
mosaik

Simulator with
High-level API

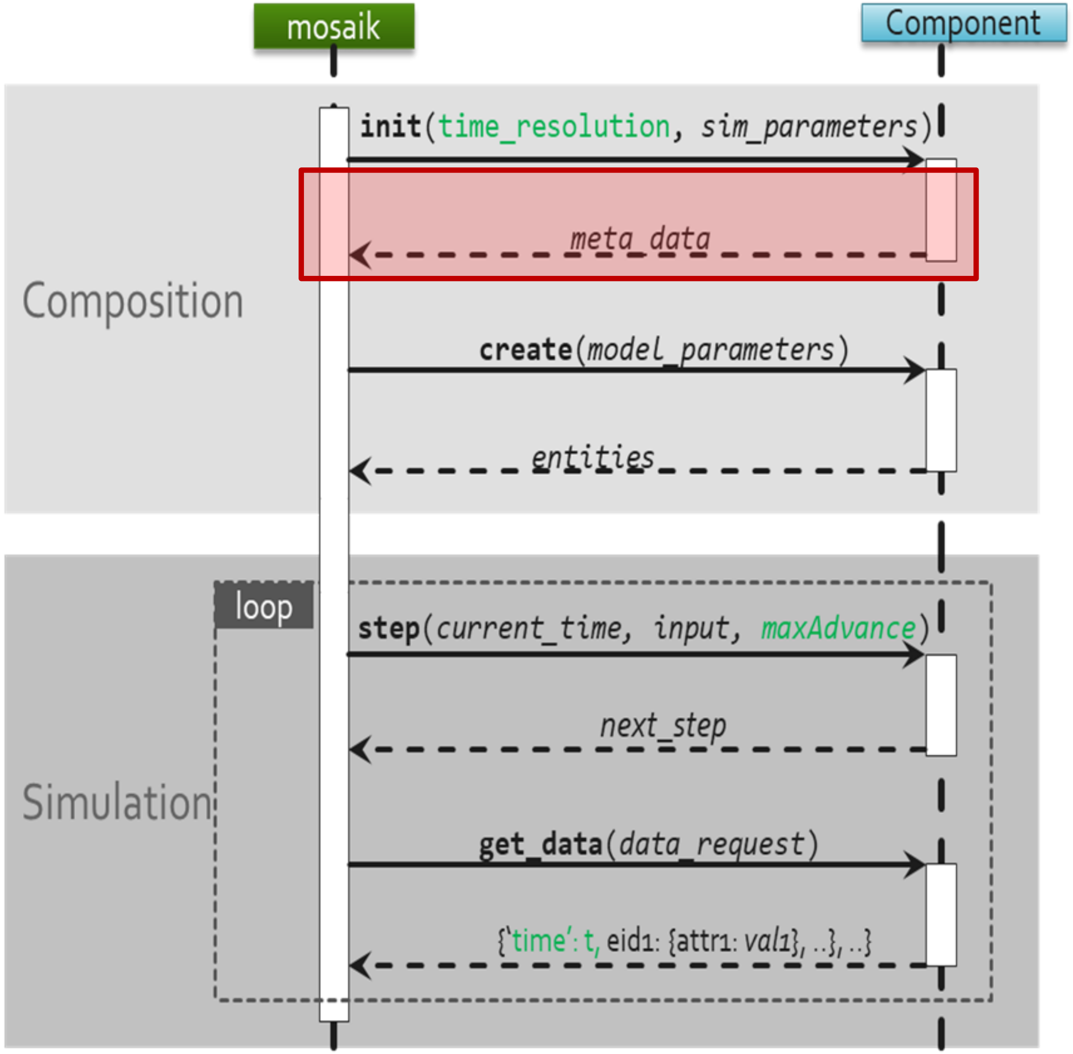
Simulator
Interface

BaseInterface
JSON
socket
mosaik-api

socket
mosaik

- **Low-level API:** Every tool supporting TCP-sockets and JSON
- **High-level API:** Several easier solutions for specific tools
  - E.g., create a subclass of mosaik_api.Simulator

Component API of mosaik



Meta data returned by init()

```
{
    'api_version': 'x.y‘,
    'type': 'time-based'|'event-based'|'hybrid',
    'models': {
        'ModelName': {
            'params': ['param_1', ...],
            'attrs': ['attr_1', ...],
            'trigger': ['attr_1', ...],
            'non-persistent': ['attr_2', ...],
        },
    ...
    },
}
```

Simulator’s type:

- **Time-based:** Traditional mosaik 2 simulator with self-stepping and persistent data
- **Event-based:** Triggered by all attributes and non-persistent data
- **Hybrid:** Attribute lists within ‘trigger’ and ‘non-persistent’ specify the behavior

Triggering attributes (optional):

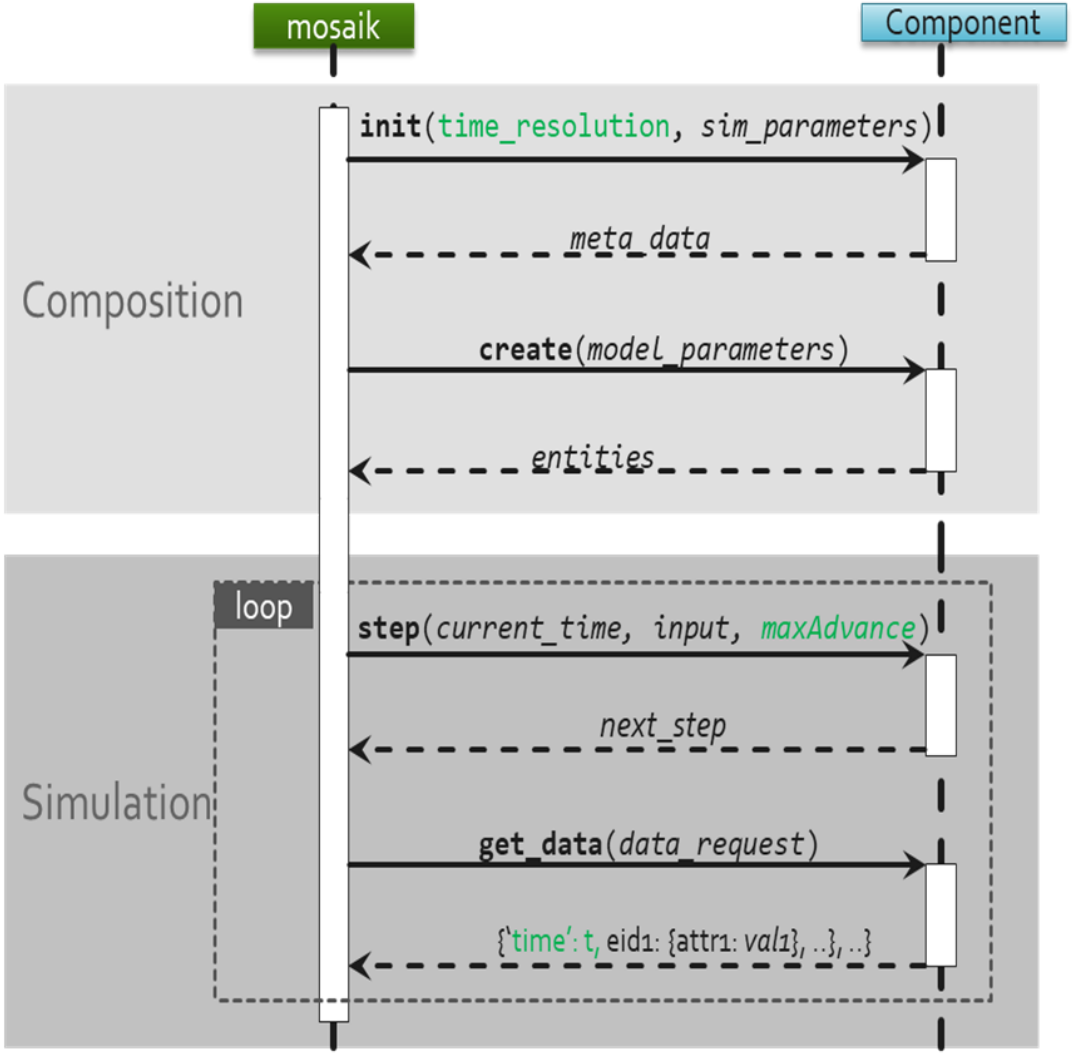
- simulator will be stepped automatically as soon as there’s new data available for this attribute

Non-persistent or transient attributes (optional):

- data of these attributes is only valid for a single time step

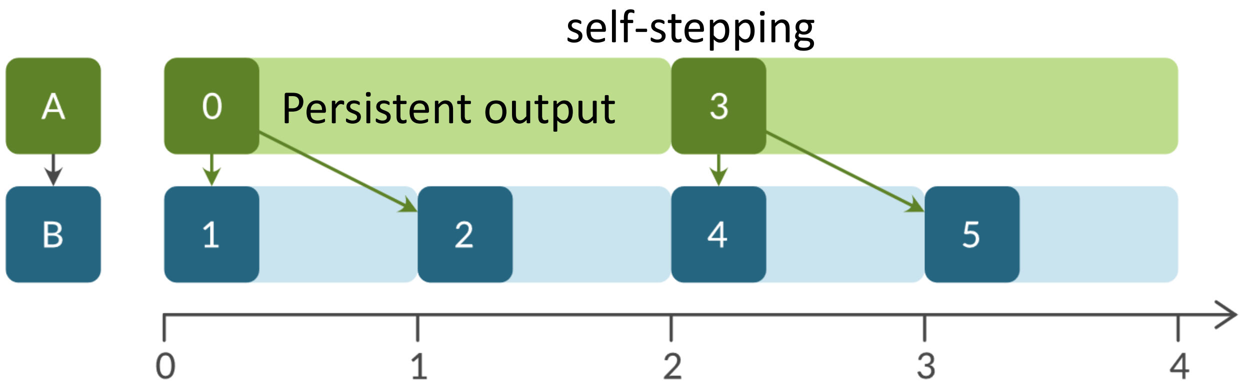
EU H2020 Programme GA No. 870620

Component API of mosaik

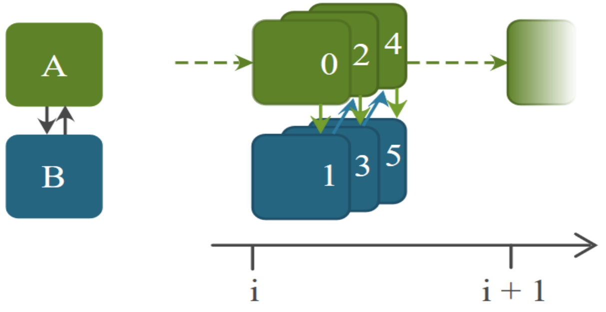


Scheduling/Synchronization

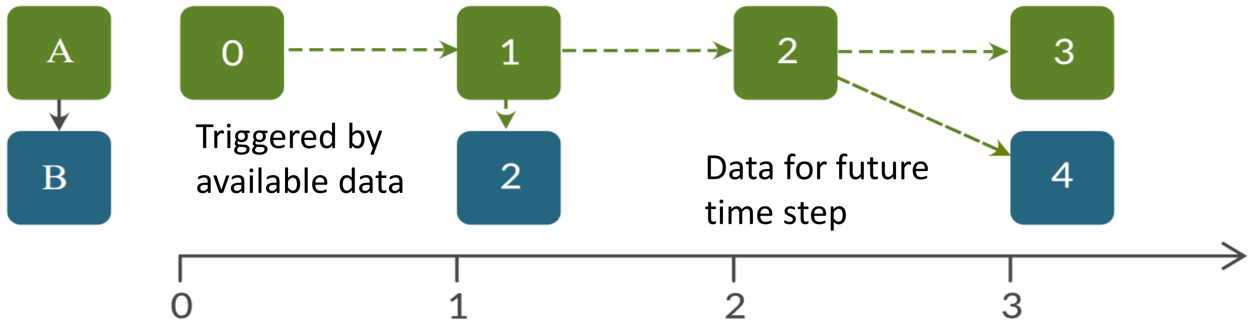
- Time-based



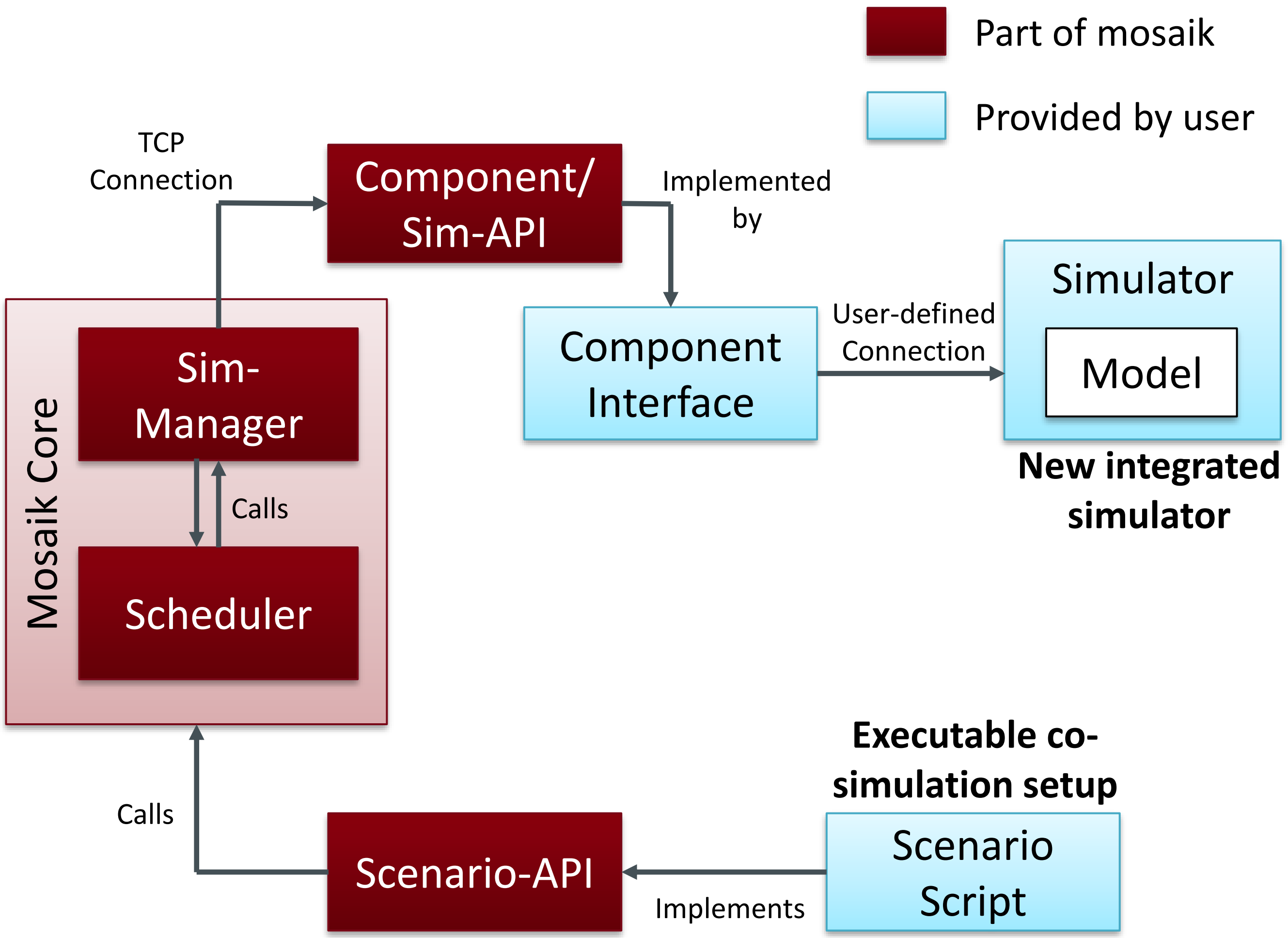
Superdense time / Same time loop:



- Event-based / hybrid



Mosaik architecture



🟥 Part of mosaik

🟦 Provided by user

TCP Connection

Component/ Sim-API

Implemented by

Mosaik Core

Sim-Manager

Calls

Scheduler

Component Interface

User-defined Connection

Simulator

Model

New integrated simulator

Calls

Scenario-API

Implements

Executable co-simulation setup

Scenario Script

Scenario API

- 1. Provide addresses of simulators to mosaik

```
import mosaik

sims = {grid: Python,
        house: Java,
        PV: MATLAB,
        control: Python}

world = mosaik.World(sims)
```

- Executable Python script
- Presented in pseudo code

Scenario API

- 2. Start simulator processes

```
gsim = world.start(grid)

hsim = world.start(house)

pvsim = world.start(PV)

csim = world.start(control)
```

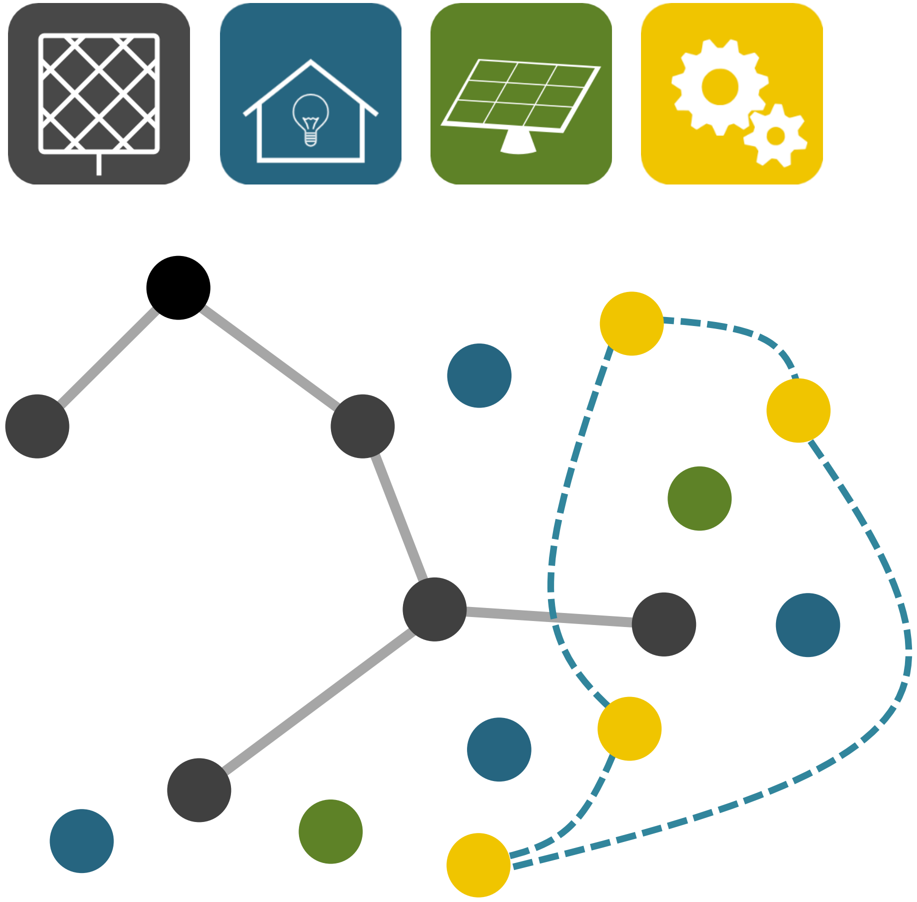
Simulator: Programm which controls models of a specific type or acts as an interface to external tools (e.g. pandapower, HDF5, …)

EU H2020 Programme GA No. 870620

Scenario API

- 3. Instantiate model entities & parameterize

```
grid =
gsim.create(gridmodel,
            param=topology)

houses =
hsim.create(housemodel, 4)

pvs =
pvsim.create(pvmodel, 2,
             param=size)

ctrl =
csim.create(MAScontrol)
```



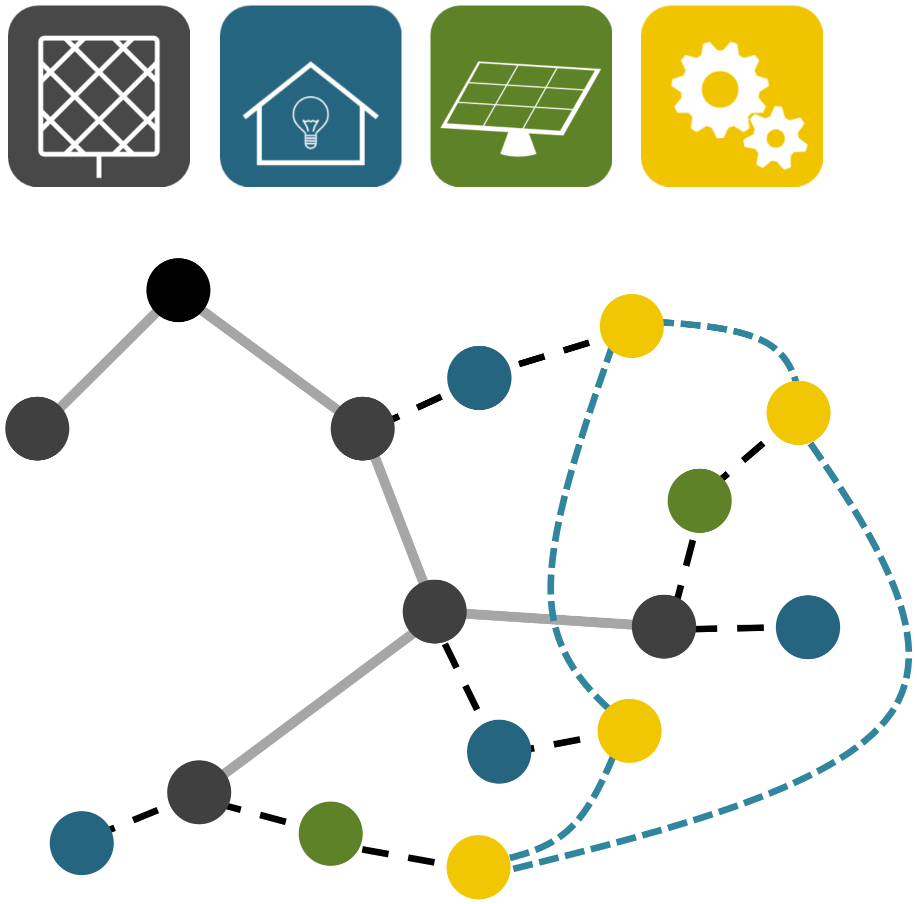
EU H2020 Programme GA No. 870620

Scenario API

- 4. Connect models via dataflow

```
connect(houses, grid,
         ‘active power’)

connect(pvs, grid,
         ‘active power’)

connect(ctrl[1&2],
        houses[1&2],
        ‘setpoint’)

connect(ctrl[3&4], pvs,
        ‘setpoint’)

world.run(3600)
```

Execute!



Demo

- Tutorials as JupyterLab:
  https://gitlab.com/mosaik/examples/mosaik-tutorials-on-binder
- Tutorials in Documentation:
  https://mosaik.readthedocs.io/en/latest/tutorials
- More Demos:
  https://gitlab.com/mosaik/examples

Summary

- Integration and re-use of heterogeneous simulation components
- Flexible specification of simulation scenarios
- Coordination of data exchange and scheduling
- Discrete time and discrete event simulation
- Open source (LGPL)
- Extensive ecosystem of models, interfaces and wrappers

Code:

https://gitlab.com/mosaik

Documentation:

https://mosaik.readthedocs.io/

Get in contact:

Mailing List: mosaik-users@lists.offis.de

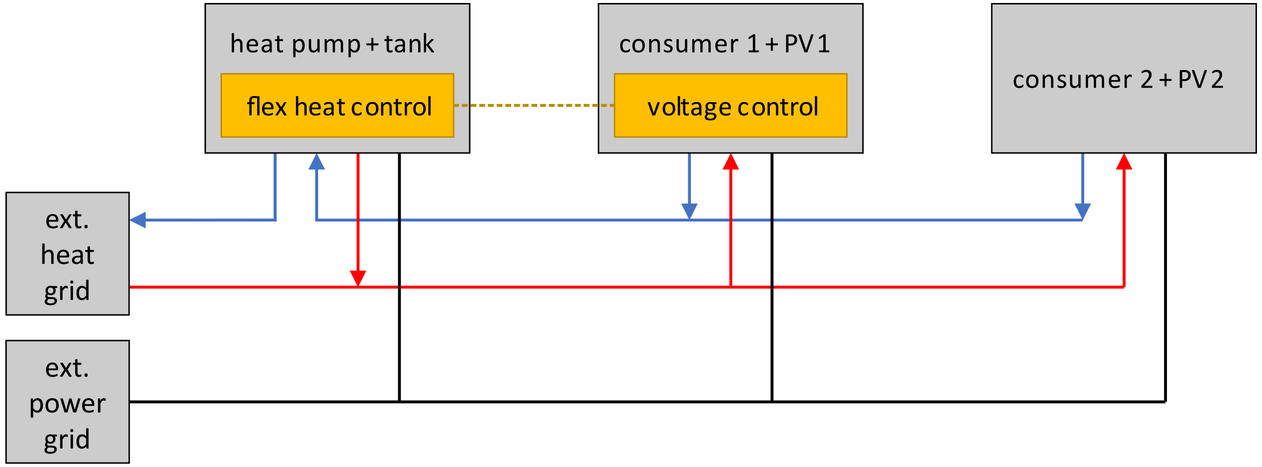
Direct mail: mosaik@offis.de

Open issues in GitLab

Part C: Example multi-energy network application

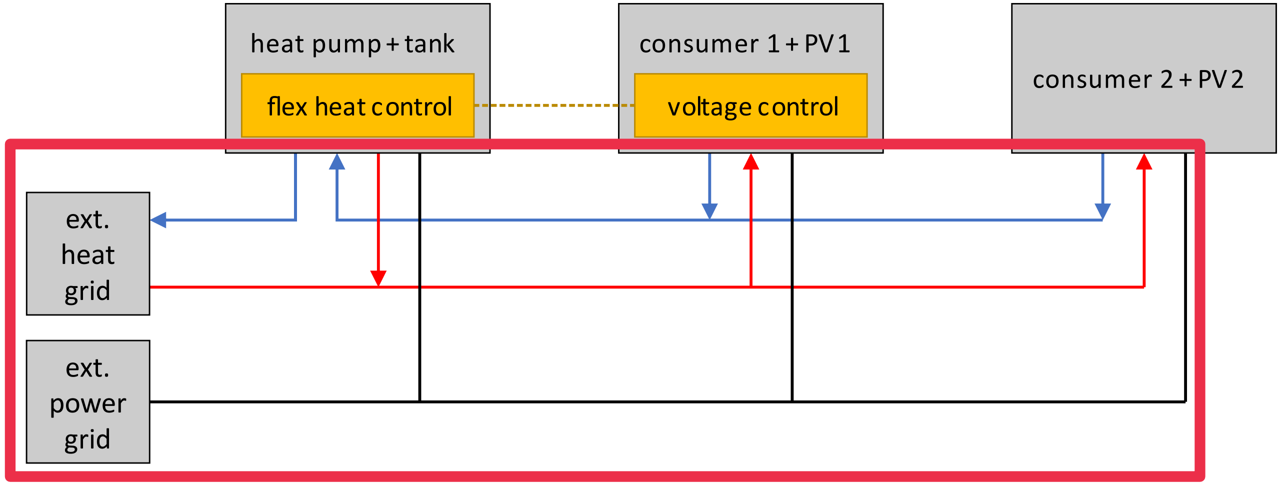
Example Multi-Energy Network Application (1/5)

- system configuration overview:
  - electrical network
  - thermal network
  - consumers
  - generation units
  - power-to-heat facility
- simple on purpose: focus on concept and use of co-simulation for multi-energy systems



Example Multi-Energy Network Application (2/5)

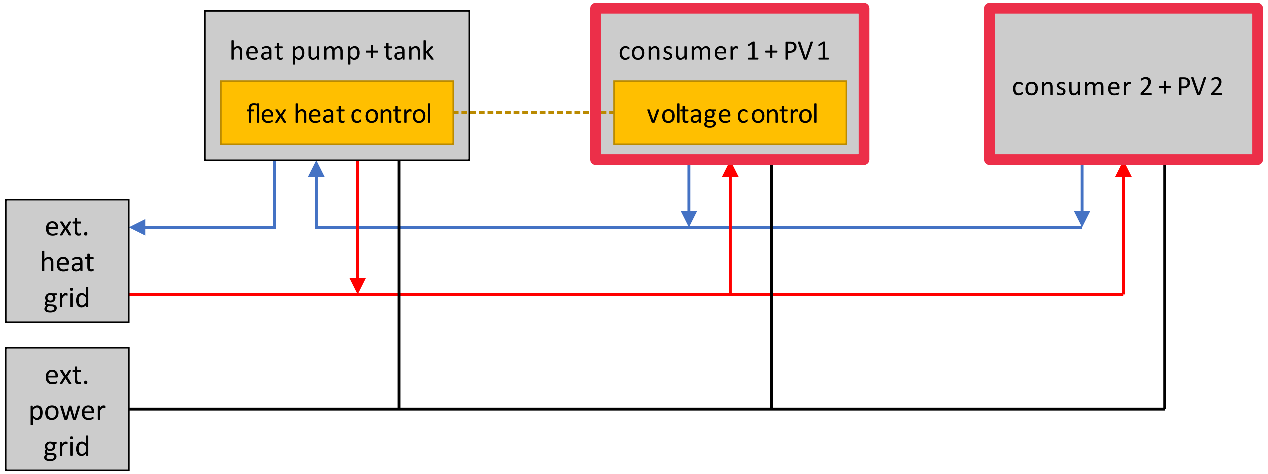
- electrical network:
  - 2 consecutive lines (0.3 km each)
  - connected to external grid
- thermal network:
  - 3 main consecutive pipes (supply and return, 0.5 km each)
  - connected to external grid



EU H2020 Programme GA No. 870620

Example Multi-Energy Network Application (3/5)

- consumers:
  - 2 consumers connected to both networks
  - aggregated loads (electrical and thermal) of an urban quarter
- generation units:
  - 2 PV systems with 150 $kW_{el, peak}$ and 50 $k_{Wel, peak}$



© The ERIGrid 2.0 Consortium
EU H2020 Programme GA No. 870620

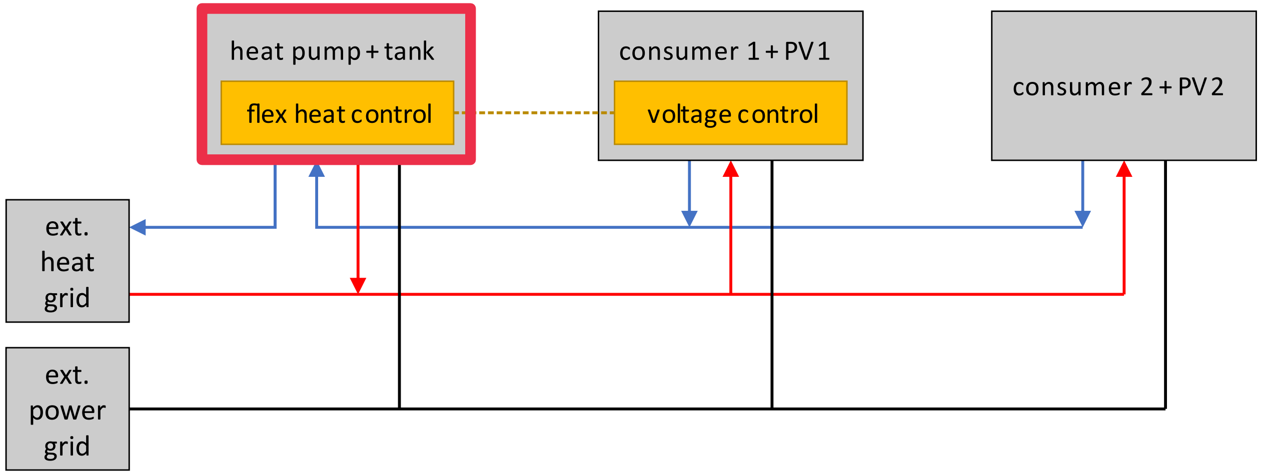
Example Multi-Energy Network Application (4/5)

- power-to-heat facility
  - heat pump and hot water storage tank
  - couples both networks
- flex heat controller operates the power-to-heat facility:

→ heat supply is covered entirely through the external grid

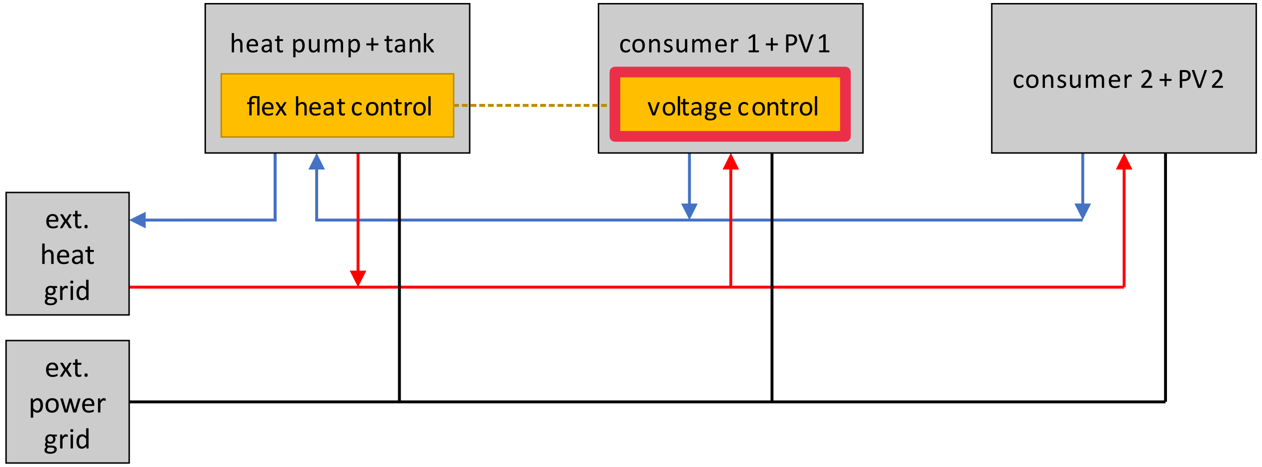
or

→ power-to-heat facility supports by discharging the tank



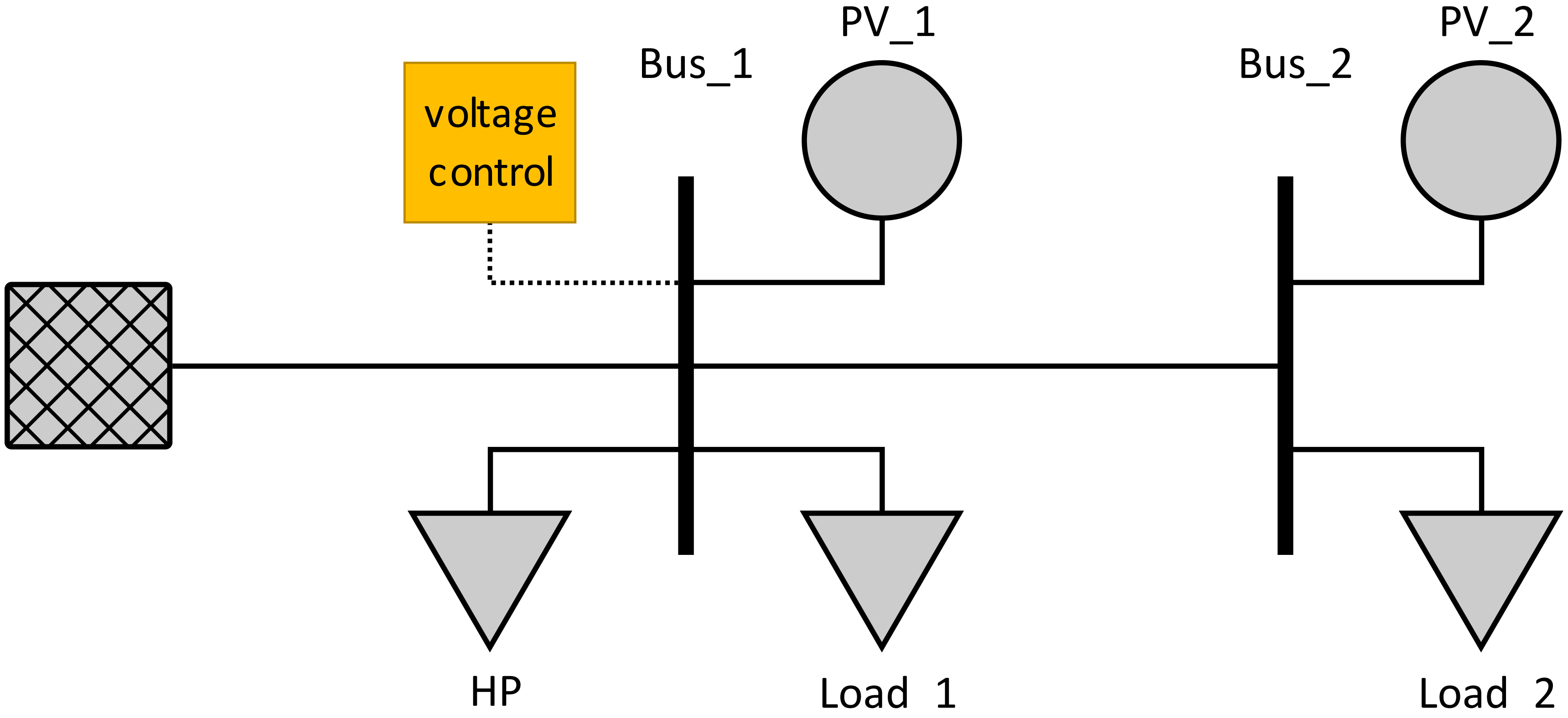
Example Multi-Energy Network Application (5/5)

- voltage controller uses power-to-heat facility as controllable load
  - voltage is monitored
  - the power consumption setpoint of the heat pump is adjusted
  - when activated, the heat pump is used to charge the tank with hot water

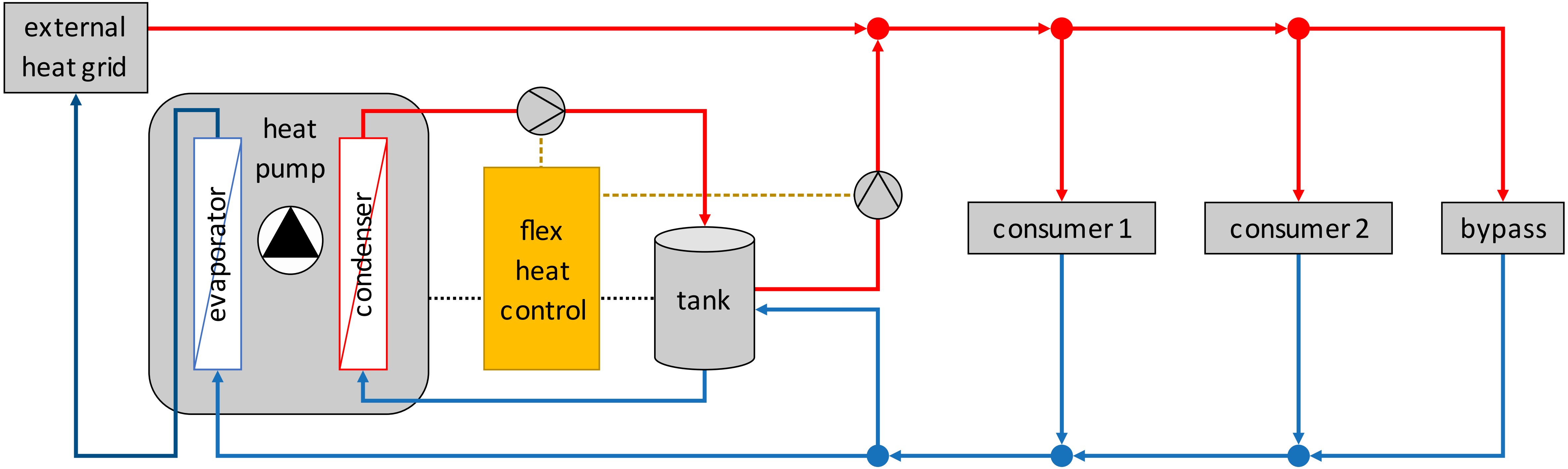


© The ERIGrid 2.0 Consortium
EU H2020 Programme GA No. 870620

Detailed View of Electrical Network

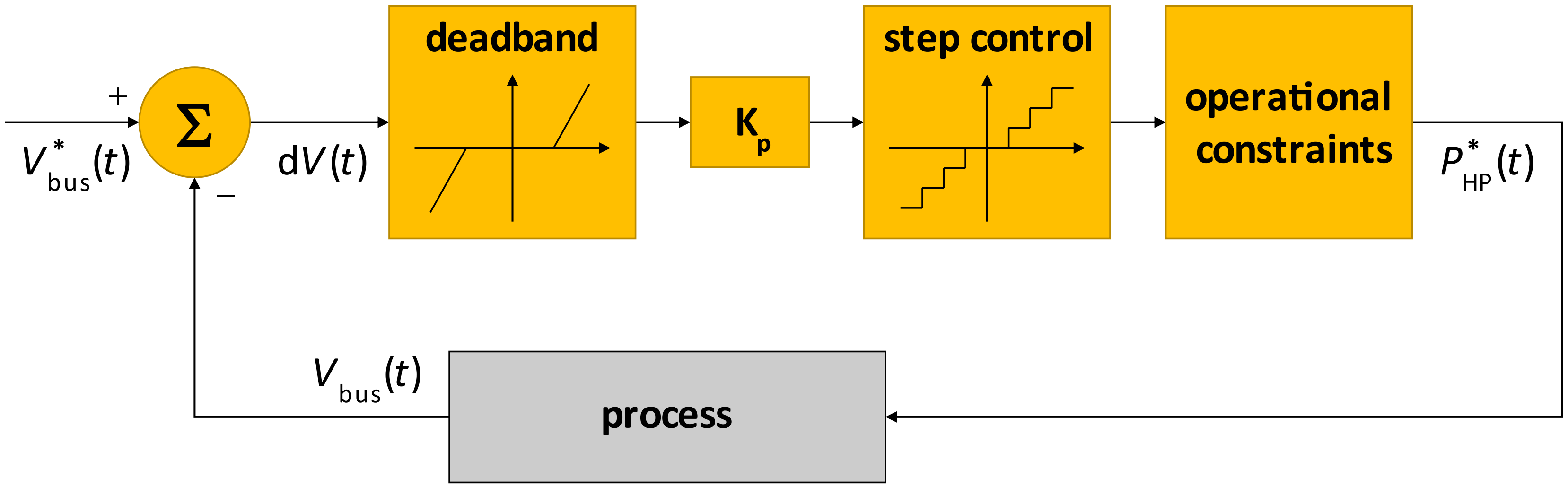


Detailed View of Thermal Network



Voltage Controller

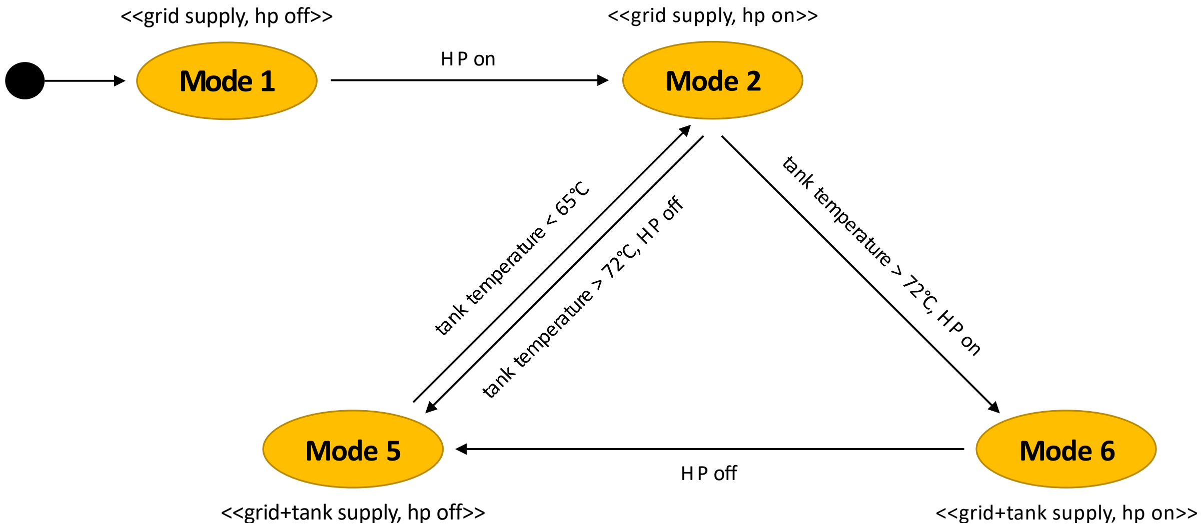
- voltage at *bus_1* is monitored
- power consumption setpoint of heat pump is adjusted to keep voltage within acceptable limits



EU H2020 Programme GA No. 870620

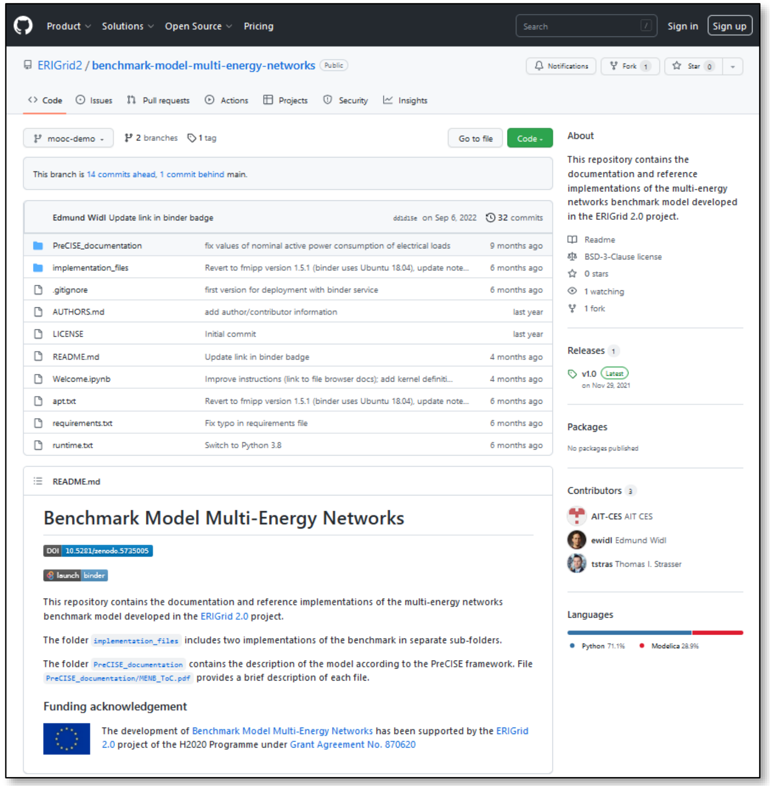
Flex Heat Controller

- regulate heat supply for thermal network
- operate power-to-heat facility to supply additional heat from the tank
- if required, heat pump is used to charge the tank, respecting the power consumption threshold from the voltage controller



Simulation Benchmark

- example application has been published as **simulation benchmark**:
  - detailed documentation
  - 2 reference implementations of the simulation setup
  - available at: https://github.com/ERIGrid2/benchmark-model-multi-energy-networks/tree/mooc-demo



This branch is 14 commits ahead, 1 commit behind main.

Edmund Widl Update link in binder badge dd1d15e on Sep 6, 2022 32 commits

PreCISE_documentation	fix values of nominal active power consumption of electrical loads	9 months ago
implementation_files	Revert to fmipp version 1.5.1 (binder uses Ubuntu 18.04), update note...	6 months ago
.gitignore	first version for deployment with binder service	6 months ago
AUTHORS.md	add author/contributor information	last year
LICENSE	Initial commit	last year
README.md	Update link in binder badge	4 months ago
Welcome.ipynb	Improve instructions (link to file browser docs); add kernel definiti...	4 months ago
apt.txt	Revert to fmipp version 1.5.1 (binder uses Ubuntu 18.04), update note...	6 months ago
requirements.txt	Fix typo in requirements file	6 months ago
runtime.txt	Switch to Python 3.8	6 months ago

README.md

Benchmark Model Multi-Energy Networks

DOI 10.5281/zenodo.5735005

launch binder

This repository contains the documentation and reference implementations of the multi-energy networks benchmark model developed in the ERIGrid 2.0 project.

The folder `implementation_files` includes two implementations of the benchmark in separate sub-folders.

The folder `PreCISE_documentation` contains the description of the model according to the PreCISE framework. File `PreCISE_documentation/MENB_ToC.pdf` provides a brief description of each file.

Funding acknowledgement

The development of Benchmark Model Multi-Energy Networks has been supported by the ERIGrid 2.0 project of the H2020 Programme under Grant Agreement No. 870620

About

This repository contains the documentation and reference implementations of the multi-energy networks benchmark model developed in the ERIGrid 2.0 project.

Readme
BSD-3-Clause license
0 stars
1 watching
1 fork

Releases 1
v1.0 Latest
on Nov 29, 2021

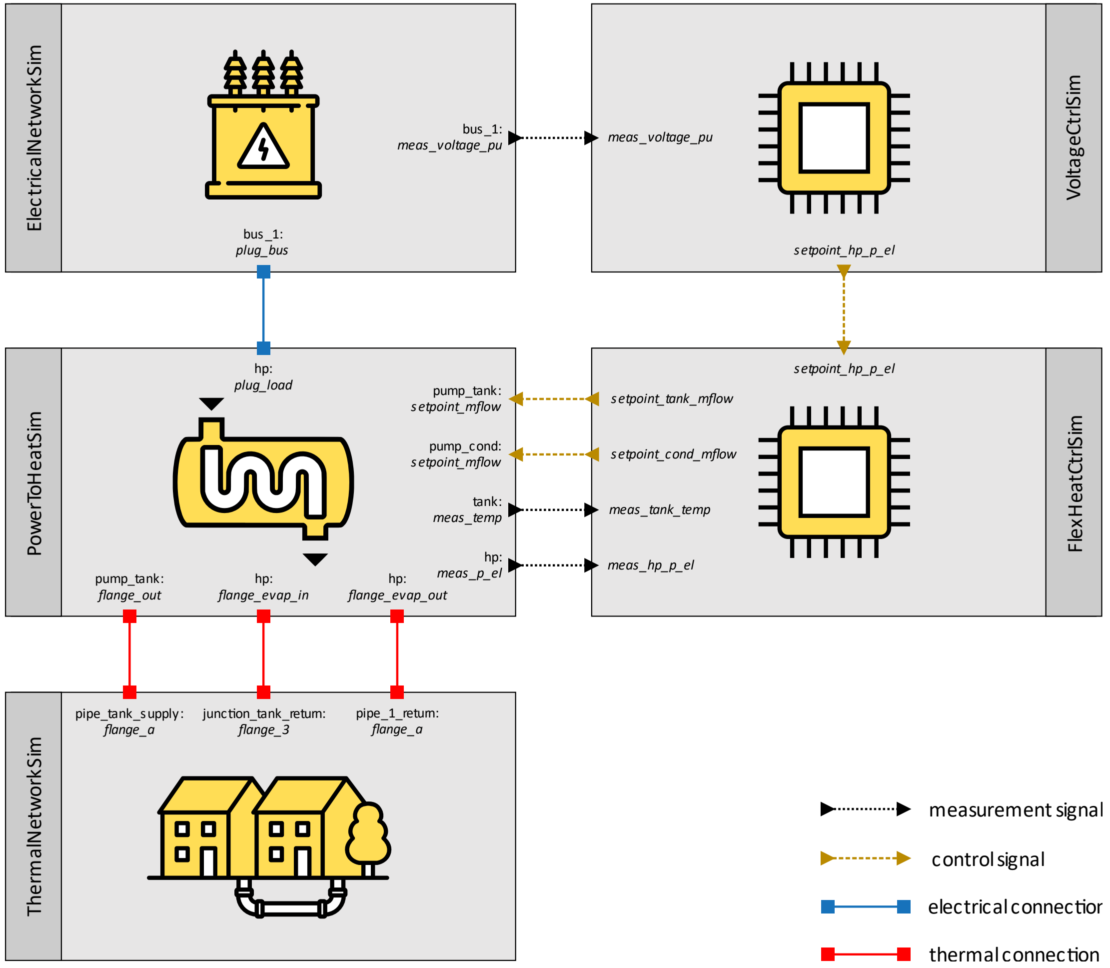
Packages
No packages published

Contributors 3
AIT-CES AIT CES
ewidl Edmund Widl
tstras Thomas I. Strasser

Languages
Python 71.1%
Modelica 28.9%

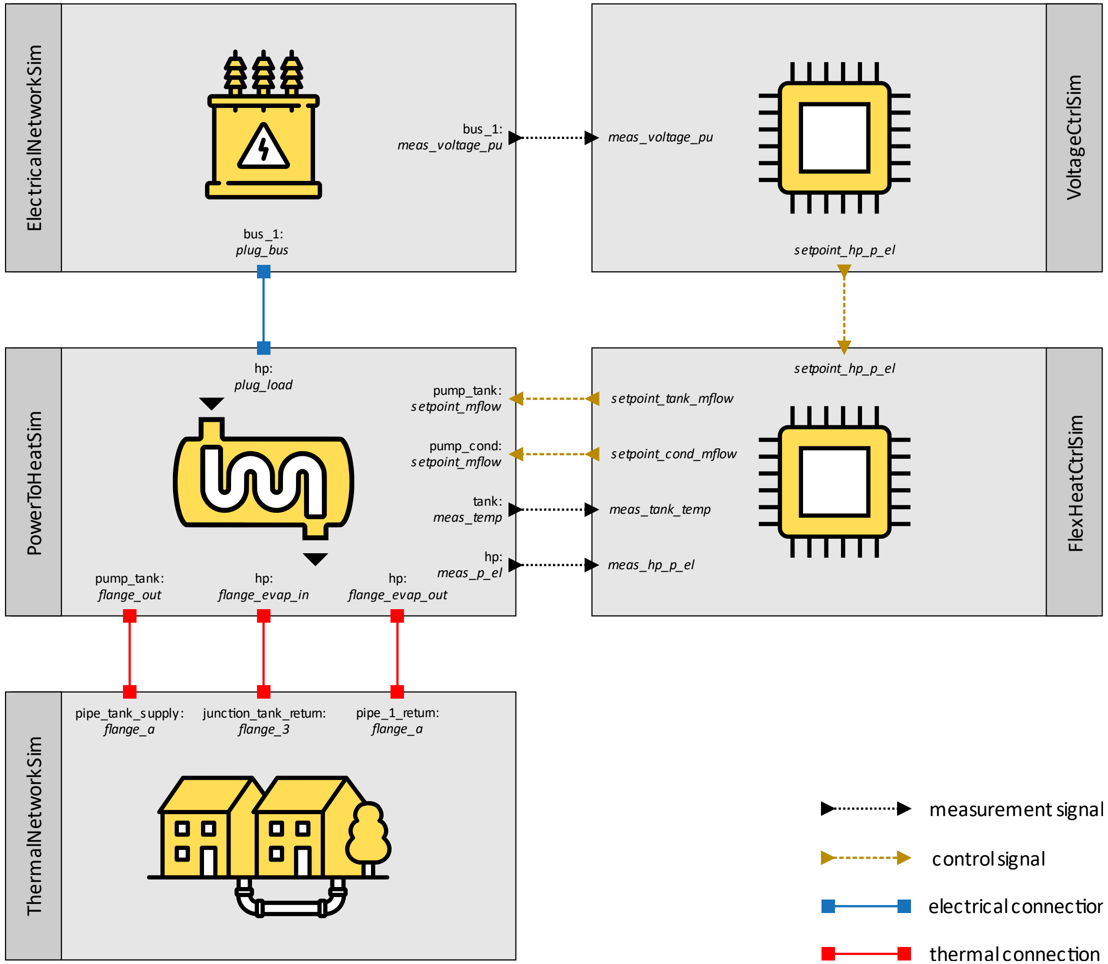
Simulated Subsystems

- use co-simulation for simulating multi-energy systems:
  - different domains (power, heat, control) are covered by domain-specific simulation tools
  - partitioning of system under test into subsystems, each implemented by dedicated simulators



Simulated Subsystems

- use the mosaik co-simulation framework:
  - pandapower for the electrical subsystem
  - standalone implementations of the controllers’ logic in Python
  - pandapipes / DisHeatLib for the thermal domain



EU H2020 Programme GA No. 870620

Thermal Network Simulators

- simulation benchmarks provides 2 reference implementations using different approaches for modelling thermal networks:
  - **Python package pandapipes**
    - developed at the Fraunhofer Institute for Energy Economics and Energy System Technology (IEE) and the University of Kassel
  - **Modelica library DisHeatLib**
    - developed at the AIT Austrian Institute of Technology

EU H2020 Programme GA No. 870620

Comparison of Modeling Approaches (1/2)

pandapipes

- implements *pipe flow* calculations (compare to load flow for electrical grids)
  - static or quasi-static analysis of balanced fluid systems
  - computation of temperature, pressure and velocity distributions in pipe networks

DisHeatLib

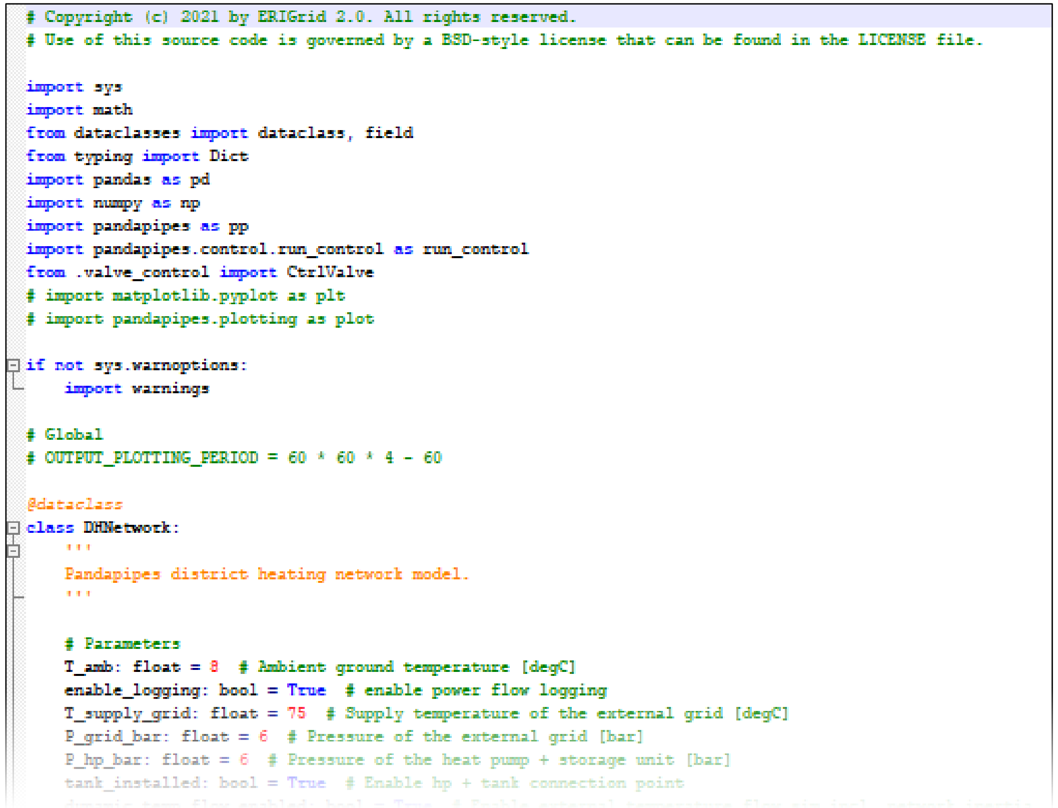
- implements *plug flow* calculations (with non-linear pressure drop relation)
  - analysis of thermo-hydraulic transients in fluid systems
  - flow reversals & time-delayed propagation of fluid properties in the pipe system

EU H2020 Programme GA No. 870620

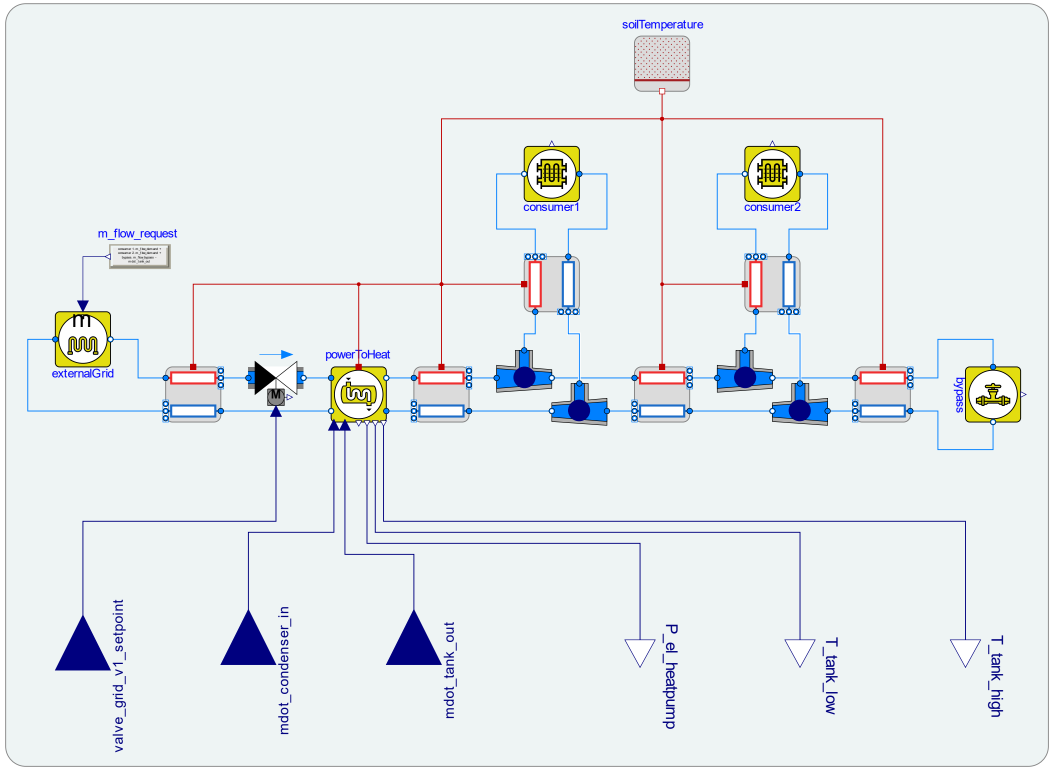
Comparison of Modeling Approaches (2/2)

pandapipes

DisHeatLib



```
# Copyright (c) 2021 by ERIGrid 2.0. All rights reserved.
# Use of this source code is governed by a BSD-style license that can be found in the LICENSE file.

import sys
import math
from dataclasses import dataclass, field
from typing import Dict
import pandas as pd
import numpy as np
import pandapipes as pp
import pandapipes.control.run_control as run_control
from .valve_control import CtrlValve
# import matplotlib.pyplot as plt
# import pandapipes.plotting as plot

if not sys.warnoptions:
    import warnings

# Global
# OUTPUT_PLOTTING_PERIOD = 60 * 60 * 4 - 60

@dataclass
class DHNetwork:
    '''
    Pandapipes district heating network model.
    '''

    # Parameters
    T_amb: float = 8  # Ambient ground temperature [degC]
    enable_logging: bool = True  # enable power flow logging
    T_supply_grid: float = 75  # Supply temperature of the external grid [degC]
    P_grid_bar: float = 6  # Pressure of the external grid [bar]
    P_hp_bar: float = 6  # Pressure of the heat pump + storage unit [bar]
    tank_installed: bool = True  # Enable hp + tank connection point
```



EU H2020 Programme GA No. 870620

Comparison of Tool Coupling Capabilities

pandapipes

- so-called “controllers” implement the coupling points
  - read values from one network
  - apply efficiency factors and unit conversions
  - write the results to another network
- for more advanced use cases, pandapipes’ API can be used for tool coupling in Python

DisHeatLib

- models can be exported as standalone executable models according to the Functional Mock-up Interface (FMI) specification
- enables a straightforward coupling with any other FMI-compliant simulation tool

EU H2020 Programme GA No. 870620

Comparison of Availability

pandapipes

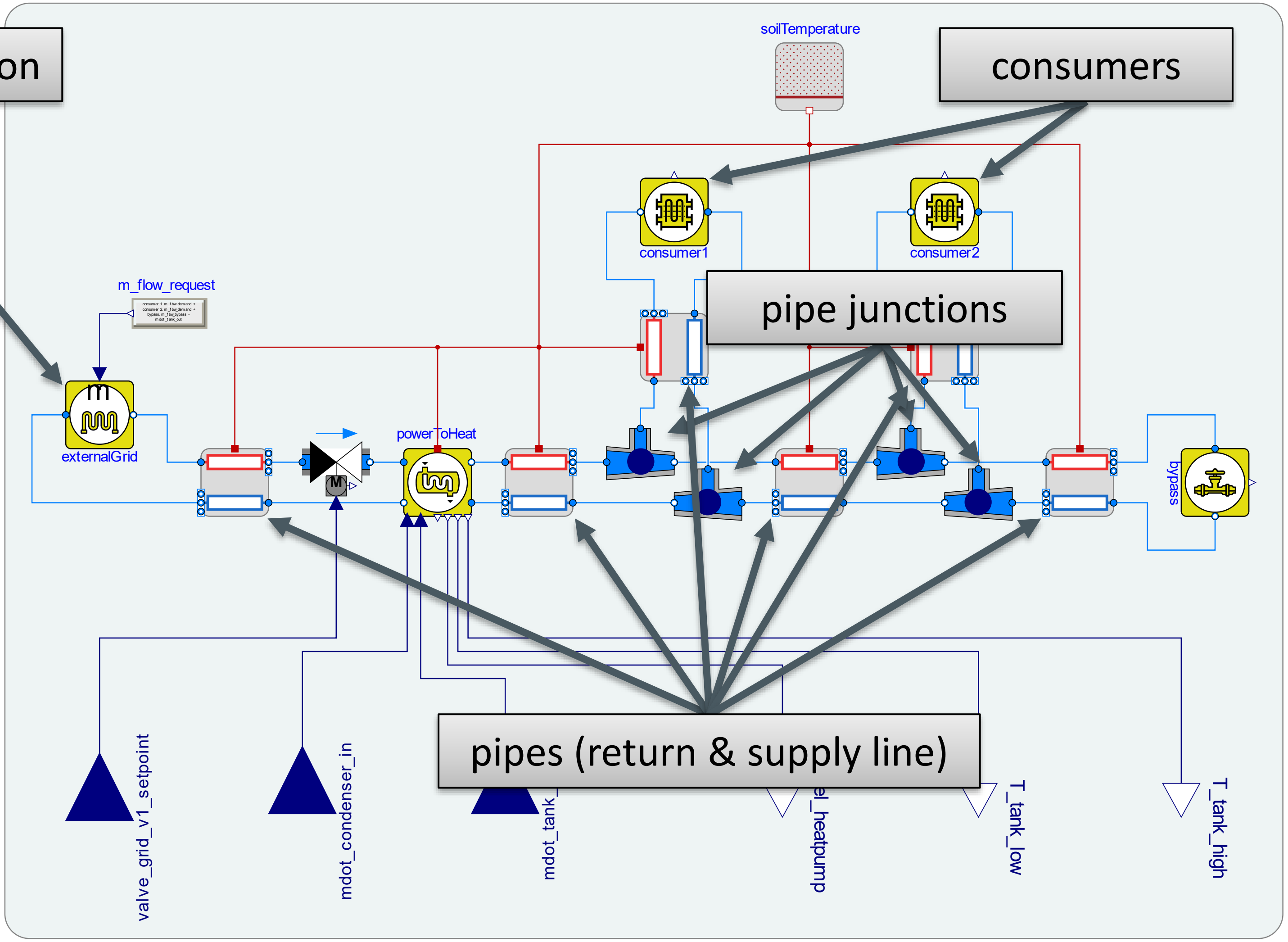
- publicly available open-source Python package
- since Python is also a publicly available open-source tool, pandapipes can be used without a commercial license

DisHeatLib

- publicly available open-source Modelica library
- for compiling models created from the DisHeatLib to an executable, the proprietary Dymola tool is required
- in the future, a complete open-source tool support through OpenModelica is expected

EU H2020 Programme GA No. 870620

DisHeatLib: Thermal Network Model



external grid connection

consumers

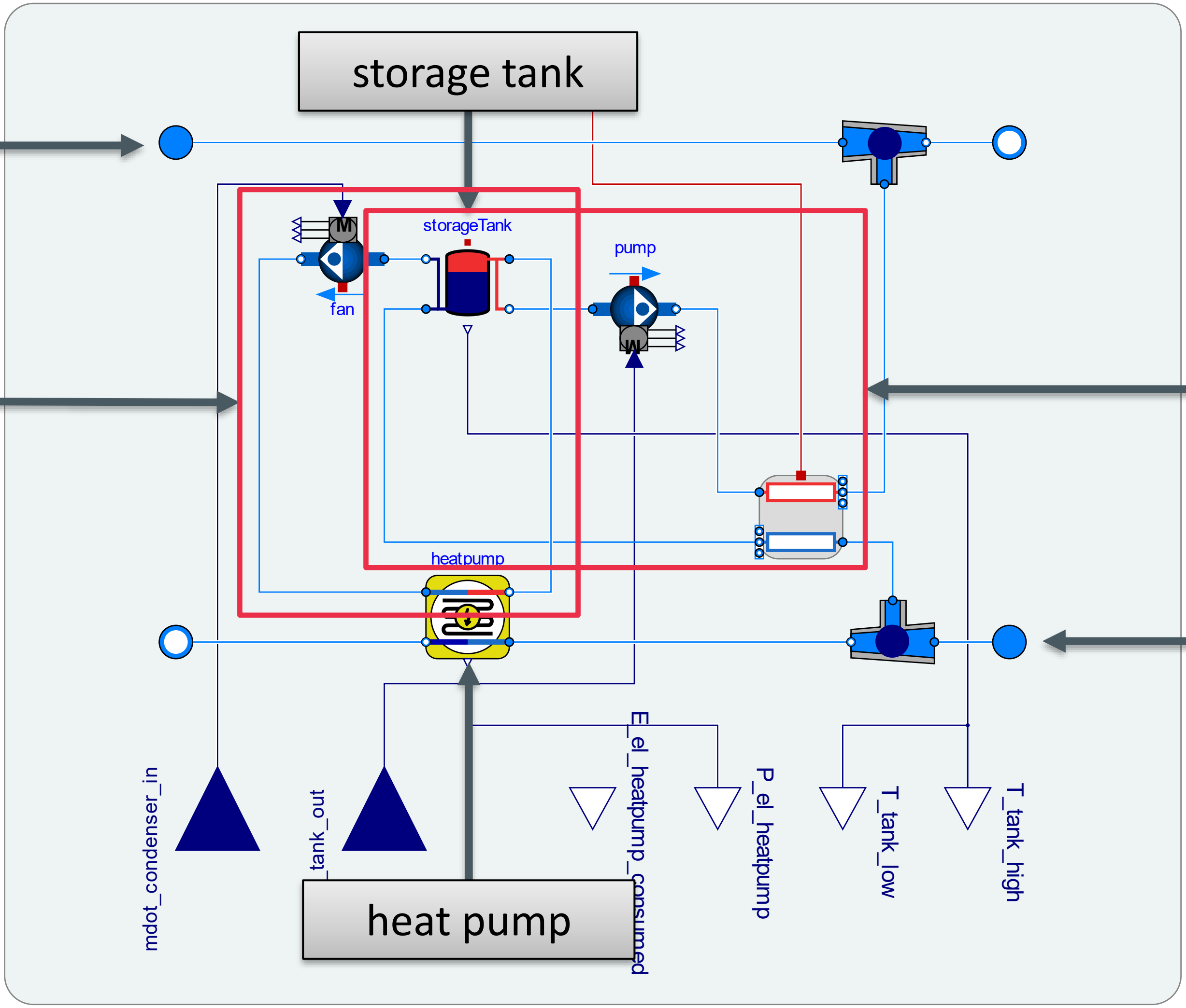
pipe junctions

pipes (return & supply line)

DisHeatLib: Power-to-Heat Facility Model

supply line

hydraulic loop:
- heat pump condenser
- pump
- storage tank



storage tank

hydraulic loop:
- storage tank
- pump
- supply line (feed hot water)
- return line (draw cold water)

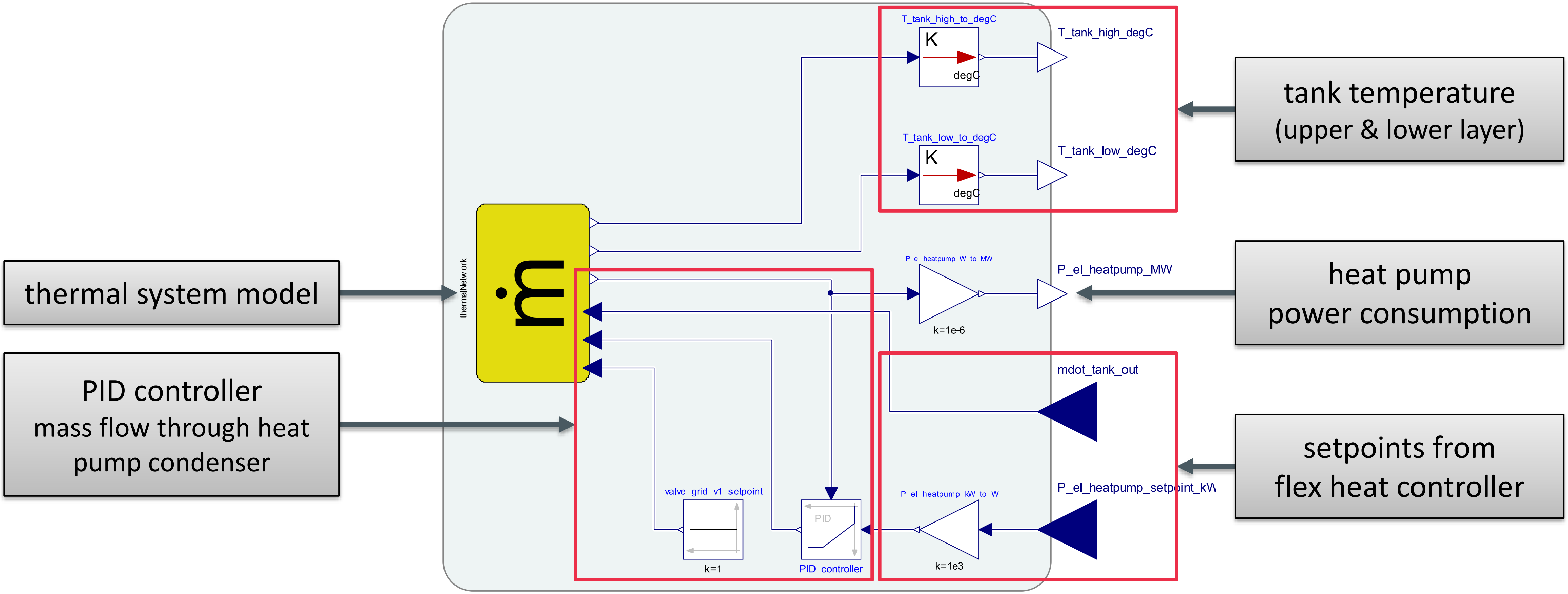
return line

heat pump

EU H2020 Programme GA No. 870620

DisHeatLib: Thermal System Model

- thermal system model is exported as standalone executable binary (Functional Mock-up Unit, FMU)



thermal system model

PID controller
mass flow through heat pump condenser

tank temperature
(upper & lower layer)

heat pump
power consumption

setpoints from
flex heat controller

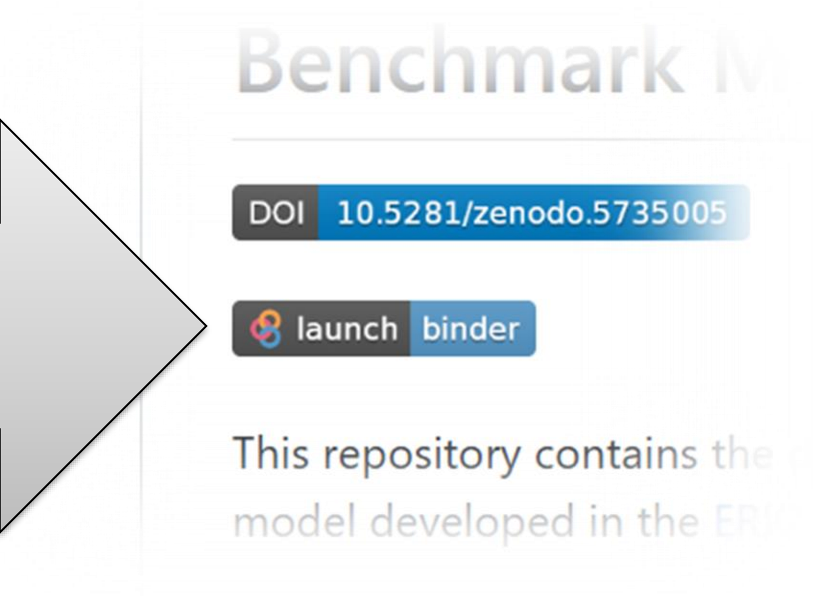
EU H2020 Programme GA No. 870620

Hands-on Demo

available online via Binder:

https://mybinder.org/v2/gh/ERIGrid2/benchmark-model-multi-energy-networks/mooc-demo?labpath=Welcome.ipynb

… or click
“launch binder”
on Github page

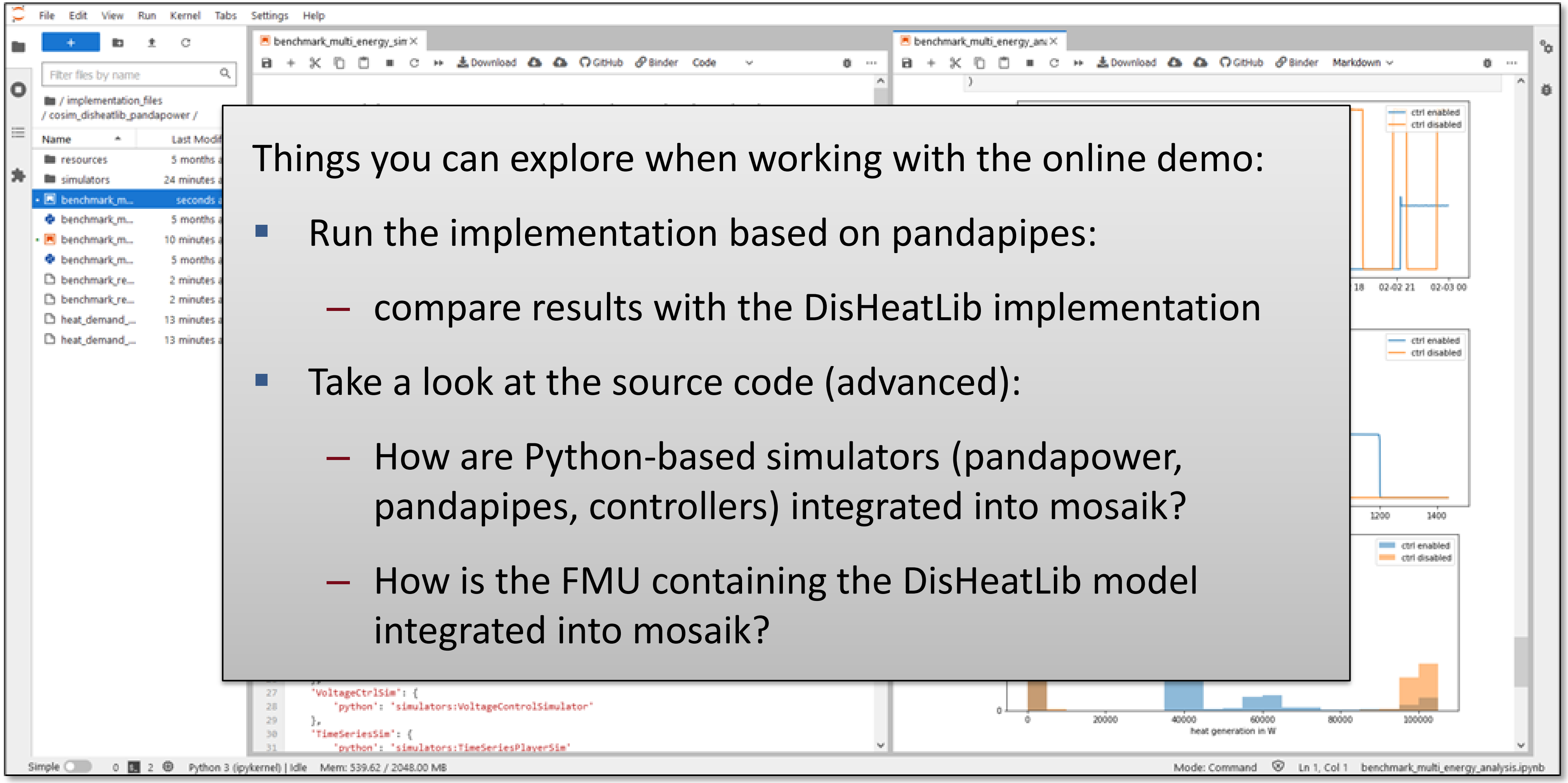


Benchmark

This repository contains the
model developed in the

EU H2020 Programme GA No. 870620

Hands-on Demo



```
    },
    'VoltageCtrlSim': {
        'python': 'simulators:VoltageControlSimulator'
    },
    'TimeSeriesSim': {
        'python': 'simulators:TimeSeriesPlayerSim'
```

Things you can explore when working with the online demo:

- Run the implementation based on pandapipes:
  - compare results with the DisHeatLib implementation
- Take a look at the source code (advanced):
  - How are Python-based simulators (pandapower, pandapipes, controllers) integrated into mosaik?
  - How is the FMU containing the DisHeatLib model integrated into mosaik?

EU H2020 Programme GA No. 870620

Conclusion

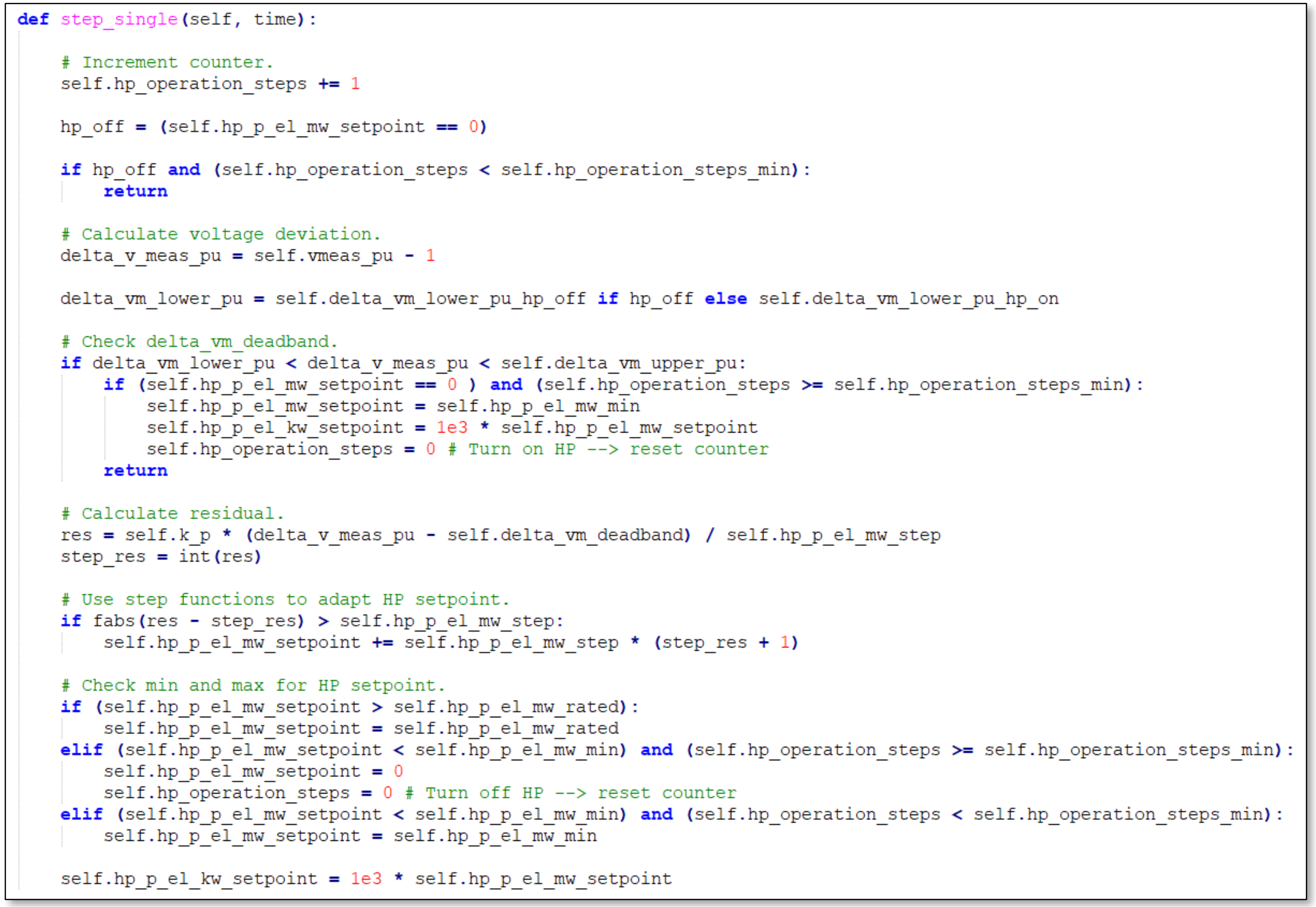
- co-simulation is a powerful approach for the assessment of multi-domain energy systems
- the mosaik framework provides a versatile environment for co-simulation
- ERIGrid 2.0’s multi-energy network benchmark provides a typical use case for using co-simulation
  - couple multiple domains: power, heat, control
  - connect domain-specific simulators
  - combined analysis of all domains in high detail

Links

- ERIGrid 2.0 multi-energy networks benchmark: https://github.com/ERIGrid2/benchmark-model-multi-energy-networks/tree/mooc-demo
- ERIGrid 2.0 multi-energy networks benchmark online demo: https://mybinder.org/v2/gh/ERIGrid2/benchmark-model-multi-energy-networks/mooc-demo?labpath=Welcome.ipynb
- mosaik: https://mosaik.readthedocs.io/
- DisHeatLib: https://github.com/AIT-IES/DisHeatLib
- pandapipes: https://pandapipes.readthedocs.io
- pandapower: https://pandapower.readthedocs.io

Backup: Voltage Controller Implementation

- Python implementation
- straightforward procedural implementation:
  - simple calculations:
    - voltage deviation
    - step function residual
    - power consumption setpoint for heat pump
  - simple logical comparisons:
    - check deadband
    - min/max of power consumption setpoint
    - ramp up/down constraints



```
def step_single(self, time):

    # Increment counter.
    self.hp_operation_steps += 1

    hp_off = (self.hp_p_el_mw_setpoint == 0)

    if hp_off and (self.hp_operation_steps < self.hp_operation_steps_min):
        return

    # Calculate voltage deviation.
    delta_v_meas_pu = self.vmeas_pu - 1

    delta_vm_lower_pu = self.delta_vm_lower_pu_hp_off if hp_off else self.delta_vm_lower_pu_hp_on

    # Check delta_vm_deadband.
    if delta_vm_lower_pu < delta_v_meas_pu < self.delta_vm_upper_pu:
        if (self.hp_p_el_mw_setpoint == 0 ) and (self.hp_operation_steps >= self.hp_operation_steps_min):
            self.hp_p_el_mw_setpoint = self.hp_p_el_mw_min
            self.hp_p_el_kw_setpoint = 1e3 * self.hp_p_el_mw_setpoint
            self.hp_operation_steps = 0 # Turn on HP --> reset counter
        return

    # Calculate residual.
    res = self.k_p * (delta_v_meas_pu - self.delta_vm_deadband) / self.hp_p_el_mw_step
    step_res = int(res)

    # Use step functions to adapt HP setpoint.
    if fabs(res - step_res) > self.hp_p_el_mw_step:
        self.hp_p_el_mw_setpoint += self.hp_p_el_mw_step * (step_res + 1)

    # Check min and max for HP setpoint.
    if (self.hp_p_el_mw_setpoint > self.hp_p_el_mw_rated):
        self.hp_p_el_mw_setpoint = self.hp_p_el_mw_rated
    elif (self.hp_p_el_mw_setpoint < self.hp_p_el_mw_min) and (self.hp_operation_steps >= self.hp_operation_steps_min):
        self.hp_p_el_mw_setpoint = 0
        self.hp_operation_steps = 0 # Turn off HP --> reset counter
    elif (self.hp_p_el_mw_setpoint < self.hp_p_el_mw_min) and (self.hp_operation_steps < self.hp_operation_steps_min):
        self.hp_p_el_mw_setpoint = self.hp_p_el_mw_min

    self.hp_p_el_kw_setpoint = 1e3 * self.hp_p_el_mw_setpoint
```