Building Distributed Co-simulations using CoHLA

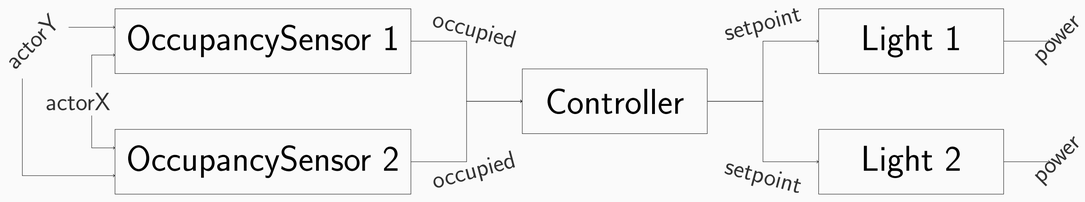
Thomas Nägele[a], Jozef Hooman[a,b], Jack Sleuters[b]
[a]Radboud University, [b]ESI (TNO), The Netherlands

CoHLA: Configuring HLA

- Goal: support concurrent **multi-disciplinary development** of Cyber-Physical Systems (CPSs)
- High Level Architecture (**HLA**) standard is used for co-simulation execution
- Domain Specific Language (**DSL**) to specify co-simulation of different types of models on HLA
- Generates runnable co-simulation code for use with OpenRTI
- Supported model formats:
  - Functional Mock-up Interface standard (**FMI**)
  - Parallel Object-Oriented Specification Language (**POOSL**)
- Allows addition of loggers, actors and basic metric processors

Example: Smart lighting system

- Smart lighting system for an office building
- Building consists of different types of areas
- Three types of models
  - **Lighting controllers**: discrete-time, POOSL
  - **Lights**: continuous-time, FMU
  - **Occupancy sensors**: discrete-time, FMU



Implementation using CoHLA

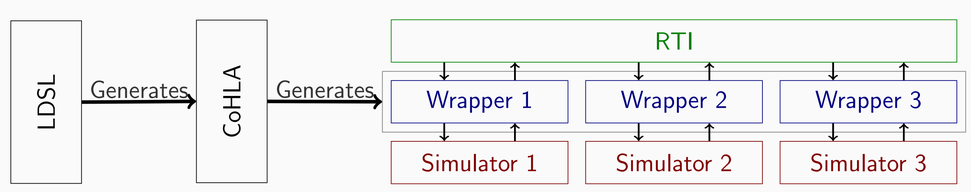
```
1  Confederation VerySmallBuilding {
2    Instances {
3      Instance r1Controller as BasicRoomController
4      Instance r1Light1 as DimmableLight
5      Instance r1Sensor1 as OccupancySensor
6      Instance r2Controller as BasicRoomController
7      Instance r2Light1 as DimmableLight
8      Instance r2Sensor1 as OccupancySensor
9      Instance corridorController as CorridorController
10     Instance corridorLight1 as DimmableLight
11     Instance corridorSensor1 as OccupancySensor
12   }
13   Connections {
14     Connection { r1Light1.setpoint <- r1Controller.setpoint }
15     Connection { r2Light1.setpoint <- r2Controller.setpoint }
16     Connection { corridorLight1.setpoint <- corridorController.setpoint }
17     Connection { r1Controller.occupied - r1Sensor1.occupied }
18     Connection { r2Controller.occupied - r2Sensor1.occupied }
19     Connection { corridorController.activity - corridorSensor1.occupied }
20     Connection { corridorController.relatedActivity <- r1Sensor1.occupied }
21     Connection { corridorController.relatedActivity <- r2Sensor1.occupied }
22   }
23 }
```

Listing 1: Definition of a small building co-simulation in CoHLA.

- **Scalability** is difficult: buildings quickly become very large
  - CoHLA **definition** also becomes very large
  - **Simulation time** increases rapidly

Lighting DSL

- A DSL to simplify the definition of a co-simulation for a building
- Allows the user to specify a building by specifying areas
- From this building specification, the following is generated
  - A **CoHLA definition** for the building
  - An **image** of the building
  - A web page to show display resulting log files



- The building from Listing 1 can be defined using the following Lighting DSL code.

```
Building VerySmallBuilding {
  Room r1 {
    Draw: (0,0) (600,0) (600,300)
      (0,300)
    Devices {
      Light l1 on (300,150)
      Sensor s1 on (300,150)
    }
  }
  Room r2 {
    Draw: (0,450) (600,450) (600,750)
      (0,750)
    Devices {
      Light l1 on (300,600)
      Sensor s1 on (300,600)
    }
  }
  Corridor cor {
    Draw: (0,300) (600,300) (600,450)
      (0,450)
    Rooms: room1 room2
    Devices {
      Light l1 on (300,375)
      Sensor s1 on (300,375)
    }
  }
}
```

Distributed simulation

- Options for distributed simulation execution added to CoHLA
  - **Automatic**, without configuration
  - **Manual**, simulations assigned to computation nodes
    Generated by Lighting DSL: group connected simulations
- Distribution as supported by OpenRTI
- Sample building: 36 lights, 38 sensors, 14 controllers
- Distribute over up to 7 computation nodes in the **cloud**

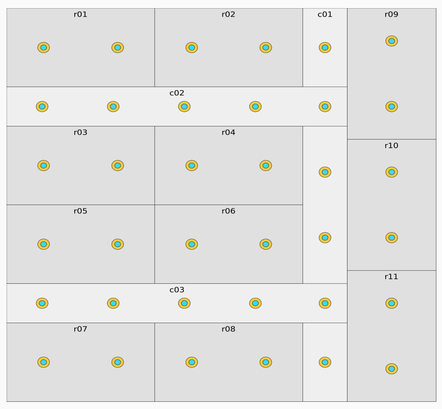
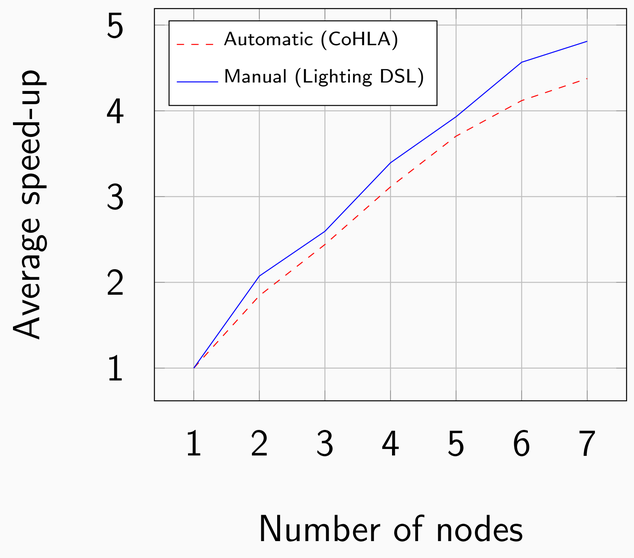


Figure: *Sample building.*



Conclusion

- CoHLA allows fast construction of co-simulations
- Development of a separate DSL saves time
- Distributed simulation improves simulation speed

NWO Applied and Engineering Sciences
Radboud University