Python and .NET adapters for HLA-based co-simulations of CPS

Bastian Cornelsen & Dennis Weller

ABOUT US

- Bastian Cornelsen

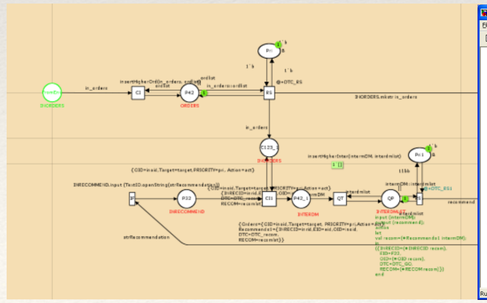
- Dennis Weller

- Computer science students at the University of Oldenburg
- Research assistants at the OFFIS (Research Institute in Oldenburg) in the deparment for energy informatics

CO-SIMULATION FOR CYBER-PHYSICAL SYSTEMS

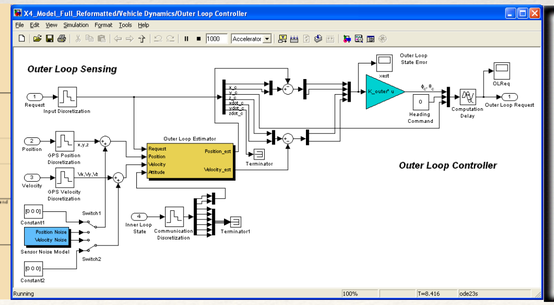
- Large System-of-Systems (SOSs)
  - Interdependent systems which require special-purpose simulators
- Challenges:
  - Integrate simulated heterogeneous system components
  - Integrate simulation engines
  - Integrate hardware, systems, and humans that operate in real-time
  - Rapidly synthesize and deploy integrated simulations
  - Simulators have different timing models
  - Execution needs to be coordinated
  - Data needs to be shared
  - Different modelling languages
  - Different semantics (Continuous time, discrete time, discrete event)
  - No support for specifiying experiments

Organization/Coordination



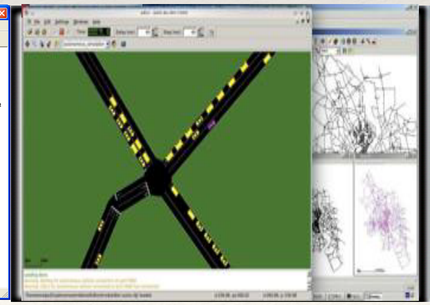
CPNTools

Controller/Vehicle Dynamics



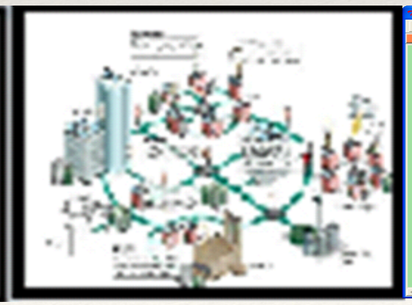
Simulink

Road Traffic Network



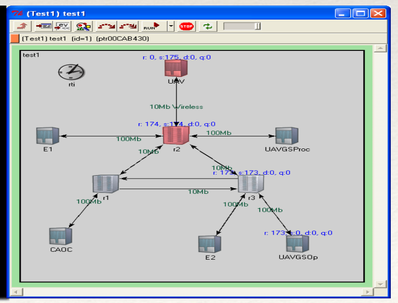
SUMO

SmartGrid tools



Gridlab-D

Network Architecture

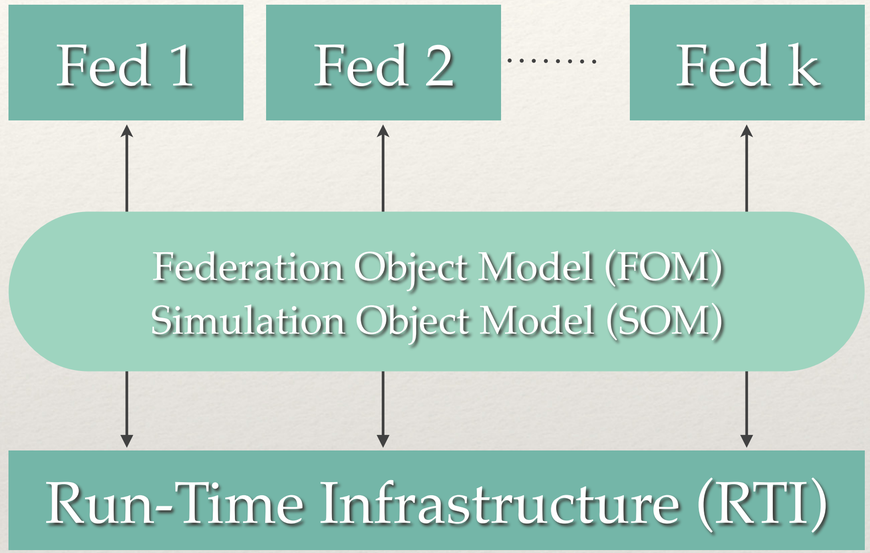


OMNet++

HIGH LEVEL ARCHITECTURE (HLA)

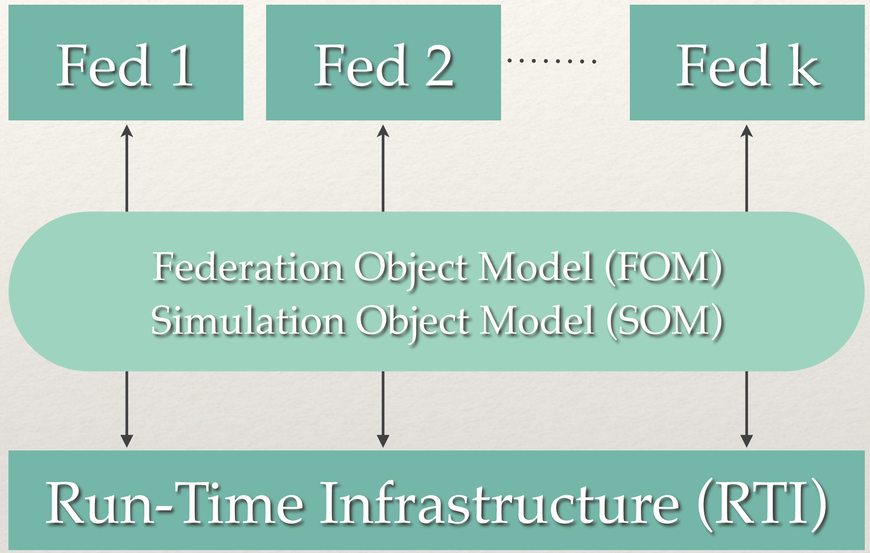
- IEEE standard that defines how distributed simulations interoperate

1. Interface specification
2. Object Model Template
   - FOM, SOM, MOM
3. Rules and HLA services
   - time, object, federation, ownership, declaration management, data distribution services



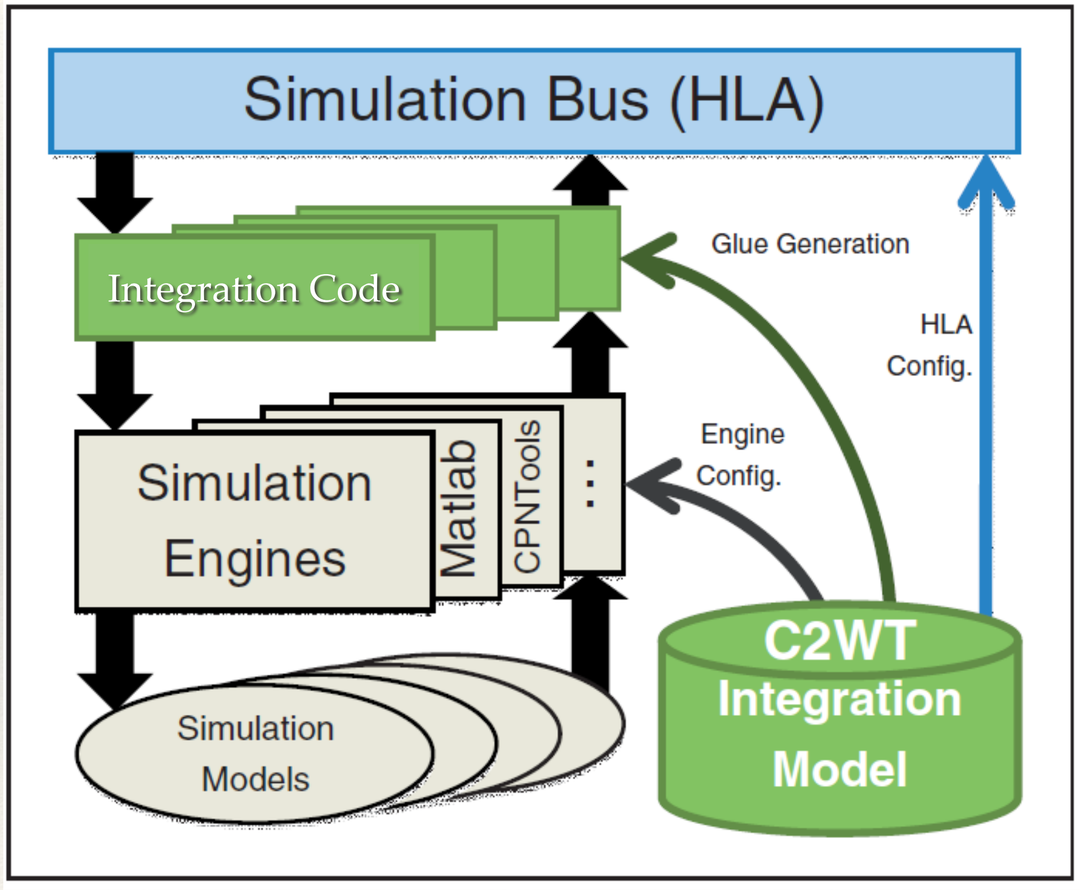
HLA ADVANTAGES

- Flexible data model tu support all types of simulation like real-time, HIL, synchronized
- Standardized way to support fundamental requirements of distributed simulations
- Well-defined rules, interfaces, and APIs with clear semantics
- Flexible architecture for customization and extensibility



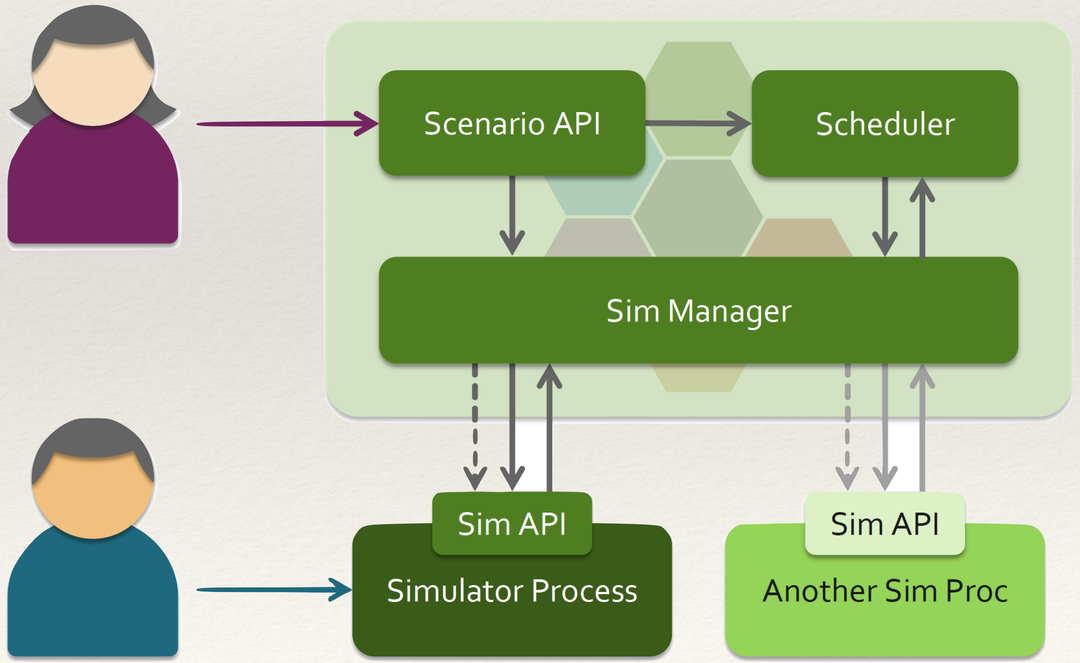
MODEL-BASED SIMULATION INTEGRATION IN CYBER-PHYSICAL SYSTEMS WIND TUNNEL

- Supports commonly used simulation tools
  - Matlab/Simulink
  - OMNeT++
  - Colored Petri Nets
  - SUMO Road Transportation Simulator
  - TrainDirector Railroad Simulator
  - GridlabD Dibstribution Grid Simulator
  - Generic languages Java, C++
- Advanced Technologies
  - Advanced deployment of simulation experiments
  - Creating scenario models for
  - Interacting/perturbing simulations at run-time
  - Reusable communication network simulation
  - A modular cyber-attack library



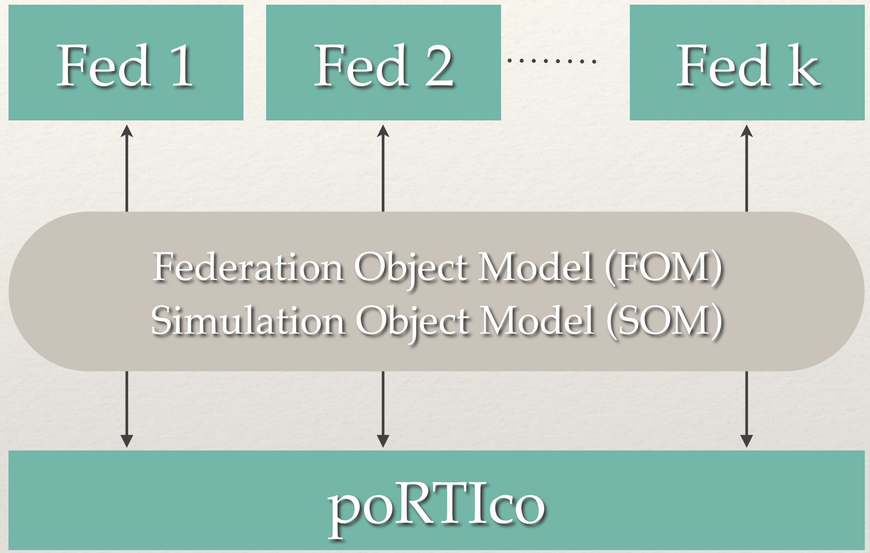
mosaik

- Flexible smart grid co-simulation framework developed at the OFFIS
- Written in Python 3
- Open Source
- APIs for Python (2.x & 3.x), Java, Matlab
- Event-discrete simulation



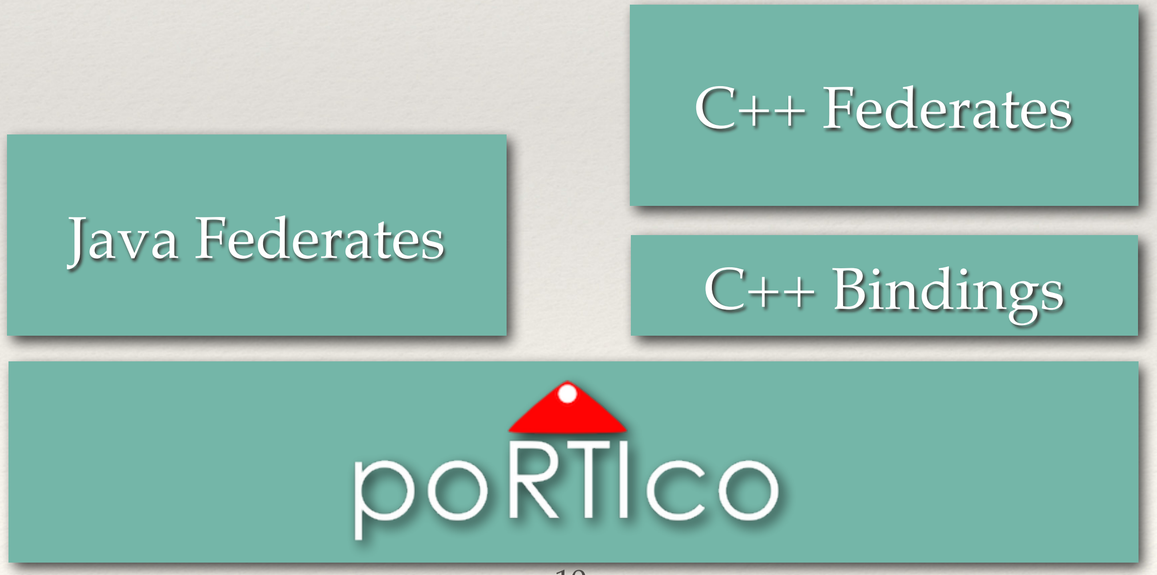
the poRTIco project

- Fully supported HLA RTI implementation
- Open Source
- Cross-platform -> Unix-like systems, Windows
- Provides Java and C++ APIs



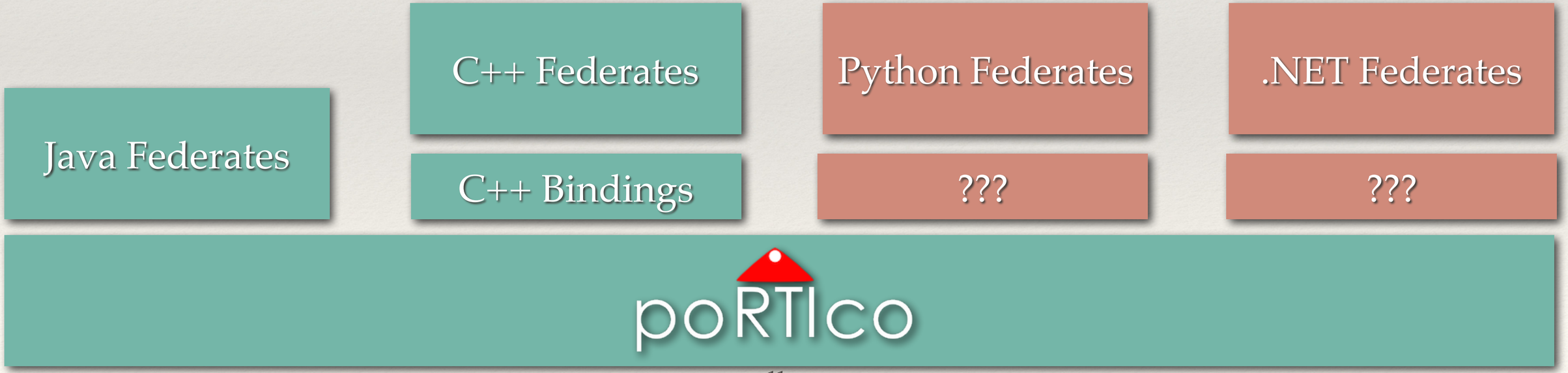
OUR TASKS

- Creating an adapter to enable application of simulators written in Python and .NET languages like C#



OUR TASKS

- Creating an adapter to enable application of simulators written in Python and .NET languages like C#

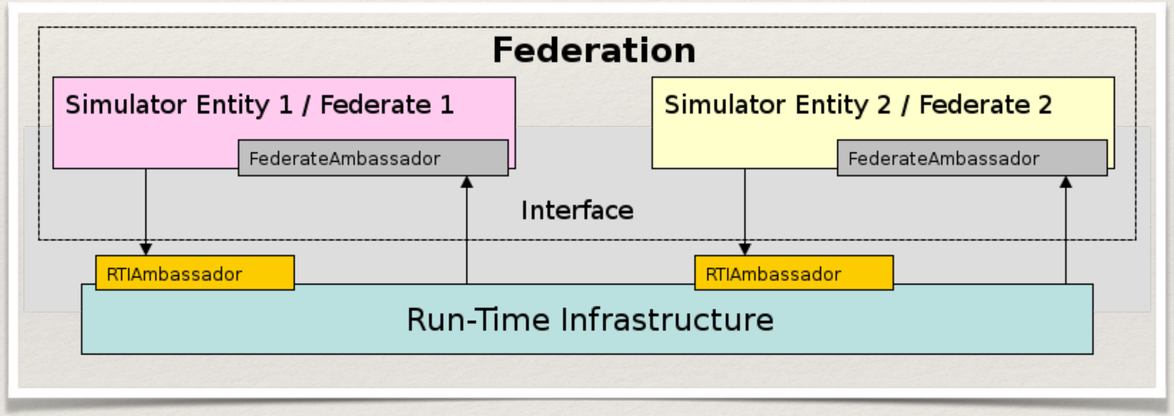


python™

Python ⇄ Java

FROM PYTHON TO JAVA

- Requirements:
  - Use Java Code in Python
  - Inject Callbacks from Python into Java
- Solutions:
  - Jython
  - Py4J

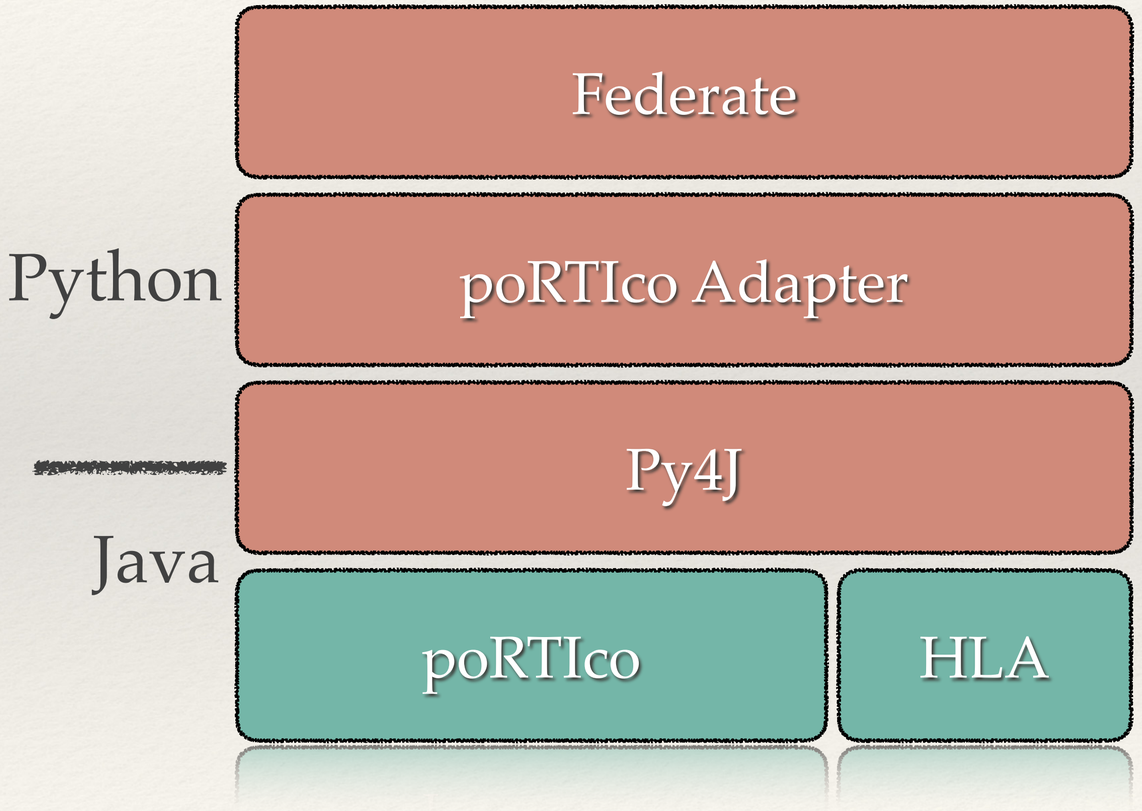


PY4J

- Connects Java and Python via sockets
- Supports:
  - Callbacks
  - Essential non primitive datatypes
  - Inheritance from java interfaces

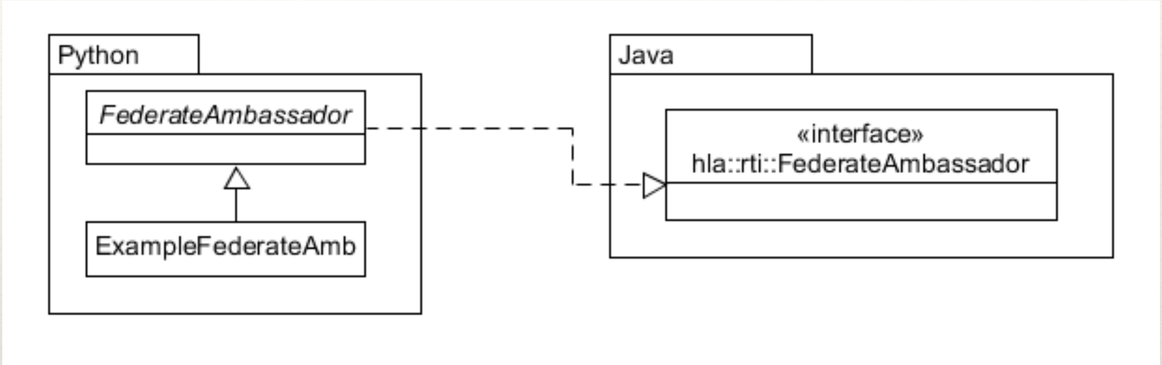
PY4J

- Connects Java and Python via sockets
- Supports:
  - Callbacks
  - Essential non-primitive datatypes
  - Inheritance from java interfaces



INHERITANCE

- Implement Java interfaces in python and inject it into java objects
- Used for Callbacks
- Method distinction:
  - Python: Just by name
  - Java: By name and attribute types

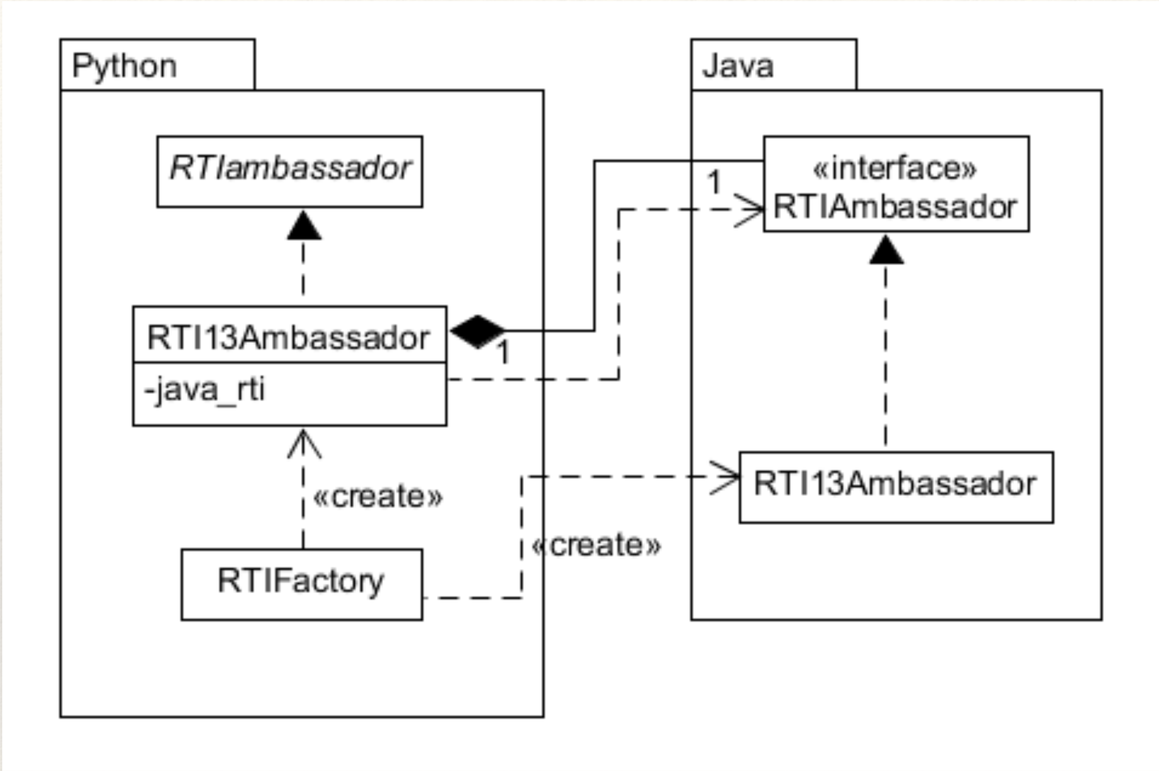



```java
package hla.rti;

public interface LogicalTime
{
   public void decreaseBy( LogicalTimeInterval subtrahend )
      throws IllegalTimeArithmetic;
                                ⋮
}
```

```python
class LogicalTime:

@abstractmethod
def decreaseBy(self, subtrahend):
    """subtrahend LogicalTimeInterval"""
    raise NotImplementedError

                              ⋮

    class Java:
        implements = ["hla.rti.LogicalTime"]
```

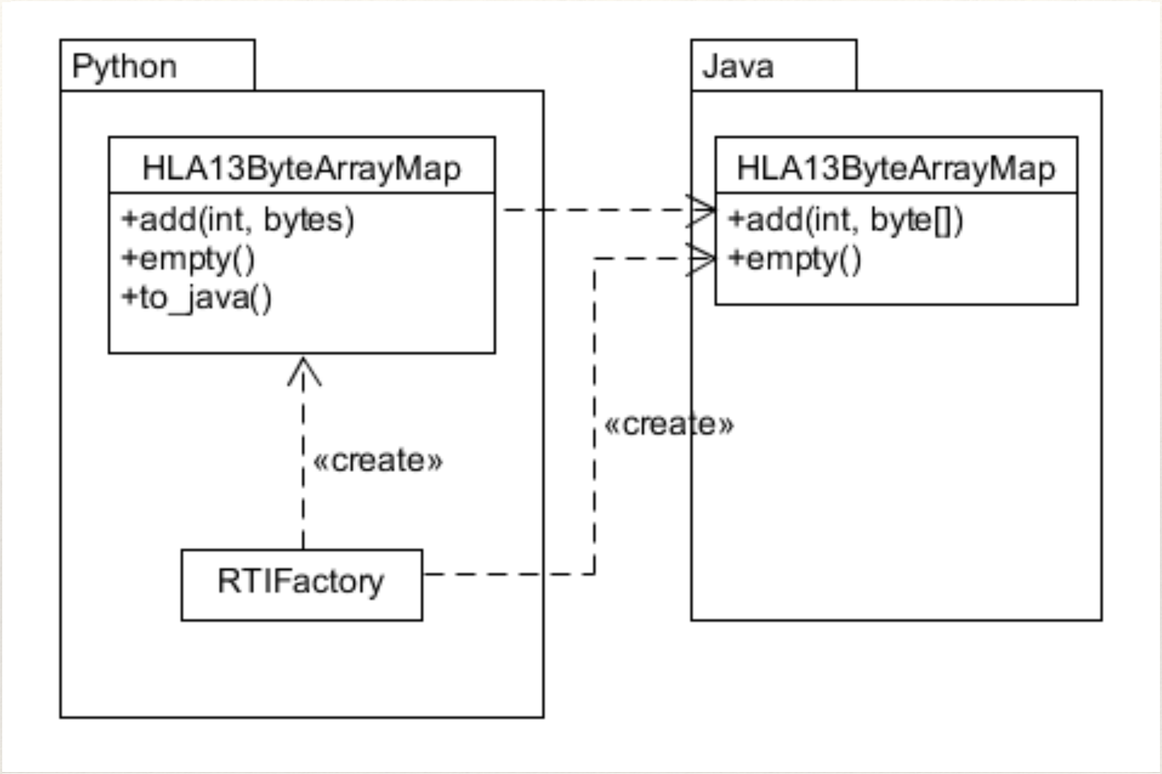
WRAPPING

- Saved java instance in wrapping class
- Relays called methods to java object
- If necessary: to_java() function



WRAPPING

- Saved java instance in wrapping class
- Relays called methods to java object
- If necessary: to_java() function




```
def publish_object_class(self, class_handle: int, attributes: HLA13AttributeHandleSet):
    self.java_rti_ambassador.publishObjectClass(class_handle, attributes.to_java())
```

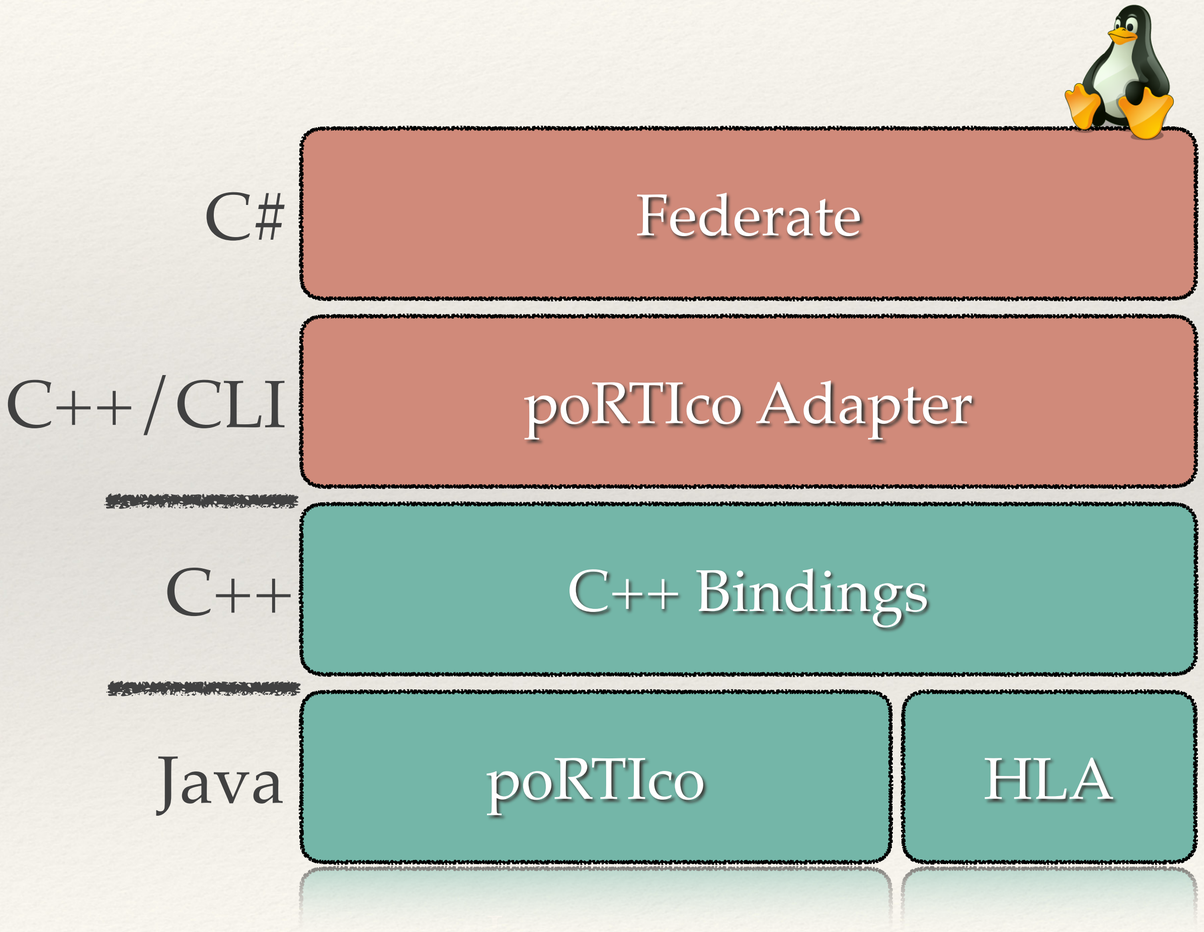
OTHER TRICKY ISSUES

- Exception handling
  - Conversion from Java to Python
  - Parsed automatically with reflection
- Dynamic ports in Py4J
- Debugging: Run java and python part separately
- Multithreading synchronization

C#.NET ⇄ Java/C++

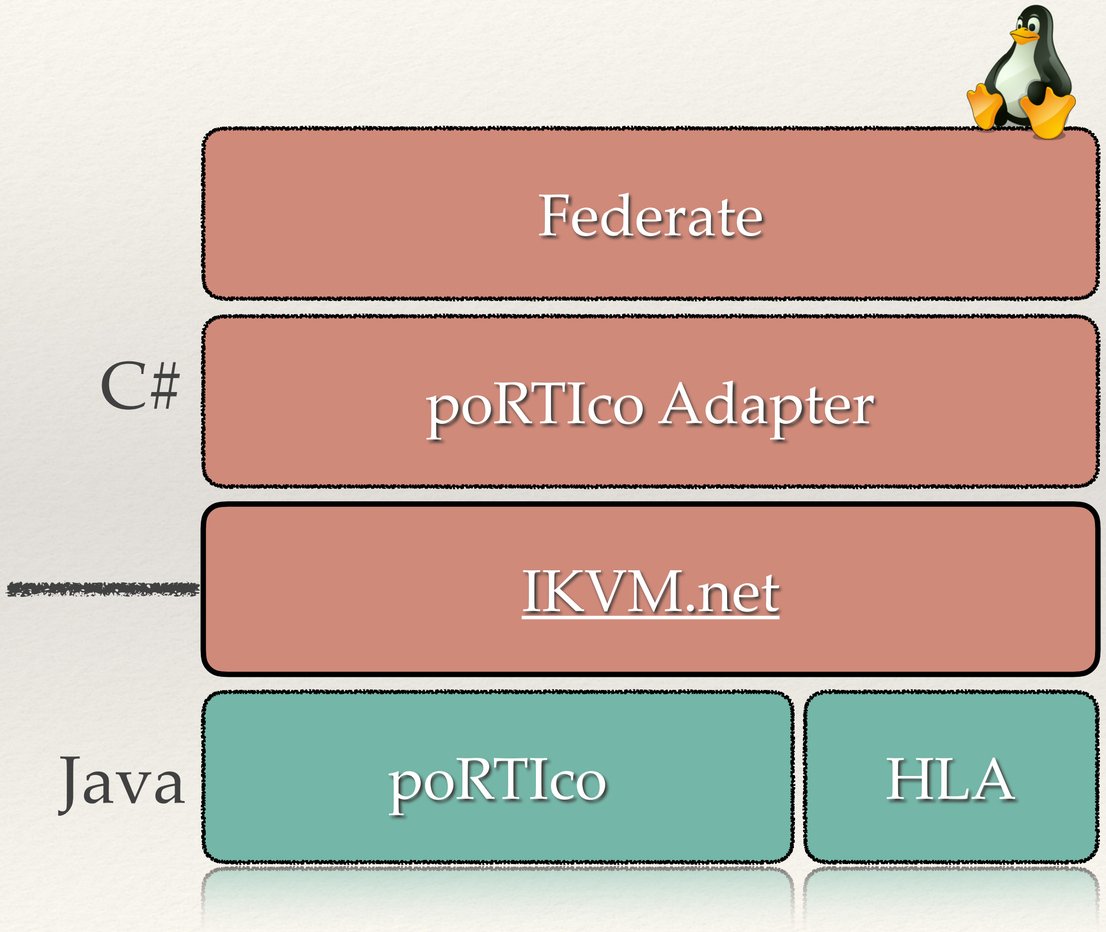
ADAPTER FOR C#/.NET (1)

- Utilize existing C++ bindings, write managed code
- C++/CLI as extension for .NET access
- Example: char* -> String^
- Problem: C++/CLI is supported under .NET Core, but not Linux



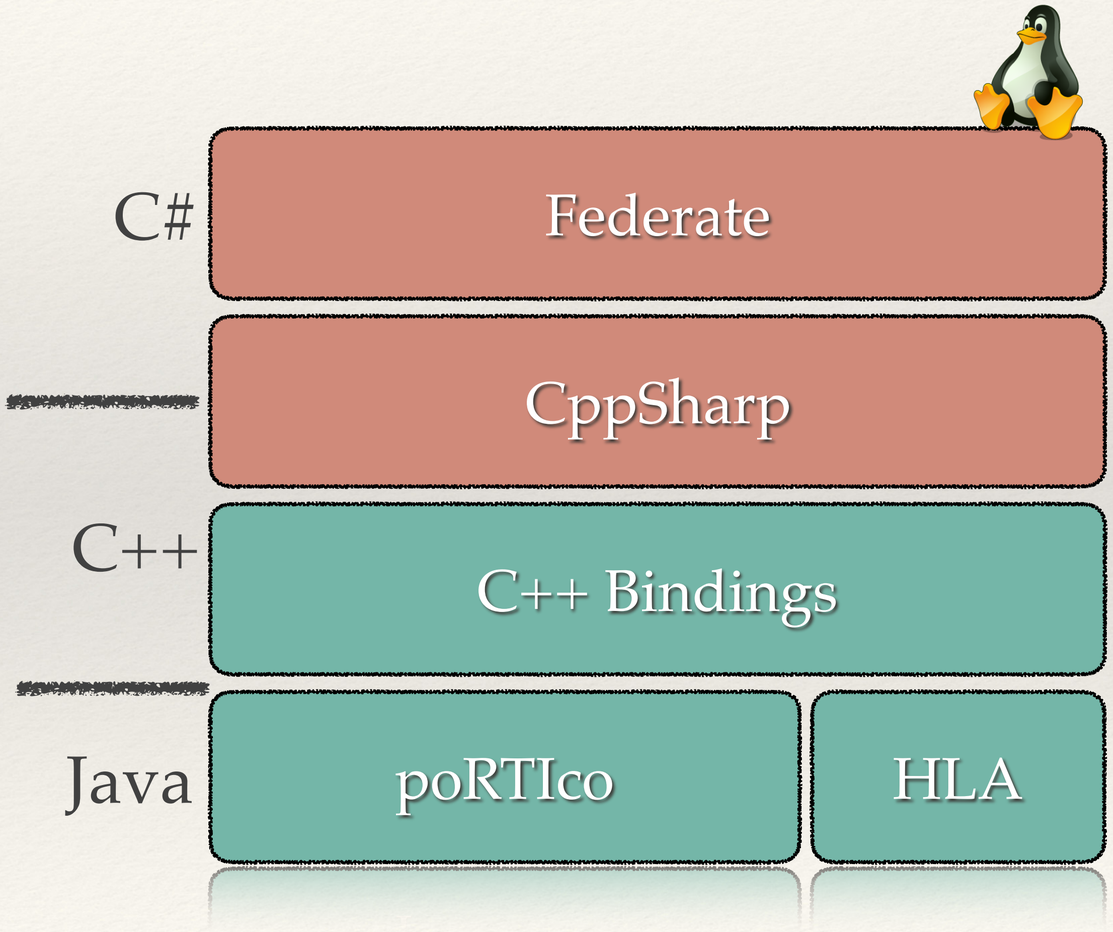
ADAPTER FOR C#/.NET (2)

- Make Java classes work in .NET
- IKVM.net -> Implementation of Java for Mono and .NET
- Problem: Java 8 not supported, development stopped recently



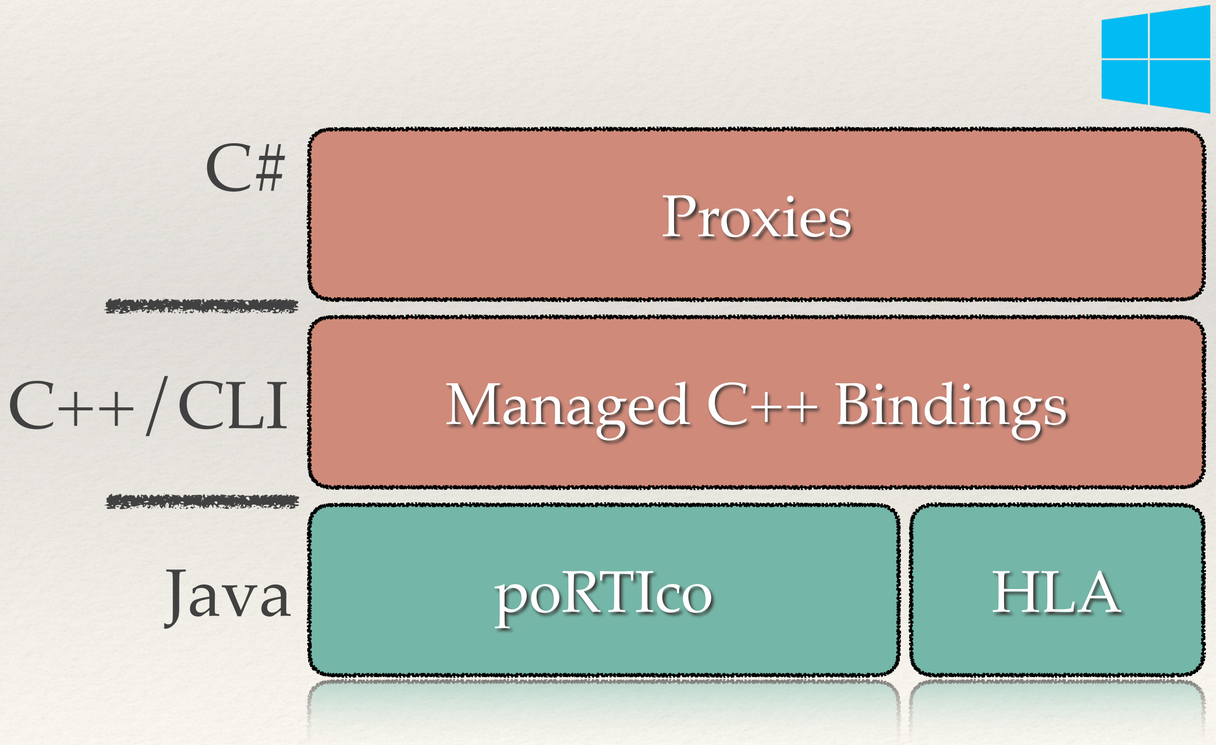
ADAPTER FOR C#/.NET (3)

- CppSharp -> generates a managed API from C++ headers and libraries for Mono
- Linux implementation experimental and feasibility not predictable
- Current problem: Demands (Visual C++).lib files



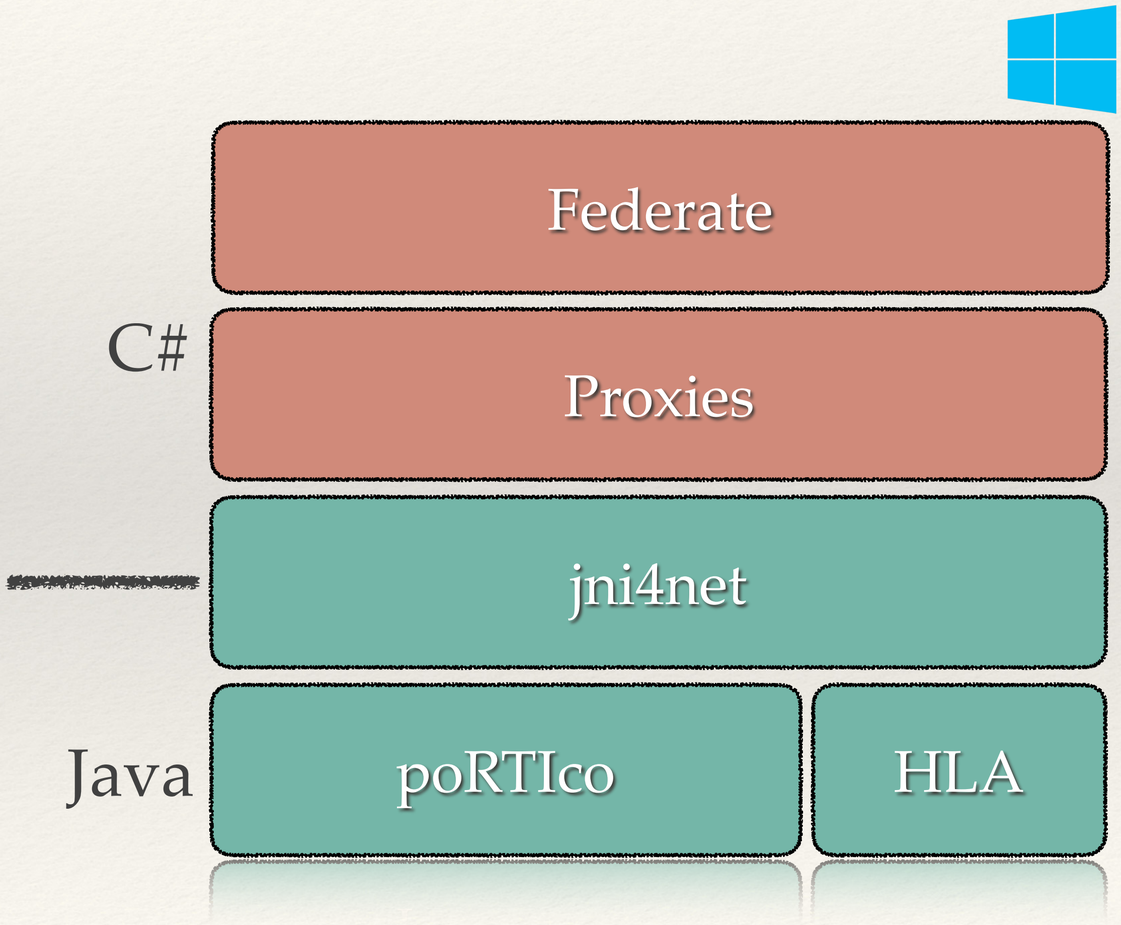
ADAPTER FOR C#/.NET (4)

- Compromise: C# federates could run in Windows
- Create "managed" C++ Bindings with C++/CLI
- Problem: Difficult C++ code and lack of experience lead to:
  - Incalculable time exposure
  - Complicated debugging



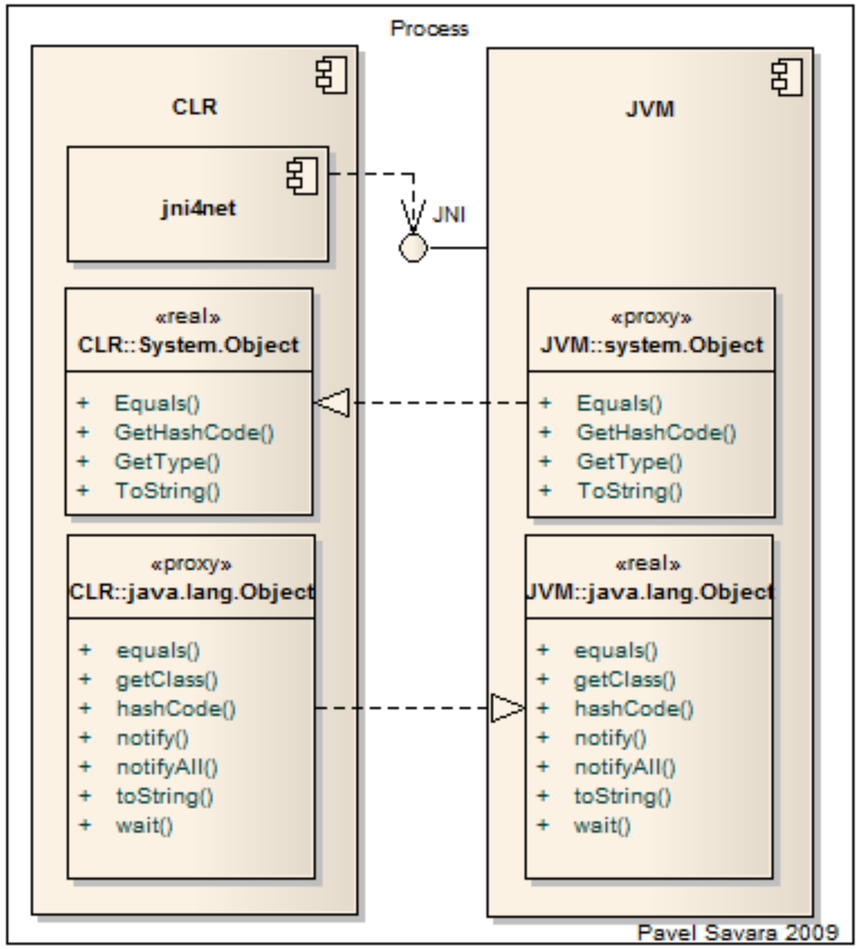
ADAPTER FOR C#/.NET (5)

- jni4net: bridge between Java and .NET
- C++ bindings already implemented via JNI
- Use JNI to forward the call from .NET proxies to methods on real Java objects



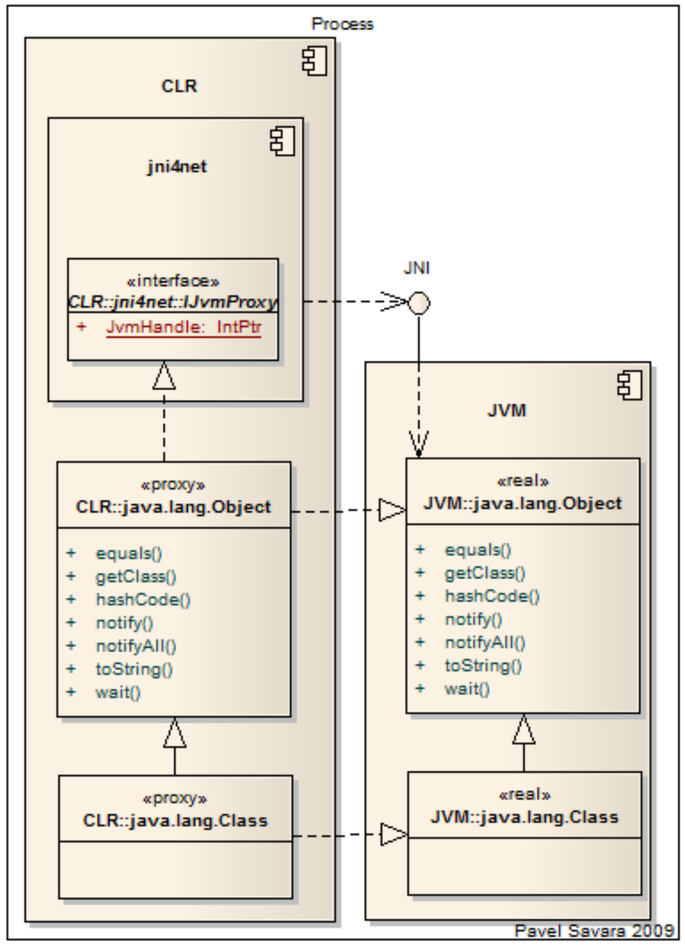
JNI4NET – GENERAL IDEA

- Generate proxy classes by using reflection to grab public method signatures
- Use JNI to forward call from .NET proxies to methods on real Java objects
- Use JNI to register .NET implementation of native methods of Java proxies to forward call to methods on real .NET objects



JNI4NET - .NET CALLS JAVA

- jni4net wraps JNI interface of JVM
- Proxies use JNI to marshall method calls from CLR to JVM
- Proxy instance holds JNI handle of real Java instance
- Proxies form same hierarchies



CREATING PROXY FILES

- Portico comes with more than one thousand java classes
- Proxy files can be created with proxygen

```
public void setTime(double time)
{
  this.time = time;
}
```

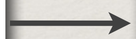


```
[global::net.sf.jni4net.attributes.JavaMethodAttribute("(D)V")]
public virtual void setTime(double par0) {
    global::net.sf.jni4net.jni.JNIEnv @__env = this.Env;
    using(new global::net.sf.jni4net.jni.LocalFrame(@__env, 12)){
    @__env.CallVoidMethod(this, global::org.portico.impl.hla13.types.DoubleTime.j4n_setTime16,
        global::net.sf.jni4net.utils.Convertor.ParPrimC2J(par0));
    }
}
```

JNI4NET CHALLENGES

- .NET IDE Visual Studio returns odd errors what impedes debugging
- Creating proxies with proxygen works only partly ....

```
package org.portico.impl.hla13.types;

import hla.rti.IllegalTimeArithmetic;
import hla.rti.LogicalTime;
import hla.rti.LogicalTimeInterval;
import java.io.ByteArrayOutputStream;
import java.io.DataOutputStream;
import java.io.IOException;

public class DoubleTime
  implements LogicalTime
{
  public static final double INITIAL = 0.0D;
  public static final double FINAL = Double.MAX_VALUE;
  private double time;

  public DoubleTime()
  {
    this.time = 0.0D;
  }
```



```
namespace org.portico.impl.hla13.types {


    #region Component Designer generated code
    [global::net.sf.jni4net.attributes.JavaClassAttribute()]
    public partial class DoubleTime : global::java.lang.Object {

        internal new static global::java.lang.Class staticClass;

        internal static global::net.sf.jni4net.jni.MethodId j4n_encode0;

        internal static global::net.sf.jni4net.jni.MethodId j4n_isFinal1;
```

OUTLOOK

- Thread safety of CPSWT in Python
- Interconnection of MOSAIK and poRTIco
- Finish .NET implementation and create simple example
- Explore option to run federates on Linux via Wine compatibility layer

Thank you for your attention.

Any questions?