SDL Code Generation for Network Simulators

Mihal Brumbulli, Joachim Fischer

{brumbull, fischer}@informatik.hu-berlin.de

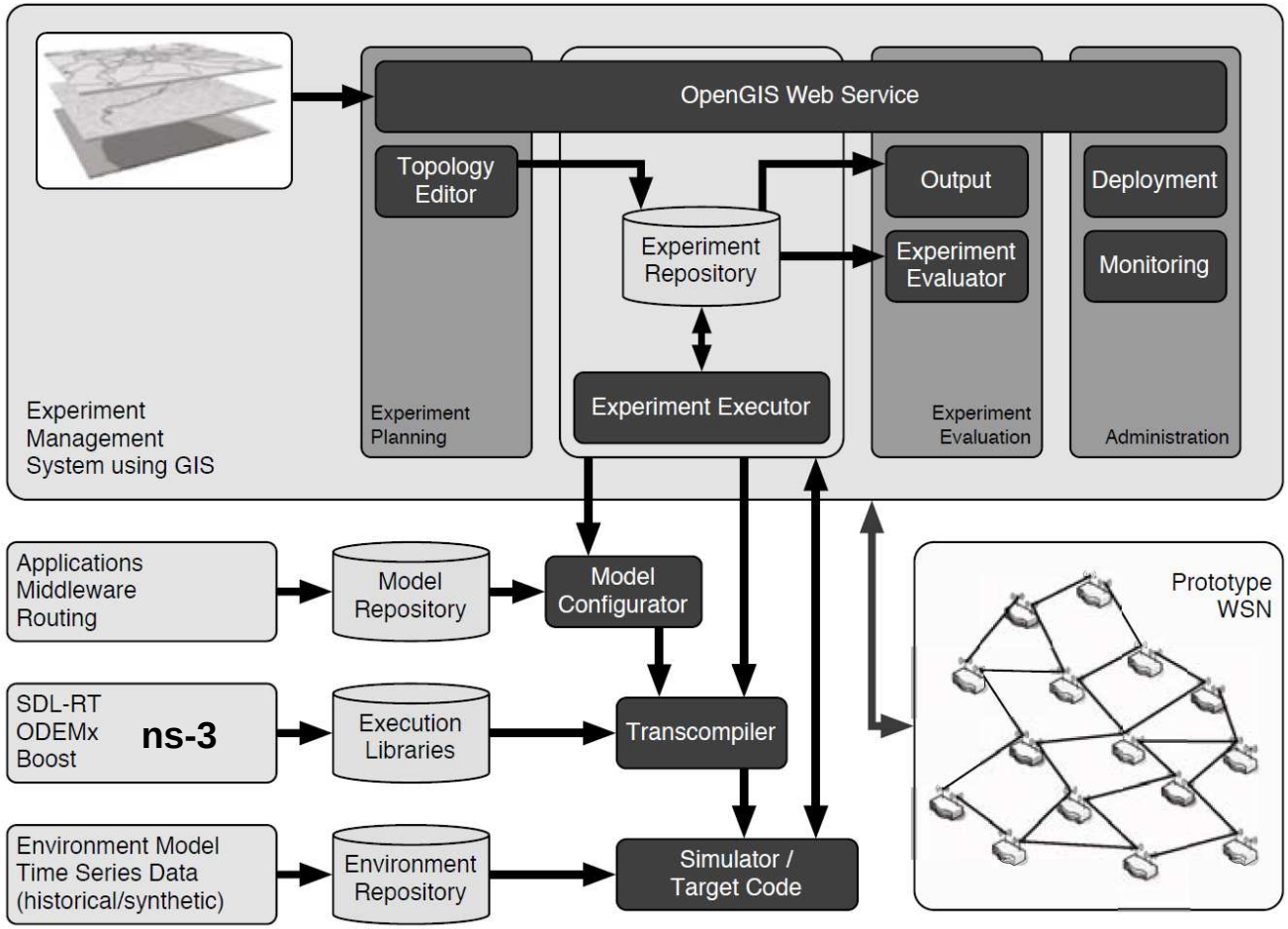
Outline

- Introduction
- Motivation
- The Network Simulator ns-3
- Code Generation
- Conclusions

Introduction

- Network simulators:
  - Distributed and networked environment
  - Hand coded system implementation
- System development tools:
  - Automatic system implementation from graphical representation
  - No distributed environment
- Goal: Automatic system implementation and simulation in a distributed and networked environment

Motivation


OpenGIS Web Service
Topology Editor
Experiment Repository
Output
Experiment Evaluator
Deployment
Monitoring
Experiment Executor
Experiment Management System using GIS
Experiment Planning
Experiment Evaluation
Administration
Applications Middleware Routing
Model Repository
Model Configurator
SDL-RT ODEMx Boost
ns-3
Execution Libraries
Transcompiler
Environment Model Time Series Data (historical/synthetic)
Environment Repository
Simulator / Target Code
Prototype WSN

The Network Simulator ns-3

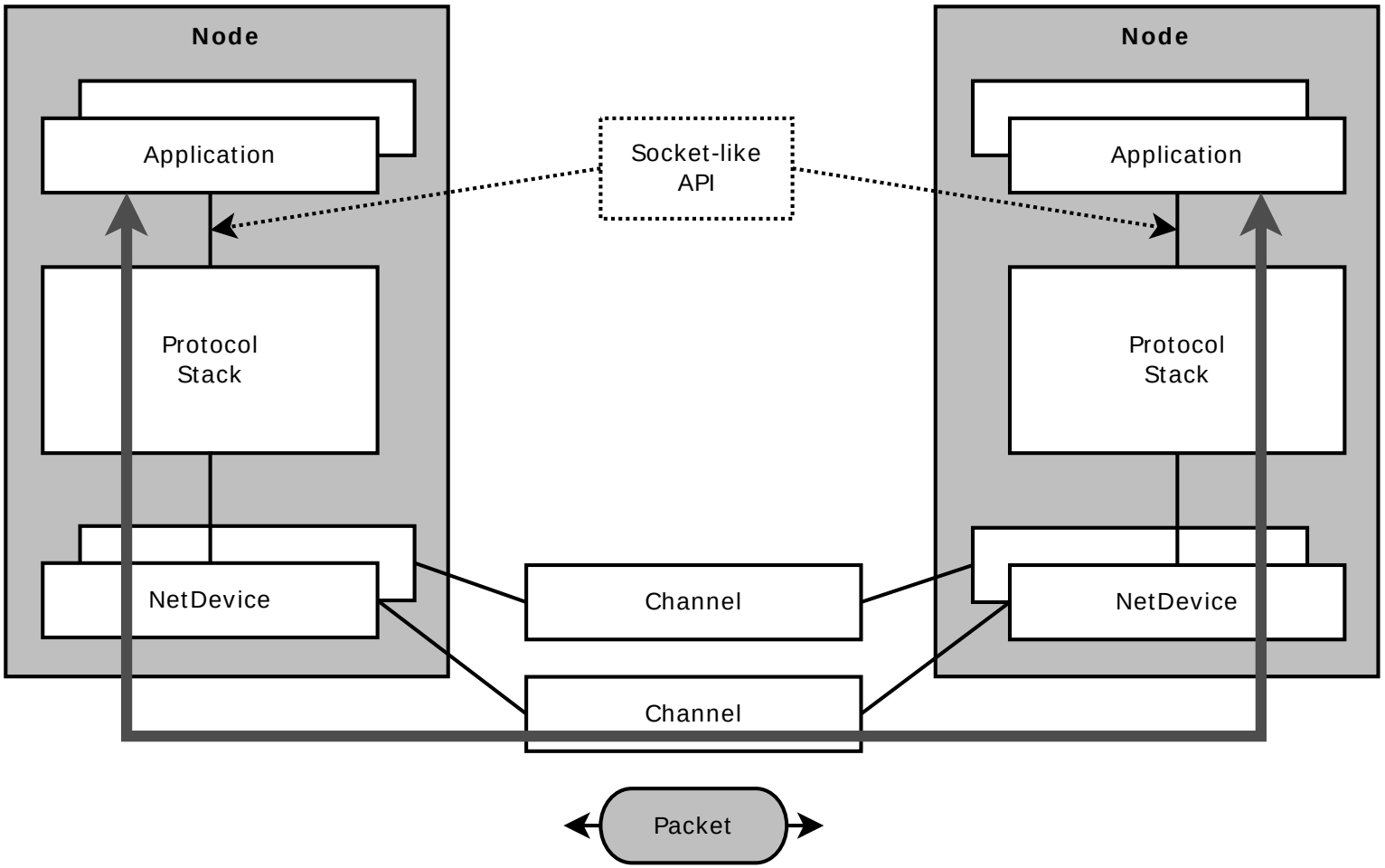
└> Introduction

- Discrete-event network simulator written in C++
- Simulations are C++ executables
- Composed of modules:

Core	Callbacks, debugging facilities, random variable distributions, objects, smart pointers
Simulator	Time management, scheduler, simulator
Common	Packet class to create and manipulate simulation packets
Mobility	Mobility models, traces for course changes of a mobility model
Node	Network node, MAC-layer, application-layer API
Devices	Bridge, CSMA, Emulated device, Mesh, Point-to-Point, Tap Bridge, Wi, WiMAX
Internet-Stack	Ipv4 stack, ARP, UDP and TCP
Routing	AODV, Global Routing, List Routing, Nix-vector Routing, OLSR, Static Routing
Helper	High-level API wrappers for everything else

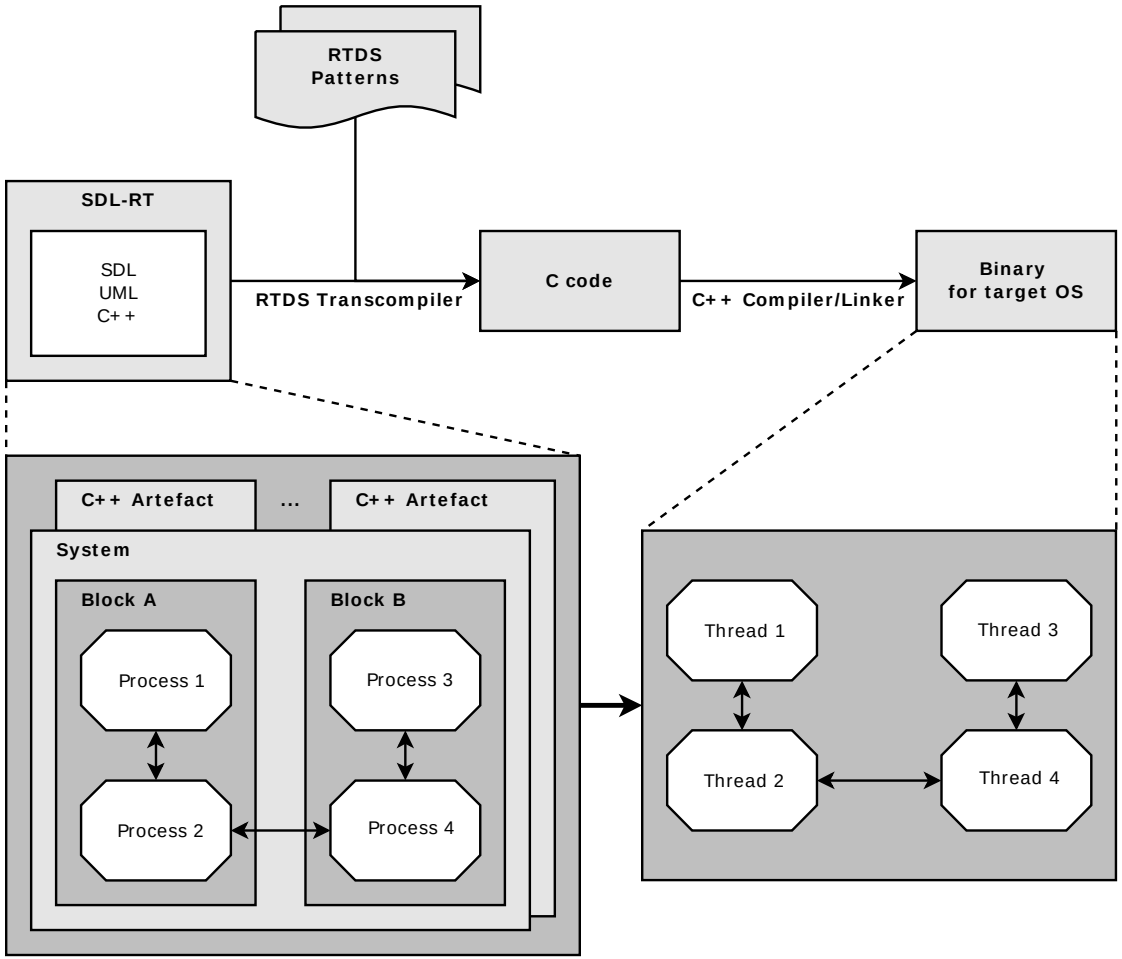
The Network Simulator ns-3

└> Simulation Model



Code Generation

└> RTDS



Code Generation

└> HUB Transcompiler

- Maps SDL blocks to C++ classes that are populated with other blocks or processes
- SDL processes become C++ classes that have SDL procedures as their methods

Component	ns-3 implementation
RTDS_system	ns3::ApplicationContainer which encapsulates all building blocks and contained processes
RTDS_block	ns3::ApplicationContainer which encapsulates all sub-blocks and contained processes
RTDS_process	ns3::Application with custom attributes and functions for providing a process like behavior
RTDS_procedure	Procedures are implemented as RTDS_process member functions

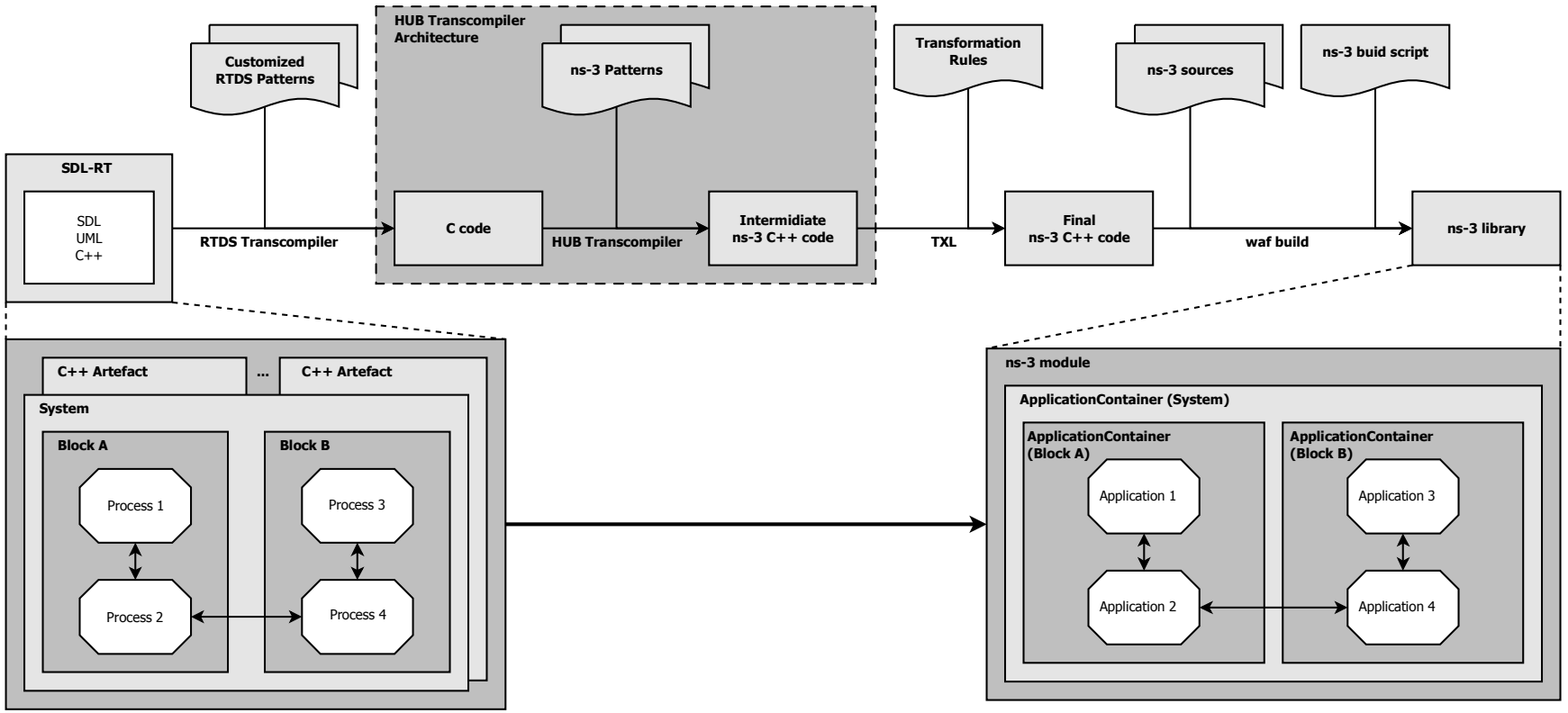
Code Generation

└> Problems and Solutions

- Process based
  - Convert to event based using RTDS patterns
- State variables
  - Make them member variables using code transformation
- Naming
  - Rename using code transformation

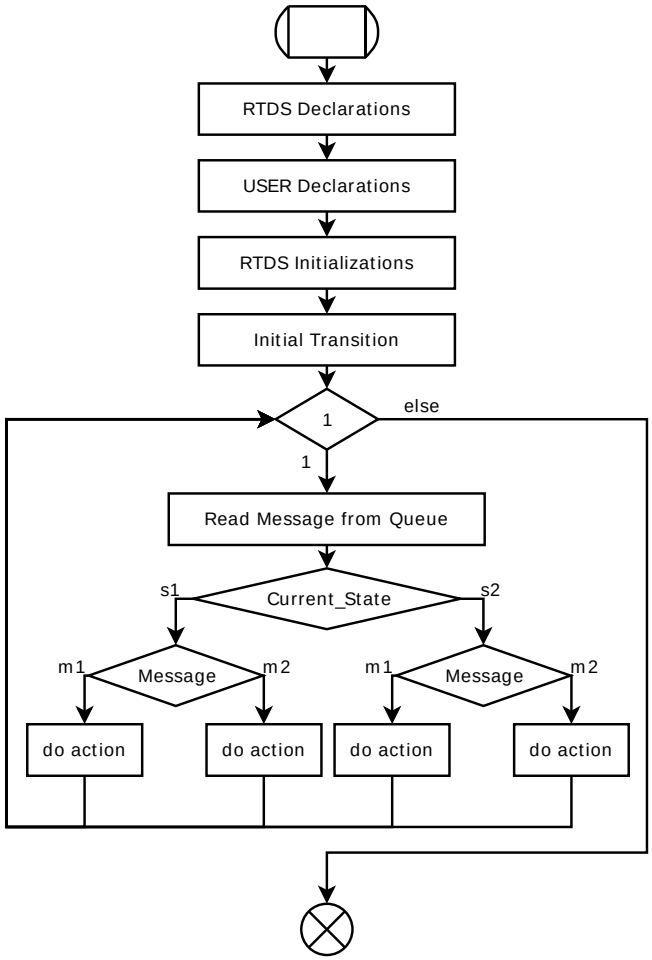
Code Generation

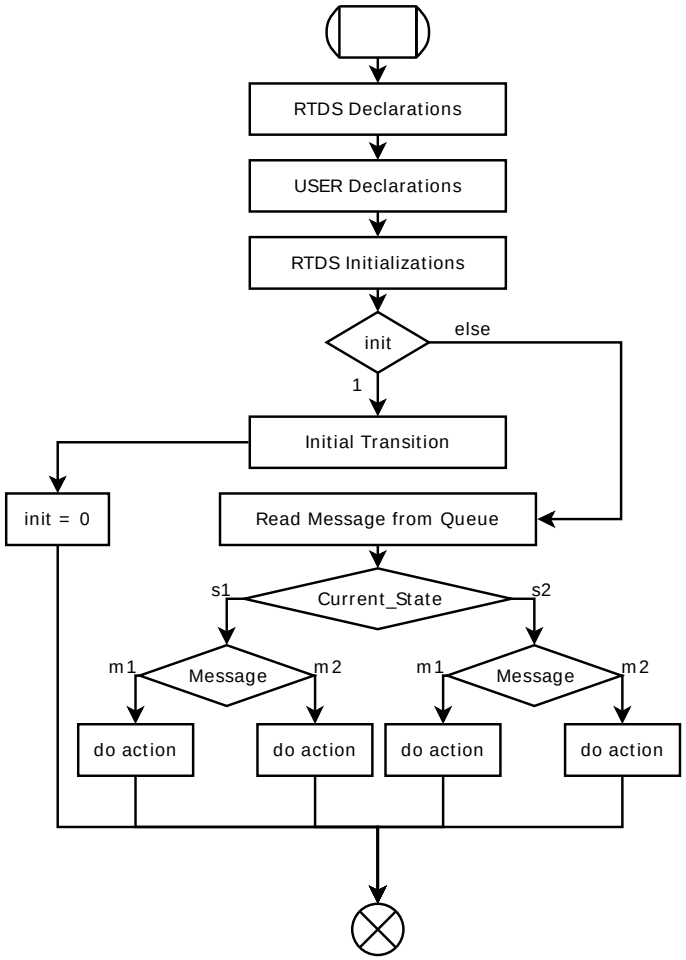
└> The model



Code Generation

↳ TXL Transformations





Code Generation

└> ns-3 Integration

- Source files and headers integrated within ns-3
- Build script automatically generated
- New ns-3 library including system implementation module
- Same as ns3::Application-s available in ns-3

Conclusions

- Combine SDL-RT design specifications with ns-3 network models
- No external mechanism are used for bridging system implementation with the simulator
- New ns-3 module integrated within the simulator
- Suitable for simulating large scale distributed networked systems

Thank You!

?|!