IKR EmuLib

A Library for Seamless Integration of Simulation and Emulation

Marc Necker, Christoph Gauger
[necker | gauger]@ikr.uni-stuttgart.de

ITG FG 5.2.1 Workshop Simulationstechnik
Mittweida, November 3, 2005

Outline:
• Introduction of IKR SimLib
• Motivation of emulation approach
• Realization of IKR EmuLib
• Evaluation of accuracy
IKR SimLib

- **History**
  - origin: Pascal simulation library (1980ies)
  - object-oriented redesign in the context of a dissertation (1992)
  - since then continuously enhanced and improved

- **Implementation**
  - C++ class library
  - usage of additional libraries (e.g., container class library)
  - tested under various platforms: Linux, Solaris, CygWin

- **Main Features**
  - support for transformation of an abstract model into source code
  - control of event-driven simulation
  - random number generation (various distributions and source models)
  - statistical evaluation
  - reading parameter values and printing results
Basic Structure

- basic concepts
  - simulation control
  - event handling
  - random distributions
  - statistical evaluation
  - I/O concept

- modelling concepts
  - basic entity
  - port concept
  - filters & meters

- standard components
  - model components
  - utilities

user
Model Components: Object Hierarchy

Decomposition support by

- **Hierarchy**
  - has-relationship
  - pointer to owner

- **Name Concept**
  - local name as attribute
  - identification of components
  - access via central component manager
Port Concept

Message exchange between model components via ports

- distinction between input and output ports
- central port registration using owner address and port name
- connection of the ports using function call `connect`
- communication via handshake protocol

⇒ uniform interface for communication between model components
Simulation

- fast and easy exploration of vast parameter space
- difficulty to model complex componentes & protocols
Simulation vs. Emulation vs. Prototype

**Simulation**
- fast and easy exploration of vast parameter space
- difficulty to model complex components & protocols

**Emulation**
- easy integration of real world components
- unsuitable to explore large parameter space
Simulation vs. Emulation vs. Prototype

**Simulation**
- Fast and easy exploration of vast parameter space
- Difficulty to model complex components & protocols

**Emulation**
- Easy integration of real world components
- Unsuitable to explore large parameter space

**Prototype**
- Trustworthy results
- Difficult setup, limited parameter space
Integrated Simulation and Emulation

Combined Simulation and Emulation
- simulative exploration of vast parameter space with approximate models
- emulative evaluation of selected parameter points with included real-world components

Design Objectives
- Create emulation extension for existing simulation library
- Enable the reuse of existing simulation models
- Switch between simulation and emulation in zero time
  ➤ Seamless integration
Basic Idea

- Implement emulation as an IP packet router
- Delay or drop IP packets according to model behavior

- Interfaces to model are traffic generators and traffic sinks
Basic Idea

- Implement emulation as an IP packet router
- Delay or drop IP packets according to model behavior

→ Interfaces to model are traffic generators and traffic sinks
Interface to network realized by multithreaded-design

- **Listener Thread**
  - reception of IP packets, filtering, time stamping, buffering

- **Model Thread**
  - encapsulation into simulation messages, model processing, transmission to network interface
Packet arrival
assign timestamp $t_0$
append packet to buffer

Listener Thread

Model Thread

real-time

real-time
Interaction between threads

- **Listener Thread**
  - packet arrival
  - assign timestamp $t_0$
  - append packet to buffer

- **Model Thread**
  - check buffer: read packet
  - post packet event to time $t_0$
  - process next event
  - model execution

**Real-time**
Interaction between threads

Listener Thread
- packet arrival
- assign timestamp t₀
- append packet to buffer

Model Thread
- control transfer
- check buffer: read packet
- post packet event to time t₀
- process next event
- model execution

real-time

Interaction between threads
Interaction between threads

Listener Thread
- packet arrival
- assign timestamp $t_0$
- append packet to buffer

Model Thread
- control transfer
- check buffer: read packet
- post packet event to time $t_0$
- process next event
- model execution

Inaccuracy

Packet arrival
assign timestamp $t_0$
append packet to buffer
control transfer
real-time

Packet arrival
assign timestamp $t_1$
control transfer
real-time
Interaction between threads

Listener Thread
- packet arrival
- assign timestamp $t_0$
- append packet to buffer

Model Thread
- packet arrival
- assign timestamp $t_1$
- Inaccuracy

Classification of inaccuracies
- Inaccuracies due to asynchronous packet arrivals
- Operating system imposed inaccuracies
- Model imposed inaccuracies

Careful examination of emulation error
Measurement Setup

- Constant rate UDP source
- Infinite server model with constant service time \( T_D = 10\text{ms} \)
- External measurement of delay error (Agilent Internet Advisor)
- Internal measurement of delay error (emulation self-determined error)
• Excellent correlation between measured and self-determined error
• Only 0.1% of all packets have an error of more than 0.2 ms
**HSDPA Scenario**

- Detailed model of High Speed Downlink Packet Access (HSDPA)
- one emulated UDP traffic flow in downlink direction
- one simulated cross-traffic flow with TCP bulk data transfer in downlink direction
• Only 1% of all packets have an error of more than 2-4 ms
• Higher absolute error due to model imposed inaccuracies

Evaluate relative error
- Very good relative error in low — medium load situations
- Only 0.1% of all packets have a relative error of more than 5%
Conclusion

- IKREmuLib: Integrated simulation and emulation environment
- Flexible usage through powerful filtering and routing possibilities
- Quick transition between simulation and emulation domain
- Efficiently combine simulated with emulated flows and components
- Good accuracy for models with delay on the order of tens of ms

Outlook

- Explore possibility for protocol interfaces
- Evaluate different strategies to enhance accuracy