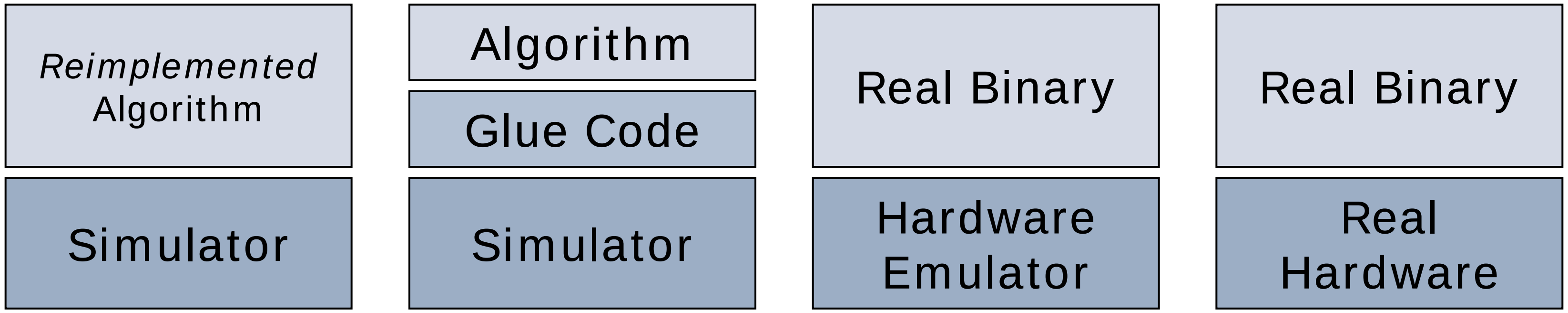
Software Integration in Simulation

Georg Wittenburg, Freie Universität Berlin

GI Research Seminar “Modeling Techniques for Computer Networks Simulation”, Dagstuhl, Germany

Comparison of Platforms

- How do we evaluate network protocols and algorithms?



Plain simulation:
- ns-2
- OMNet++
- OPNet
- SWAN
- ...

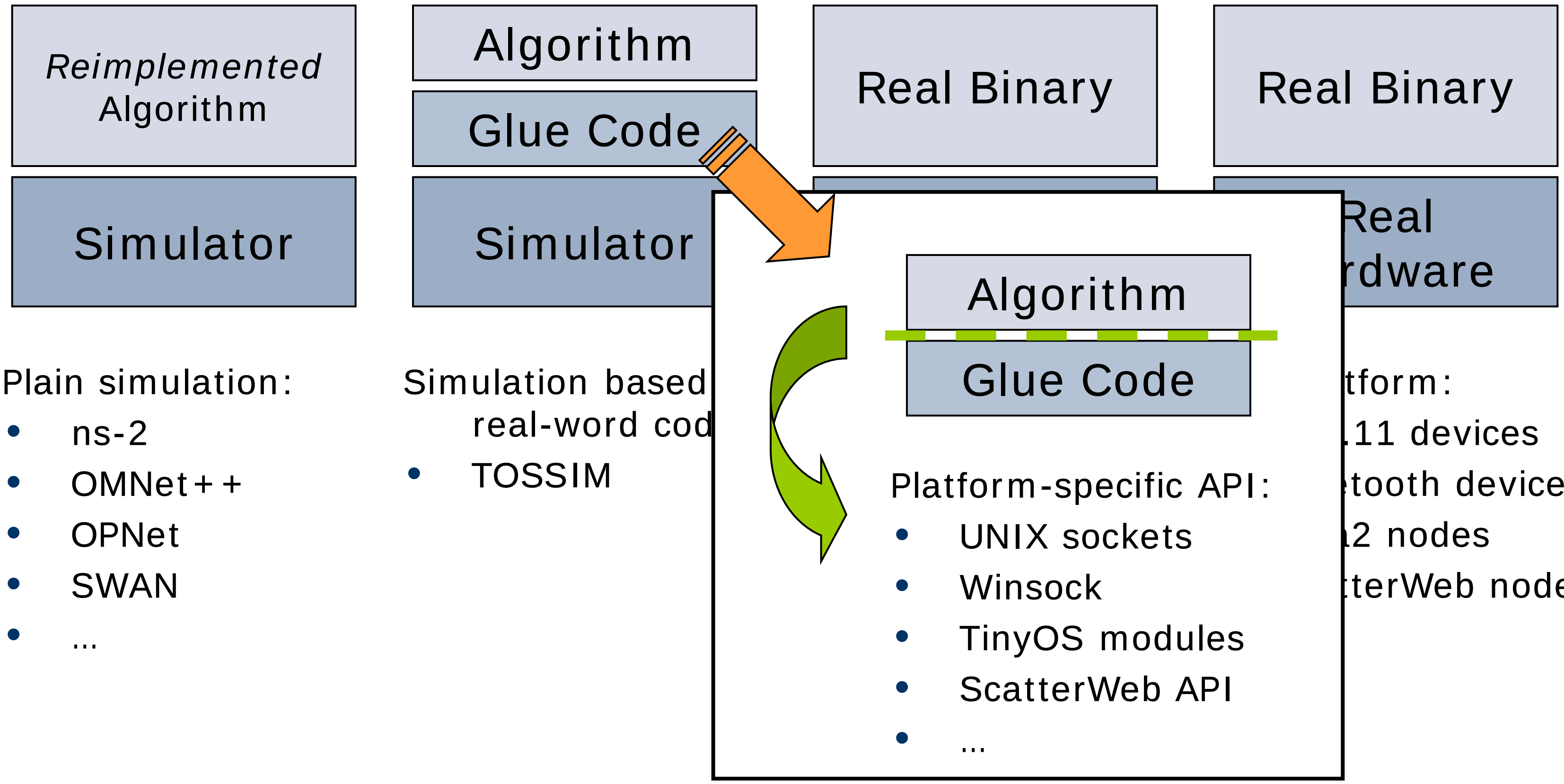
Simulation based on real-word code:
- TOSSIM

HW emulation using real binaries:
- Avrora
- VMNet

Real platform:
- 802.11 devices
- Bluetooth devices
- Mica2 nodes
- ScatterWeb nodes
- ...

Comparison of Platforms

- How do we evaluate network protocols and algorithms?



Plain simulation:
- ns-2
- OMNet++
- OPNet
- SWAN
- ...

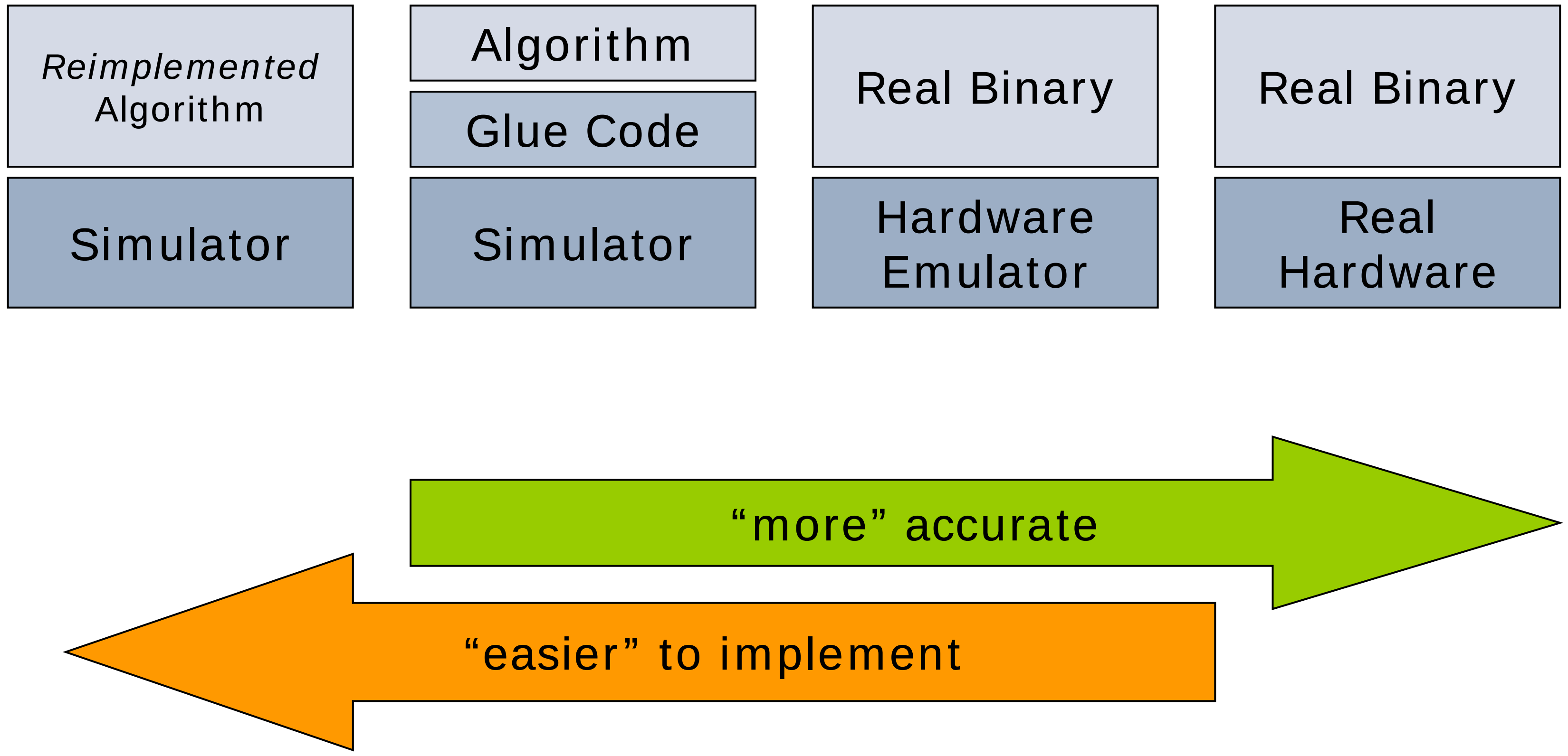
Simulation based
real-word cod
- TOSSIM

Platform-specific API:
- UNIX sockets
- Winsock
- TinyOS modules
- ScatterWeb API
- ...

tform:
.11 devices
etooth devices
a2 nodes
tterWeb nodes

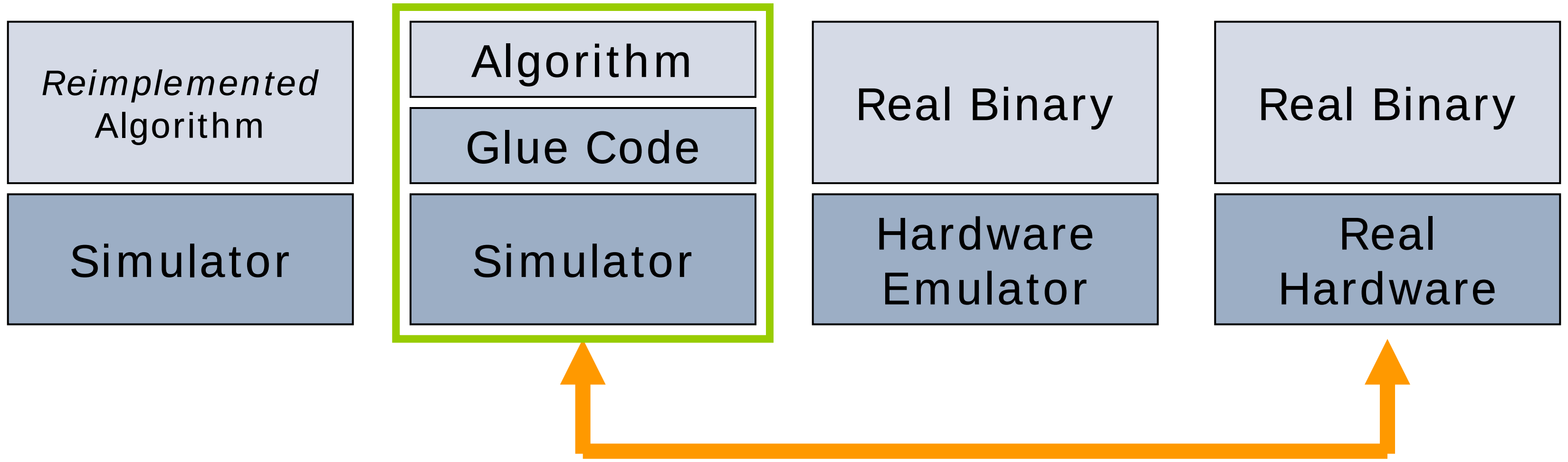
Comparison of Platforms

- Advantages / Disadvantages



Comparison of Platforms

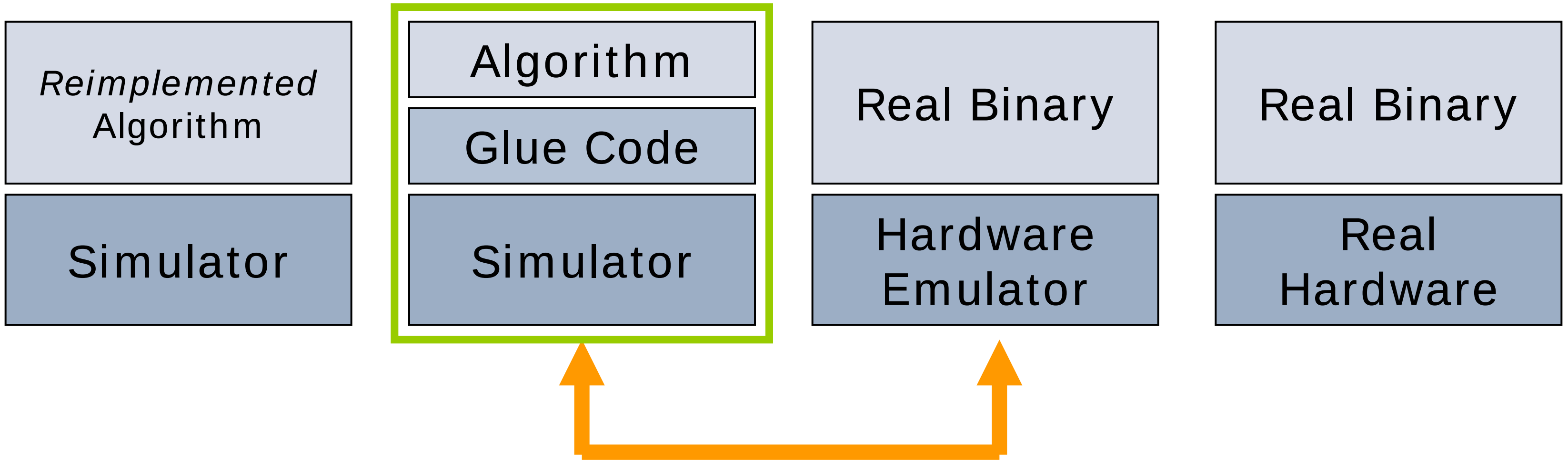
- Advantages / Disadvantages in Detail



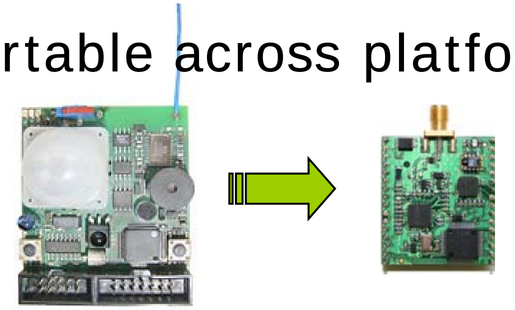
- Pro:
  - Faster development cycle
  - Algorithms first
  - Testing under reproducible conditions
- Contra:
  - Less accurate

Comparison of Platforms

- Advantages / Disadvantages in Detail

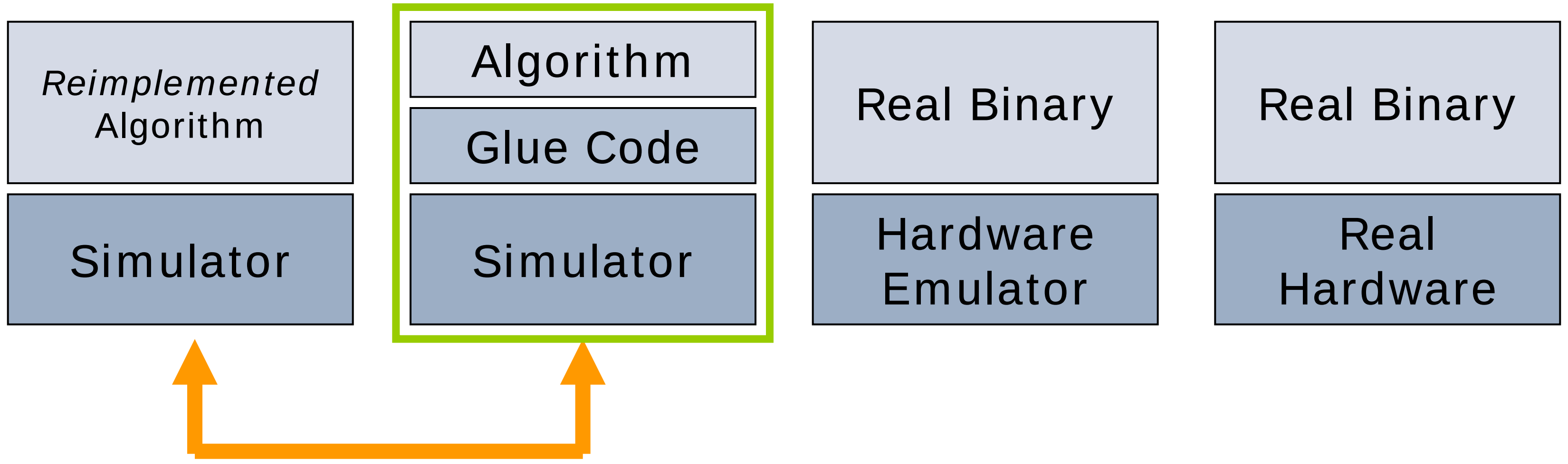


- Pro:
  - Portable across platforms.
- Contra:
  - Less accurate



Comparison of Platforms

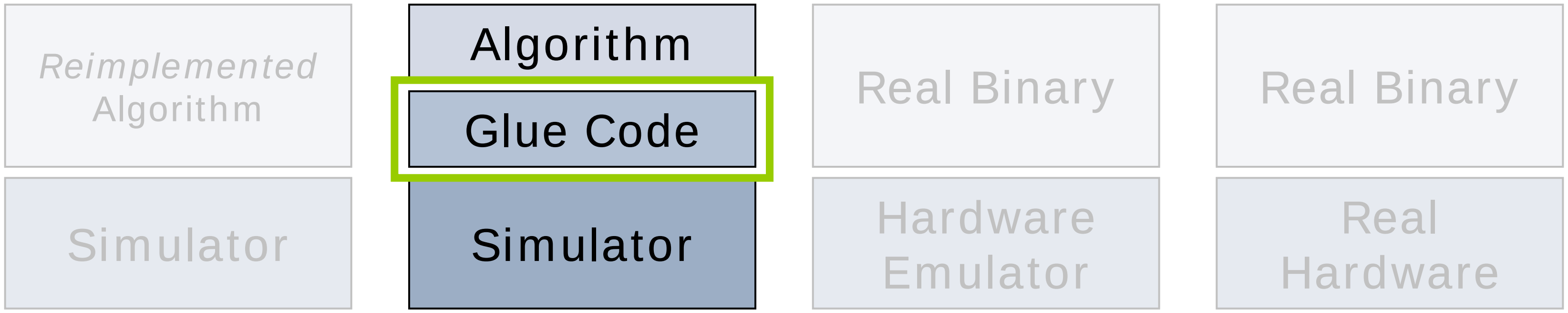
- Advantages / Disadvantages in Detail



- Pro:
  - No reimplementation
  - No programming inaccuracies
  - More realistic simulation
- Contra:
  - ?

Software Integration

- Simulating a real API and implementing algorithms on top ease the development and add credibility to results.



Key problems of integrating APIs and simulators:

1. Choice of API – Which API should be provided by the glue code?
2. Language adaptation – What if implementation language of API and simulator differ?
3. Concurrency / Isolation – How to make several instances of one algorithm run in the same address space?
4. Modeling lower layers – Which existing simulator components are best suited to support the API?

Choice of API

Which API should be provided by the glue code?

- Existing API:
  - Examples: UNIX sockets, Winsock, TinyOS modules, ScatterWeb API
  - Pro: No need to change existing code base, easy for new developers
  - Con: Usually complex functionality, subtle semantics, lots of work
- Newly developed API:
  - For most projects only minimal functionality is required:
    - Packet sending and reception; address handling
    - Timers / callbacks
  - Pro: Clear semantics, reasonable work, portability (?)
  - Con: Slight learning effort required

Language Adaptation

What if implementation language of API and simulator differ?

- Common scenario:
  - API in C, e.g. operating systems, embedded development
  - Simulator in C++/Java
- Solution generally depends on language combination:
  - C API / C++ simulator: C is subset of C++
  - C API / Java simulator: External library; native interface wrappers

➢ May lead to concurrency / isolation issues...

Concurrency and Isolation Issues

How to make several instances of one algorithm run in the same address space?

- Why not use existing process/thread abstraction?
  - Hard to integrate with control flow of the simulator
  - Requires either
    - fine-grained synchronization
    - assume real-time execution, i.e. move to emulation
  - Problematic with scalability and debugging

- Approaches generally redirect access to global variables:
  - At compile time by automatic code adaptation
  - At runtime time by swapping memory regions
- Usually closely related to language adaptation

Modeling Lower Layers

Which simulator components are best suited to support the API?

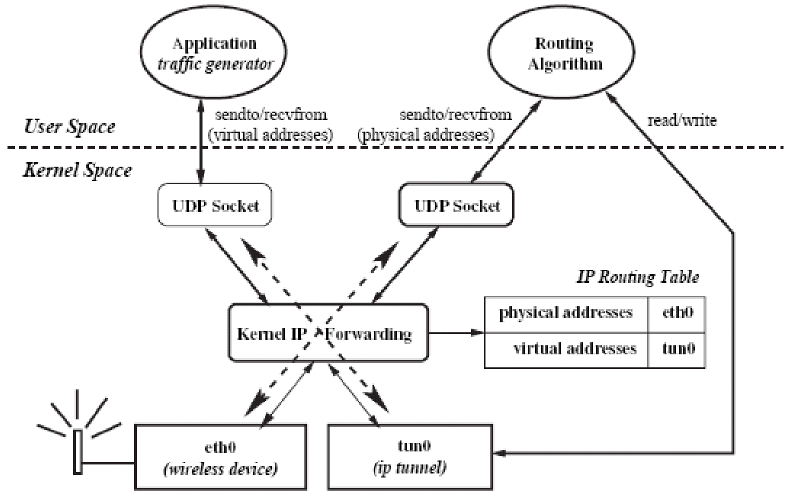
- Relevant if API is available on real-world system
- Depends on two factors:
  - Level of abstraction, i.e. high-level vs. low-level API
  - Properties of the HW platform
    - ➢ Data sheets, own measurements
- Trade-off between simulation accuracy and scalability
  - ➢ Try more accurate models first, checks how bad things really get

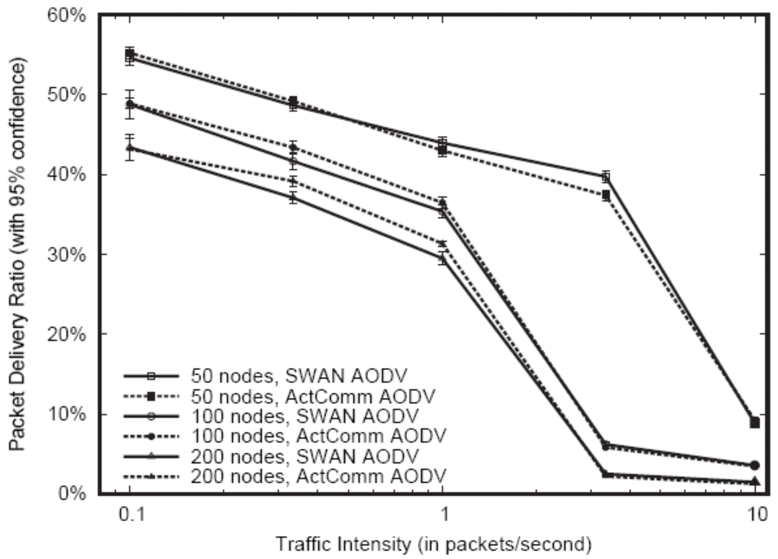
➢ Verification very easy if platform available!

Approaches to “Glue Code”

- Direct execution on SWAN simulator
  - Source code adaptation, reimplementation of socket and timing API
  - Comparison of real (user-space) and simulated protocols

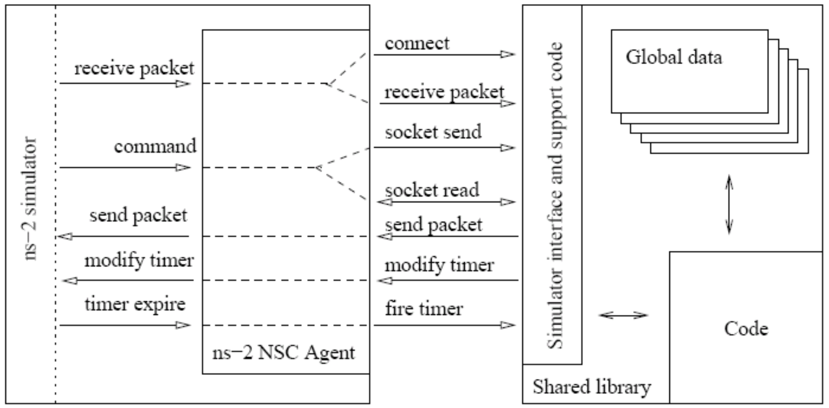
[Jason Liu, Yougu Yuan, David M. Nicol, Robert S. Gray, Calvin C. Newport, David Kotz, and Luiz Felipe Perrone. Empirical Validation of Wireless Models in Simulations of Ad Hoc Routing Protocols. Simulation: Transactions of The Society for Modeling and Simulation International, 81(4):307-323, April 2005.]

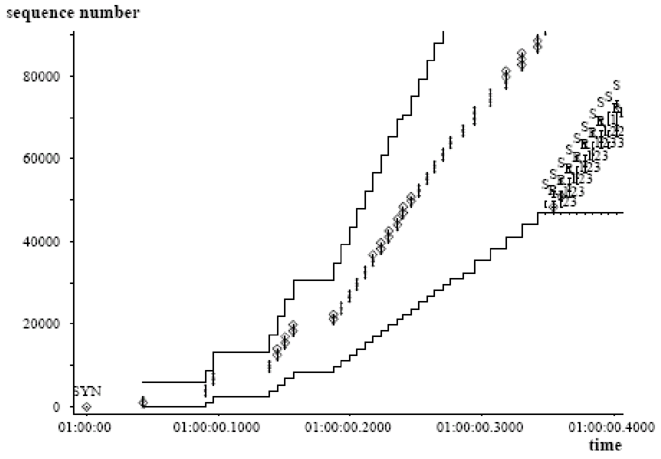




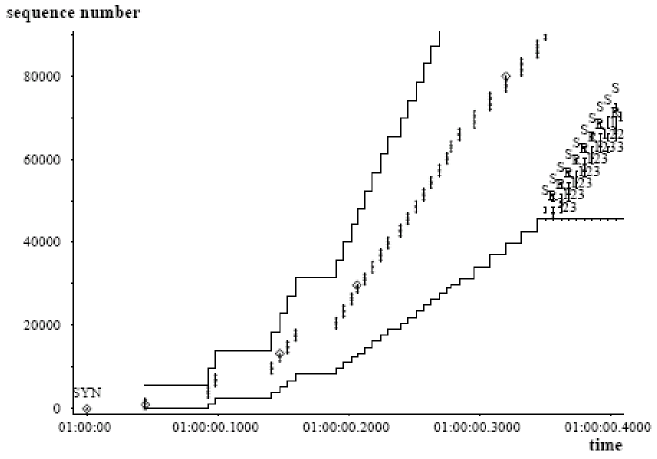
Approaches to “Glue Code”

- Network Simulation Cradle
  - Integration of kernel-space protocol stacks into ns-2
  - Preprocessor, shared library
- http://research.wand.net.nz/software/nsc.php





(c) Simulated Linux



(d) Measured Linux

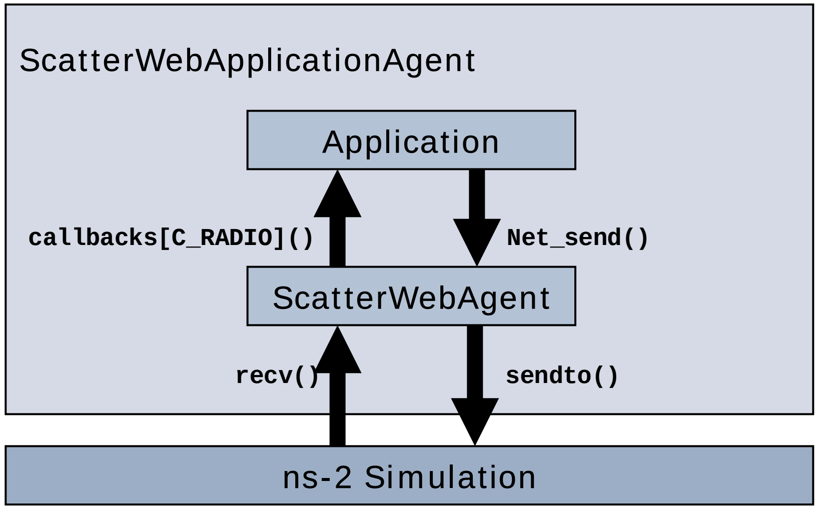
[Sam Jansen and Anthony McGregor. Validation of simulated real world TCP stacks. In Proceedings of the Winter Simulation Conference, Washington, USA, 2007.]

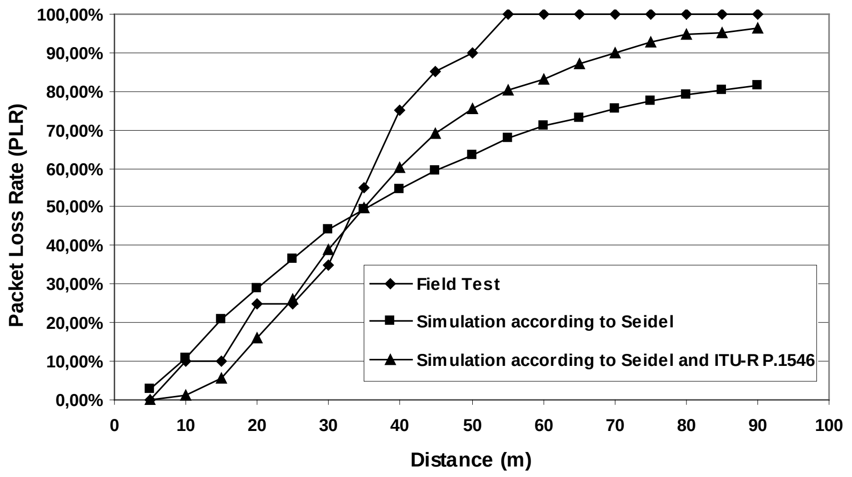
Approaches to “Glue Code”

- ScatterWeb on ns-2
  - Re-implementation of low-level functions in ns-2
  - C++ wrapper to transparently link C code
  - Implementation support, validation of protocols (e.g. directed diffusion), simulation accuracy

- http://cst.mi.fu-berlin.de/projects/ScatterWeb/software/ns2.html

[Georg Wittenburg and Jochen Schiller. Running Real-World Software on Simulated Wireless Sensor Nodes. In Proceedings of the ACM Workshop on Real-World Wireless Sensor Networks (REALWSN '06), pages 7-11, Uppsala, Sweden, June 2006.]





Conclusion

Major points for the reader:

- Abstraction layer (API) between your code and simulator adds flexibility to your research project.
- Be aware of the trade-offs involving design and implementation of the abstraction layer (“glue code”).
- Consider adapting existing approaches.

➢ Never write code for any particular simulator!

More Approaches to “Glue Code”

- GEA
  - André Herms and Daniel Mahrenholz. Unified Development and Deployment of Network Protocols. In Proceedings of MeshNets 2005, Visegard/Budapest, Hungary, July 2005.
- COOJA
  - http://www.sics.se/~fros/osterlind06crosslevel.pdf
- TOSSIM
  - http://www.cs.berkeley.edu/~pal/research/tossim.html
- Liu et al. WiDS @ NSDI ’07 (?)