# Approaches for Evaluating the Application Performance of Future Mobile Networks

## ITG 5.2.4 — July 2011 — Aachen

Thomas Werthmann, Matthias Kaschub,
Christian Blankenhorn, and Christian M. Mueller
thomas.werthmann@ikr.uni-stuttgart.de

Universität Stuttgart
Institute of Communication Networks
and Computer Engineering (IKR)
Prof. Dr.-Ing. Andreas Kirstädter

# Outline

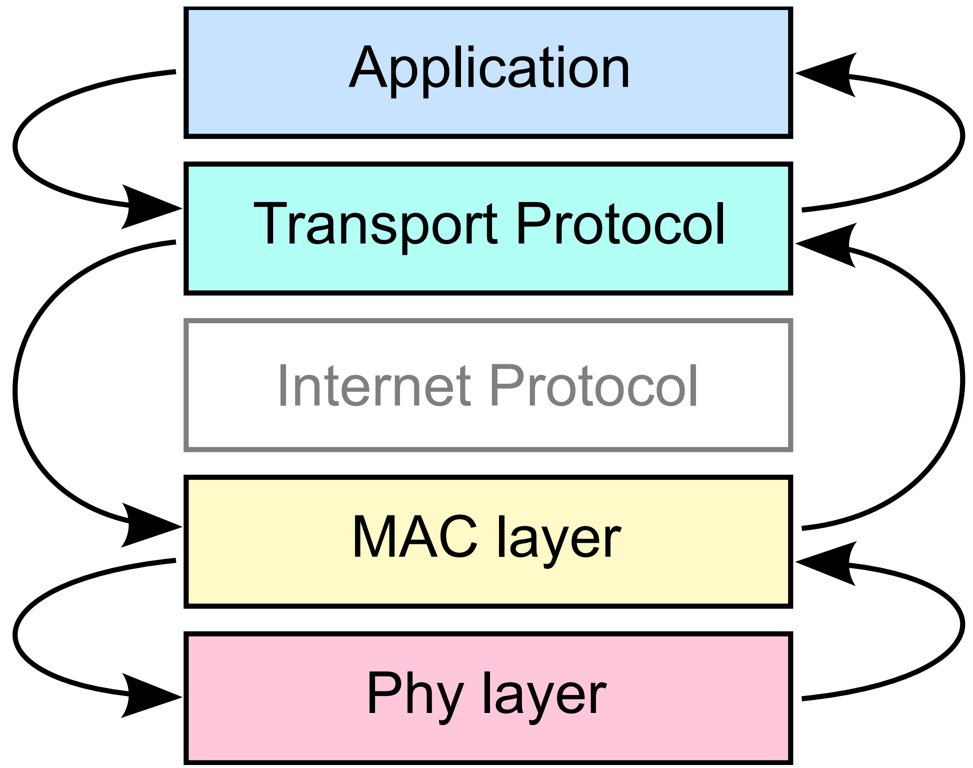
## Motivation & Problem Statement

## Approaches

## Our Implementation

## Conclusion

# Motivation




## Subject of evaluation

Algorithms and techniques in the Phy and MAC layers

## Metric

- Achievable application performance
- Difficult to derive from average delay & throughput

→ Models required for evaluation

## Feedback to lower layers

- Object sizes transmitted by applications
- Effects from parallel TCP connections
- Queues running empty

see also: Muhammad Amir Mehmood, Cigdem Sengul, Nadi Sarrar and Anja Feldmann, 2011, Understanding Cross-Layer Effects on Quality of Experience for Video over NGMN

C. M. Mueller, 2011, Analysis of interactions between Internet data traffic characteristics and Coordinated Multipoint transmission schemes

# Problem

**Cross-Layer evaluation required**

How to bring network simulation
and real world protocols / applications together?

# Approaches (1/3)

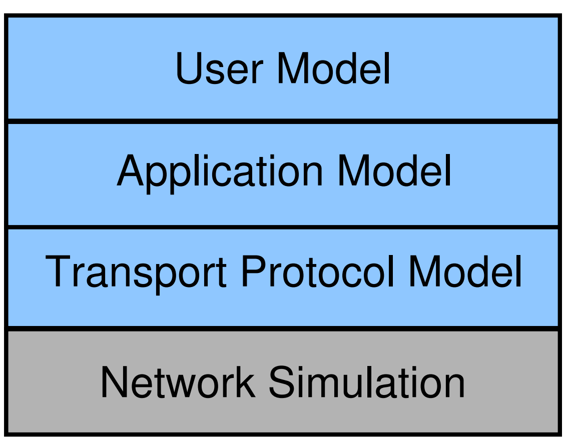
*Model all components*

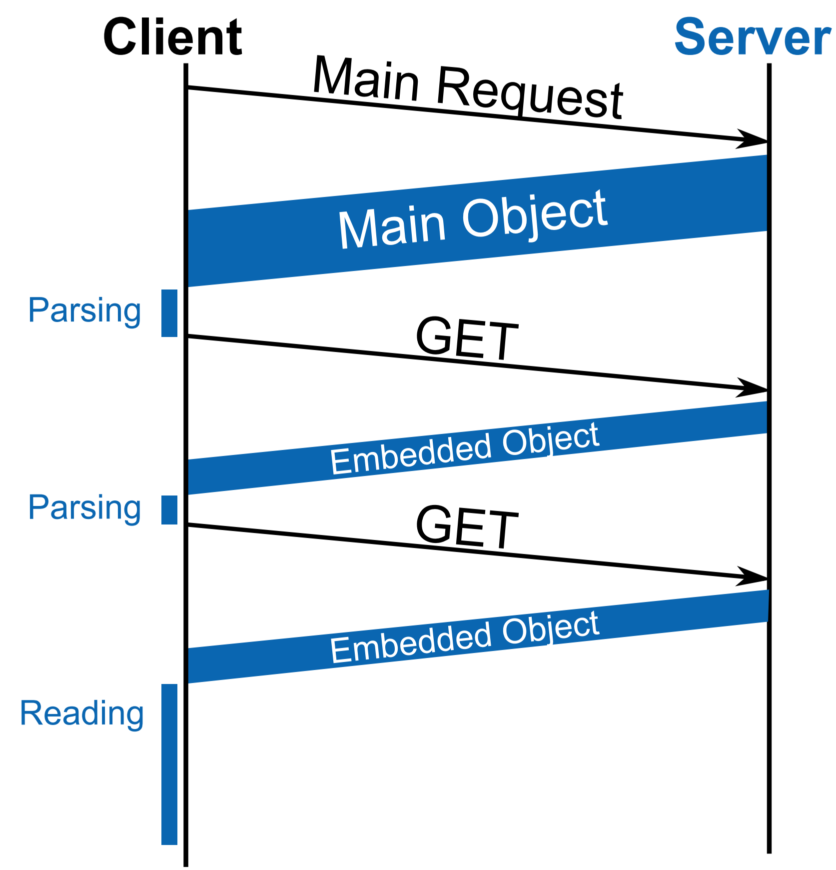
### Simulation models for Applications and Transport Protocols

- TCP models:
  various implementations
- Application moels:
  e.g. NGMN web model (on the right)

→ complicated algorithms

→ models usually simplified — still realistic?



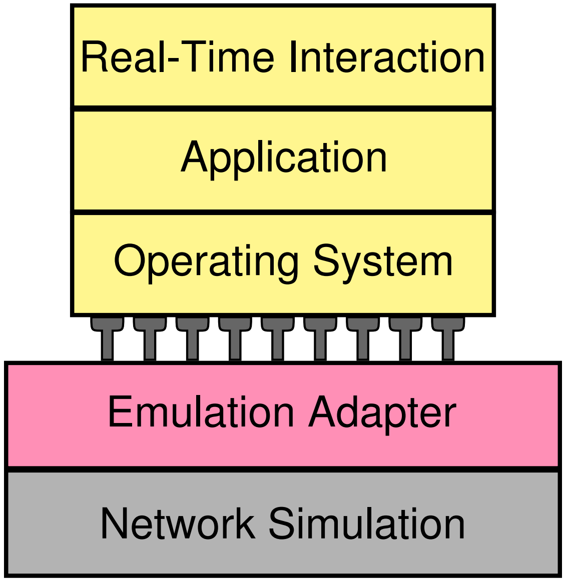
# Approaches (2/3)

*Bring the network simulation into a lab setup*

## Real-Time Emulation

- Build up lab setup with real computers
- Connect real network devices to simulated network
- Optionally communicate with the real internet

→ Requires fast (abstract) simulation models




## Slowed Emulation

- Decelerate the computers' clock speed to gain time for emulation

→ Requires special setup & synchronization

# Approaches (3/3)

*Bring real code into the Simulation*

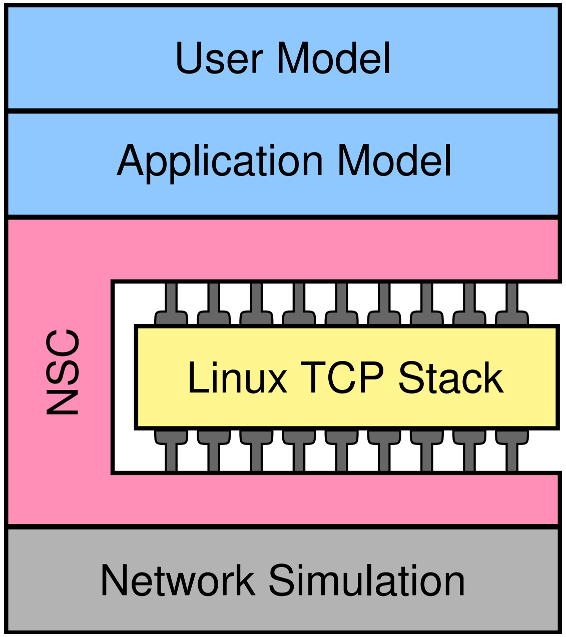
**Example: Network simulation cradle**

- Linux kernel code is modified
  by partly automated scripts
- TCP stack can be loaded as shared library into ns-2
- Clocks of the kernel are driven by the simulation

→ Adapts to the speed of the simulation

→ Authentic protocol behavior

→ The chosen interface makes it difficult
  to port a new linux kernel



# Our Approach

### Existing IKR Simulation Ecosystem

- IKR SimLib → fixed network simulation
- IKR RadioLib → radio transmissions (LTE)
- IKR EmuLib → real time emulation
- IKR nscadapter → wrapper for the NSC
- **QEMU simulation adapter**

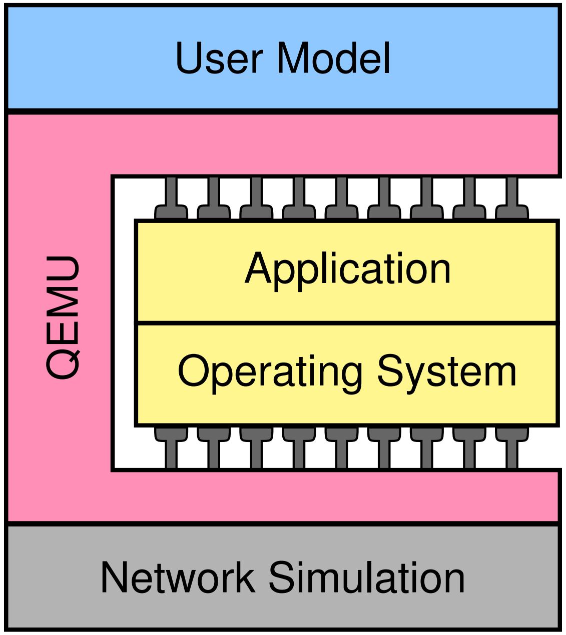
### Main Idea

- Use Computer Virtualization as interface between simulation and real code
- Let virtual clock be driven by the simulation

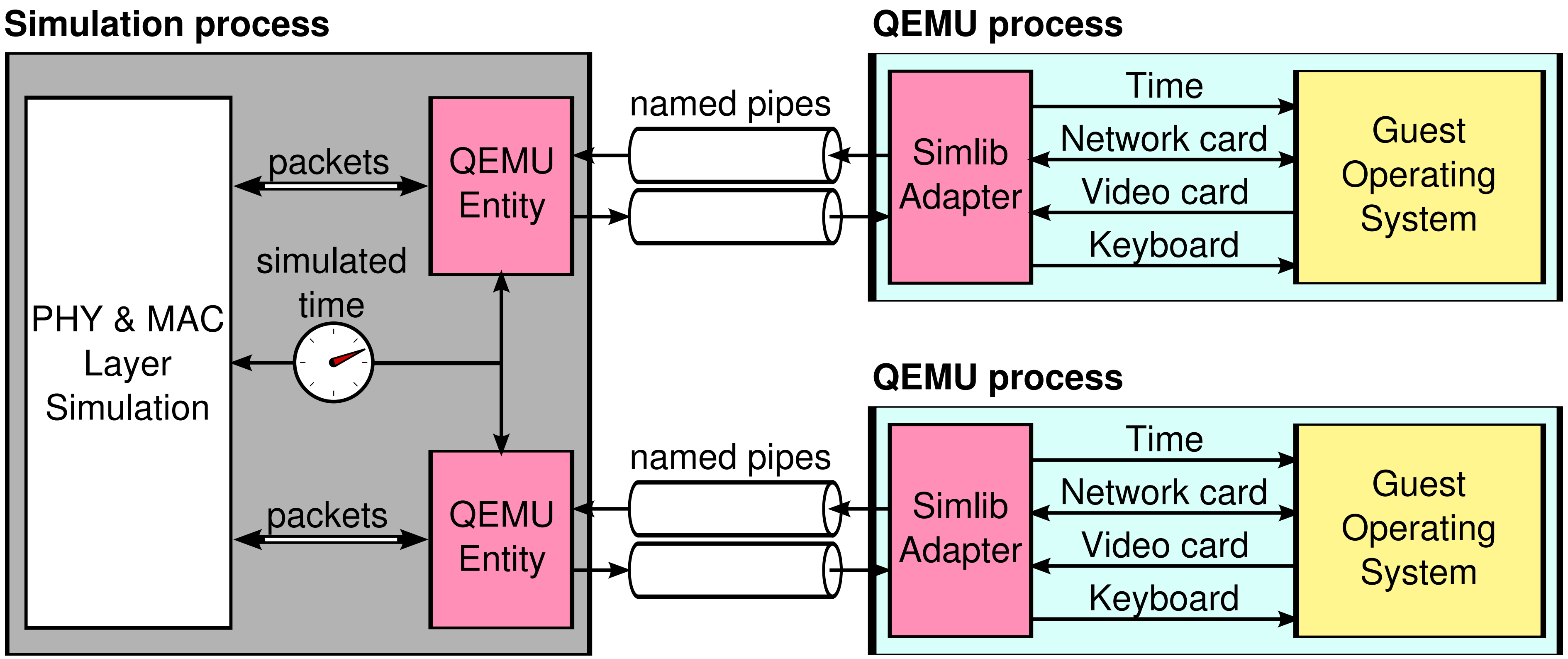
### Our ambition

- Easy handling (hundreds of simulations)
- No special requirements for hardware etc.

→ **Possible to perform simulations on a standard computer cluster**



# Architecture




→ **Independent processes communicating via pipes**

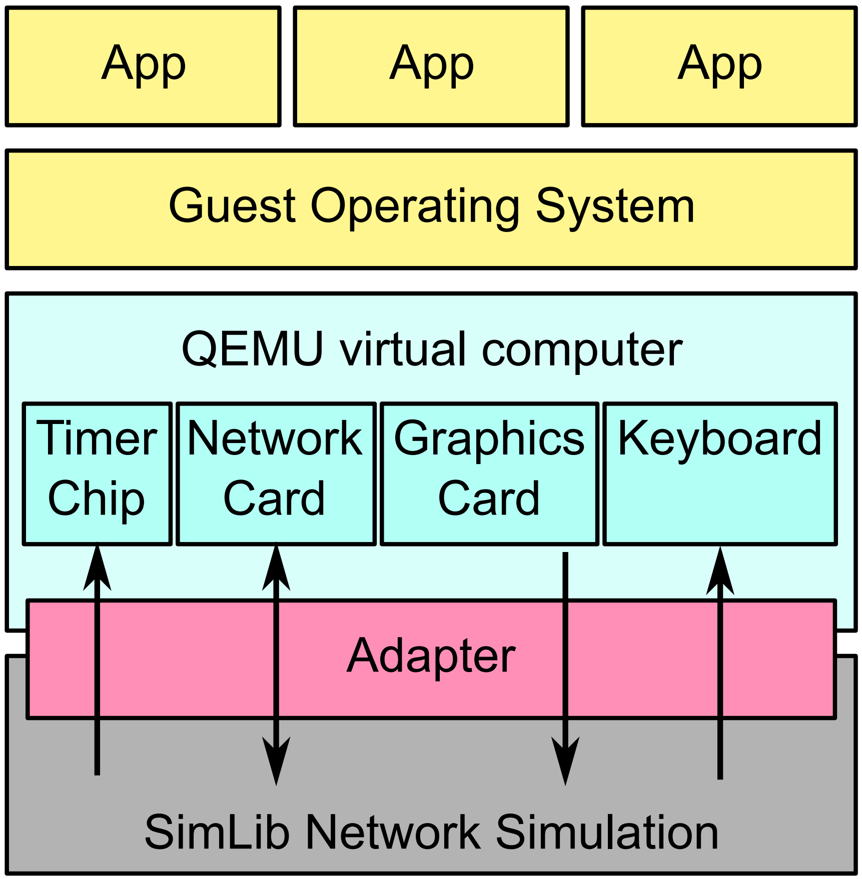
# QEMU as interface between Simulation and OS

## Virtual Computer as Interface

Operating system and applications do only see the virtual computer

→ No modifications required

→ Easy to install new applications / OSs




## Modifications inside QEMU

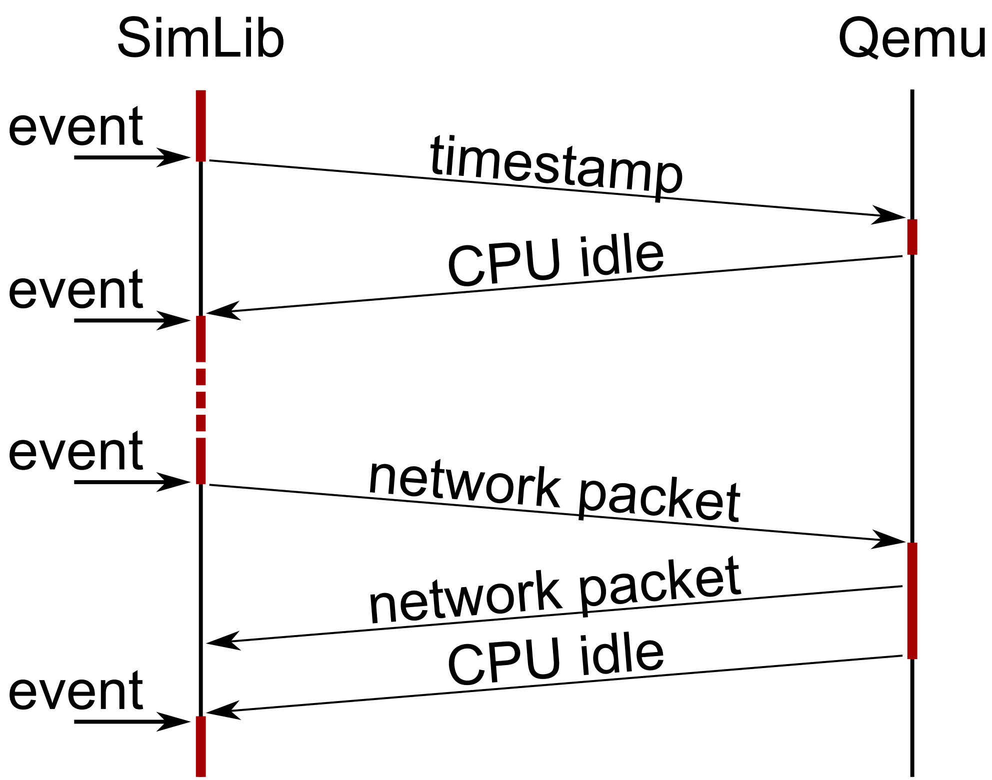
- Layered architecture: QEMU consists of device emulations and backends
- Our adapter provides additional backends, emulation layer remains unchanged
- We have modified the QEMU main loop to support a shared control flow

# Control Flow

## Interaction of SimLib and QEMU

- Either the simulation or one single QEMU instance executes at a time
- Timer events correspond to events in the simulation calendar
- Calculations inside the virtual computer are performed in zero simulated time

  → follows paradigm of event-driven simulation
- Virtual computer can spend nearly infinite time for computations




→ **Virtual computer is not restricted by host CPU power**

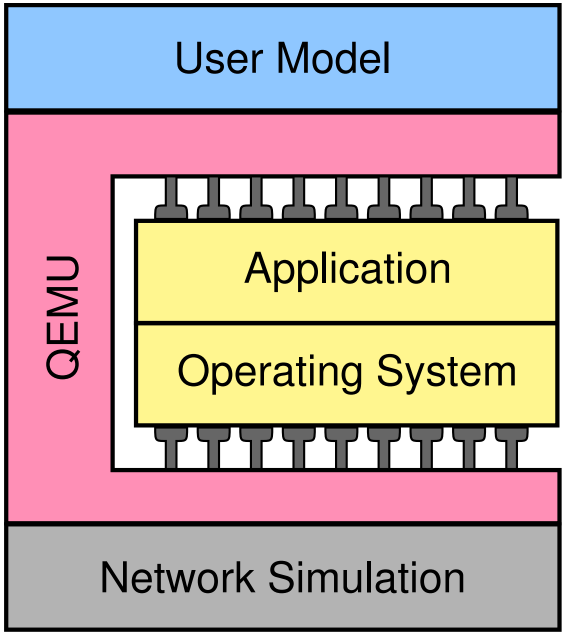
→ **Strictly synchronous interaction**

# User Model and Metrics

## User Models

- Interactive use not desired and not possible

→ Automatic user models required
(e.g. stochastic reading time)




## Subjective Metrics

- Record and play back screencast
- Compare different parametrizations

## Objective Metrics

- Automatically analyze screenshots
(e.g. determine if pixels change)
- Investigate network packets
(e.g. first and last packet of a TCP session)
- Modify applications to print their state
in a machine-readable format

# Scalability

## Memory

- Overhead of emulator and adapter: negligible
- Small operating system without GUI:
  32MB per instance
- Modern operating system with graphical applications:
  at least **512MB per instance**
- Requirements can possibly be reduced
  with Linux Kernel Samepage Merging (KSM)

## Processing Power

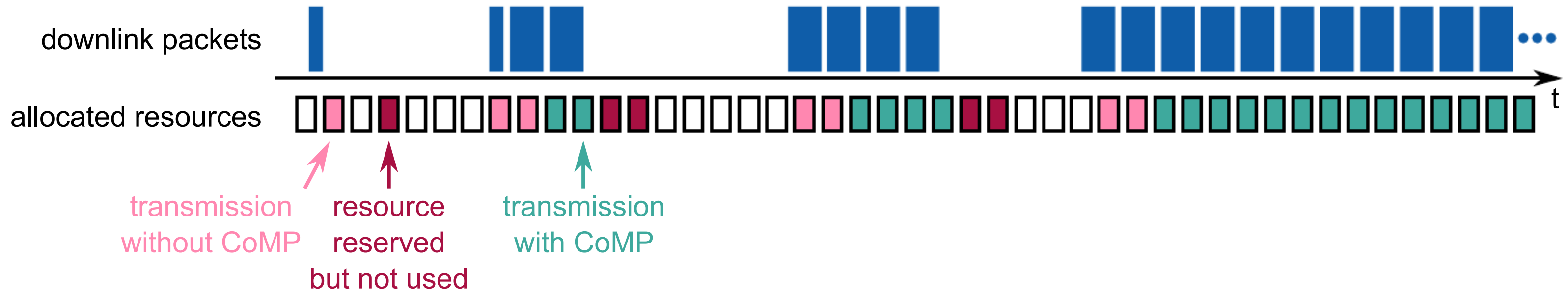
- High CPU load: User space QEMU about factor 10 slower than host computer
  - Example: booting Ubuntu Maverick takes about 15 minutes
- Low CPU load: About factor 10 faster than real time
- Typically no CPU-intensive applications on the virtual computer
- Simulated time spans have to be large to capture the upper layer effects

→ **Complex Phy models become the limiting factor**

# Usage Scenarios (1/2)

## Evaluation of a coordinated scheduling algorithm

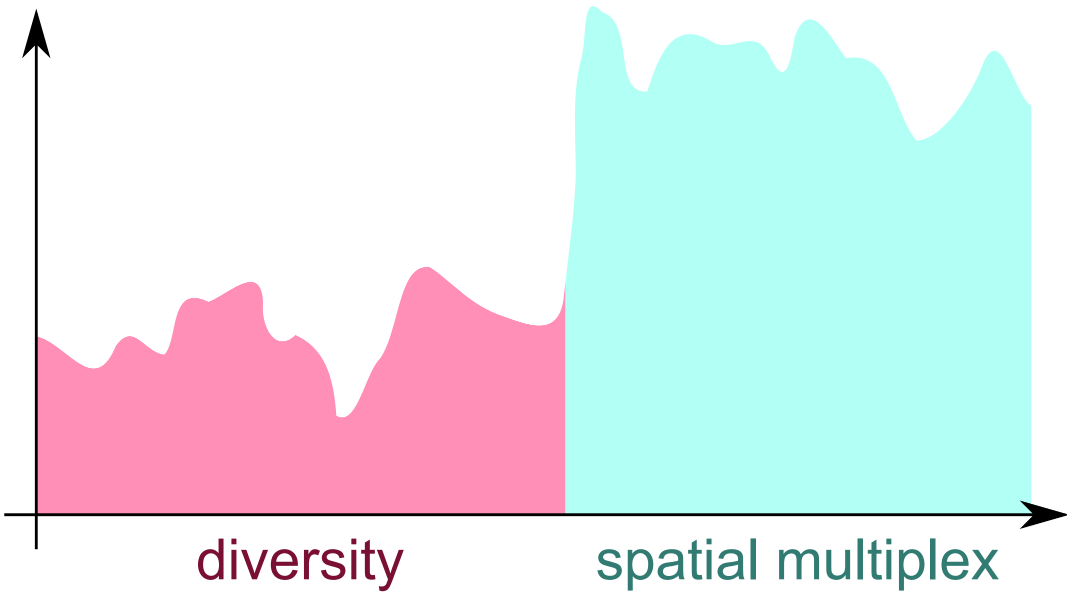
- Focus: Interactions between coordination and TCP control loops
- Abstract Phy model
- Application: Simplified → TCP downloads only
- Stochastic models for reading times and web object sizes (including heavy tail)
- Metric: Object finish times, miscoordinated frames
- Simulated time: 2 to 8 hours



# Usage Scenarios (2/2)

**Link layer evaluation of a complex MIMO scheme**

- Focus: Influence of setup time on application performance
- Moderately abstract Phy model
- Application: Web browser loading a single web page
- Metric: Time it needs to load the web page
- Multiple drops to achieve statistically valid results
- Simulated time: 15 seconds



# Summary & Conclusion

## Summary

- Cross layer evaluation required
- Modelling of all effects is difficult
- Using real code is often easier
- Presented architecture allows to use OSs and applications without modification

## Conclusion

- Easy to try new applications — just install on the virtual computers
- Easy to use new kernel versions
- Also nice for demonstration!
- Abstract Phy models still required because of long simulation time spans