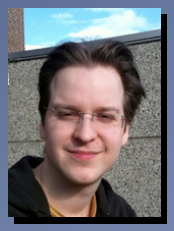
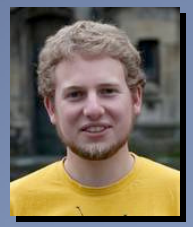
SliceTime

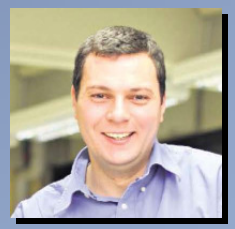
A platform for accurate and scalable network emulation


Elias Weingärtner


Florian Schmidt


Hendrik vom Lehn


Tobias Heer

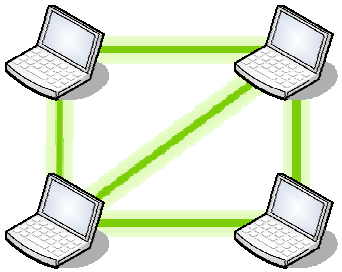

Klaus Wehrle

http://comsys.rwth-aachen.de/

Aachen, 14.7.2011

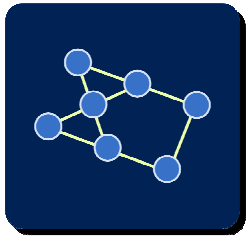
Motivation

How to evaluate networking software at large scale?



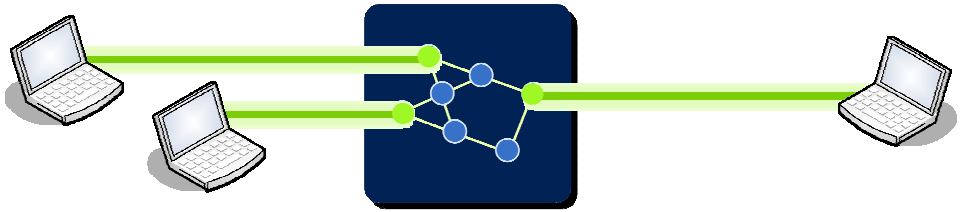
Network Testbeds

Drawbacks: Scalability & Cost



Network Simulation

Models instead of software, no operating system…

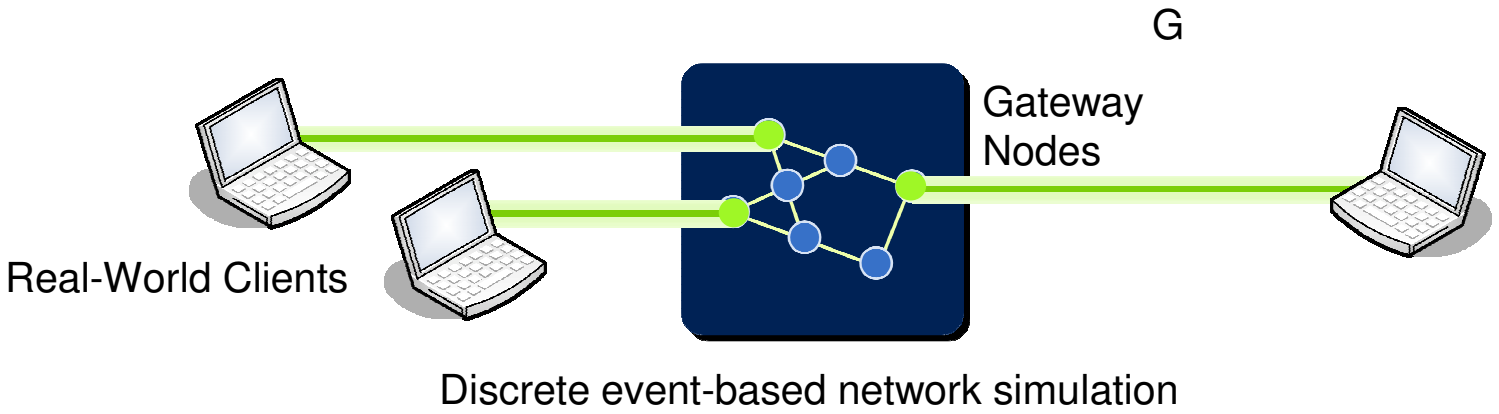


Network Emulation

Requires real… simulations

SliceTime solves this problem

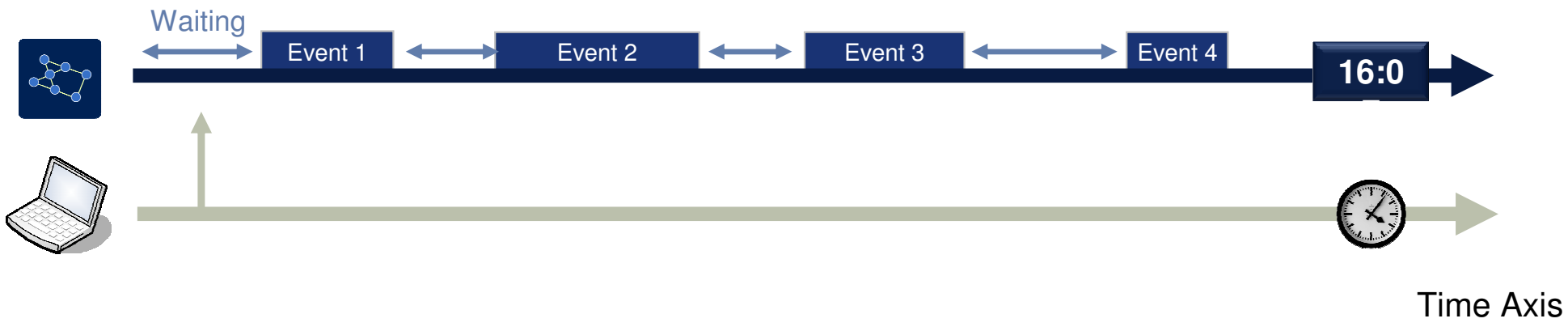
Network Emulation



- **Real-World clients**
  - Execute communications software & operating system
- **Discrete event-based network simulator**
  - Models interconnecting network
  - Examples: ns-2, ns-3, OMNeT++
  - Also provides simulated hosts → scalability
  - Simulated environment: virtual mobility, radio propagation…

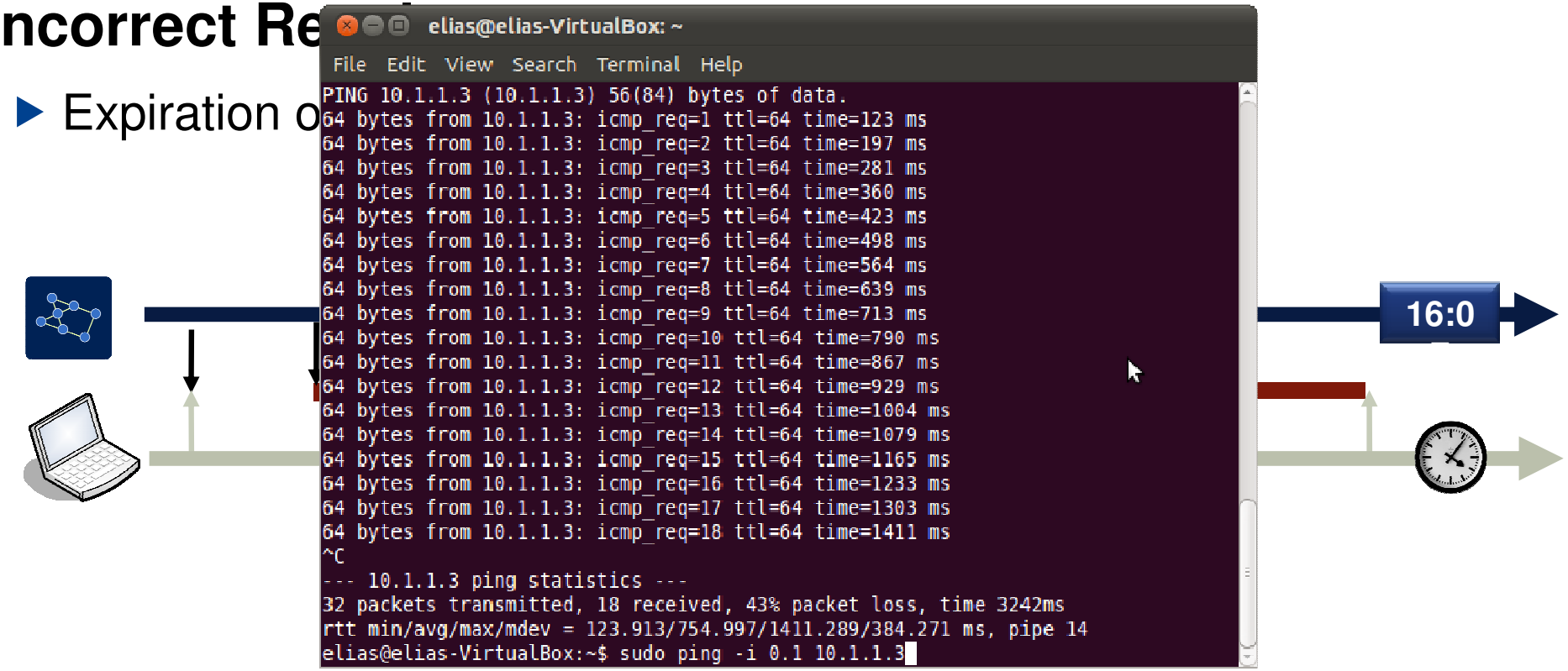
Network Emulation: Timing

- **Different timing concepts**
  - Network simulation: series of discrete events
  - Real-world clients: continuous wall-clock time
- **Current common solution**
  - Pin simulation events to wall-clock time
  - Wait between events



Time Drifting Issue

- **Problem: Many Simulations are not real-time capable**
  - Computationally complex models
  - Many simulated nodes
- **Simulation is overloaded → time drift**
- **Incorrect Re**
  - Expiration o

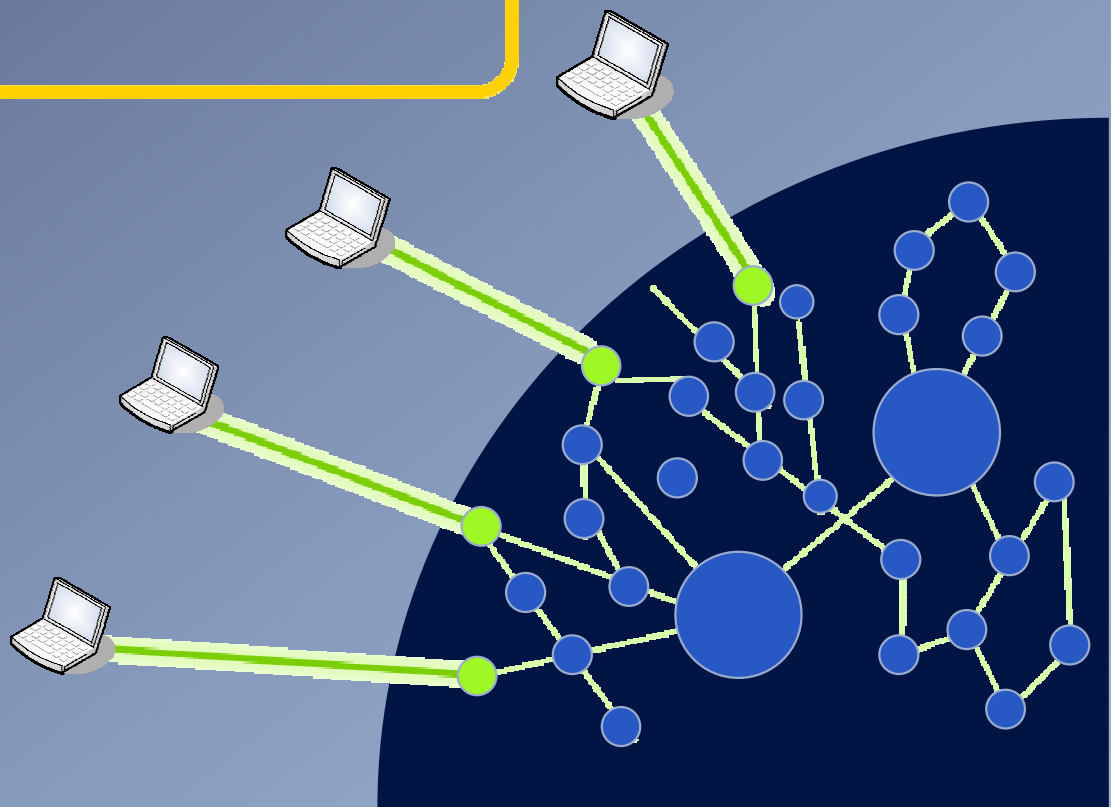


How can time drifting be prevented to enable large-scale and complex network emulation scenarios?

Two options:

1. Make the simulation fast enough

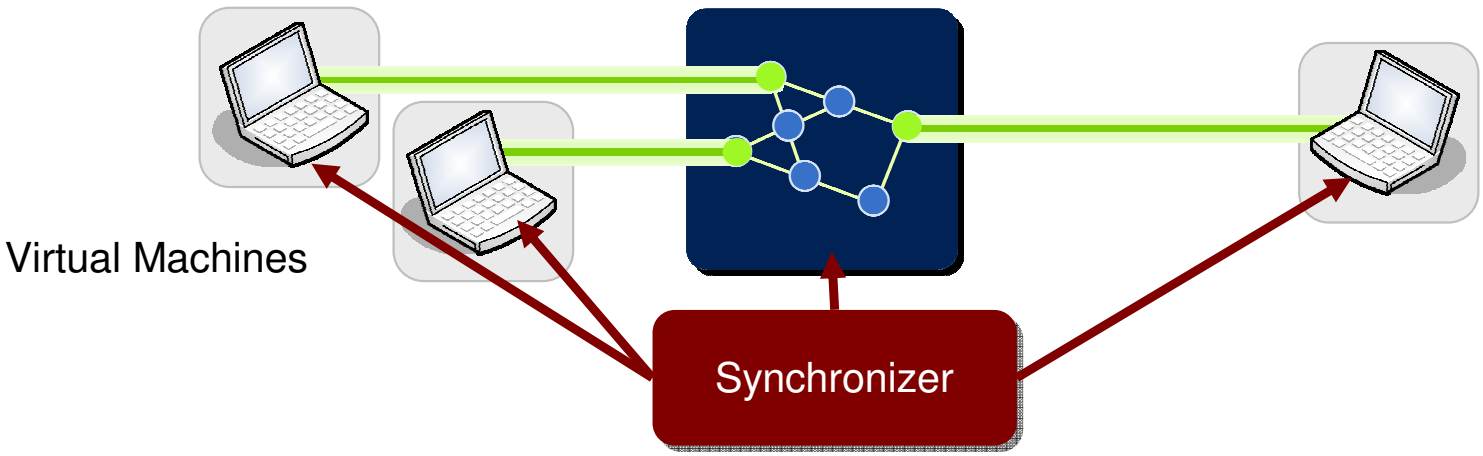
2. Slow down the real clients to match the simulation's speed



Requirements for SliceTime

1. **Tight synchronization of clients and simulation**
   - Limit drifting to 1ms or less (for WAN scenarios)
2. **Need to slow down real-world software clients**
   - Unmodified communications software
   - Legacy operating systems (Linux or Windows)
   - Slow down must be transparent to the clients
     → provision of virtual time
3. **Little overhead due to synchronization**
   - Additional run-time
   - Additional delays or measurement artifacts

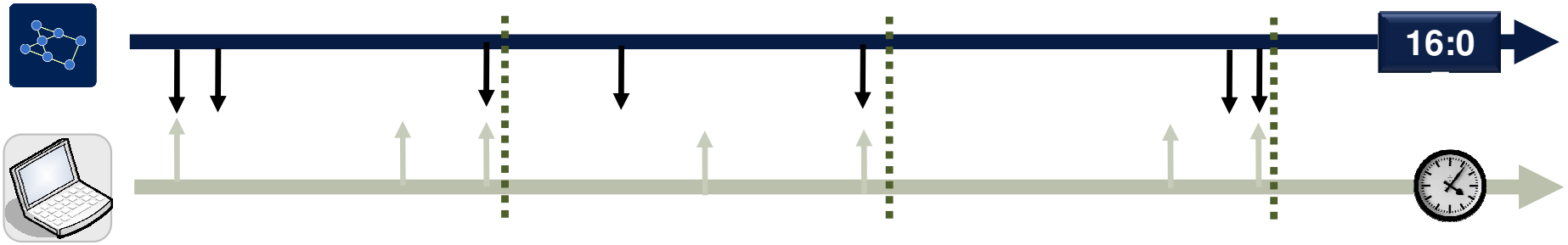
SliceTime: A Synchronized Network Emulation platform



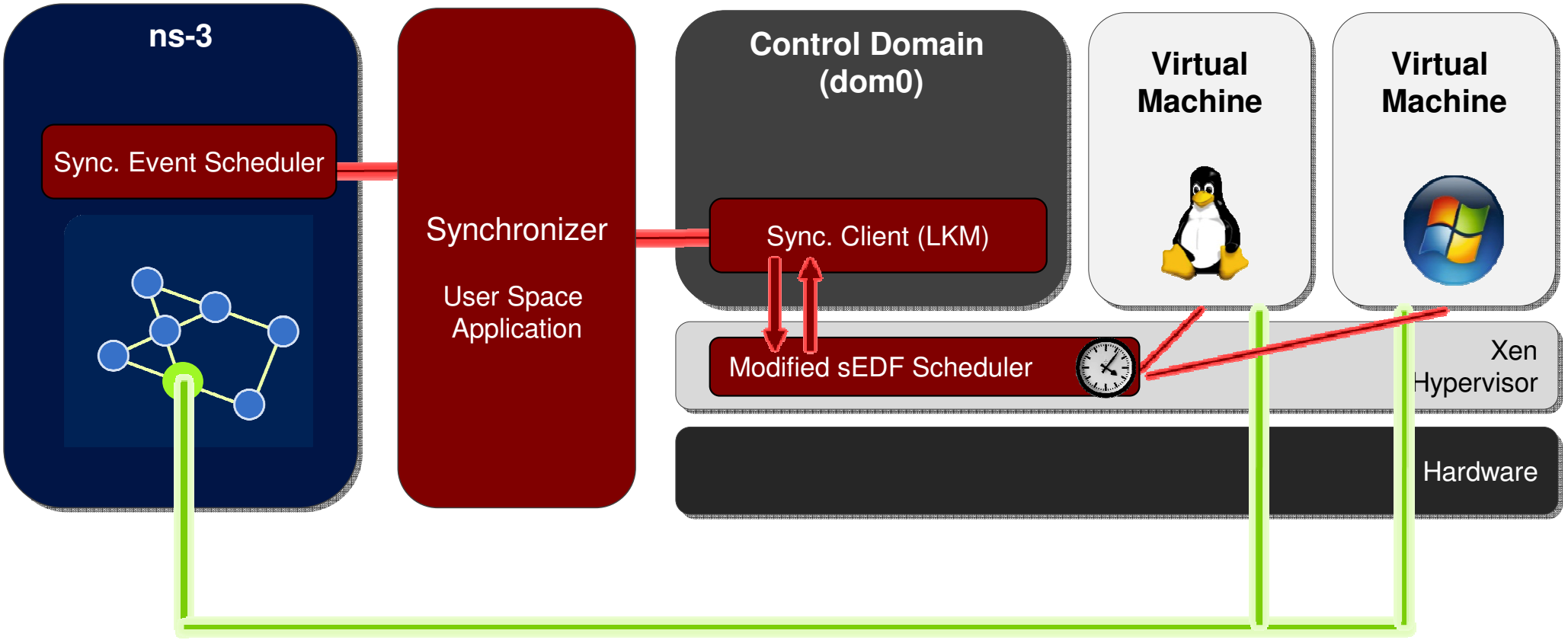
- **Synchronizer**
  - Synchronization algorithm aligns execution of clients and simulation
- **Virtual machines provide needed level of control**
  - Control over run-time behavior
  - Full control over system context/timers → provision of virtual continuous time

Synchronization Algorithm

- **Goal: Limit time drifting**
  - No assumptions about future run-time behavior
  - No snapshotting & rollbacks
- **Barrier Algorithm**
  - Assign slices of run-time
  - Blocking at end of time slice
  - Clients notify synchronizer after they have finished
- **Synchronization accuracy corresponds to time slice size**



SliceTime Implementation



Data Communication Flow
- **Tunneled EtherNet Frames**
- **802.11 Frame Tunnel**

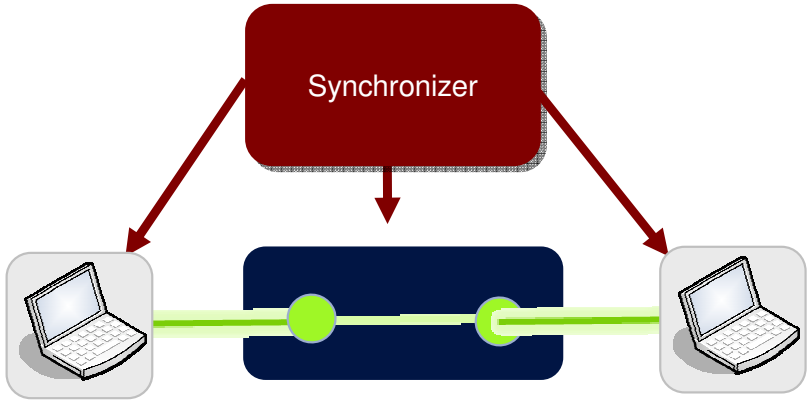
Evaluation

How accurate is SliceTime?

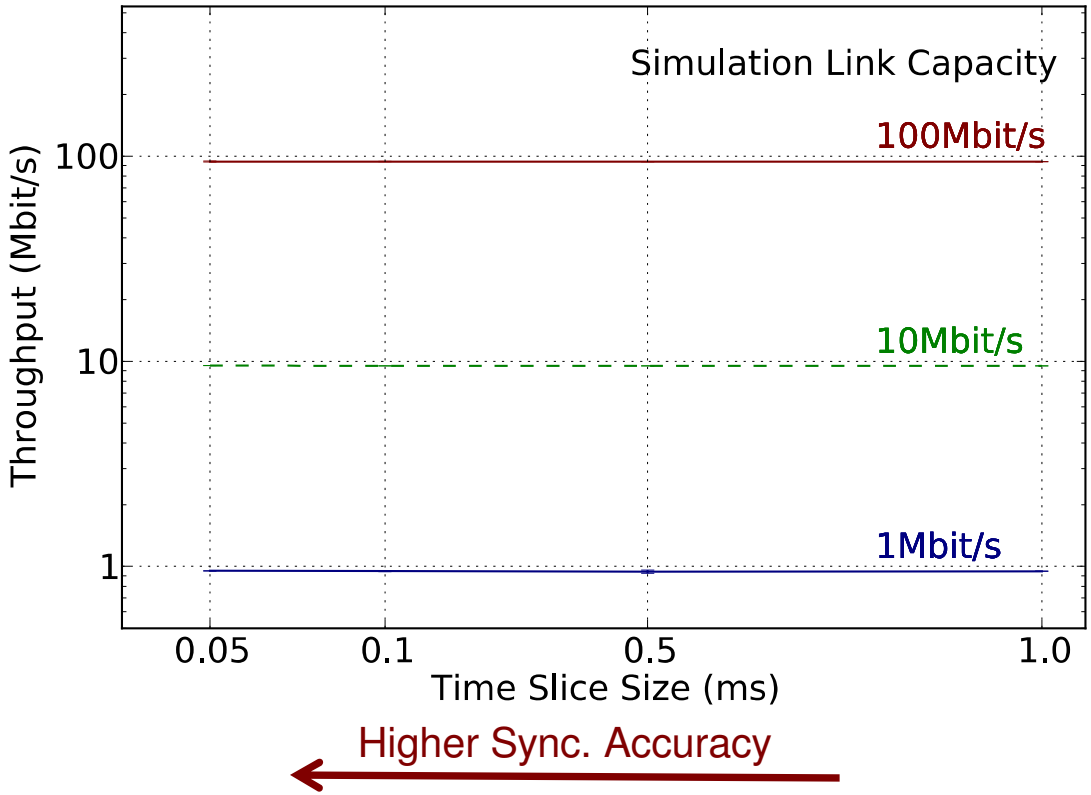
How much overhead is caused by the synchronization?

Network Throughput

How is network throughput affected by time slice size?



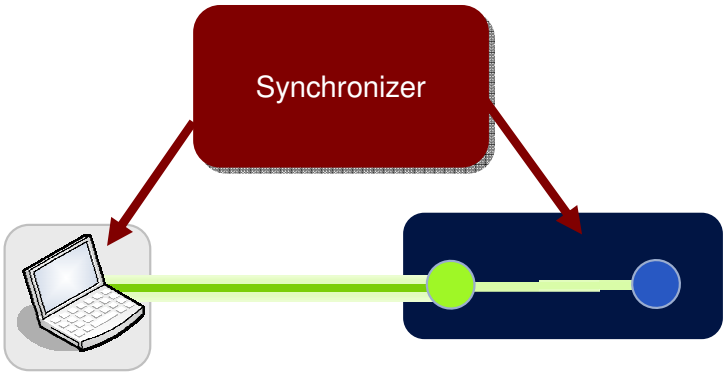
Measurement: netperf TCP_STREAM benchmark
Different levels of sync. accuracy



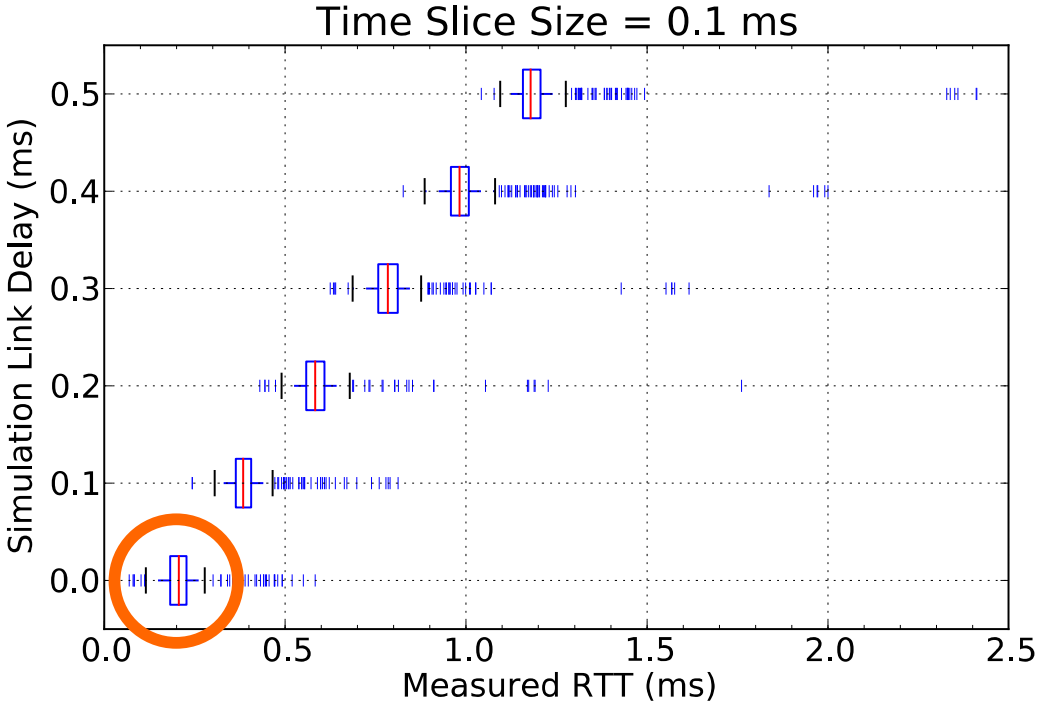
- **Perceived bandwidth is invariant to time slice size**

Evaluation: SliceTime Timing

How accurate is the time integration of VMs and the simulation?



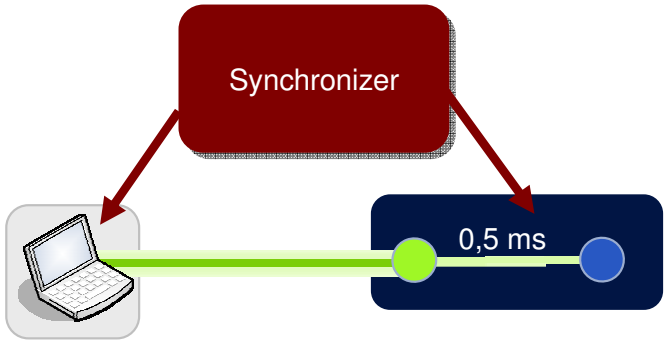
Measurement: 1500 RTTs (ICMP Echo Replies)
Simulated Link Delays between: 0,0 – 5ms
Static time slice size of 0.1ms



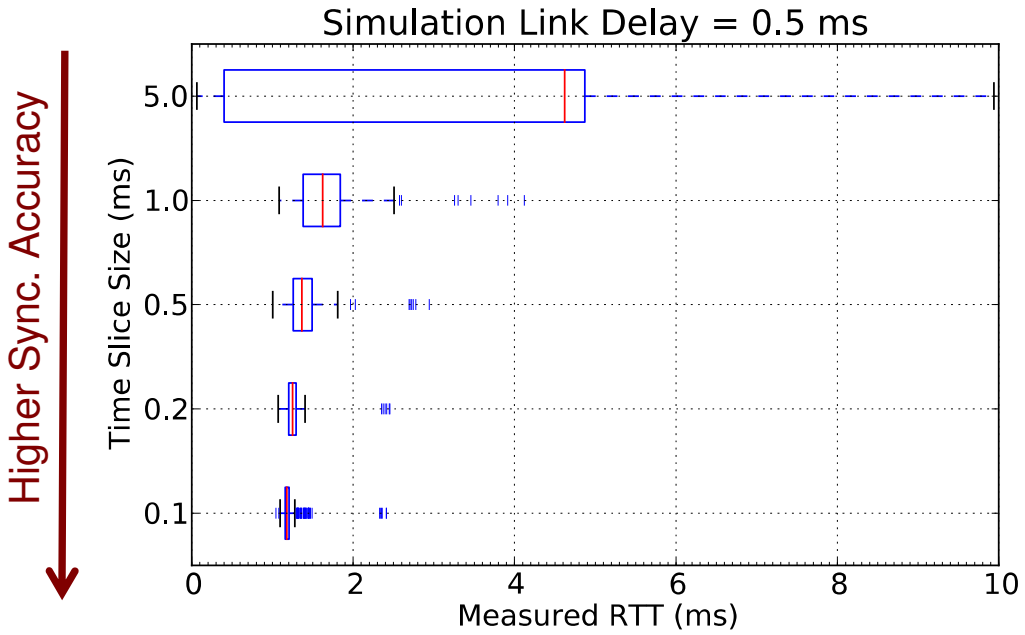
- **If no simulation delay is present → RTTs around ~ 0.2ms**
  - Base delay: Time needed for data exchange between VM & sync
- **RTT distributions shifted by twice simulation delay**

Evaluation: SliceTime Timing

How do different time slice sizes influence the results?



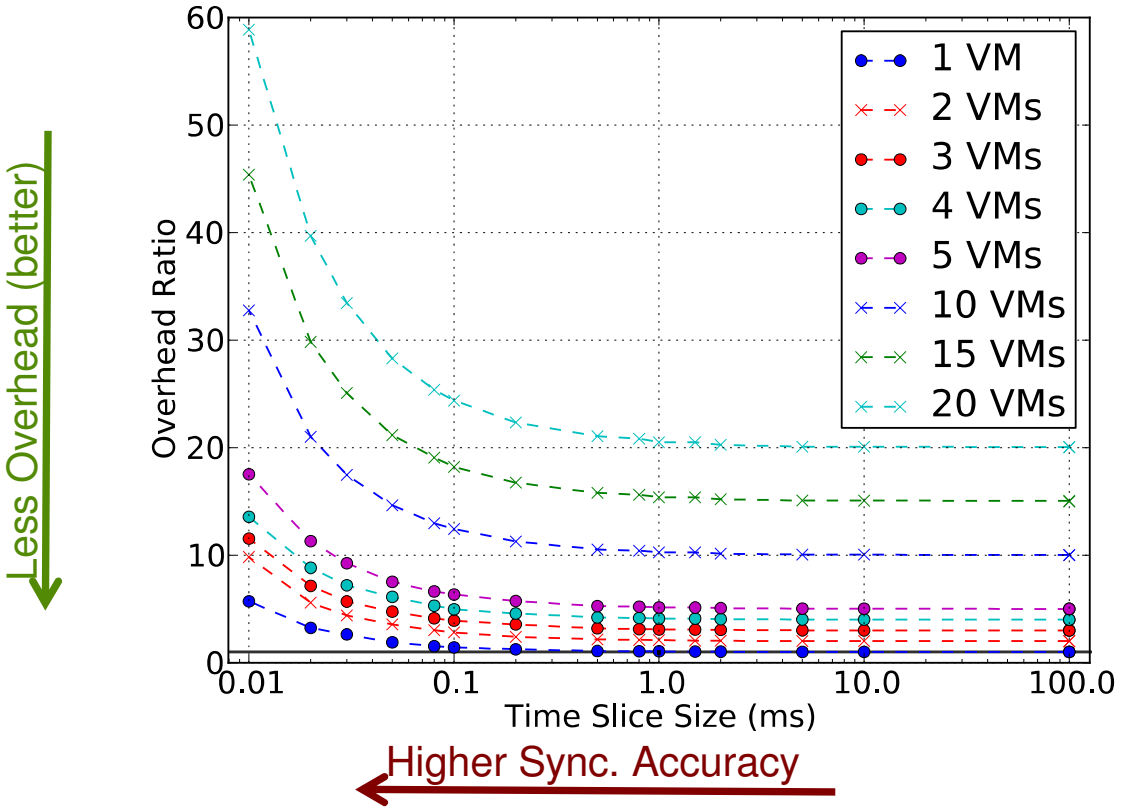
Measurement: 1500 RTTs (ICMP Echo Replies)
Variation: Time Slice Sizes



- **RTT distributions converge to base delay for smaller time slices (higher accuracies)**

Synchronization Overhead

How long does it take to execute 1s of virtual time?



- **Synchronization introduces additional run-time overhead**
  - Less than 5% for time slices > 0,5ms
  - Linear in the number of VMs

Conclusion

- **SliceTime allows network emulation scenarios with network simulations of any run time behavior**
- **SliceTime is accurate regarding timing and throughput**
- **SliceTime is resource efficient**
  - Low overhead even for time slices less 1ms
  - Saves physical hardware resources in comparison to real test beds
- **SliceTime is open source**
  - Get it at http://www.comsys.rwth-aachen.de/projects/slicetime
- **SliceTime extends the applicability of network emulation**

Questions?

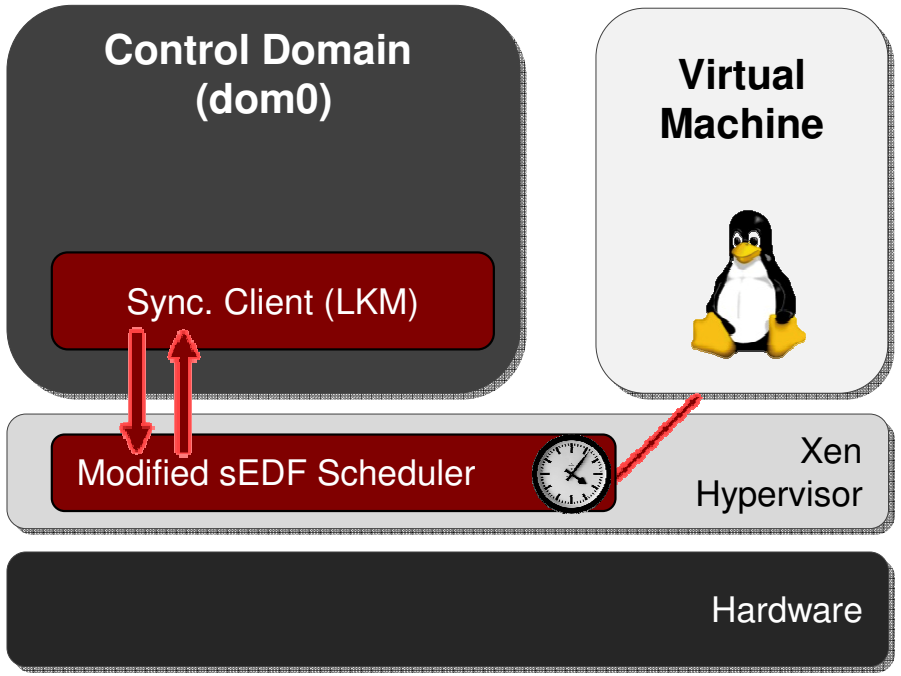
Implementation

Two main tasks:

- Execute VMs & simulation for exact time slice duration
- Provide VMs with illusion of gapless virtual time

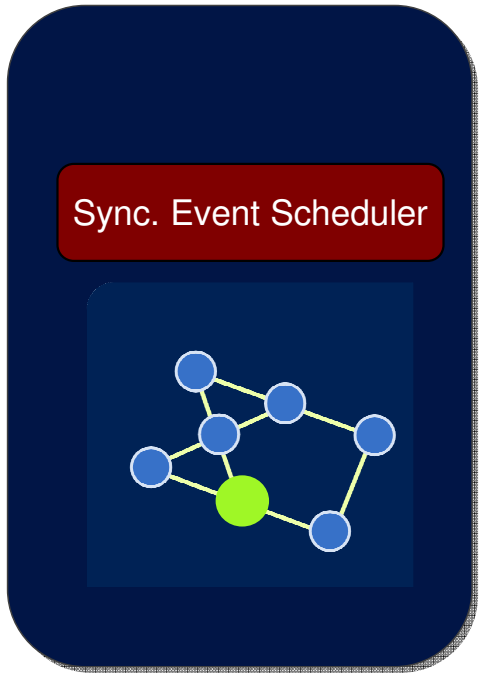
Implementation: Modified Xen environment

- **Synchronization Client**
  - Linux Kernel Module → save context switches
- **Modified sEDF scheduler**
  - Execute Xen domains for time slice duration
    - Extra scheduling queue for synchronized domains
    - Self-correction mechanism to overcome misattribution of run-time
  - Virtualizes time progression for synchronized domains
    - Calculates delta values for timers and clock sources



Network Simulation

- **Synchronized Event scheduler**
  - Synchronizes any ns-3 simulation with synchronizer/VMs
  - Checks if next event in queue resides in current time slice
- **Different ns-3 extensions**
  - Tunnel protocol → data exchange with VMs
  - WiFi emulation extensions
    - Provides VMs with wireless networking interface
    - Interface is intergrated with 802.11 model of ns-3



Synchronizer

- **Implements barrier synchronization algorithm**
  - Assignment of time slices
  - Synchronizes multiple VMs with multiple simulations
- **User-space application**
  - Can run on VM, simulation slave or dedicated host
  - Lightweight signaling protocol
- **VMs and simulations may join sync. dynamically**
  - Allows VM bootstrapping out of synchronization

Evaluation: Applicability

Can SliceTime ease the evaluation of networking software?

AODV Experiment
(Gray et al, 2003)

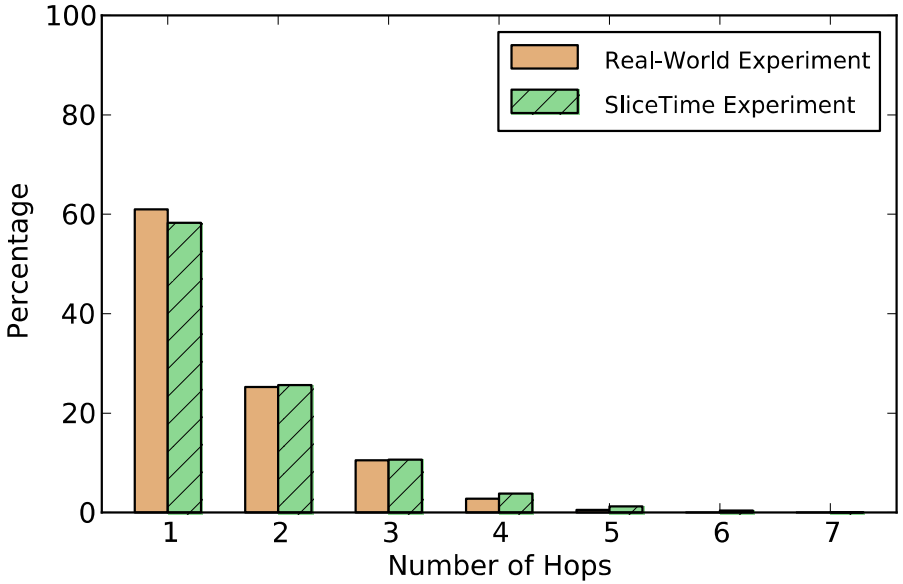
- 33 laptops running AODV
- 40 people carrying them around (on an athletic field)
- Random UDP traffic
- Laptops log traffic + position (GPS)
  - Logs available at CRAWDAD

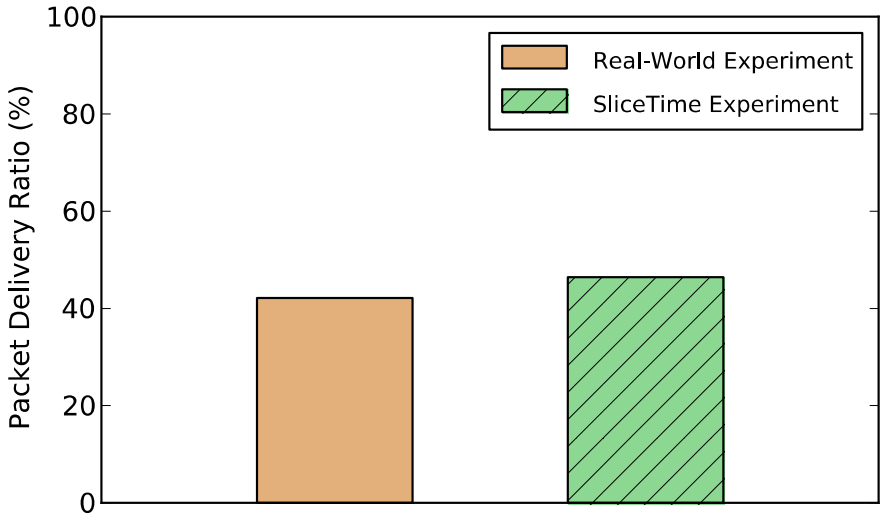
The SliceTime equivalent

- 33 Xen HVM domains / AODV
- SliceTime 802.11 extensions
- 1 physical PC
- Ns-3 mobility model based on GPS traces
- Traffic generator

Reproducing the AODV experiment by Gray

How do the results compare?

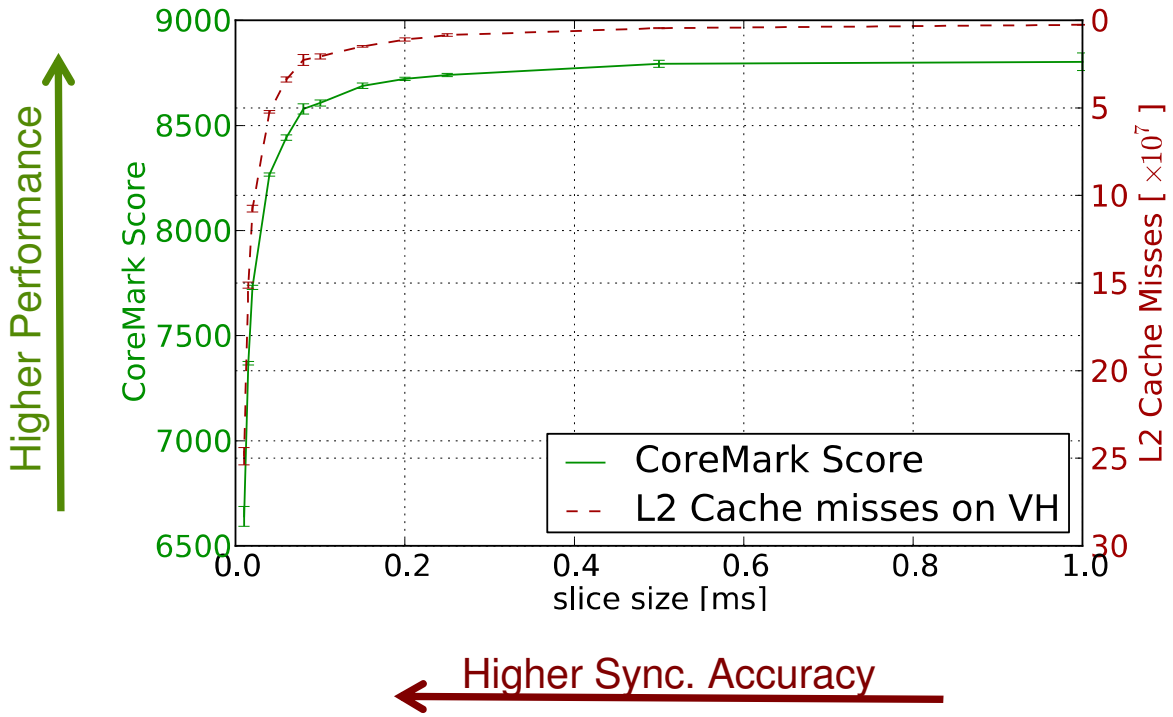




- **SliceTime produces results close to real-world measurements**
- **Always differences due to real-world/simulation disparity**

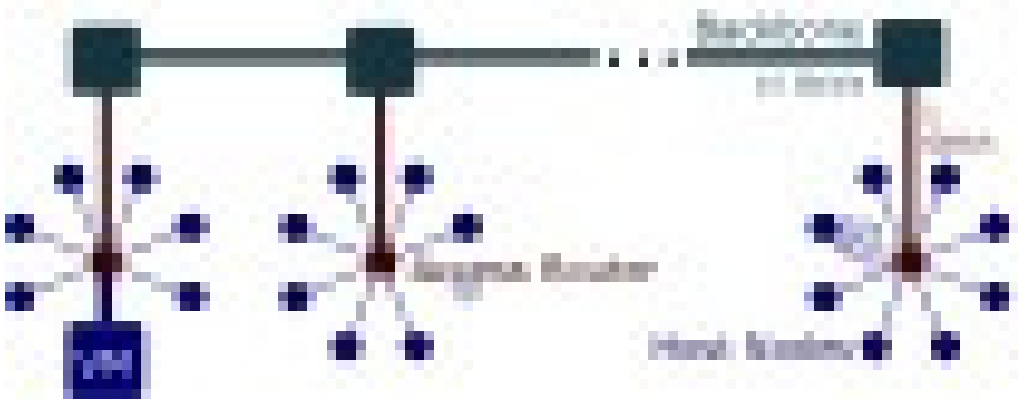
CPU Performance Impact

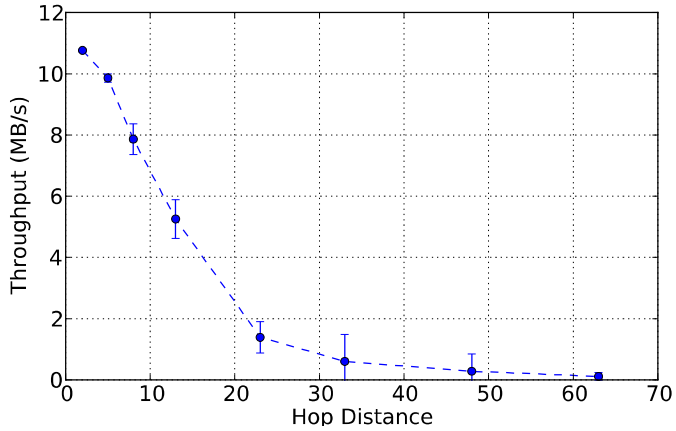
How about the CPU performance?
Doesn't the synchronization cause artifacts?



- **CoreMark score decreases for small time slices**
  - Almost no impact for slices greater than 0.1ms
  - Explanation: More L2 cache misses

SliceTime Simulation scalability

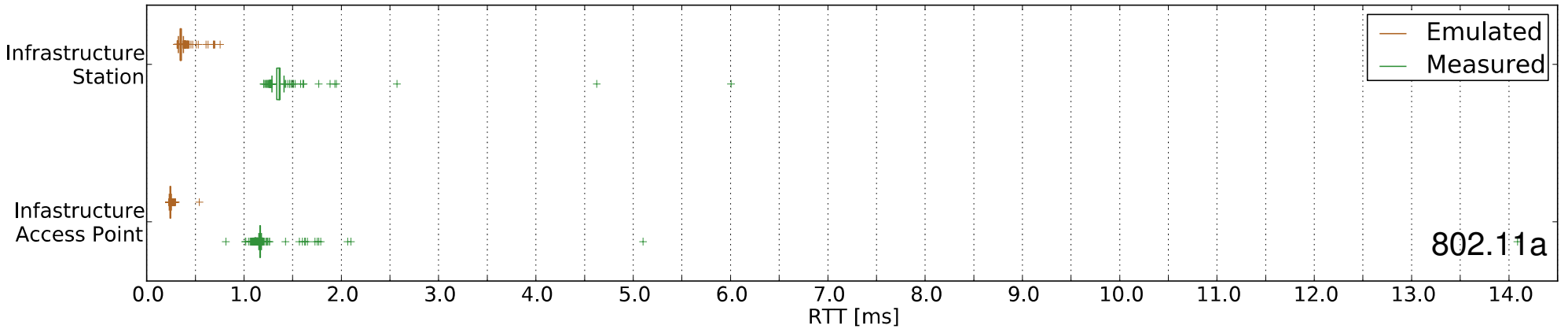


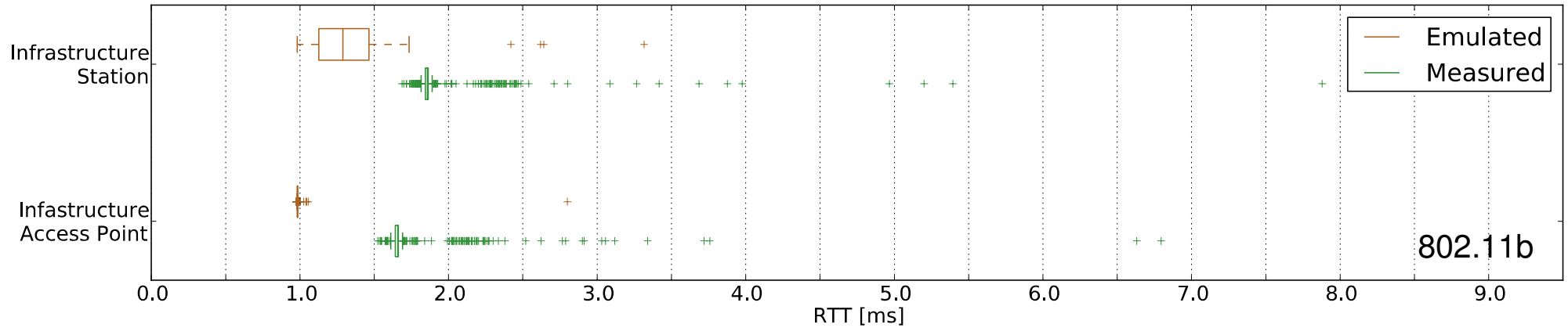


- **Setup: 15000 simulated nodes (60 stars with 250 nodes)**
  - Exchange data blocks among each other using HTTP
  - Executes~15 times slower than real-time
  - 1 VM attached to backbone
- **HTTP perormance measured with curl**
  - Expected result

802.11 Round Trip Times

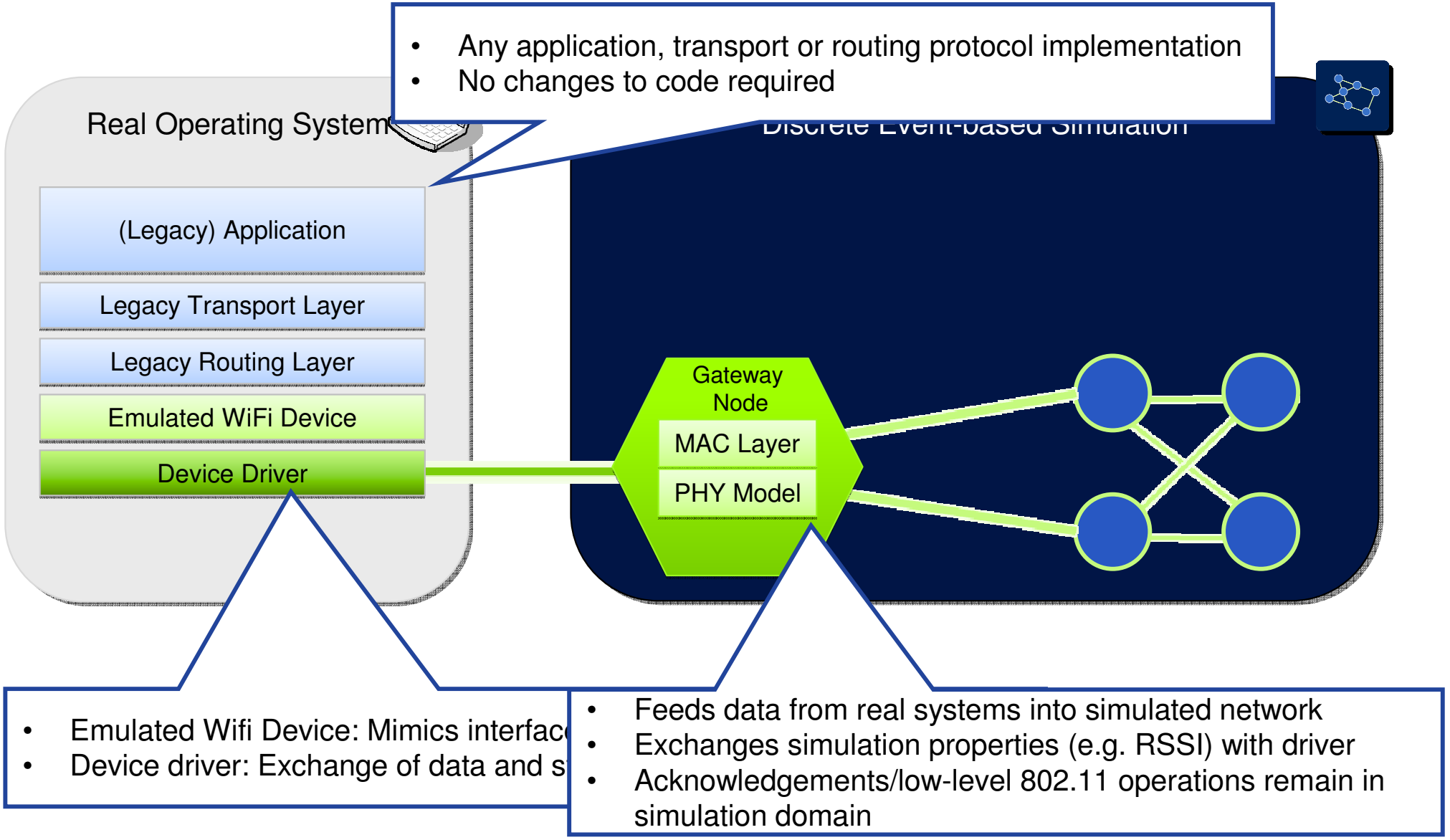
How do round trip times compare to real world 802.11?



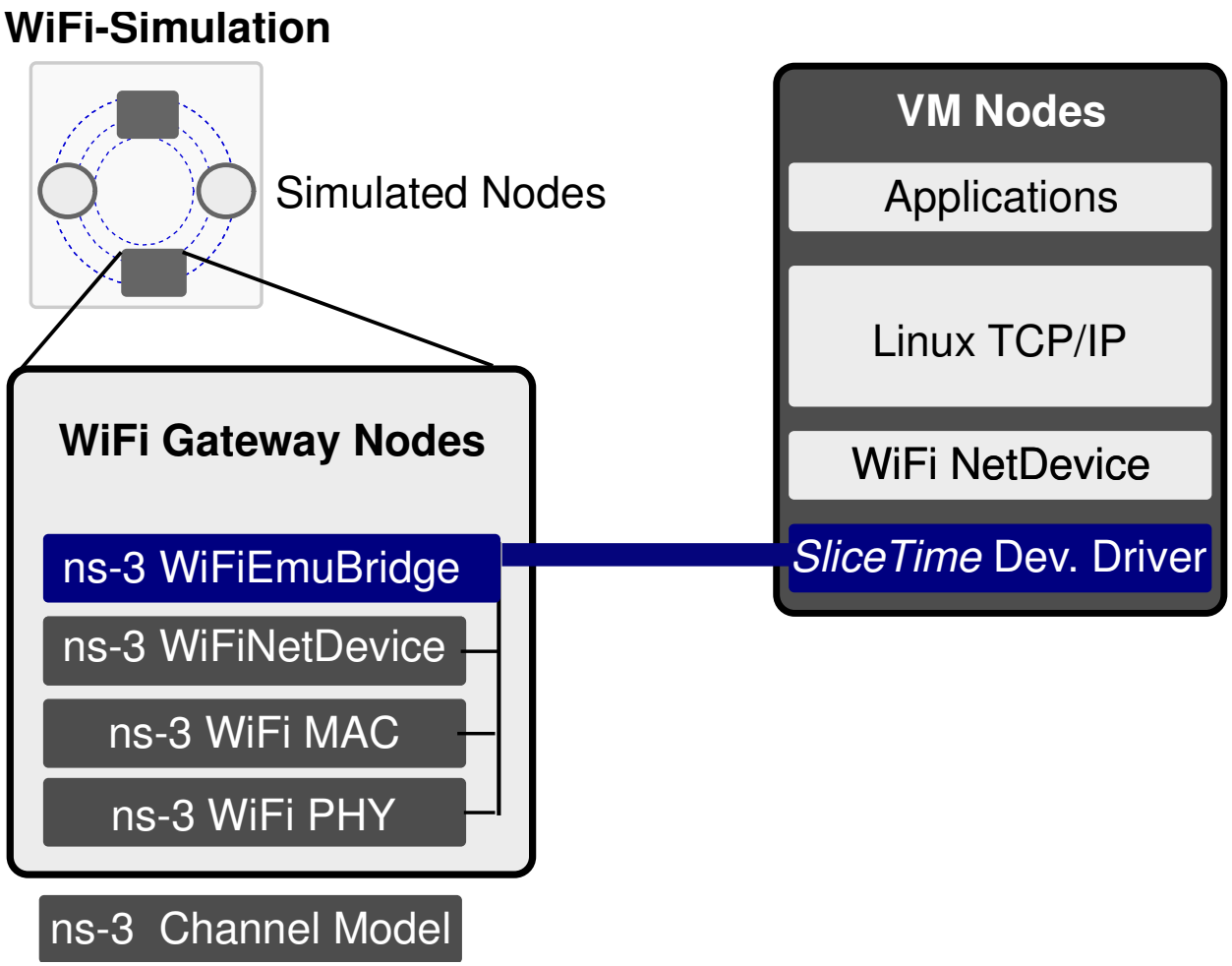


- **Emulated RTTs are lower than real world measurements**
  - ns-3 only approximations for link-level delays; no system delays

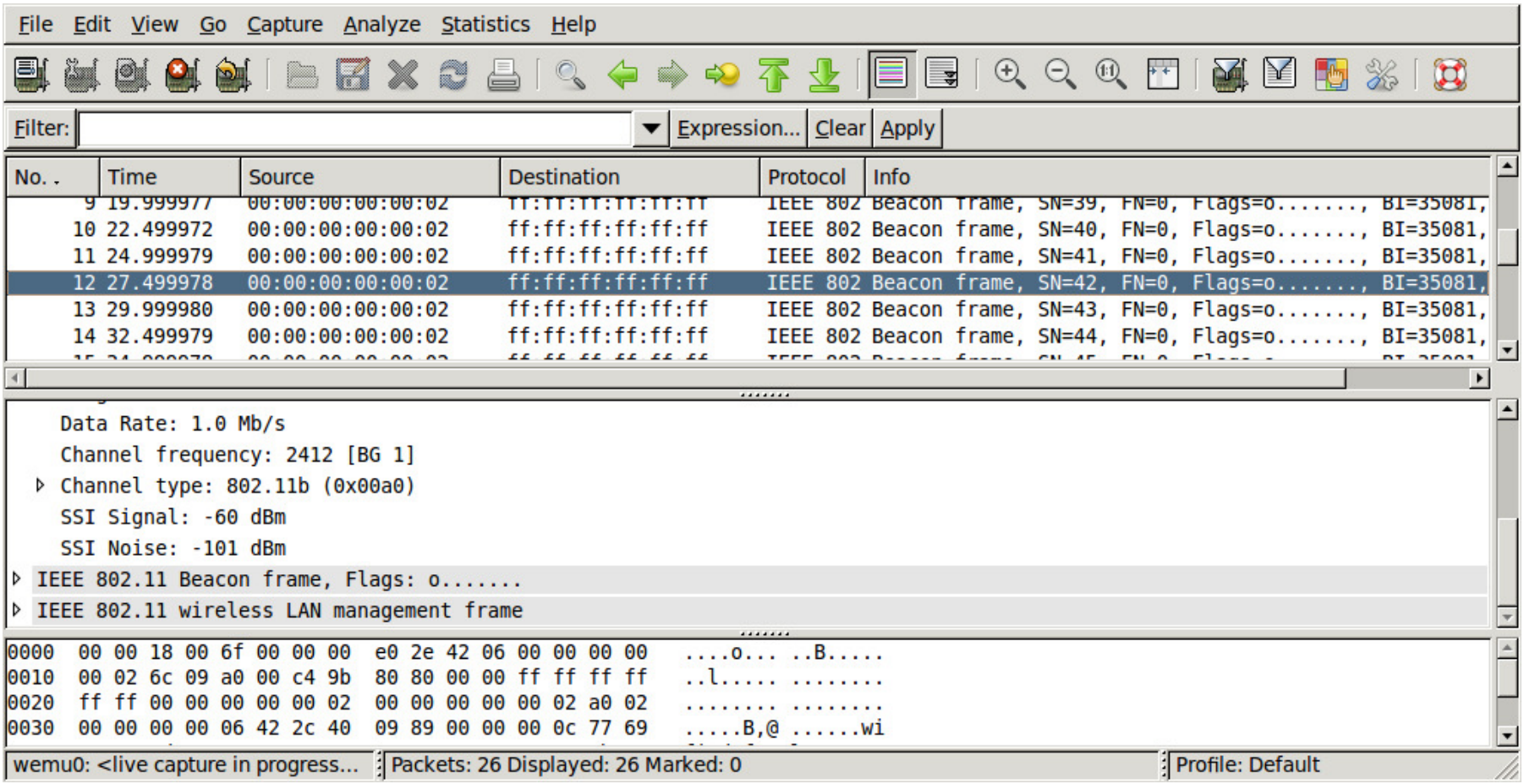
Device Driver-enabled Wireless Network Emulation



SliceTime WiFi extensions

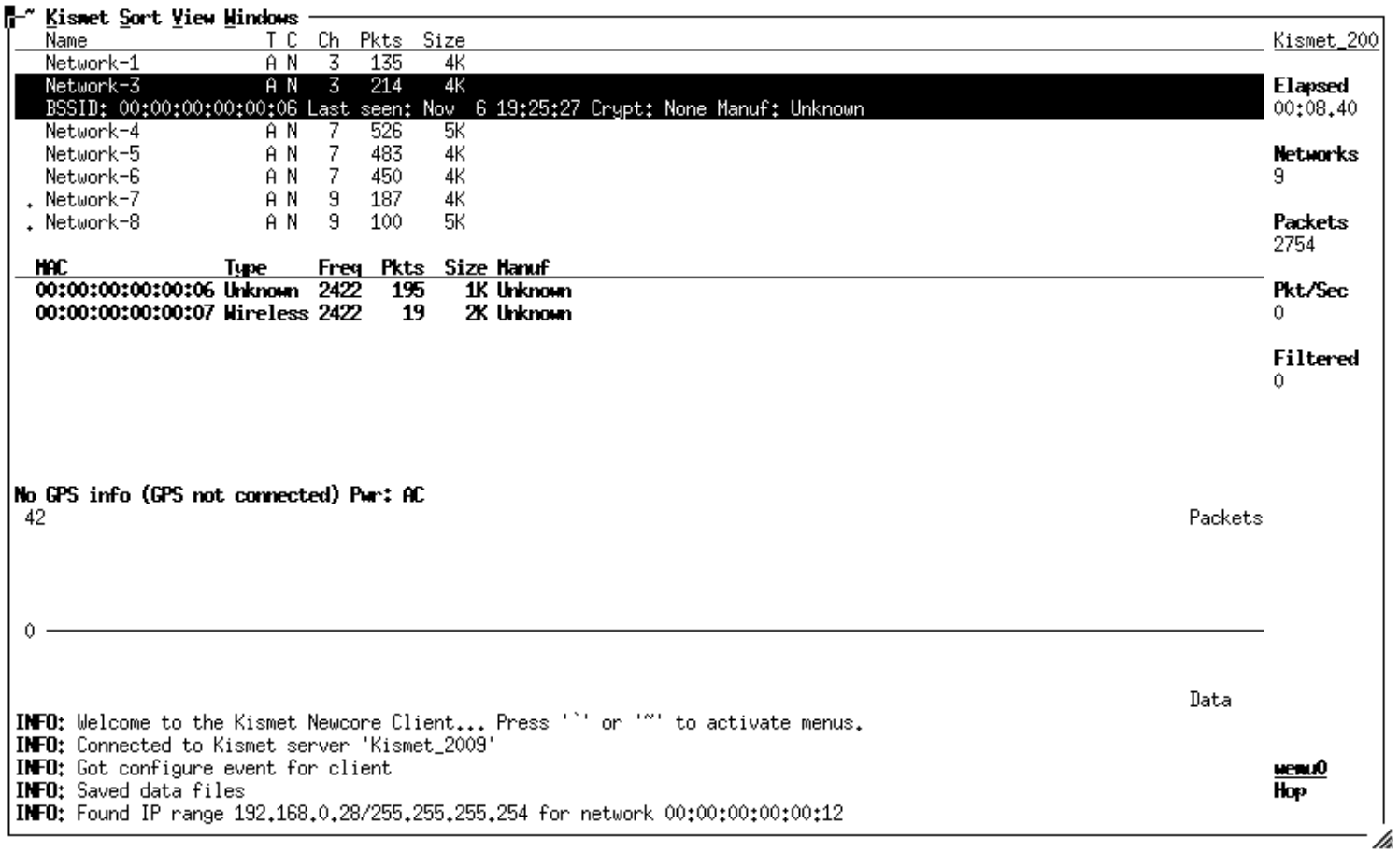


Legacy Applications



- **Wireshark for live monitoring of simulated WiFi networks**
  - Inspection of low-level 802.11 properties using Radiotap headers

Legacy Applications



- **Kismet being executed in simulated network**
  - Allows the execution of unmodified legacy applications that make use of Linux Wireless Extensions