Network Emulation as a Basis for Performance Analysis of Groupware

Prof. Dr. rer. nat. habil. Dr. h. c. Alexander Schill
Dipl. Inf. Robert Lübke
1. Motivation

2. Network characteristics and effects

3. State of the art
   - Network simulation
   - Network emulation
   - Hybrid approach

4. NESSEE Emulation platform
   - General architecture
   - Network testing with NESSEE
   - Demonstration

5. Conclusion
1. Motivation

Why do we need to reproduce network characteristics?

- Performing experiments in real test networks is only feasible in small scenarios (high coordination effort)
  → Need for **reproducible** and **controllable large-scale** experiments

- **Domains**:
  - Network protocol development
  - Network application development
  - Mobile app development

- **Use cases**:
  - Testing and validation
  - Performance analysis

Focus of this talk
Scenario: **Video conferencing system** with collaboration features
Motivation: Video Conferencing Scenario

offered as Software as a Service

Data Center (multiple Video Servers)

ISP1

ISP2

ISP3

Internet

Leased Line 100MBit/s
Delay: 20ms
Loss: 0.3%

DSL 6MBit/s
Delay: 15ms
Loss: 0.2%

LTE 16MBit/s
Delay: 50ms
Loss: 0.6%

Corporate Network

LAN 1
Delay: 5ms
Loss: 0.1%

LAN 2
Delay: 3ms
Loss: 0.1%

Gateway

Router
Possible testing goals:

- Does the video conference software properly reconnect after a disconnect?
- Does the ISP failback mechanism work?
- How many clients does the video server support?
- How is the end user experience *(video quality)* under bad network conditions?
2. Network characteristics and effects

What are relevant characteristics and effects for reproducing network behavior?

- Network topologies
- Data rate
- Packet delay and its variation (jitter)
- Packet-level effects: loss, reordering, duplication
- Connection-level effects: disconnection, handover
• Communication via multi hop connections with possible network impairments on every link / node

• Consideration of the cumulated effects on one path

\[ \text{Tx: Sender} \quad \text{Rx: Receiver} \quad \text{P: Path} \quad \text{H: Path length (hops)} \]
• Specifies how much data can be transferred in a given time unit
• Often called (network / data / digital) **bandwidth**, because of its proportionality to the analog bandwidth of the physical channel (*in Hz*)

• Common metrics:
  • **Capacity** $C_P$
    Maximum data rate of a given network path $P$

\[
C_P = \min_{i=1,\ldots,H} C_i
\]

• **Available bandwidth** $A$
  time-dependent, specified by the current utilizations ($0 \leq u \leq 1$) and capacities of the corresponding links

\[
A_P = \min_{i=1,\ldots,H} (C_i(1 - u_i))
\]

• **Bulk TCP Throughput** $BTT$
  specifies how much payload data can be transferred in a given amount of time using TCP
• Time between sending and receiving a message

• Common metrics:
  • **One Way Delay** (OWD) – only one communication direction
  • **Round Trip Delay** (RTD) – time between sending a message and receiving the corresponding answer

• **Overall delay**: 
  • routing delay $D_R$ – processing and queueing inside the router
  • transmission delay $D_T$ – placing the packet onto the link
  • propagation delay $D_P$ – passing the packet from one end of the link to the other

$$OWD_P = \sum_{i=1}^{H} OWD_i = \sum_{i=1}^{H} (D_{R,i} + D_{T,i} + D_{P,i})$$
Variability of Packet Delay - Jitter

- In real networks packet delay is usually not constant.

- **Packet Jitter** $\Delta \text{OWD}$
  - Variability of packet inter-arrival times
  - Calculation is based on packet delay definition of two subsequent packets ($j$ and $j-1$)

$$
\Delta \text{OWD}^j = \text{OWD}^j - \text{OWD}^{j-1} = (D_R^j - D_R^{j-1}) + (D_T^j - D_T^{j-1}) + (D_P^j - D_P^{j-1})
$$

- If packets $j$ and $j-1$ have the same size and are sent over the same path, the transmission and propagation delays equalize.

  $\Delta \text{OWD}$ mainly depends on varying load and queue sizes of the routers due to parallel traffic.
Packet Loss

- **Causes**
  - **Packet content is corrupted** during transmission → Error is recognized (check sum), but cannot be corrected → Packet is dropped
  - **Congestion** in network nodes: queues of a router are full → Packets are dropped

- **Metric**
  - **One Way Loss Ratio** OWLR:
    
    \[ OWLR = 1 - \frac{P_{rec}}{P_{sent}} \]

  - Cumulated Packet Loss rate:
    
    \[ OWLR_P = 1 - \prod_{i=1}^{H} (1 - OWLR_i) \]

\[ P_{sent} : \text{Number of sent packets} \]
\[ P_{rec} : \text{Number of received packets} \]
\[ P : \text{Path of length } H \]
Further network effects

- **Packet reordering**
  - Causes: parallel processing, load balancing on multiple paths, path changes
  - Metric: Packet Reordering Rate (PRR)
    \[ PRR_P \leq 1 - \prod_{i=1}^{H} (1 - PRR_i) \]

- **Packet duplication**
  - Causes: defective router
  - Metric: Duplication Rate (DR)
    \[ DR_P = -1 + \prod_{i=1}^{H} (1 + DR_i) \]

- **Disconnection**
  - Causes: defective hardware, power outage, insufficient signal strength in wireless technologies, firewalls

- **Handover** in wireless technologies
  - Cause: user mobility
  - *Intra-* and *inter-technology* handover
  - *Hard* and *soft* handover
What are current means to reproduce network behavior?

- Network simulation
- Network emulation
  - Using software
  - Using hardware
  - Emulation environments
Network Simulation

• **Synthetic approach** to reproduce network conditions

• **Simulation model**
  • Network topology and characteristics are transformed into a software model
  • Quality of the model affects quality of the results
  • Large scenarios and complex models → high computation effort

• **Advantages**
  • Reproducibility
  • Controllability

• **Disadvantages**
  • Huge scenarios cannot be simulated on one machine
  • Limited to simulation of the network – not suitable for testing applications
Network Simulation with ns-2

Example: Network Simulator *ns-2*

- Discrete event simulator targeted at networking research
- Supports simulation of TCP, routing, and multicast protocols over wired and wireless (local and satellite) networks
- Available for Linux, BSD, OS X and Windows (Cygwin)
- Visualization with Network Animator:

![Network Animator Screenshot](http://www.isi.edu/nsnam/nam/)
Network Simulation with ns-2

Create six nodes

set n0 [ns node]
set n1 [ns node]
set n2 [ns node]
set n3 [ns node]
set n4 [ns node]
set n5 [ns node]

$ns duplex-link $n0 $n2 2Mb 10ms DropTail
$ns duplex-link $n1 $n2 2Mb 10ms DropTail
$ns simplex-link $n2 $n3 0.3Mb 100ms DropTail
$ns simplex-link $n3 $n2 0.3Mb 100ms DropTail
$ns duplex-link $n3 $n4 0.5Mb 40ms DropTail
$ns duplex-link $n3 $n5 0.5Mb 30ms DropTail

Create links between the nodes

Setup a TCP connection

set tcp [new Agent/TCP]
$ns attach-agent $n0 $tcp
set sink [new Agent/TCPSink]
$ns attach-agent $n4 $sink
$ns connect $tcp $sink
$tcp set fid_1
$tcp set packetSize_ 552

Setup a FTP over TCP connection

set ftp [new Application/FTP]
$ftp attach-agent $tcp

Source: [AJ03]
Network Emulation

- Characteristics of real network traffic are modified according to a given configuration (traffic shaping)
- Can take place on different levels of the network stack – **IP level emulation** is most established

**Categories:**
- Hardware-based
- Software-based
- Emulation environment
Hardware-based Network Emulation

- Shape traffic between two or more network ports
- Mostly web-based user interfaces

**Advantages**
- Dedicated hardware → high performance and precision

**Disadvantages**
- Expensive
- Missing flexibility (fixed configuration)

**Example:** Linktropy 7500 PRO (*Computer Networks Group*)
Software-based Network Emulation

- Packet interception in the network stack of the operating system
- Manipulation of incoming and outgoing traffic (based on rules)

**Advantages**
- Flexibility due to dynamic configuration
- Many different implementations for various use cases

**Disadvantages**
- Performance and accuracy depend on concrete implementation and underlying hardware

**Examples:**
- Linux: NISTNet [CS03], NetEm [Hem05], WANEm [KN11]
- FreeBSD: Dummynet [CR10], ModelNet [VYW+02], KauNet [GHB08]
### Software-based Network Emulation

<table>
<thead>
<tr>
<th>Emulator</th>
<th>OS</th>
<th>Further devel.</th>
<th>Data rate</th>
<th>delay</th>
<th>jitter</th>
<th>loss</th>
<th>corruption</th>
<th>duplication</th>
<th>reordering</th>
<th>hand over</th>
<th>disconnect</th>
<th>topologies</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>NISTNet</td>
<td>Linux</td>
<td>×</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>GUI available, detailed filters</td>
</tr>
<tr>
<td>NetEm</td>
<td>Linux</td>
<td>√</td>
<td>×</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>Part of Linux kernel, GUI available</td>
</tr>
<tr>
<td>WANEm</td>
<td>Linux</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>×</td>
<td>√</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>Good usability, Live CD</td>
</tr>
<tr>
<td>Dummynet</td>
<td>FreeBSD, Linux, Windows</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>√</td>
<td>Part of FreeBSD kernel</td>
</tr>
<tr>
<td>ModelNet</td>
<td>FreeBSD</td>
<td>×</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>×</td>
<td>×</td>
<td>√</td>
<td>×</td>
<td>×</td>
<td>√</td>
<td>Distributed emulation</td>
</tr>
<tr>
<td>KauNet</td>
<td>FreeBSD</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>√</td>
<td>Pattern-based approach</td>
</tr>
<tr>
<td>NetPath</td>
<td>Linux</td>
<td>×</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>High precision / performance</td>
</tr>
</tbody>
</table>
Network Emulation Environments

- Test networks with many distributed nodes
- Mostly based on federation of multiple universities / research institutes that share their infrastructure with each other

- **Advantages**
  - Possibility to perform large experiments

- **Disadvantages**
  - Resources must be reserved before the experiment
  - Limited flexibility (mostly fixed network emulation configuration)

- Examples:
# Network Emulation Environments

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>PlanetLab</td>
<td>O</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>√</td>
<td>×</td>
<td>×</td>
<td>Large, distributed testbed</td>
</tr>
<tr>
<td>EmuLab</td>
<td>O</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>(√)</td>
<td>Testbed &amp; open source software</td>
</tr>
<tr>
<td>OneLab</td>
<td>O</td>
<td>√</td>
<td>√</td>
<td>×</td>
<td>O</td>
<td>√</td>
<td>×</td>
<td>Contains multiple specialized testbeds</td>
</tr>
<tr>
<td>ORBIT</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>×</td>
<td>√</td>
<td>√</td>
<td>×</td>
<td>Grid of 400 wireless nodes in laboratory environment</td>
</tr>
<tr>
<td>VINI</td>
<td>O</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>√</td>
<td>√</td>
<td>×</td>
<td>Based on PlanetLab; adds better controllability and realistic emul.</td>
</tr>
<tr>
<td>WhyNet</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>√</td>
<td>×</td>
<td>√</td>
<td>×</td>
<td>Wireless network emulation</td>
</tr>
</tbody>
</table>

Common focus: **Network protocol development**, instead of testing and experimenting with network applications.
Hybrid Approach

Two distinct methods:

- **Perform a simulation step before the actual emulation** to determine concrete network parameter values
  - Example: *Wireless-NINE* [CPGD10]
  - Advantage: Increase accuracy of the emulation
  - Disadvantage: Experiment partition in two steps is not always possible

- **Usage of the physical network interface during simulation**
  - Example: *ns-3* [nsnam.org]
  - Advantage: simulated traffic is injected into real networks
  - Disadvantage: no real-time behavior for complex models or large scenarios
What are means to reproduce network behavior for large-scale experiments with distributed systems?

Simulation
- Too abstract, because software under test must be part of simulation model

Emulation
- Using Hardware
  - High performance / accuracy
  - Missing flexibility, because of fixed configuration
- Using Software
  - Highly flexible, extensible, adaptable
- Emulation environments
  - Limited flexibility (mostly fixed configuration)
  - Often: Cannot be used until adding machines to the federated testbeds

Used as a basis of our own emulation platform
How to facilitate network tests and experiments for large-scale distributed systems?

- Research project in cooperation with Citrix Systems, Online Services Division

- Network Endpoint Server Scenario Emulation Environment
NESSEE Features

- Emulation platform that facilitates the setup and coordination of different kinds of testing

- Main features
  - **Network testing**: fine-grained configuration of all relevant network characteristics
  - **Scalability testing**: control and coordinate large numbers of software under test (SUT) instances (experiments with >6000)
  - **Functional testing**: monitor the SUT, log file evaluation
  - **Test Description Language (TDL)** and scriptable test cases to achieve flexibility and reusability (GUI for test case authoring)
  - **Continuous Integration (CI)**: run NESSEE tests from within the development workflow using common build and CI tools
  - **Platform independence**: SUT on different operating systems supported (tested with Linux and Windows)
  - Convenient **web front end** to manage the whole platform
- **Manage the whole platform**: administrative tasks (test systems, configuration), test case management, test execution & monitoring
- **Handle real-time updates** from the NESSEE Server with WebSockets or long polling – signalR framework
- **Implemented in JavaScript / HTML5** – dojo framework
Test Description Language (TDL)

- **XML-based** generic description language for test cases
- As **simple** as possible to allow manual editing
- Support for **authoring environment** (NESSEE Editor)
- **Modular design** with independent components

```
TestCaseDescription

Network Topology

Behavior

Coordination functions

Server functions

Client functions
```
**TDL Example** (Network Topology)

```xml
<ClientComponents>
  <NetworkNode NetworkCapabilitiesId="DSL2000">
    <NetworkEndpointType id="dsl2000endpoint"/>
  </NetworkNode>
  <NetworkNode NetworkCapabilitiesId="WiFiNRouter">
    <NetworkEndpointType id="wifiendpoint" NetworkCapabilitiesId="OldWiFiEndpoint"/>
  </NetworkNode>
  ...
</ClientComponents>

<ServerComponents>
  <Datacenter id="datacenter_asia">
    <ISP id="isp03" NetworkCapabilitiesId="cap03"/>
    <ISP id="isp04" NetworkCapabilitiesId="cap04"/>
    <ISP id="isp05" NetworkCapabilitiesId="cap05"/>
    <VideoCluster id="vidCluster03" ISPId="isp03"/>
  </Datacenter>
</ServerComponents>
```
Network testing with NESSEE

- Network emulator: extended version of KauNet for FreeBSD (based on DummyNet)
- Based on pipes and rules
  - **Pipe**: configured with certain network characteristics (delay, loss, bandwidth, ...)
  - **Rule**: Defines which packets have to pass which pipes (like firewall rules)
• NESSEE fulfills all requirements of an emulation platform for automated functional, load and network tests of distributed systems

• Non-functional requirement: **Scalability**

  • **Scalability and accuracy of the network emulation**
    ▪ Deviations of configured and measured values in large and complex scenarios
    ▪ Results (relative deviations):
      - Data rate: ±3.50%
      - Packet Delay: ±0.20%
      - Packet loss: ±0.10%
      - Packet Reordering: ±0.13%
      - Packet duplication: ±0.00%
• **Scalability of the script engine approach**
  - Comparison with other script engines in multiple scenarios
    → Sufficient results (not always the best)
  - Example scenario: Calls from script to native code

![Graph showing time in ms vs number of calls from script to original code for different engines: NESSEE Script Engine, Chakra, RemObjects, V8.NET.](image-url)
Evaluation

- **Scalability of the IPC approach for controlling the SUT**
  - Maximum number of SUT instances per test system
  - Dependence on underlying hardware
    - 500 instances can be started in about 80s
    - Active video conferences: 200…250 instances / test system
5. Conclusion

- **Reproduction of network conditions** is essential in tests and evaluation of network applications and distributed systems.

- All **relevant characteristics** should be covered: **data rate, delay, jitter, loss, reordering, duplication, hand over, disconnect**

- **Network Simulation**: synthetic approach; uses simulation model

- **Network Emulation**: practical approach; shapes real traffic

- **NESSEE Emulation Platform** facilitates the setup and coordination of large-scale network, load and functional tests
Literature & References

- **Literature:**
  - TANENBAUM, Andrew; WETHERALL, David: *Computer Networks*. Prentice Hall International, July 2013

- **References:**
  - **[AJ03]**: ALTMAN, Eitan; JIMÉNEZ, Tania: *NS Simulator for beginners*, Lecture Notes, Univ. de Los Andes, Mérida, Venezuela and ESSI, Sophia-Antipolis, France
  - **[Hem05]**: HEMMINGER, Stephen: *Network Emulation with NetEm*. In: linux.conf.au. Canberra, Australia, April 2005