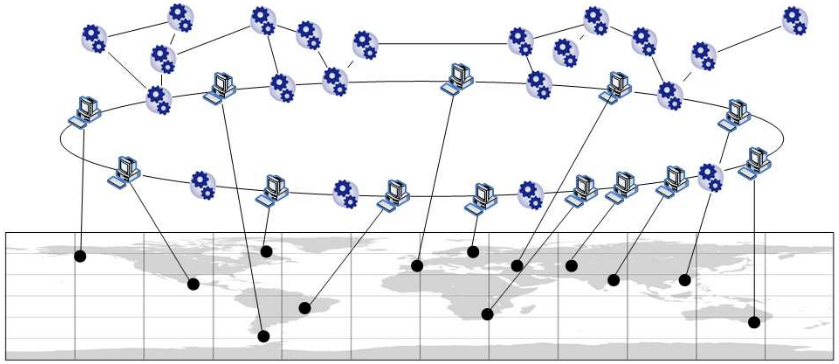
UNIVERSITÄT PADERBORN
Die Universität der Informationsgesellschaft

Institute of Computer Science
Research Group – Theory of Distributed Systems

Tutorial Slides on PeerfactSim.KOM



PeerfactSim.KOM: A Peer-to-Peer System Simulator

Dr.-Ing. Kalman Graffi
Faculty for Electrical Engineering, Computer Science at the University of Paderborn

Overview

1 General Evaluation Methods

2 Overview on PeerfactSim.KOM

3 How to Use PeerfactSim.KOM – A Step by Step Guide

- 3.1 Downloading and Installation
- 3.2 Running a Simulation
- 3.3 Simulation Visualization (Replay)
- 3.4 Setting up a first Simulation - the Config - File
- 3.5 Observing what is happening → Analyzers
- 3.6 Simple Example: Chord Lookup
- 3.7 Plotting the results
- 3.8 Using GnuPlot

1 General Evaluation Methods

Analysis

- Often simplified model
  - Homogeneous nodes, probabilistic actions
- Leads to proofs (under specific conditions)

Example:

- Given: Weighted DAG
- Results:
  - Proofs
  - Complexities - O (log N)

Good: General results

Weakness:

- Details lost in abstraction
- Sometimes the constants are important

General Evaluation Methods

Simulation

- Advanced and heterogeneous model
  - Specific node characteristics, capacities, behavior
- Investigates emerging behavior
- Often focus on quality of service
  - Response times, induced traffic, specific node load
  - A response time of 1s to 5s matters!

Examples:

Given:

- 10.000 nodes, capacity distribution X
- 70% altruistic nodes, 20% selfish nodes, 10% malicious nodes
- Protocol XY, workload Z

Results:

- Statistics on quality of service over time

General Evaluation Methods

Prototype – in Testbed / in real world

- Deployment of code in real testbed (e.g. PlanetLab)
- Most adequate models, unpredictable user behaviour
- Challenging to coordinate the tests, gather results
- Logging and coordination might disturb the results

Example

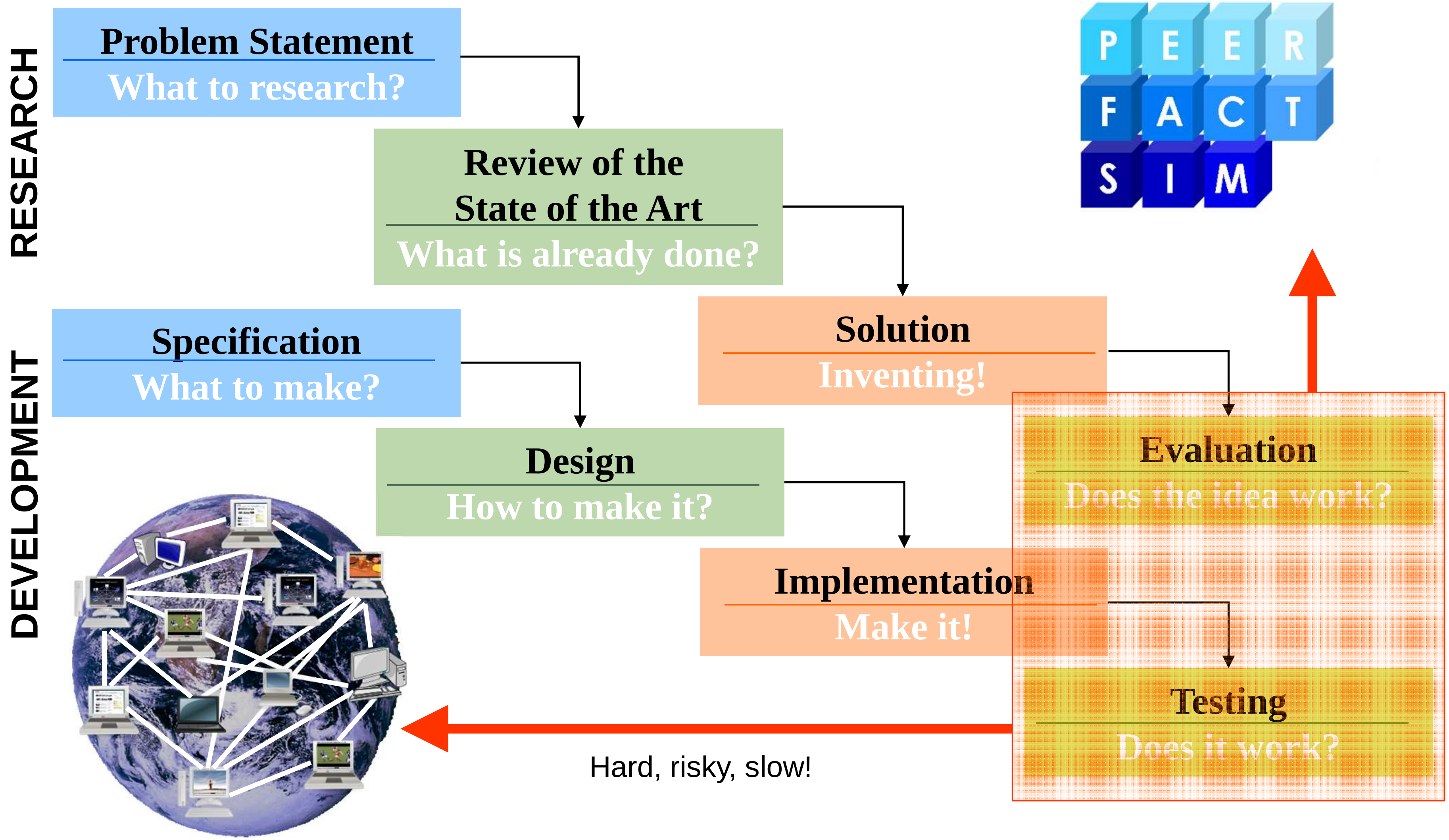
Given:

- 733 PlanetLab nodes
- Full protocol stack: IP, TCP/UDP, middleware, application, virtual users
- Deployment in global PlanetLab

Results:

- Behavior under realistic network conditions
  - Delays, jitter, node load …

Research & Development of New (Peer-to-Peer) Applications?



RESEARCH

Problem Statement
What to research?

Review of the State of the Art
What is already done?

Solution
Inventing!

DEVELOPMENT

Specification
What to make?

Design
How to make it?

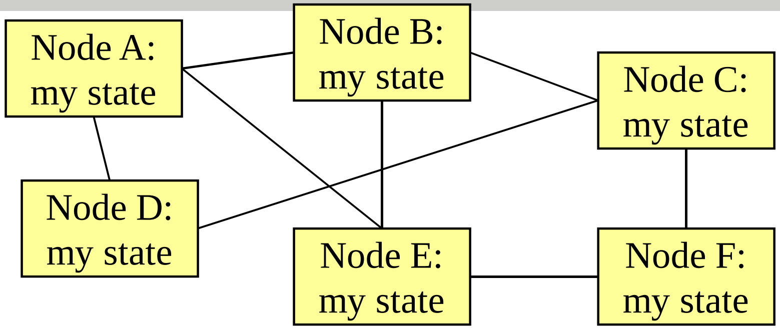
Implementation
Make it!

Evaluation
Does the idea work?

Testing
Does it work?

Hard, risky, slow!

Simplified Overview on Simulations



Node A: my state

Node B: my state

Node C: my state

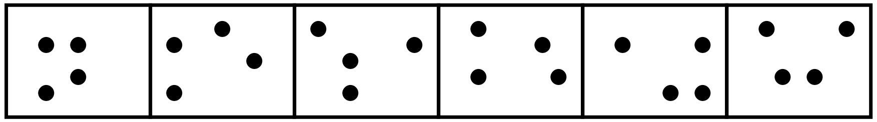
Node D: my state

Node E: my state

Node F: my state

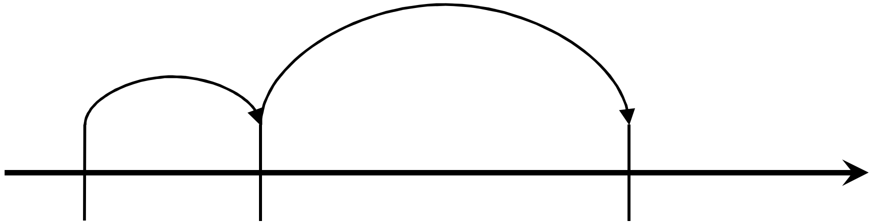
Simulated hosts

- Every node has own state
  - Current load, capacities, strategies …
- Set of possible actions
  - Triggered by workload / autonomously
- Defined reactions on incoming messages



Round-based simulations

- All actions in one round in parallel
- Round i only affects round i+1
- Unrealistic behavior
- Easier to implement



Timeline of events

Event-based simulations

- Every event is scheduled for a time point
- Only passed to receiver when time is due
- Events may initiate new events
- Strict order of events, more realistic

2 Overview on PeerfactSim.KOM

History

- Started in 2005 as evaluation tool for a Ph.D.
- At TU Darmstadt, Multimedia Communication Lab
- Used and heavily extended in the project
  - DFG – Forschergruppe 733 – QuaP2P
  - Improvement of the Quality of Peer-to-Peer Systems by Systematically Researching Quality Features and Their Interdependencies
  - Continuously 7+ researchers
  - From 2006 - now

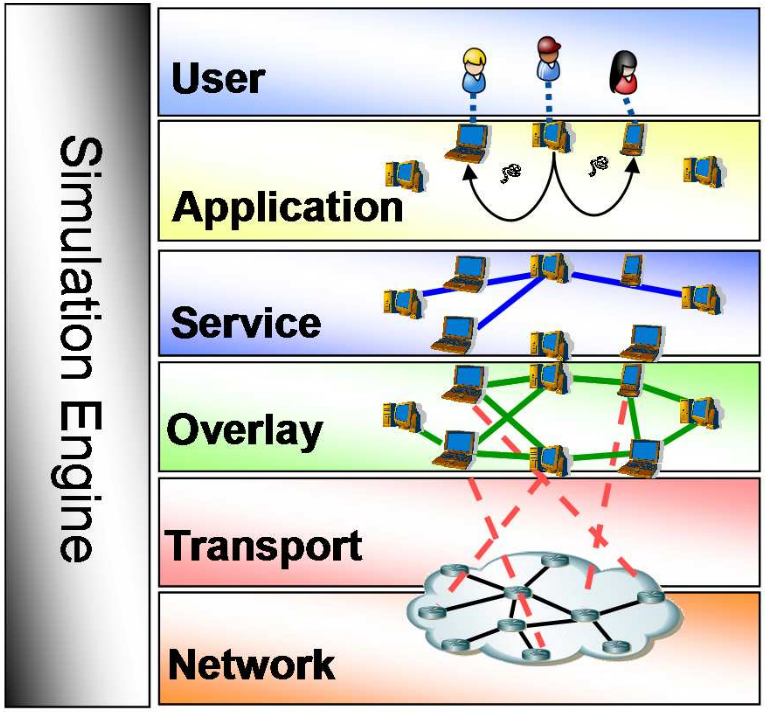
Type

- Event-based simulator
- Written in Java
- Simulations up to 100K peers possible
- Focus on simulation of p2p systems on various layers
  - Remember 7+ researchers looking at interdependecies

Layered View

Layered Architecture

- Easy exchange of components
- Testing of new applications
- Testing of new mechanisms



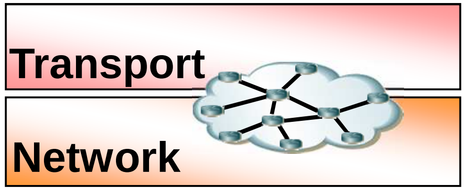
Main idea

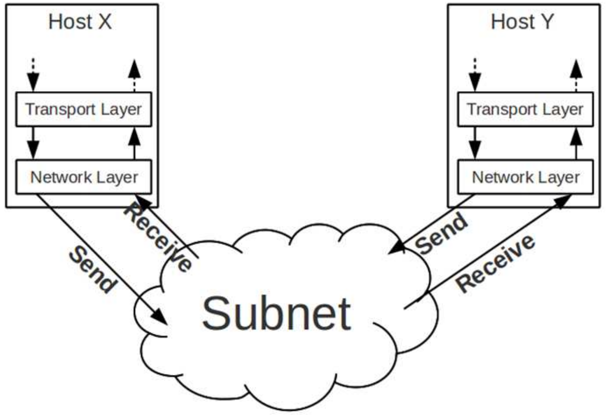
- Every layer has a simple implementation
- Enables testing of individual layer mechanisms
  - on its own
  - in combination with other layers

Underlay

General Concept

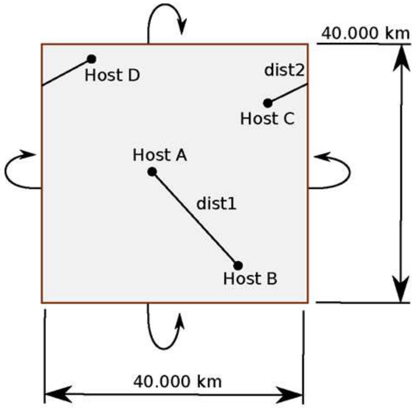
- Hide topology of the Internet
- Consider only End-to-End connections
- Dedicated component for the logic





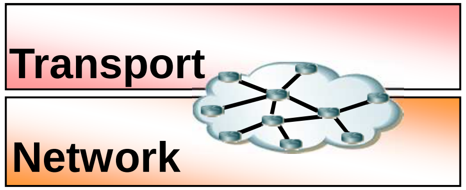
Simple Network Layer

- Simple latency models
  - Static latency
  - Distance-based latency
- No packet loss
- Omission of packet size and bandwidth
- Supporting simplified UDP

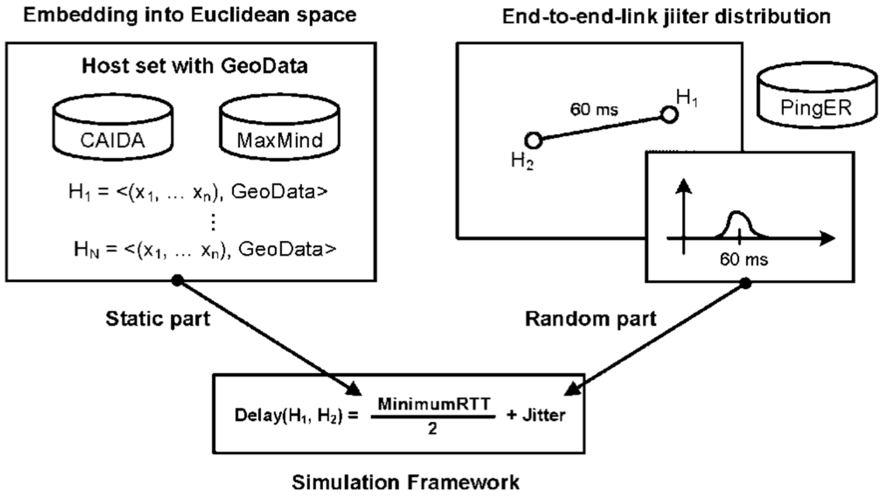


Underlay

GNP Network Layer

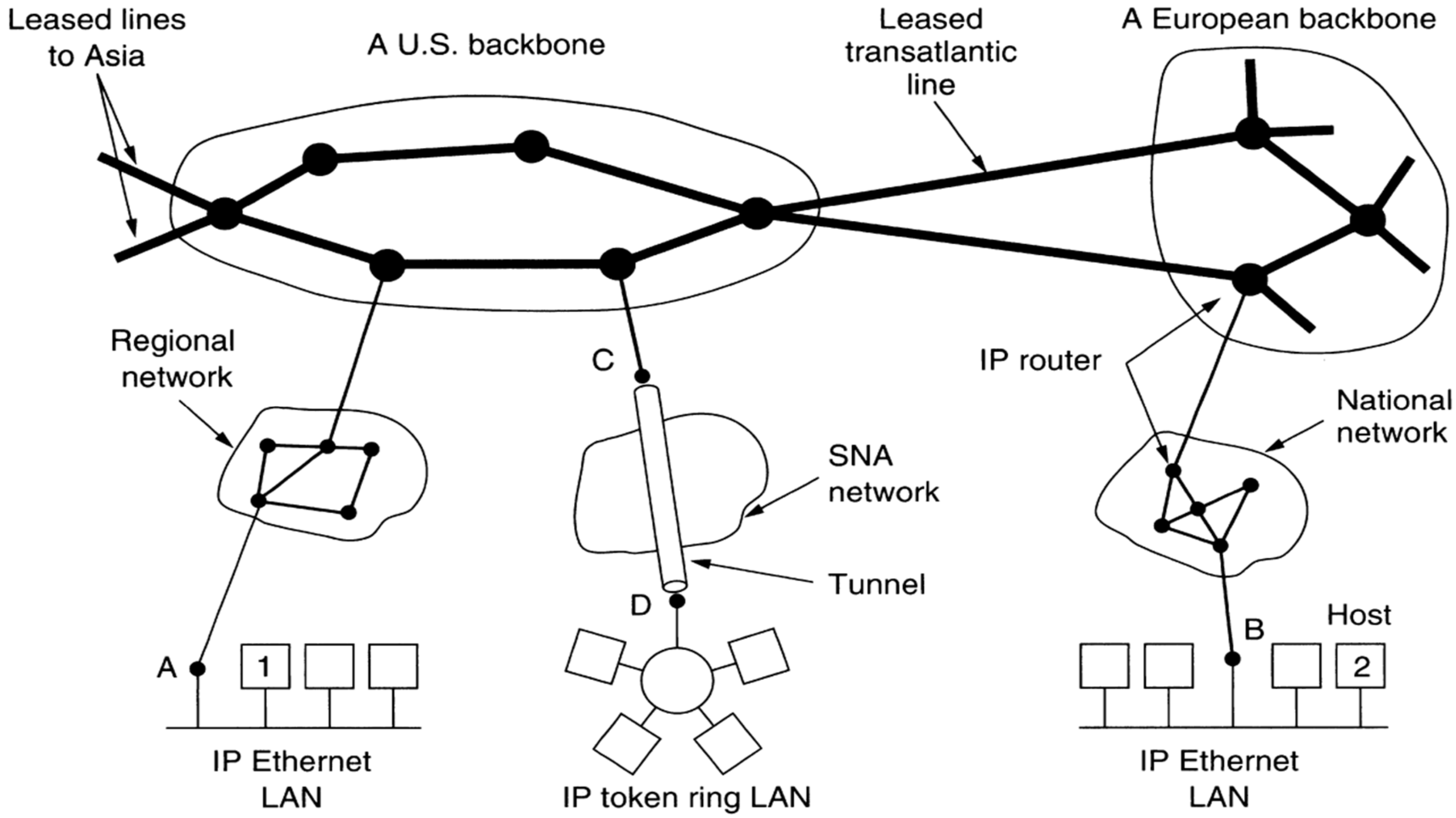


- Based on different Internet measurement projects
- Uses approach of global network positioning
- Advanced latency models
  - Dynamic latency model
    - Static part based on CAIDA
    - Dynamic part based on probability distribution derived from PingER
  - PingER-based latency model
  - Analytical latency model based on the haversine formula
- Packet loss depending on the geographical positions
- Supporting UDP and simplified TCP


Embedding into Euclidean space
End-to-end-link jiiter distribution
Host set with GeoData
CAIDA
MaxMind
$H_1 = <(x_1, \ldots x_n), GeoData>$
⋮
$H_N = <(x_1, \ldots x_n), GeoData>$
60 ms
H_1
H_2
PingER
60 ms
Static part
Random part
$Delay(H_1, H_2) = \frac{MinimumRTT}{2} + Jitter$
Simulation Framework

Technical System

Impact of the Heterogeneous Internet



Source: Andrew Tanenbaum. Computer Networks. Prentice Hall Professional Technical Reference, 2002.

Entities/Attributes of Potential Interest

End-to-end systems

- Geographic location
- Available upload/download bandwidth

Intermediate router(s)

- Utilization/load

Overlay messages

- IP-Packets
- Size

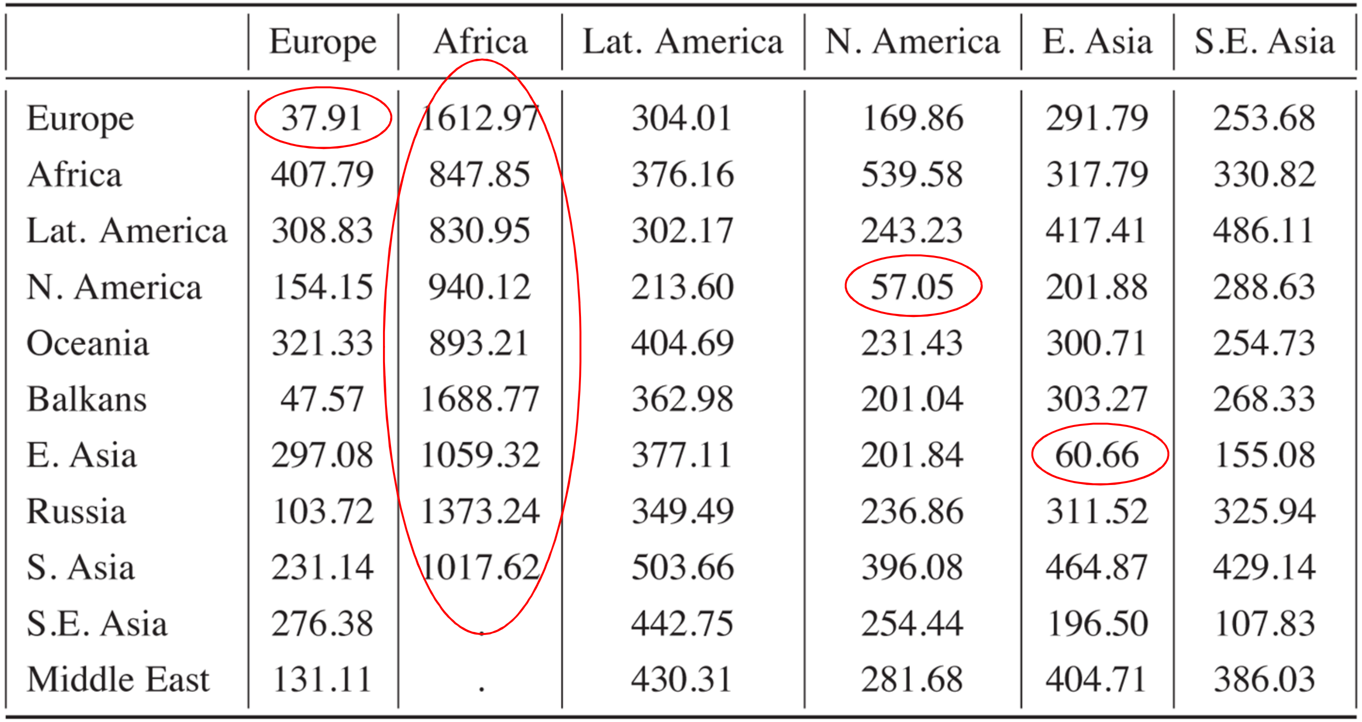
Physical links

- Bandwidth
- Packet loss probability

The Influence of the Geographical Position

IEPM PingEr Project

- ~ 40 monitor hosts and 670 destination hosts / ~ 960 RTTs per link per day
- aggregated RTTs, RTT variation for inter and intra country and region links



	Europe	Africa	Lat. America	N. America	E. Asia	S.E. Asia
Europe	37.91	1612.97	304.01	169.86	291.79	253.68
Africa	407.79	847.85	376.16	539.58	317.79	330.82
Lat. America	308.83	830.95	302.17	243.23	417.41	486.11
N. America	154.15	940.12	213.60	57.05	201.88	288.63
Oceania	321.33	893.21	404.69	231.43	300.71	254.73
Balkans	47.57	1688.77	362.98	201.04	303.27	268.33
E. Asia	297.08	1059.32	377.11	201.84	60.66	155.08
Russia	103.72	1373.24	349.49	236.86	311.52	325.94
S. Asia	231.14	1017.62	503.66	396.08	464.87	429.14
S.E. Asia	276.38	.	442.75	254.44	196.50	107.83
Middle East	131.11	.	430.31	281.68	404.71	386.03

Region-to-region average round-trip time in ms (November 2007)

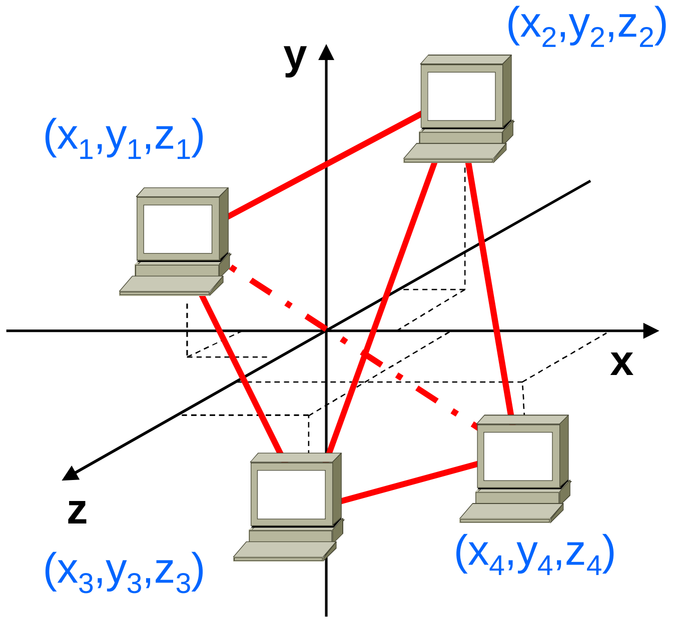
Static Part: Global Network Positioning

Model the Internet as an d-dimensional geometric space

Characterize the position of any end host with coordinates

RTT prediction

- Use computed distances to predict actual distances

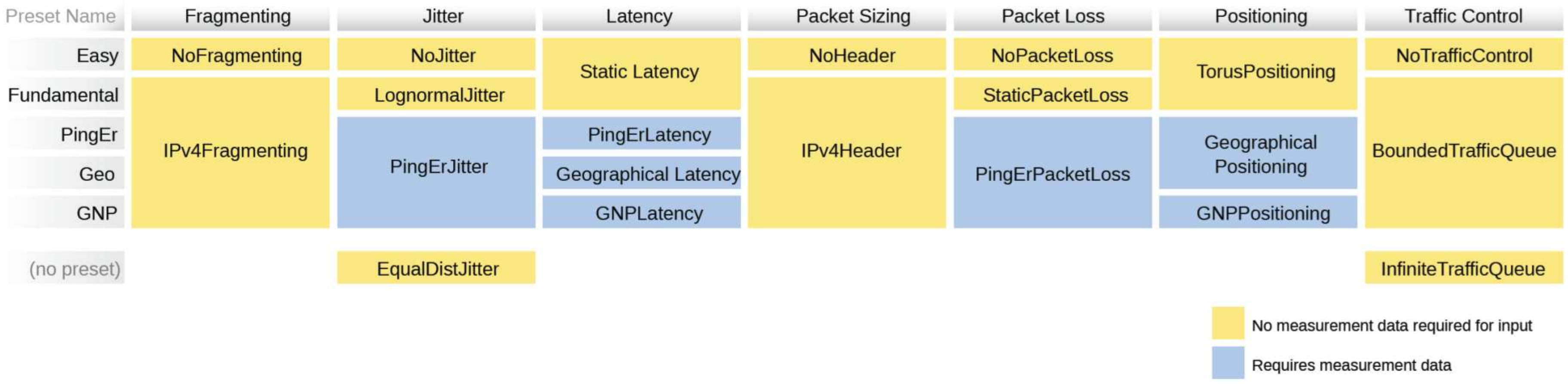


→ In the file measured_data.xml

Modular Network Layer

Configurable details of network layer

- Allows to have realistic network, but slow
- Or fast network simulation, but less realistic



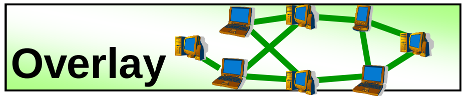
Preset Name	Fragmenting	Jitter	Latency	Packet Sizing	Packet Loss	Positioning	Traffic Control
Easy	NoFragmenting	NoJitter	Static Latency	NoHeader	NoPacketLoss	TorusPositioning	NoTrafficControl
Fundamental	IPv4Fragmenting	LognormalJitter		IPv4Header	StaticPacketLoss		BoundedTrafficQueue
PingEr		PingErJitter	PingErLatency		PingErPacketLoss	Geographical Positioning	
Geo			Geographical Latency				
GNP			GNPLatency			GNPPositioning	
(no preset)		EqualDistJitter					InfiniteTrafficQueue

🟨 No measurement data required for input

🟦 Requires measurement data

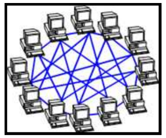
→ In the file mod_measured_data.xml

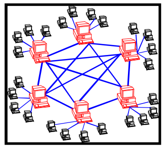
Overlay Layer



Unstructured overlays

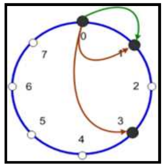
- Gnutella 0.4
- Hierarchical overlays
  - Gnutella 0.6
  - Gia

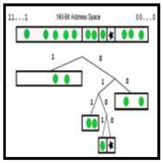


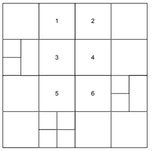


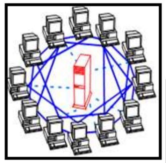
Distributed Hash Tables

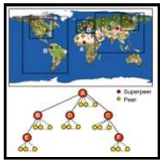
- Chord
- Kademlia
  - Pure
  - Kandy
  - KAD
  - Hierarchical Kademlia
- CAN
- Centralized Hash Table
- Globase.KOM





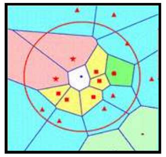




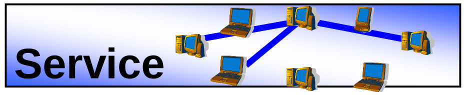


Information Dissemination Overlays

- VON

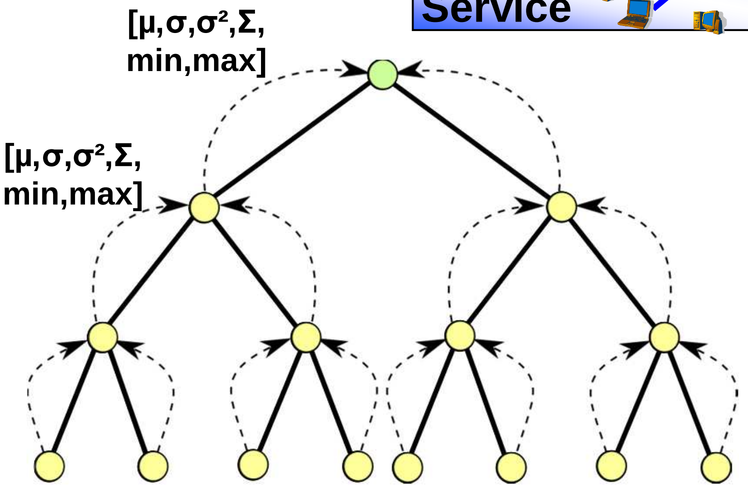


Service Layer



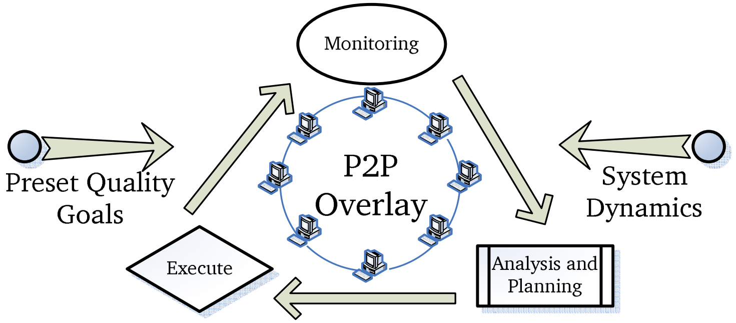
Monitoring

- SkyEye.KOM
  - Applicable on DHTs
  - Tree topology for data collection and dissemination
  - Statistical representation of the P2P system



Management

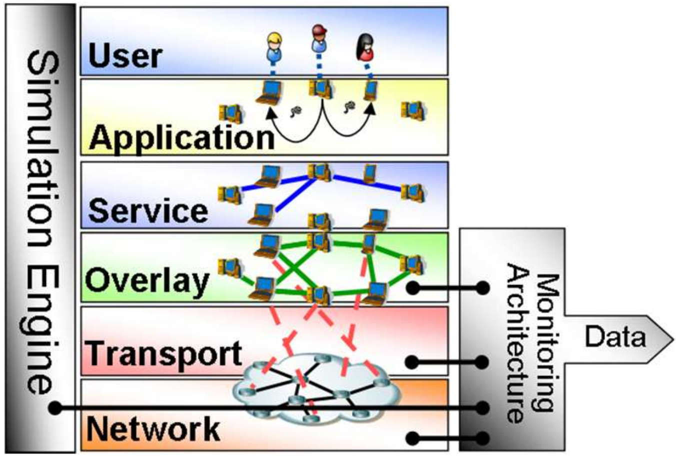
- SkyNet.KOM
  - Based on SkyEye.KOM
  - Supports capacity-based peer-search
  - Maintains the P2P system based on given constraints
    - Adapting the parameters of the system to meet the preset goals



Additional Components

Monitoring Architecture

- Integrated Pub/Sub system for collecting data
  - Network traffic and type
  - Arrival and departure behavior
  - KBR-relevant information
  - Simulator-specific information

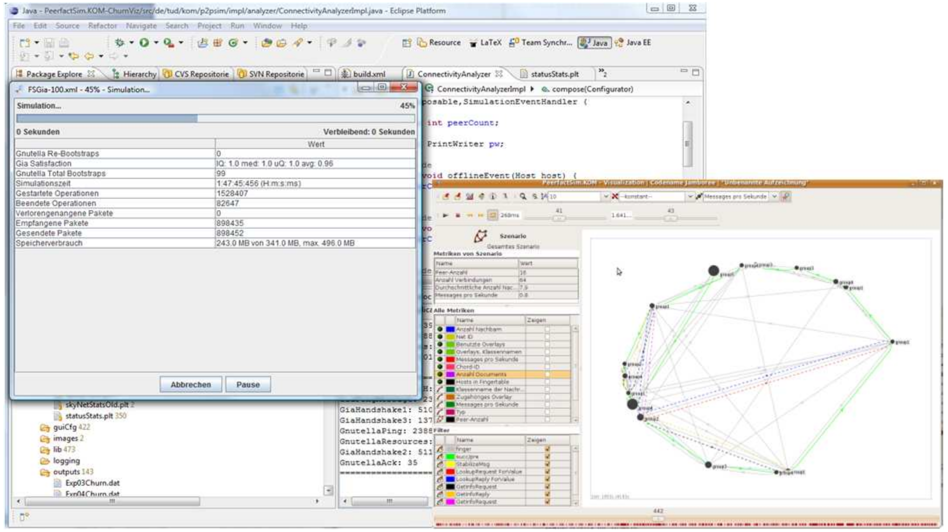


Churn

- Different models for simulating the arrival and departure of peers
  - KAD churn model
  - Exponential churn model

Visualization

- Graphical representation of running simulations
- Visualization of recorded simulations


Java - PeerfactSim.KOM-ChurnViz/src/de/tud/kom/p2psim/impl/analyzer/ConnectivityAnalyzerImpl.java - Eclipse Platform
File Edit Source Refactor Navigate Search Project Run Window Help
Resource LaTeX Team Synchr... Java Java EE
Package Explore Hierarchy CVS Repositorie SVN Repositorie build.xml ConnectivityAnalyzer statusStats.plt
FSGia-100.xml - 45% - Simulation...
Abbrechen Pause

Simulation... 45%

0 Sekunden Verbleibend: 0 Sekunden

	Wert
Gnutella Re-Bootstraps	0
Gia Satisfaction	lQ: 1.0 med: 1.0 uQ: 1.0 avg: 0.96
Gnutella Total Bootstraps	99
Simulationszeit	1:47:45:456 (H:m:s:ms)
Gestartete Operationen	1528407
Beendete Operationen	82647
Verlorengenangene Pakete	0
Empfangene Pakete	898435
Gesendete Pakete	898452
Speicherverbrauch	243.0 MB von 341.0 MB, max. 496.0 MB

Future Work

Integration with benchmarking platform

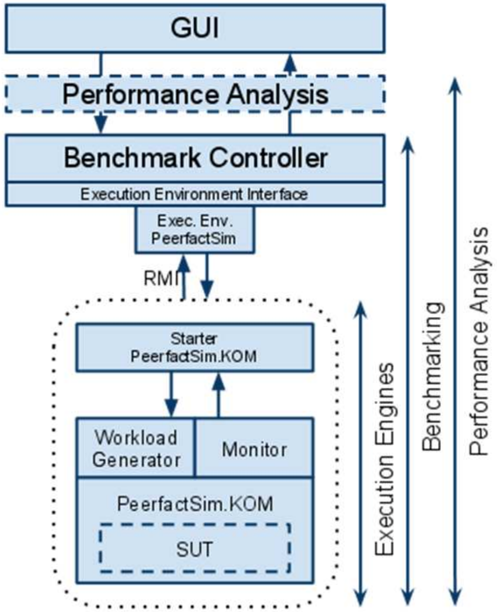
- Remote configuration via Benchmarking Controller
- Communication between the two entities
  - Periodic delivery of results
  - Adaption of the workload by the controller

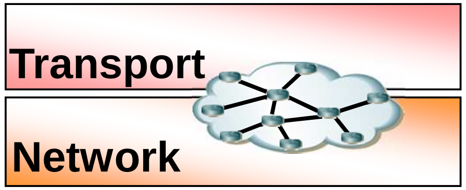
At the underlay

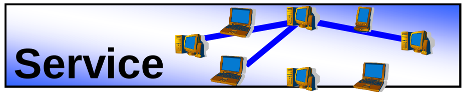
- Network Address Translation
  - Central solution with server
  - Distributed solution

At the service layer

- Different monitoring approaches
  - Gossip-based solutions
  - Central solution

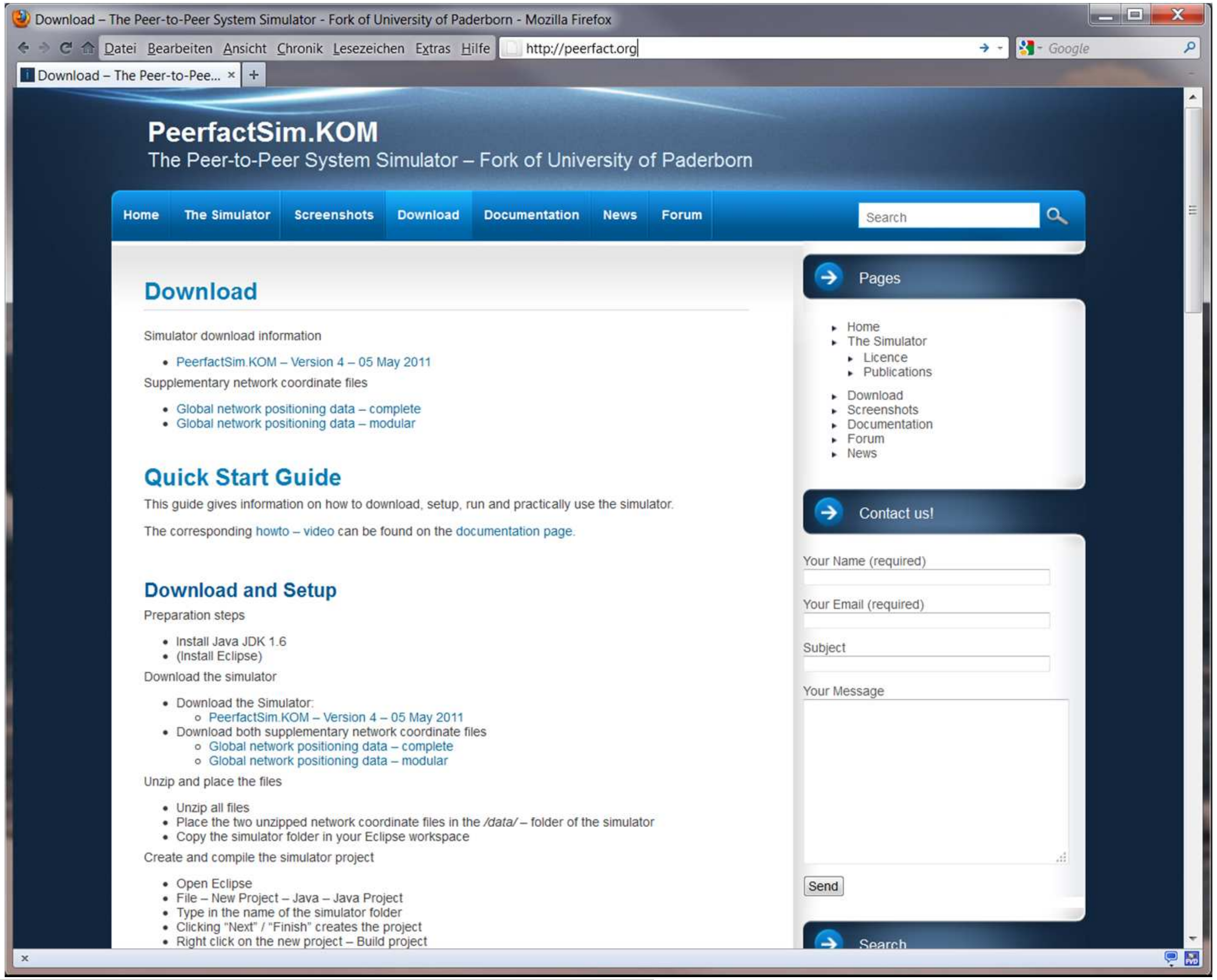






3 How to Use PeerfactSim.KOM – A Step by Step Guide

→ Up-to-date guide at www.peerfact.org


Download – The Peer-to-Peer System Simulator - Fork of University of Paderborn - Mozilla Firefox
Datei Bearbeiten Ansicht Chronik Lesezeichen Extras Hilfe
http://peerfact.org
Google
Download – The Peer-to-Pee... ×
Home
The Simulator
Screenshots
Download
Documentation
News
Forum
Search
Send

PeerfactSim.KOM

The Peer-to-Peer System Simulator – Fork of University of Paderborn

Download

Simulator download information

- PeerfactSim.KOM – Version 4 – 05 May 2011

Supplementary network coordinate files

- Global network positioning data – complete
- Global network positioning data – modular

Quick Start Guide

This guide gives information on how to download, setup, run and practically use the simulator.

The corresponding howto – video can be found on the documentation page.

Download and Setup

Preparation steps

- Install Java JDK 1.6
- (Install Eclipse)

Download the simulator

- Download the Simulator:
  - PeerfactSim.KOM – Version 4 – 05 May 2011
- Download both supplementary network coordinate files
  - Global network positioning data – complete
  - Global network positioning data – modular

Unzip and place the files

- Unzip all files
- Place the two unzipped network coordinate files in the */data/* – folder of the simulator
- Copy the simulator folder in your Eclipse workspace

Create and compile the simulator project

- Open Eclipse
- File – New Project – Java – Java Project
- Type in the name of the simulator folder
- Clicking "Next" / "Finish" creates the project
- Right click on the new project – Build project

Pages

- Home
- The Simulator
  - Licence
  - Publications
- Download
- Screenshots
- Documentation
- Forum
- News

Contact us!

Your Name (required)

Your Email (required)

Subject

Your Message

Search

3.1 Downloading and Installation

1. Download Eclipse
   - http://eclipse.org/
2. Download gnuplot
   - http://www.gnuplot.info/
3. Download PeerfactSim.KOM and both network files
   - http://www.peerfact.org

3. OR visit SVN repository

- https://svn-serv.cs.uni-paderborn.de/peerfactsim

4. Copy network measurement files to

- PeerfactSim-Main/data

5. Read the documentation or watch the tutorials

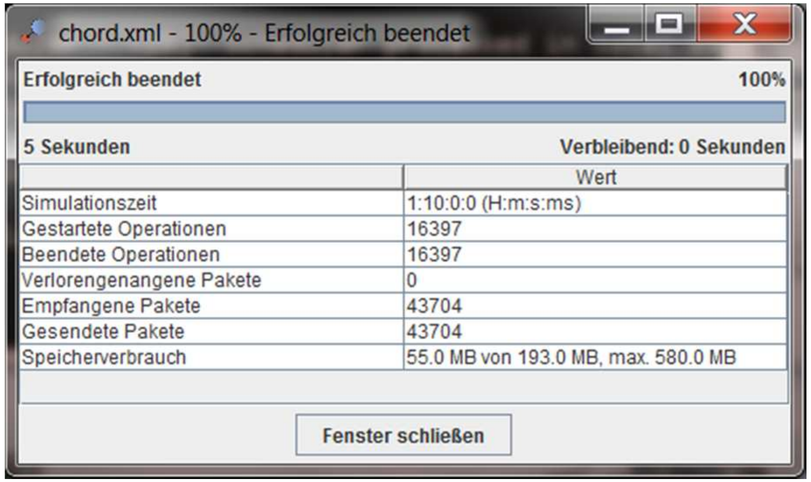
3.2 Running a Simulation

In Eclipse:

- Run as Application: SimulationRunner.java
- Program arguments (example)
  - config/chord2.xml
- VM arguments
  - -Xms200m -Xmx600m

Using the .bat / .sh files: similar

- Start runGui.bat
- Choose a configuration
  - See the visualized ones below
  - /visualization/chord.xml

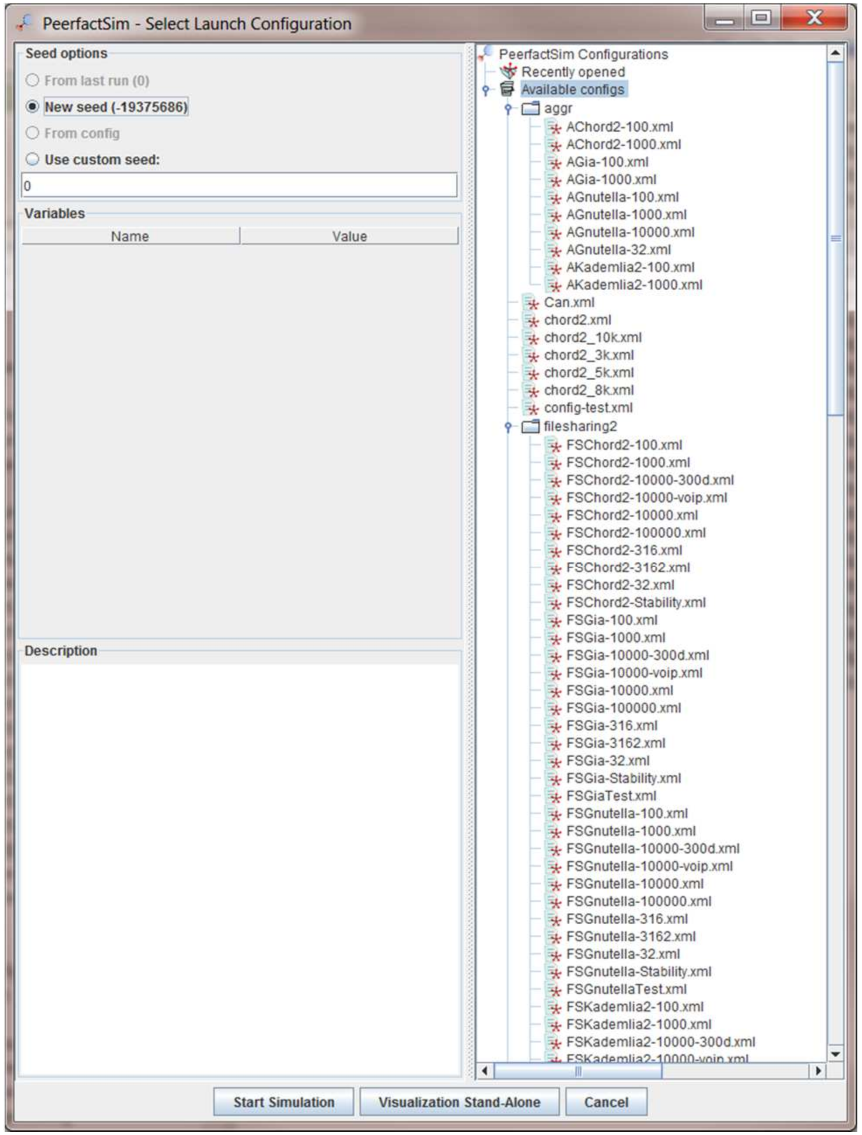


chord.xml - 100% - Erfolgreich beendet

Erfolgreich beendet 100%

5 Sekunden Verbleibend: 0 Sekunden

	Wert
Simulationszeit	1:10:0:0 (H:m:s:ms)
Gestartete Operationen	16397
Beendete Operationen	16397
Verlorengenangene Pakete	0
Empfangene Pakete	43704
Gesendete Pakete	43704
Speicherverbrauch	55.0 MB von 193.0 MB, max. 580.0 MB



PeerfactSim - Select Launch Configuration

Seed options

○ From last run (0)
● New seed (-19375686)
○ From config
○ Use custom seed:

0

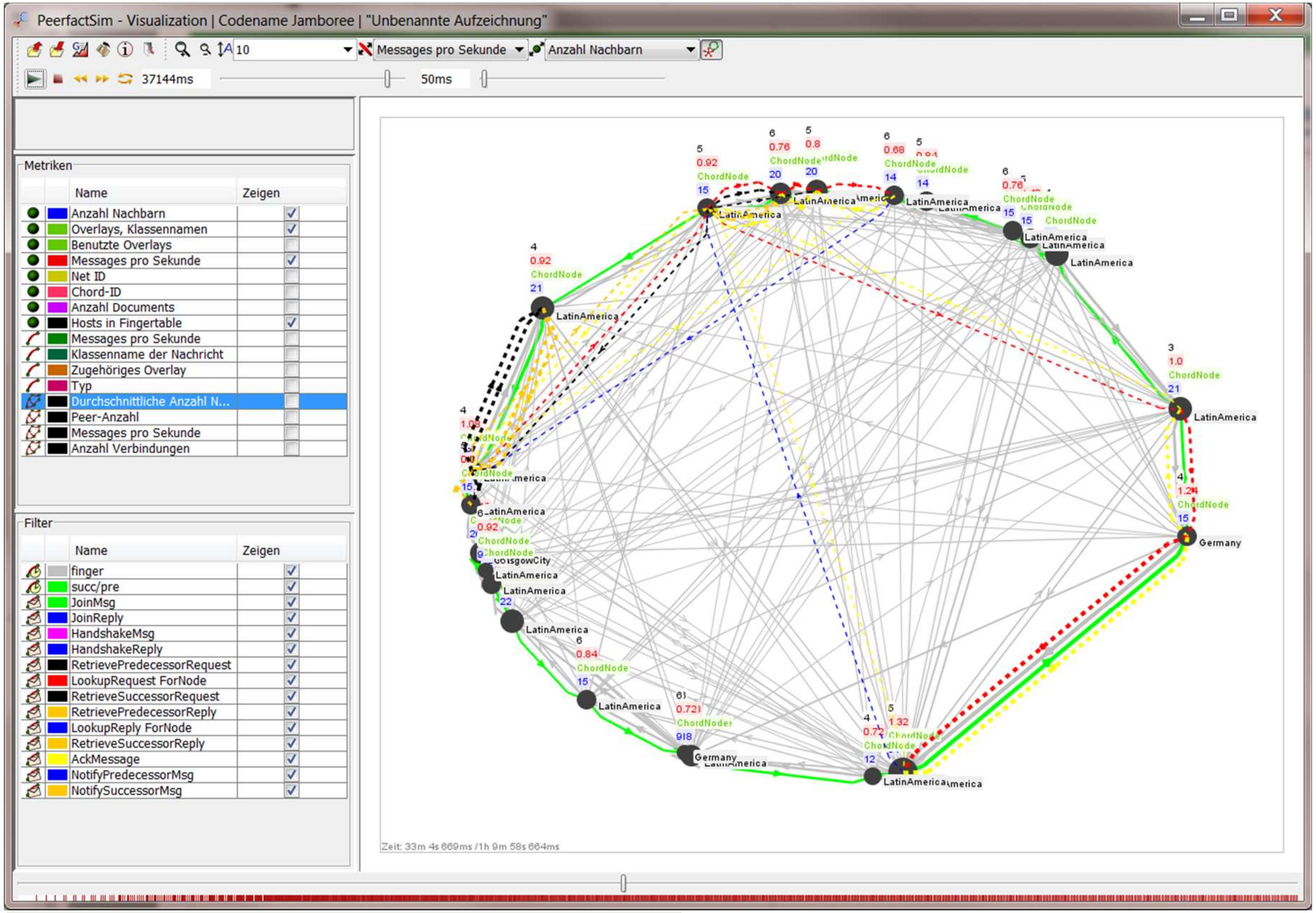
Variables

Name	Value

Description

```
PeerfactSim Configurations
  Recently opened
  Available configs
    aggr
      AChord2-100.xml
      AChord2-1000.xml
      AGia-100.xml
      AGia-1000.xml
      AGnutella-100.xml
      AGnutella-1000.xml
      AGnutella-10000.xml
      AGnutella-32.xml
      AKademlia2-100.xml
      AKademlia2-1000.xml
    Can.xml
    chord2.xml
    chord2_10k.xml
    chord2_3k.xml
    chord2_5k.xml
    chord2_8k.xml
    config-test.xml
    filesharing2
      FSChord2-100.xml
      FSChord2-1000.xml
      FSChord2-10000-300d.xml
      FSChord2-10000-voip.xml
      FSChord2-10000.xml
      FSChord2-100000.xml
      FSChord2-316.xml
      FSChord2-3162.xml
      FSChord2-32.xml
      FSChord2-Stability.xml
      FSGia-100.xml
      FSGia-1000.xml
      FSGia-10000-300d.xml
      FSGia-10000-voip.xml
      FSGia-10000.xml
      FSGia-100000.xml
      FSGia-316.xml
      FSGia-3162.xml
      FSGia-32.xml
      FSGia-Stability.xml
      FSGiaTest.xml
      FSGnutella-100.xml
      FSGnutella-1000.xml
      FSGnutella-10000-300d.xml
      FSGnutella-10000-voip.xml
      FSGnutella-10000.xml
      FSGnutella-100000.xml
      FSGnutella-316.xml
      FSGnutella-3162.xml
      FSGnutella-32.xml
      FSGnutella-Stability.xml
      FSGnutellaTest.xml
      FSKademlia2-100.xml
      FSKademlia2-1000.xml
      FSKademlia2-10000-300d.xml
      FSKademlia2-10000-voip.xml
```

3.3 Simulation Visualization (Replay)



PeerfactSim - Visualization | Codename Jamboree | "Unbenannte Aufzeichnung"

10 · Messages pro Sekunde · Anzahl Nachbarn

37144ms · 50ms

Metriken

		Name	Zeigen
●	🟦	Anzahl Nachbarn	☑
●	🟩	Overlays, Klassennamen	☑
●	🟩	Benutzte Overlays	☐
●	🟥	Messages pro Sekunde	☑
●	🟨	Net ID	☐
●	🟥	Chord-ID	☐
●	🟪	Anzahl Documents	☐
●	⬛	Hosts in Fingertable	☑
	🟩	Messages pro Sekunde	☐
	🟩	Klassenname der Nachricht	☐
	🟧	Zugehöriges Overlay	☐
	🟥	Typ	☐
	⬛	Durchschnittliche Anzahl N...	☐
	⬛	Peer-Anzahl	☐
	⬛	Messages pro Sekunde	☐
	⬛	Anzahl Verbindungen	☐

Filter

		Name	Zeigen
	⬜	finger	☑
	🟩	succ/pre	☑
	🟩	JoinMsg	☑
	🟦	JoinReply	☑
	🟪	HandshakeMsg	☑
	🟦	HandshakeReply	☑
	⬛	RetrievePredecessorRequest	☑
	🟥	LookupRequest ForNode	☑
	⬛	RetrieveSuccessorRequest	☑
	🟧	RetrievePredecessorReply	☑
	🟦	LookupReply ForNode	☑
	🟧	RetrieveSuccessorReply	☑
	🟨	AckMessage	☑
	🟦	NotifyPredecessorMsg	☑
	🟧	NotifySuccessorMsg	☑

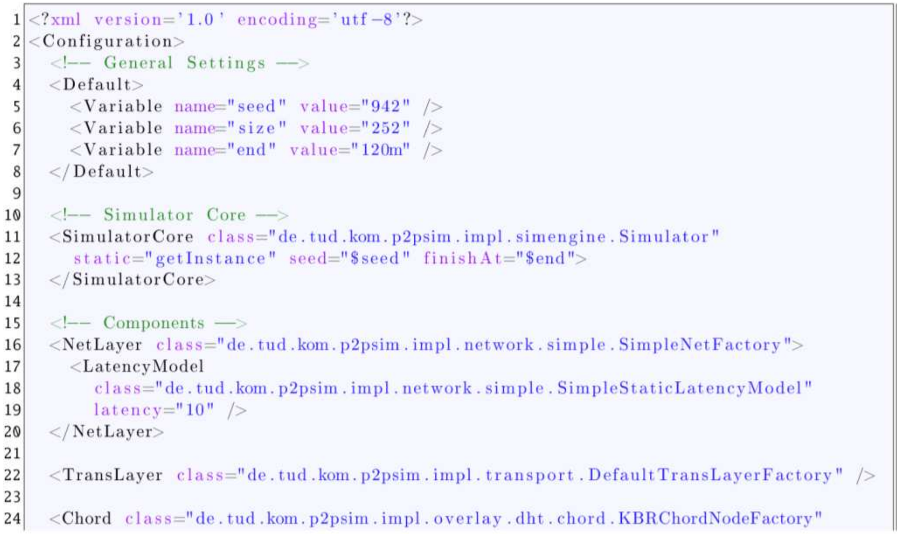
Zeit: 33m 4s 669ms /1h 9m 58s 664ms

3.4 Setting up a first Simulation - the Config - File

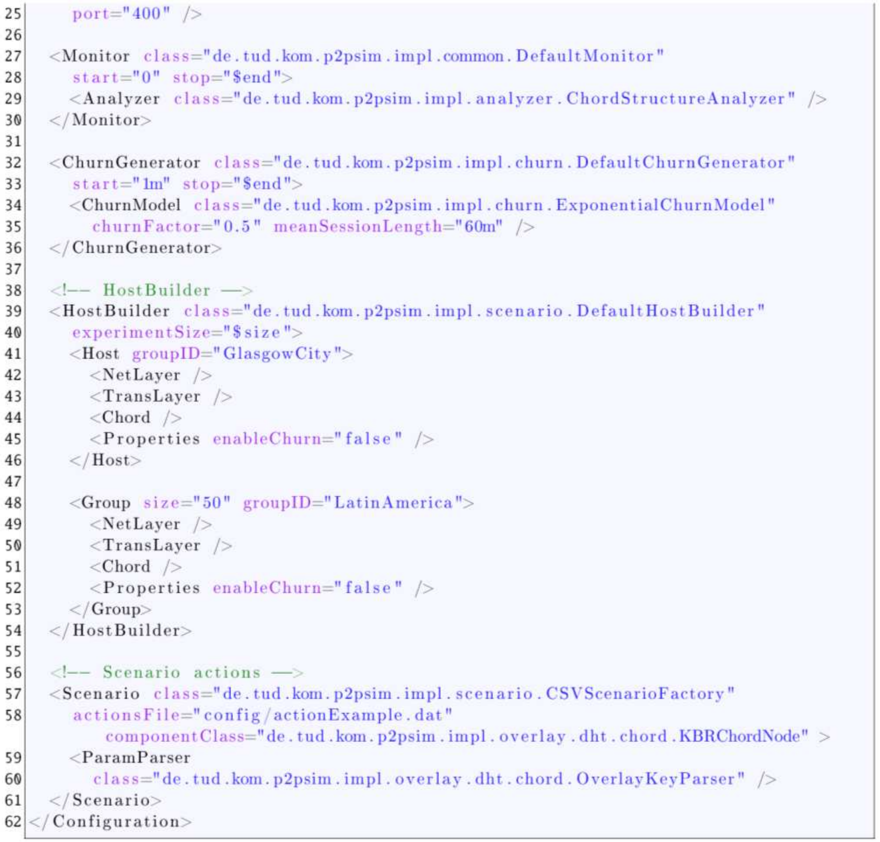
In the folder: /config

Defines

- the components to be simulated
- the action file to use



```xml
<?xml version='1.0' encoding='utf-8'?>
<Configuration>
  <!-- General Settings -->
  <Default>
    <Variable name="seed" value="942" />
    <Variable name="size" value="252" />
    <Variable name="end" value="120m" />
  </Default>

  <!-- Simulator Core -->
  <SimulatorCore class="de.tud.kom.p2psim.impl.simengine.Simulator"
    static="getInstance" seed="$seed" finishAt="$end">
  </SimulatorCore>

  <!-- Components -->
  <NetLayer class="de.tud.kom.p2psim.impl.network.simple.SimpleNetFactory">
    <LatencyModel
      class="de.tud.kom.p2psim.impl.network.simple.SimpleStaticLatencyModel"
      latency="10" />
  </NetLayer>

  <TransLayer class="de.tud.kom.p2psim.impl.transport.DefaultTransLayerFactory" />

  <Chord class="de.tud.kom.p2psim.impl.overlay.dht.chord.KBRChordNodeFactory"
```



```xml
    port="400" />

  <Monitor class="de.tud.kom.p2psim.impl.common.DefaultMonitor"
    start="0" stop="$end">
    <Analyzer class="de.tud.kom.p2psim.impl.analyzer.ChordStructureAnalyzer" />
  </Monitor>

  <ChurnGenerator class="de.tud.kom.p2psim.impl.churn.DefaultChurnGenerator"
    start="1m" stop="$end">
    <ChurnModel class="de.tud.kom.p2psim.impl.churn.ExponentialChurnModel"
      churnFactor="0.5" meanSessionLength="60m" />
  </ChurnGenerator>

  <!-- HostBuilder -->
  <HostBuilder class="de.tud.kom.p2psim.impl.scenario.DefaultHostBuilder"
    experimentSize="$size">
    <Host groupID="GlasgowCity">
      <NetLayer />
      <TransLayer />
      <Chord />
      <Properties enableChurn="false" />
    </Host>

    <Group size="50" groupID="LatinAmerica">
      <NetLayer />
      <TransLayer />
      <Chord />
      <Properties enableChurn="false" />
    </Group>
  </HostBuilder>

  <!-- Scenario actions -->
  <Scenario class="de.tud.kom.p2psim.impl.scenario.CSVScenarioFactory"
    actionsFile="config/actionExample.dat"
        componentClass="de.tud.kom.p2psim.impl.overlay.dht.chord.KBRChordNode" >
    <ParamParser
      class="de.tud.kom.p2psim.impl.overlay.dht.chord.OverlayKeyParser" />
  </Scenario>
</Configuration>
```

Actions File

Describes what happens

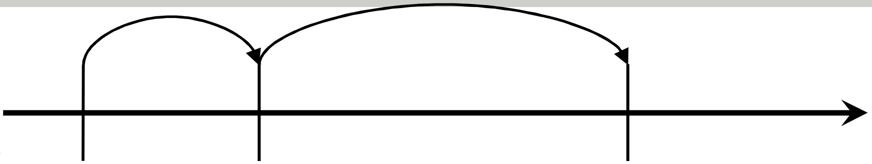
- Joins: who, when
  - Single peer
  - Group of peers

- Specific actions to be done by peers
  - Call operations
  - At specific time

Chord-actions-randomfail.dat

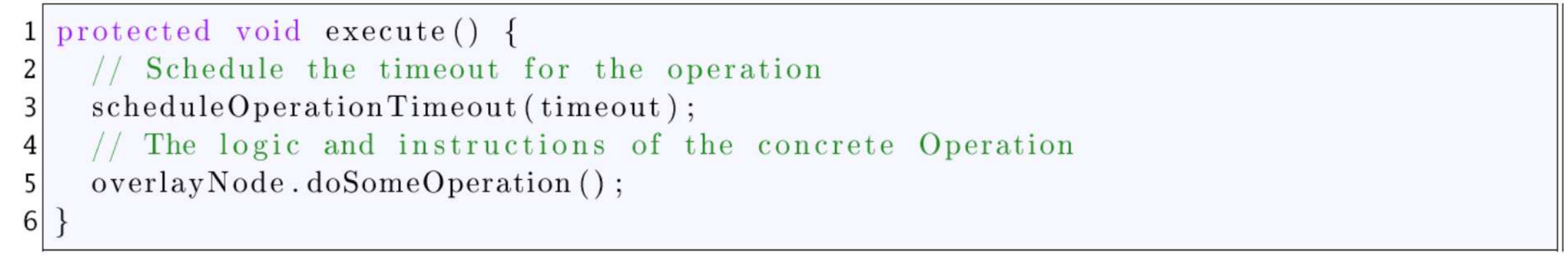
```
#Scenario randomFail

peer1 1m join callback
group1 2m-50m join callback
group2 51m-100m join callback
group3 101m-400m join callback
group6 401m-1000m join callback

peer1 999m store data3 data3 callback
peer1 1000m store data2 data2 callback
peer1 1001m store data1 data1 callback

group1 1020m-1070m valueLookup data1 callback
group2 1070m-1120m valueLookup data2 callback
group2 1120m-1170m valueLookup data3 callback
```

What happens inside

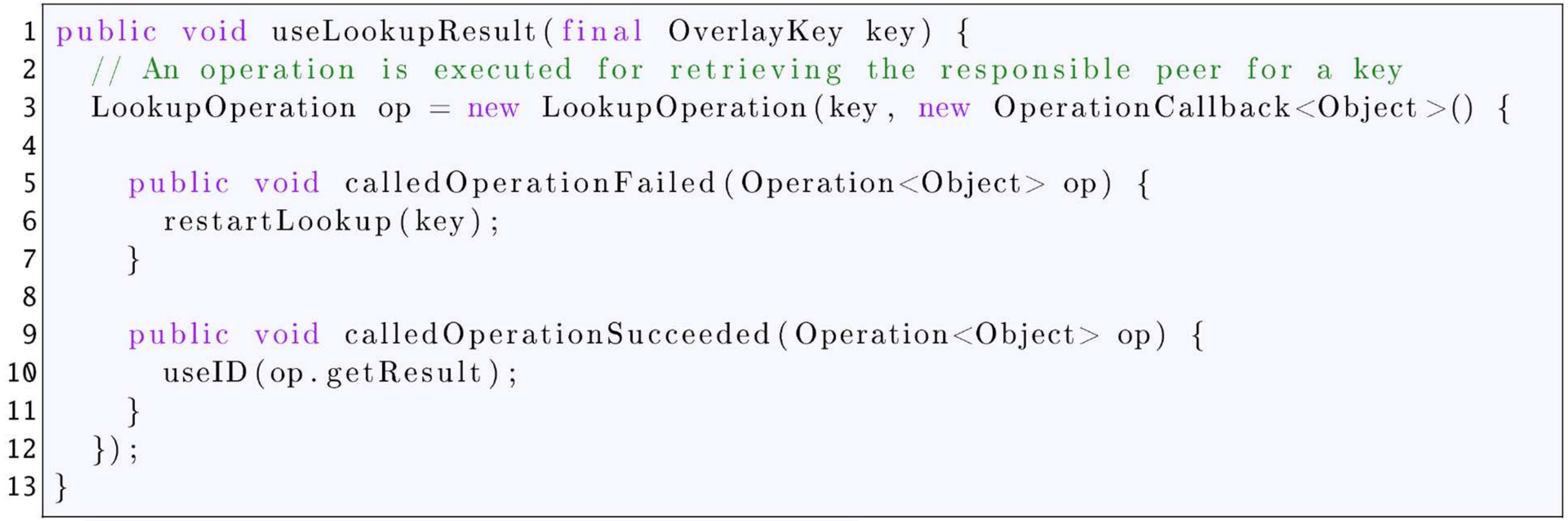


Insides

- Operations are schedulable events
- Events are scheduled for a specific time
- .execute() is called at that time



```
protected void execute() {
  // Schedule the timeout for the operation
  scheduleOperationTimeout(timeout);
  // The logic and instructions of the concrete Operation
  overlayNode.doSomeOperation();
}
```



```
public void useLookupResult(final OverlayKey key) {
  // An operation is executed for retrieving the responsible peer for a key
  LookupOperation op = new LookupOperation(key, new OperationCallback<Object>() {

    public void calledOperationFailed(Operation<Object> op) {
      restartLookup(key);
    }

    public void calledOperationSucceeded(Operation<Object> op) {
      useID(op.getResult);
    }
  });
}
```

Parts of the Code

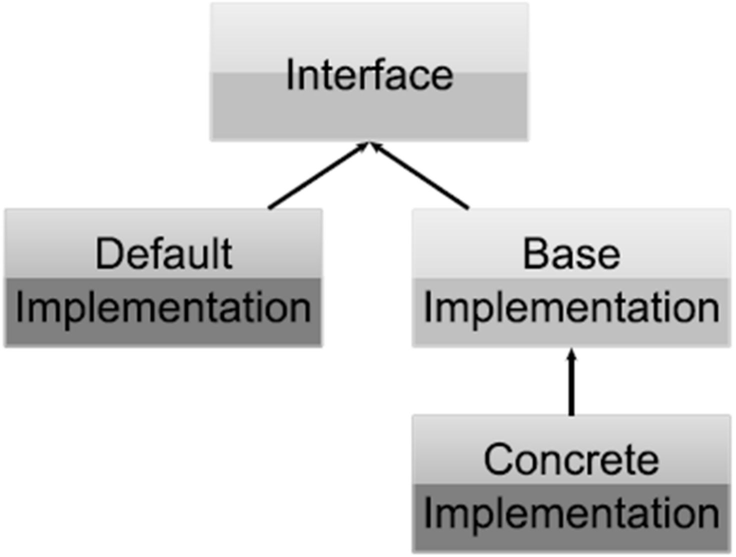
Component Design Pattern

de.tud.kom.p2psim.api

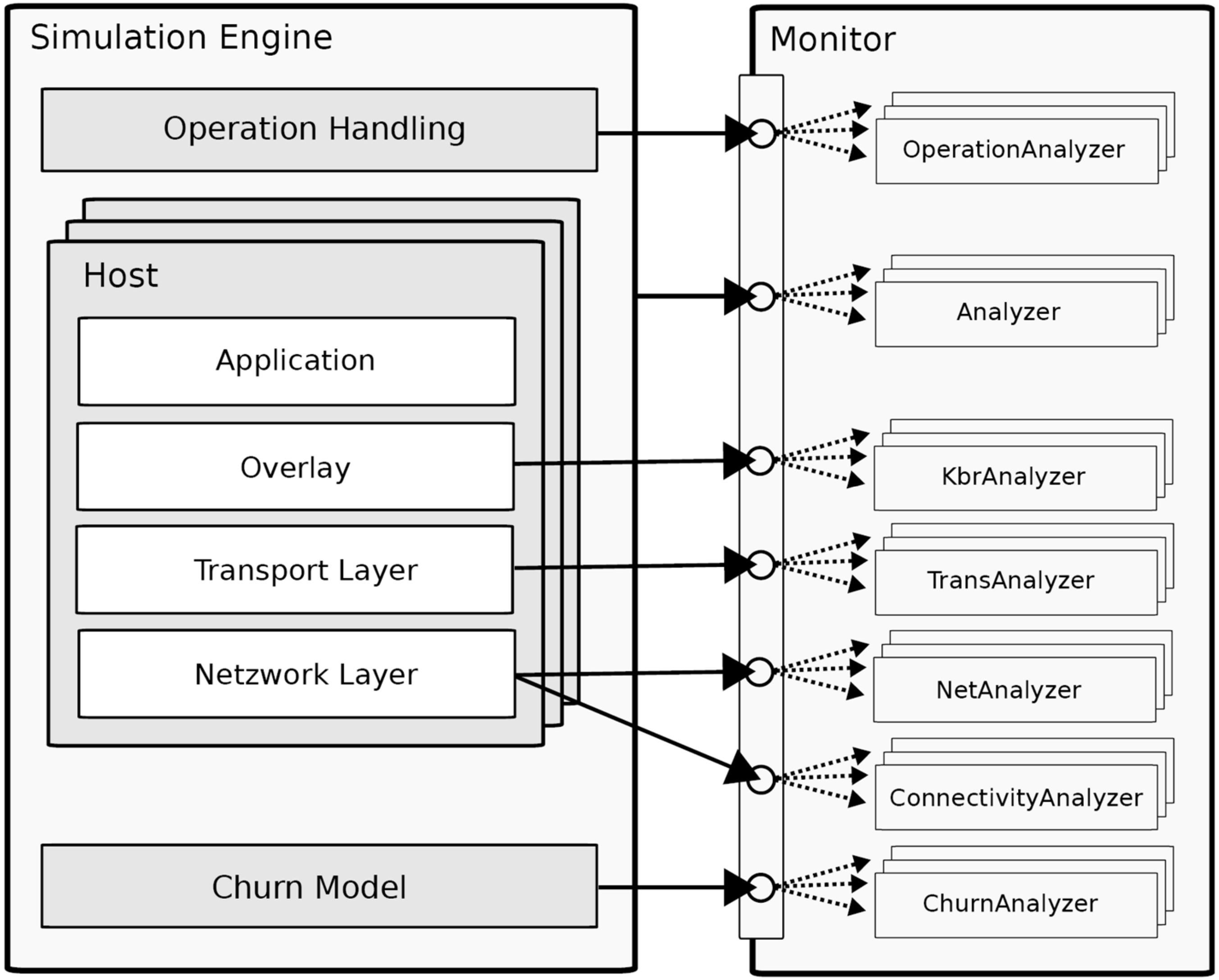
- Interfaces for all components
- Chord: de.tud.kom.p2psim.api.overl

de.tud.kom.p2psim.impl

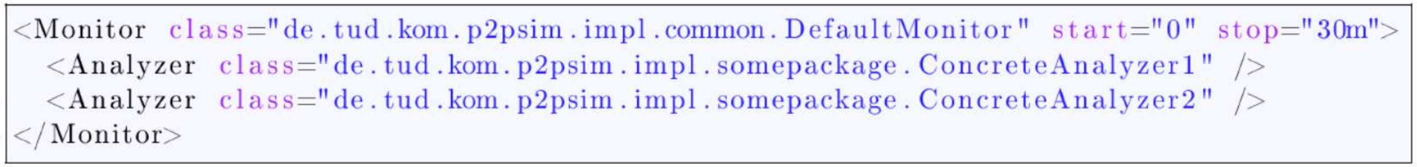
- Basic and specific implementation
- Chord: de.tud.kom.p2psim.impl.ove



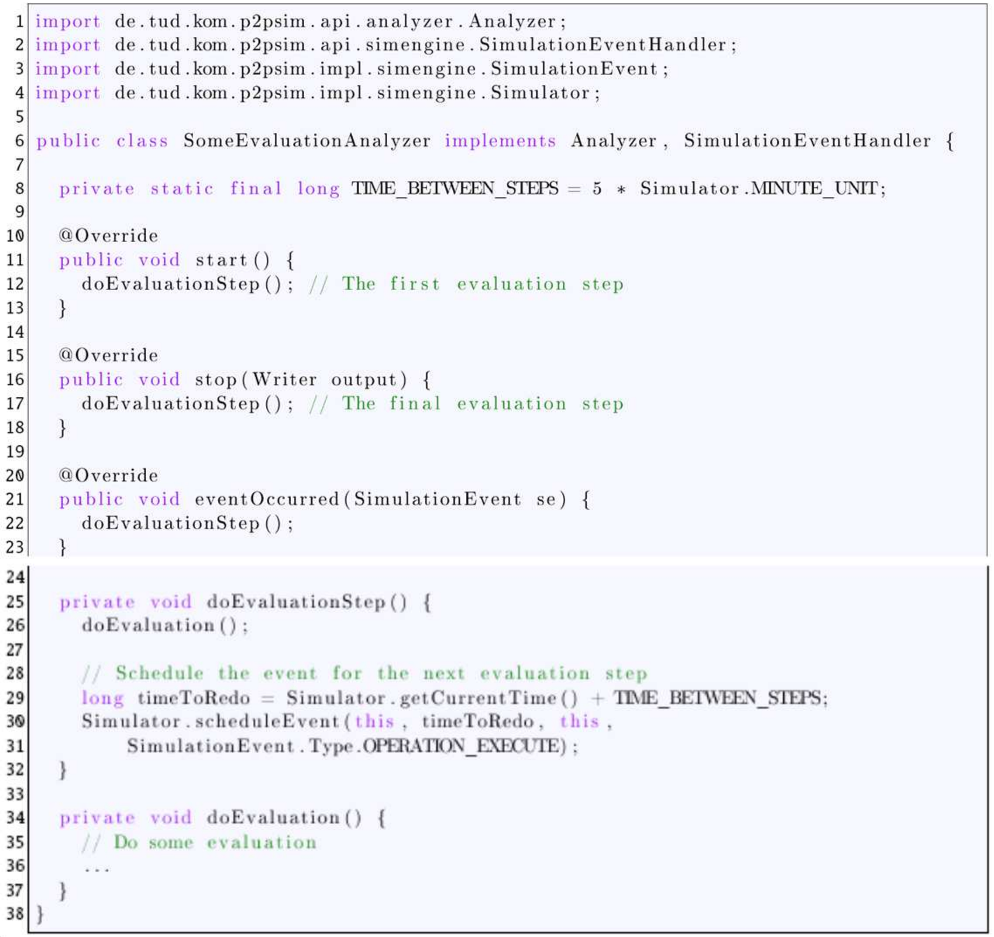
3.5 Observing what is happening → Analyzers



Registering and Using Analyzers



```xml
<Monitor class="de.tud.kom.p2psim.impl.common.DefaultMonitor" start="0" stop="30m">
  <Analyzer class="de.tud.kom.p2psim.impl.somepackage.ConcreteAnalyzer1" />
  <Analyzer class="de.tud.kom.p2psim.impl.somepackage.ConcreteAnalyzer2" />
</Monitor>
```



```java
import de.tud.kom.p2psim.api.analyzer.Analyzer;
import de.tud.kom.p2psim.api.simengine.SimulationEventHandler;
import de.tud.kom.p2psim.impl.simengine.SimulationEvent;
import de.tud.kom.p2psim.impl.simengine.Simulator;

public class SomeEvaluationAnalyzer implements Analyzer, SimulationEventHandler {

  private static final long TIME_BETWEEN_STEPS = 5 * Simulator.MINUTE_UNIT;

  @Override
  public void start() {
    doEvaluationStep(); // The first evaluation step
  }

  @Override
  public void stop(Writer output) {
    doEvaluationStep(); // The final evaluation step
  }

  @Override
  public void eventOccurred(SimulationEvent se) {
    doEvaluationStep();
  }

  private void doEvaluationStep() {
    doEvaluation();

    // Schedule the event for the next evaluation step
    long timeToRedo = Simulator.getCurrentTime() + TIME_BETWEEN_STEPS;
    Simulator.scheduleEvent(this, timeToRedo, this,
        SimulationEvent.Type.OPERATION_EXECUTE);
  }

  private void doEvaluation() {
    // Do some evaluation
    ...
  }
}
```

3.6 Simple Example: Chord Lookup

Setup of the logger: in the config.xml

```
final static Logger log = SimLogger.getLogger(LookupOperation.class);
```

Creating a new Lookup operation

```
public LookupOperation(ChordNode component, ChordID target,
OperationCallback<List<ChordContact>> callback, int lookupId) {

this(component, target, callback);
this.lookupId = lookupId;
}
```

Executing the Lookup event

```
protected void execute() {

// Log the current event
log.debug("start lookup id = " + lookupId + " redo = "+ redoCounter);
if (redoCounter == 0) {
if (ChordConfiguration.DO_CHORD_EVALUATION)
LookupStore.getInstance().registerNewLookup(
masterNode.getLocalChordContact(), lookupId,
Simulator.getCurrentTime());
}

// Start Operation Timer
new OperationTimer(this, ChordConfiguration.OPERATION_TIMEOUT);


// Routing - Protocol
ChordRoutingTable routingTable = masterNode.getChordRoutingTable();
if (routingTable.responsibleFor(target))
…

}
```

Successful Lookup

```
private void analyzeLookupResult(ChordContact responsibleContact,
ChordID targetKey, int lookupOperationID, int hopCount) {

…
// Log the current event
log.debug("incorrect lookup result" + " key = " + targetKey
+ " correct responder " + responder + " found = "+ responsibleContact);

//
LookupStore.getInstance().lookupFinished(lookupOperationID,
Simulator.getCurrentTime(), hopCount, valid);
}
```

In the Analyzer: LookupStore

```
public void lookupFinished(int id, long timeStamp,int hopCount ){

if(! ChordOverlayAnalyzer.lookupStats ){
return;
}
for (LookupProxy lookup : lookupList) {
if (lookup.getLookupID() == id) {
lookup.setEndStatus(LookupProxy.Status.FINISHED);
lookup.setReplyTimestamp(timeStamp);
lookup.setHop(hopCount);
lookup.setValidResult(valid);
return;
}
}
log.error("Lookup is not in store id = " + id);
}
```

3.7 Plotting the results

LookupStore gathers all statistics

```
public double getMeasureValue(String metric, long begin, long end) {
double min = (double) (end - begin) / Simulator.MINUTE_UNIT;

if (Metrics.AverageLookupTimeInSec.equals(Metrics.valueOf(metric))) {
return getAverageLookupTime(begin, end);
}
else if (Metrics.AverageHopsPerLookup.equals(Metrics.valueOf(metric))) {
return getAverageHopsPerLookup(begin, end);
}

…..
```

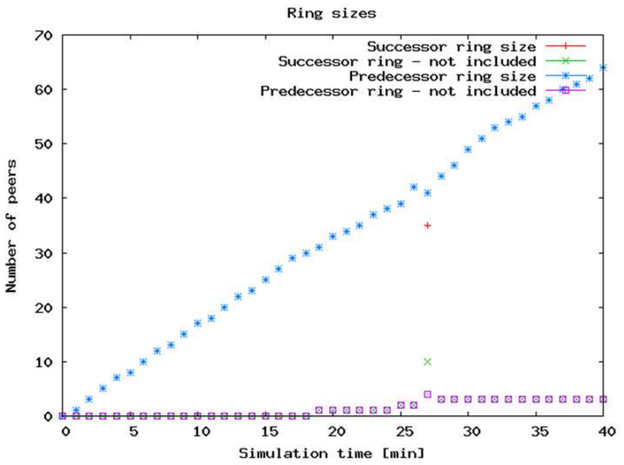
ChordStructurePostProcessor: output in a file

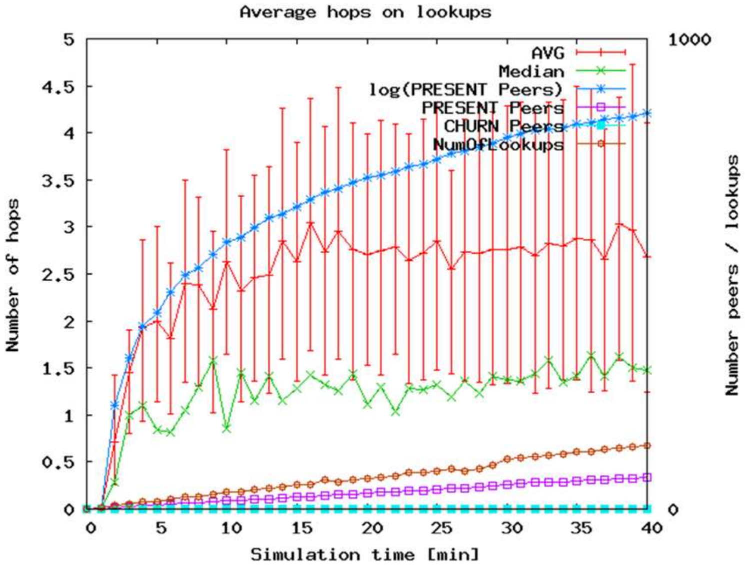
```
/output/results/Chord/2011-05-11_08-28-47_size101_seed500/Structure.dat

#time[sec]
#time[min]
#PRESENT nodes
#TO_JOIN nodes
#CHURN nodes
#Succ ring size
#Succ ring connected?
#Succ num succ ring breaks
#Succ ring connected (using backups)?
#Succ ring includes all?
#Succ num not included nodes
#Pred ring size
#Pred ring connected?
#Pred num pred ring breaks
#Pred ring connected (using backups)?
#Pred ring includes all?
#Pred num not included nodes

0               0          0          101        0
                0          true       0          true
                true       0          0          true
                0          true       true       0

60              1          1          100        0
                1          true       0          true
                true       0          1          true
                0          true       true       0

120             2          3          98         0
                3          true       0          true
                true       0          3          true
                0          true       true       0

180             3          5          96         0
                5          true       0          true
                true       0          5          true
                0          true       true       0

240             4          7          94         0
                7          true       0          true
                true       0          7          true
                0          true       true       0

5               8          93         0          8
                true       0          true       true
                0          8          true       0
                true       true       0
…
```

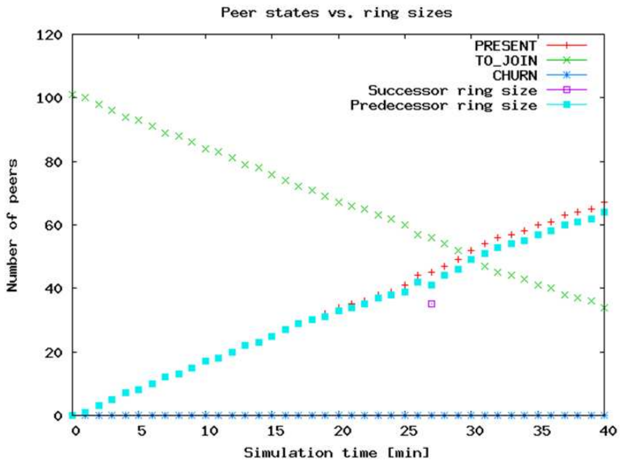
3.8 Using GnuPlot

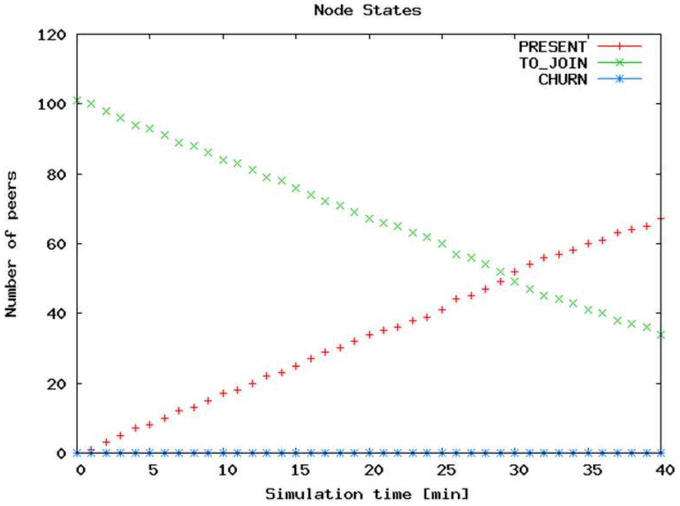
Files: /output/gnuplotScripts

- chord2_structure_complexity.plt









Thanks for Your Attention

?