PeerfactSim: A Simulation Framework for Peer-to-Peer Systems (and more)

Project Group
NODES - Offering Dynamics to Emerging Structures
www.peerfact.org

Dr.-Ing. Kalman Graffi
Email: graffi@mail.upb.de
Fachgruppe Theorie verteilter Systeme
Fakultät für Elektrotechnik, Informatik und Mathematik
Universität Paderborn
Fürstenallee 11, D-33102 Paderborn, Deutschland
Tel.+49 5251 606730, Fax. +49 5251 606697
http://www.cs.uni-paderborn.de/fachgebiete/fg-ti.html
Overview

1 General Evaluation Methods
2 Overview on PeerfactSim
3 How to Use PeerfactSim – 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?

**DEVELOPMENT**
- **Specification**
  - What to make?
- **Review of the State of the Art**
  - What is already done?
- **Design**
  - How to make it?
- **Solution**
  - Inventing!
- **Implementation**
  - Make it!
- **Testing**
  - Does it work?

**Evaluation**
- Does the idea work?

Hard, risky, slow!
Simplified Overview on Simulations

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

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

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 interdependencies
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

![Diagram of Underlay and GNP Network Layer]

- Embedding into Euclidean space
- End-to-end-link jitter distribution
- PingER
- Simulation Framework
Impact of the Heterogeneous Internet

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

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

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

→ 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
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 – A Step by Step Guide
3.1 Downloading and Installation

1. Download Eclipse
   - http://eclipse.org/

2. Download gnuplot
   - http://www.gnuplot.info/

3. Download Subclipse
   - http://subclipse.tigris.org/update_1.6.x

3. Create SVN repository
   - https://svn-serv.cs.uni-paderborn.de/peerfactsim
   - Your branch: PeerfactSim-Main / branches / SS11 / PG-Nodes / trunk

4. Checkout the trunk

5. Checkout network measurement files and documentation
   - https://svn-serv.cs.uni-paderborn.de/peerfactsim/PeerfactSim-Documents

6. Copy network measurement files to
   - PeerfactSim-Main/settings/network-measurements

7. Read the file PeerfactSim-2011-Documentation.pdf
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
3.3 Simulation Visualization (Replay)
3.4 Setting up a first Simulation - the Config - File

In the folder: /config

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

```xml
<Configuration>
  <Default>
    <Variable name="seed" value="942" />
    <Variable name="size" value="256" />
    <Variable name="end" value="1200" />
  </Default>

  <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="1000" />
    </LatencyModel>
  </Components>

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

  <Chord class="de.tud.kom.p2psim.impl.overlay.dht.chord.KBBChordNodeFactory">
    port="4000" />
  </Chord>

  <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="1000" stop="$end">
    <ChurnModel class="de.tud.kom.p2psim.impl.churn.ExponentialChurnModel">
      churnFactor="0.5" meanSessionLength="60000" />
  </ChurnGenerator>

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

  <Group size="50" groupID="Latin America">
    <NetLayer />
    <TransLayer />
    <Chord />
    <Properties enableChurn="false" />
  </Group>

  <HostBuilder />

  <Scenario actions="">
    <Scenario class="de.tud.kom.p2psim.impl.csv.CSVScenarioFactory">
      actionsFile="/config/actionExample.dat" componentClass="de.tud.kom.p2psim.impl.overlay.dht.chord.KBBChordNode">
      <ParamParser class="/config/actionExample.dat" componentClass="de.tud.kom.p2psim.impl.overlay.dht.chord.OverlayKeyParser" />
    </Scenario>
  </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

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

```java
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

**org.peerfact.api**
- Interfaces for all components
- Chord: org.peerfact.api.overlay.dht

**org.peerfact.impl**
- Basic and specific implementation
- Chord: org.peerfact.impl.overlay.dht.chord
3.5 Observing what is happening → Analyzers

Simulation Engine
- Operation Handling
- Host
  - Application
  - Overlay
  - Transport Layer
  - Network Layer
- Churn Model

Monitor
- OperationAnalyzer
- Analyzer
- KbrAnalyzer
- TransAnalyzer
- NetAnalyzer
- ConnectivityAnalyzer
- ChurnAnalyzer
Registering and Using Analyzers

```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**

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

**Creating a new Lookup operation**

```java
public LookupOperation(ChordNode component, ChordID target,
OperationCallback<List<ChordContact>> callback, int lookupId) {
    this(component, target, callback);
    this.lookupId = lookupId;
}
```

**Executing the Lookup event**

```java
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**

```java
private void analyzeLookupResult(ChordContact responsibleContact,
ChordID targetKey, int lookupOperationID, int hopCount) {
    ...
    // Log the current event
    log.debug("incorrect lookup result" + " key = " + targetKey
    + " correct responder " + responsibleContact + 
    " found = " + responsibleContact);
    // LookupStore.getInstance().lookupFinished(lookupOperationID,
    Simulator.getCurrentTime(), hopCount, valid);
}
```

**In the Analyzer: LookupStore**

```java
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

```

#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 num not included nodes
#Pred ring size #Pred ring connected? #Pred num pred ring breaks #Pred ring connected (using backups)? #Pred num not included nodes

<p>| | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>101</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>true</td>
<td>0</td>
<td>true</td>
<td>0</td>
<td>true</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>true</td>
<td>0</td>
<td>false</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>100</td>
<td>0</td>
<td>true</td>
<td>0</td>
<td>false</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>true</td>
<td>0</td>
<td>true</td>
<td>0</td>
<td>true</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>true</td>
<td>0</td>
<td>false</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>true</td>
<td>0</td>
<td>false</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>true</td>
<td>0</td>
<td>false</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>true</td>
<td>98</td>
<td>0</td>
<td>true</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>3</td>
<td>true</td>
<td>98</td>
<td>0</td>
<td>true</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>true</td>
<td>98</td>
<td>0</td>
<td>true</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>3</td>
<td>true</td>
<td>98</td>
<td>0</td>
<td>true</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>true</td>
<td>96</td>
<td>0</td>
<td>true</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>5</td>
<td>true</td>
<td>96</td>
<td>0</td>
<td>true</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>true</td>
<td>94</td>
<td>0</td>
<td>true</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>7</td>
<td>true</td>
<td>94</td>
<td>0</td>
<td>true</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>true</td>
<td>93</td>
<td>0</td>
<td>true</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>93</td>
<td>true</td>
<td>8</td>
<td>true</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>true</td>
<td>93</td>
<td>0</td>
<td>true</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>
3.8 Using GnuPlot

Files: /output/gnuplotScripts
- chord2_structure_complexity.plt
On proper Simulations / Benchmarking

1. Application Class
2. Functional Requirements
3. System Description
4. Functional characteristics to benchmark

- Workload Model
- Metrics

5. Test Generation
6. Measurements of real applications

7. Variances in workload/setup in order to determine quality of system
8. Testing Methodology
9. Testing tool

10. Performing tests on system
11. Testing
12. Mechanism, Systems, Solutions

13. Weighting of results (depends on perspective/audience)
14. Generation of Benchmarks, Conclusions
15. Evaluation

16. Improvement
17. Optimisation
Benchmarking

**Benchmark :=**
- Test (-set), developed to compare the performance of one computer system with the performance of an other

**Elements**
- System under Test (SUT)
- Test environment
- Basis benchmarking scenario
- Load model
- Metrics
- Testing approach
- Aggregation method
Benchmarking - Metrics

**Macro metric**
- On application level
- Relevant for rating the quality of a mechanism

**Micro metric**
- Explains results of macro metrics
- Necessary for scientific analyses
Benchmarks - Workload

Application model
- Models the application functions
- E.g. BitTorrent, File sharing

User model
- E.g. Service usage pattern, service popularity
- Malicious behaviour

Churn model
- Join and leave patterns of nodes
- Life time distribution

Service model
- Type of service (e.g. Streaming),
- E.g. file size, initial distribution of files,
  i.e. characteristics of the specific files and services

Synthetic vs. realistic models
Benchmarking – P2P Characteristics

SUT
- Consists of a large number of heterogeneous entities
- Entities generating load for each other, they are not independent

Churn influences load and system
- Load for maintenance and application are influence differently

How to test a large scale system?
- Without a sufficiently large testbed?

Benchmark gives insight on
- Influence of various user types …
- Of varying peer capacities …
- Of varying application specifics …
- … on the quality of
  - a single mechanism
  - interdependent mechanism

Evaluation of p2p mechanisms require
N > 10,000 nodes
Questions?