D-P2P-Sim: A Distributed P2P Simulator

Sioutas S.\textsuperscript{1} Papaloukopoulos G.\textsuperscript{2} Sakkopoulos E.\textsuperscript{2} Tsichlas K.\textsuperscript{3} Manolopoulos Y.\textsuperscript{3}

Department of Informatics, Ionian University

Computer Engineering and Informatics Department, University of Patras

Department of Informatics, Aristotle University

CIKM’09, November 2–6, 2009, Hong Kong, China.
Introduction

Cloud computing and social networking applications are designed to operate over billions of data objects and nodes.

In P2P bibliography the experimental evaluation of new algorithms is done with overlays 5 orders of magnitude smaller than real life systems support.

Scalable algorithms need to be tested by scalable simulators, adopting real life execution environments.

The idea is to design a distributed P2P simulator to overcome the resource limitations posed by the standalone systems. Achieving both scalability and realism at the same time.

We present the **D-P2P-Sim**, a Distributed P2P Simulator.
Intuition

Create an overlay network with millions of peers by partitioning the overlay to several segments and assigning the responsibility of each segment to a separate system.

Each system instantiates, initializes, stores and operates only over a part of the whole overlay network. We call such a system **Application Node**.

The Application Nodes run independently and in parallel communicating with each other via a Remote Method Invocation mechanism.

The Application Nodes’ cluster provides a unique realistic execution environment for the evaluation of the overlay protocols.
**Application Node**

The Application Node (AN)...

- Implements a Message Passing Environment (MPE)
  - A Queue (Network) temporally stores Messages
  - A Thread (Network Monitor) scans the queue and sends notifications
  - A Remote Service (Network Pipe) interconnects the ANs
- Simulates thousands peers’ instances
  - A Thread Pool executes in parallel the peer instances
  - The Main Thread fires up the overlay’s operations
  - The Peers communicates with each other via the MPE
- Collects and reports statistical data
  - The Messages the Peers and the Operations are monitored
  - The evaluation data are collected to Frequency Tables (FT)
  - The Report Service aggregates the per Application Node FT
Application Node’s Tasks

The Application Node performs the following tasks ...

1. Reads the configuration and the simulation files (i.e. config.xml)

2. Initializes the several components of the simulator (i.e. Network Queue, Thread Pool)

3. Registers the remote services (i.e. Network Pipe Service, Report Service)

4. Loads the Protocols (i.e. NBDT.jar, BatonStar.jar)

5. Instantiates the desired number of peers and creates the overlay (i.e. Peer.join())

6. Performs lookup and update operations (i.e. Peer.insertKey())

7. Collects and publishes the protocol’s evaluation report (i.e. NBDT_report.txt)
Figure: Components and simulation flow.
D-P2P-Sim’s Features

The key-features of the D-P2P-Sim are:

- **Realism**, provides a real life execution environment
- **Distributed**, is capable to run over a cluster or LAN
- **Unbiased**, exports protocol independent statistical data
- **Extensible**, implements the plug-in mechanism
- **Efficient**, allows simulations with very large number of peers
- **Practical**, simplifies the simulation process
- **Portable**, is Java based
Components & Responsibilities

The D-P2P-Sim’s key aim is to deliver a code organized carefully in software packages achieving low coupling and high cohesion.

Figure: The D-P2P-Sim’s architectural modules
Components & Responsibilities

Administrative Tools

The Administrative Tools takes over the responsibility to:

- Instantiate and initialize the several parts constituting the simulator
- Control the operation of the Overlay and Network
- Synchronize the overall execution of the simulator
- Extract, collect and report statistical data
Components & Responsibilities

Overlay Network

The Overlay Network is responsible to:

- Instantiate and initialize the overlay peers
- Create the overlay by executing the join operation of the peers
- Execute lookup and update operations for peers and keys
- Monitor the execution of the peers
Components & Responsibilities
Message Passing Environment

The Message Passing Environment simulates the physical underlying network. A Thread continuously runs at the background and ...

- Scans the Message Queue to see if there are any messages waiting to be served
- Notifies the message’s destination peer to wake up and retrieve the message
- Filters out the messages in order to collect statistical data
- Routes out of bounds messages to the right Application Node
Implementing new protocols

The evolution of a new protocol requires the implementation of:

- The peer’s functionalities
- The protocol’s specific messages
- The protocol’s package

Three interfaces are used towards that direction:

- Peer
- Message
- Network
Implementing new protocols
Peer Interface

The abstract class Peer declares methods for:

- Instantiating and initializing a peer
- Joining and leaving the overlay
- Searching and updating keys and peers
- Administrating the peer’s state

Every peer should implement this class.
Implementing new protocols
Message Interface

The class Message declares methods for:

- Instantiating and initializing a message
- Updating the message’s fields (i.e. source, destination)
- Controlling the message’s payload

The payload of every protocol-specific message should implement the abstract class MessageBody which declares a method for:

- Retrieving the message’s type
Implementing new protocols

Network Interface

The class Network declares methods for:

- Storing and retrieving a message to/from the Network-Queue
- Multicasting and broadcasting a single message
Implementing new protocols
Protocol Packaging

All the protocol’s classes should be integrated into a single jar file, where

- The class names should comply to Java’s naming conventions
- The Main-Class attribute of the jar file should point the protocol’s class name that implements the abstract class Peer
Four different metrics are widely used in the P2P bibliography to evaluate the efficiency and the robustness of P2P protocols:

- The number of messages that are being exchanged during the lookup and update operations
- The portion of total number of messages that are being served by each peer
- The size of the Routing Table that each peer stores
- The percentage of correct responses when there are peer failures in the overlay
Protocol Evaluation
Mechanism for Retrieving Statistical Data

The Simulator derives the benefits of the Message Passing Environment to retrieve statistical data for the protocols evaluation. Provides...

- Special message types (i.e. REQ_LOOKUP_KEY, RES_INSERT_KEY, etc)
- Filters and retrieves the values of certain fields
- Stores extracted values into dedicated frequency tables
- Processes and publishes an aggregated report
# Protocol Evaluation

## Message Structure

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
<th>Responsibility</th>
</tr>
</thead>
<tbody>
<tr>
<td>Message ID</td>
<td>The message’s ID</td>
<td>Simulator</td>
</tr>
<tr>
<td>Source ID</td>
<td>The sender’s peer ID</td>
<td>User</td>
</tr>
<tr>
<td>Destination ID</td>
<td>The receiver’s peer ID</td>
<td>User</td>
</tr>
<tr>
<td>Hops</td>
<td>The times the message has been forwarded</td>
<td>Simulator</td>
</tr>
<tr>
<td>Type</td>
<td>The message’s type</td>
<td>User</td>
</tr>
<tr>
<td>Payload</td>
<td>The message’s data</td>
<td>User</td>
</tr>
</tbody>
</table>

Table: Message structure
Protocol Evaluation
Frequency Table Structure

Performance of Lookup algorithm

<table>
<thead>
<tr>
<th>Path Length</th>
<th>Frequency</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>27</td>
</tr>
<tr>
<td>3</td>
<td>61</td>
</tr>
<tr>
<td>4</td>
<td>1465</td>
</tr>
<tr>
<td>5</td>
<td>2912</td>
</tr>
<tr>
<td>6</td>
<td>3725</td>
</tr>
<tr>
<td>7</td>
<td>453</td>
</tr>
<tr>
<td>8</td>
<td>1307</td>
</tr>
<tr>
<td>9</td>
<td>47</td>
</tr>
<tr>
<td>10</td>
<td>2</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Aggregated Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>Minimum</td>
</tr>
<tr>
<td>Maximum</td>
</tr>
<tr>
<td>Average</td>
</tr>
<tr>
<td>Total</td>
</tr>
</tbody>
</table>
Running Simulation Scenarios

The simulator supports the following modes of operation:

- Standalone or Distributed execution
- Interactive or Non-interactive mode execution

Minimum requirements:

- Edit of a configuration file
- An operating LAN
- JRE 1.6 or higher
Running Simulation Scenarios
Configuration File

Three main categories of tunable parameters are provided:

- Application Nodes
  - AppNode < IPaddress, ID, Percentage >
- Distribution Functions
  - Uniform, PowerLaw, Beta, RandomSeed
- Execution Environment
  - ThreadPool, Files&Directories, DebugLevel
Running Simulation Scenarios

Simulation File

A simulation scenario consists of ...

- The protocol of the overlay
- The size of the overlay
- The key space of the overlay
- The number of the lookup and update operations
- The distributions according to the keys will be picked up for each operation
- The report file

It’s written in XML format and it is used instead of the GUI.
Running Simulation Scenarios
Graphical User Interface

Supports ...
- Protocol selection
- Overlay initialization
- Step-by-step execution
- Batch mode execution
- Generation of performance charts
- Real time statistics
- Real time message monitoring
Features and plans in progress include the...

- Enrichment of the protocol suite
- Design of an API for high level applications
- Implementation of various failure model scenarios
- Extension of the statical data gathering mechanism
- Development of a hot spot auto-discovery tool
Wiki, Issues, Source Code, Protocols, etc. @

http://code.google.com/p/d-p2p-sim
For the simulator please reference: