PeerSim: A Brief Introduction

Toward simulating gossip protocols
Why this lesson?
Why this lesson?
Why this lesson?

- **Objective**: learn the basic of PeerSim to develop your own (most likely the final term one) gossip protocol
  - First part: slides on PeerSim concepts
  - Second part: practical hands on session (warm up your laptops!)
Why this lesson?

**Objective:** learn the basic of PeerSim to develop your own (most likely the final term one) gossip protocol

- First part: slides on PeerSim concepts
- Second part: practical hands on session (warm up your laptops!)

Questions are welcome, please do them :)
Credits

• These slides take inspiration (and some images) from those written by
  • Stefano Ferretti, Moreno Marzolla, Francesco Gavazzo (UNIBO)
  • Alberto Montresor, Gianluca Ciccarelli (UNITN)
What is PeerSim

- PeerSim is a Java open source P2P simulator developed at the University of Bologna
- Designed to study the properties of P2P systems and to be friendly to configure and run
- Highly flexible (it is possible to tune it for your needs if required) and scalable (up to 1M peers)

http://peersim.sourceforge.net/
for jar, documentation and examples

PeerSim: A Peer-to-Peer Simulator
PeerSim components

(Most of this will be detailed later, don't despair)
Simulation Engine

**Cycle Driven (CD)**
- No explicit messages, no transport, synchronised
- Nice for epidemic protocols

**Event Driven (CD)**
- Explicit messages, realistic transport
- Nice for complex simulations

either case the execution is single threaded, which means **no concurrency**
Internals: Network

- The network is composed by nodes. It is the core data structure of the simulation.
- Practically, the class Network is a singleton that works as an enhanced container for a set of Node objects.
Internals: Node

- Interface `Node` defines one node in the network.
- A node is uniquely identified by an ID (an integer, think of it as the address of the node).
- A node itself does basically nothing (think of it as a computer connected to the Internet with no programs running). All the fancy stuff happens in the protocols.
- `GeneralNode` is the default implementation of `Node`, which defines an array of protocol.
  - 99.99999% of the times `GeneralNode` is ok in when simulating a gossip protocol.
Designing CD simulation

• A CD simulation is divided into cycles. In each cycle, a node is scheduled once for the execution.

• A minimal simulation is composed of 4 elements:
  • [protocol] A Java class that defines the behaviour of the nodes
  • [initialiser] A Java class that initialises the data of the protocol
  • [observer] One or more Java classes that log the behaviour of the system
  • [configuration] A text file that defines the simulation and the network
Peersim Protocol

• A protocol defines the behaviour (local computation and interaction) each node will follow during the simulation

• Each node executes its own protocol once per cycle sequentially

• Protocols can interact:
  • with different protocols on the same node by method calling
  • with the same protocol on another node (most common case) by message passing*
  • with other protocols on other nodes (rare) by message passing*
Message Passing in CD

- Message passing in CD simulations happens by direct method calling to the other’s node protocol.

- This means that:
  - (1) there is no delay in sending a message and that
  - (2) the effects of the message are immediately visible on the recipient.

- The effect of point (2), coupled with the fact that PeerSim execution is sequential, is that the order in which nodes are scheduled affects the outcome of the simulation.

- `Network.shuffle()` can mitigate this issue.
CDProtocol Interface

- `nextCycle(Node node, int protocolID)` is called once per cycle
  
  - `node`: the node executing the protocol
  
  - `protocolID`: the id of this protocol in the protocol array

- The `clone()` method must be carefully implemented, as it will be used to generate all the protocols of the kind. Specifically, PeerSim create a prototype protocol with the regular constructor, and then all other instance will be cloned from the prototype

CDProtocol extends Protocol
Linkable Interface

- Linkable is an interface that define the methods to organize the node view (think of it as that the node implementing Linkable is the node of a graph).
  - add/remove/get neighbours
  - get node’s degree
- The Linkable interface is useful to have all of above automatise by the simulator engine
The Control Interface

- The interface control is used to specify operations that necessitate of a **global** view on the Network
- The interface is very general, as the method `execute()` allows the definition of arbitrary operations over the simulation

In fact, three kinds of controls are normally used:

- **Initializers**: run at the beginning of the simulation to perform initialisation tasks (setting up the initial topology, populate initial node state)
- **Controllers**: run once per n cycles (usually n = 1) and perturb the simulation in some way (adding/removing nodes, shuffling networks)
- **Observers**: run once per n cycles (usually n = 1) and log the state of nodes and networks. Typically, observers gather data that represents the result of the simulation
The Configuration File

- The configuration file is a plain ASCII text file
  - define what components to use (protocols, controls, etc..)
  - how the components interact each other
- Based on Java properties file
  - a list of <key, value> properties that define the simulation
  - java peersim.Simulator <configuration_file>
Configuration File
The Average Example

```
######## VARIABLES ########
DEGREE 2
SIZE 10000

######## SIMULATION PARAMETER ####
random.seed 1234567890
simulation.cycles 5000
control.shuffle Shuffle
network.size SIZE

######## PROTOCOL ####
protocol.average peersim.Average

######## INITIALIZER ####
# Creates the topology
init.connector WireKOOut
init.connector.protocol protocol average
init.connector.k DEGREE

# Initializes the values
init.initializer AverageInitializer
init.initializer.protocol average

######## CONTROLS ####
# Prints statistics of the network
control.observer peersim.AverageObserver
control.observer.protocol average
```
Configuration File

The Average Example

```
######## VARIABLES ######
DEGREE 2
SIZE 10000


######## SIMULATION PARAMETER ######
random.seed 1234567890
simulation.cycles 5000
control.shuffle Shuffle
network.size SIZE


## PROTOCOL
protocol.average peersim.Average

## INITIALIZER
# Creates the topology
init.connector WireKOut
init.connector.protocol protocol average
init.connector.k DEGREE

# Initializes the values
init.initializer AverageInitializer
init.initializer.protocol protocol average

## CONTROLS
# Prints statistics of the network
control.observer peersim.AverageObserver
control.observer.protocol protocol average
```
Configuration File
The Average Example

```plaintext
#----- VARIABLES ----#
DEGREE 2
SIZE 10000

#----- SIMULATION PARAMETER ----#
random.seed 1234567890
simulation.cycles 5000
control.shuffle Shuffle
network.size SIZE

#----- PROTOCOL ----#
protocol.average peersim.Average

#----- INITIALIZER ----#
# Creates the topology
init.connector WireKOut
init.connector.protocol average
init.connector.k DEGREE

# Initializes the values
init.initializer AverageInitializer
init.initializer.protocol average

#----- CONTROLS ----#
# Prints statistics of the network
control.observer peersim.AverageObserver
control.observer.protocol average
```

macro definition

general setup
Configuration File
The Average Example

```
#### VARIABLES ####
DEGREE 2
SIZE 10000

#### SIMULATION PARAMETER ####
random.seed 1234567890
simulation.cycles 5000
control.shuffle Shuffle
network.size SIZE

#### PROTOCOL ####
protocol.average peersim.Average

#### INITIALIZER ####
# Creates the topology
init.connector WireKOut
init.connector.protocol average
init.connector.k k DEGREE

# Initializes the values
init.initializer AverageInitializer
init.initializer.protocol average

#### CONTROLS ####
# Prints statistics of the network
control.observer peersim.AverageObserver
control.observer.protocol average
```
Configuration File

The Average Example

```plaintext
#### VARIABLES ####
DEGREE 2
SIZE 10000

#### SIMULATION PARAMETER ####
random.seed 1234567890
simulation.cycles 5000
control.shuffle Shuffle
network.size SIZE

#### PROTOCOL ####
protocol.average peersim.Average

#### INITIALIZER ####
# Creates the topology
init.connector WireKOut
init.connector.protocol average
init.connector.k DEGREE

# Initializes the values
init.initializer AverageInitializer
init.initializer.protocol average

#### CONTROLS ####
# Prints statistics of the network
control.observer peersim.AverageObserver
control.observer.protocol average
```

macro definition
general setup
protocol definition
topology creation (*WireKOut* takes a *Linkable* object and adds DEGREE random connections)
Configuration File

The Average Example

```
### VARIABLES ###
DEGREE 2
SIZE 10000
### SIMULATION PARAMETER ###
random.seed 1234567890
simulation.cycles 5000
control.shuffle Shuffle
network.size SIZE
### PROTOCOL
protocol.average peersim.Average
### INITIALIZER
# Creates the topology
init.connector WireKOut
init.connector.protocol average
init.connector.k DEGREE
# Initializes the values
init.initializer AverageInitializer
init.initializer.protocol average
### CONTROLS
# Prints statistics of the network
control.observer peersim.AverageObserver
control.observer.protocol average
```

- **macro definition**
- **general setup**
- **protocol definition**
- **topology creation** *(WireKOut takes a Linkable object and adds DEGREE random connections)*
- **data initialisation**
Configuration File
The Average Example

```plaintext
####### VARIABLES ######
DEGREE 2
SIZE 10000

####### SIMULATION PARAMETER ######
random.seed 1234567890
simulation.cycles 5000
control.shuffle Shuffle
network.size SIZE

## PROTOCOL
protocol.average peersim.Average

## INITIALIZER
# Creates the topology
init.connector WireKOut
init.connector.protocol average
init.connector.k DEGREE

# Initializes the values
init.initializer AverageInitializer
init.initializer.protocol average

## CONTROLS
# Prints statistics of the network
control.observer peersim.AverageObserver
control.observer.protocol average
```

- macro definition
- general setup
- protocol definition
- topology creation (WireKOut takes a Linkable object and adds DEGREE random connections)
- data initialisation
- control declaration