Peersim P2P Simulator

Stefano Arteconi

Dipartimento di Scienze
dell'Informazione
Università di Bologna

Peersim Introduction

- Peersim is an open source, Java based, P2P simulation framework aimed to develop and test any kind of P2P algorithm in a dynamic environment.
- High scalability (up to 1 million nodes)
- Extremely flexible
  - Highly configurable
  - Architecture based on pluggable components

Peersim Introduction (cont'd)

- Peersim documentation and support
  - JavaDoc and Tutorials available online
  - Web based mailing list available on sourceforge
- Peersim community
  - All the algorithms developed in the Bison project have been developed and tested on Peersim
  - Even though Peersim is not advertised its popularity is growing outside BISON project as well

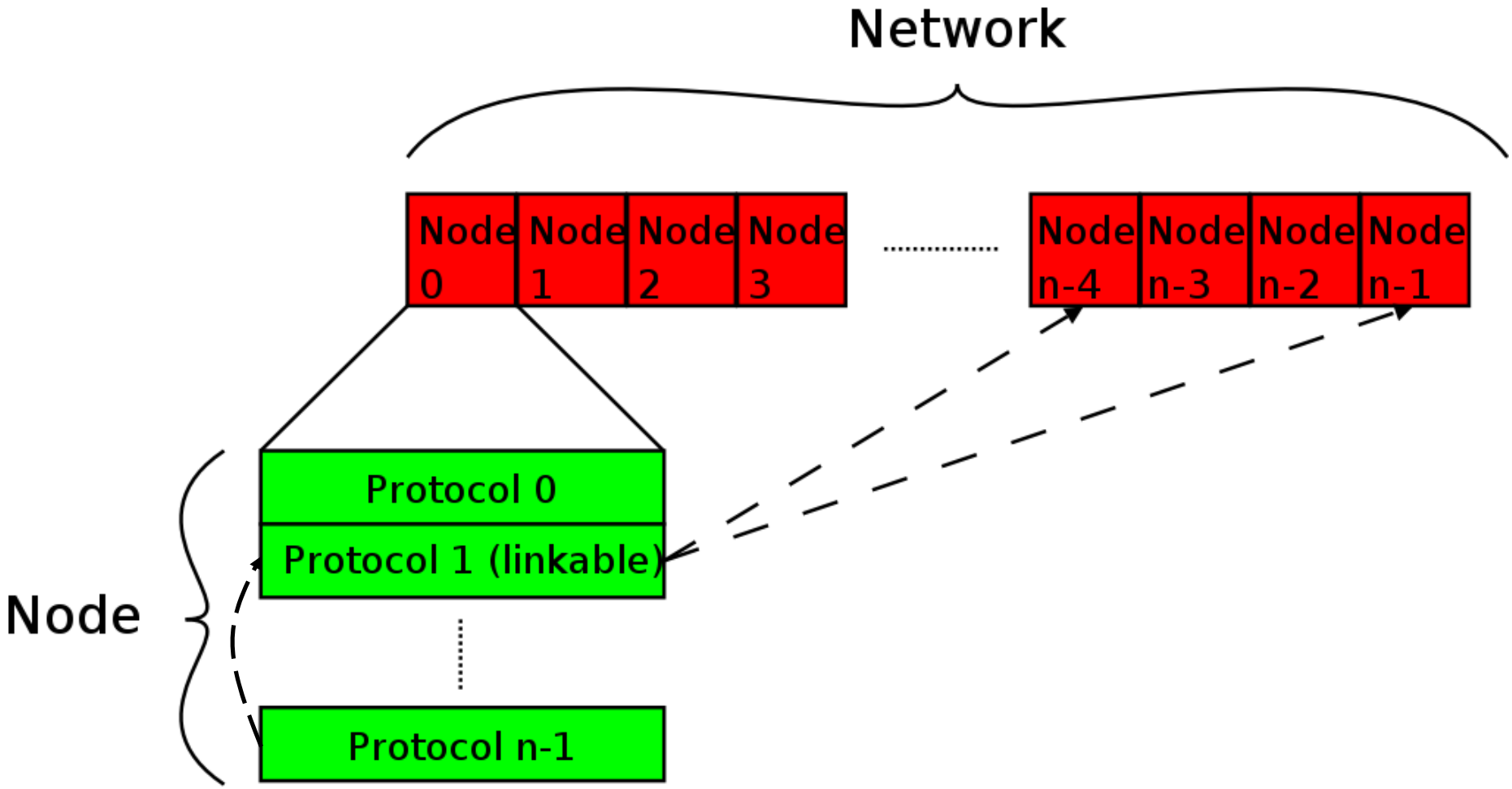
Peersim Simulation Engine

- Single-threaded simulator
- Two different simulation models
  - Cycle-driven: sequential simulation, in each cycle each protocol's actions are executed by every node (we will focus on this)
  - Event-driven: a set of events (messages) are scheduled at different simulation time and the node protocols are run accordingly to messages' delivery times.

Network Representation in Peersim

- *Network*: global array containing all the network's nodes
- *Node*: basic network element, each node's state and actions are described through a stack of protocols
- *Linkable*: interface used to access and manage nodes' view
- *CDProtocol*: interface used to describe nodes' actions at each cycle
  - A generic protocol can both perform specific action (*CDProtocol*) and mange local view (*Linkable*)
- *Control*: some kind of global control on the network needed to perform operation such as initialization and performances analysis

Network Representation in Peersim (cont'd)



Peersim Scheduler

Initializers are executed
while(simTime<cycles){
  for each *Node* in *Network*{
    CDProtocol actions are executed
  }
  Control are executed
}

Main Peersim Interfaces: Node

- *Node*: used to define nodes state and characteristics such as ID and failure state (DEAD, DOWN, OK)
- Default implementation provided by class *GeneralNode*
- Possible extensions: type, coordinates, etc.

Main Peersim Interfaces: Linkable

- *Linkable*: interface used to access and manage node's view, hence network topology with actions like
  - Adding neighbors
  - Getting neighbors
  - Checking for neighbors
  - Evaluating node's degree (number of neighbors)

Main Peersim Interfaces: Protocol

- *CDProtocol*: Interface used to define cycle-driven protocols, that is the actions performed by each node at each simulation cycle
  - Each node can run more than one protocol
  - Protocols are ordered as a stack and executed sequentially by each node

Main Peersim Interfaces: Control

- *Control*: Interface used to define operations that require global network knowledge and management, such as:
  - Initialization of protocols at the beginning of the simulation
    - Initial topology
    - Nodes state
  - Network Dynamism
    - Adding nodes
    - Removing nodes
    - Resetting nodes
  - Statistical analysis
    - Aggregated values from all the nodes

Configuration File

- Once all the components have been implemented the whole simulation has to be set up
  - Declare what components to use
  - Define the way they should interact
- In Peersim simulations are defined through a plain text configuration file
- Configuration file is divided in 3 main parts
  - General setup
  - Protocol definition
  - Control definition

Configuration File: Example

```
01 random.seed 1234567890
02 simulation.cycles 30
03 control.shf Shuffle
04 network.size 50000
05
06 protocol.lnk example.newscast.SimpleNewscast
07 protocol.lnk.cache 20
08
09 protocol.avg example.aggregation.AverageFunction
10 protocol.avg.linkable lnk
11
12 order.protocol lnk avg
13
14 init.rnd WireKOut
15 init.rnd.protocol lnk
16 init.rnd.k 20
17
18 init.pk example.aggregation.PeakDistributionInitializer
19 init.pk.protocol avg
20 init.pk.value 10000
21
22 control.dnet DynamicNetwork
23 control.dnet.add -500
24 control.dnet.minsize 4000
25
26 control.ao example.aggregation.AverageObserver
27 control.ao.protocol avg
28
29 order.control ao dnet
```

General settings

Protocols settings

Controls settings

Configuration File: General Settings

```
01 random.seed 1234567890
02 simulation.cycles 30
03 control.shf Shuffle
04 network.size 50000
```

- Line 1 defines random seed to be used, if not defined a random one is chosen
- Line 2 defines simulation length (simulation will stop after 30 cycles)
- Line 3 declare that *Network* array should be shuffled at the beginning of each cycle (to increase randomness)
- Line 4 defines network's initial size as 50000

Configuration File: Linking Protocols Together

```
06 protocol.lnk example.newscast.SimpleNewscast
07 protocol.lnk.cache 20
08
09 protocol.avg example.aggregation.AverageFunction
10 protocol.avg.linkable lnk
11
12 order.protocol lnk avg
```

- Line 6 defines *lnk* as newscast protocol with a maximum view size of 20 neighbors (line 7)
- Line 9 defines *avg* as average aggregation which uses network topology obtained by newscast protocol (line 10)
- Line 12 defines protocols execution order

Configuration File: Initialization

```
14 init.rnd WireKOut
15 init.rnd.protocol lnk
16 init.rnd.k 20
17
18 init.pk example.aggregation.PeakDistributionInitializer
19 init.pk.protocol avg
20 init.pk.value 10000
```

- Line 14 defines initialization for newscast protocol (*lnk* in line 15) as a random network with maximum degree 20 (line 16)
- Line 17 defines initialization for aggregation protocol (*avg* line 19) as a peak distribution with value 10000 (line 20)

Configuration File: Dynamicity

```
22 control.dnet DynamicNetwork
23 control.dnet.add -500
24 control.dnet.minsize 4000
```

- Line 22 defines network dynamicity *dnet* through DynamicNetwork class, 500 nodes are removed (line 23) until the network reaches a minimum size of 4000 nodes (line 24)

Configuration File: Logging Data

```
26 control.ao example.aggregation.AverageObserver
27 control.ao.protocol avg
28
29 order.control ao dnet
```

- Line 26 defines *ao* as an observer logging data as defined in AverageObserver class to be applied to *avg* protocol (line 27)
- Line 29 defines controls execution order

Peersim Summary

- P2P network simulator
- Extremely flexible
  - Any kind of P2P network and protocol can be defined by main interfaces implementation
  - Configuration file provides a way to easily plug together different components
- High abstraction level
  - Single threaded sequential simulation engine
  - No message abstraction provided

Links

- Peersim homepage: http://peersim.sourceforge.net
- Peersim project page: http://sourceforge.net/projects/peersim