PlanetSim Release 3.0 Candidate in depth

Jordi Pujol Ahulló

Universitat Rovira i Virgili

jordi.pujol@urv.net

Departament d'Enginyeria [DEIM]

Informàtica i Matemàtiques

Universitat Rovira i Virgili

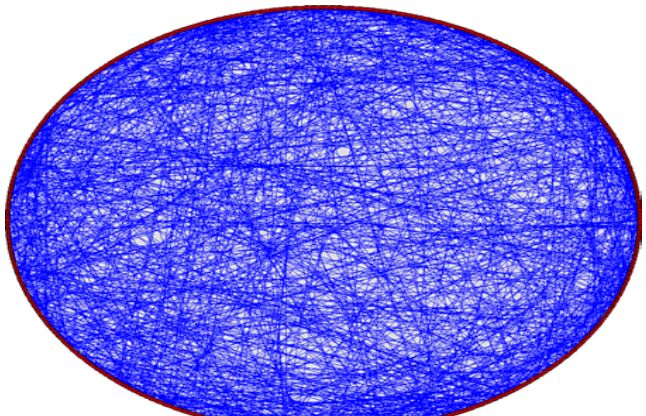
Copyright

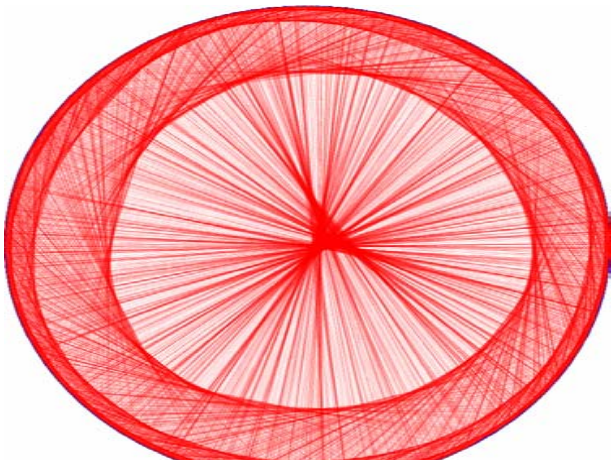
Index

- Introduction: Use cases
- Architecture: Simulator, Node, RouteMessage structure, Factories, RouteMessagePool
- Configuration: tuning
- Life cycle: Simulation, Applications, Application level messages

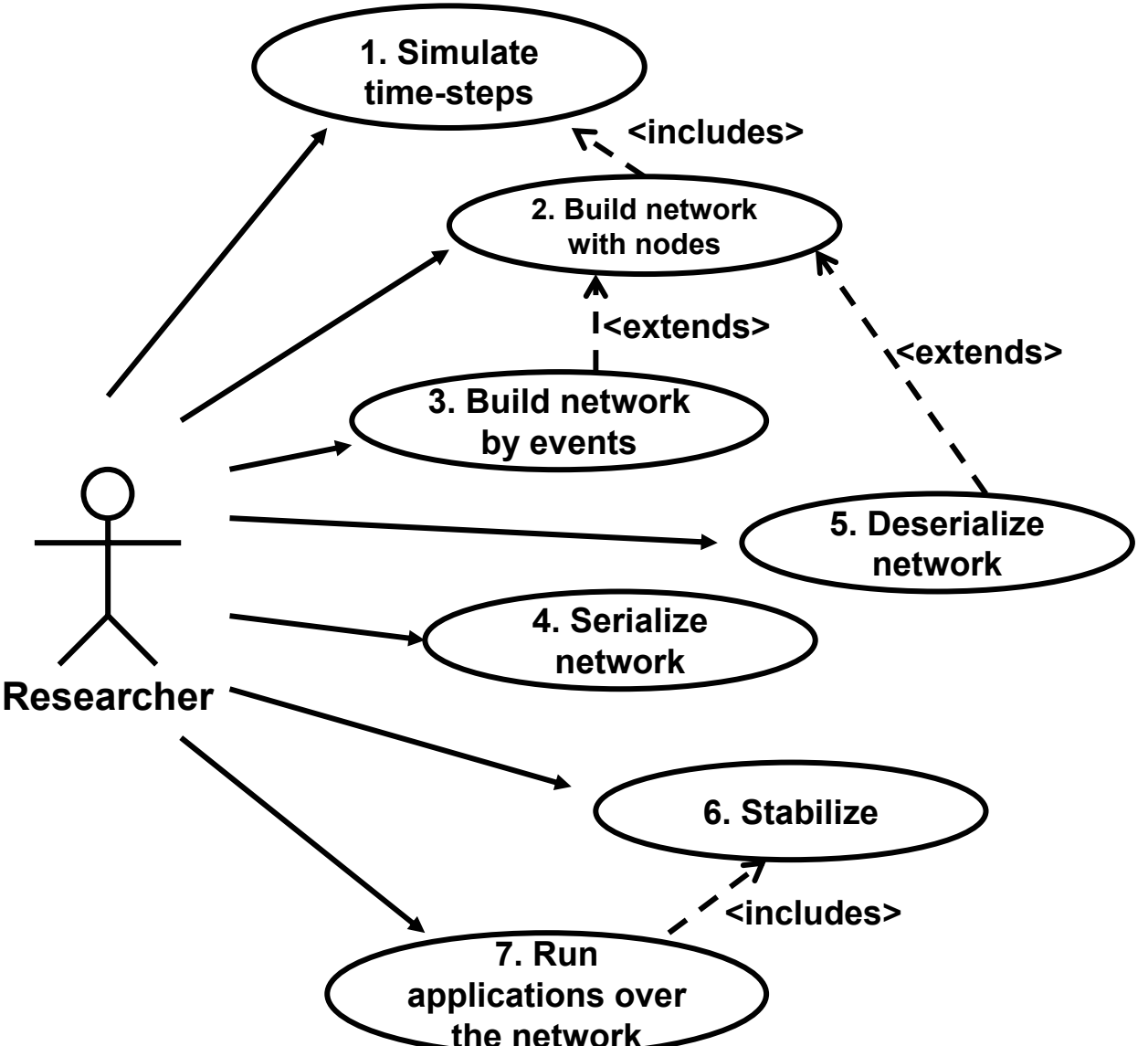
Introduction

- Open code project
- Framework for overlay network simulation
- Implemented in Java
- Layered architecture (decouple p2p protocols from applications)
- Advanced visualization output (Pajek, GML)





Use Cases: Researcher



Use cases: Researcher (II)

1. Simulate time-steps:

   Run specified number of time-steps within current network. The simulator processes all nodes this number of time-steps and delivers generated messages at each time-step to the target nodes.

2. Build network with nodes:

   Build network with the specified number of nodes. Can be built a zero-sized network. A cofigured number of time-steps is run between each node join.

Use cases: Researcher (III)

3. Build network by events
    1. A zero-sized network has to be built.
    2. All events in specified file will be loaded.
    3. Only node events: { JOIN | LEAVE | FAIL }
4. Serialize network

    Save on disk the current network. Waste time only once: reuse simulated networks different times.
5. Deserialize network

    Load from disk a serialized network. It will be presented as before being serialized. Continue your research at this point.

Use cases: Researcher (IV)

6. Stabilize

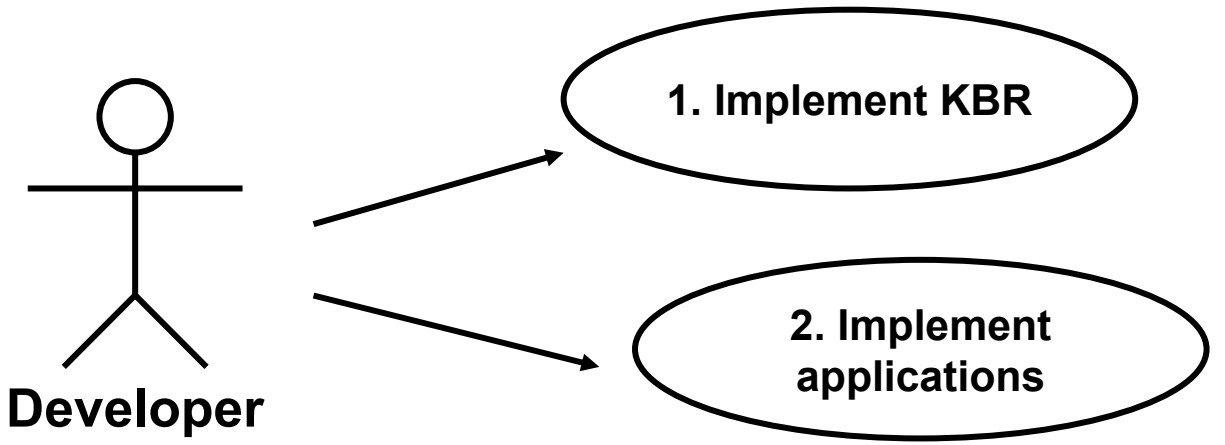
   Post-conditions:

   - Overlay network stabilized
   - Only overlay maintenance messages
   - No application level messages remain to deliver

7. Run application over the network

   Add/Register the applications to the nodes, focus of your research, and evaluate its operation.

Use cases: Developer



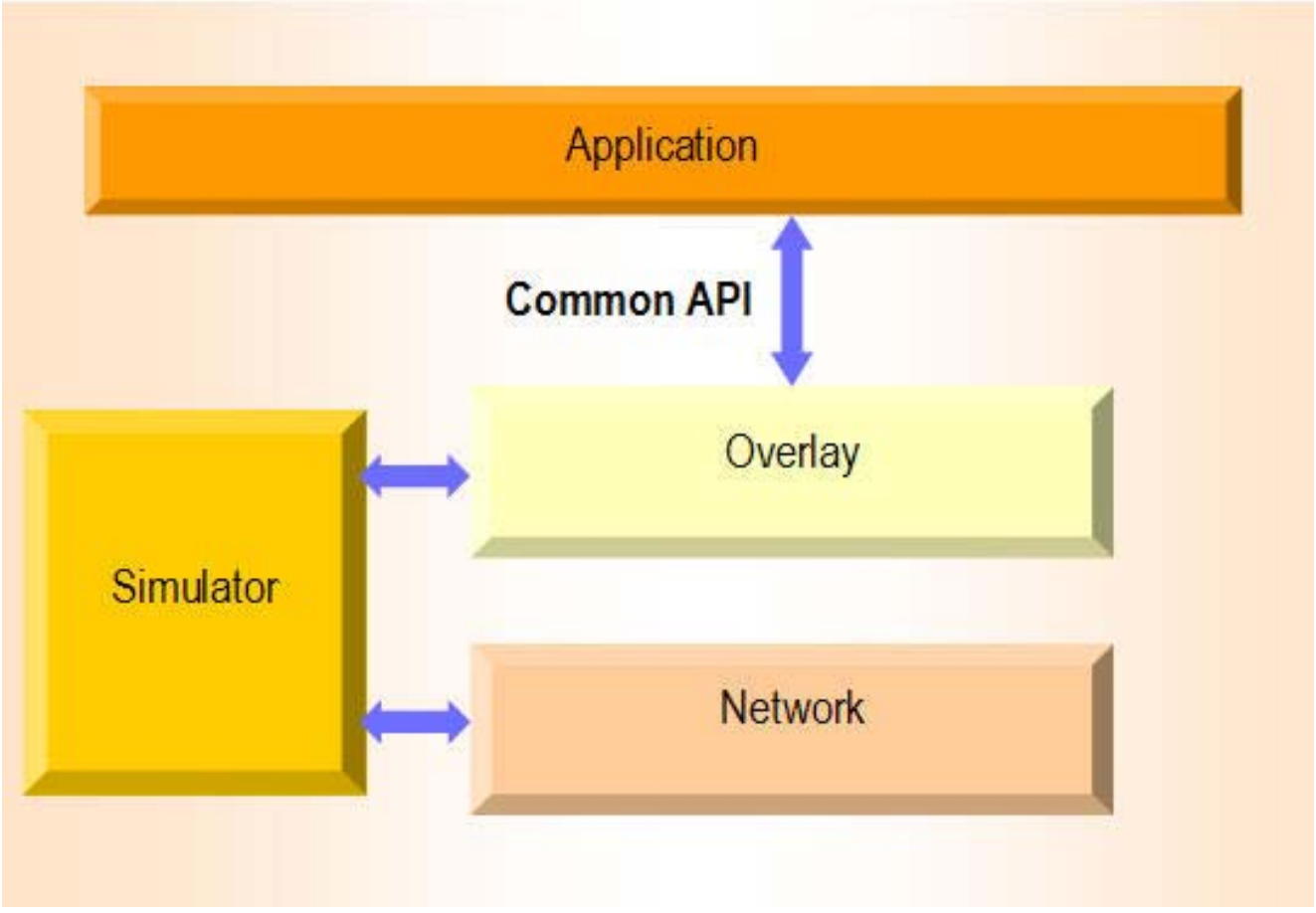
Use cases: Developer (II)

1. Implement KBR
   - Implement new KBR overlay networks, like Pastry or Chord.
   - Mainly, it requires the related node implementation, following the new overlay protocol.

2. Implement applications

   Develop new compatible applications for these KBR overlay networks, not focused on any specific KBR.

PlanetSim Architecture

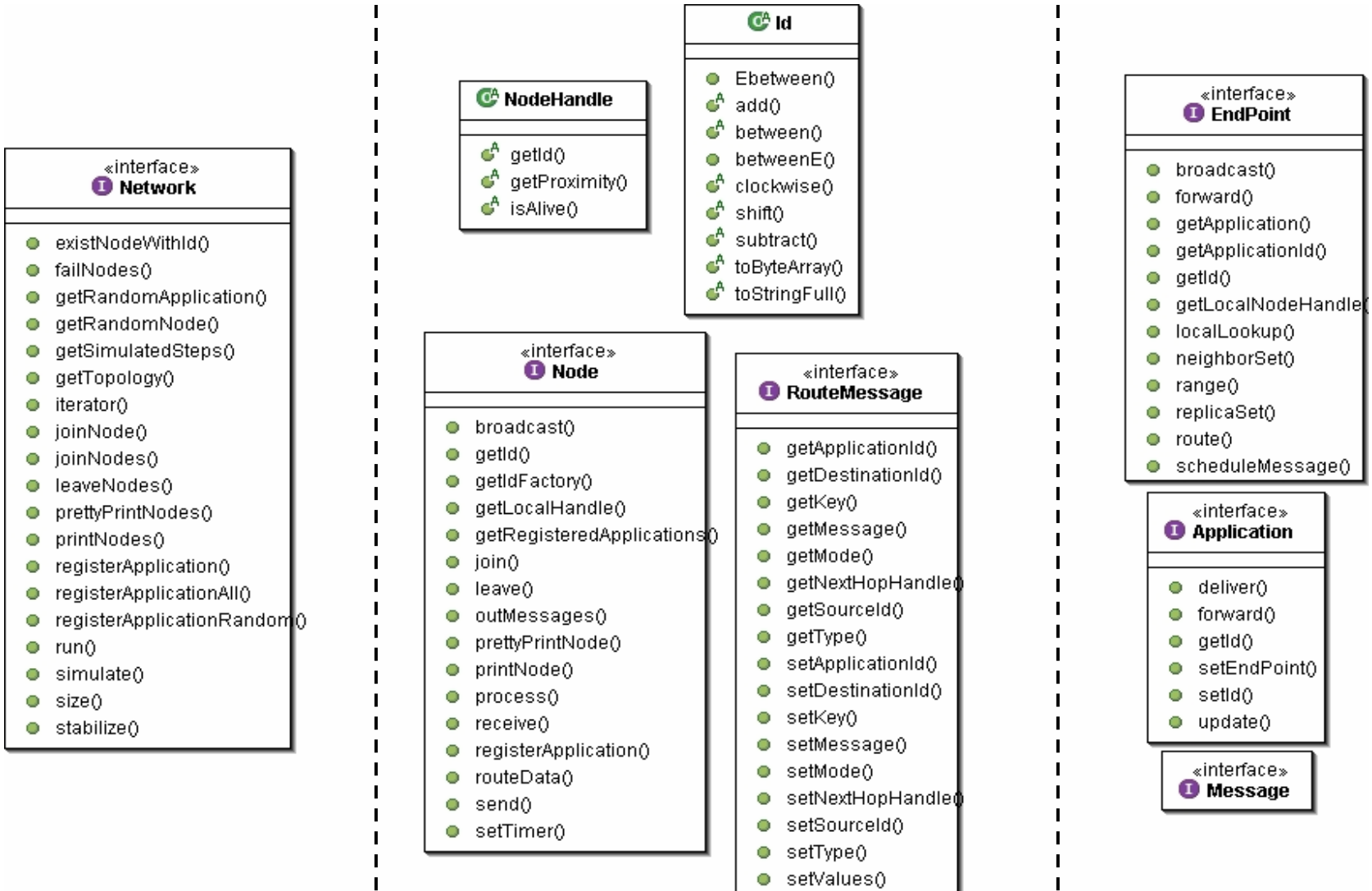


PlanetSim Architecture cont'd

Network layer

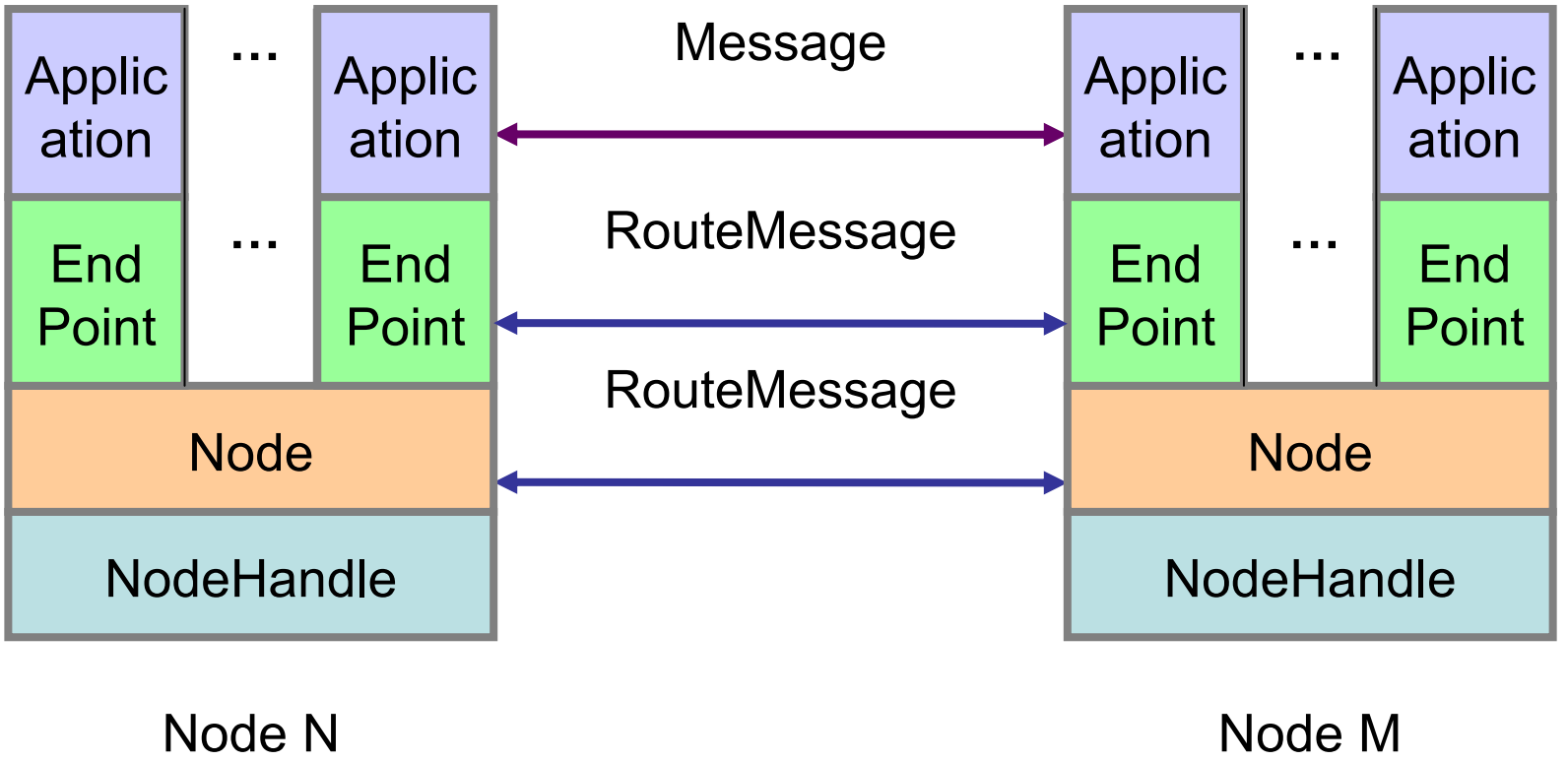
Node layer

Application layer



Node Architecture

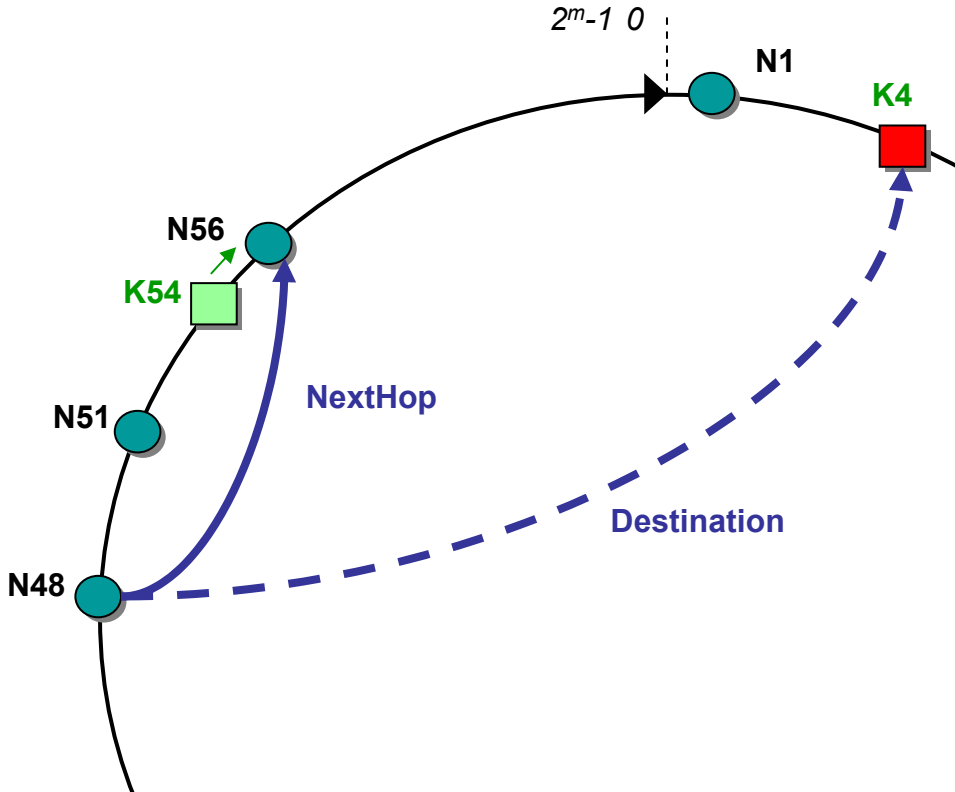
- Layered definition



RouteMessage Structure

Source	Destination	NextHop	Key	AppId	Message

- **Extensible**
- **Invariants:**
  - Source: 48
  - Destination: 4
  - Key: 8123
  - AppId: “”
  - Message: “Put sth”
- **Variants:**
  - NextHop: 56



Factories

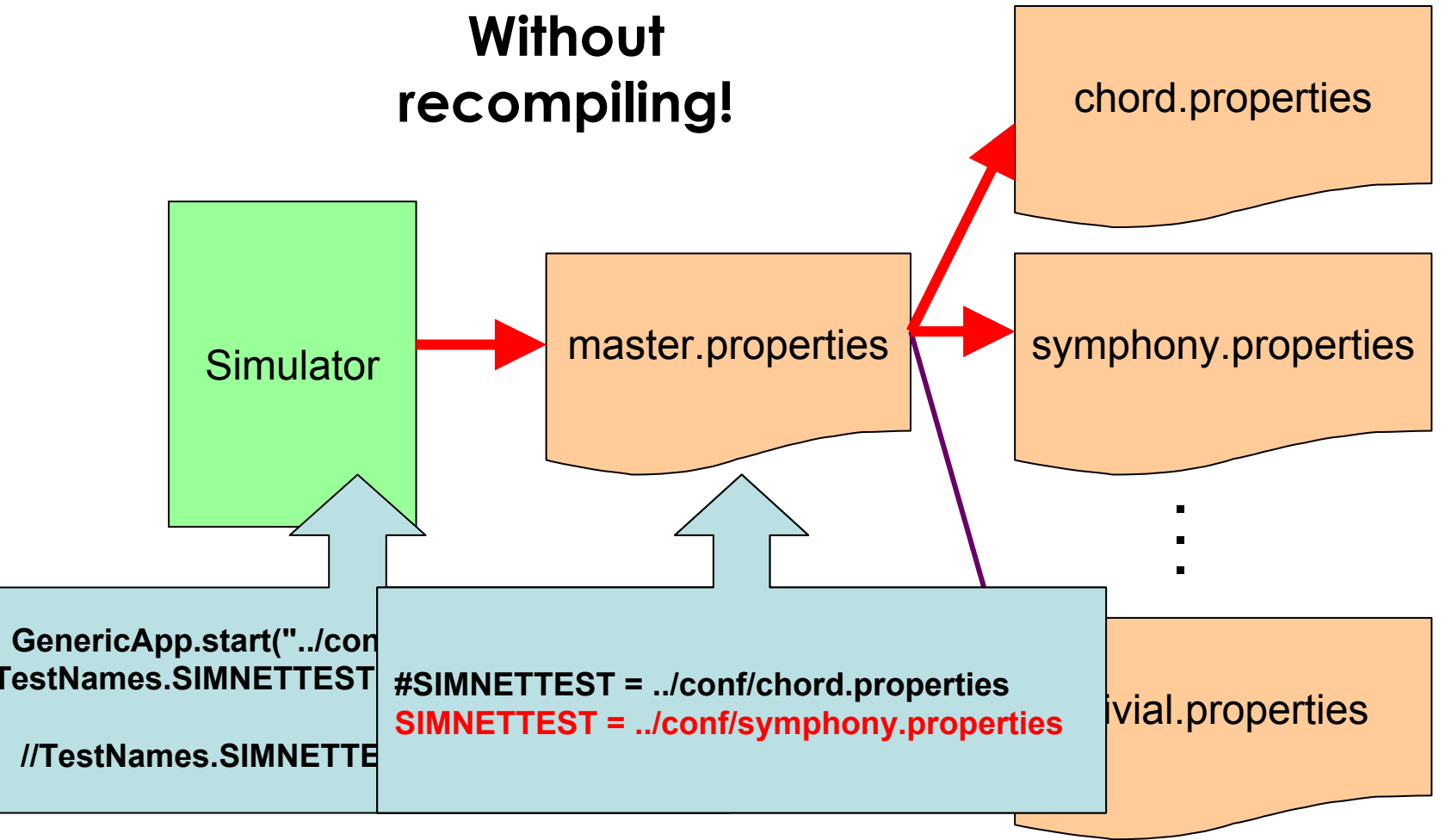
- Interfaces that follow the Factory Method design pattern
  - Goal: Build elemental instances, without mean real implementations
  - {Application | EndPoint | Node | NodeHandle | Id}Factory
- GenericFactory:
  - A “super-factory”, that uses specified factories implementations within simulation

RouteMessagePool

- Time and resources eficiency is required
- Java Garbage Collector is our “enemy”. No “new XYZ(…);” in your code!!
- A lot of RouteMessages are used within a simulation
- RouteMessages have a very short life time
- A Pool of RouteMessages is the most suitable design:
  getRouteMessage( ) & freeRouteMessage( )
- **Note:** The RouteMessage’s life cycle is handled by the programmer

Configuration: Tuning

- **Example:** Running a test called "**SIMNETTEST**"



Configuration: Tuning cont'd

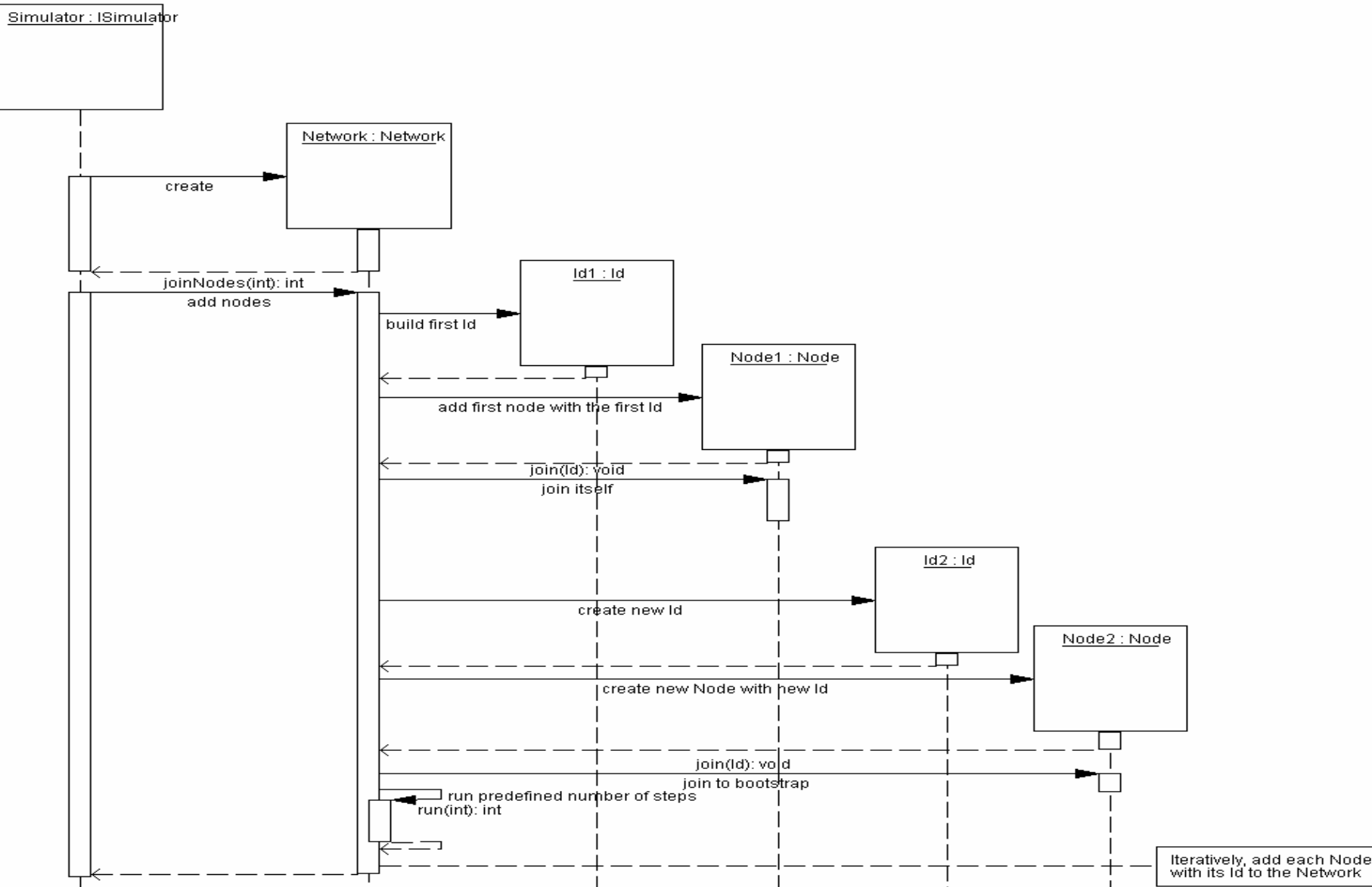
- Steps for adding a new test (recommended):
  - Select a non repeated name for the test and put it in planet.test.TestNames
  - Put this name into the conf/master.properties and associate the final(s) configuration files

    FACTORY_TESTAPPFACTORY = ../conf/chord.properties

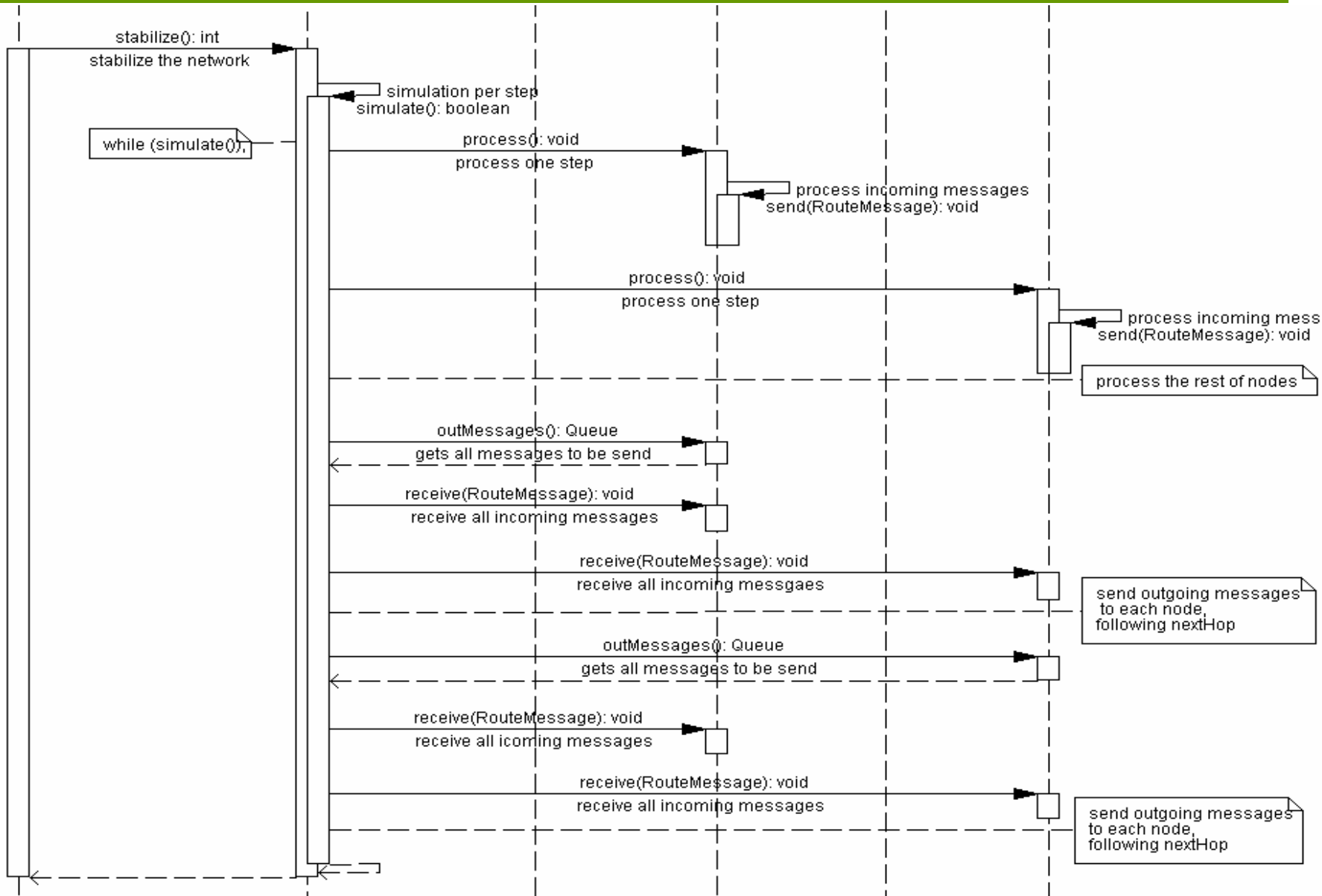
    #FACTORY_TESTAPPFACTORY = ../conf/symphony.properties

    #FACTORY_TESTAPPFACTORY = ../conf/trivial.properties
  - Create your own configuration files (if necessary), based on {chord|symphony}.properties
  - Build the test itself under planet.test hierarchy

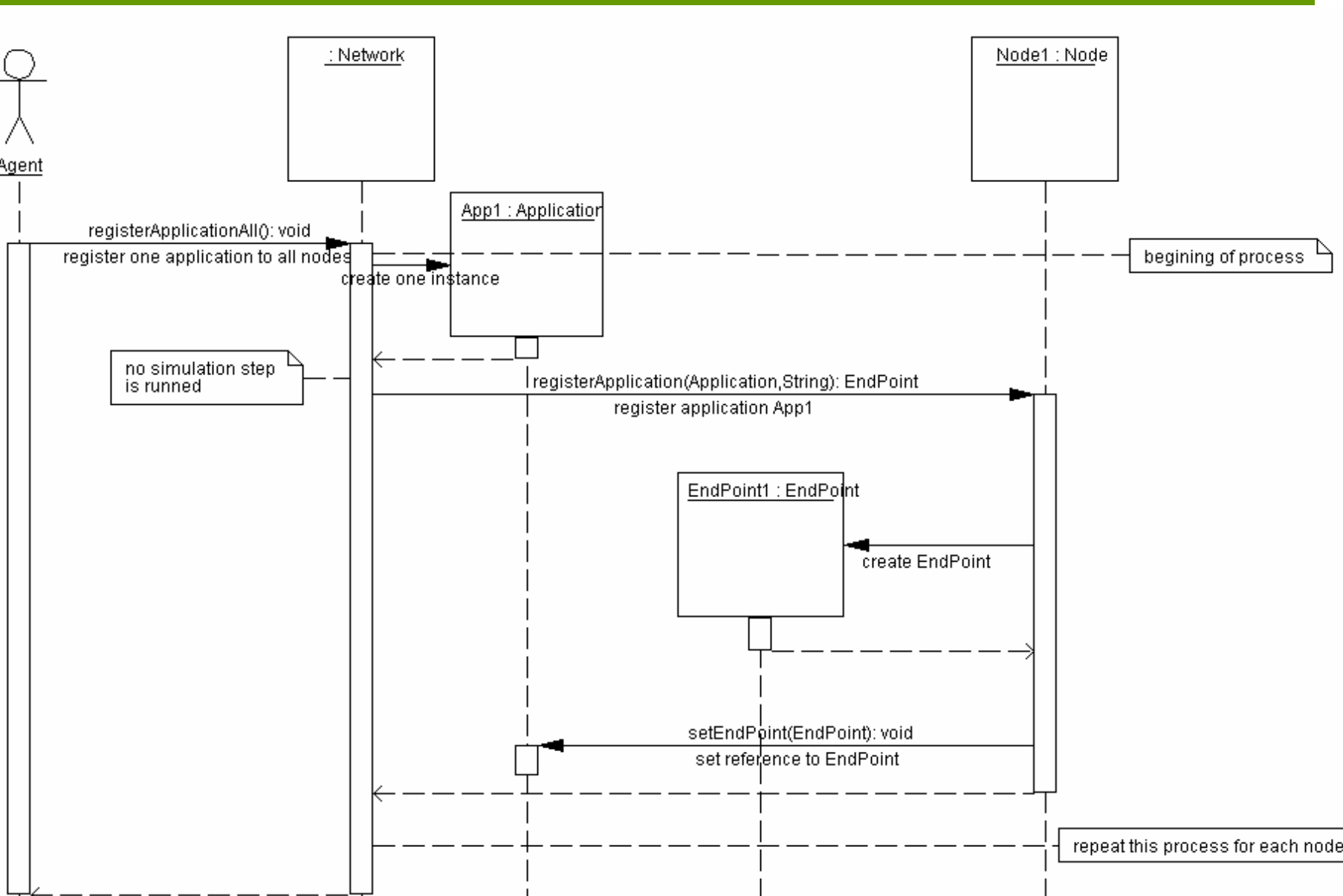
Simulation's life cycle



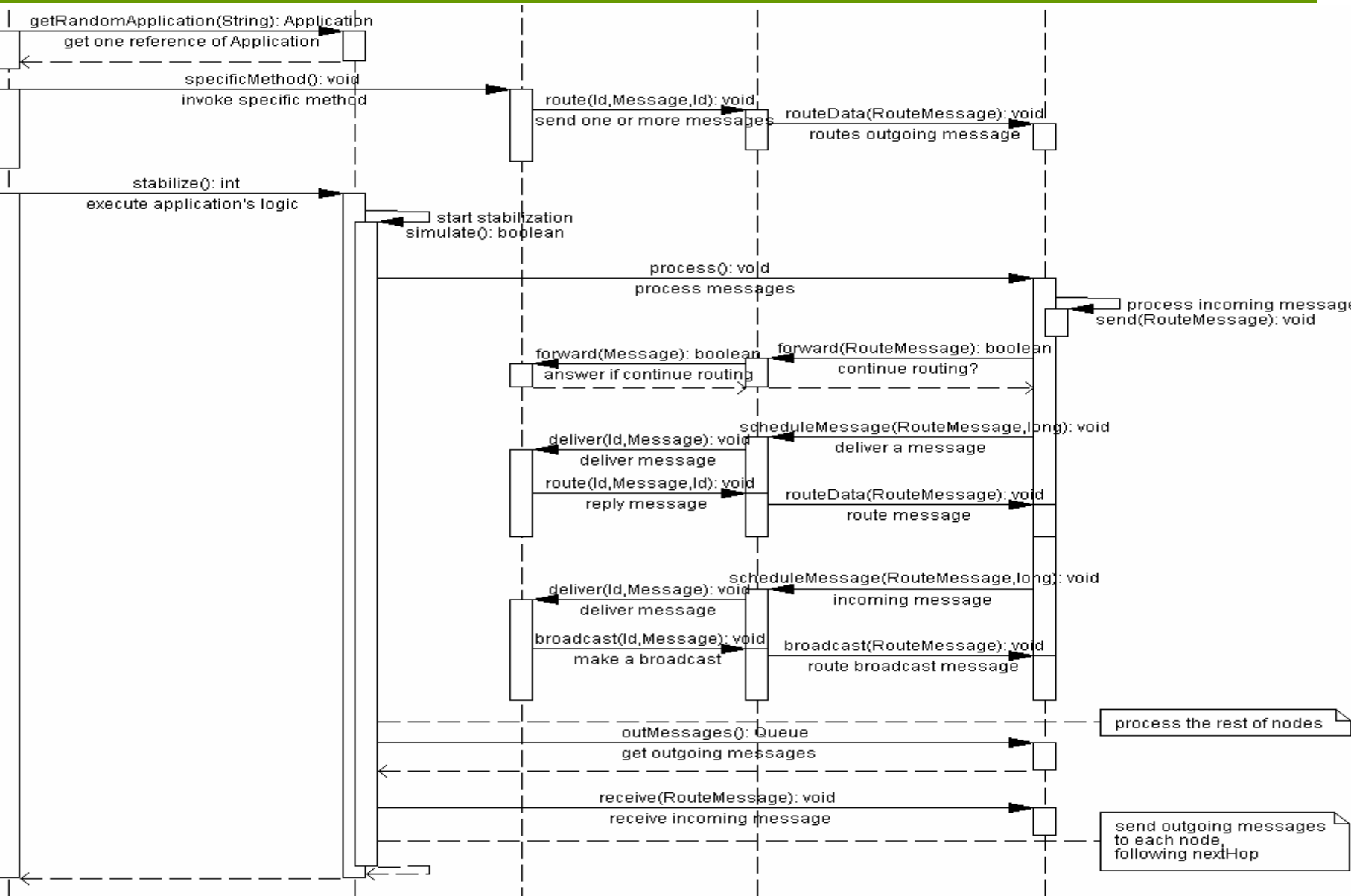
Simulation's life cycle cont'd



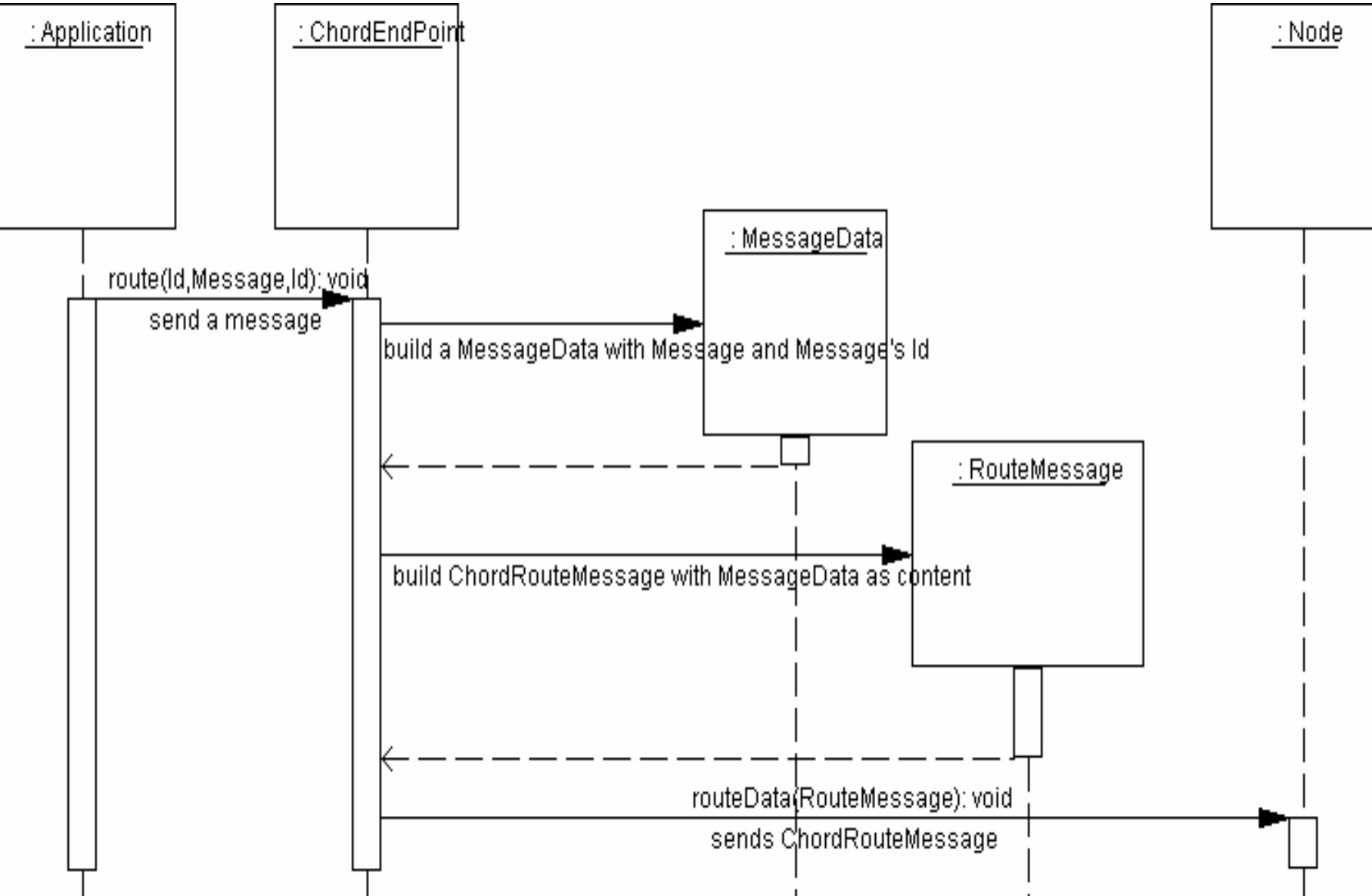
Applications management



Applications management cont'd



Application level messages life cycle



Application level messages life cycle