LUNES: Simulation of P2P Networks

Gabriele D’Angelo

<g.dangelo@unibo.it>

http://www.cs.unibo.it/gdangelo/

Systems simulation, 2013-2014

Department of Computer Science and Engineering

University of Bologna

Outline

- LUNES: Large Unstructured NEtwork Simulator
- Parallel And Distributed Simulation (PADS)
- Adaptive PADS
- ARTÌS/GAIA and LUNES
- Data Dissemination in P2P Networks
- Gossip Protocols
- Simulation-based Performance Evaluation

What's the problem? **Scalability** issues

- **Traditional simulation tools** are **unable to cope** with very large, dynamic, complex and detailed models
- Many systems (i.e. P2P networks) are often made of a **very large number** of nodes
- Such nodes can be **heterogeneous** (*with different characteristics*) and very **dynamic** (*in and out of the network*)
- The network topology can be **complex** (*random*, *scale-free*, *small-world*)
- The performance evaluation (of such systems) often requires **fine-grained** and **detailed** models of communication protocols

LUNES: Large **U**nstructured **NE**twork **S**imulator

- Overall design of **LUNES**:
  - network topology creation
  - protocol simulation
  - trace analysis
- Different tools for different tasks: all phases are quite complex

LUNES: Large Unstructured NEtwork Simulator

- Overall design of **LUNES**:
  - network topology creation
  - protocol simulation
  - trace analysis
- Different tools for different tasks: all phases are quite complex

The **initial network topology** can be generated using the more appropriate tool (e.g. igraph, custom generators ecc.) and it is exported to the "protocol simulation" module using the graphviz dot language

LUNES: Large Unstructured NEtwork Simulator

- Overall design of **LUNES**:
  - network topology creation
  - protocol simulation
  - trace analysis
- Different tools for different tasks: all phases are quite complex

This **does not** means that the network topology is **static**!

The protocol simulation can easily modify the topology at runtime

The **initial network topology** can be generated using the more appropriate tool (e.g. igraph, custom generators ecc.) and it is exported to the “protocol simulation” module using the graphviz dot language

LUNES: Large Unstructured NEtwork Simulator

- Overall design of **LUNES**:
  - network topology creation
  - protocol simulation
  - trace analysis
- Different tools for different tasks: all phases are quite complex

The core of the simulator: it implements the **specific P2P protocols** and manage the network topology.

It uses the services provided by the **simulation middleware**

LUNES: Large Unstructured NEtwork Simulator

- Overall design of **LUNES**:
  - network topology creation
  - protocol simulation
  - trace analysis
- Different tools for different tasks: all phases are quite complex

Fine-grained and detailed protocol generate very **verbose trace files**. For statistical correctness many runs have to be completed.

The output generated by medium complexity models is in the order of **gigabytes** (per run)

Parallel And Distributed Simulation (PADS)

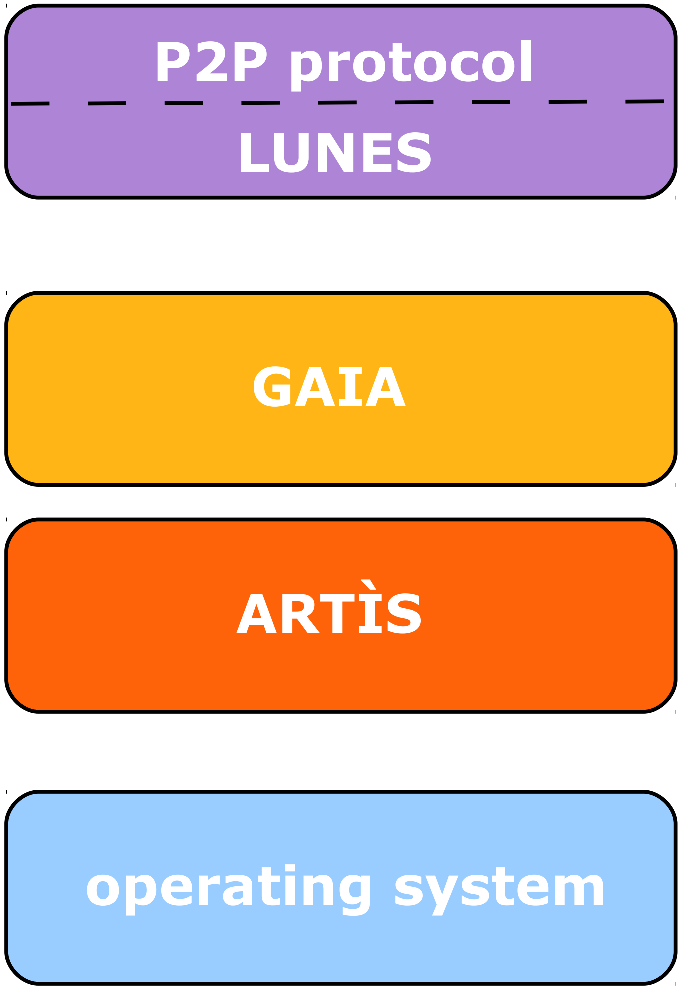
- **Why is so hard using PADS for P2P systems?**
- Because such models are **communication bounded** (much more than computation)
- … and in PADS the **communication is very costly**!
- **Execution time** saved by parallel computation is often lost in communications (e.g. synchronization, state updates)
- Such applications are **not embarrassingly parallel**
- In many models, **increasing the number of nodes** has a **linear cost** in terms of **computation** and a **super linear increase** of **communication**

Adaptive PADS

- A "suitable" allocation of **Simulated Model Entities** (**SME**s) can greatly **reduce the communication cost**
- This is the **PADS partitioning problem**: with dynamic and heterogeneous systems the static solution does not work!
- **Adaptive partitioning**: based on the simulation execution
- The idea is to **observe the communication pattern** of each **SME** and to **cluster adaptively the highly interacting SMEs in the same LP** (that is on the same CPU)
- **This can reduce the costly inter-LP communication**
- Some subtle details are missing from this high level description (e.g. **migration** of SMEs, **load balancing** and **synchronization**)

ARTÌS/GAIA and LUNES

- **ARTÌS**: **simulation middleware**, provides the **basic functionalities** (synchronization, communication, coordination etc.)
- **GAIA**: implementation of **adaptive PADS**. Insulates the middleware from the model. Provides a **Multi Agent System** (MAS) abstraction
- **LUNES**: **model skeleton** with the basic functionalities of P2P systems



For details and **software download**: http://pads.cs.unibo.it

Data dissemination in **P2P** networks

- The peers are organized in some form of **overlay network** (many **different topologies** can be used)
- The **data dissemination** is obtained by passing messages through the overlay
- **Gossip protocols** are very **simple** and **well suited** for P2P systems
- If **all nodes** in a P2P network have to be reached by **every generated message**, then traditional gossip protocols are **quite inefficient**

Gossip protocol: **probabilistic broadcast**

- If the message is locally generated then it is broadcasted to all neighbors, otherwise it is decided at random if it will be broadcasted or ignored

PARAMETERS:

- p_b = probability to broadcast a message

ADDITIONAL MECHANISMS:

- time to live (**ttl**) in each message
- local **cache** in each node

ALGORITHM

```
function INITIALIZATION()
pb ← PROBABILITY_BROADCAST()


function GOSSIP(msg)
if  (RANDOM() < pb or
    FIRST_TRANSMISSION())
    then
        for all nj in Πj do
                SEND(msg, nj)
        end for
end if
```

Gossip protocol: **fixed probability**

- For each received message, the node randomly selects those edges through which the message must be propagated

PARAMETERS:

- **v** = threshold value

ADDITIONAL MECHANISMS:

- time to live (**ttl**) in each message
- local **cache** in each node

ALGORITHM

```
function INITIALIZATION()
v ← CHOOSE_PROBABILITY()

function GOSSIP(msg)
for all n_j in Π_j do
    if RANDOM() < v then
        SEND(msg, n_j)
    end if
end for
```

Adaptive gossip

- Is it possible to build "**smarter**" gossip protocols?
- **Assumption**: events are generated at a rate that can be approximated using some probability distribution

  For example, state updates in online games

- Periodically each node **checks the reception rate** of events from all other nodes in the network
- If this rate is **lower than a threshold** value, then it can send one or more **stimuli** to **neighbor nodes**

Adaptive gossip: **implementation** and **variants**

- Many different implementations and variants are possible:
  - **stimuli associated to receivers (alg. #1)**

    upon reception of a stimulus from a neighbor, a peer increases its dissemination probability towards that node
  - **stimuli associated to generators (alg. #2)**

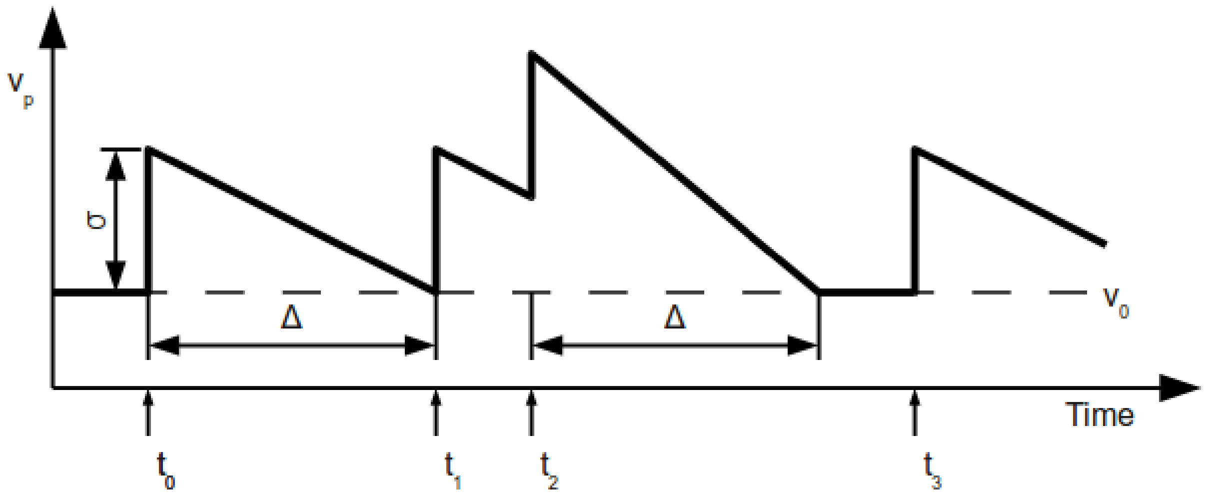
    the peer increases the dissemination probability of all messages from a given sender towards all its neighbors
  - **stimuli associated to generators and receivers (alg. #3)**

    in this case, the dissemination probability of all messages from a given sender and for a given neighbor is increased

    ***note**: this variant is much more specific than the previous ones*

Adaptive gossip: **stimulus management**

- In practice it is a "fixed probability" scheme in which the **probability to disseminate a message** to a given neighbor is **modified by the received stimuli**



- A given node receives a new stimulus (of magnitude **σ**) at times $\mathbf{t_0}$, $\mathbf{t_1}$, $\mathbf{t_2}$ and $\mathbf{t_3}$. At time $\mathbf{t_2}$, the stimulus adds **σ** to the current value of $\mathbf{v_p}$ (*current dissemination probability*)
- $\mathbf{v_p}$ *decays linearly* to $\mathbf{v_0}$ (*baseline dissemination probability*) after time **Δ** from the last received stimulus

Performance evaluation: **simulation**-based

- The following performance evaluation is based on **simulation**
- **L**arge **U**nstructured **NE**twork **S**imulator (**LUNES**)

Parameter	Value
number of **nodes**	*100*
number of **edges** per node	*2*
number of **graphs** per evaluation	*100*
construction method	*Erdos-Renyi generator*
cache size (local to each node)	256 *slots*
message Time To Live (**ttl**)	*8*
simulated time (gaming time)	5000 *time-steps* *(after building)*

Performance evaluation: **metrics**

- **Coverage**
  - percentage of nodes that have received **all the messages** that have been produced during the whole simulation

    "are the game events received by all gamers?"

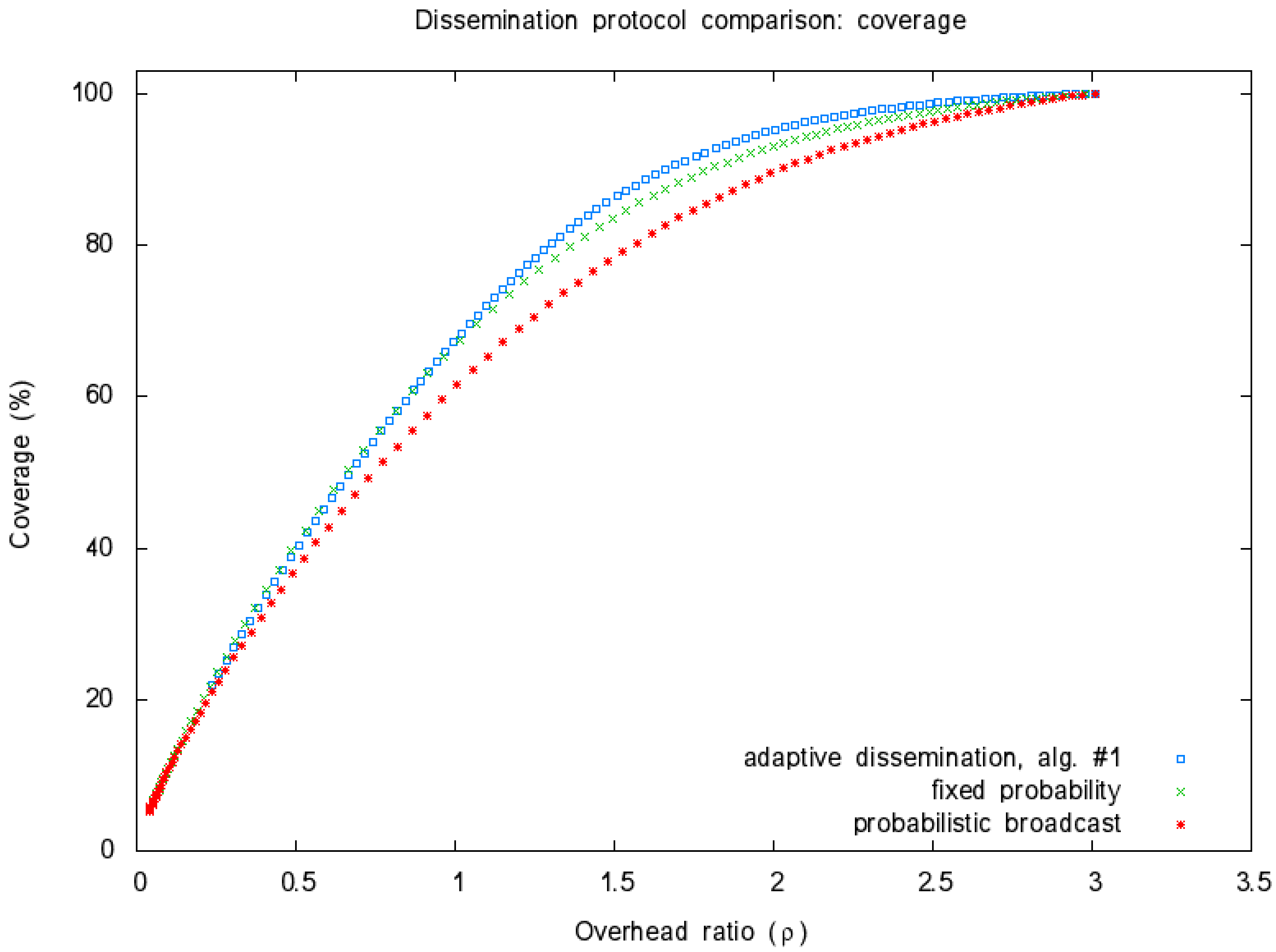
- **Delay**
  - average number of **hops** that are necessary to receive a message after its creation

    "is the dissemination of new events timely?"

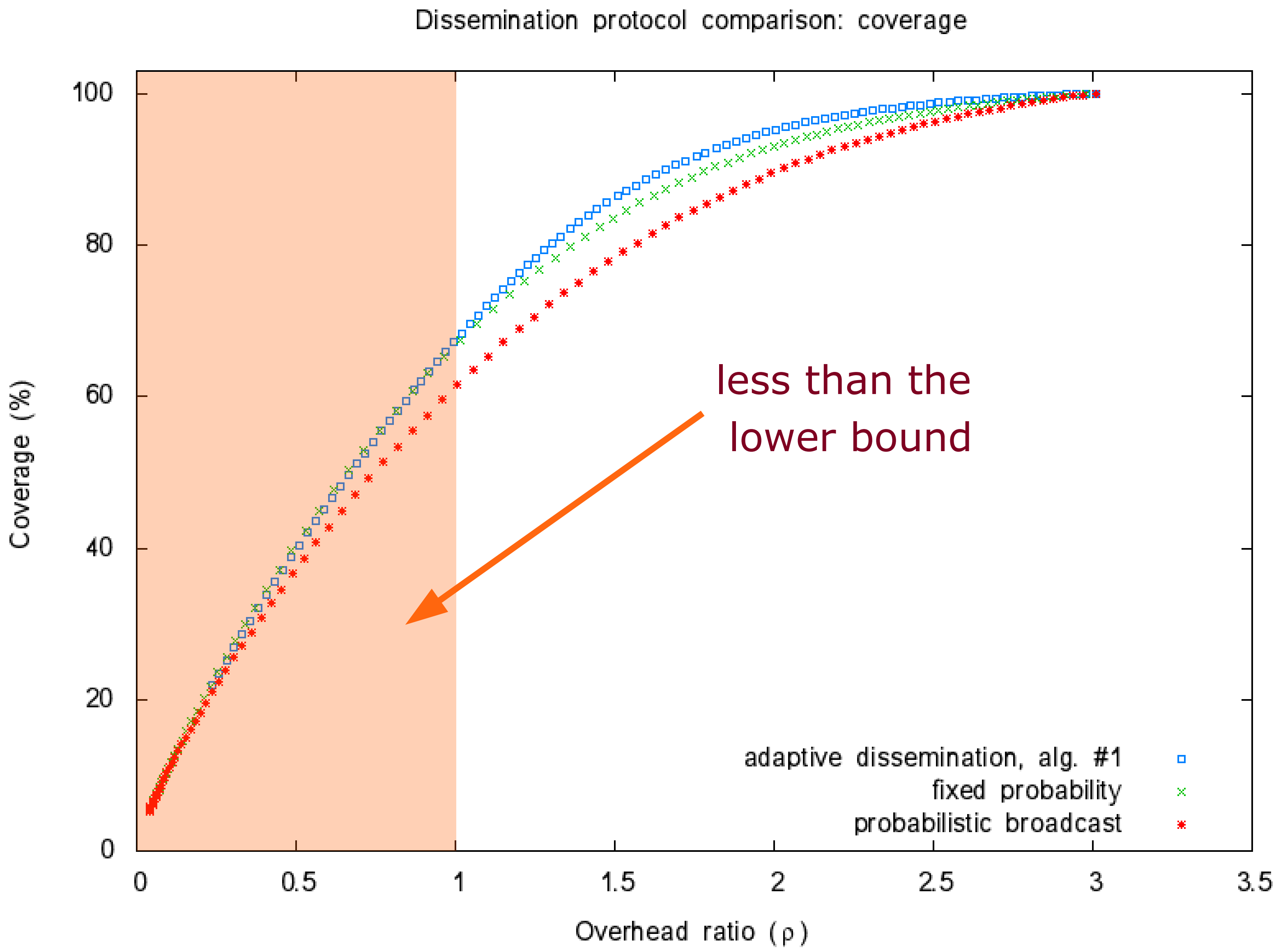
Performance evaluation: **cost metrics**

- Defining an appropriate **cost metric** is necessary to compare all the dissemination protocols in the same conditions
- **Overhead ratio** $\rho = \frac{\text{delivered messages}}{\text{lower bound}}$
  - ***delivered messages*** *=* ***total number*** *of messages delivered in a simulation run by a specific dissemination protocol*
  - ***lower bound*** *=* ***minimum number*** *of messages that have to be sent by a dissemination protocol that never sends duplicates but obtains full coverage*
- In the following we will compare all the dissemination protocols in terms of **coverage** and **delay** for **many different overhead ratios**

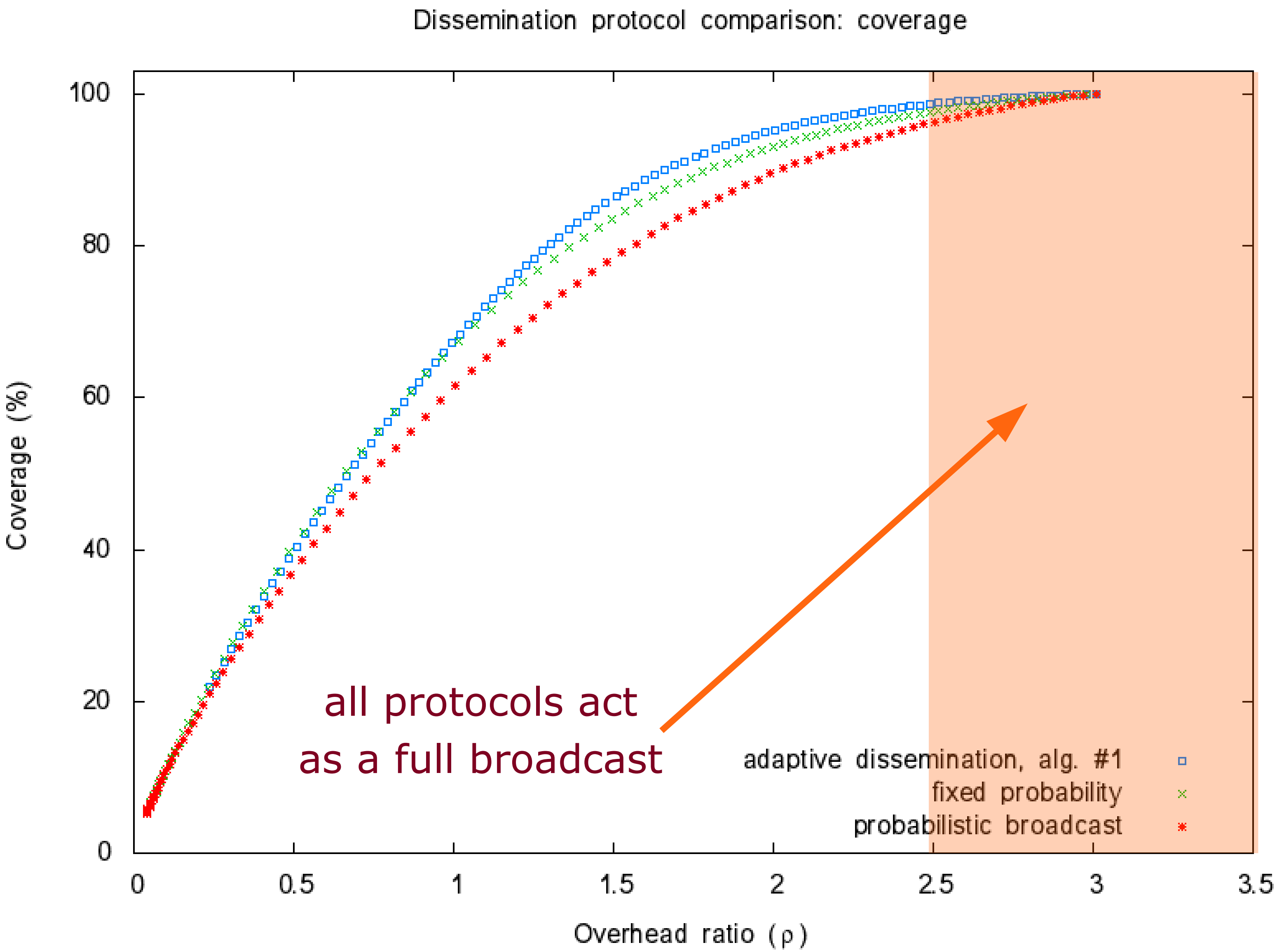
Evaluation: **coverage** rate (%)



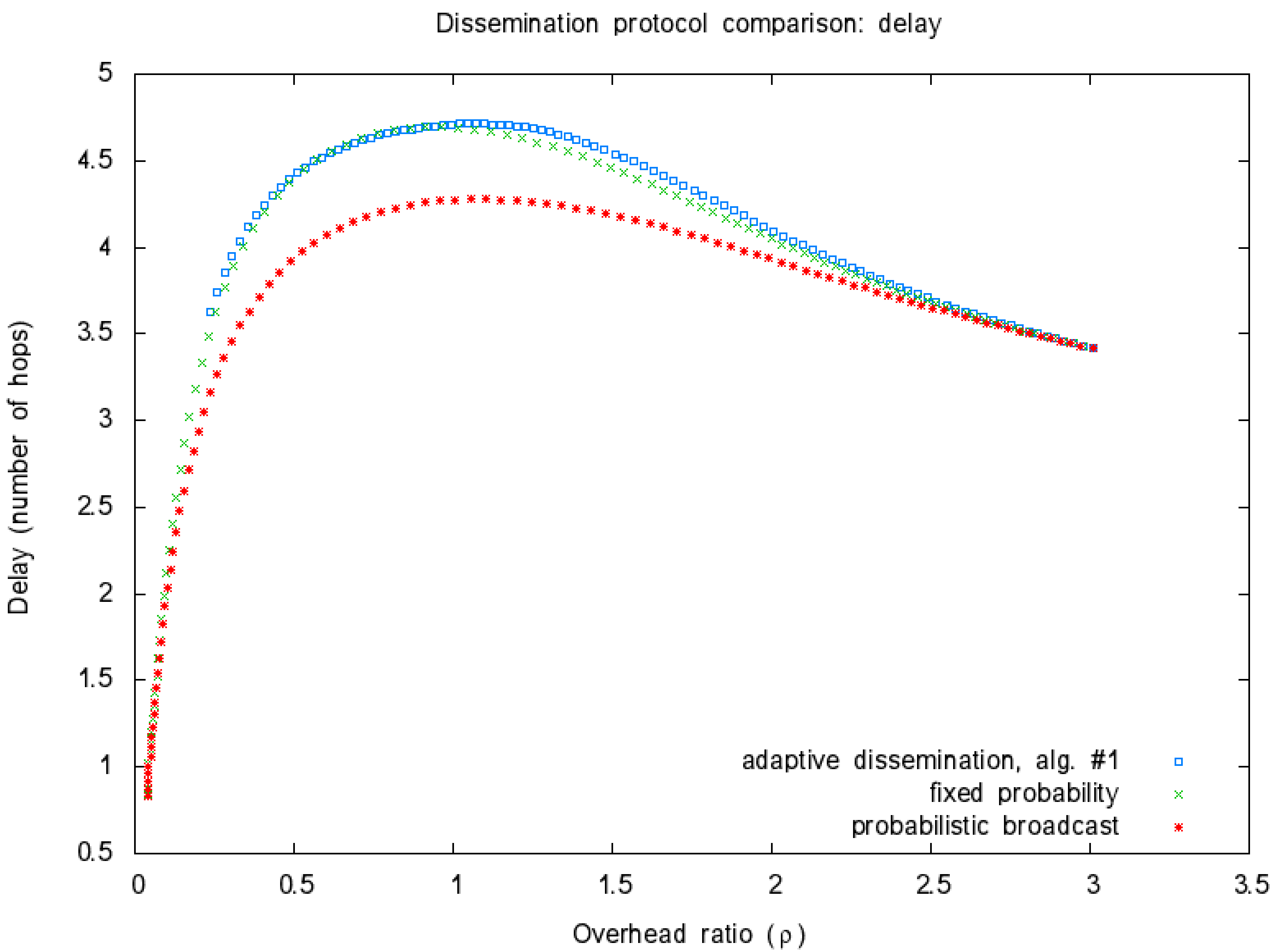
Evaluation: **coverage** rate (%)



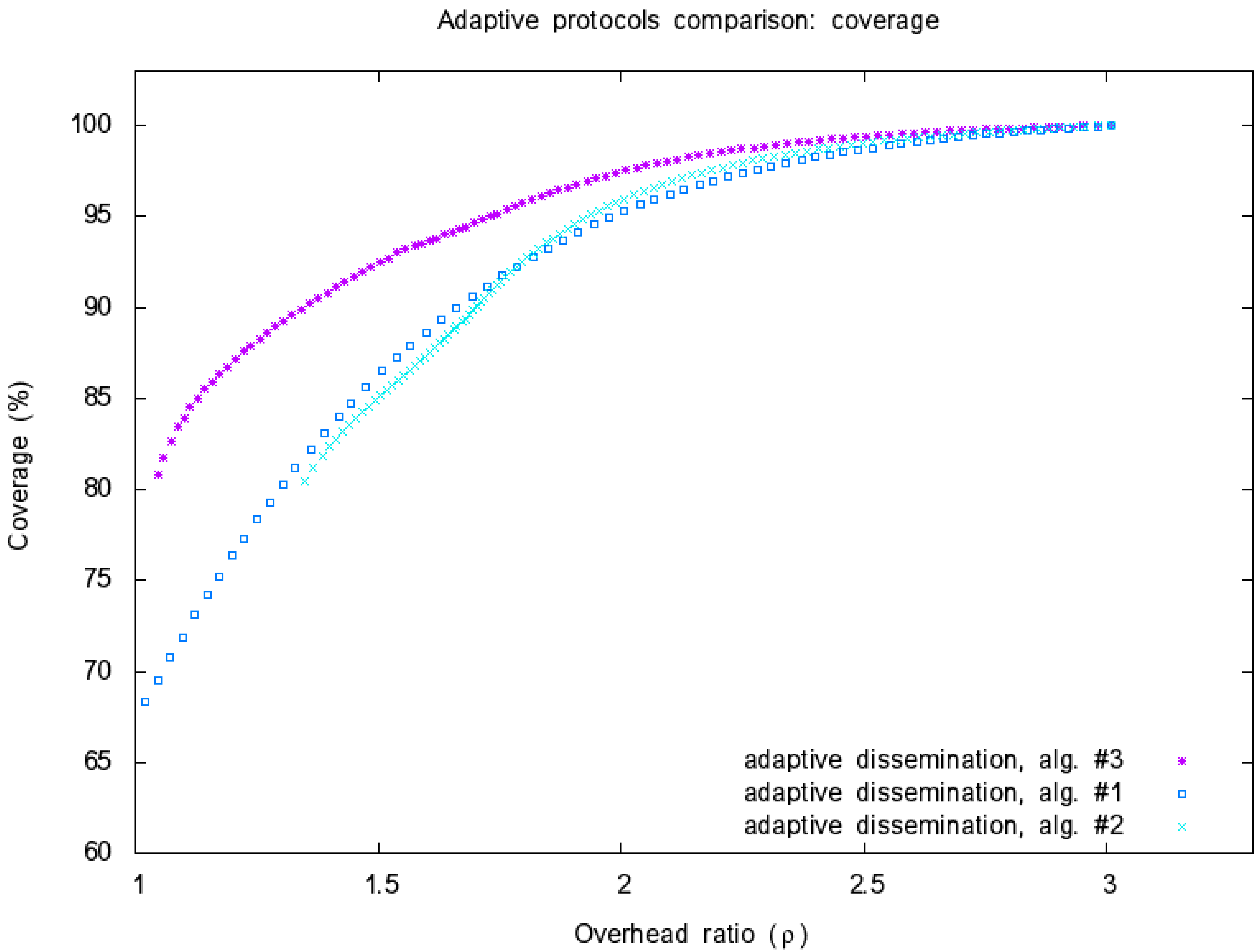
Evaluation: **coverage** rate (%)



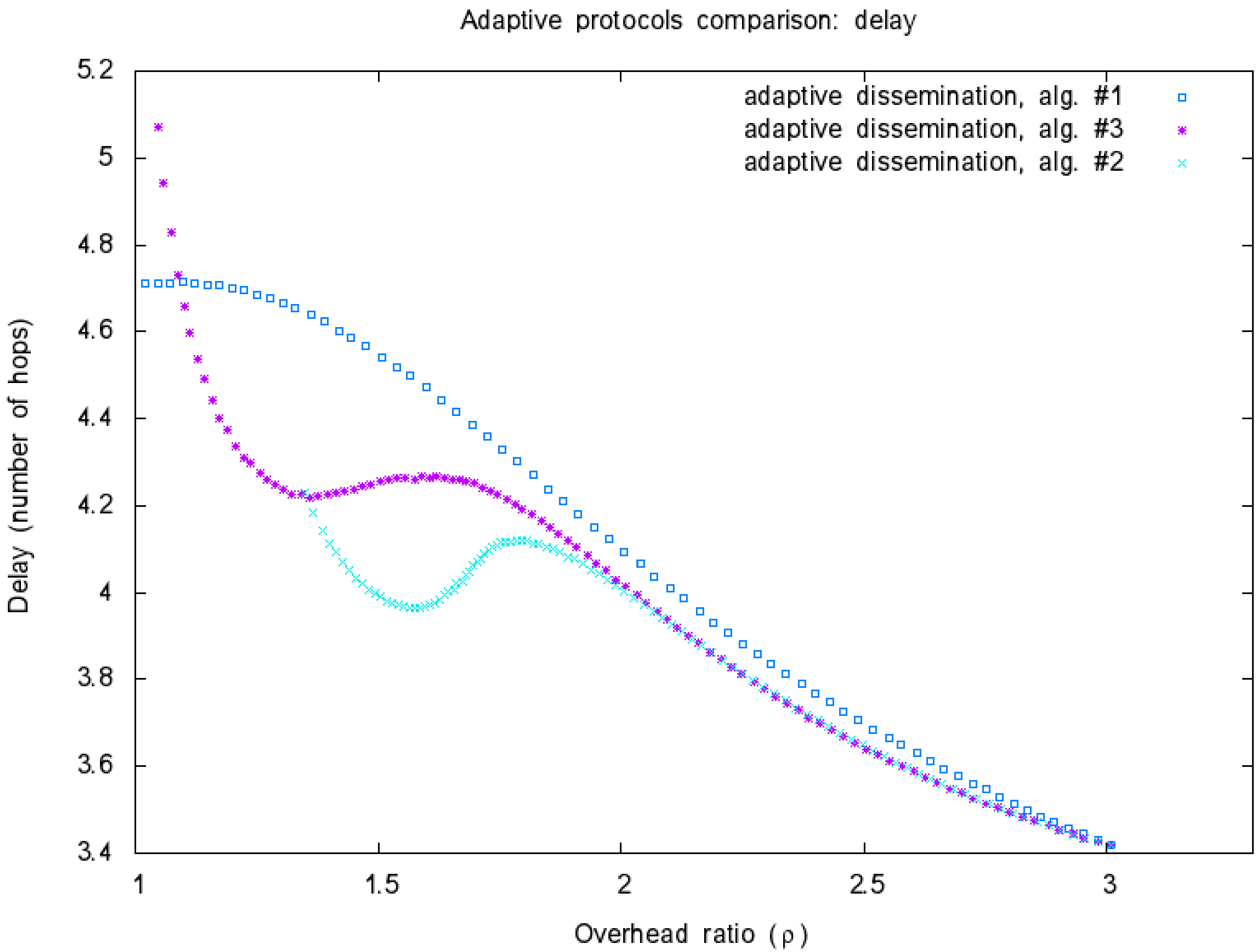
Evaluation: **delay** (number of hops)



Evaluation: **coverage** rate (%)



Evaluation: **delay** (number of hops)



LUNES: Simulation of P2P Networks

Gabriele D’Angelo

<g.dangelo@unibo.it>

http://www.cs.unibo.it/gdangelo/

Systems simulation, 2013-2014

Department of Computer Science and Engineering

University of Bologna