Bernabé Dorronsoro
Patricia Ruiz

bernabe.dorronsoro@uca.es
www.bernabe.dorronsoro.es
patricia.ruiz@uca.es
patriciaruiz.es

Universidad de Cádiz

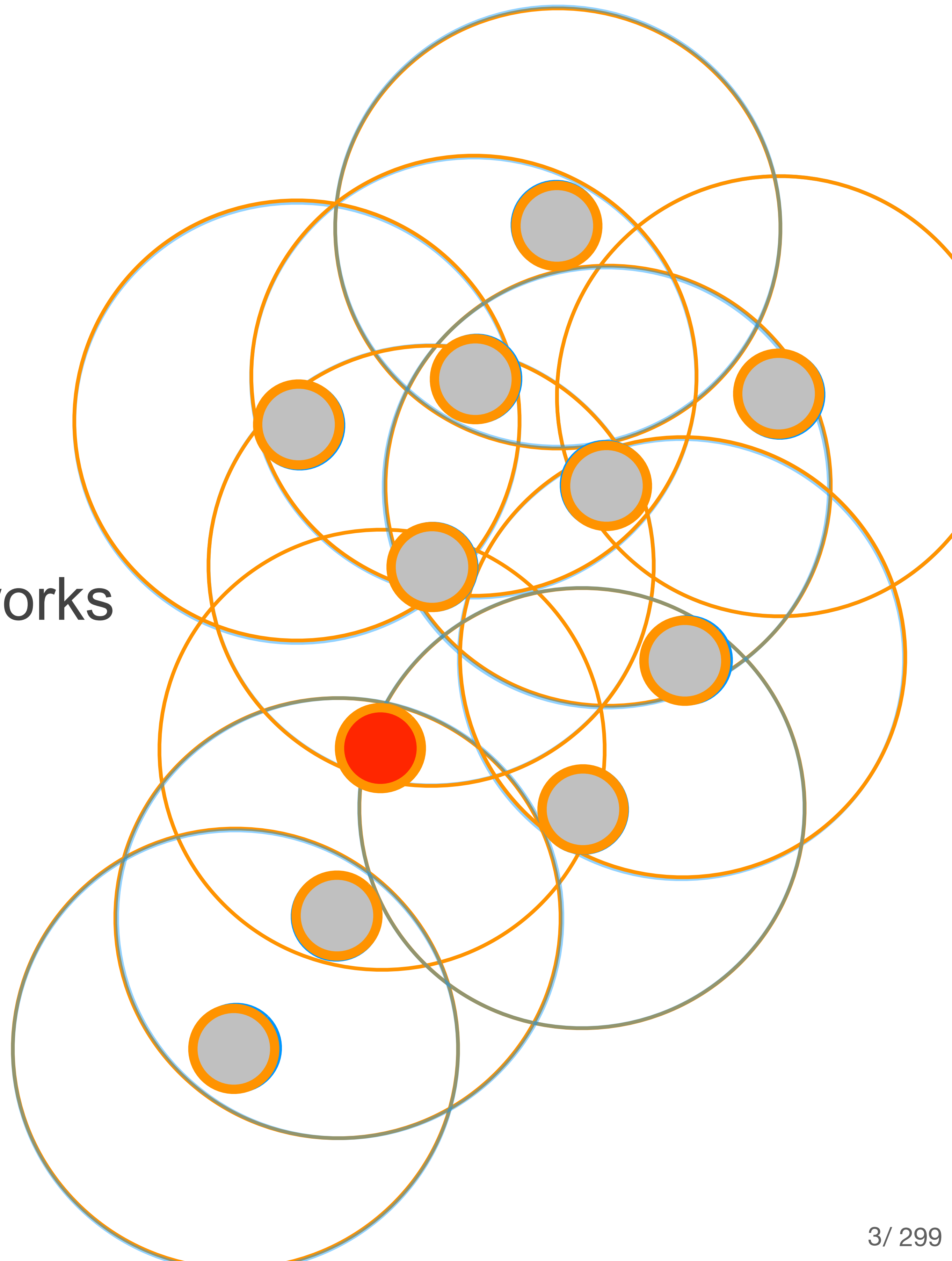
Optimization for Protocol Tuning

Network Wide Broadcast Problem

Motivations: Broadcast Algorithms

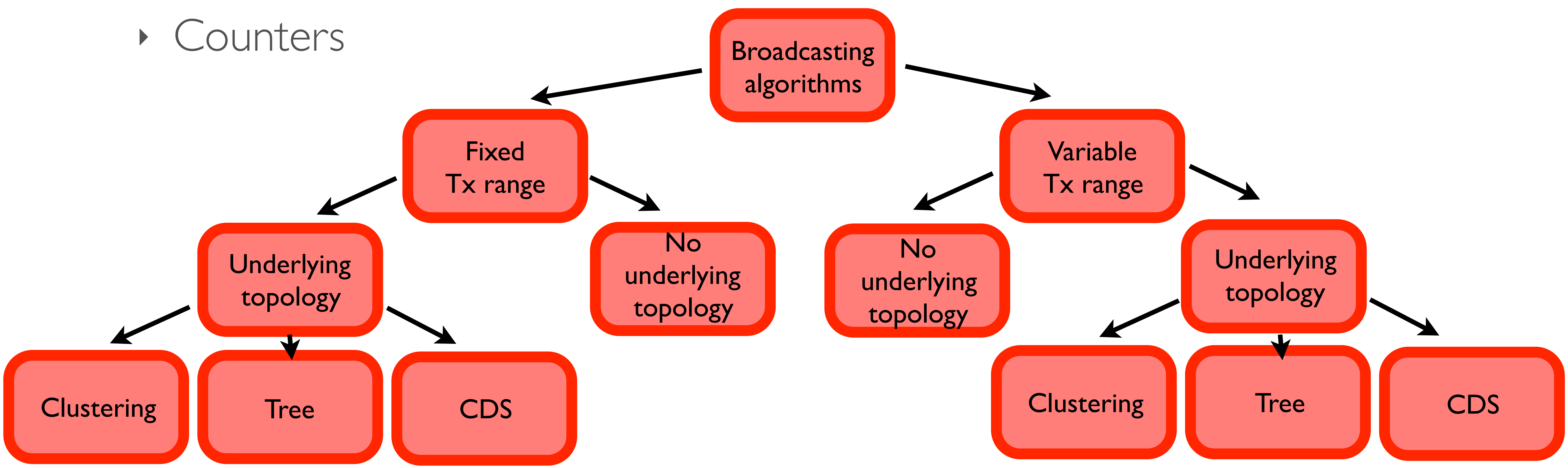
Broadcast nature of wireless networks

Cornerstone in networking



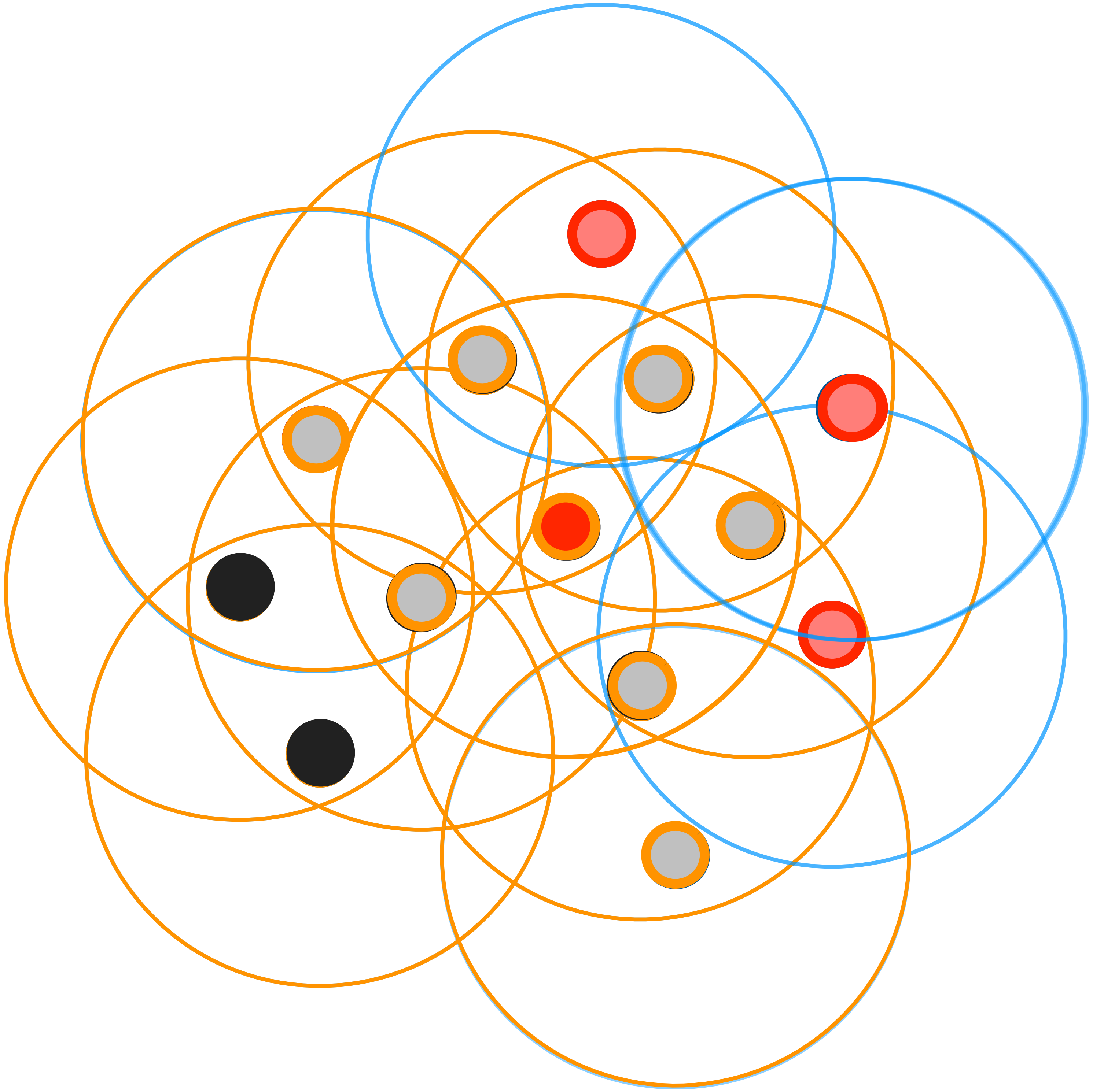
Network wide broadcasting

- Main kinds of broadcast protocols
  - Context oblivious
    - Flooding
    - Based on probabilities
  - Context aware
    - Neighborhood knowledge (1-hop, 2-hops)
    - Distances
    - Counters



The broadcast storm problem

Broadcast storm problem



Simple flooding and variants

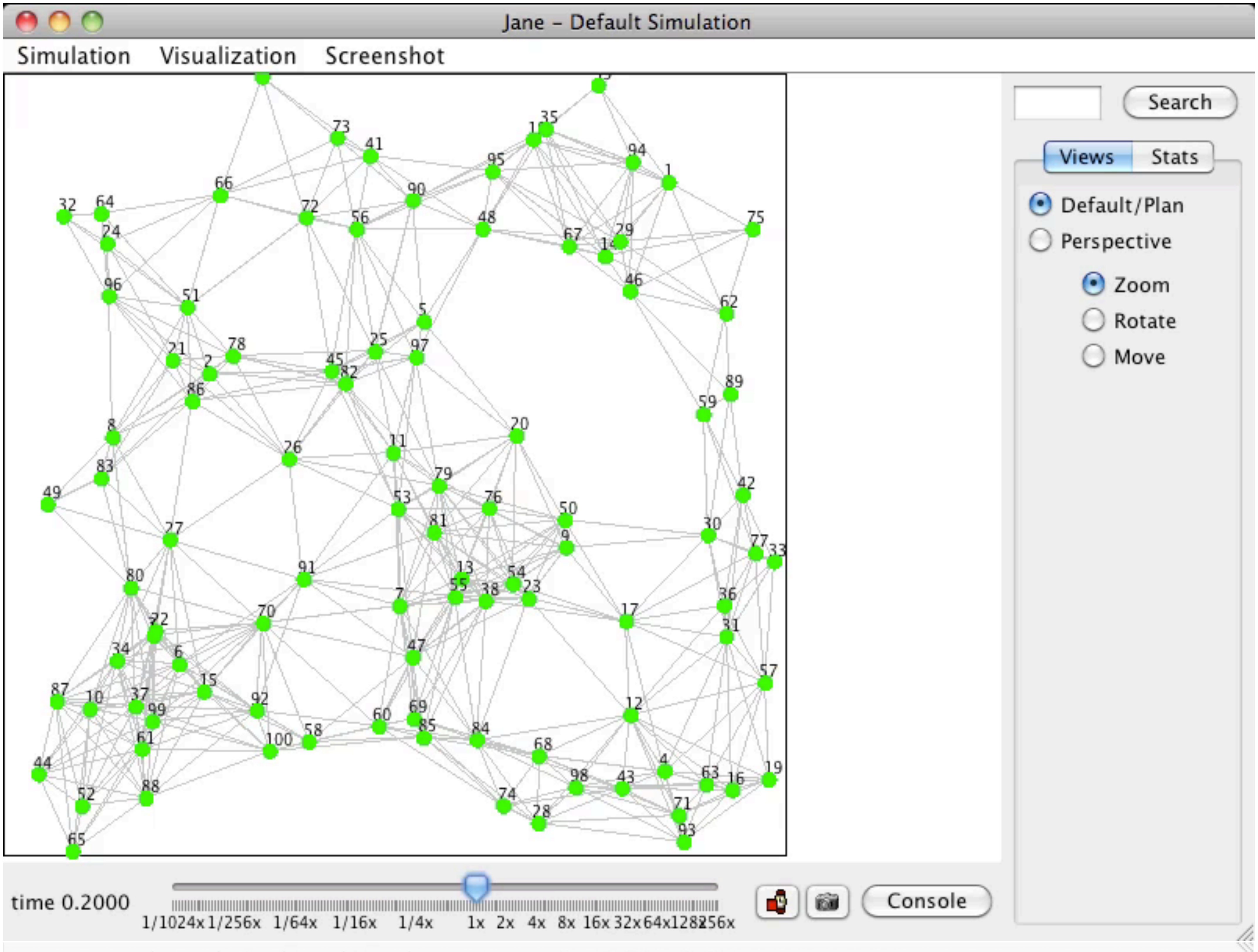
```
if (!handled) {
    resend();
    handled = true;
}
```

simple flooding

Simple flooding and variants

Universidad de Cádiz

- simple flooding, 100 devices, 500x500m, 100m coverage



Simple flooding and variants

```
if (!handled) {
    resend();
    handled = true;
}
```

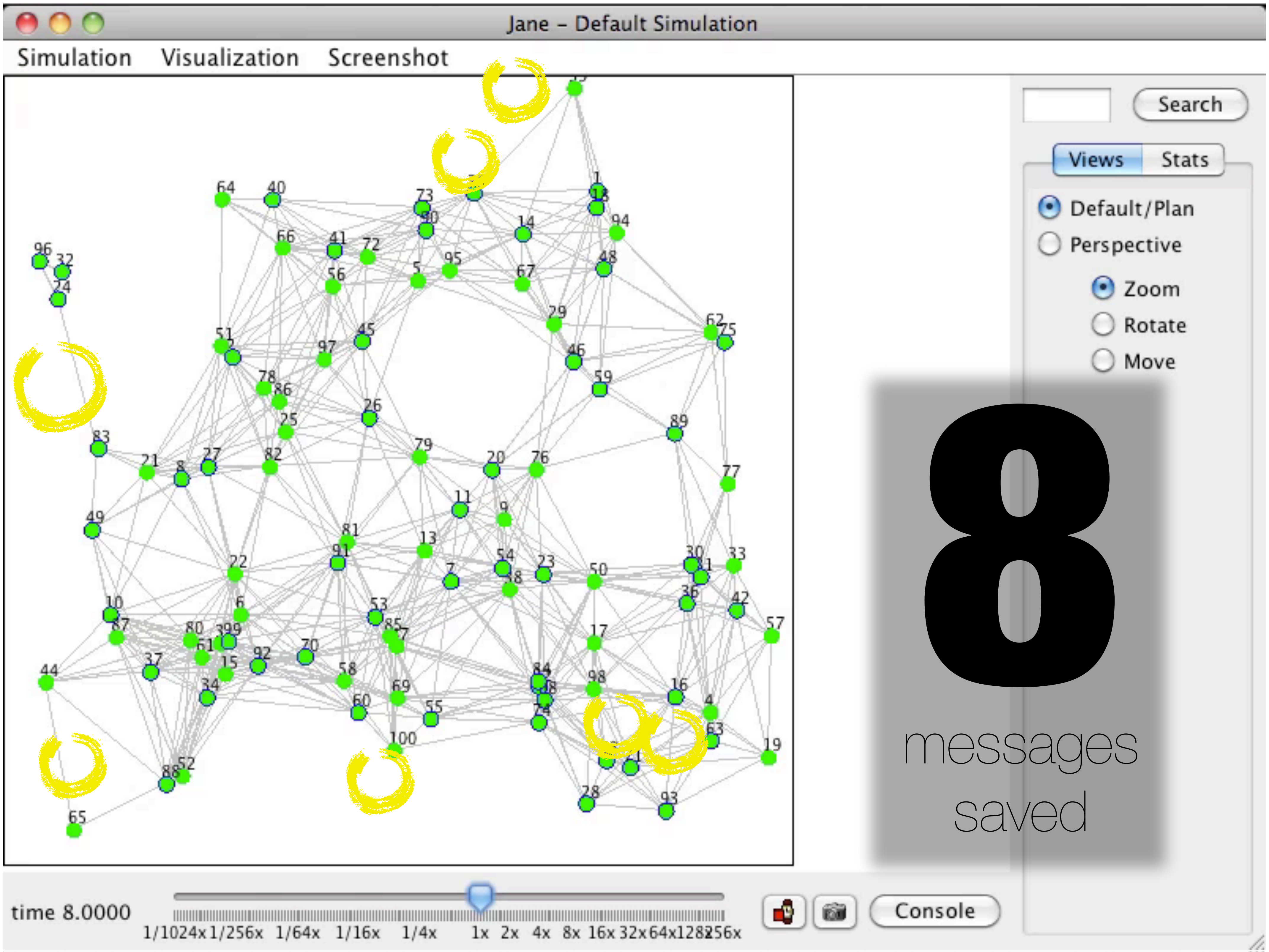
simple flooding

```
if (!handled) {
    if (haveMoreNeighborsThanSender())
        resend();
    handled = true;
}
```

flooding w/ 2-hop neighborhood

Simple flooding and variants

- two hop flooding, 100 devices, 500x500m, 100m coverage



Simple flooding and variants

```
if (!handled) {
    resend();
    handled = true;
}
```

simple flooding

```
if (!handled) {
    if (haveMoreNeighborsThanSender())
        resend();
    handled = true;
}
```

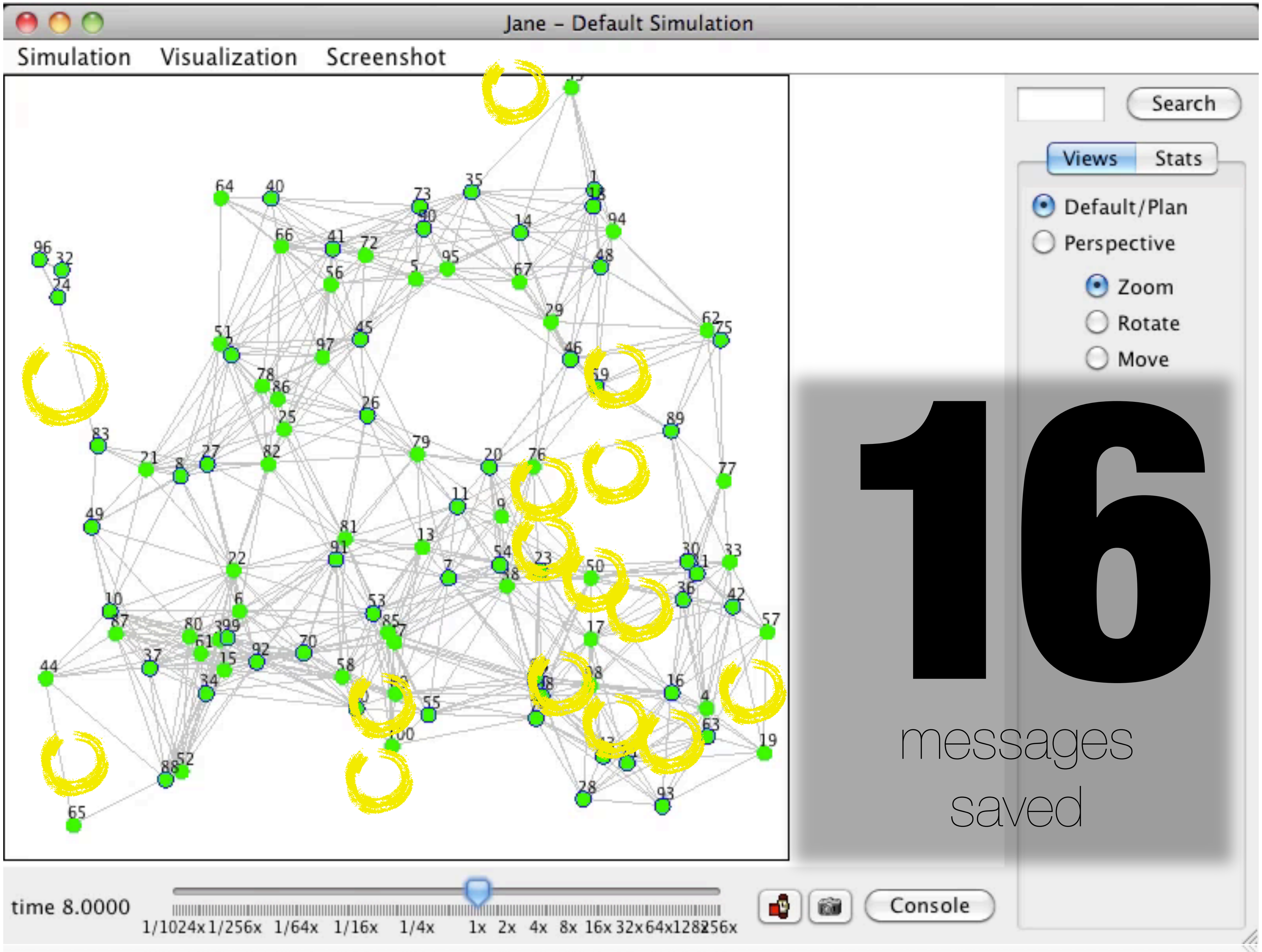
flooding w/ 2-hop neighborhood

```
if (!handled) {
    waitRandomTime();
    if (haveMoreNeighbors ThanSender())
        resend();
    handled = true;
}
```

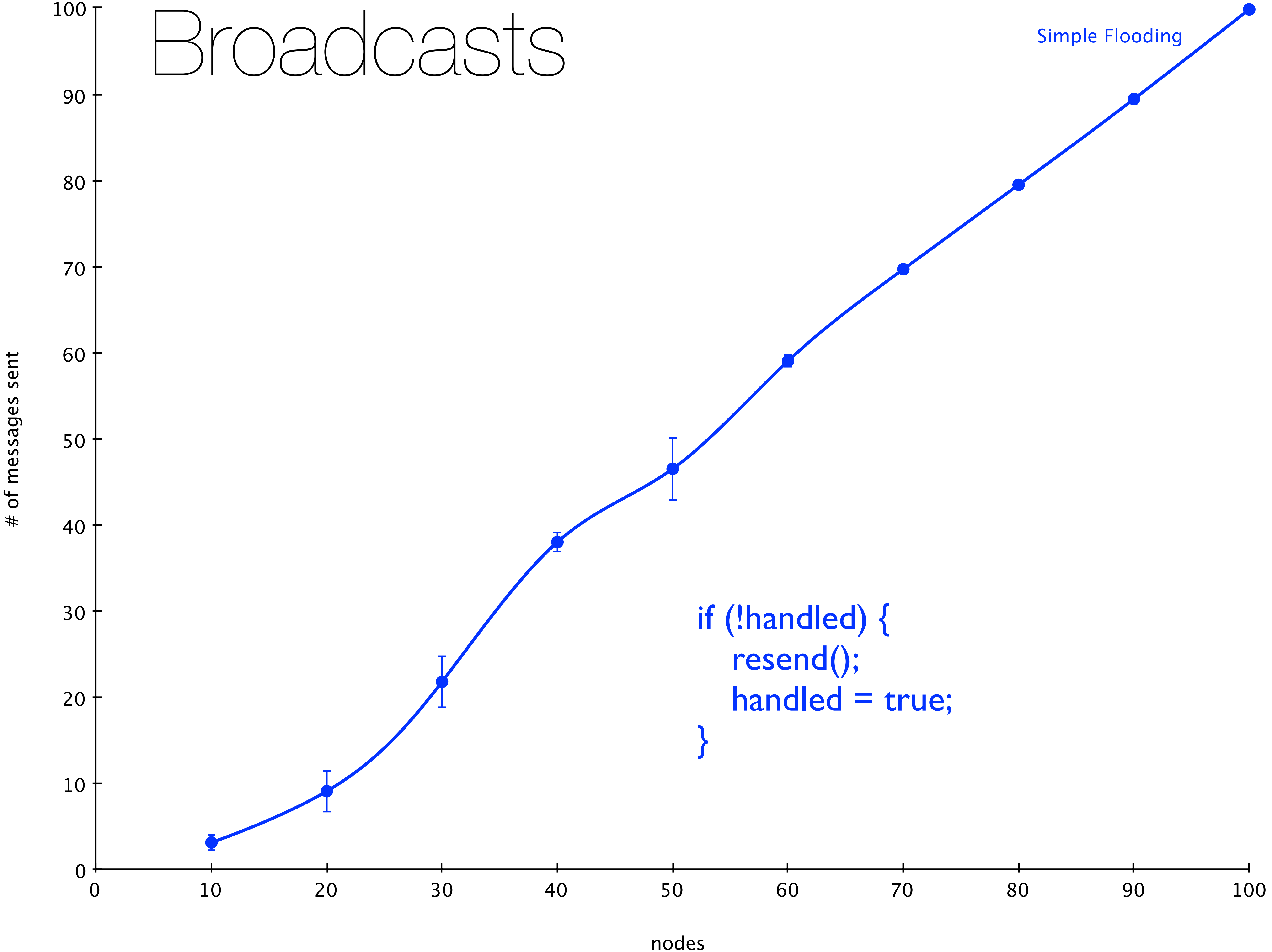
flooding w/ 2-hop neighborhood and random wait

Simple flooding and variants

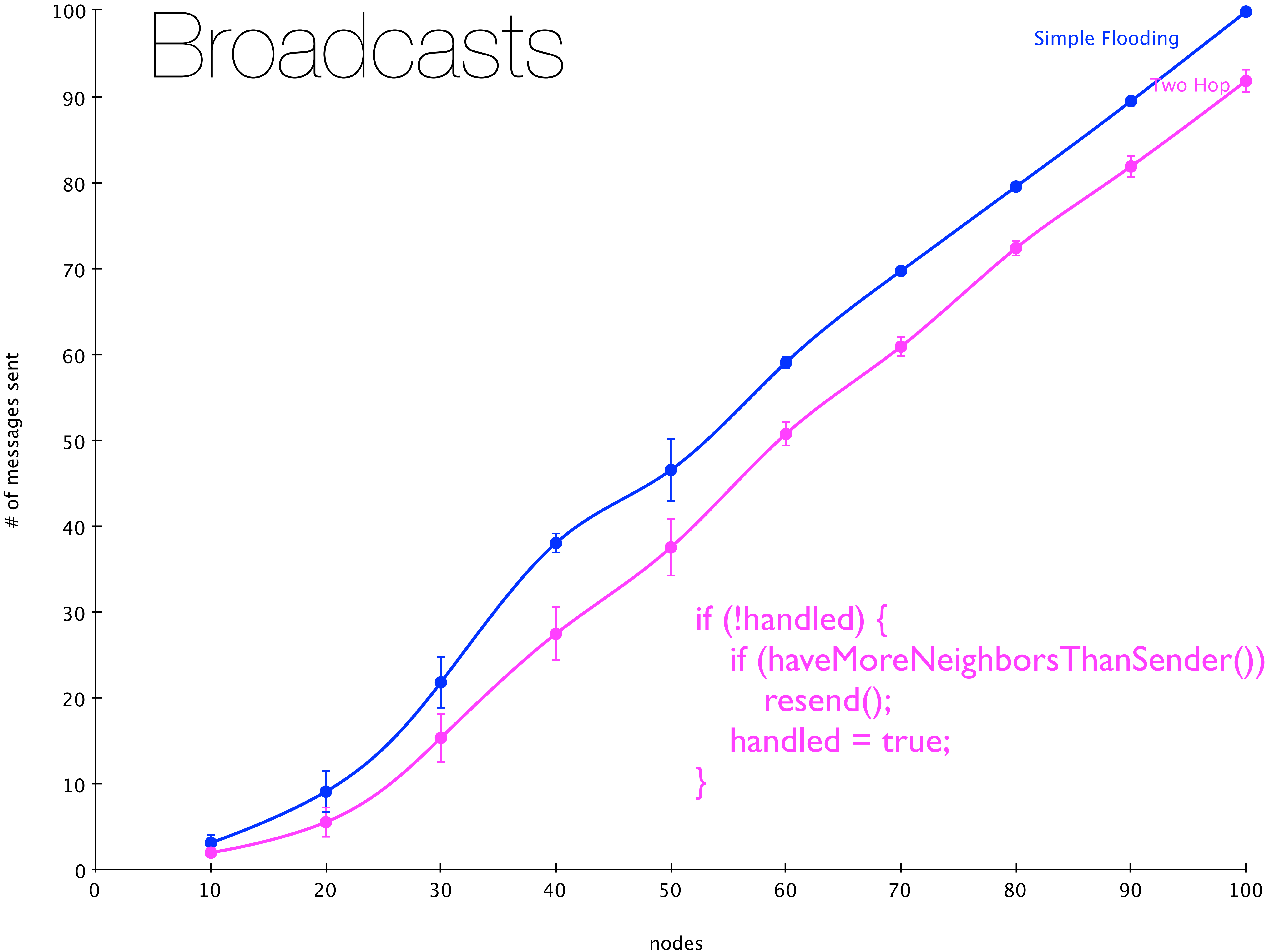
- two hop 100 ms, 100 devices, 500x500m, 100m coverage



Broadcasts

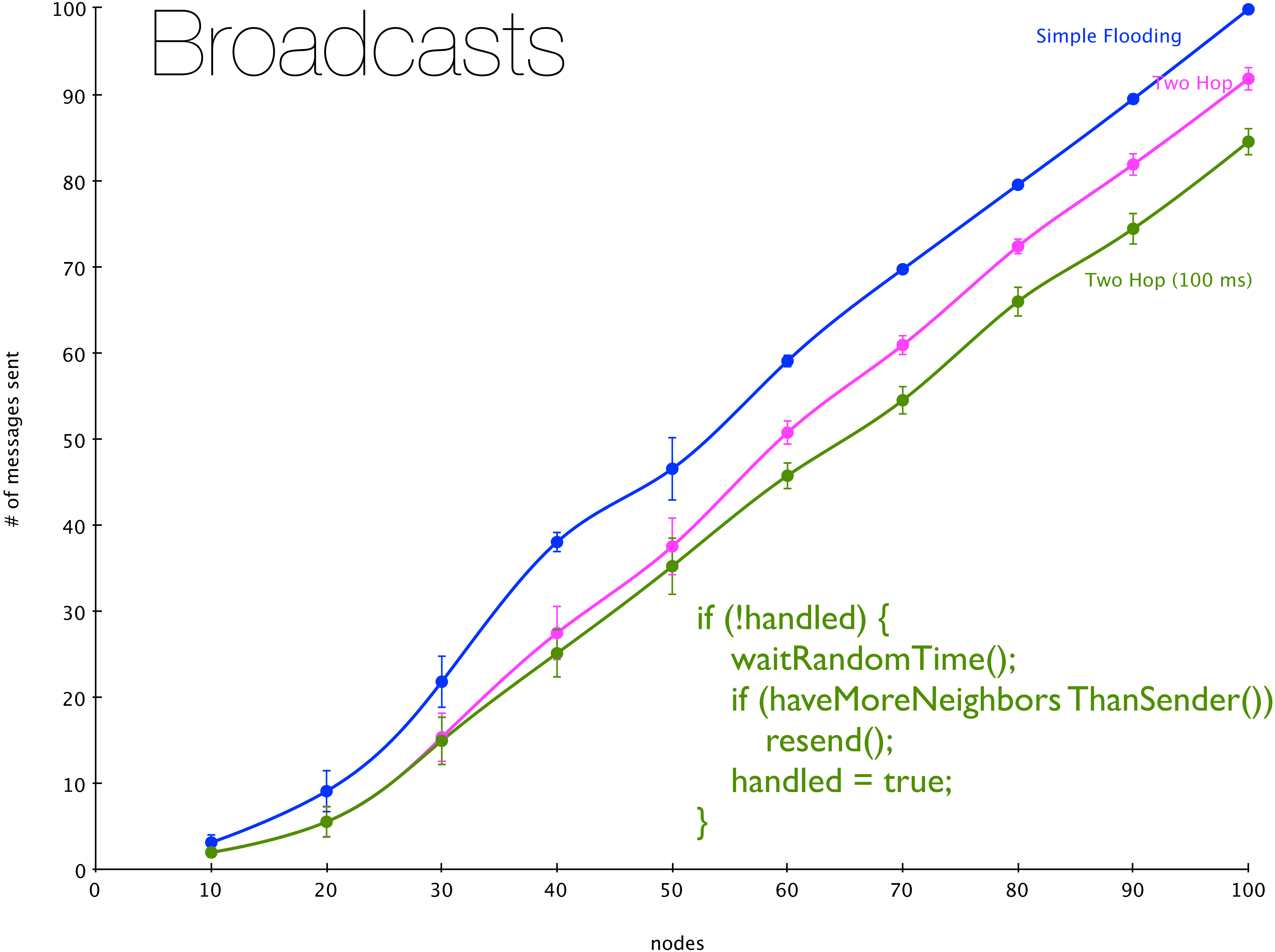

Simple Flooding
if (!handled) {
resend();
handled = true;
}
# of messages sent
nodes
0
10
20
30
40
50
60
70
80
90
100

Broadcasts

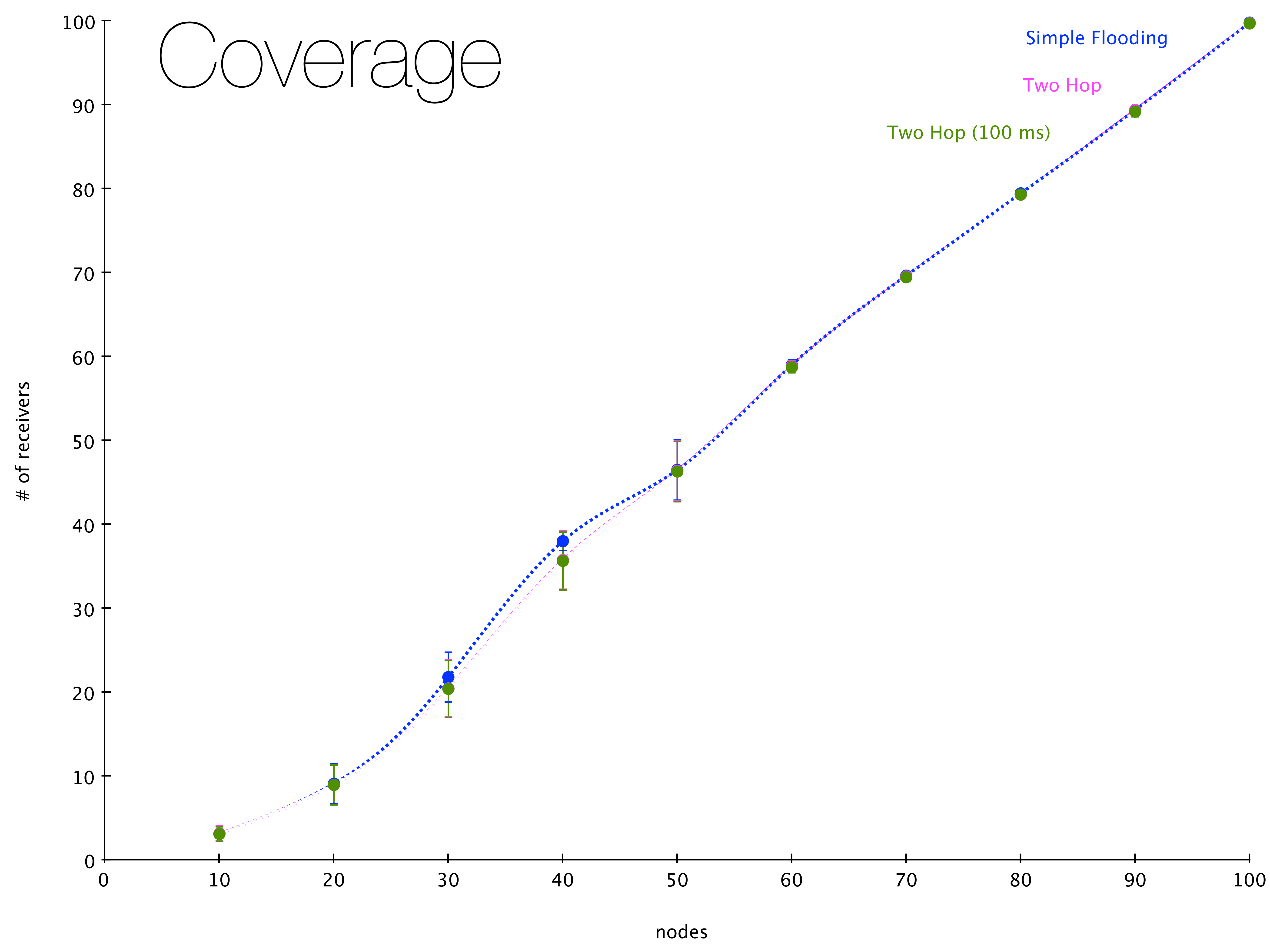


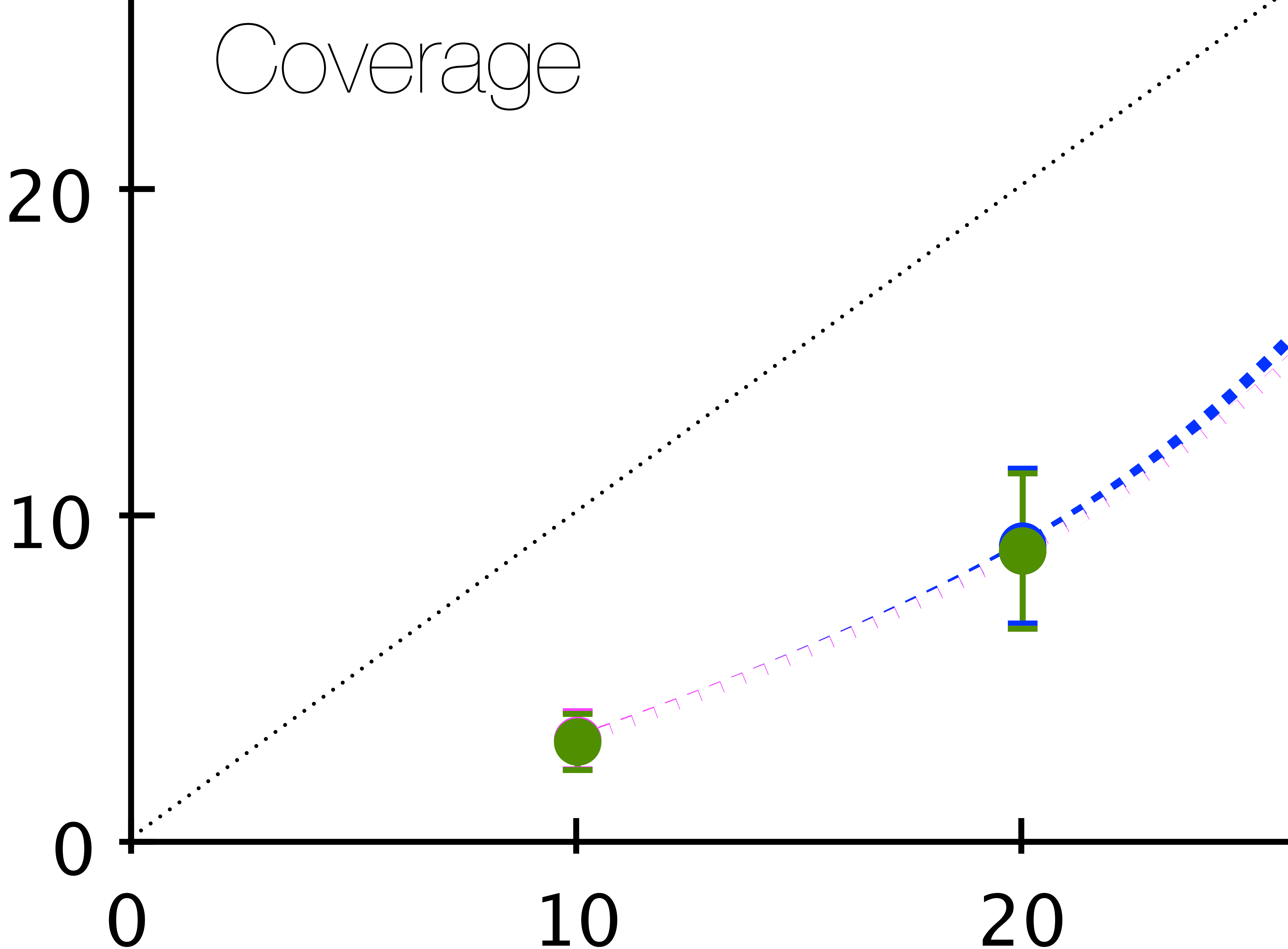

```
if (!handled) {
    if (haveMoreNeighborsThanSender())
        resend();
    handled = true;
}
```

Broadcasts



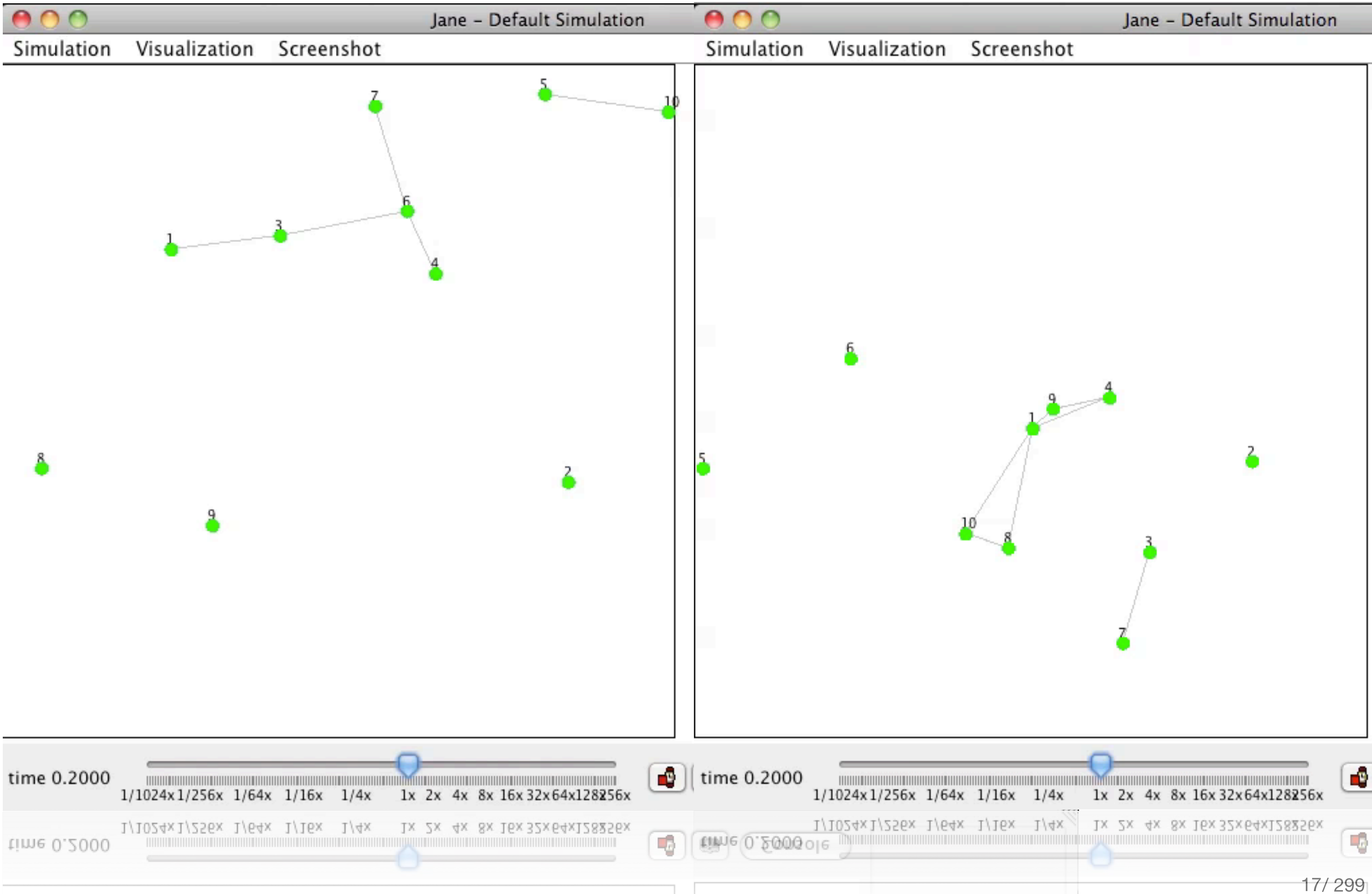

```
if (!handled) {
    waitRandomTime();
    if (haveMoreNeighbors ThanSender())
        resend();
    handled = true;
}
```


Coverage
Simple Flooding
Two Hop
Two Hop (100 ms)
# of receivers
nodes
0
10
20
30
40
50
60
70
80
90
100



Coverage
20
10
0
0
10
20

Impact of topology and starting device



Flooding with Multpoint Relays

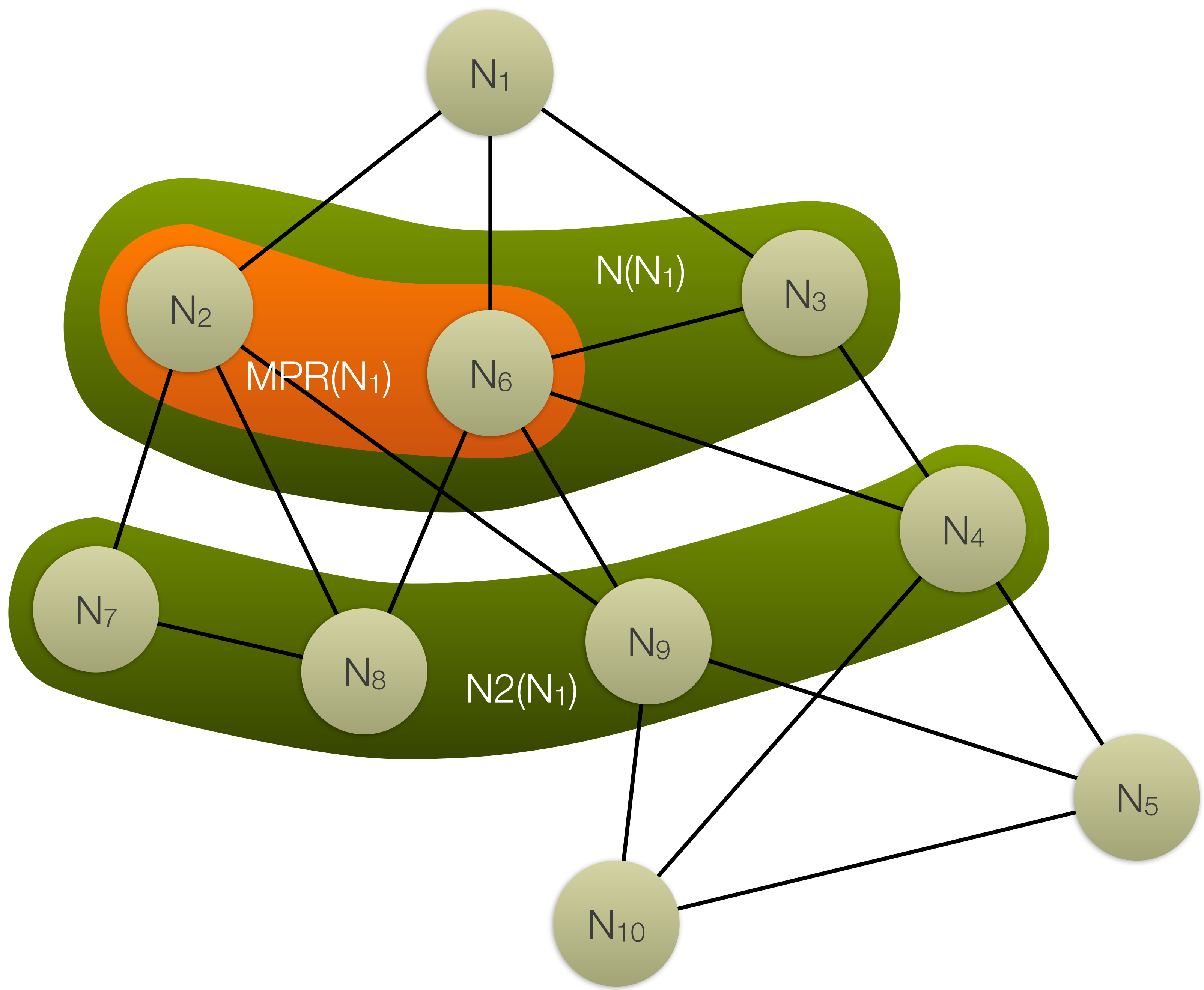
Multipoint relays

- Multipoint Relays
  - Subset of neighboring nodes used to disseminate control information
  - Build "backbone" for route selection
  - But: how to guarantee "full coverage"?
- Definitions:
  - N(Ni): set of (direct) neighbors of Ni
  - N2(Ni): set of nodes reachable in exactly 2 hops from Ni
  - MPR(Ni): set of multipoint relays selected by Ni

Multipoint relays

- Multipoint Relay must adhere to following conditions:
  - MPR(Ni) must be subset of N(Ni)
  - Every node from within N2(Ni) must be reachable (directly) from at least one member of MPR(Ni)
- Note explicitly: we don't require MPR(Ni) to be minimal
  - However, the fewer nodes in MPR(Ni), the more savings …

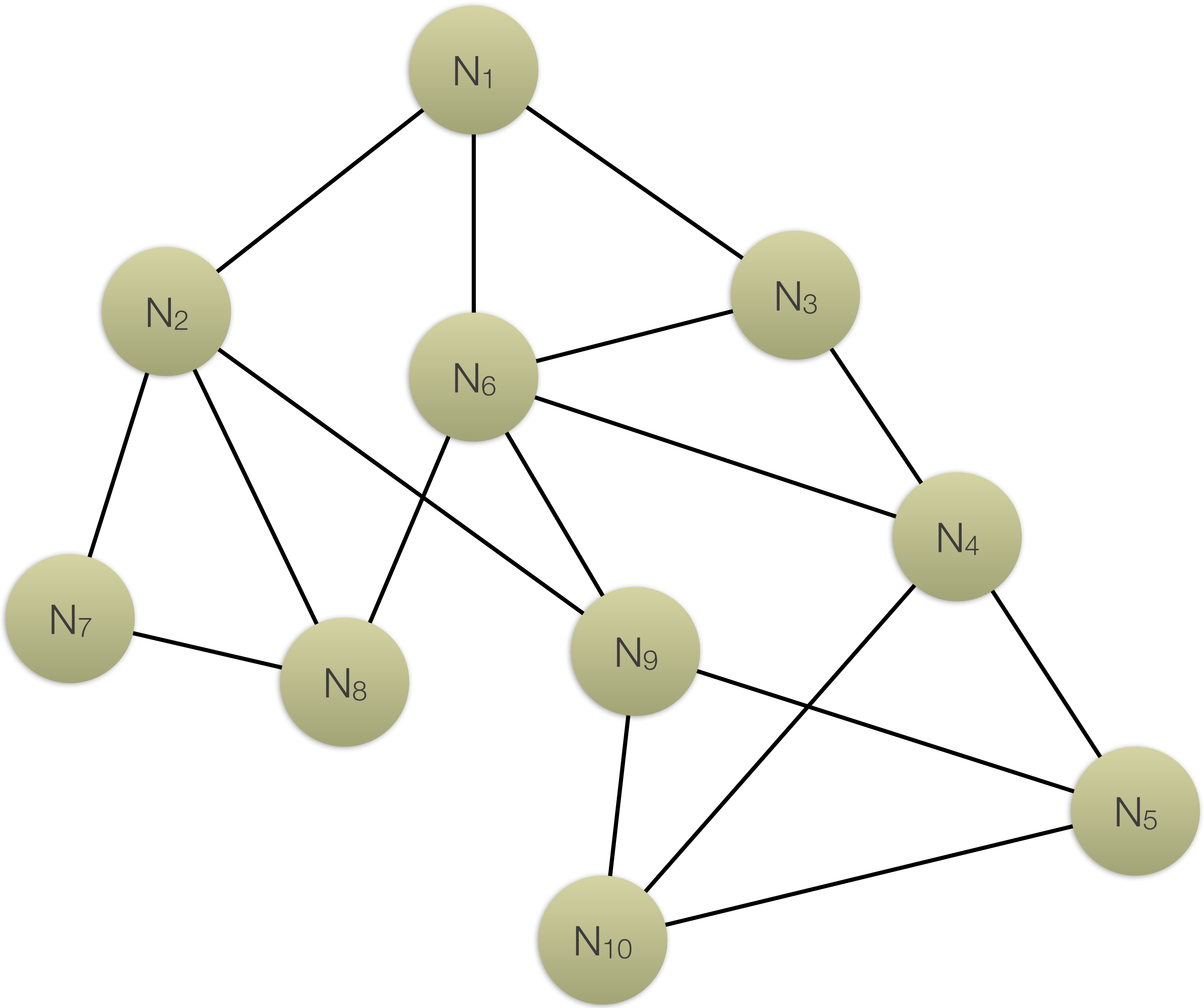
Multipoint relay example



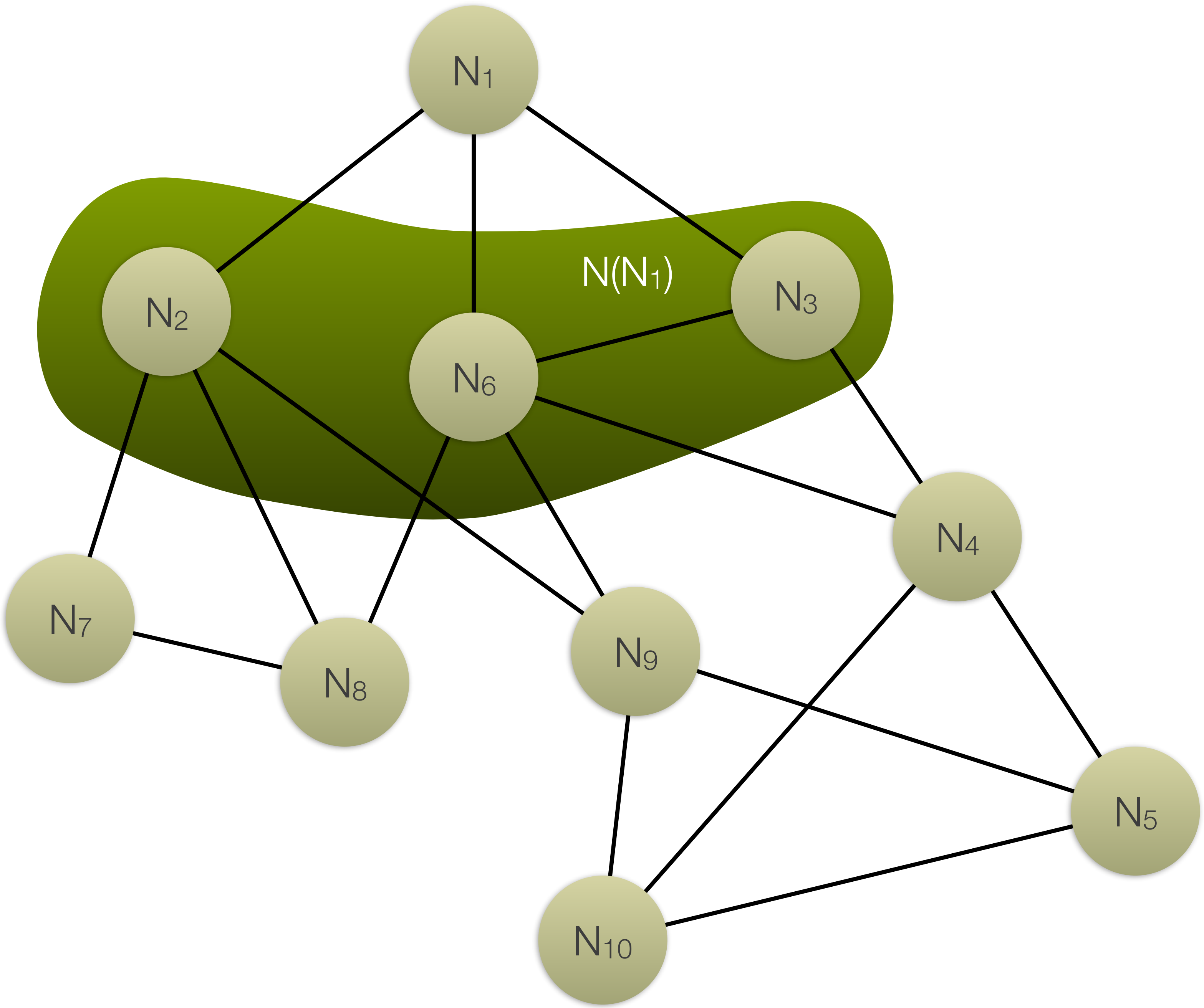
MPR determination

- Algorithm to determine MPR(Ni)
  - Start with MPR(Ni) = {}
  - For all nodes in N2(Ni) having a single link to a node Nk in N(Ni) only, add Nk to MPR(Ni)
  - As long as there are nodes in N2(Ni) not being reachable from nodes in MPR(Ni):
    - Add to MPR(Ni) those node from N(Ni) via that the most non-reachable nodes from N2(Ni) can be reached
      - If there are multiple such nodes, select the one having the most neighbors
  - As long as there are nodes Nk in MPR(Ni) such that from MPR(Ni)\{Nk} all nodes from N2(Ni) are still reachable, then discard Nk from MPR(Ni)

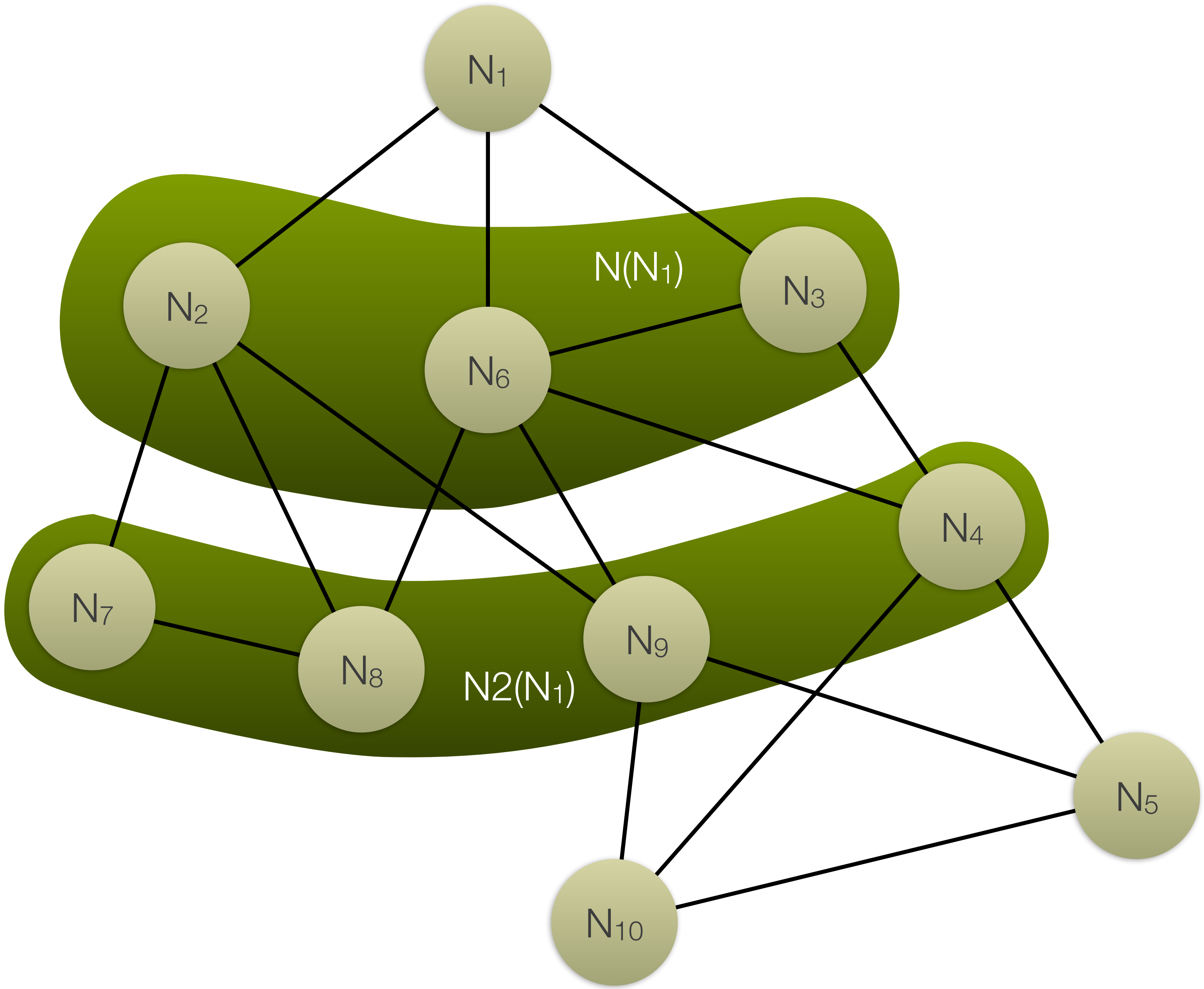
MPR determination example



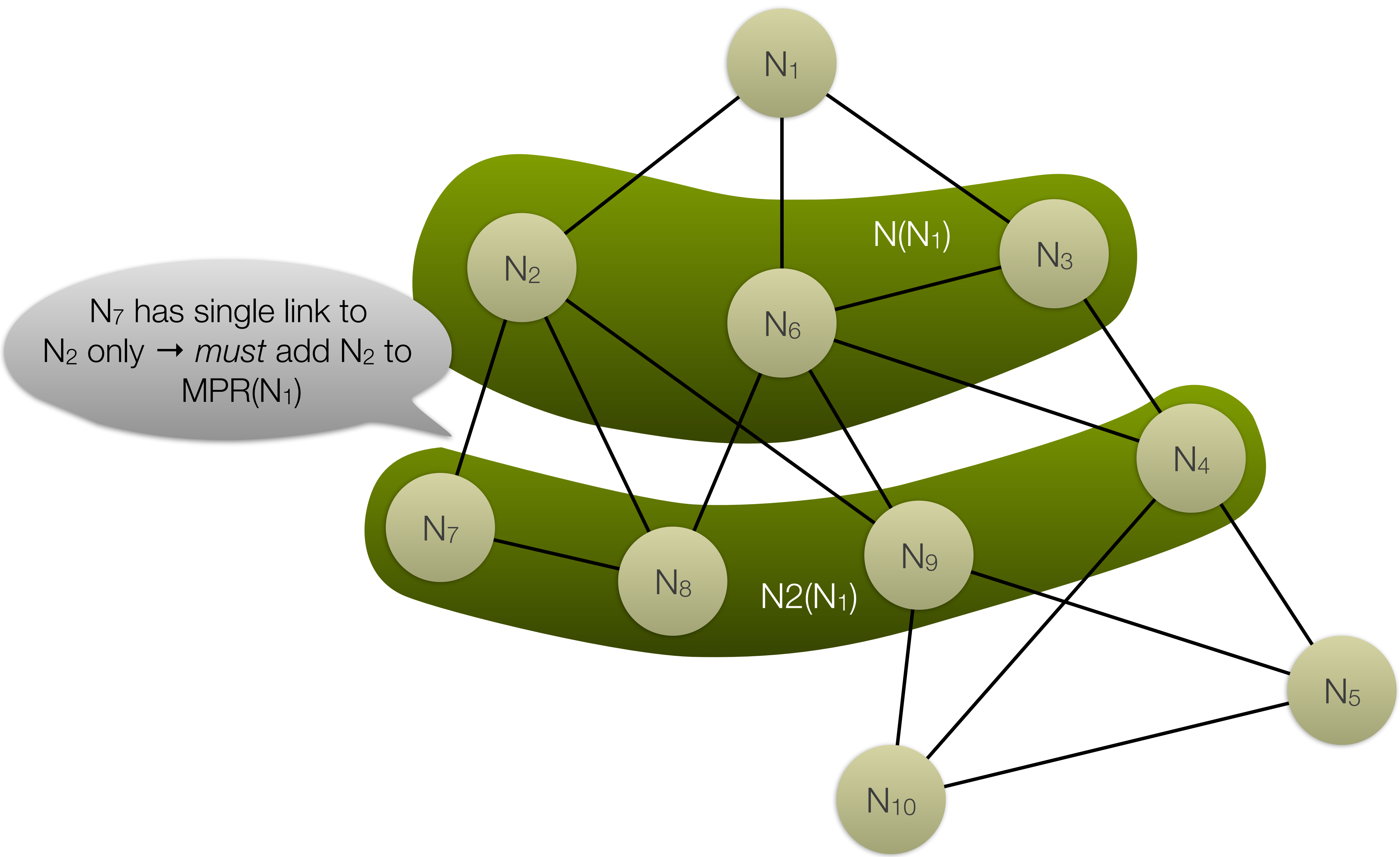
MPR determination example



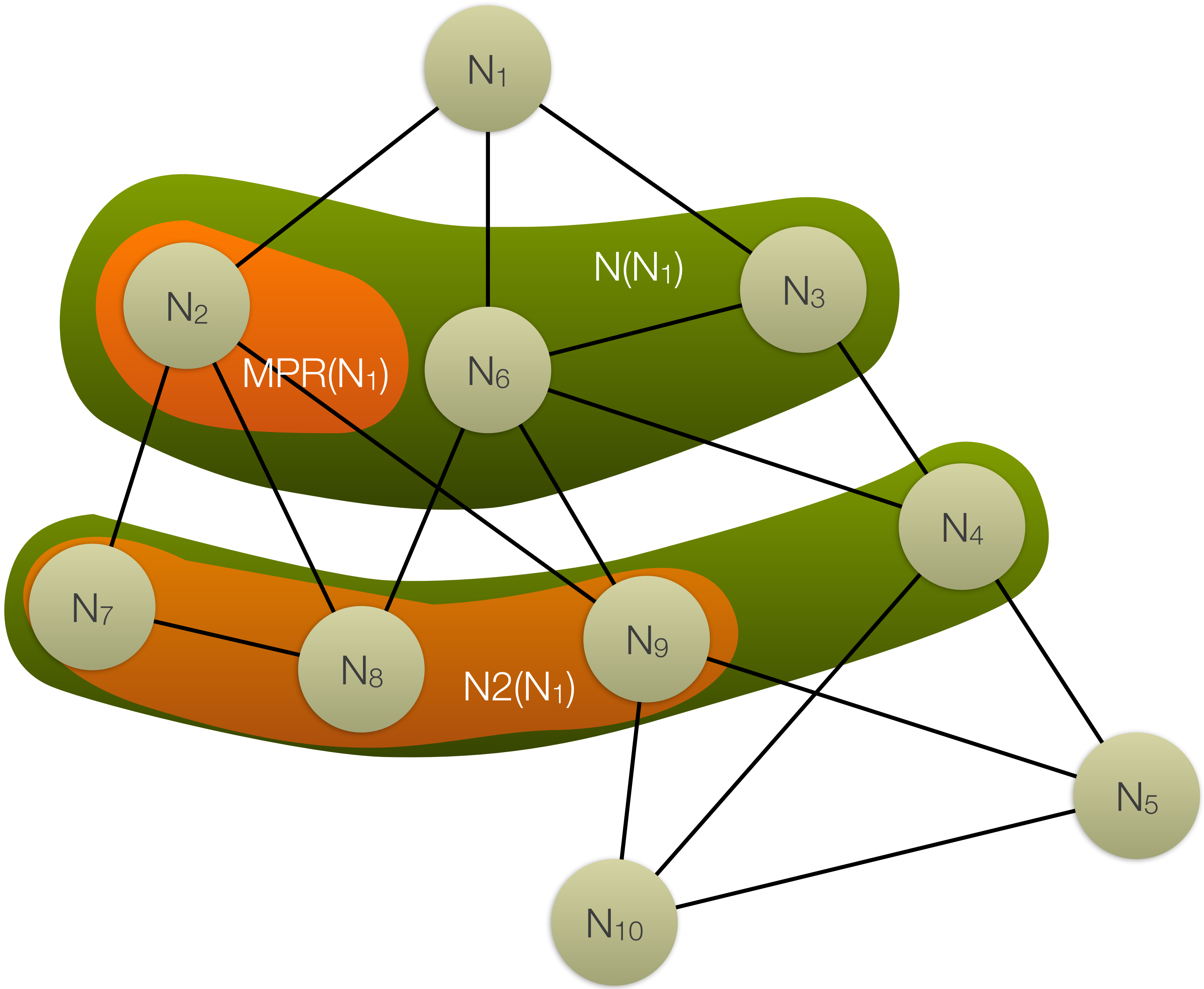
MPR determination example



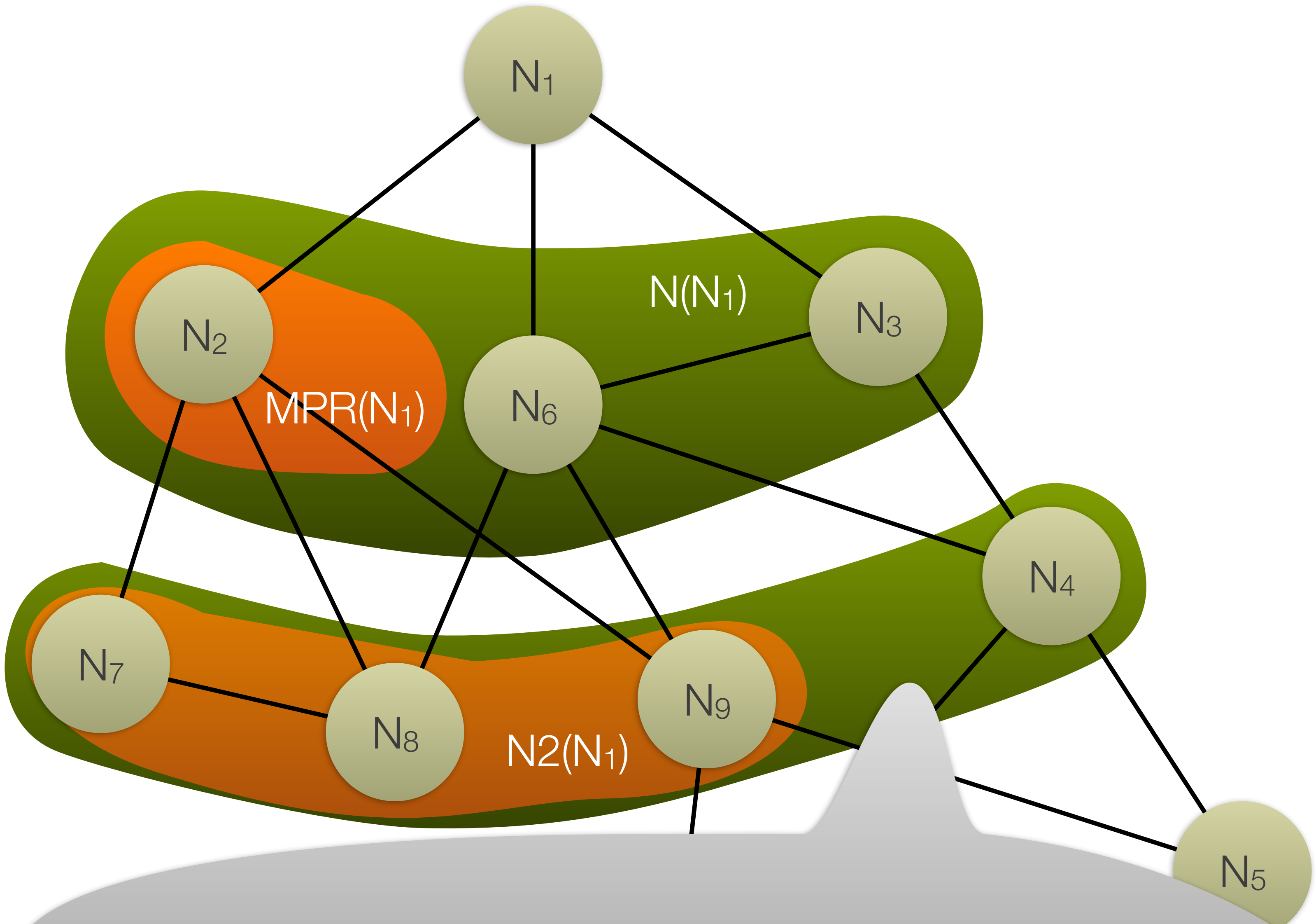
MPR determination example



MPR determination example

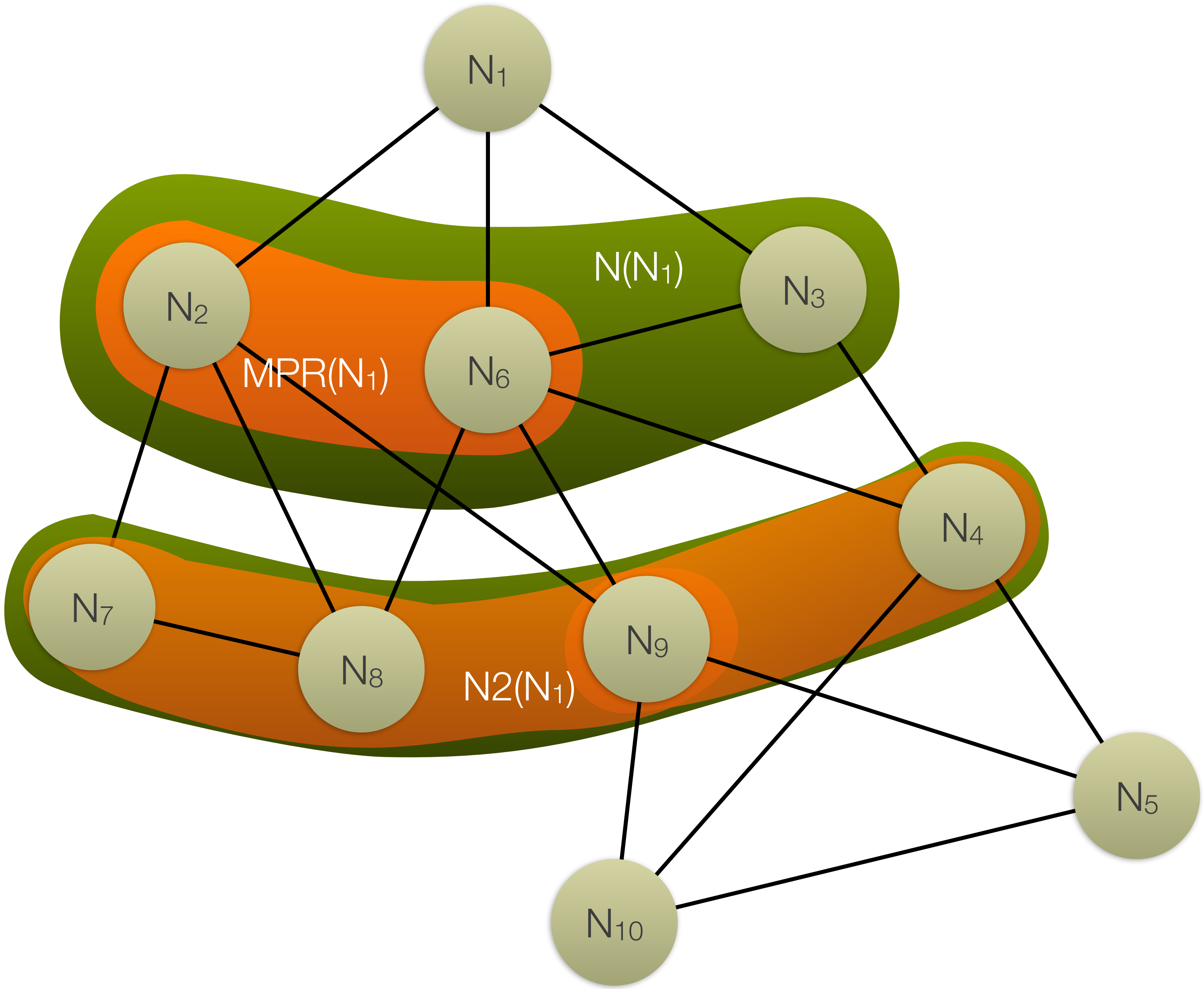


MPR determination example

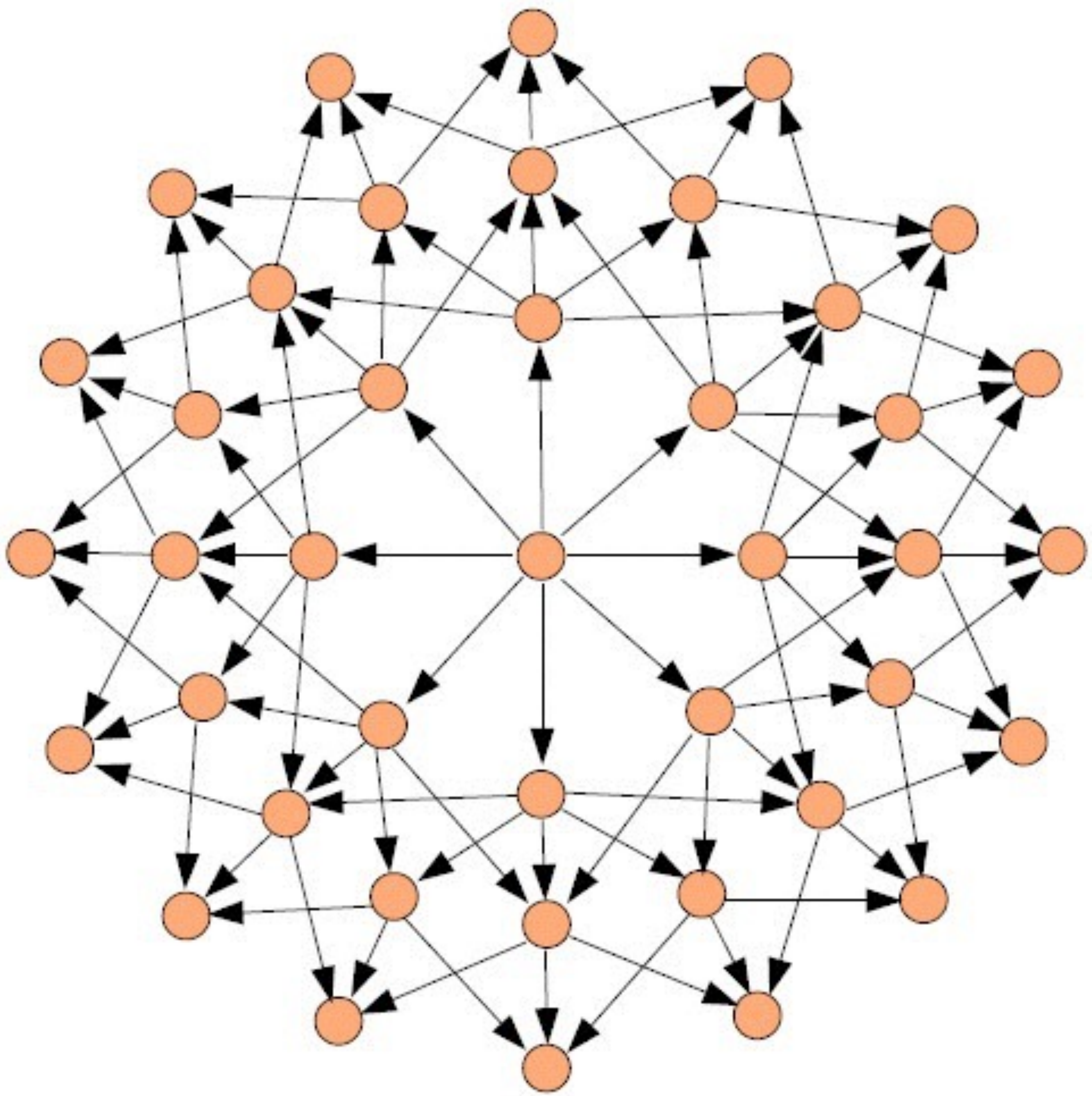


N_4 not yet reachable from MPR(N_1)
→ add node from that most (non-reachable) nodes can be reached
→ add N_6 to MPR(N_1) because it has the most children

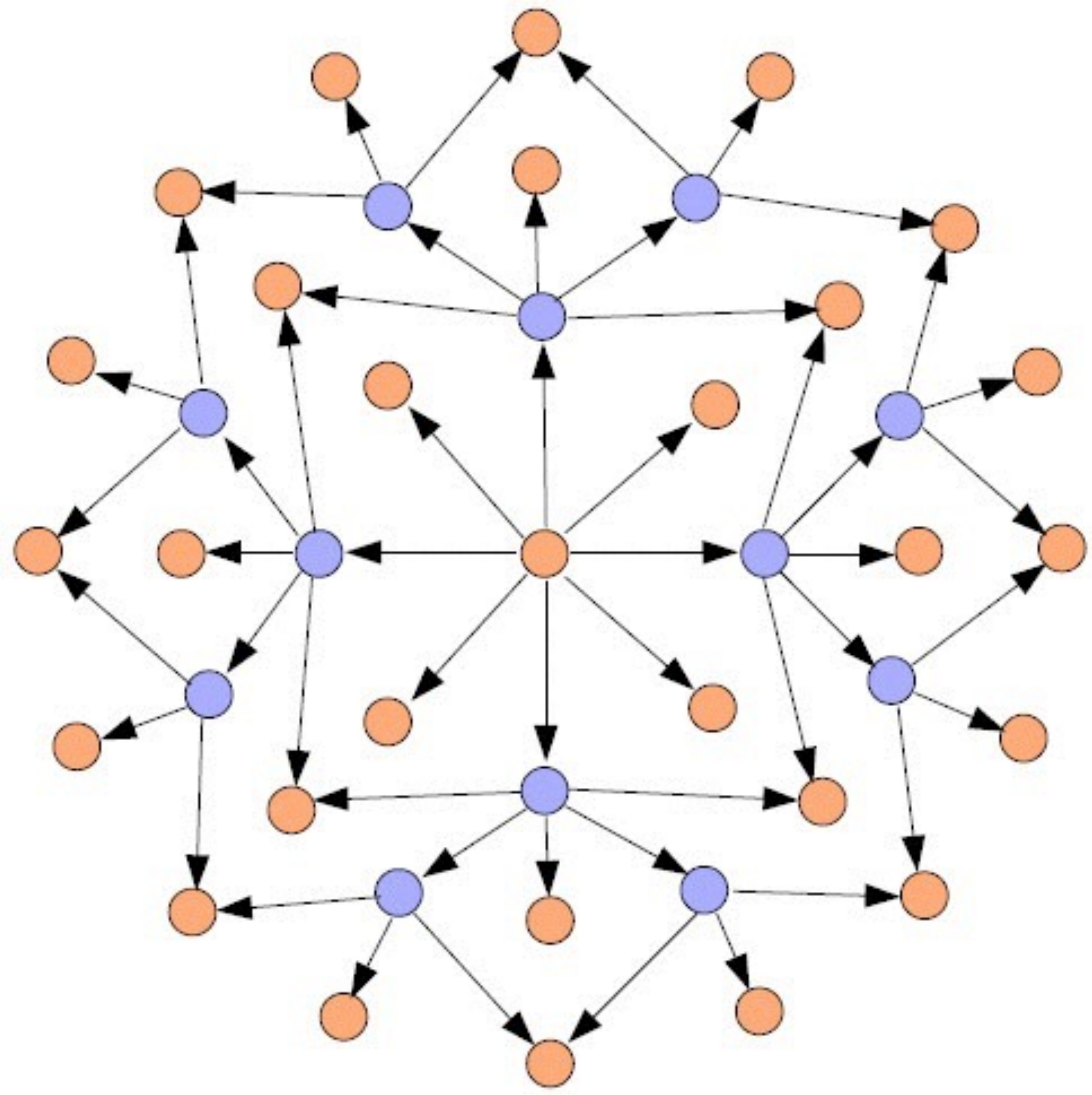
MPR determination example



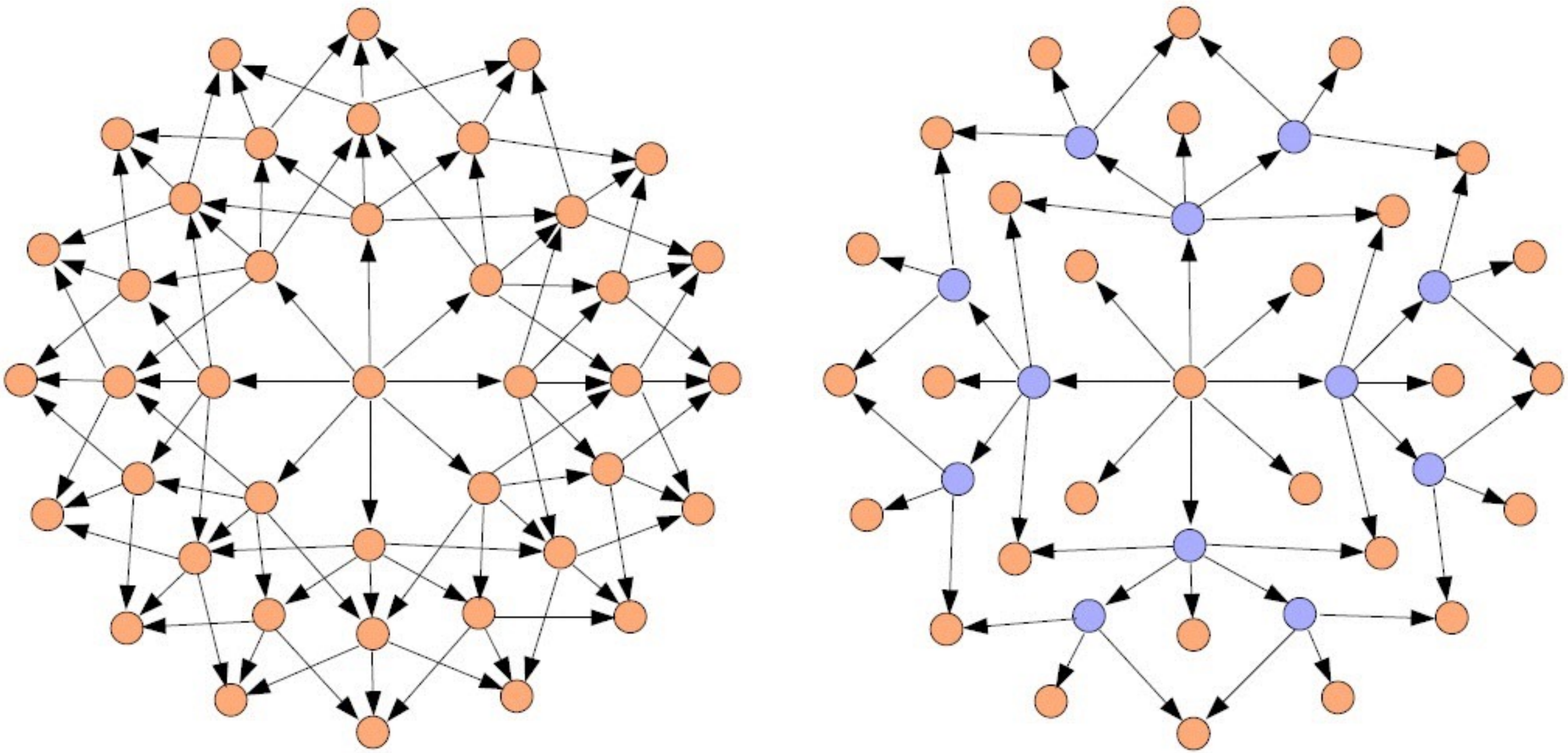
Flooding revisited



MPR-Flooding



Simple vs. MPR-flooding



Delayed Flooding with Cumulative Neighbourhood (DFCN)

Delayed Flooding with Cumulative Neighb.

- DFCN is an efficient broadcasting algorithm
- DFCN aims to minimize the network load taking into account the network density and also to avoid collisions
- DFCN attaches to the BC message a list with the neighbors of the sender
- Messages are univocally identified
- A message received more than once is discarded
- DFCN has
  - Proactive behavior
  - Reactive behavior

Delayed Flooding with Cumulative Neighb.

- When receiving a message — Reactive behavior
  - Set a Random Assessment Delay (RAD)
  - When RAD expires, take forwarding decision
  - If density ≤ safeDensity
    - Add neighbors in the message
    - Forward the message
  - Otherwise compute benefit of forwarding
  - If benefit ≥ minBenefit
    - Add neighbors in the message
    - Forward the message

$$\text{benefit} = \frac{|\,\text{TotalNumberOfNeighbors} - \text{NeighborsWithMessage}\,|}{\text{TotalNumberOfNeighbors}}$$

Delayed Flooding with Cumulative Neighb.

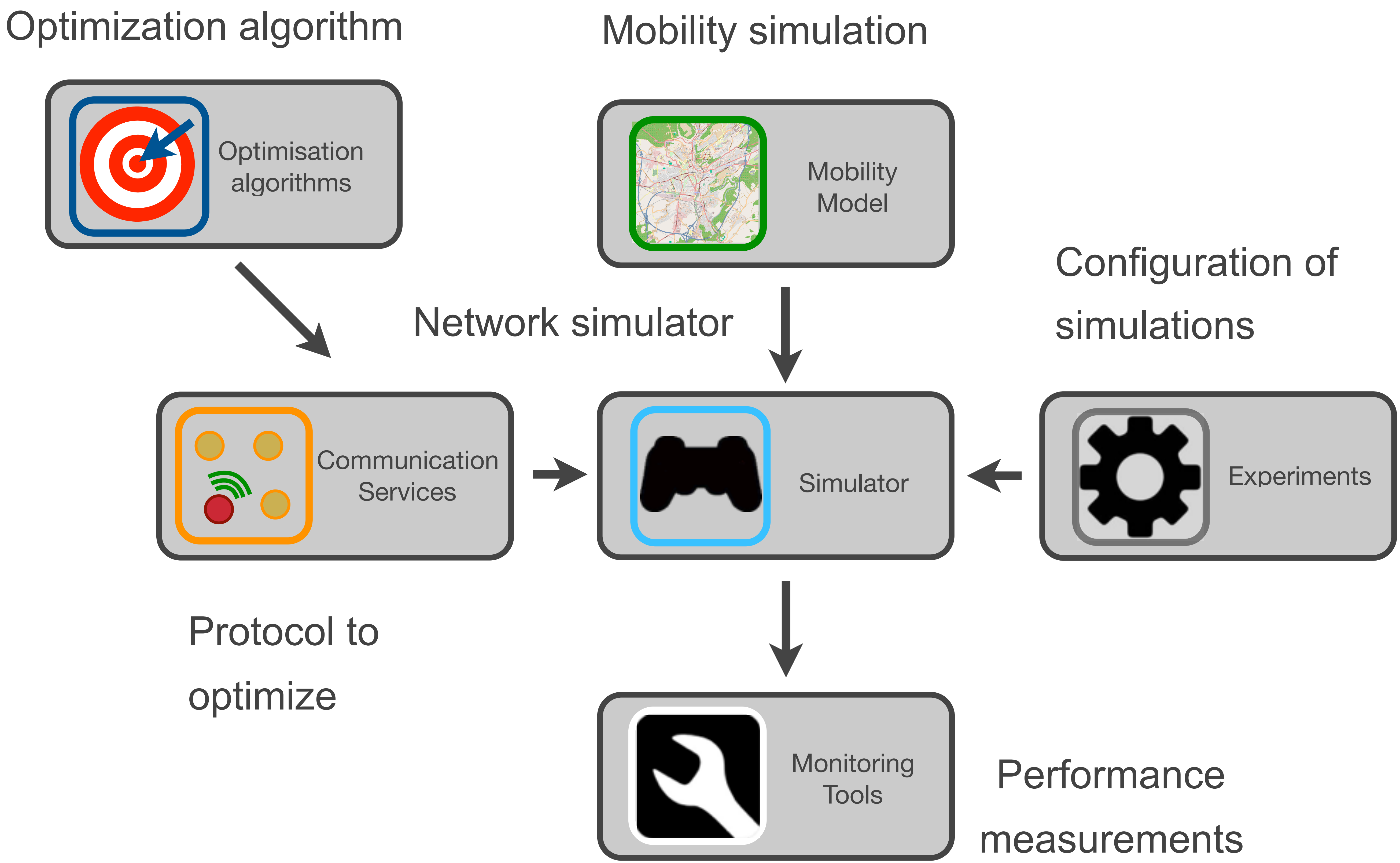
- When a new neighbor is detected — Proactive behavior
  - Avoid collisions
  - If the number of neighbors < proD
  - Set RAD to 0
- For sparse networks
  - Message is immediately candidate for resubmission
  - Promotion of diffusion

DFCN configuration parameters

- DFCN relies on different thresholds:
  - minBenefit: minimum gain for rebroadcasting
  - RAD interval: random delay before rebroadcasting
    - lowerBoundRAD
    - upperBoundRAD
    - lowerBoundRAD ≤ upperBoundRAD
  - proD: maximum number of neighbors for which it is still needed to use proactive behavior
  - safeDensity: maximum value of the local network density for rebroadcasting all messages

Multi-objective Optimization of DFCN

Optimization of DFCN



Optimization algorithm

- Maximize coverage
- Minimize number of messages
- Minimize broadcasting time

Population size	100 (ssGA, NSGAII)
	10×10 (cGA, CellDE)
	$100\times$ number of subpopulations (CCGA, CCNSGAII)
Termination Condition	$10,000$ function evaluations
Selection	Binary tournament (BT)
	Current individual + BT for cGA
Neighborhood	C9 for cellular topologies
Crossover probability	$p_c = 1.0$
Mutation probability	$p_m = 1/\text{chrom\_length}$

DFCN optimization problem representation

- Problem representation

minBenefit	lowerBoundRAD	upperBoundRAD	proD	safeDensity
Double	Double	Double	Integer	Integer

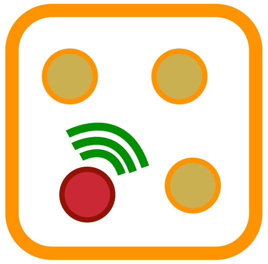
- CCNSGAII

minBenefit	lowerBoundRAD	upperBoundRAD	proD	safeDensity
16 bits	16 bits	16 bits	8 bits	8 bits

Optimization algorithm parameters

- Variable ranges

$minGain$	[0.0, 1.0]
$lowerBoundRAD$	[0.0, 10.0] seconds
$upperBoundRAD$	[0.0, 10.0] seconds
$proD$	[0, 100] devices
$safeDensity$	[0, 100] devices



- DFCN broadcasting protocol

Network and mobility simulators parameters

- Network simulator: ns3
- Transmission power: 16.02 dBm
- Signal loss model: Log distance
- IEEE 802.11b
- Simulation sime: 40 s

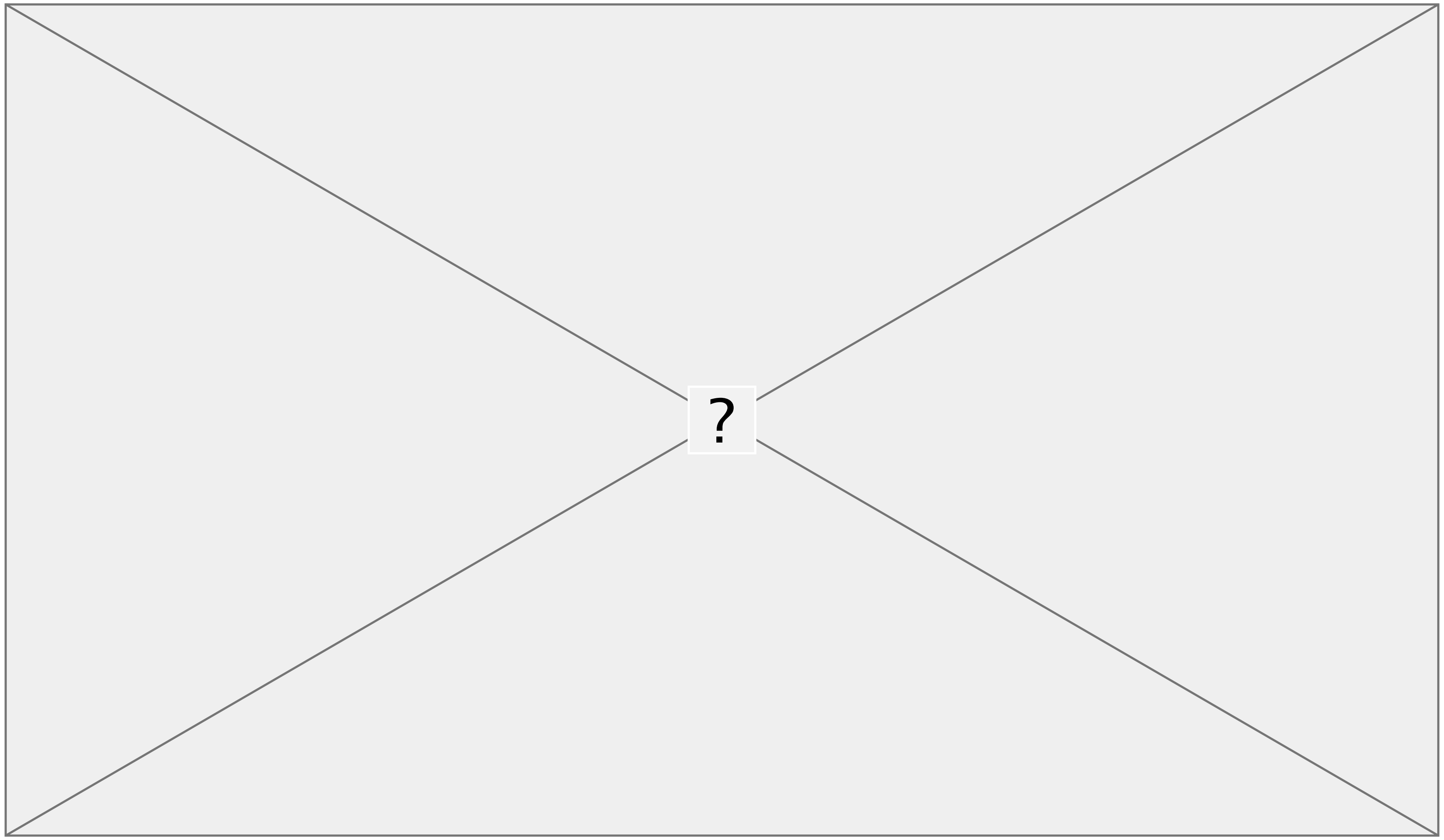
- Mobility simulator: ns3
- Random waypoint mobility model
- Speed: [0, 2] m/s
- Direction and speed change: every 20 s

Configuration of simulations and performance

- Square area 500m x 500m
- Different network densities
  - 100 devices / km^2
  - 200 devices / km^2
  - 300 devices / km^2
- Runs on 10 different networks (10 fixed seeds)

- Process the output of the simulator
  - Number of devices reached
  - Number of forwardings
  - Broadcast time

Comparison of the algorithms



Performance of solutions

- Three algorithms
- Thirty independent runs
- One hundred non-dominated solutions per run
- Three network densities

9,000 solutions for every density!

- How to choose the best solutions?
  - Build one single Pareto front from all non-dominated solutions

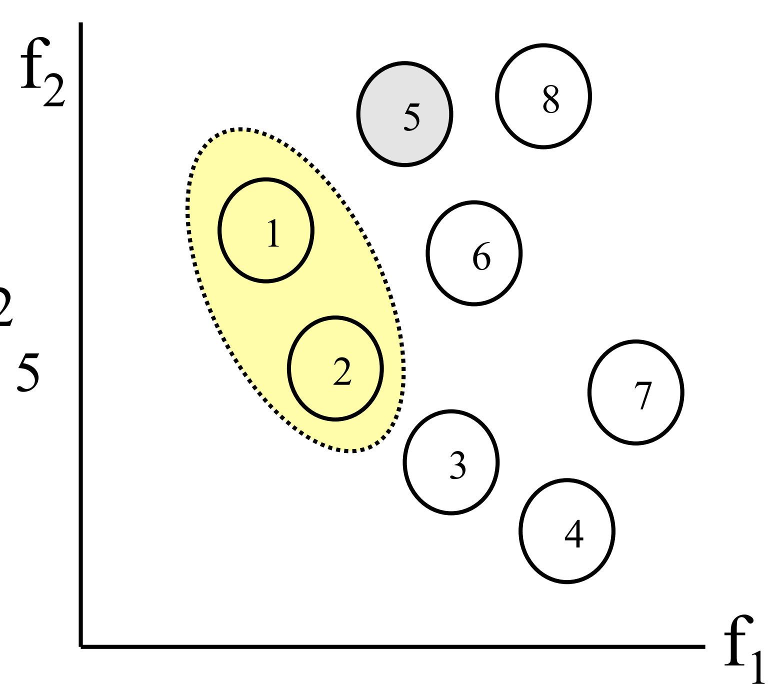
Pareto management techniques

- Strength raw fitness
- Crowding
- Adaptive grid

Strength raw fitness

- Proposed in SPEA2
- Two steps to assign a fitness value to a solution s:
  - Step 1: Strength Fitness
    - Strength fitness of s: number of individuals in the population dominated by s
  - Step 2: Raw Fitness
    - Raw Fitness of s: sum of the strength fitness of the solutions dominating s

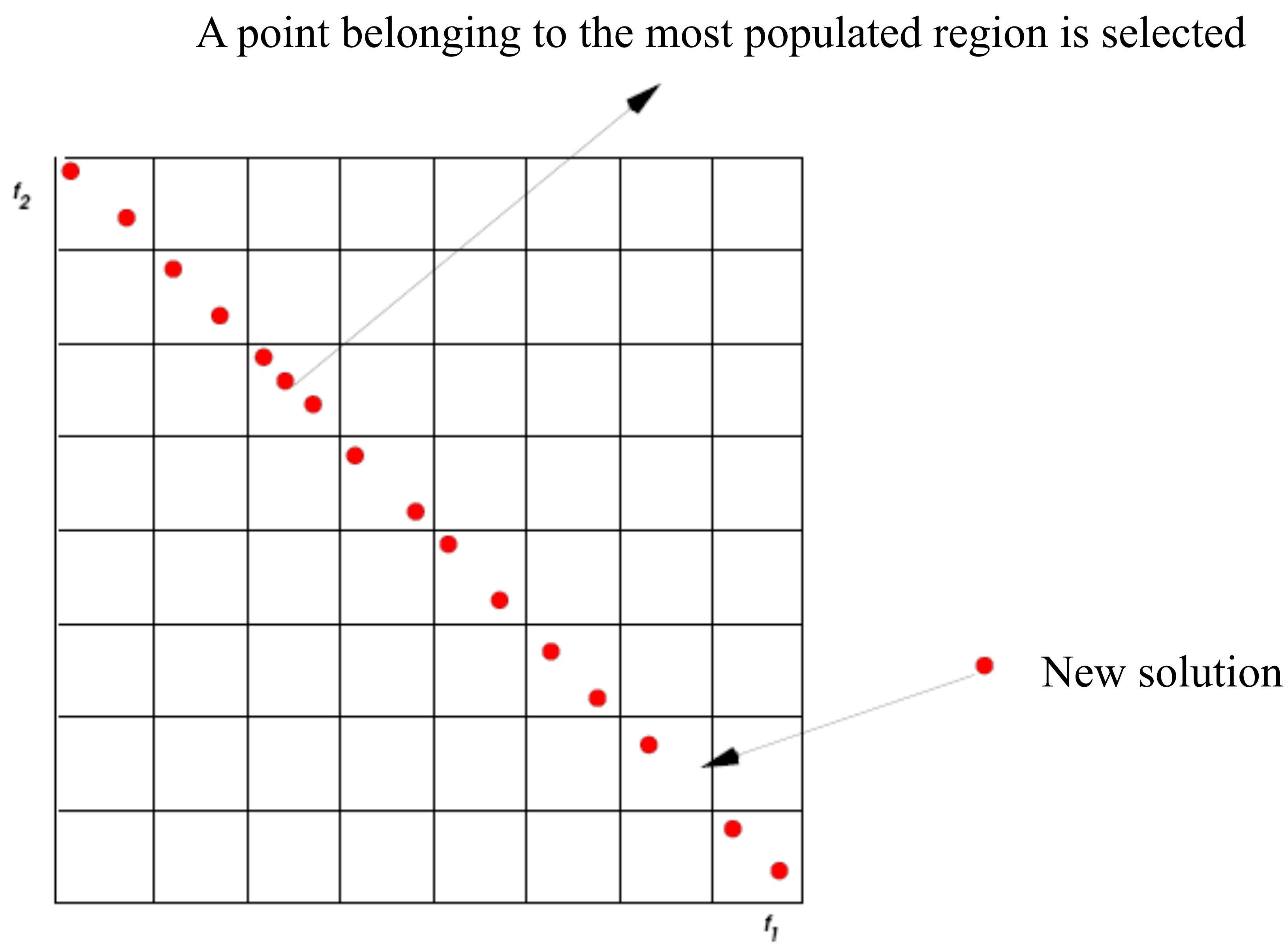
Solutions 1 and 2 dominate solution 5



Solution	Strength fit.	Raw fitness
1	2	0
2	3	0
3	3	0
4	1	0
5	1	5
6	1	8
7	0	4
8	0	10

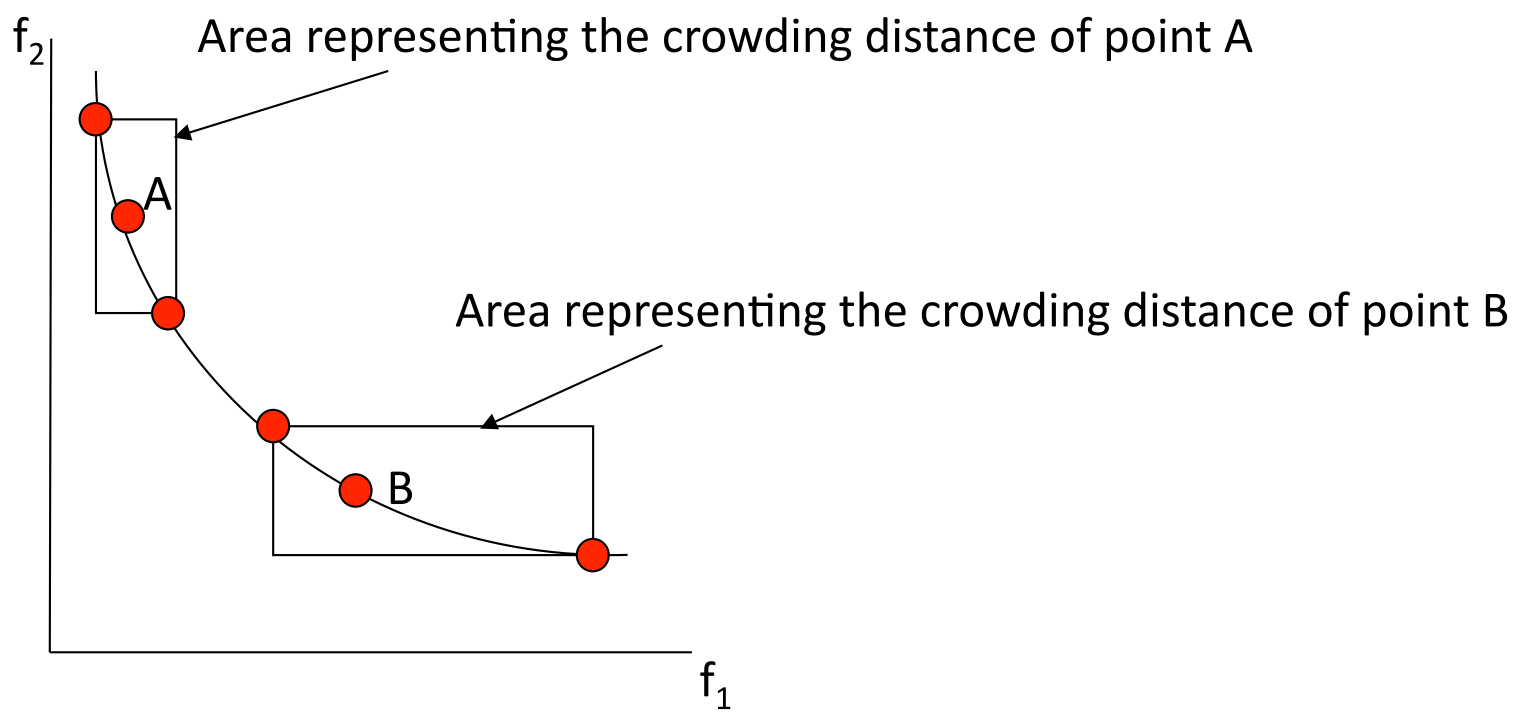
Adaptive grid

- Proposed in PAES
- The objective space is divided up in hypercubes
- Hypercubes are squares, in bi-objective problems

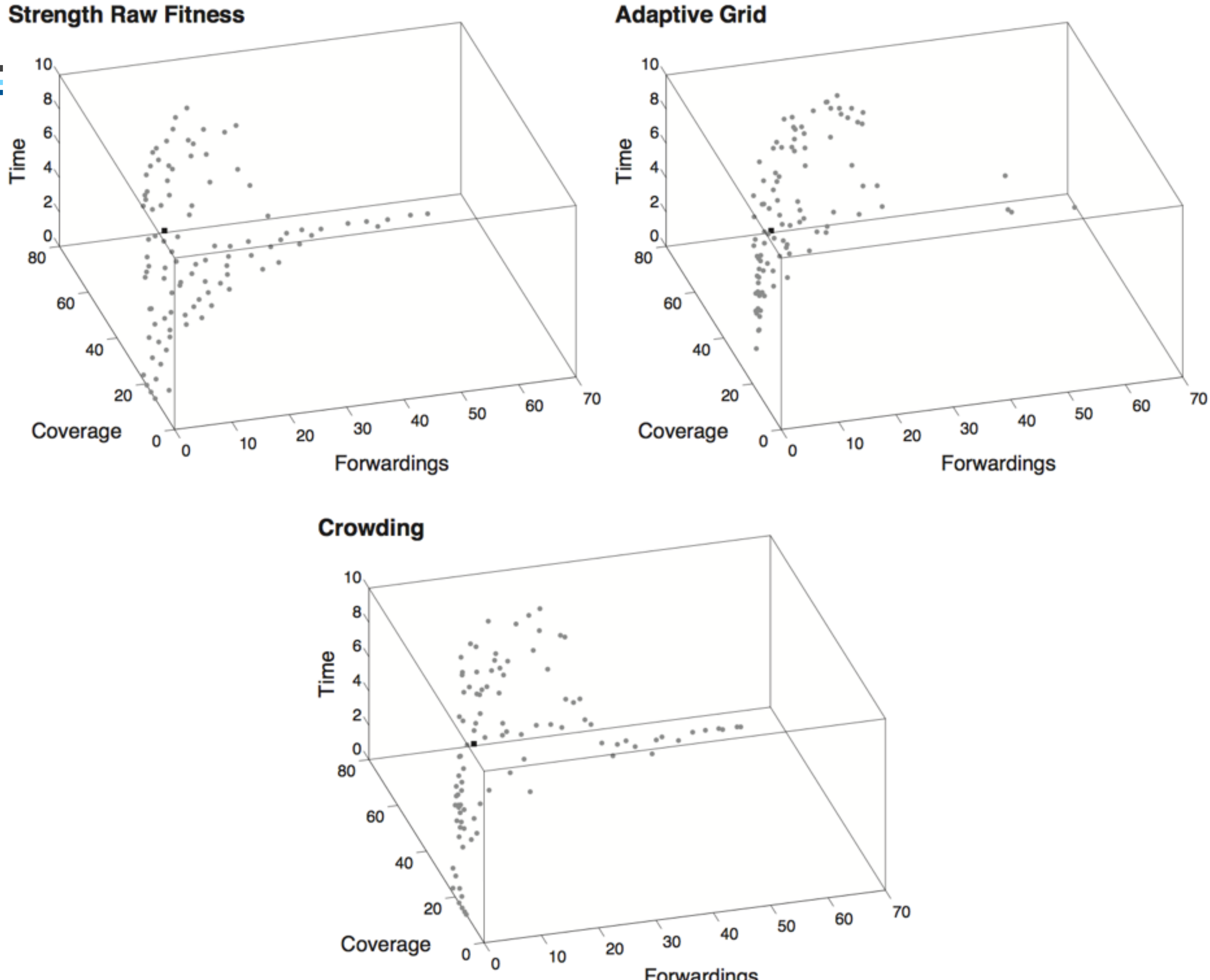


Crowding

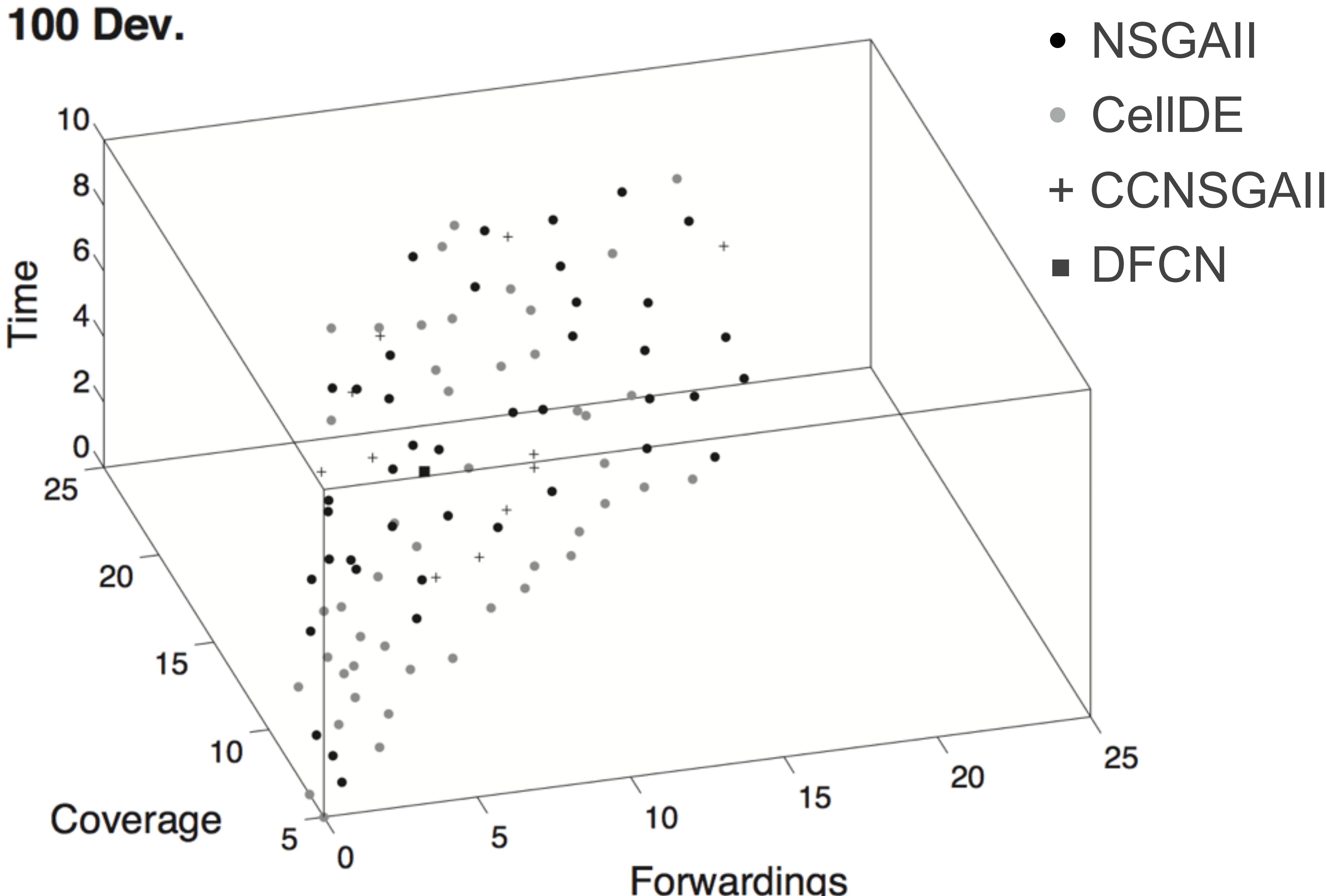
- Proposed in NSGAII
- Estimator of density in the area of the solution



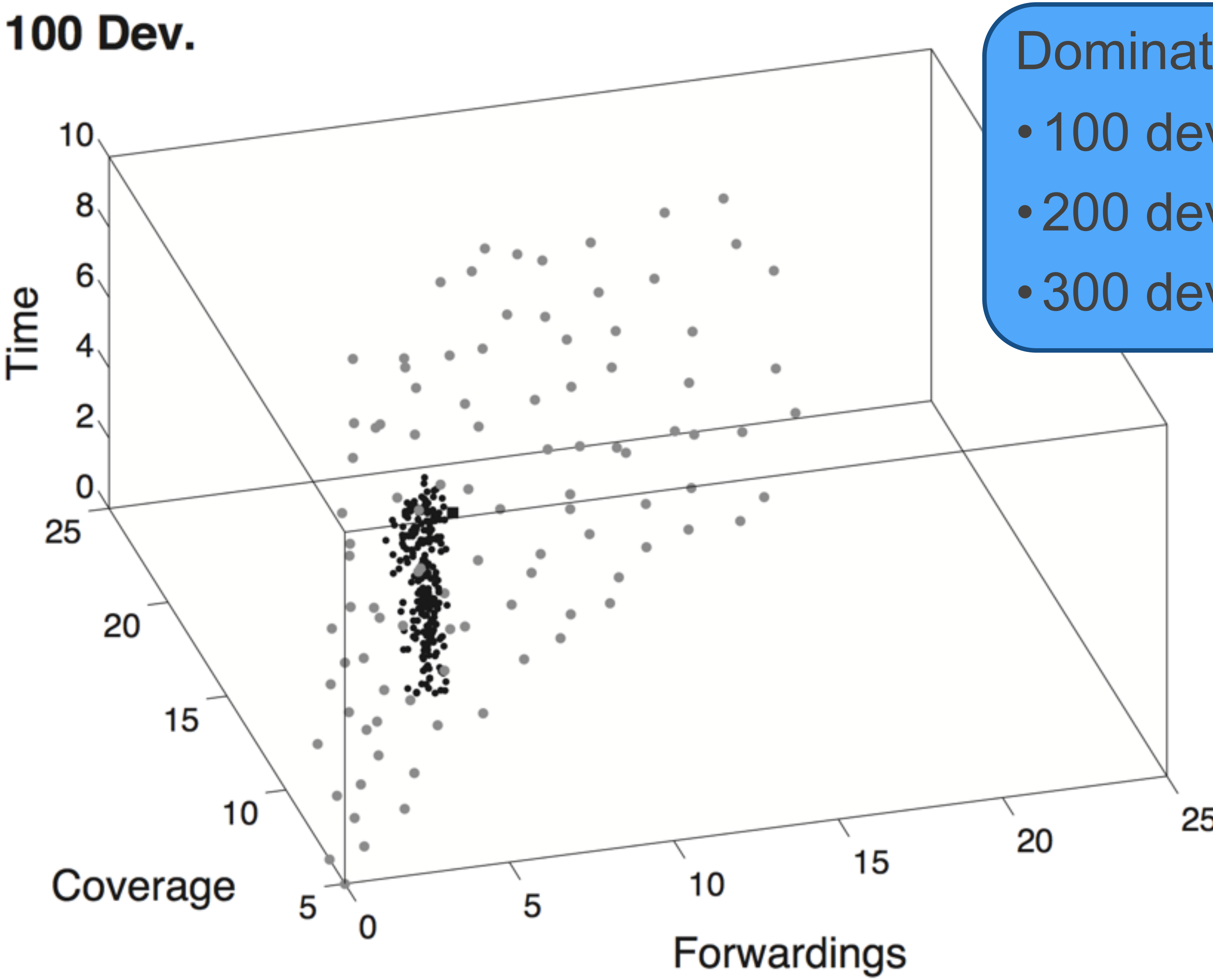
Point B is in a less crowded region than point A


Strength Raw Fitness
Adaptive Grid
Crowding
Time
Coverage
Forwardings
0
2
4
6
8
10
20
40
60
80
30
50
70

Aggregated Pareto front



Solutions dominating DFCN



Dominating DFCN

- 100 dev: 262 Sols
- 200 dev: 233 Sols
- 300 dev: 185 Sols

Solutions dominating DFCN from Pareto front

dev./km^2	Solution	Forwardings (%)	Coverage (%)	Time
100	DFCN	25.60	54.80	5.1212
	Sol1	22.80	58.40	4.8054
	Sol2	22.80	58.00	3.2082
	Sol3	20.40	57.20	1.7476
	Sol4	24.40	63.20	2.3648
	Sol5	25.60	57.60	0.2682
200	DFCN	13.80	49.40	6.0320
	Sol6	12.40	52.00	0.0042
	Sol7	12.60	69.00	6.0276
	Sol8	13.60	55.60	1.3358
	Sol9	11.20	58.20	5.1376
	Sol10	9.00	55.00	5.2180
	Sol11	8.00	52.00	4.1517
	Sol12	12.60	53.40	3.9389
300	DFCN	10.27	50.27	6.2294
	Sol13	7.20	52.53	4.0212
	Sol14	10.27	58.93	5.0449
	Sol15	8.40	51.73	0.6933

Solutions dominating DFCN from Pareto front

dev./km^2	Solution	minGain	RAD	proD	safeDensity	Algorithm
	DFCN	0.4	[0.0, 7.0]	4	12	—
100	Sol1	0.2559279959314381	[0.37838939855767806, 9.120589620531833]	84	3	NSGAII
	Sol2	0.3106136104098672	[0.0, 3.77622653439971]	11	0	CellDE
	Sol3	0.3137985713985266	[0.0, 1.8166803587138693]	88	7	CellDE
	Sol4	0.29569050500280863	[0.28417307413411563, 1.879038038884]	62	5	NSGAII
	Sol5	0.30190793008451416	[0.00216725071262841, 0.17757174835303718]	82	9	NSGAII
200	Sol6	0.5244789246681958	[0.0, 0.0]	68	12	CellDE
	Sol7	0.23262266938338613	[10.0, 10.0]	67	12	CellDE
	Sol8	0.45138885098745435	[0.005853573465718576, 1.4167981490052637]	73	5	NSGAII
	Sol9	0.35188367053794545	[10.0, 10.0]	18	24	CellDE
	Sol10	0.2935923314536946	[10.0, 10.0]	23	10	CellDE
	Sol11	0.2874099031627282	[10.0, 10.0]	43	3	CellDE
	Sol12	0.48537600157429095	[0.421648485009434, 2.8308110208460655]	37	11	NSGAII
300	Sol13	0.46147023817982497	[0.010781467931704425, 4.261884410159036]	74	2	NSGAII
	Sol14	0.410847243200888	[2.5014487499316895, 4.601211573914485]	38	13	CellDE
	Sol15	0.4977711892032174	[0.06103145898364451, 0.5542608402159395]	79	10	NSGAII

Conclusions

- Optimization of a protocol for MANETs
  - Broadcasting protocol
    - Importance of broadcasting in MANETS
    - Difficulty of broadcasting in MANETS
  - Otimization of DFCN:
    - Coverage
    - Number of forwardings
    - Time
- Performance comparison (NSGAII, CellDE, CCNSGAII)
- Selection of a representative set of Pareto solutions
- Many configurations outperforming DFCN