

HIPERCOM Reading Group, Session III

Virtual Ring Routing Network routing inspired by DHTs

M. Caesar, M. Castro, E. B. Nightingale, G. O'Shea, A. Rowstron

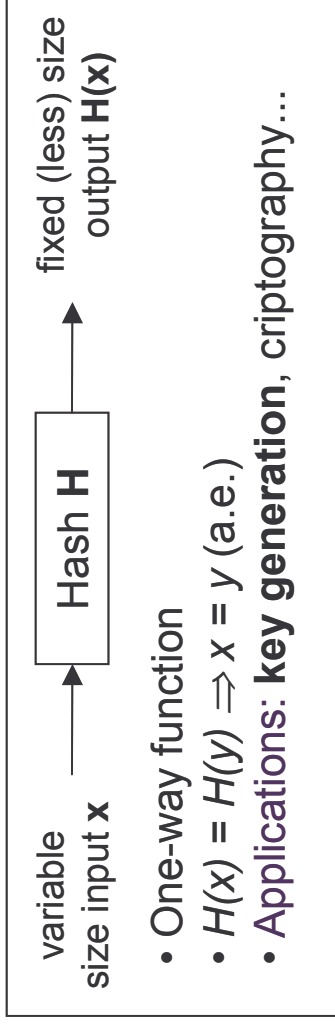
Juan Antonio Cordero Fuertes

Summary

- Previous ideas
- What is Virtual Ring Routing
- How does VRR work
- Experiments

Previous ideas

❖ Hash function



❖ Hash table

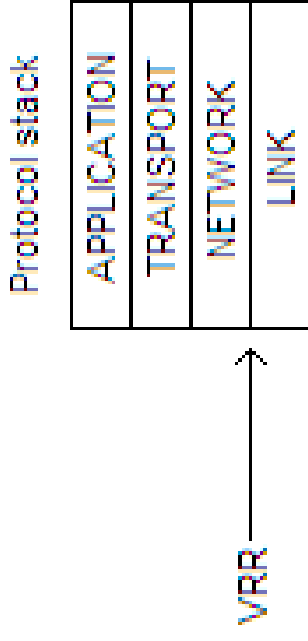


❖ Distributed Hash Table (DHT)

- Partition** of the **keyspace** among many nodes forming the **overlay network**.
- Examples: Chord, CAN, Pastry...
 - Decentralisation
 - Scalability
 - High dynamism

What is Virtual Ring Routing (VRR)

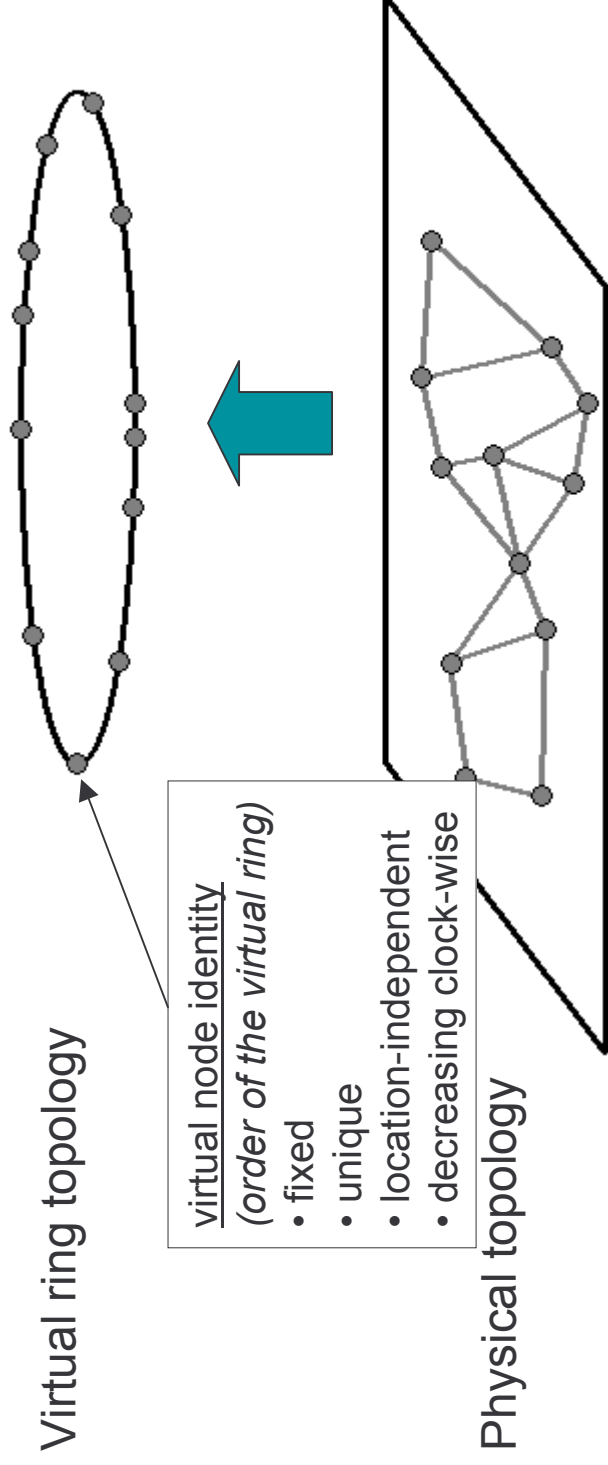
General issues

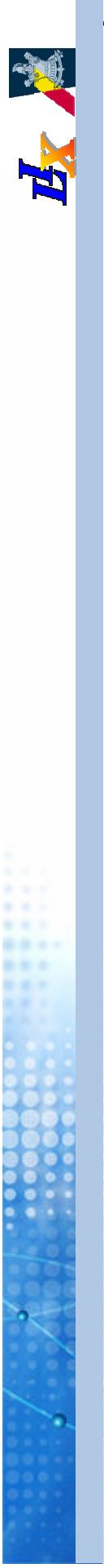


- ❖ Network routing protocol
- ❖ Provides:
 - Point-to-point routing
 - DHT functionality
- ❖ Virtual ring topology
- ❖ Location-independent identifiers

What is Virtual Ring Routing (VRR)

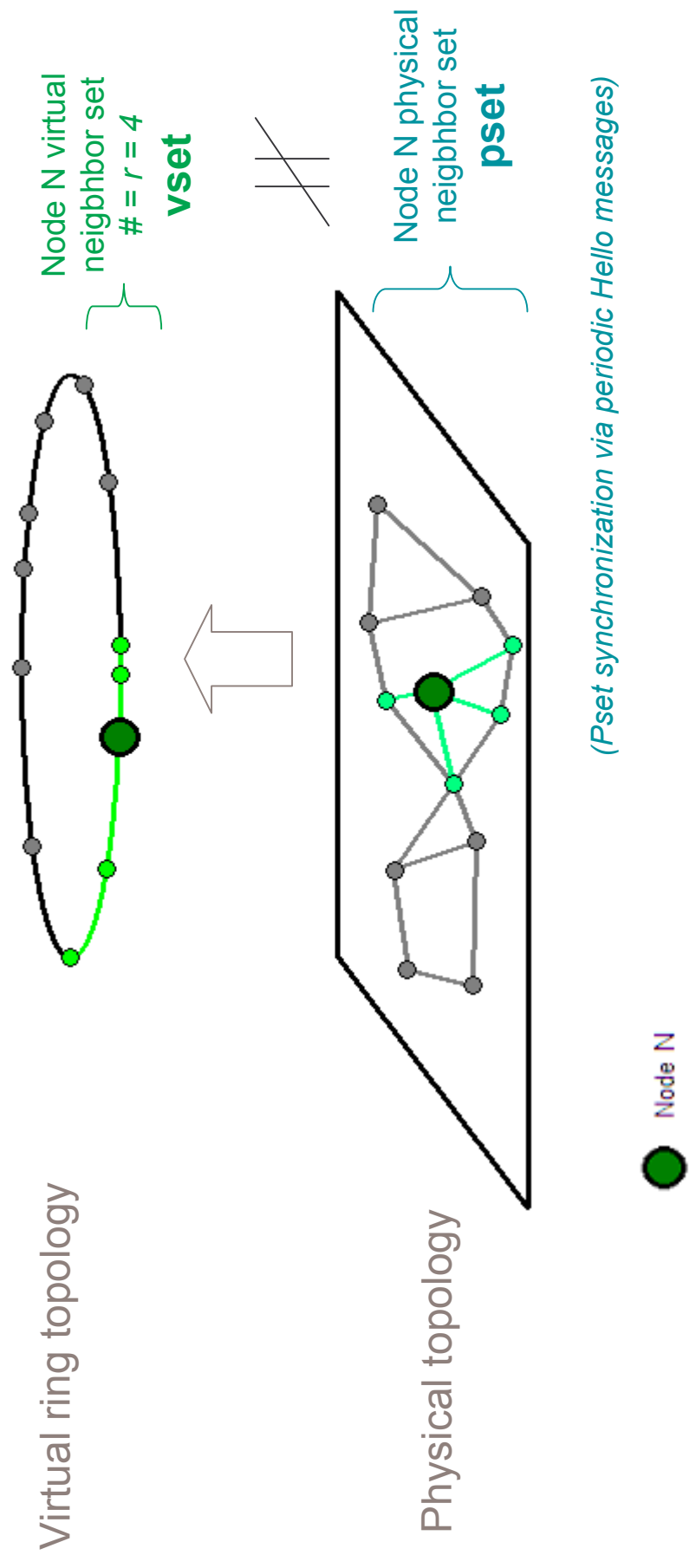
Virtual ring structure





What is Virtual Ring Routing (VRR)

vset and pset of a node



What is Virtual Ring Routing (VRR)

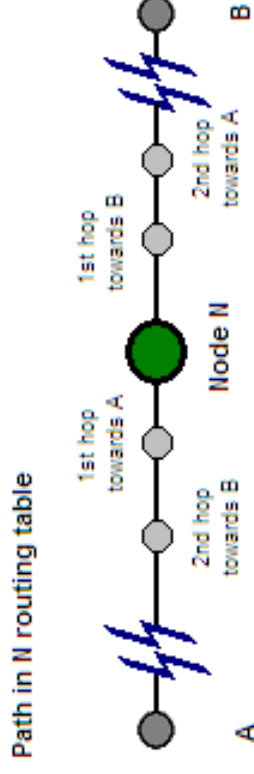
Routing table

- ❖ Routing is performed in the **virtual ring topology**
- ❖ Each node maintains routes to its v-neighbors (**vset-paths**)
- ❖ Routes are **bidirectional**
- ❖ Routing table structure

- Entries:
- routes to each of the node's v-neighbors
 - routes involving the node
 - 1-hop p-neighbors

endpoint _A	endpoint _B	nexthop _A	nexthop _B	nextnexthop _A	nextnexthop _B	path-id

- ❖ Example of path contained in the routing table



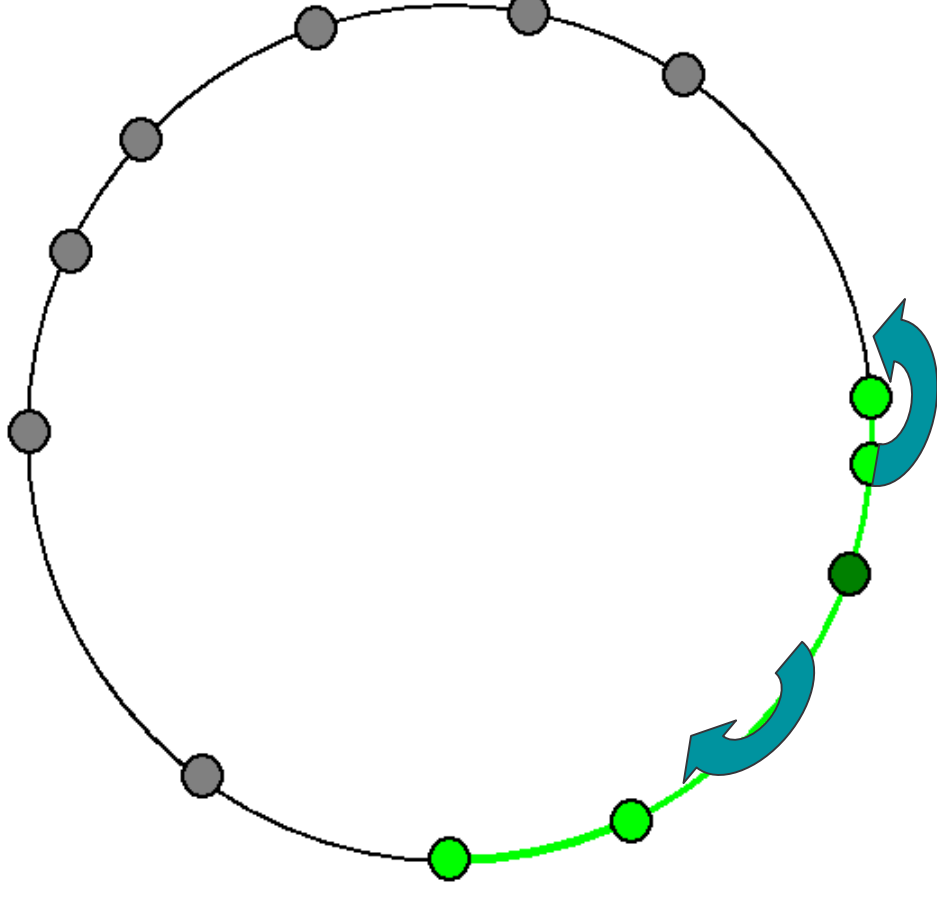
How does VRR work

Summary

- Forwarding
- Joining procedure
- p-neighborhood management
 - States, failure detection and failure repair
- Ring partition management

How does VRR work

Forwarding



- ❖ The node delivers the received packets to the neighbor selected as nexthop to the id closest to the packet destination.
- Returns NULL if the receiving node is the packet destination.

How does VRR work

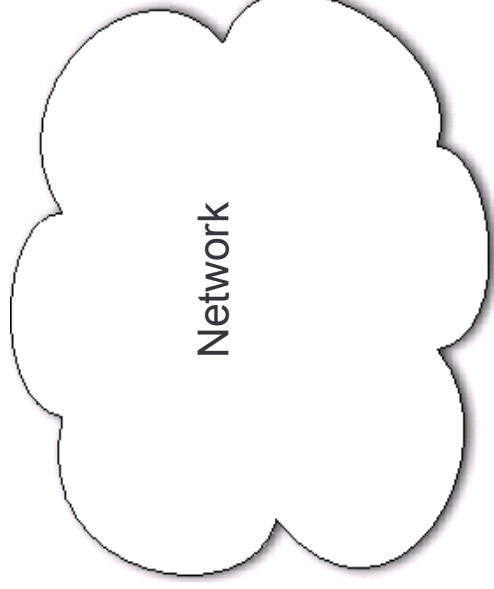
Joining procedure

New node N

Init pset(N), vset(N)



Joining node
N



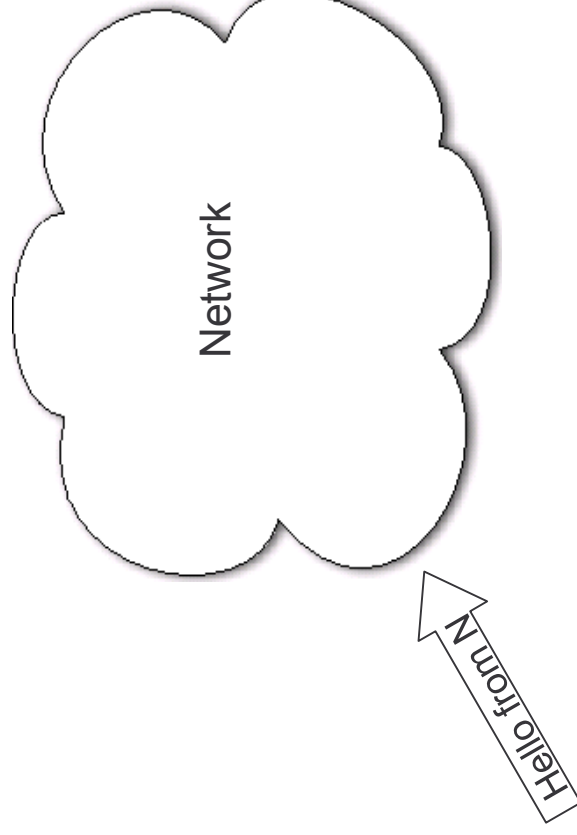
How does VRR work

Joining procedure

New node N

Init `pset(N)`, `vset(N)`

Send broadcast *hello*



Joining node
N

How does VRR work

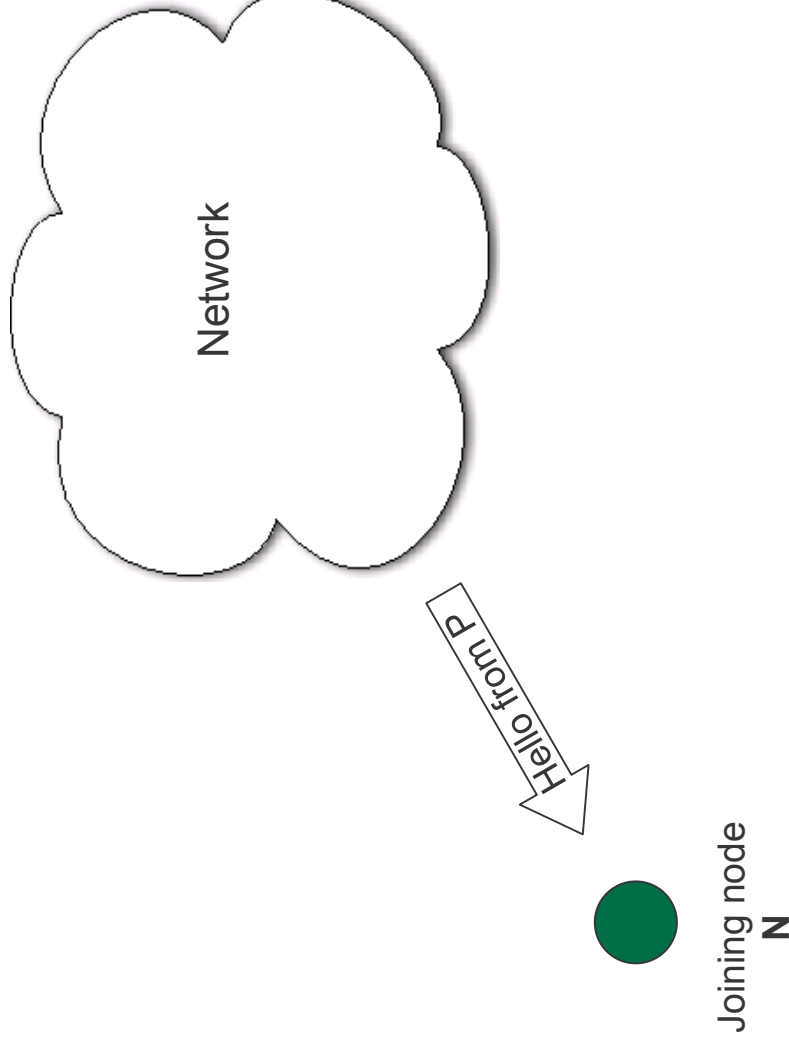
Joining procedure

New node **N**

Init pset(N), vset(N)

Send broadcast *hello*

Listen *hellos* from
pneighbors



How does VRR work

Joining procedure

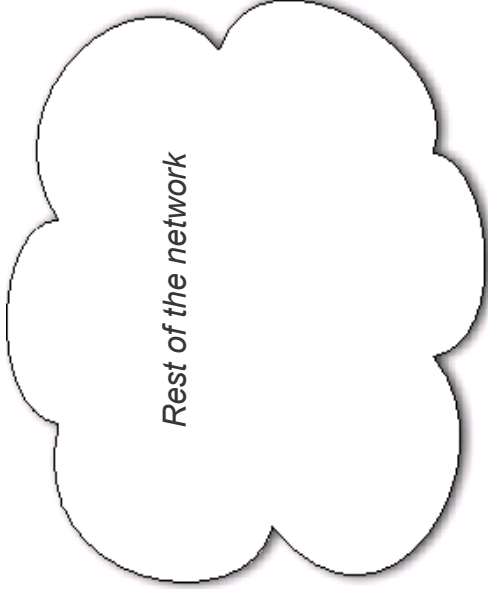
New node **N**

Init $pset(N)$, $vset(N)$

Send broadcast *hello*

Listen *hellos* from neighbors

Select proxy **P** among the neighbors



Proxy $\in pset(N)$
P



Joining node
N

How does VRR work

Joining procedure

New node **N**

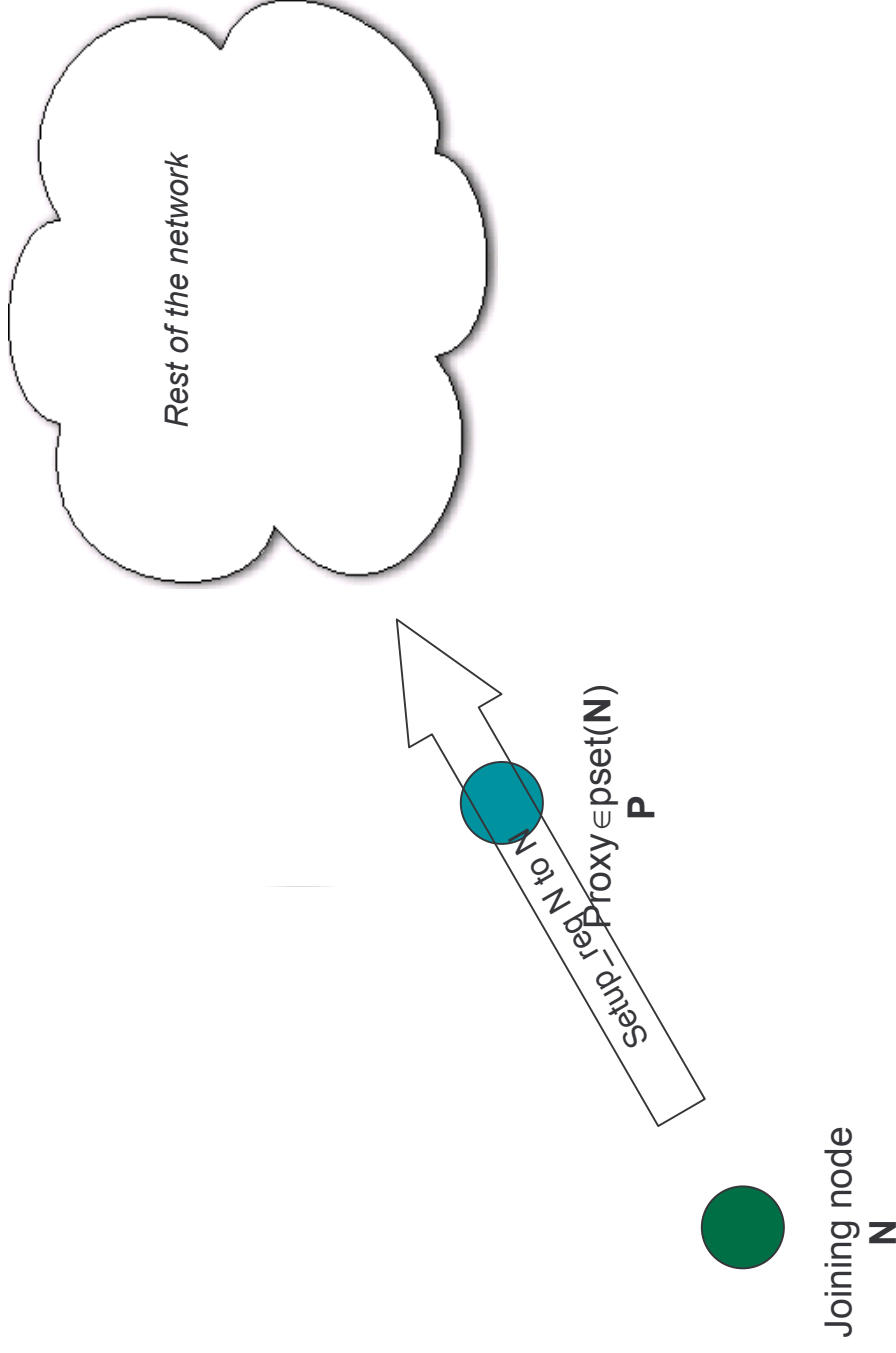
Init $pset(N)$, $vset(N)$

Send broadcast *hello*

Listen *hellos* from neighbors

Select proxy **P** among the neighbors

Send *setup_req* to the neighbor **y** with closest id



How does VRR work

Joining procedure

New node **N**

Init $pset(N)$, $vset(N)$

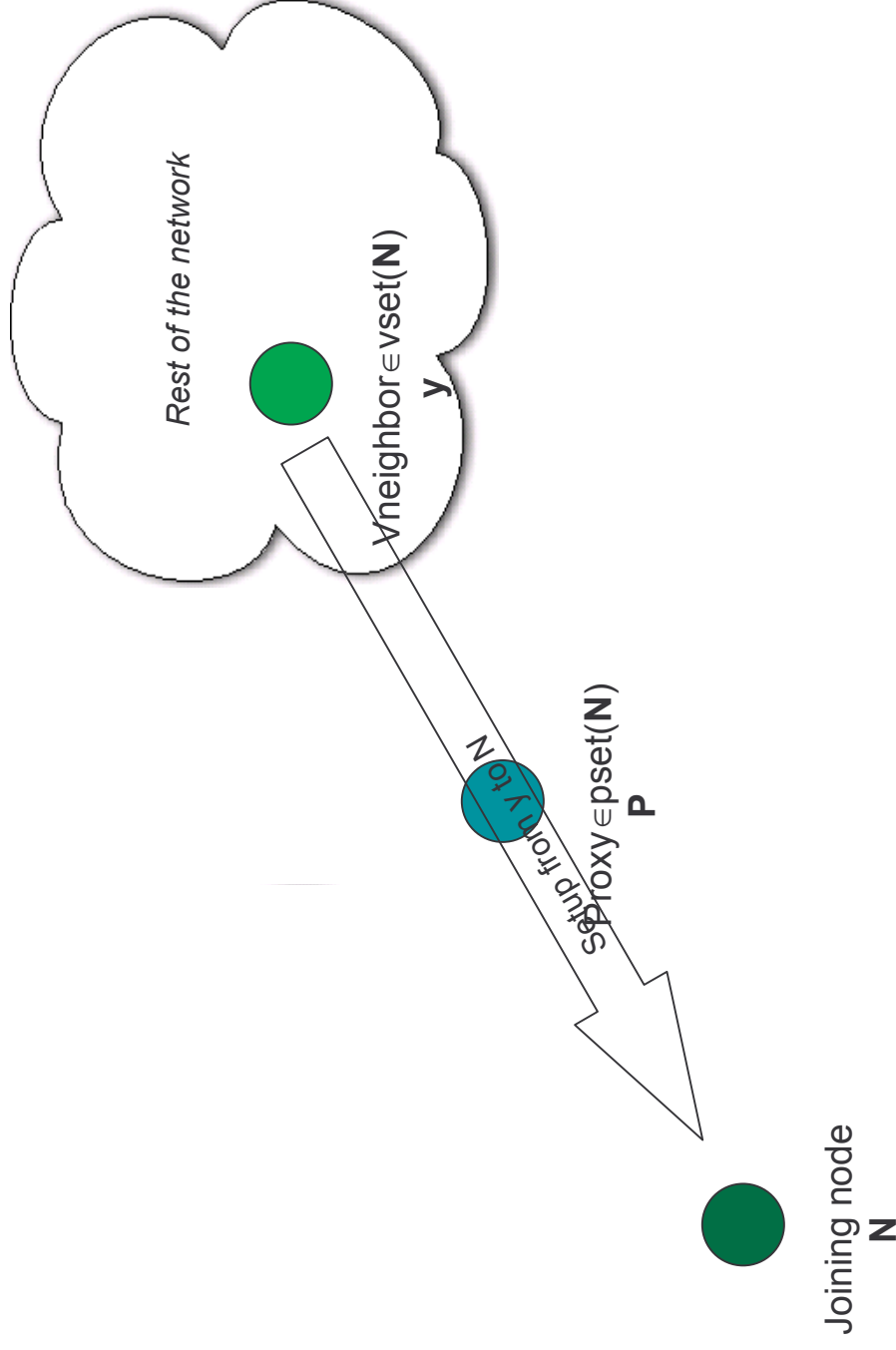
Send broadcast *hello*

Listen *hellos* from neighbors

Select proxy **P** among the neighbors

Send *setup_req* to the neighbor **y** with closest id

Listen to *setup* msg from **y**, containing the other $vset(N)$ members



How does VRR work

Joining procedure

New node **N**

Init $pset(N)$, $vset(N)$

Send broadcast *hello*

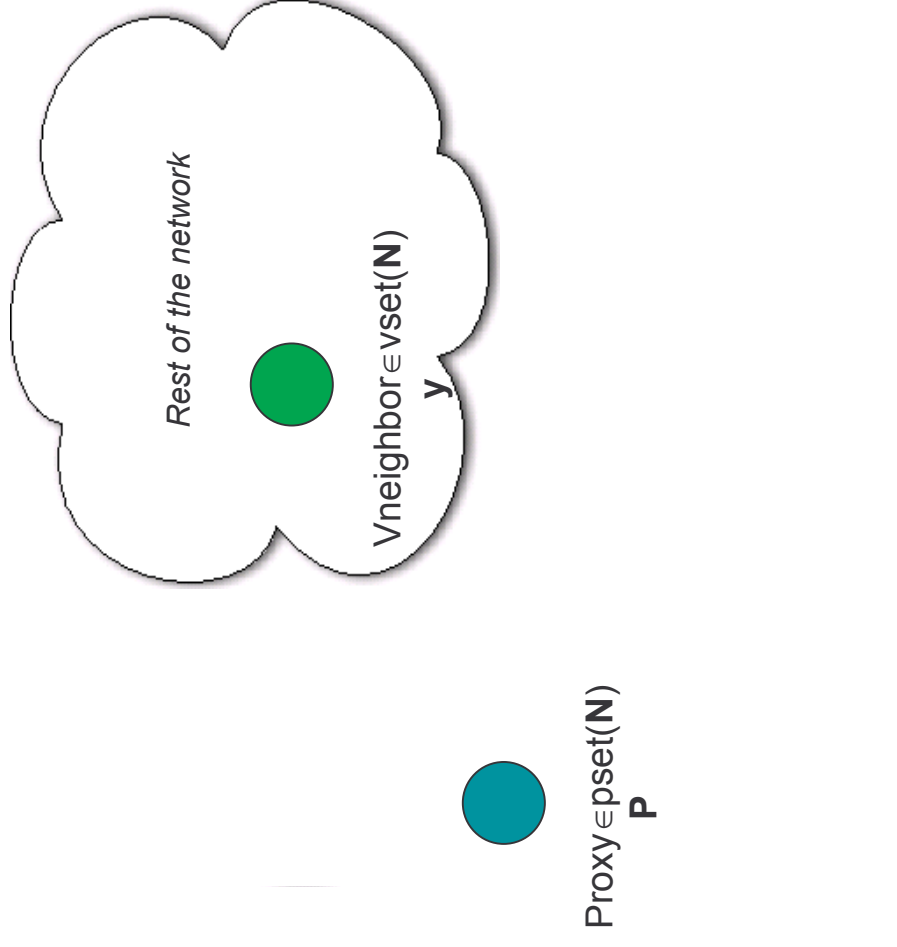
Listen *hellos* from neighbors

Select proxy **P** among the neighbors

Send *setup_req* to the neighbor **y** with closest id

Listen to *setup* msg from **y**, containing the other $vset(N)$ members

Send *setup_req* to neighbors...



How does VRRP work

Joining procedure

New node **N**

Init $pset(N)$, $vset(N)$

Send broadcast *hello*

Listen *hellos* from neighbors

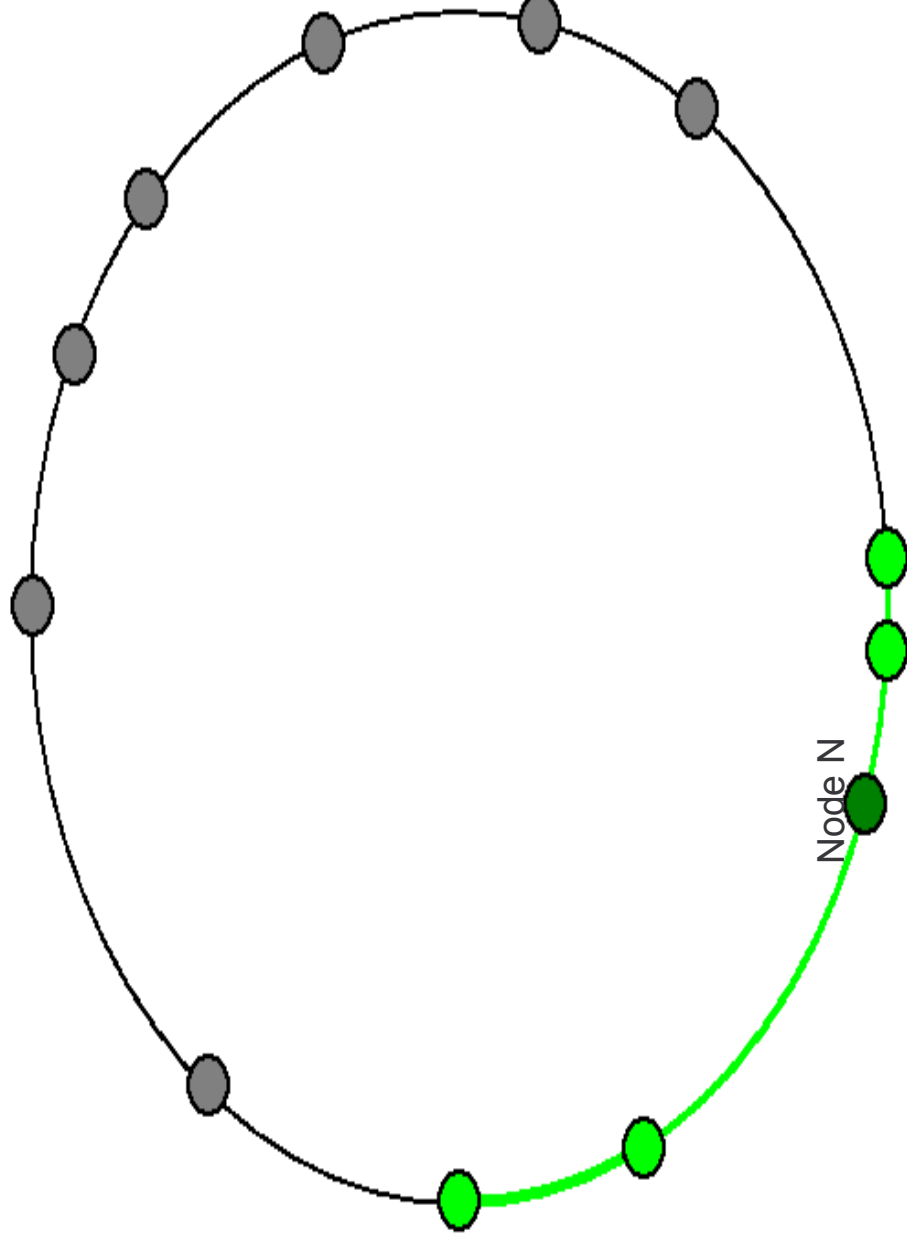
Select proxy **P** among the neighbors

Send *setup_req* to the neighbor **y** with closest id

Listen to *setup* msg from **y**, containing the other $vset(N)$ members

Send *setup_req* to neighbors...

N becomes active



➤ Additional remarks

How does VRR work

p-neighborhood handling

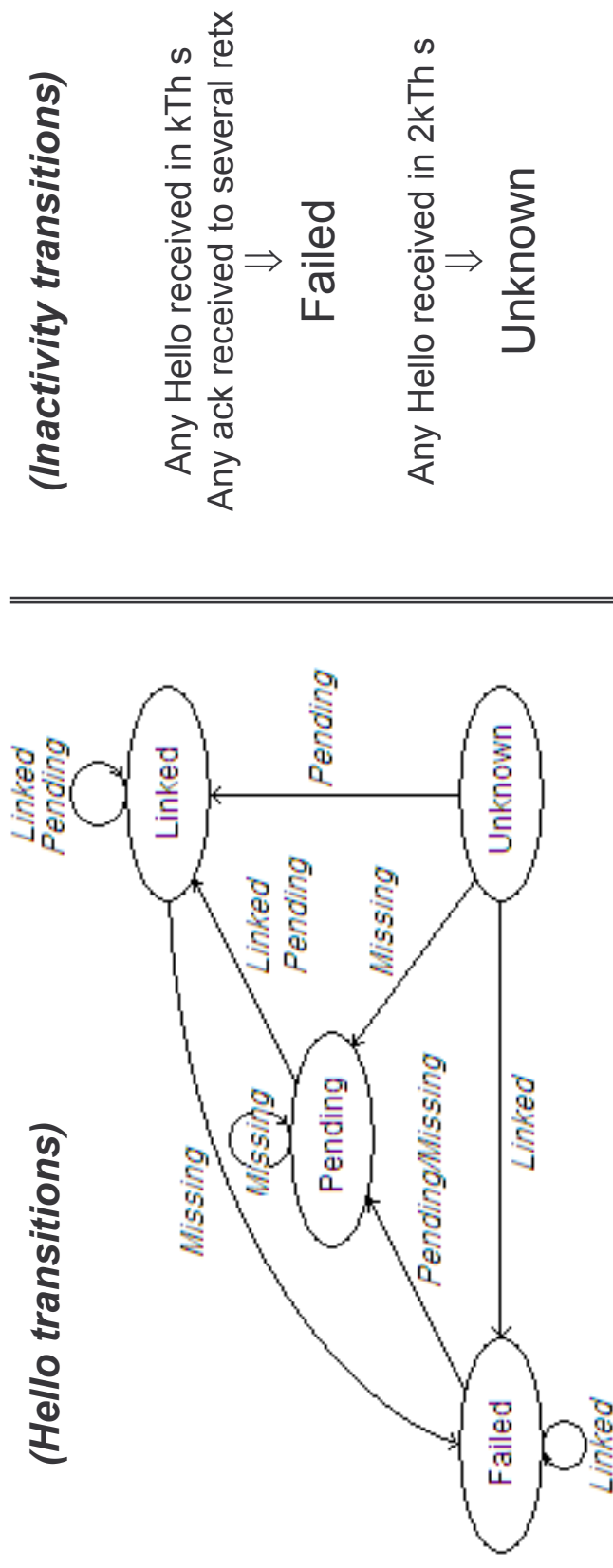
❖ Four possible neighbor states (for physical neighbors):

- **Linked** (active or non-active): two-way communication
- **Pending**: one-way communication
- **Failed**: marked as faulty
- **Unknown**

How does VRR work

p-failure detection

- ❖ p-neighbor state transitions due to *Hello* messages and inactivity timers.



Symmetric failure detection (SFD)

How does VRR work

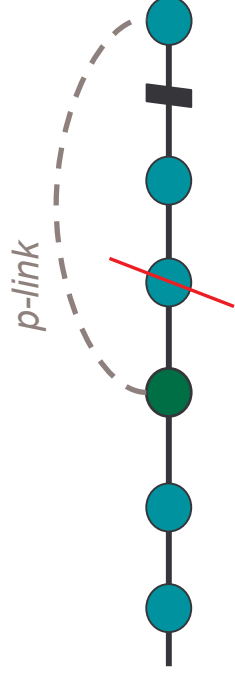
Failure repair

- ❖ Failure repair
 - Failure advertisement (*teardown*)
 - End-to-end repair

❖ Local repair extension

Requires the extended routing table format

endpoint _A	endpoint _B	nexthop _A	nexthop _B	nextnexthop _A	nextnexthop _B	path-id



- Constant cost (independent from route size)
- If unsuccessful, the teardown is delayed.

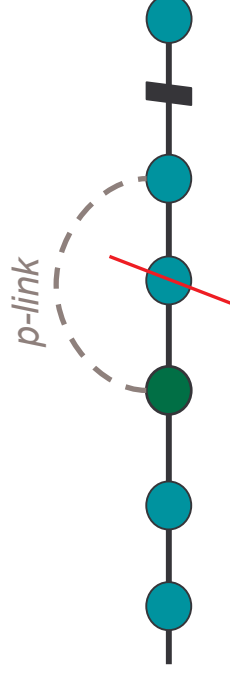
How does VRR work

Failure repair

- ❖ Failure repair
 - Failure advertisement (*teardown*)
 - End-to-end repair
- ❖ Local repair extension

Requires the extended routing table format

endpoint _A	endpoint _B	nexthop _A	nexthop _B	nextnexthop _A	nextnexthop _B	path-id



- Constant cost (independent from route size)
- If unsuccessful, the teardown is delayed.

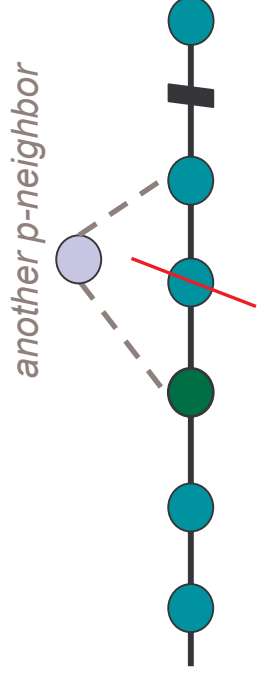
How does VRR work

Failure repair

- ❖ Failure repair
 - Failure advertisement (*teardown*)
 - End-to-end repair
- ❖ Local repair extension

Requires the extended routing table format

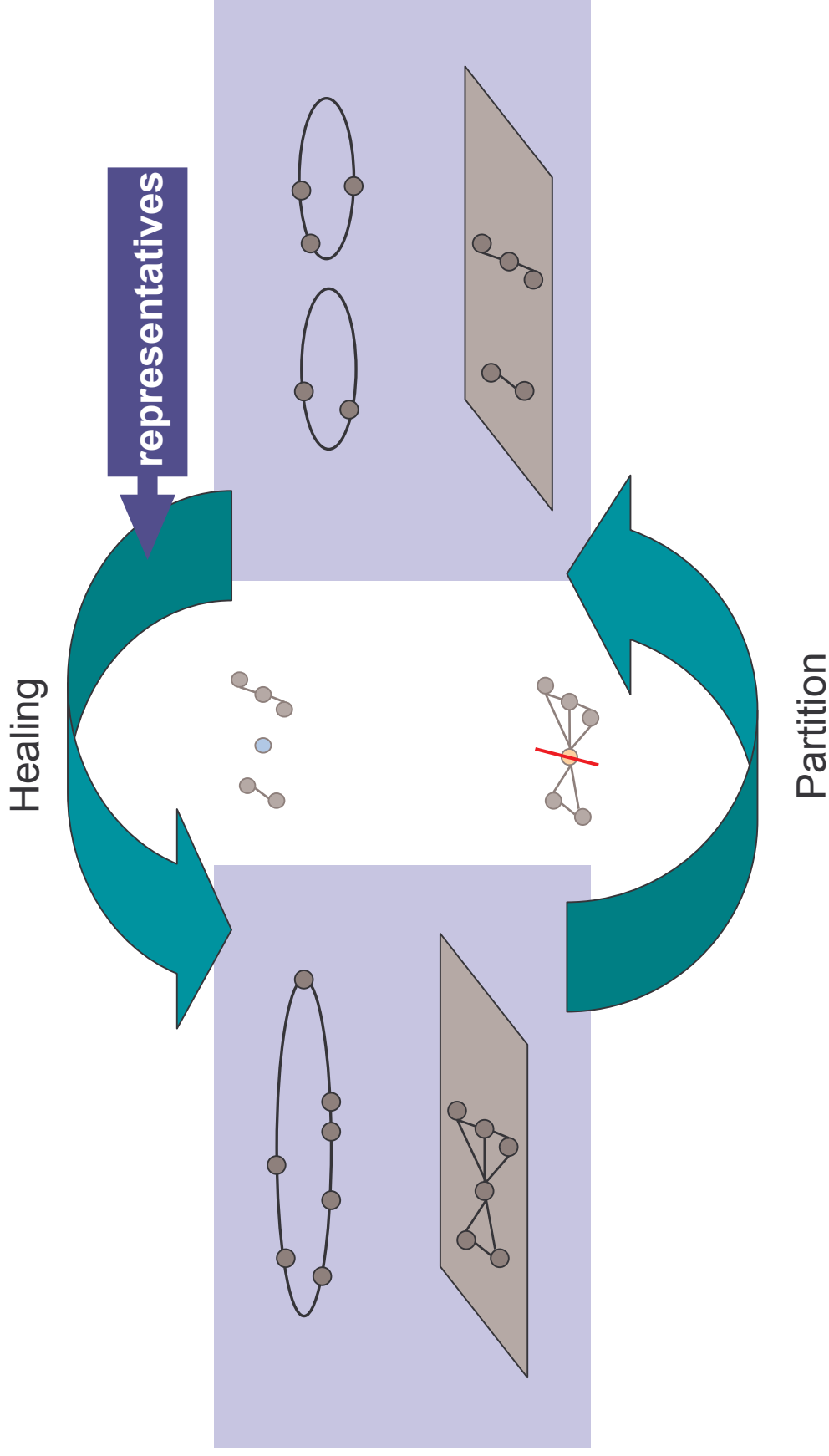
endpoint _A	endpoint _B	nexthop _A	nexthop _B	nextnexthop _A	nextnexthop _B	path-id



- Constant cost (independent from route size)
- If unsuccessful, the teardown is delayed.

How does VRR work

Network partition and healing (1)



How does VRR work

Network partition and healing (2)

❖ Representative of a ring

The node with a virtual id closest to 0

- Each ring has *one and only one* representative.
- Self-conscious
 - x representative $\Leftrightarrow id_x > id_y \quad \forall y \in vset(x)$
- Every node stores a route to the representative, which is periodically updated (via *Hello* messages).
- A node x realizing the representative $\in vset(x)$ sends a *setup_req* message \Rightarrow **vset stabilization mechanism**

Experiments

Summary

- Simulations
 - *Simulator:* NS-2.27
 - *Protocols:* VRR (with local repair), AODV, DSR and DSDV
 - *Base configuration:* 50 nodes (mobile), 1500m x 300m, CBR traffic
 - *Variable parameters:* Mobility rate, # nodes (density ct.), traffic load, flows lifetime
 - *Evaluated issues:* Delivery rate, end-to-end delay, message overhead
- Sensor network testbed
 - 67 nodes (static)
 - *Protocols:* VRR (without local repair), BVR
 - *Evaluated issues:* Delivery rate (with/without failures), routing overhead
- 802.11 Windows testbed
 - 30 nodes (static)
 - *Protocols:* VRR, MR-LSQR
 - *Evaluated issue:* Throughput

Experiments

Simulations: increasing # CBR flows

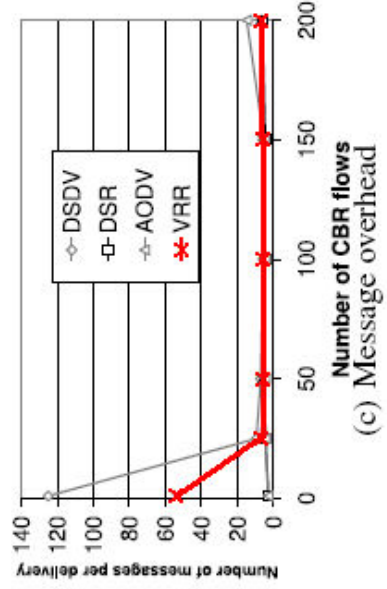
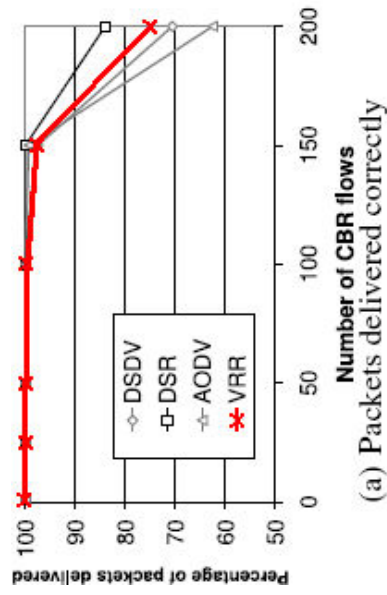


Figure 6: Performance with increasing number of CBR flows in the static scenario.

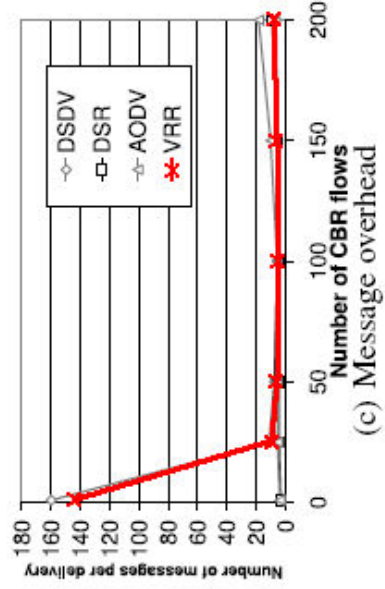
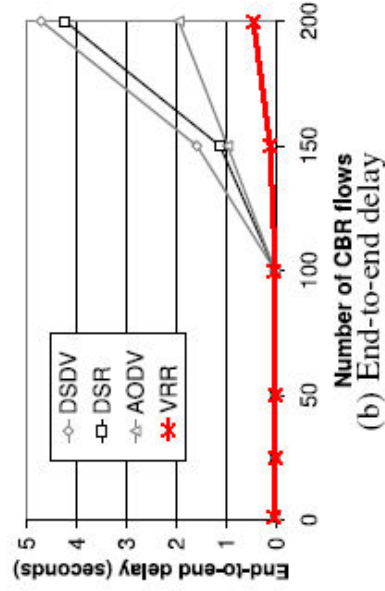
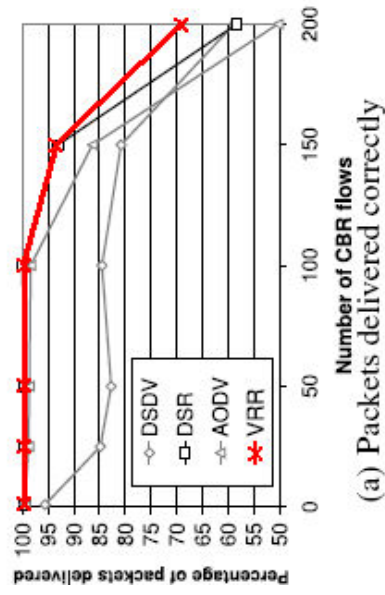


Figure 7: Performance with increasing number of CBR flows in the 20m/s mobility scenario.

Experiments

Simulations: increasing network size

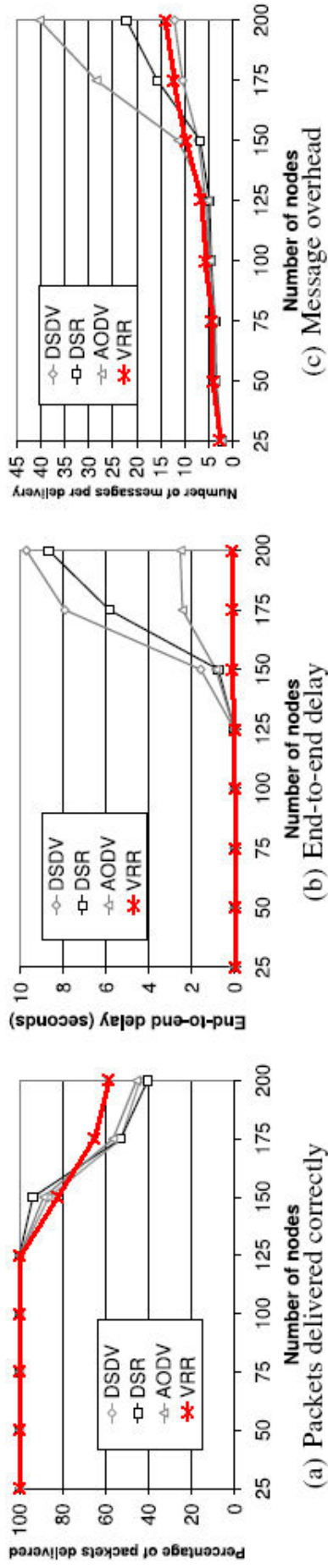


Figure 8: Performance with increasing network size in the static scenario.

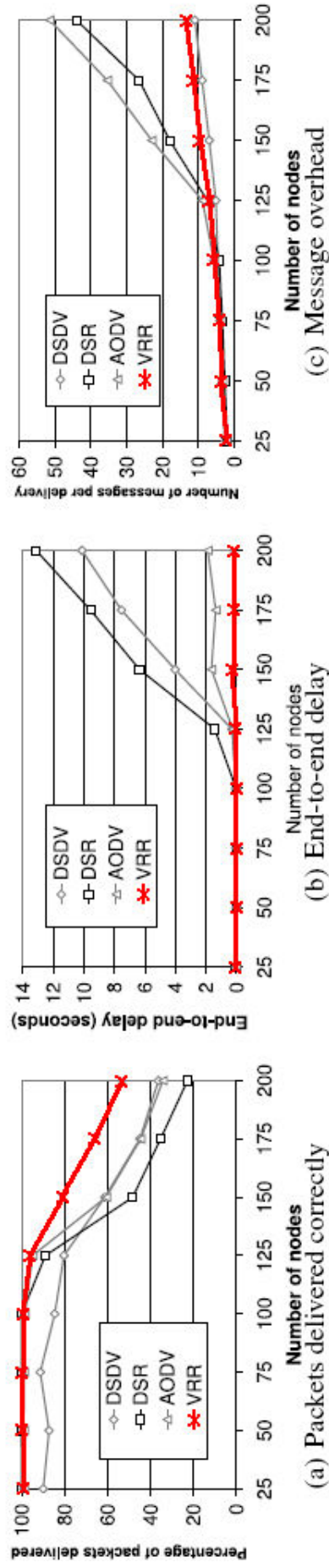


Figure 9: Performance with increasing network size in the 20 m/s mobility scenario.

Experiments

Simulations: short-lived flows

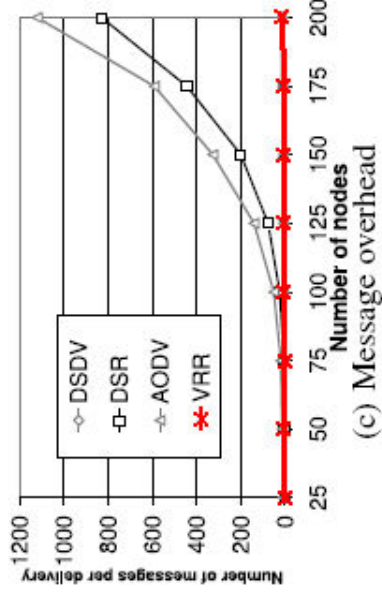
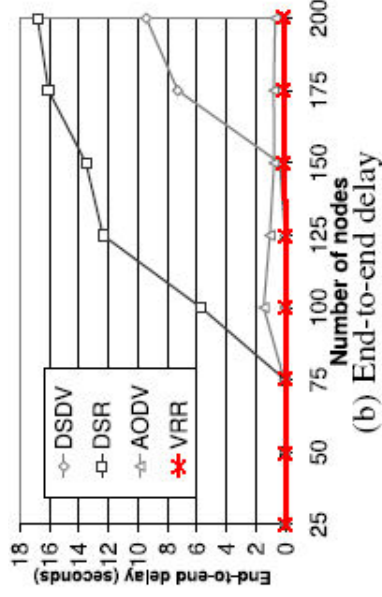
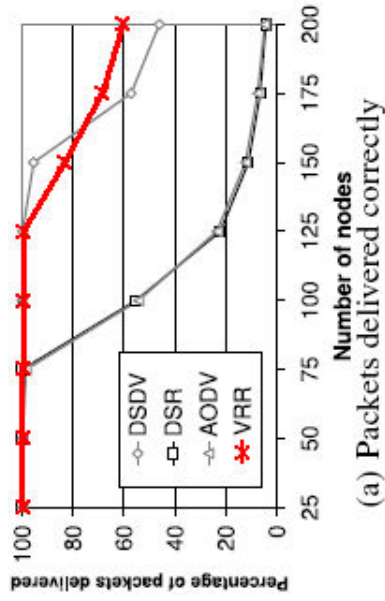


Figure 10: Performance with short-lived CBR flows in the static scenario.

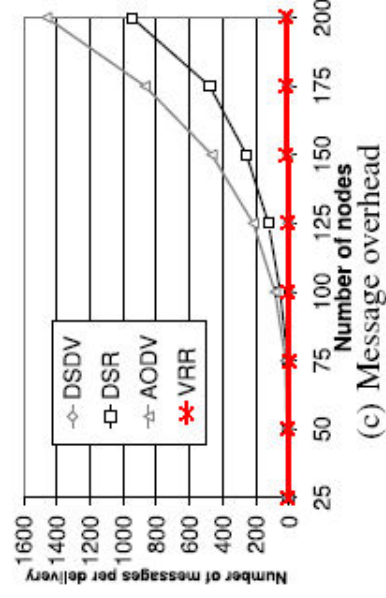
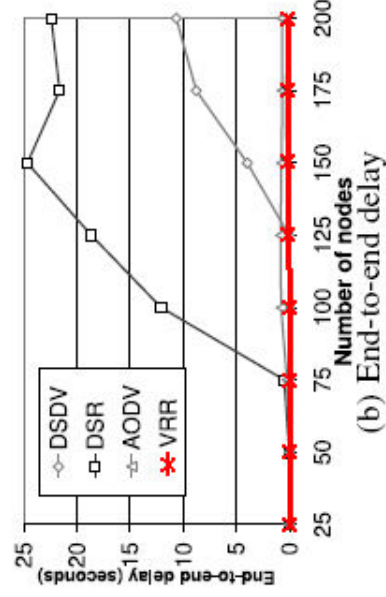
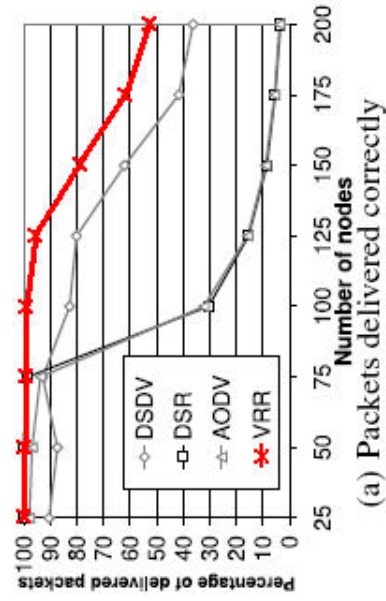
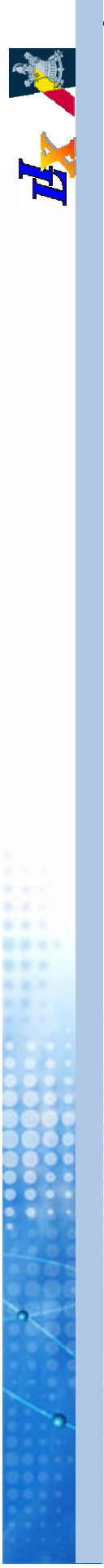


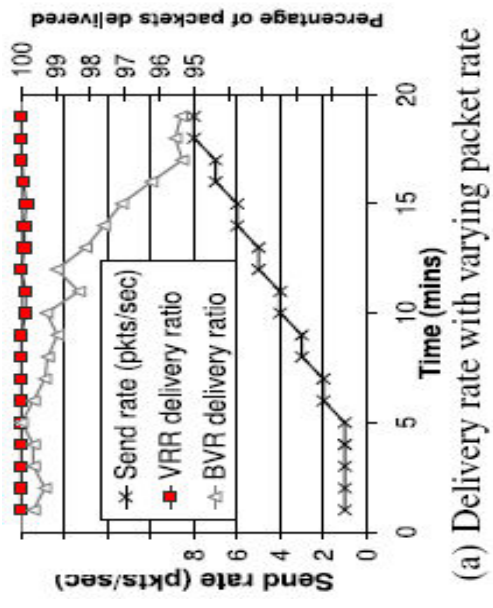
Figure 11: Performance with short-lived CBR flows in the 20m/s mobility scenario.



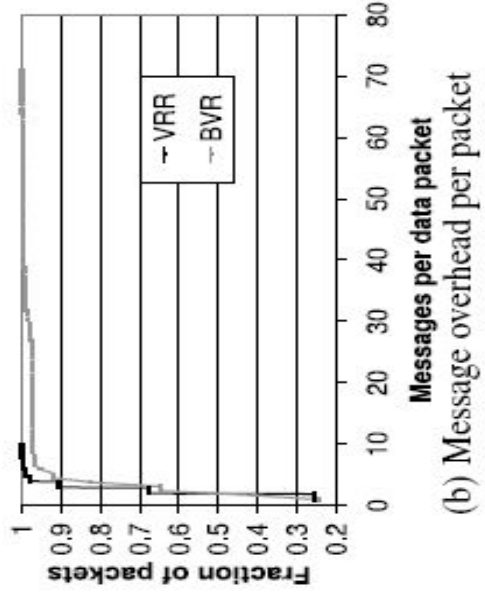
Experiments

Sensor network testbed

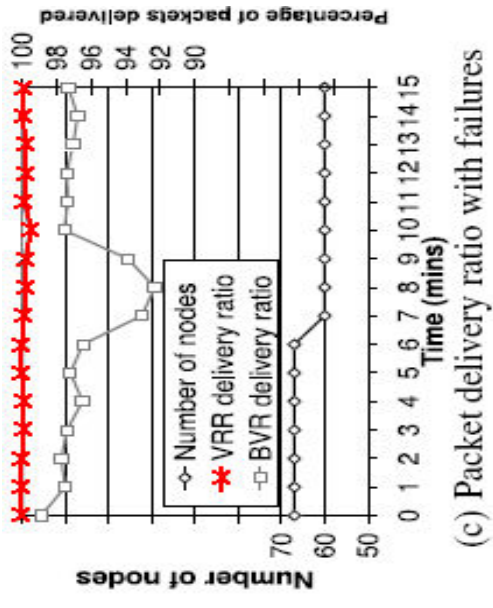
VRR vs BVR, representative of coordinate-based routing



(a) Delivery rate with varying packet rate



(b) Message overhead per packet



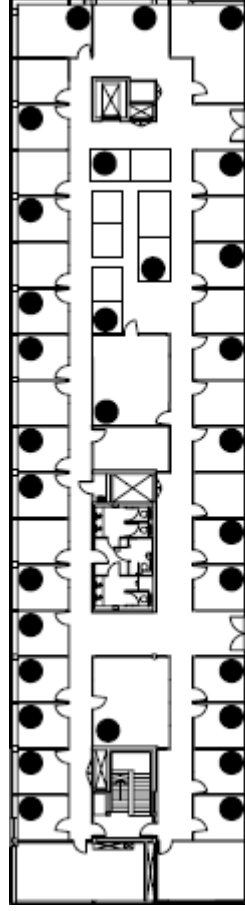
(c) Packet delivery ratio with failures

Figure 13: Sensor network testbed results.

Experiments

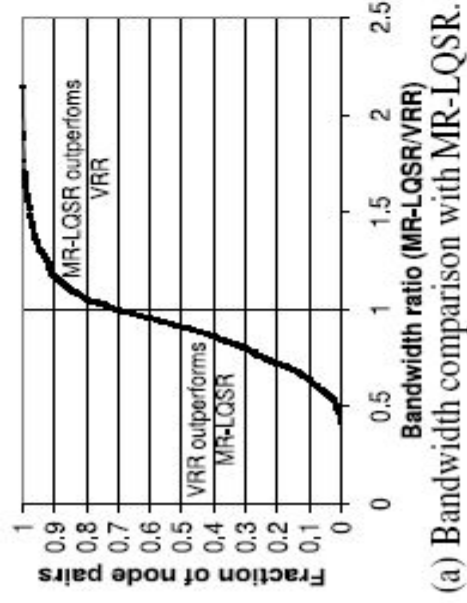
802.11 Windows network

VRR vs MR-LQSR, representative of wireless mesh routing

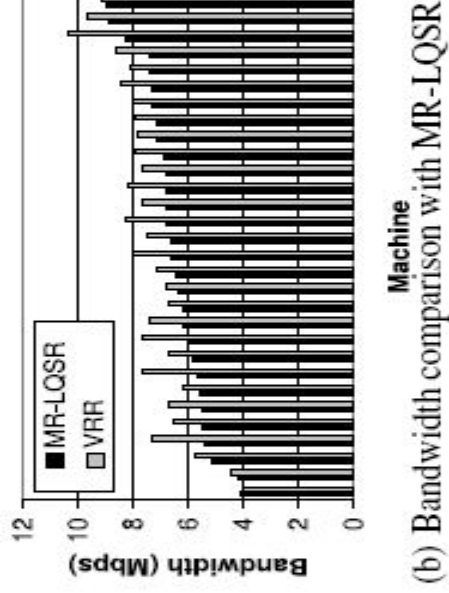


- TCP 8 MB transferred between all pairs
- Throughput evaluated

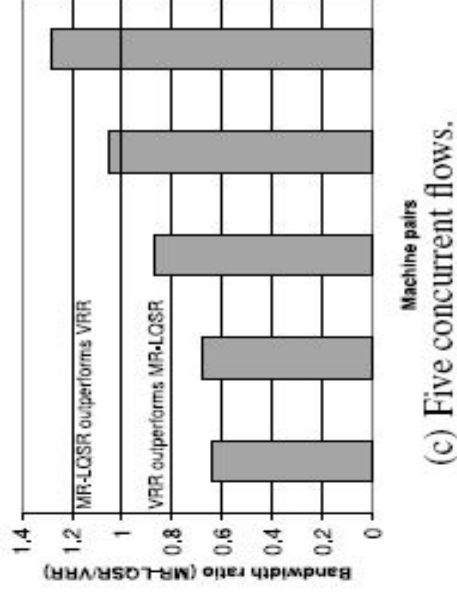
Figure 14: Floor plan of 802.11a PC testbed.



(a) Bandwidth comparison with MR-LQSR.

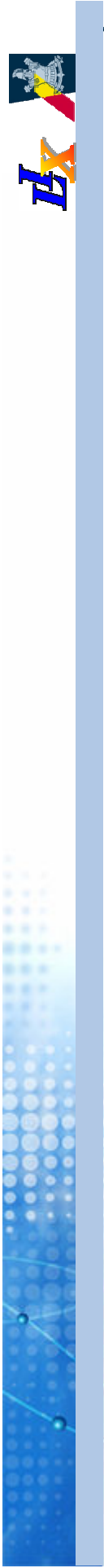


(b) Bandwidth comparison with MR-LQSR.



(c) Five concurrent flows.

Figure 15: 802.11a testbed results.



Discussion