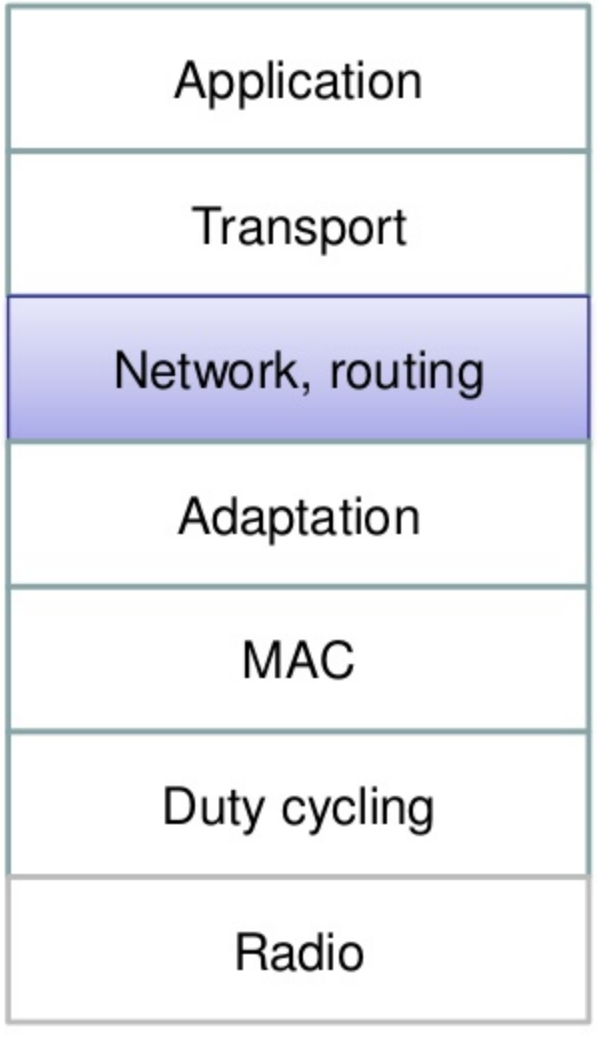
The IoT/IP Protocols

The network layer



The problem

- The general problem
  - Getting data from node A to node C, through node B
- The solutions
  - Addressing, routing
- The wireless problem
  - Intermediate nodes may move
  - Wireless medium is brittle
  - Wireless medium may change over time
- The resource problem
  - Bandwidth-constrained
  - Memory-constrained

Routing

Routing: IETF RPL

- ”Ripple”
- Proactive any-to-any routing protocol
  - But optimized for the many-to-one case
- RPL forms routing graph from root node
- Packet forwarding along graph, routing metric dynamically updated
  - Short-lived routing loops are tolerated

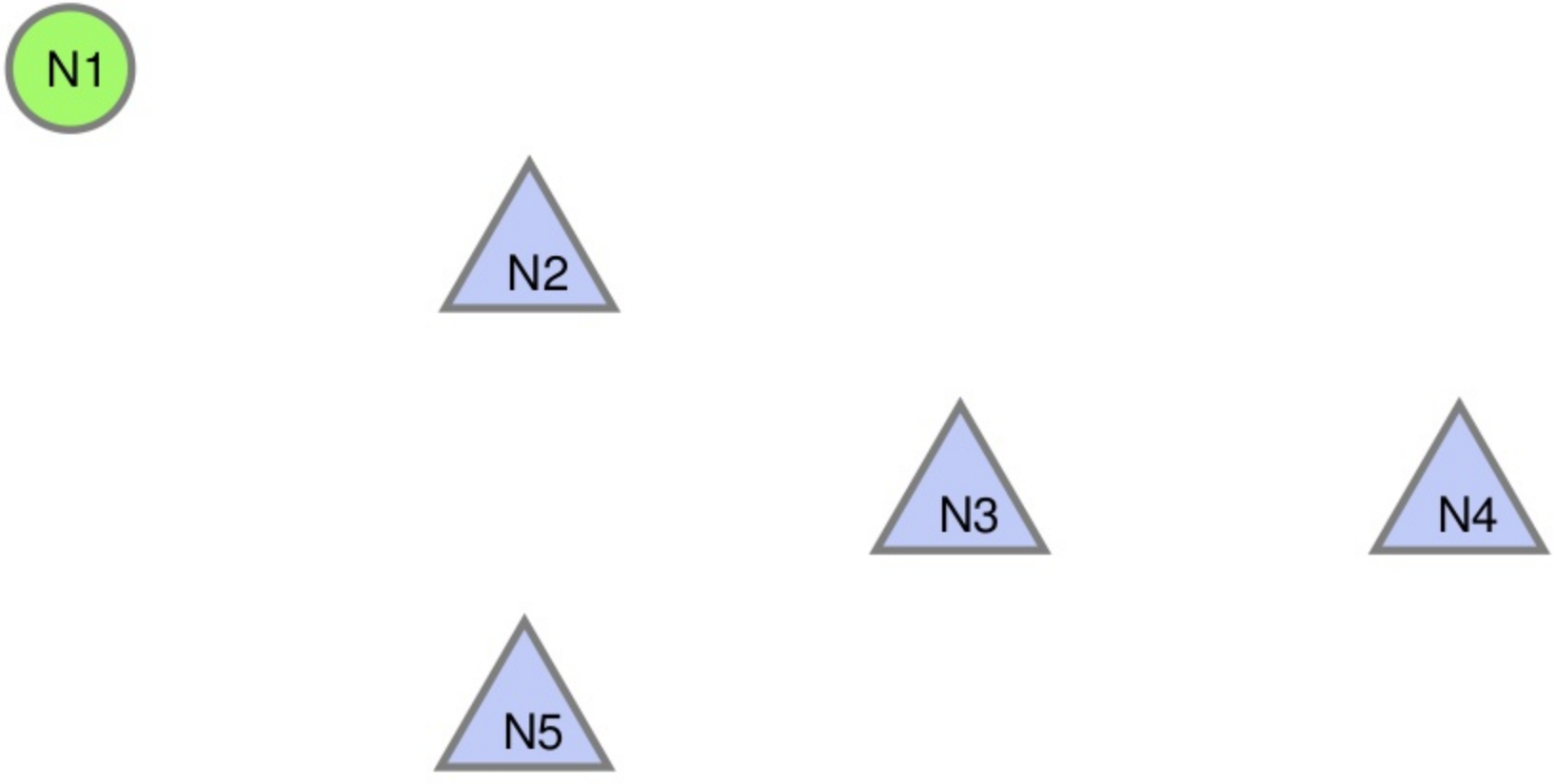
The problems that RPL solves

- Resource constraints
  - Memory constraints
  - Power constraints
  - Bandwidth constraints
- Wireless is unpredictable
  - Pick good routes

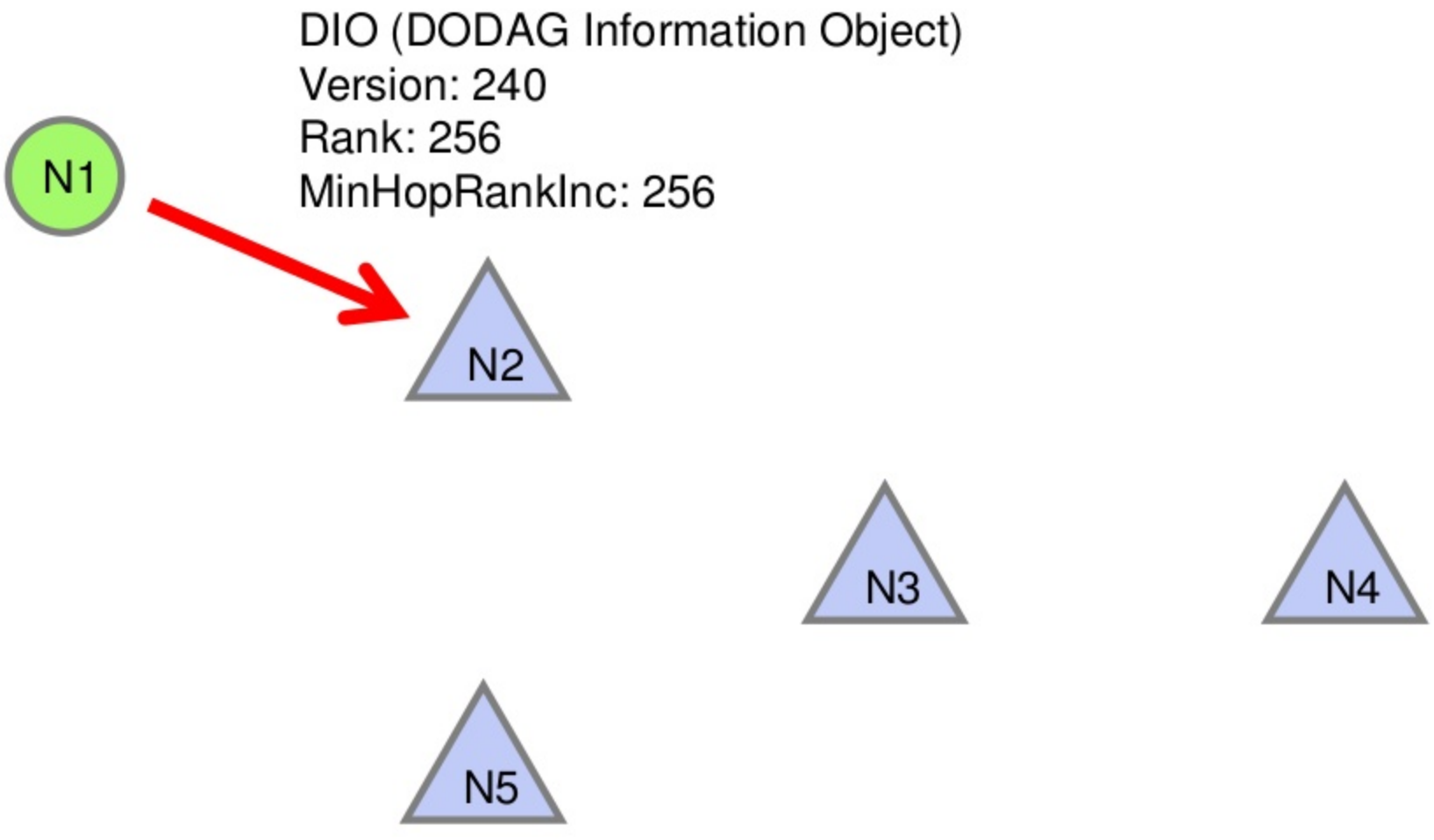
RPL network formation

- Build acyclic graph from root node
  - DODAG – Destination Oriented Directed Acyclic Graph
- DIO messages are broadcast by all nodes, starting from the root node
  - DIO – DODAG Information Object

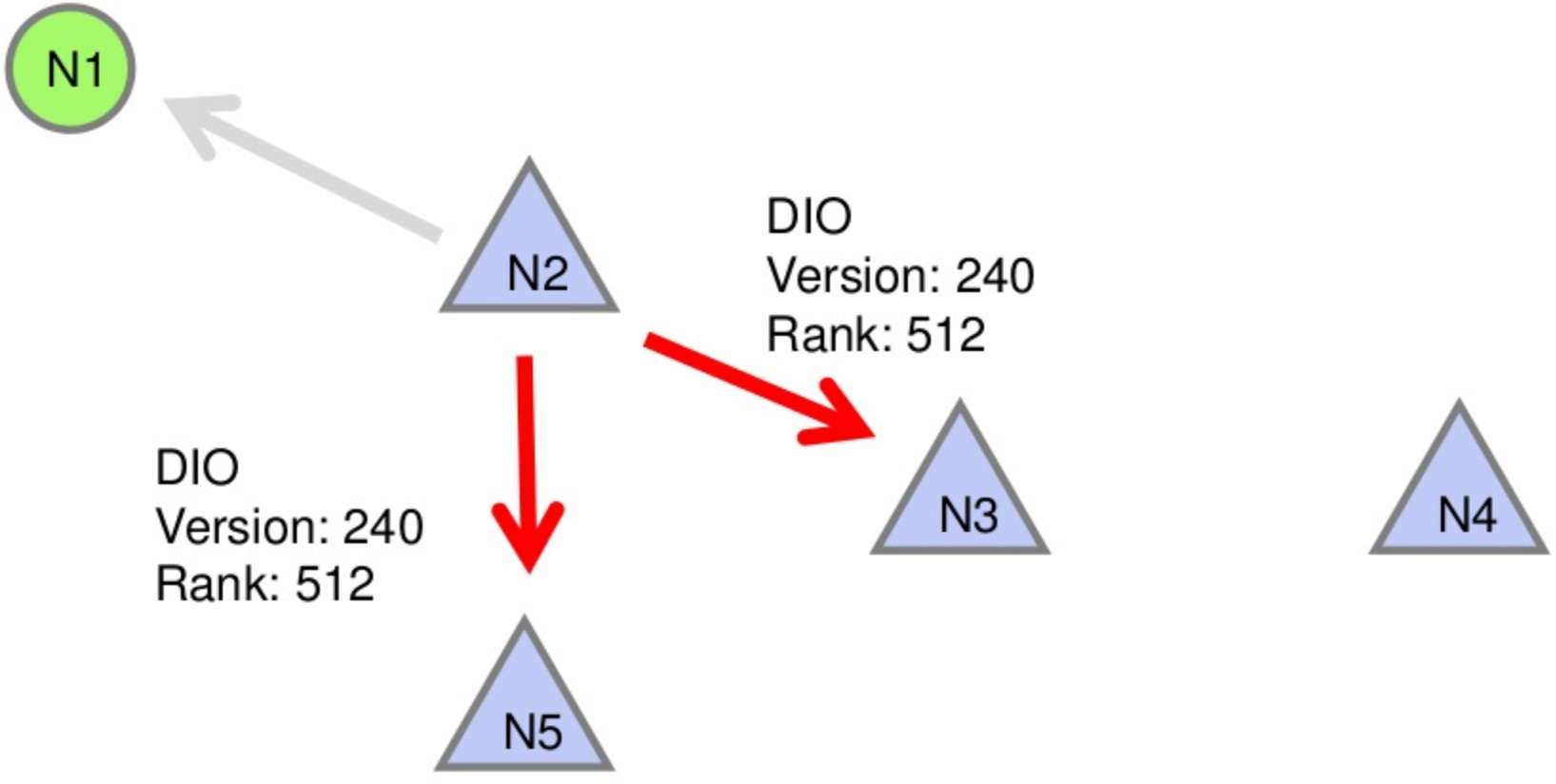
RPL Network Formation



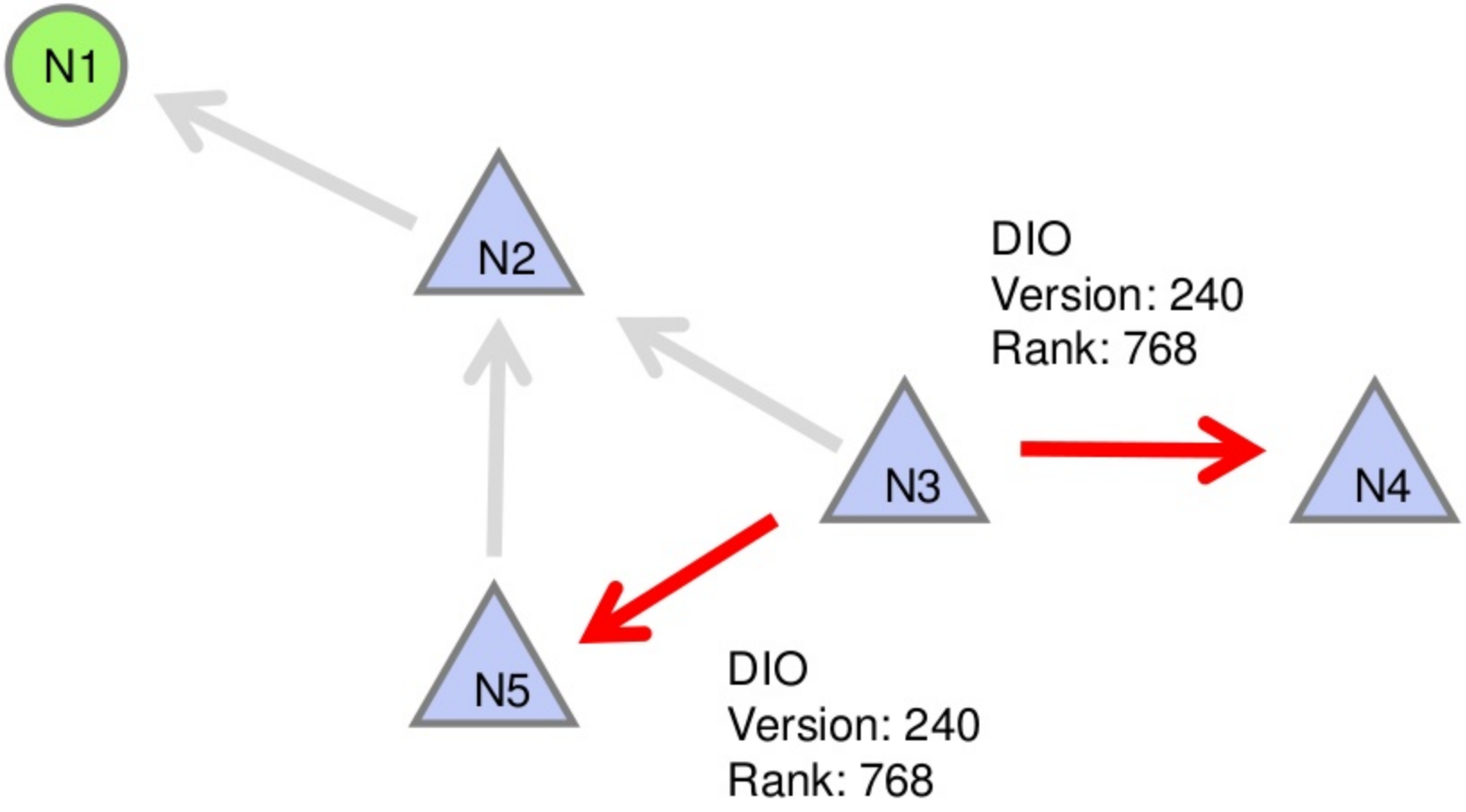
RPL Network Formation



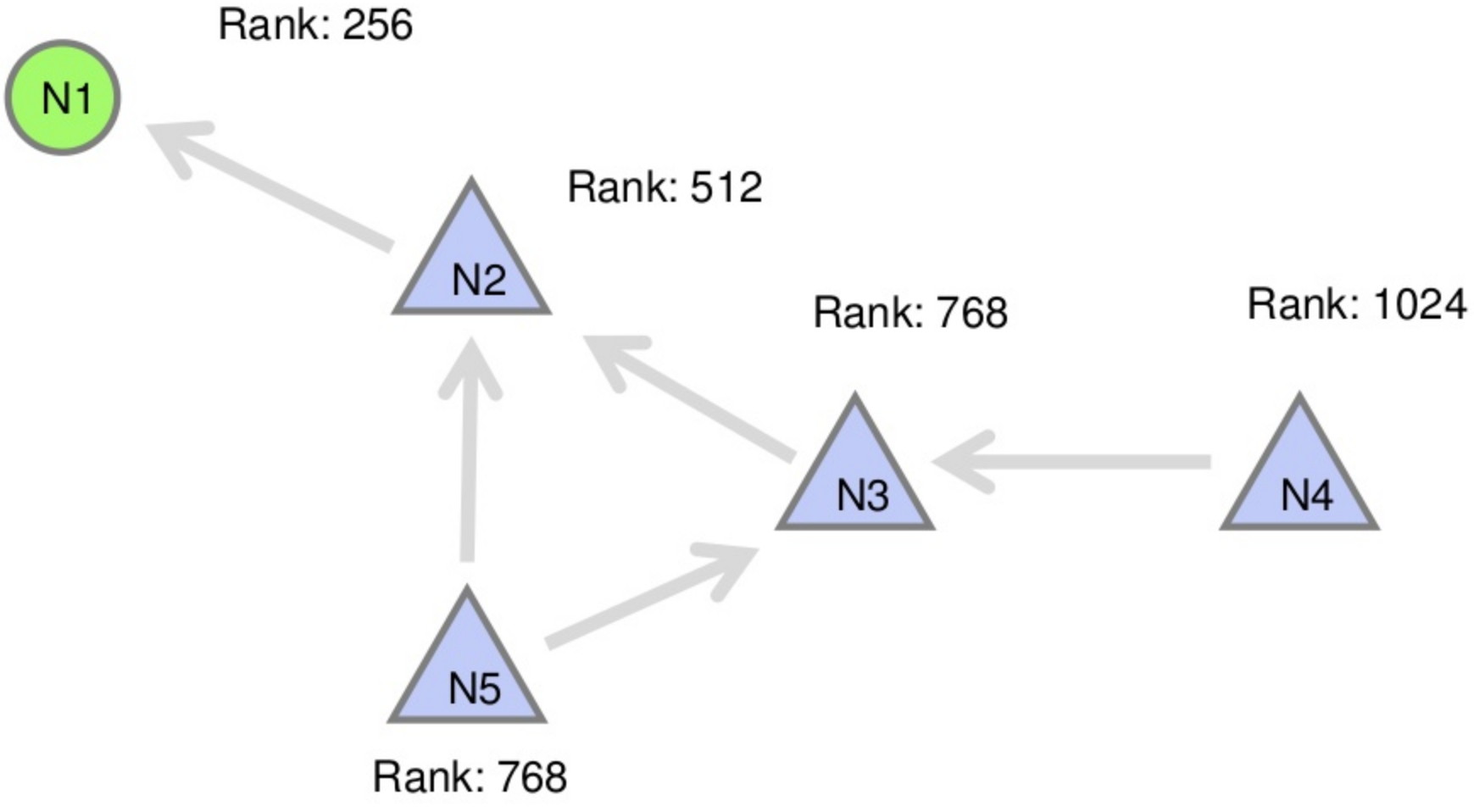
RPL Network Formation



RPL Network Formation



RPL Network Formation



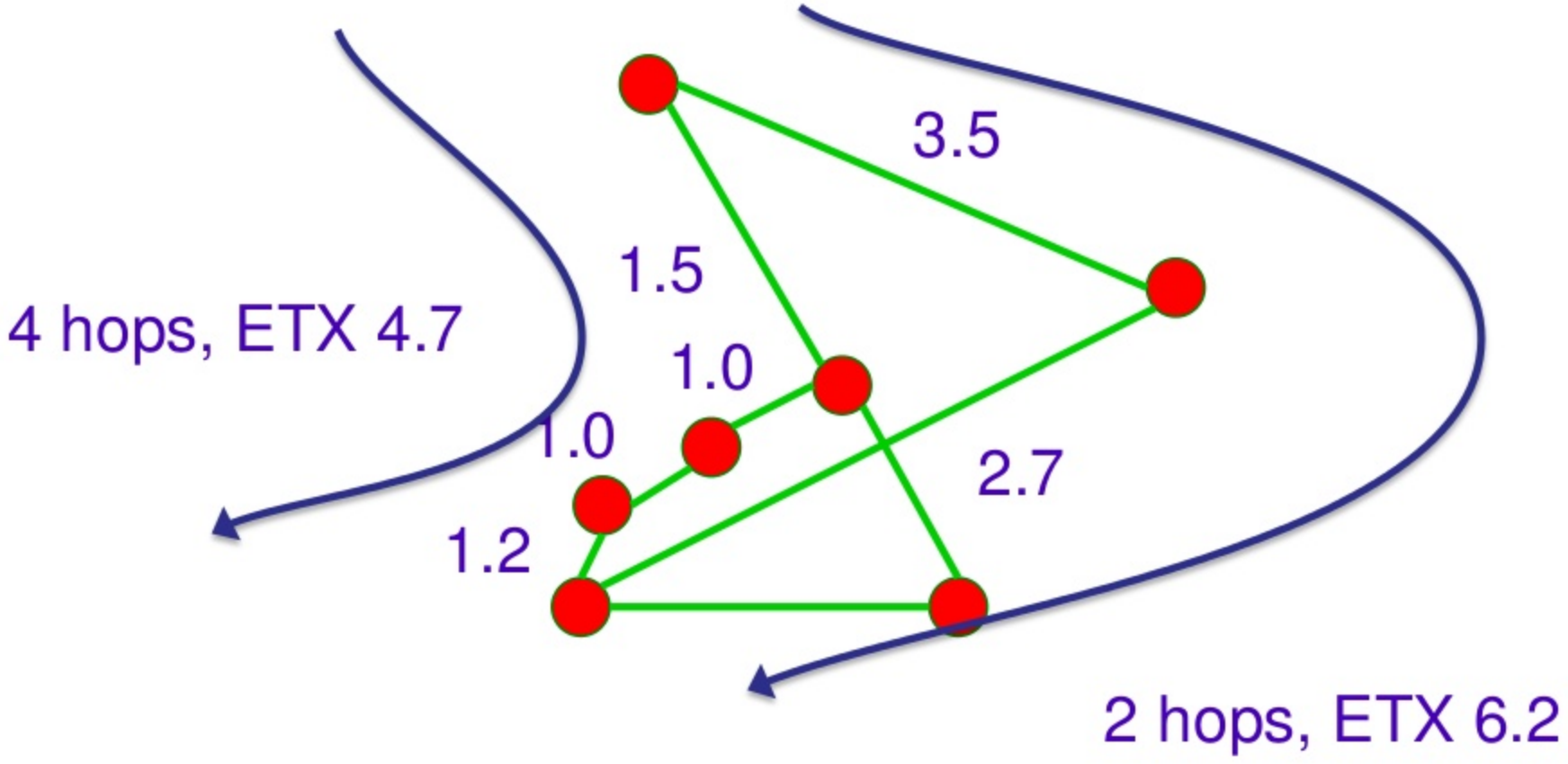
RPL packet forwarding

- Given multiple choices, what route should a packet take?
  - Hop count – shortest amount of hops to the root?
  - Some other dynamic metric?
- RPL leaves this to its Objective Function
- Two Objective Functions specified
  - of0: hop count
  - of1: ETX

ETX – Expected Transmissions

- For every packet transmission, measure how many tries it takes until an ACK is received
  - Use as a measure of how many transmissions to expect
- Keep a moving average, for every neighbor
- To build routes, simply compute the sum of all ETXes

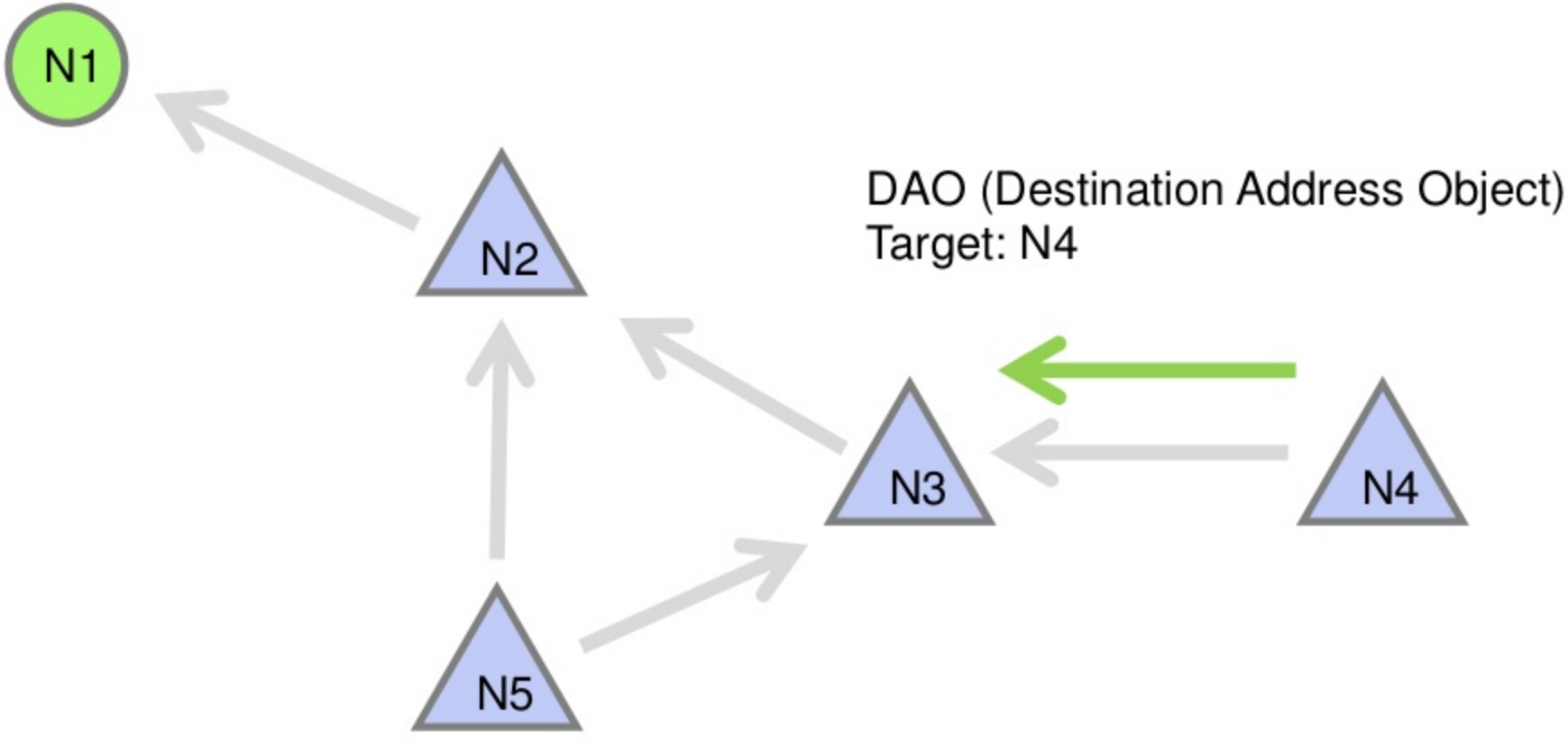
ETX – illustration



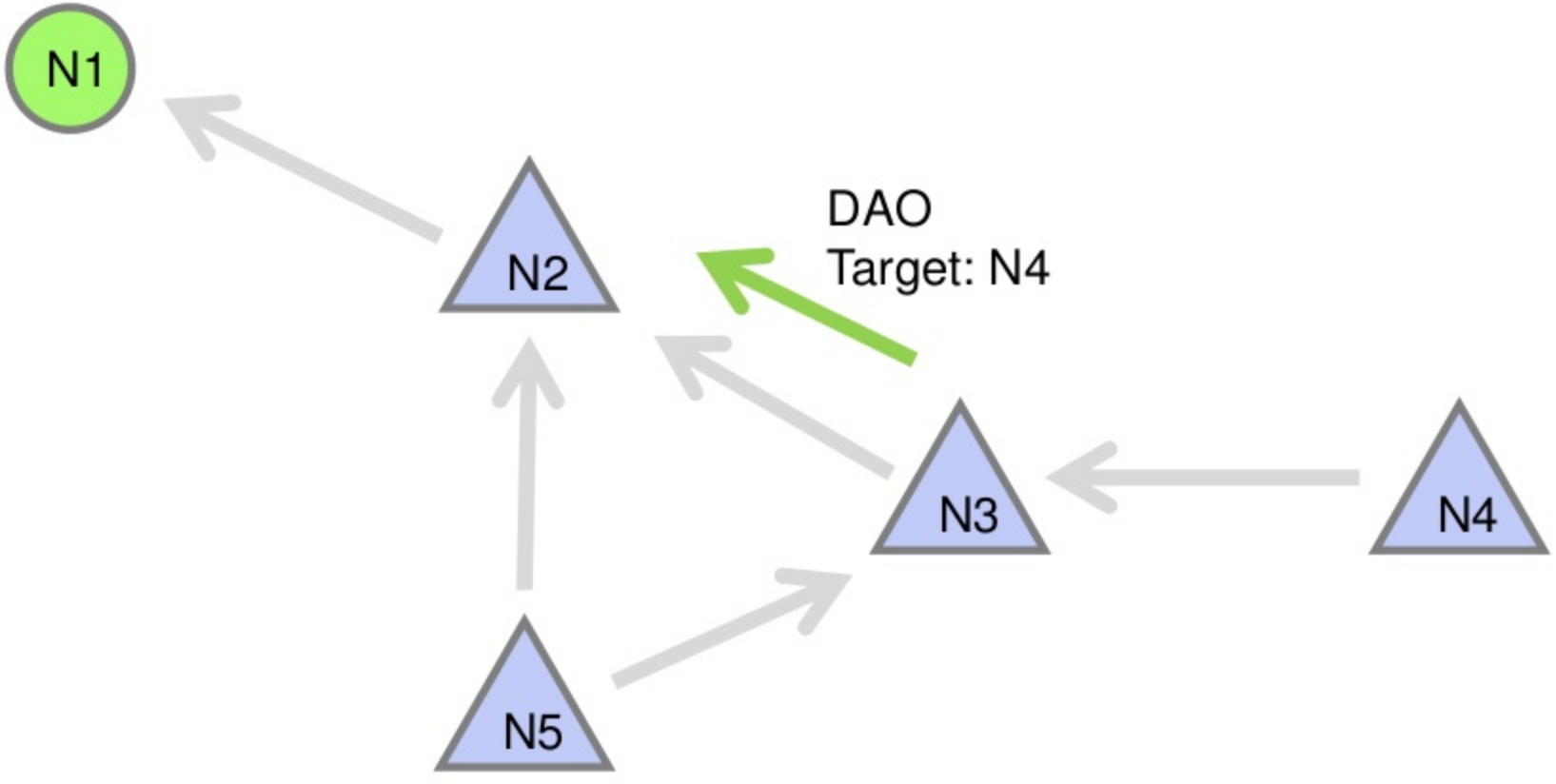
RPL downward routes

- Storing mode vs non-storing mode
  - Storing mode: all nodes store the addresses of their child nodes
    - Drawback: needs routing tables
    - This is how Thingsquare/Contiki does this
  - Non-storing mode: the root node knows the full route to all nodes, sends full route in every packet
    - Drawback: increases header overhead
    - This technique is known as source routing

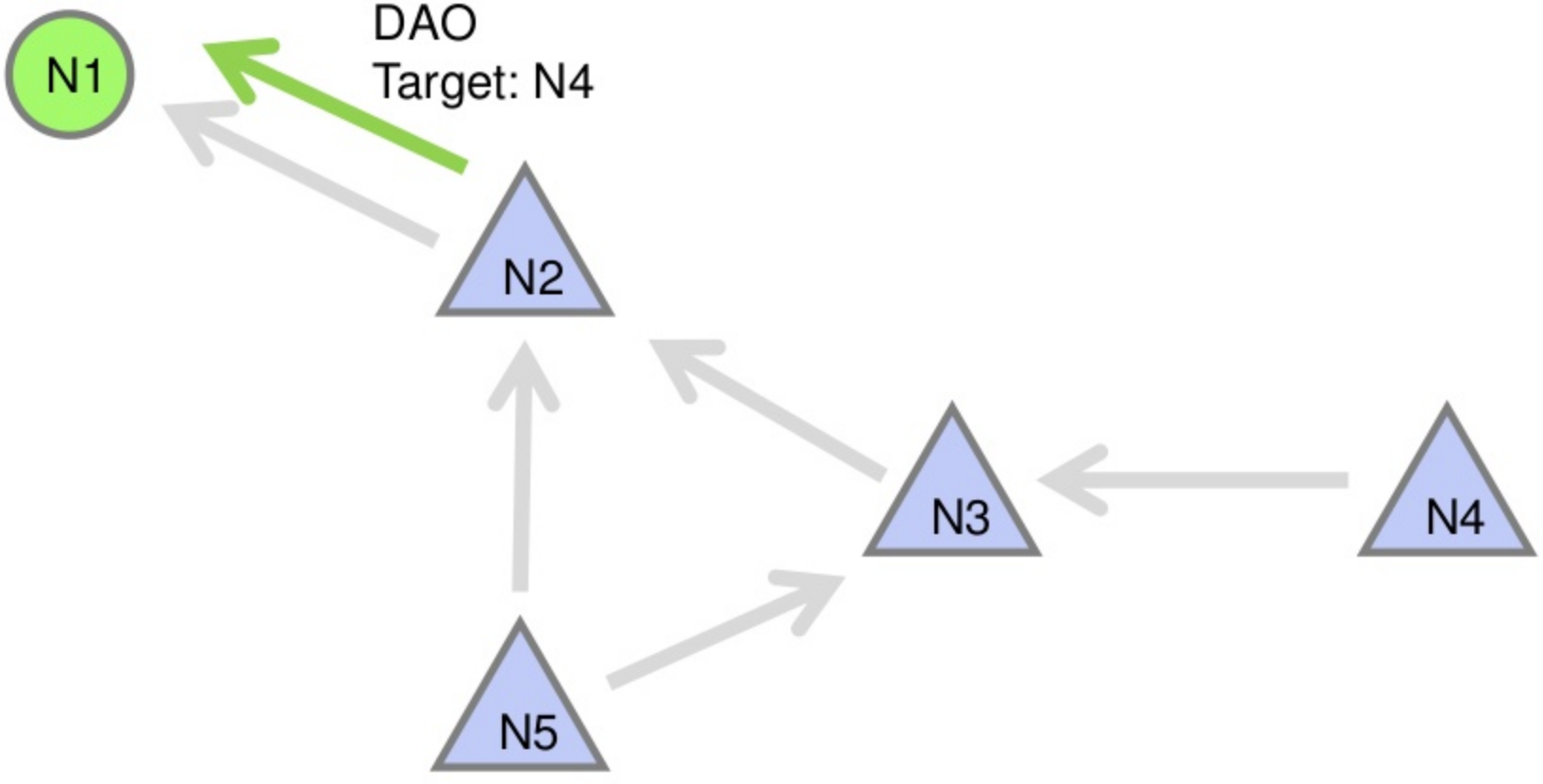
RPL downward routes



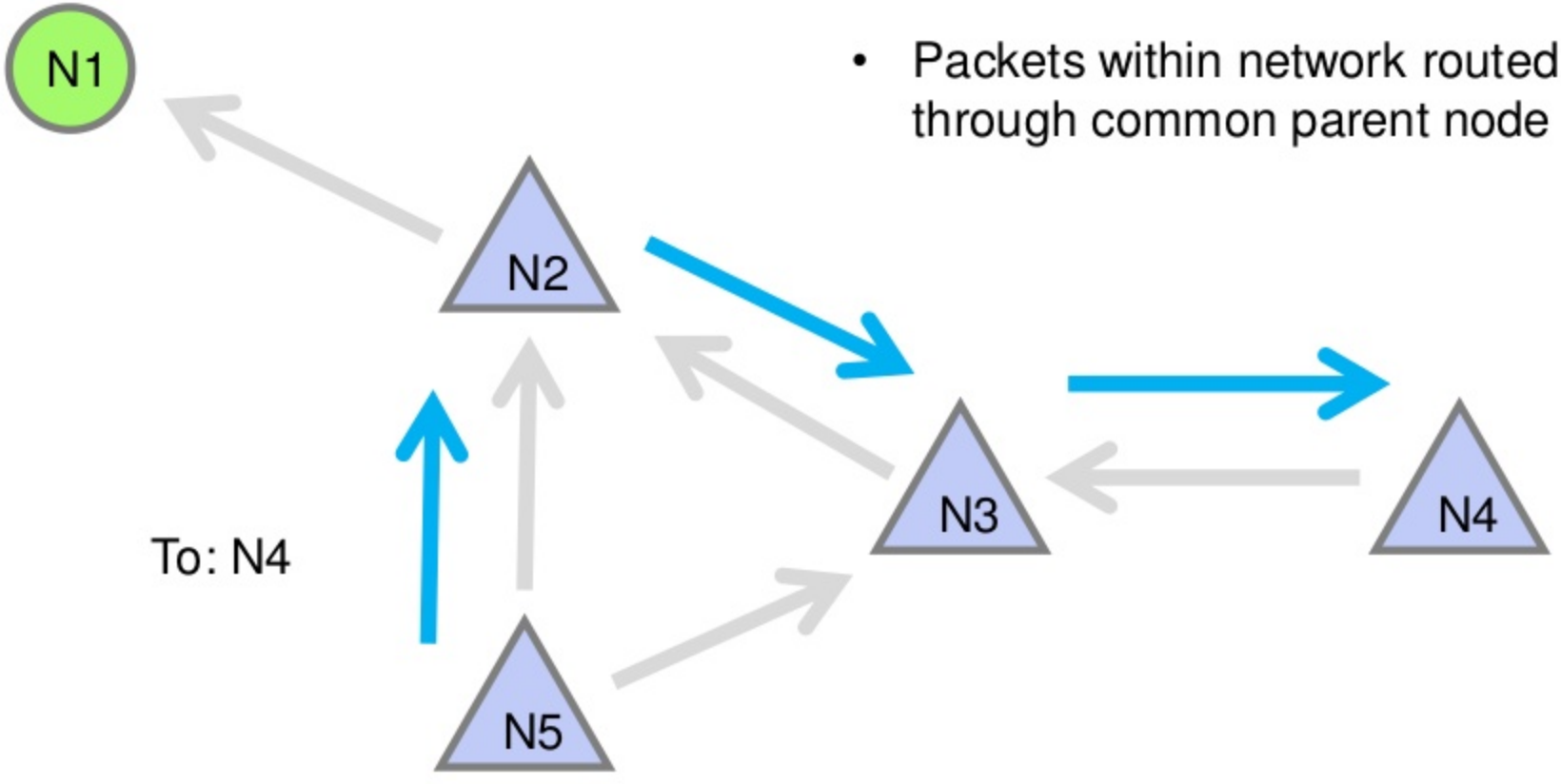
Node IPv6 RPL Network Formation



Node IPv6 RPL Network Formation



RPL Routing: Down



- Packets within network routed through common parent node

RPL network join

- New nodes that appear in the network broadcast Destination Information Solicitation (DIS) packets
- Connected nodes that hear DIS packets respond with a DIO

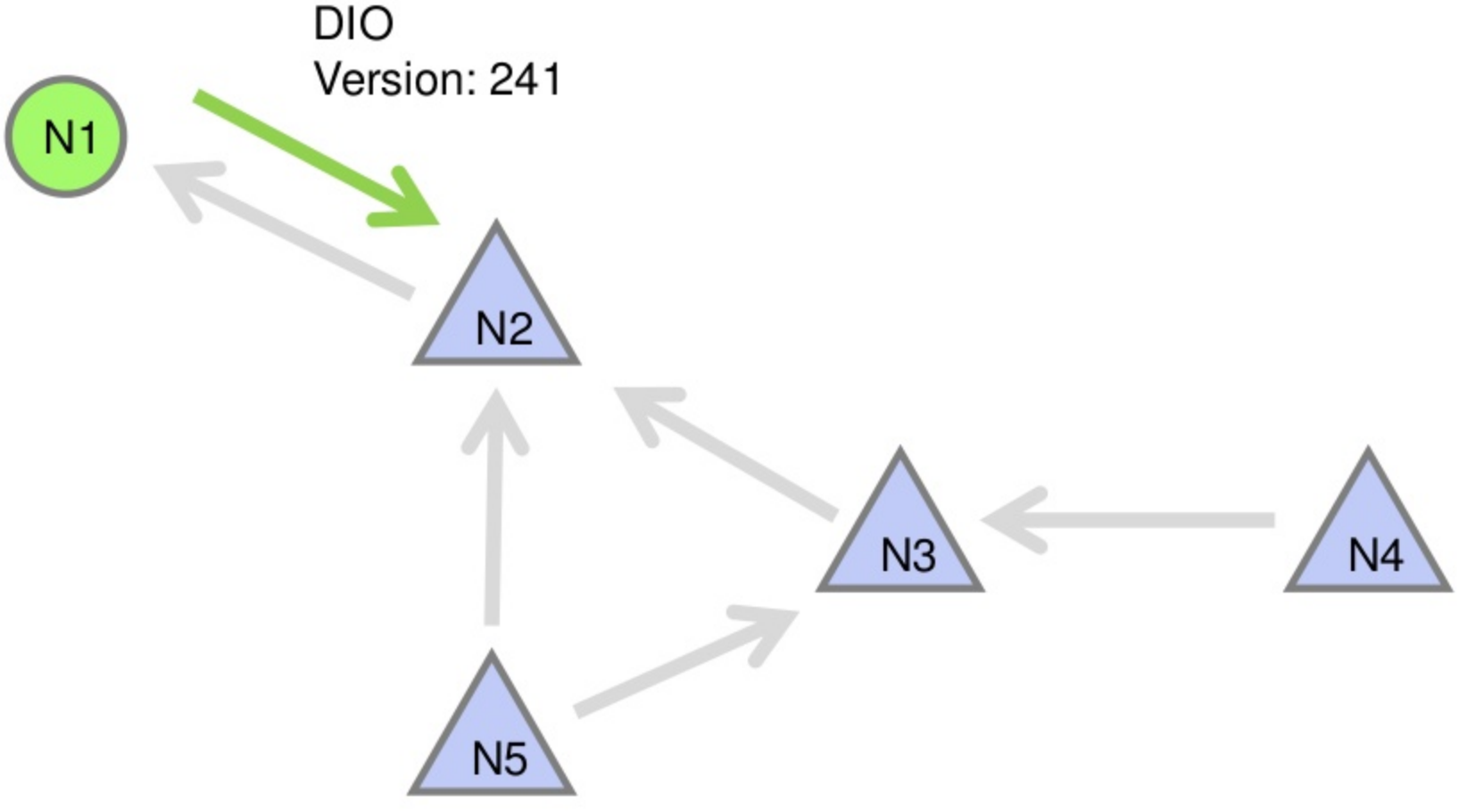
RPL control traffic suppression

- Once the RPL network is established, it reduces the rate of control messages
  - Expontential increase
- To avoid control message explosion, nodes suppress transmissions if it hears too many messages from others
  - Called the Trickle algorithm (Phil Levis et al)

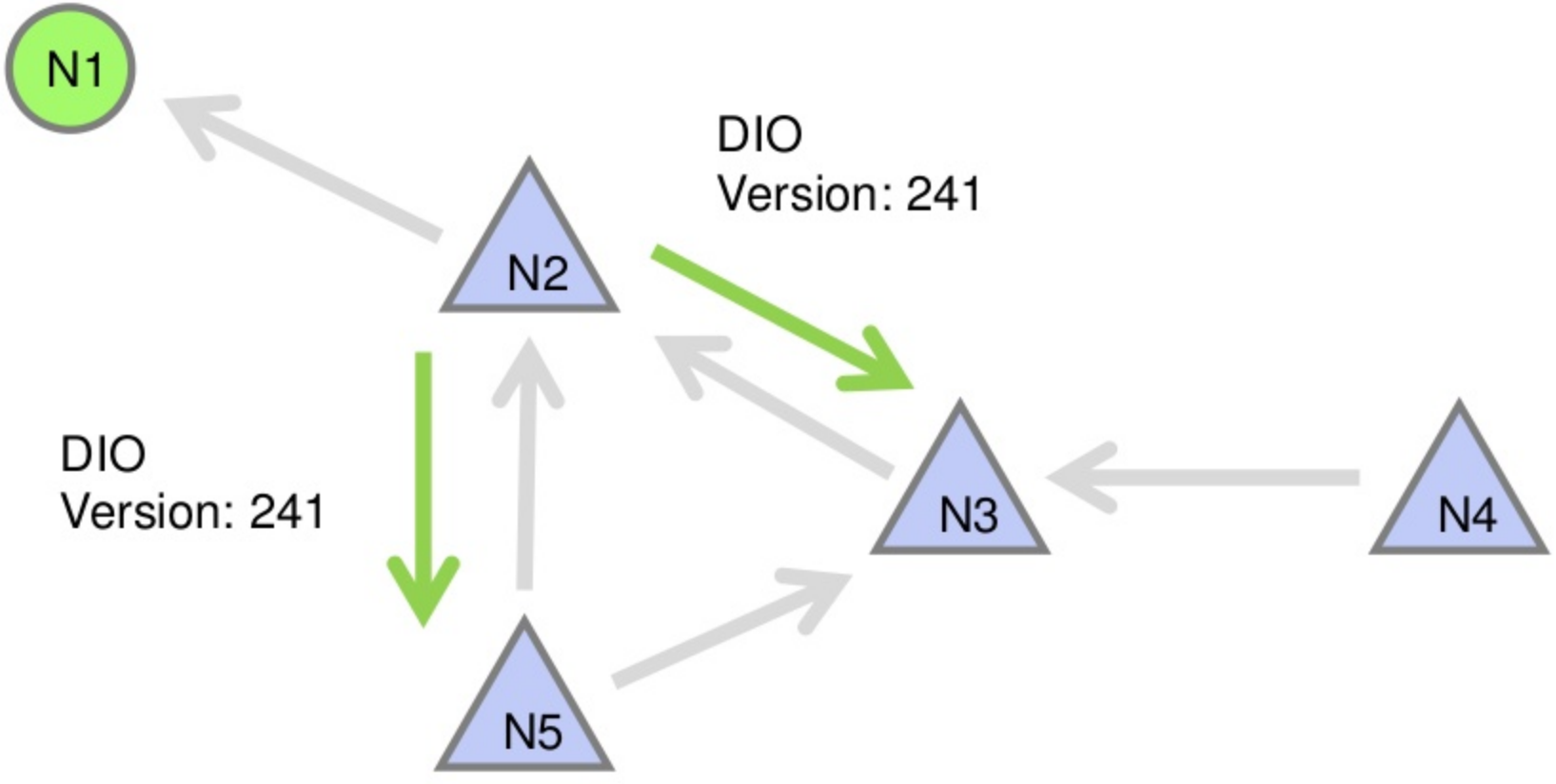
RPL repair

- Loop detection: packets carry backward path information in header
- When a loop is detected, the first few packets are allowed through the loop
- If the same packet is seen once again, a local repair is initiated
  - Node sets rank to infinity, sends DIS
- RPL global repair
  - Increase version number, rebuild DODAG

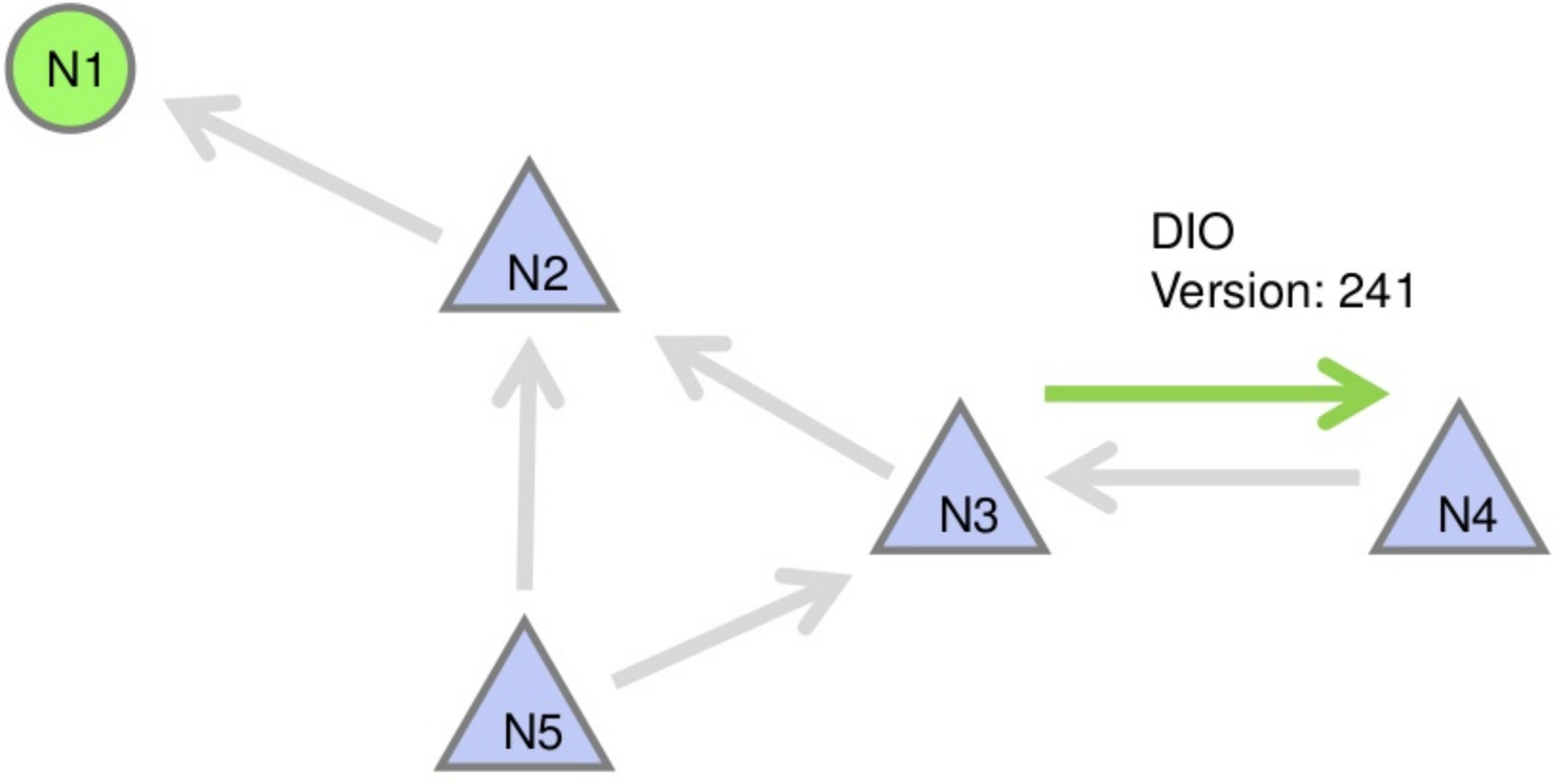
RPL Global Repair



RPL Global Repair



RPL Global Repair



RPL any-to-any routing

- RPL is optimized for many-to-one routing
  - All report packets to the sink
- Any-to-any routing must go through closest common ancestor
  - The root, in the worst case
- RPL P2P mode a suggested solution
  - AODV-like reactive route discovery with source routing
  - Contiki implementation exists: http://contiki-p2p-rpl.gforge.inria.fr/

RPL pitfalls

- Rebooting the root node
  - Must listen for a while to obtain network version
- Global repair may take a while

In Contiki

- ContikiRPL code is difficult to use
  - Integrated with the uIPv6 stack
  - Complex API
- Thingsquare provides a simple-rpl module
  - Gracefully handles root node reboots
  - Provides global repair / local repair API

In Contiki

- RPL node types
  - Mesh nodes: RPL routers, can be routed to
  - Leaf nodes: does not route RPL traffic, can be routed to
  - Feather nodes: routes RPL traffic, cannot be routed to