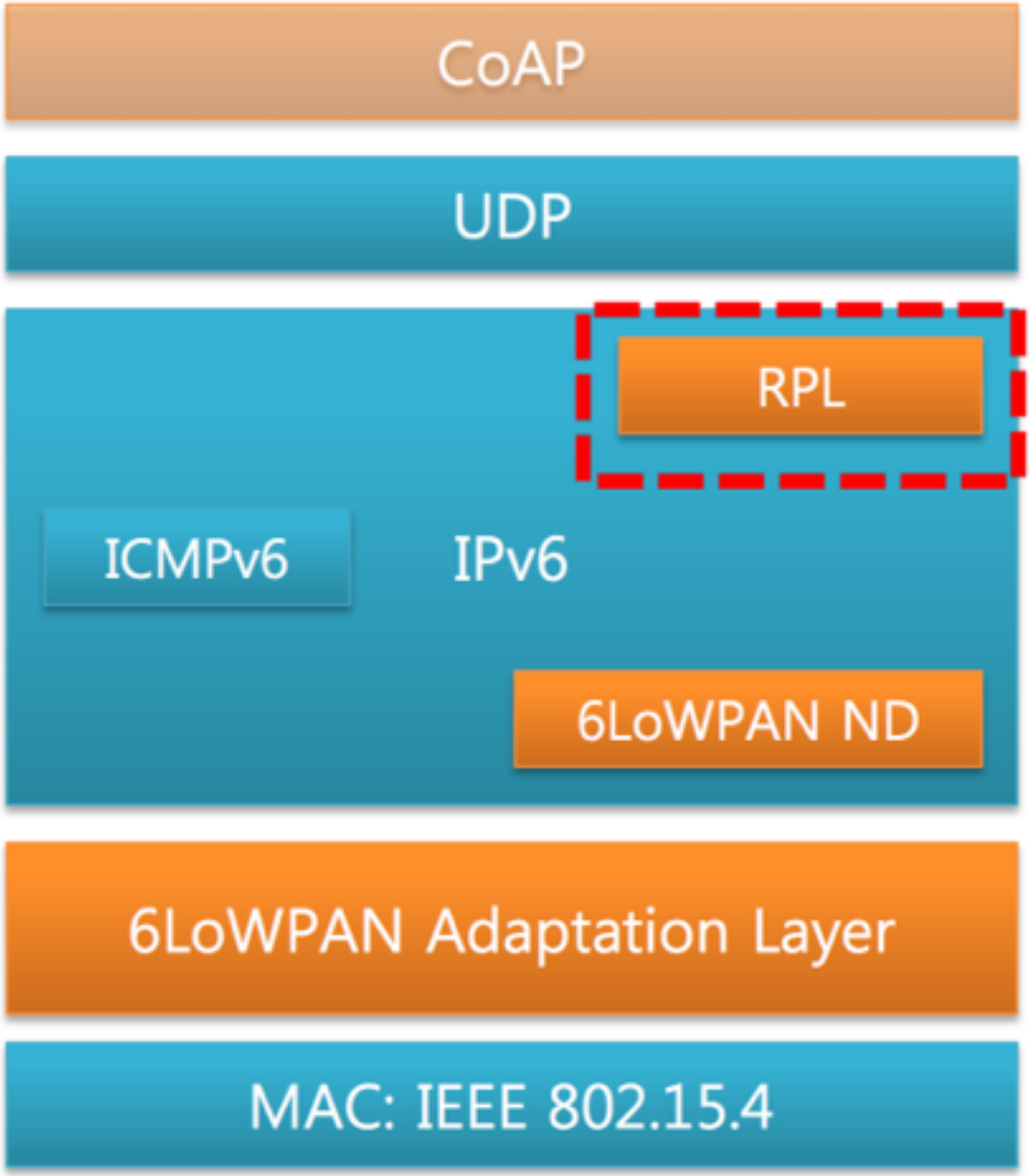
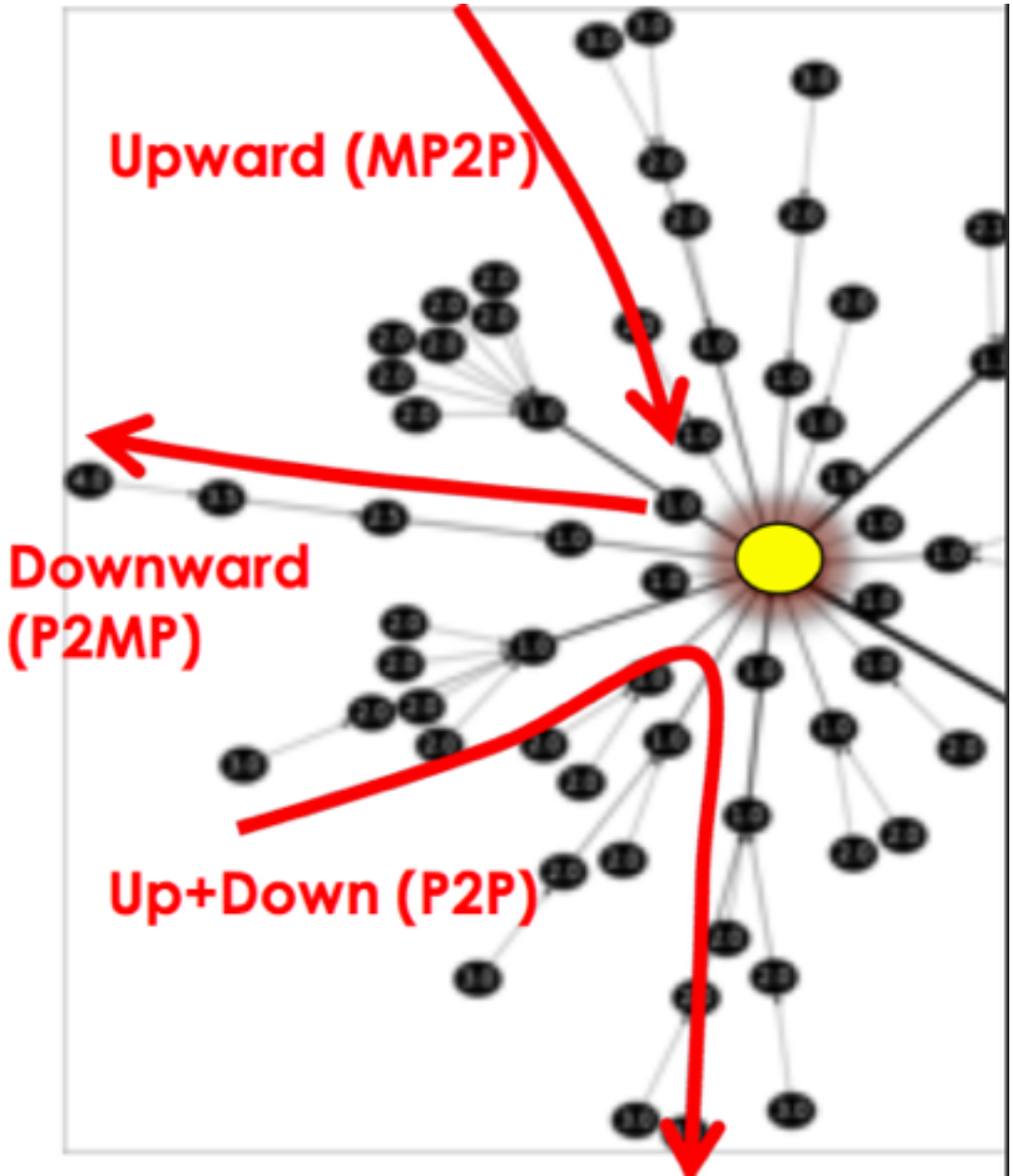
RPL

Routing Protocol for Low-Power and Lossy Networks


CoAP
UDP
RPL
ICMPv6
IPv6
6LoWPAN ND
6LoWPAN Adaptation Layer
MAC: IEEE 802.15.4

Features

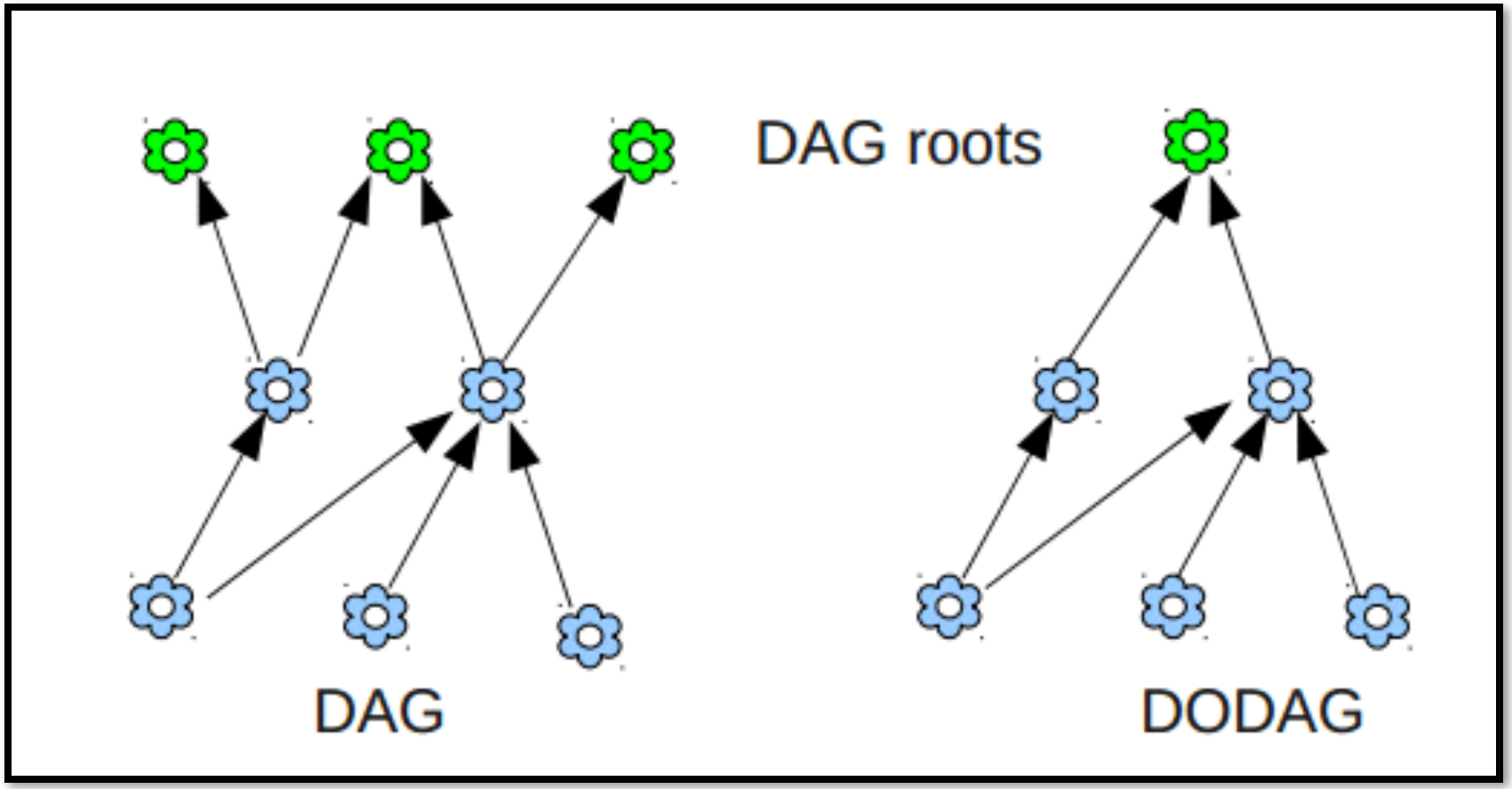
- Traffic patterns
  - **MP2P**: forward to a default router
  - **P2MP**: storing or non-storing
  - **P2P**: combination of MP2P and P2MP
- Directed Acyclic Graph
  - Distance-Vector (Rank)
  - Proactive construction
  - Trickle timer
- Objective Function
  - To achieve various goals…
    - How to calculate rank values?
    - How to construct DAG?
  - Separated from core function
  - User definable and adjustable


Upward (MP2P)
Downward
(P2MP)
Up+Down (P2P)

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

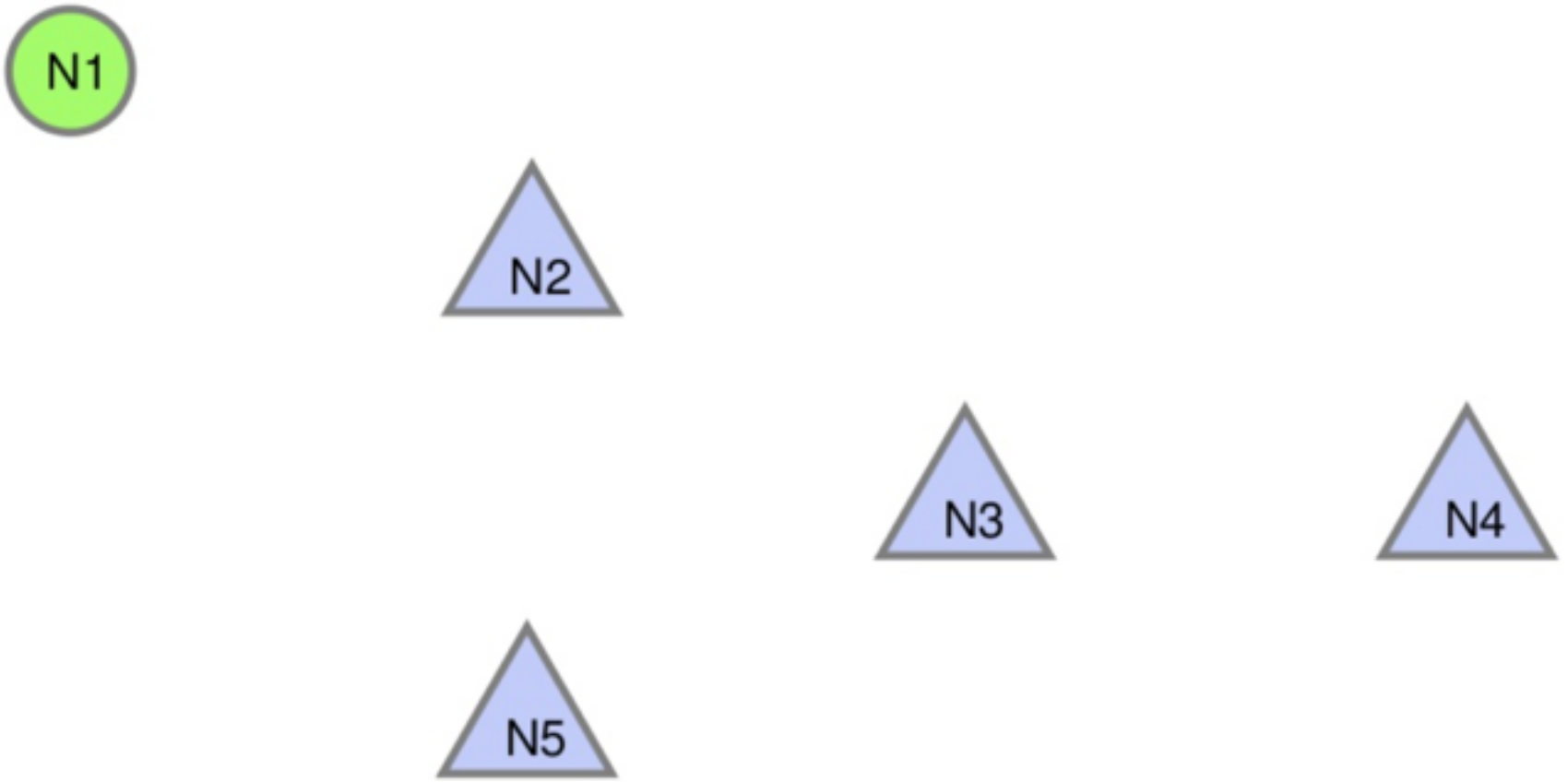
DODAG & DAG



DAG vs DODAG

A Directed Acyclic Graph (DAG) is an on oriented graph with no directed cycles, and is formed by a collection of vertices and directed edges. Each edge connects one vertex to another, so there is no way to start at some vertex v and follow a sequence of edges that eventually loops back to v again. A DAG root is a node within the DAG that has no outgoing edge; because the graph is acyclic, by definition all DAGs must have at least one DAG root and all paths terminate at a DAG root. A DAG with an unique root node forms a Destination Oriented DAG (DODAG). RPL protocol creates a DODAG structure looks like a tree where the root is named DODAG root; in this mode each node could became parent, child, ancestor or leaf.

RPL Network Formation

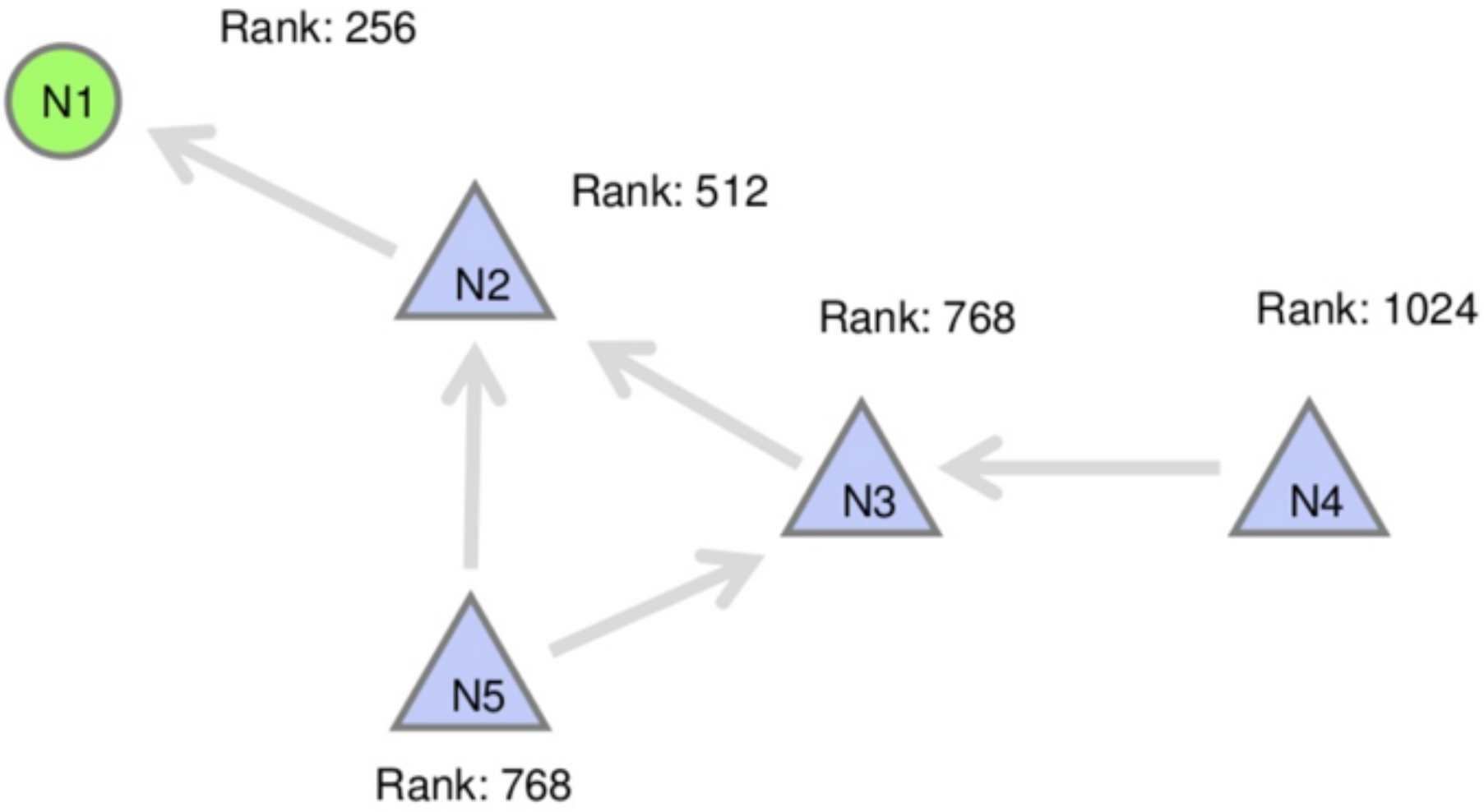


RPL Network Formation

RPL Network Formation

RPL Network Formation

RPL Network Formation



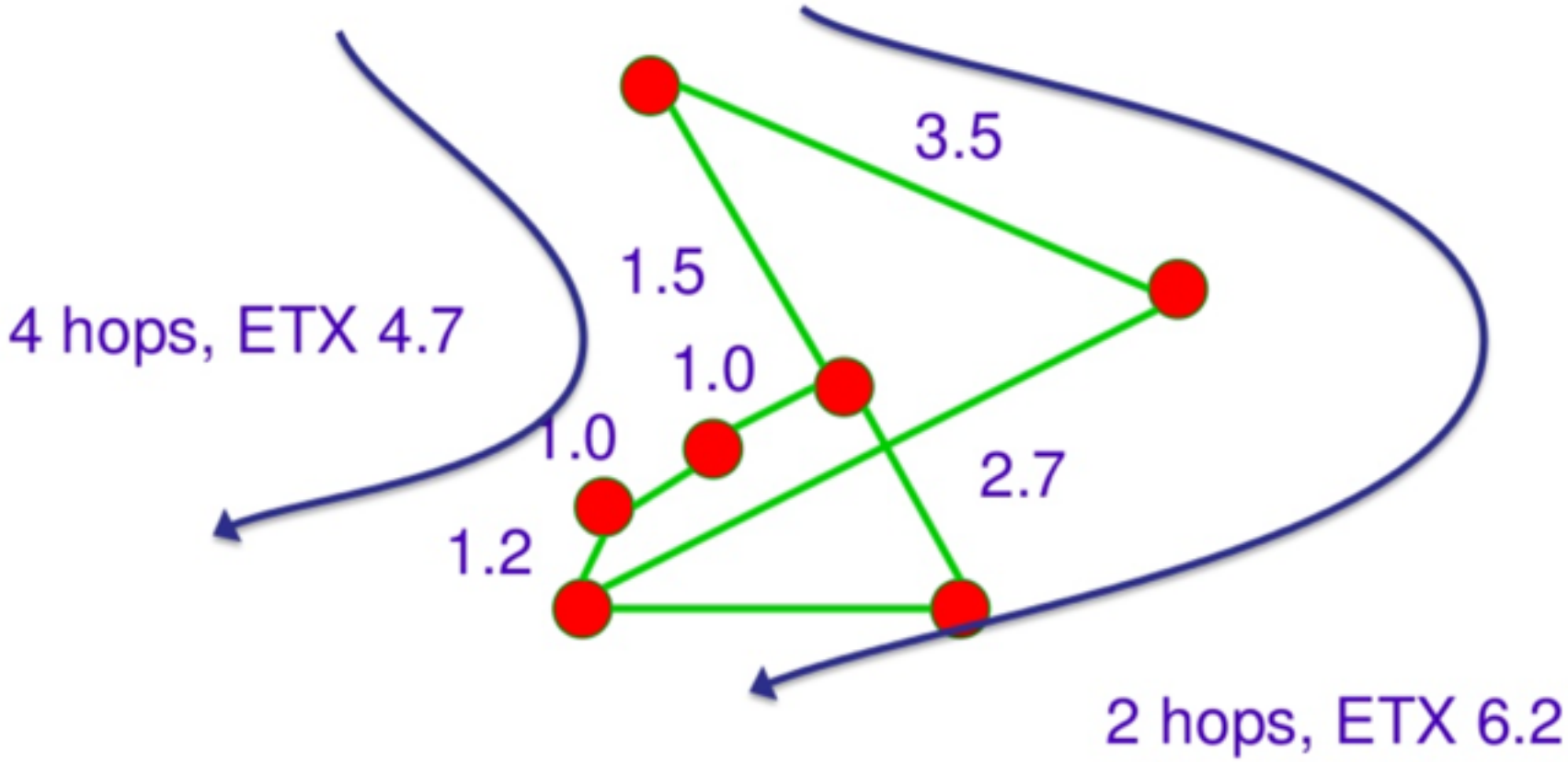
RPL packet forwarding

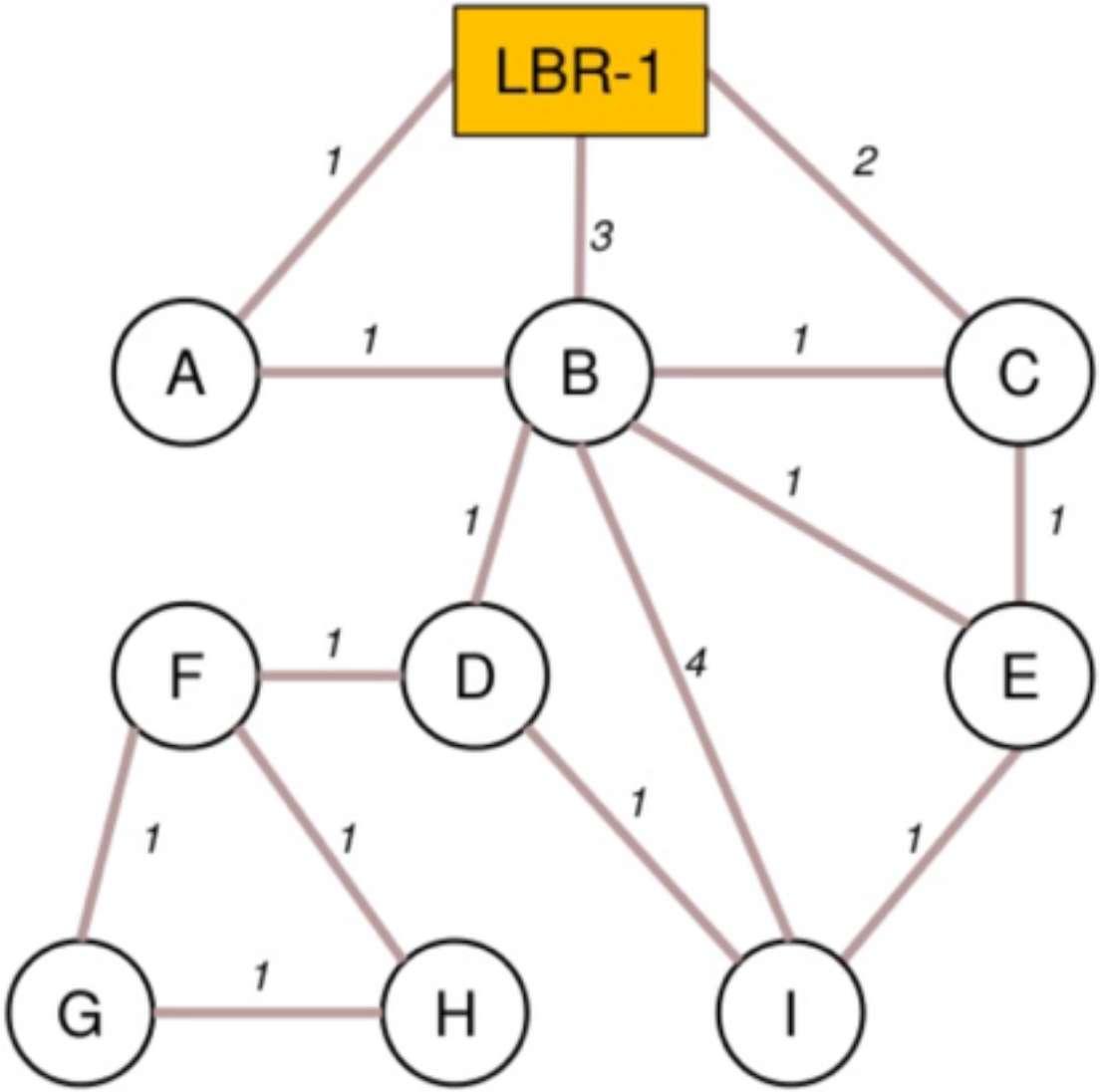
- Given multiple choices, what route should a packet take?
  - Hop count – shortest amount of hops to the root?
  - Some other dynamic metric?
- RPL leaves this to its Objective Function
- Two Objective Functions specified
  - of0: hop count
  - of1: ETX

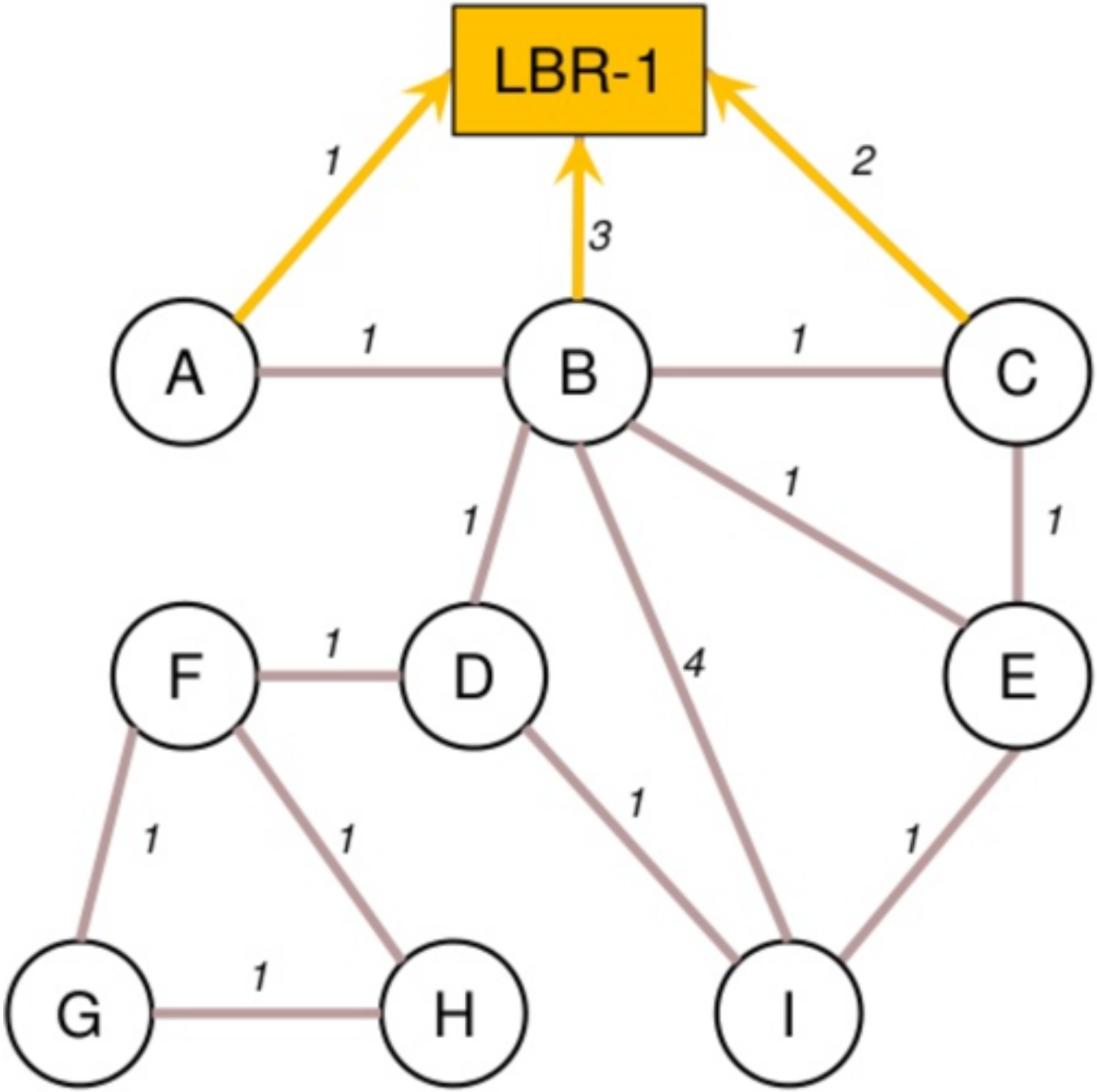
ETX – Expected Transmissions

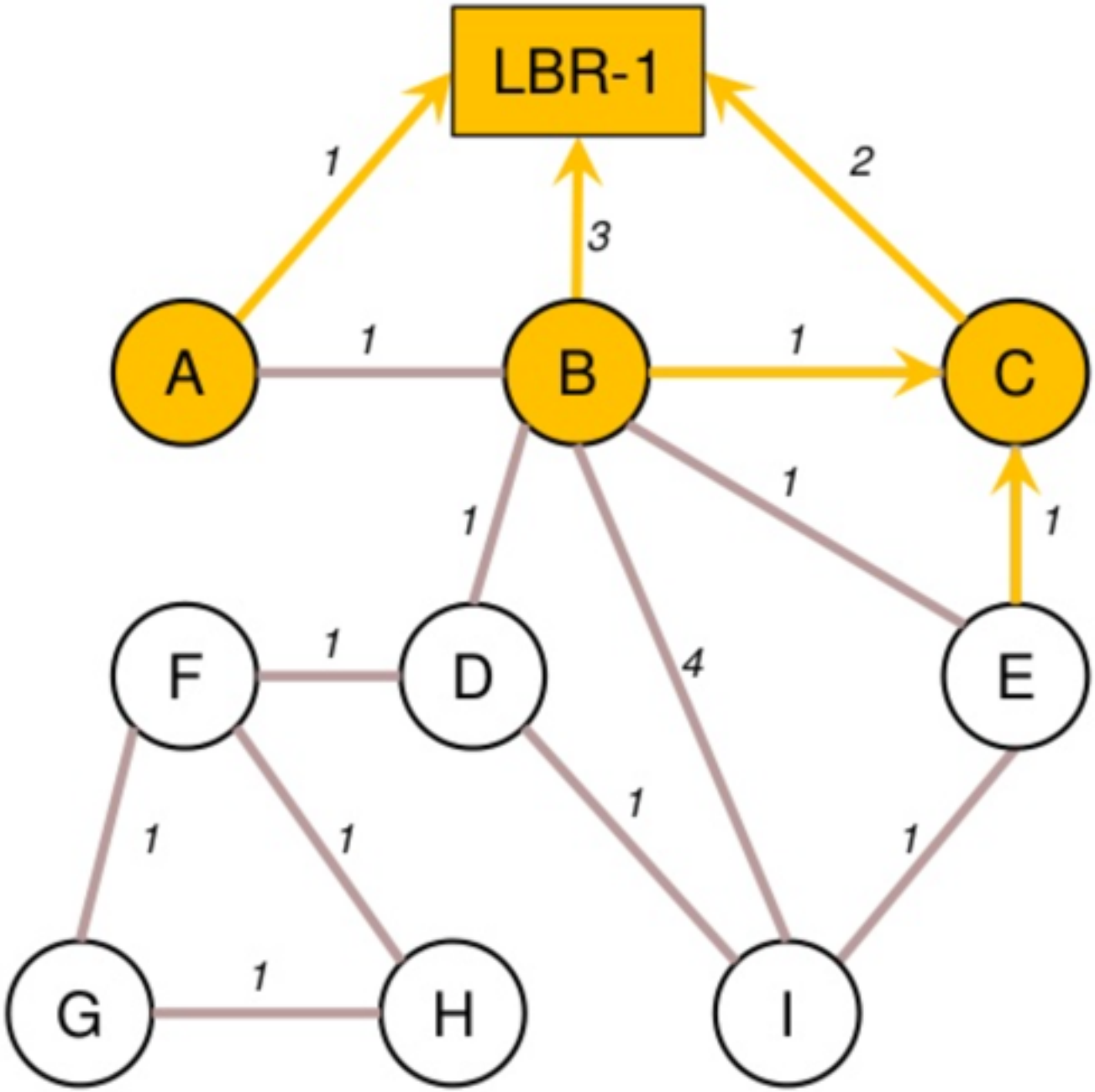
- For every packet transmission, measure how many tries it takes until an ACK is received
  - Use as a measure of how many transmissions to expect
- Keep a moving average, for every neighbor
- To build routes, simply compute the sum of all ETXes

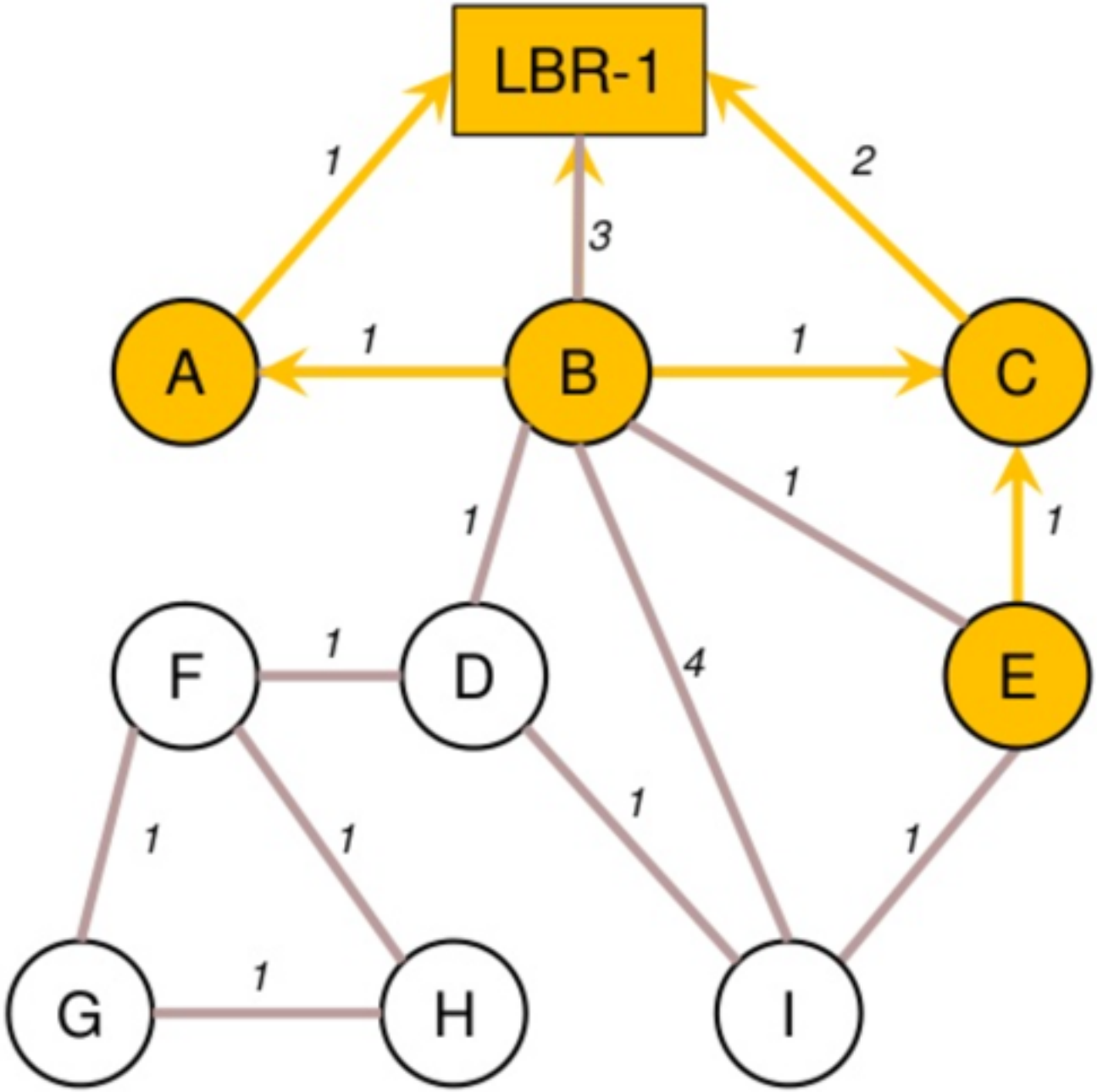
ETX – illustration

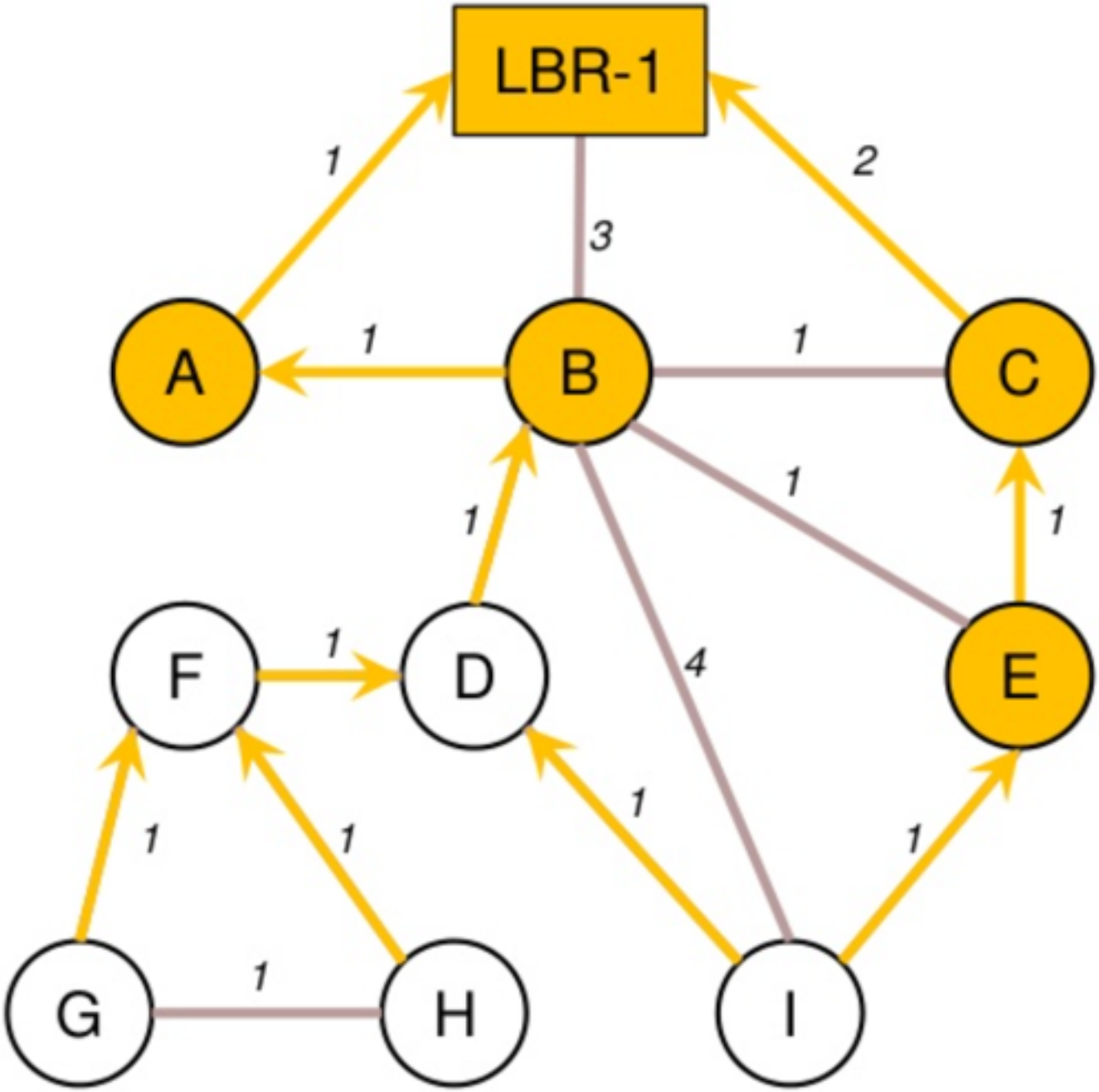



LBR-1
1
3
2
A
1
B
1
C
1
1
1
F
1
D
4
E
1
1
1
1
G
1
H
I


LBR-1
1
3
2
A
1
B
1
C
1
1
1
F
1
D
4
E
1
1
1
1
G
1
H
I


LBR-1
1
3
2
A
1
B
1
C
1
1
1
F
1
D
4
E
1
1
1
1
1
G
1
H
I


LBR-1
1
3
2
A
1
B
1
C
1
1
1
F
1
D
4
E
1
1
1
1
1
G
1
H
I


LBR-1
A
B
C
D
E
F
G
H
I
1
2
3
1
1
1
1
1
1
4
1
1
1
1
1

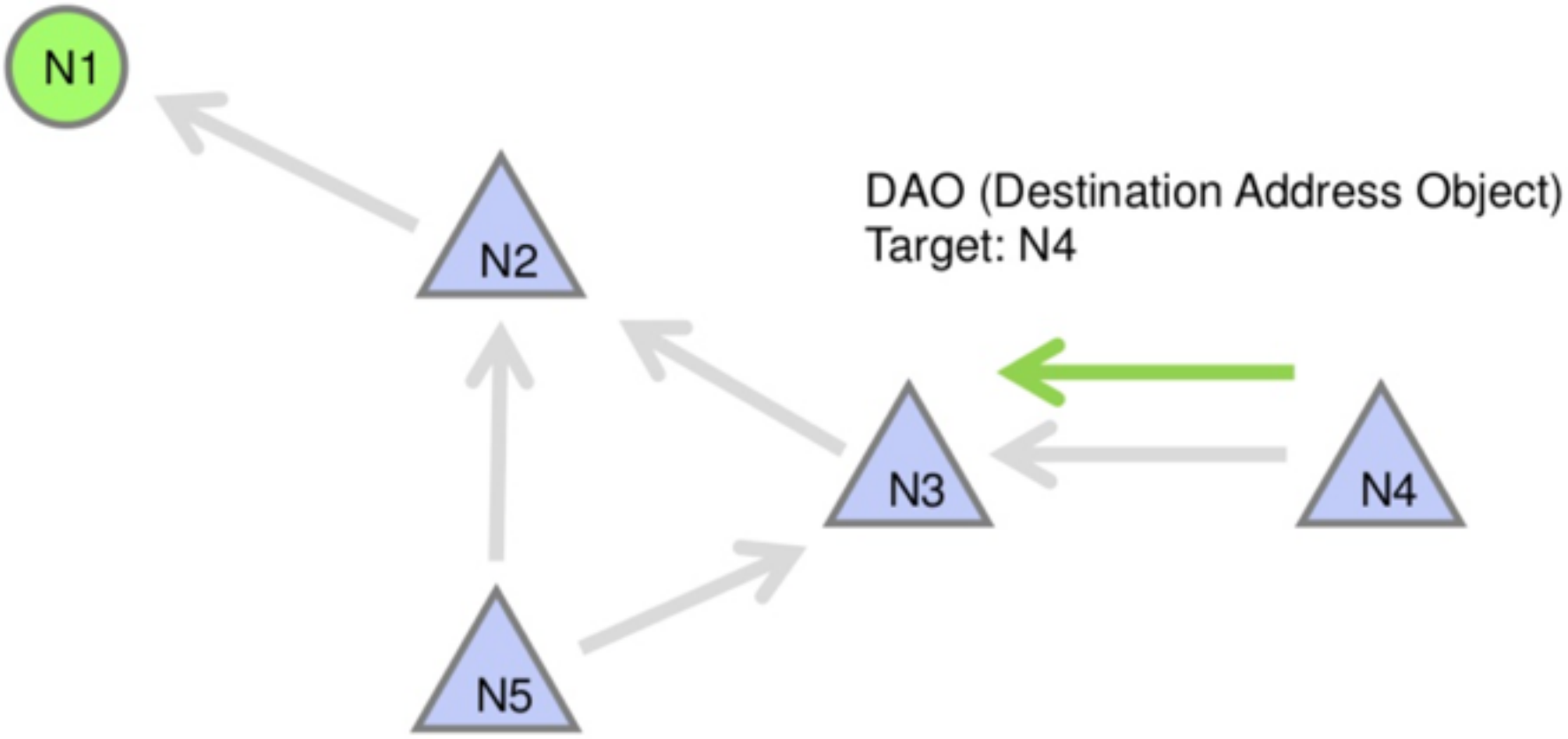
METRICS

Node Metrics	Link Metrics
Node State and Attributes Object Purpose is to reflects node workload (CPU, Memory…) "O" flag signals overload of resource "A" flag signal node can act as traffic aggregator	**Throughput Object** Currently available throughput (Bytes per second) Throughput range supported
Node Energy Object "T" flag: Node type: 0 = Mains, 1 = Battery, 2 = Scavenger "I" bit: Use node type as a constraint (include/ exclude) "E" flag: Estimated energy remaining	**Latency** Constraint - max latency allowable on path Metric - additive metric updated along path
Hop Count Object Constraint - max number of hops that can be traversed Metric - total number of hops traversed	**Link Reliability** Link Quality Level Reliability (LQL) 0=Unknown, 1=High, 2=Medium, 3=Low Expected Transmission Count (ETX) (Average number of TX to deliver a packet)
Object can be used as metric and/or constraint - metric can be additive/max/..	**Link Colour** Metric or constraint, arbitrary admin value

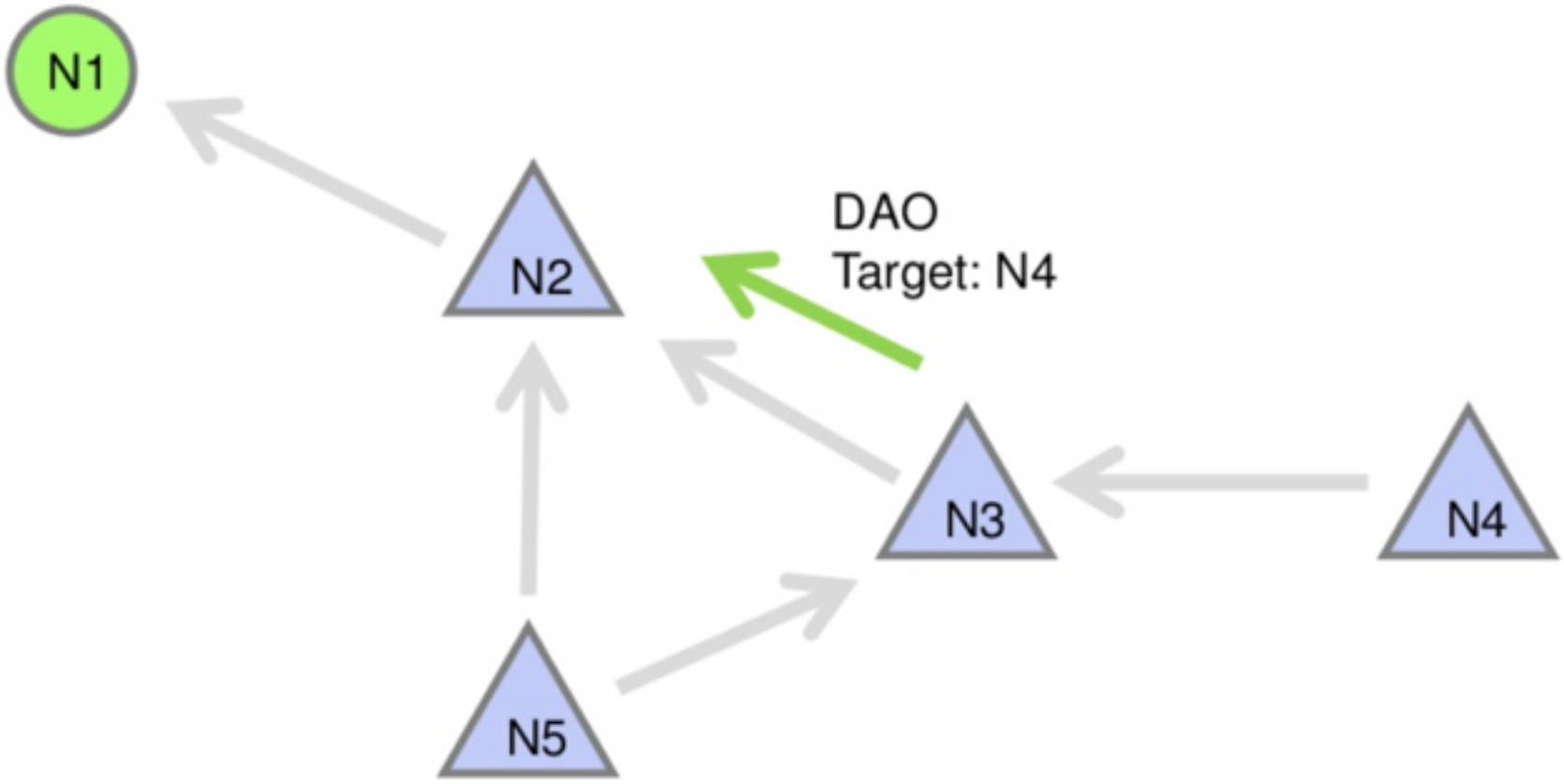
RPL downward routes

- 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

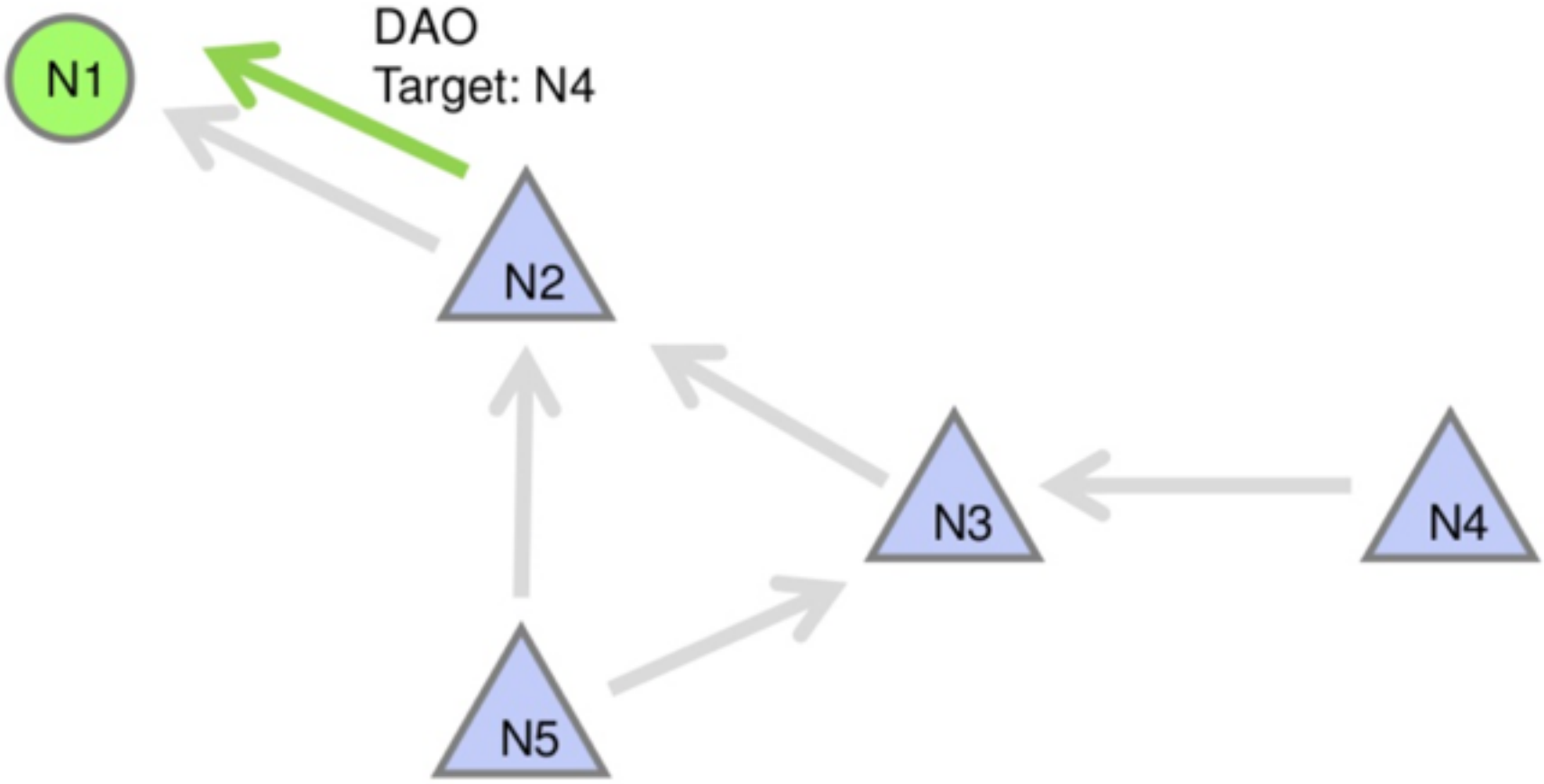
RPL downward routes



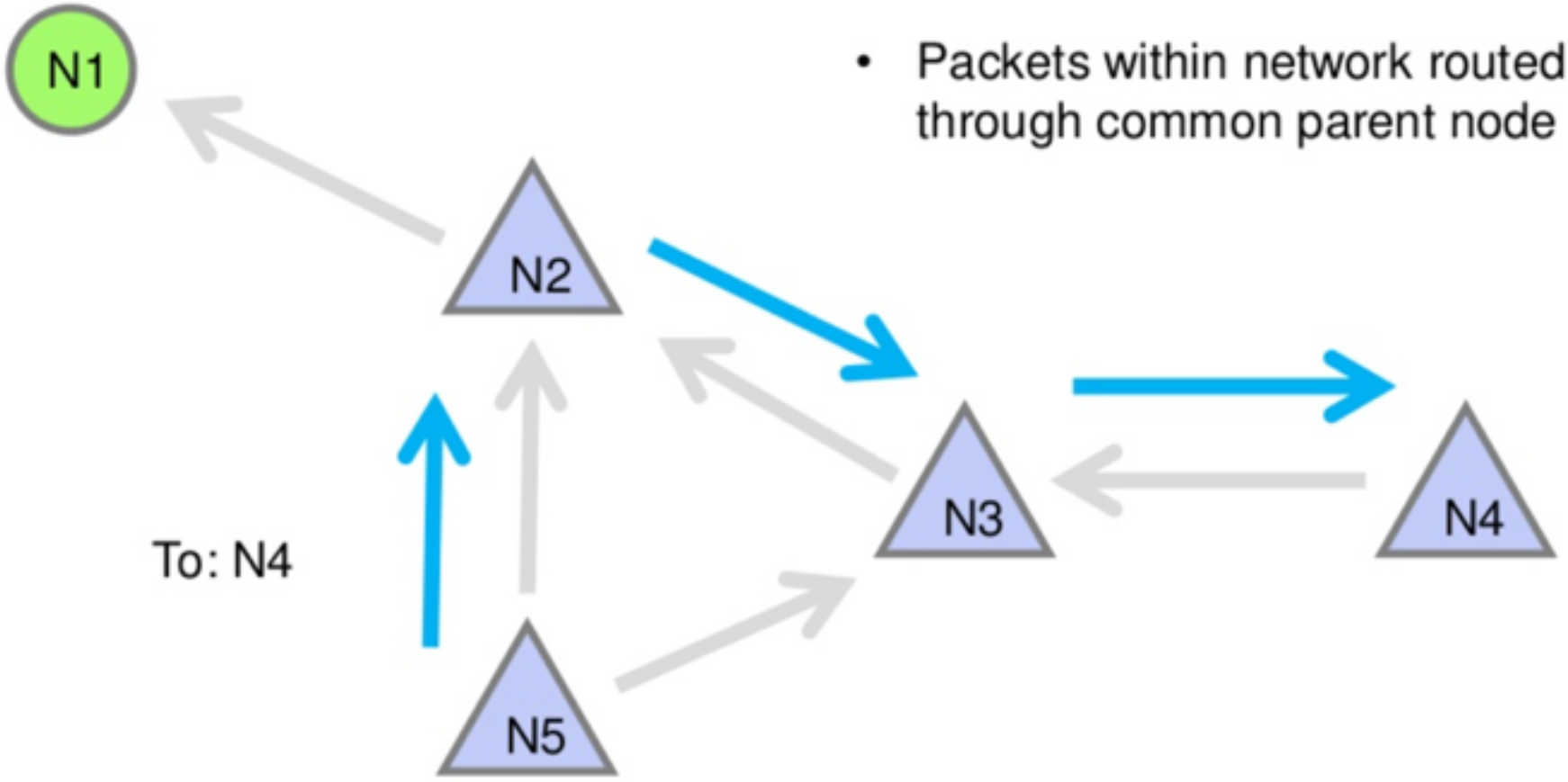
Node IPv6 RPL Network Formation



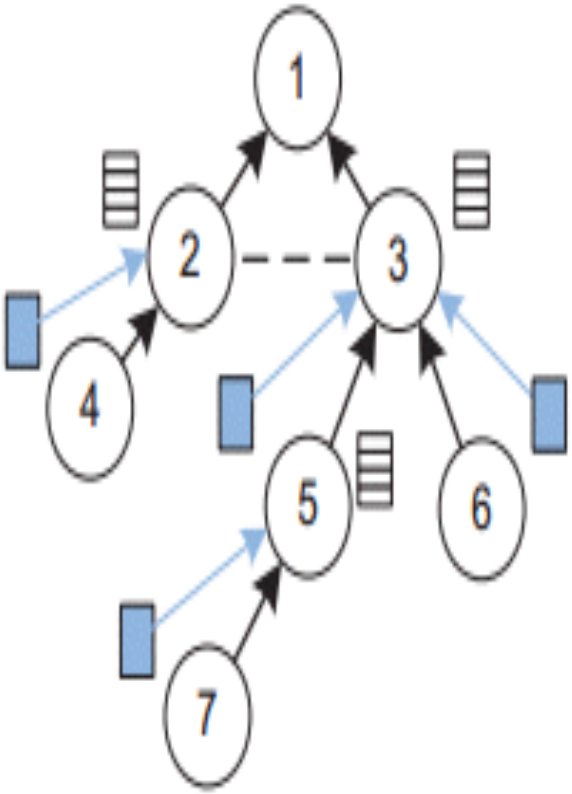
Node IPv6 RPL Network Formation



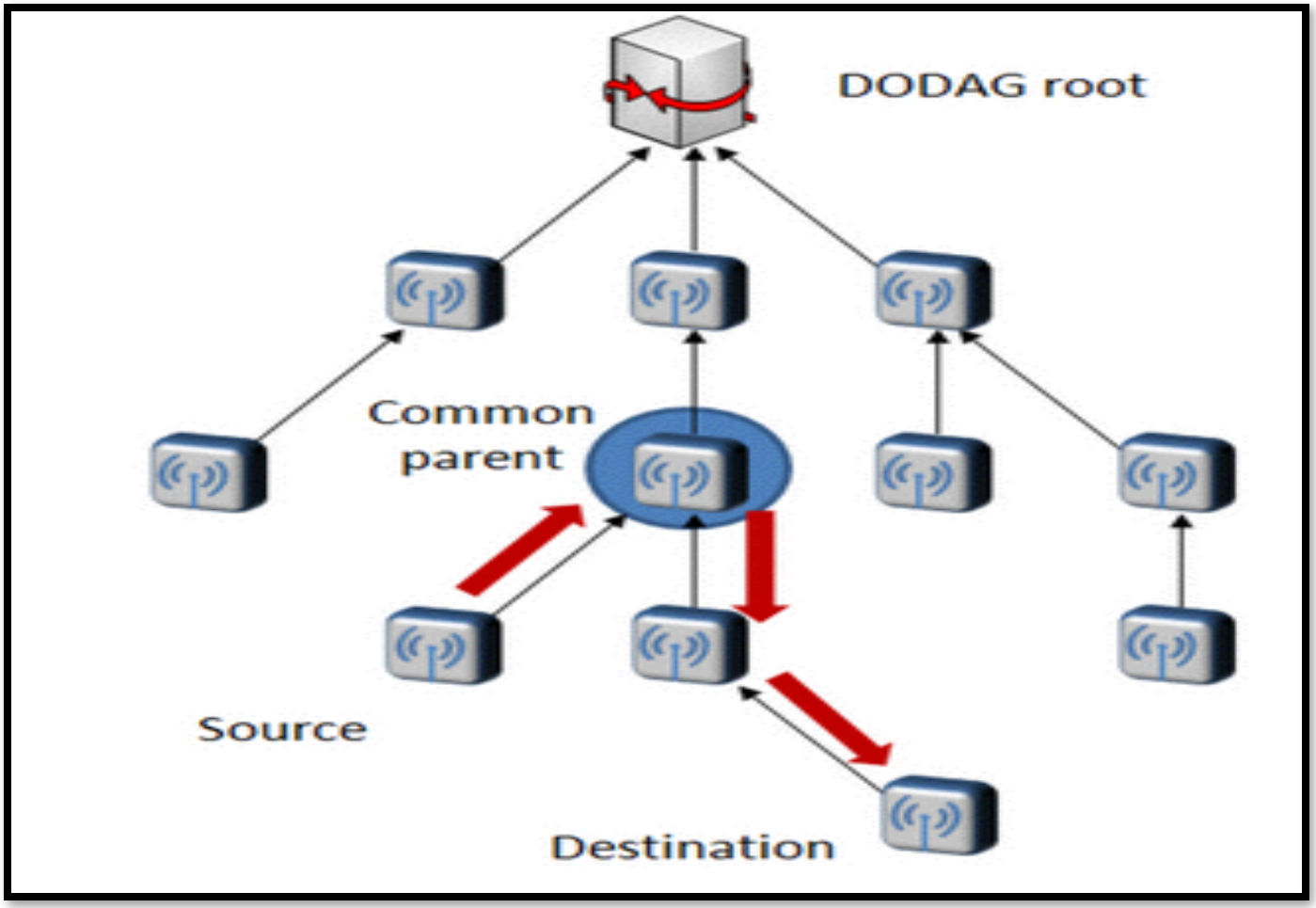
RPL Routing: Down



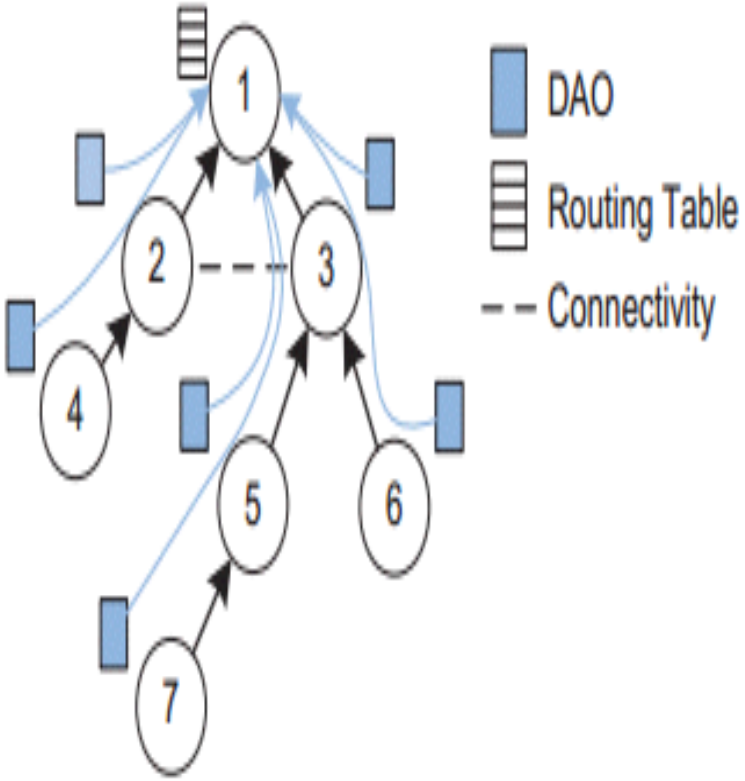
STORING MODE DAO



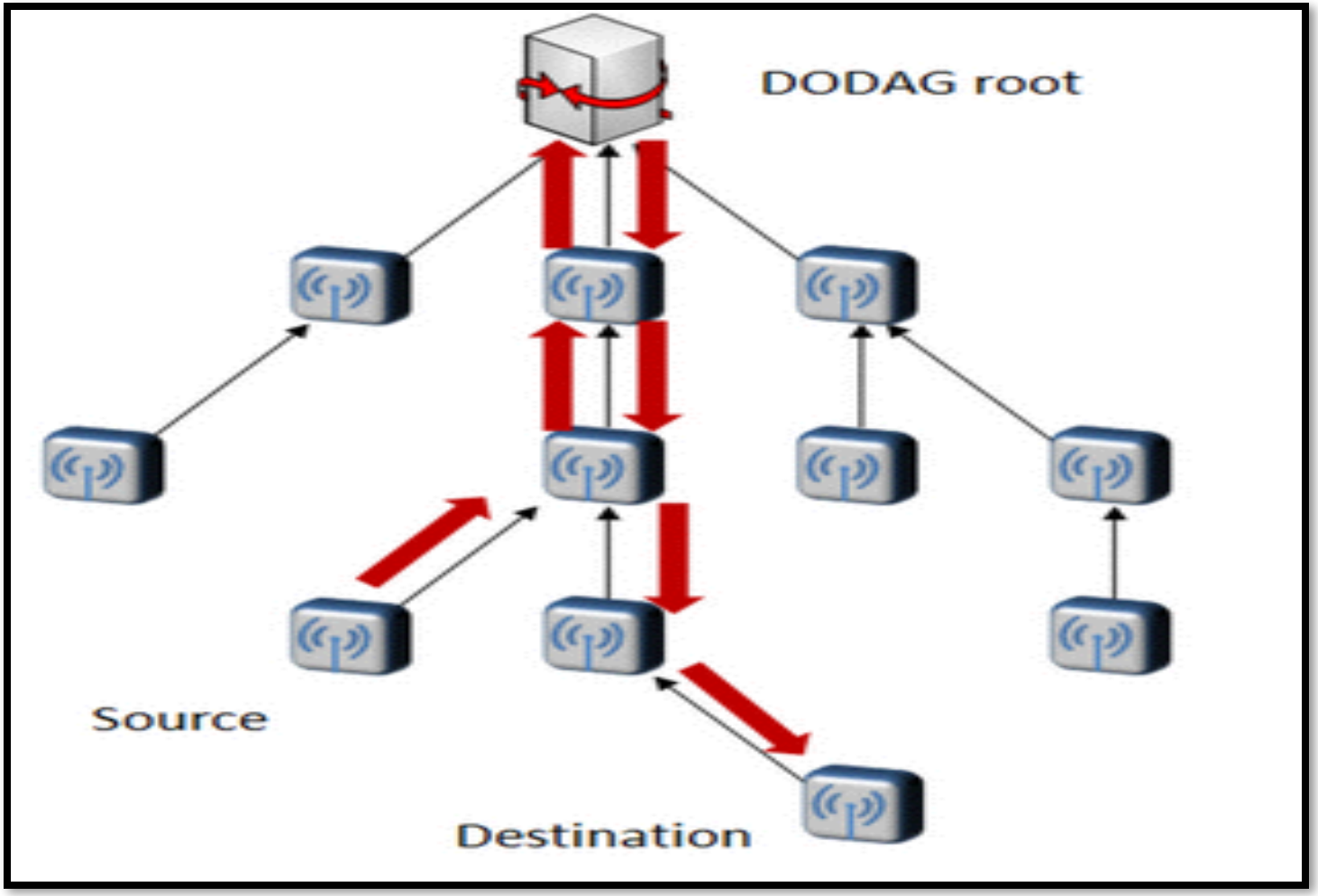
STORING MODE



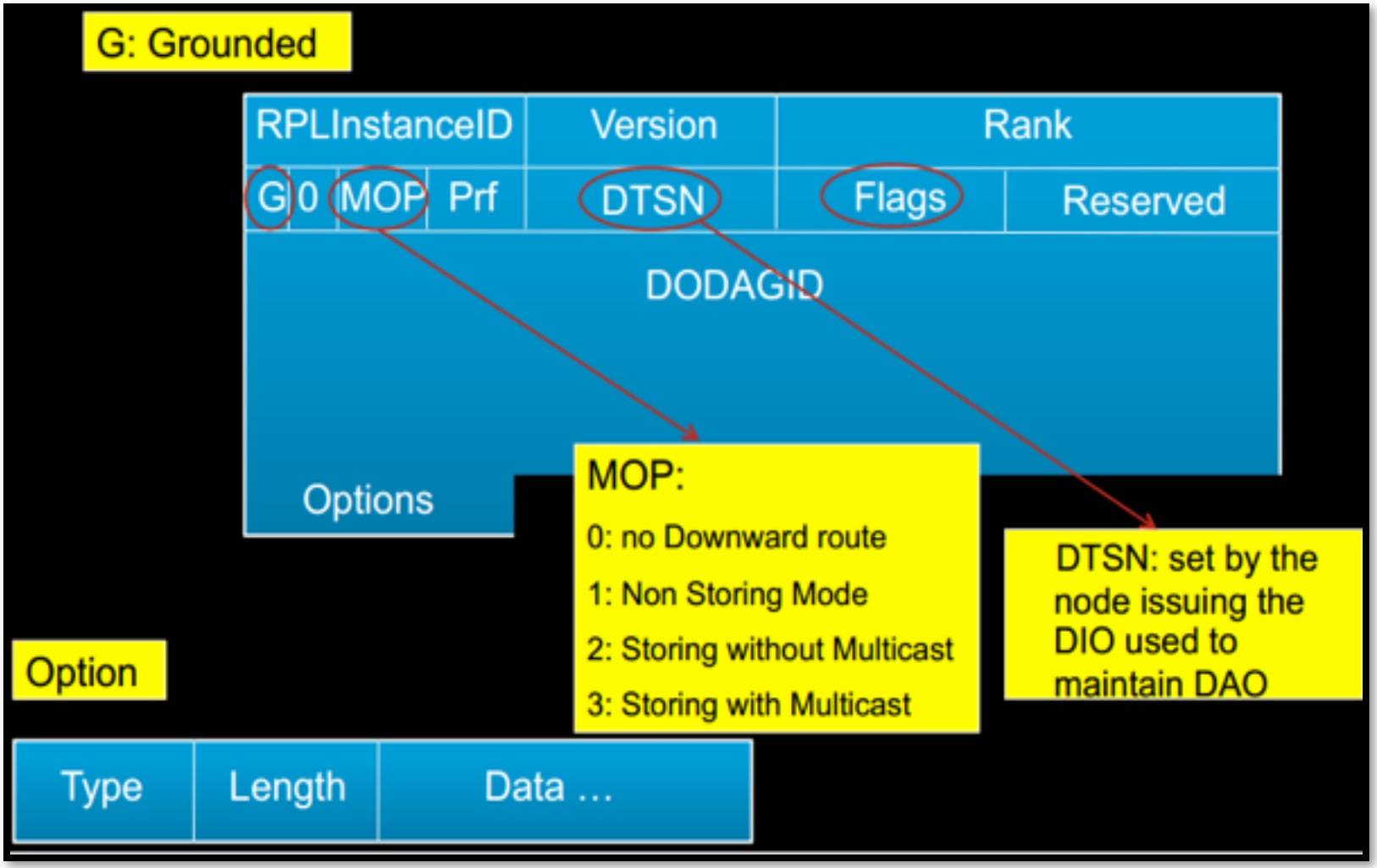
NON-STORING MODE: DAO



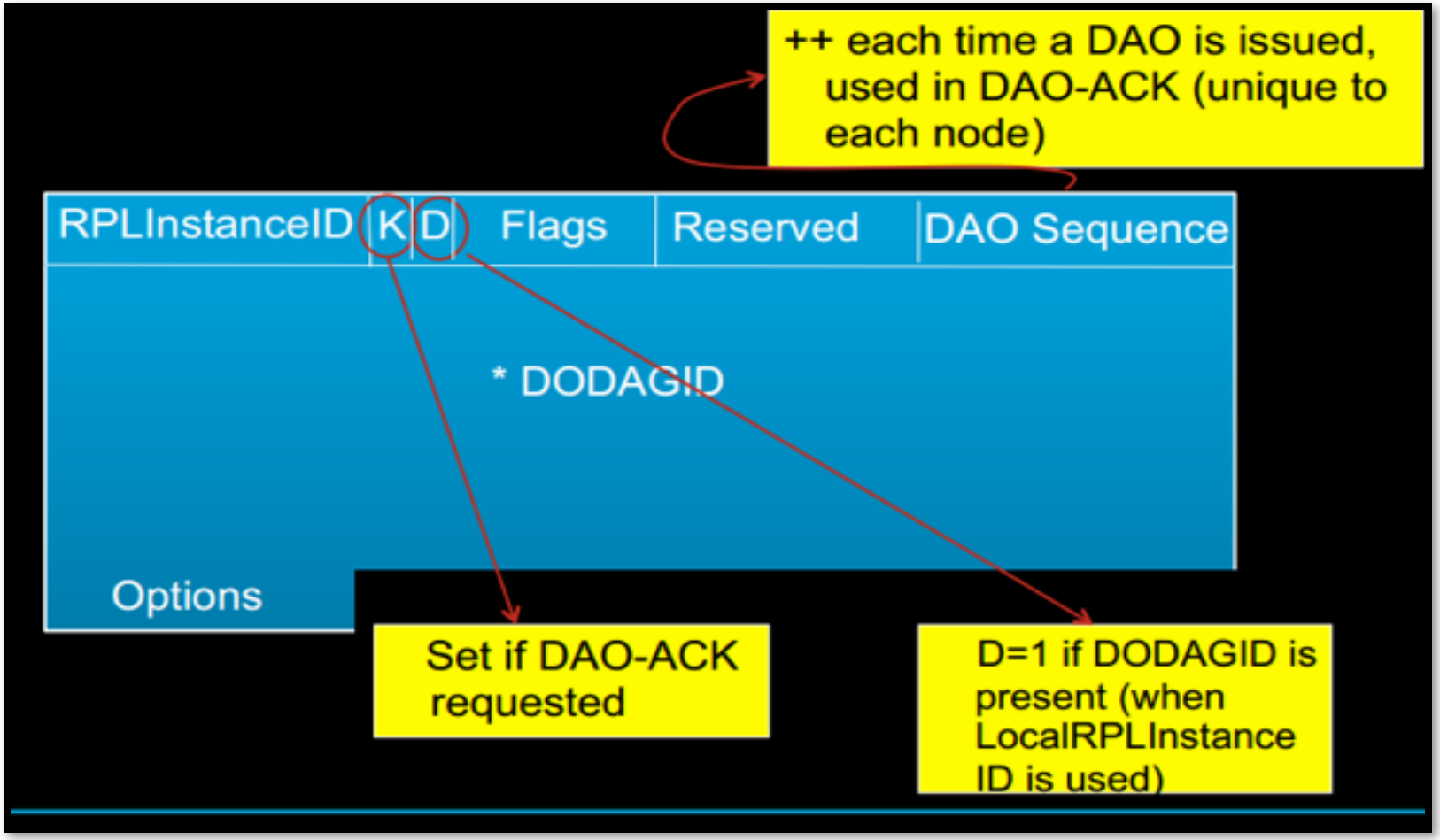
NON-STORING MODE



MESSAGE DIO


G: Grounded
RPLInstanceID
Version
Rank
G
0
MOP
Prf
DTSN
Flags
Reserved
DODAGID
Options
MOP:
0: no Downward route
1: Non Storing Mode
2: Storing without Multicast
3: Storing with Multicast
DTSN: set by the node issuing the DIO used to maintain DAO
Option
Type
Length
Data …

MESSAGE DAO


++ each time a DAO is issued, used in DAO-ACK (unique to each node)
RPLInstanceID
K
D
Flags
Reserved
DAO Sequence
* DODAGID
Options
Set if DAO-ACK requested
D=1 if DODAGID is present (when LocalRPLInstance ID is used)

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)