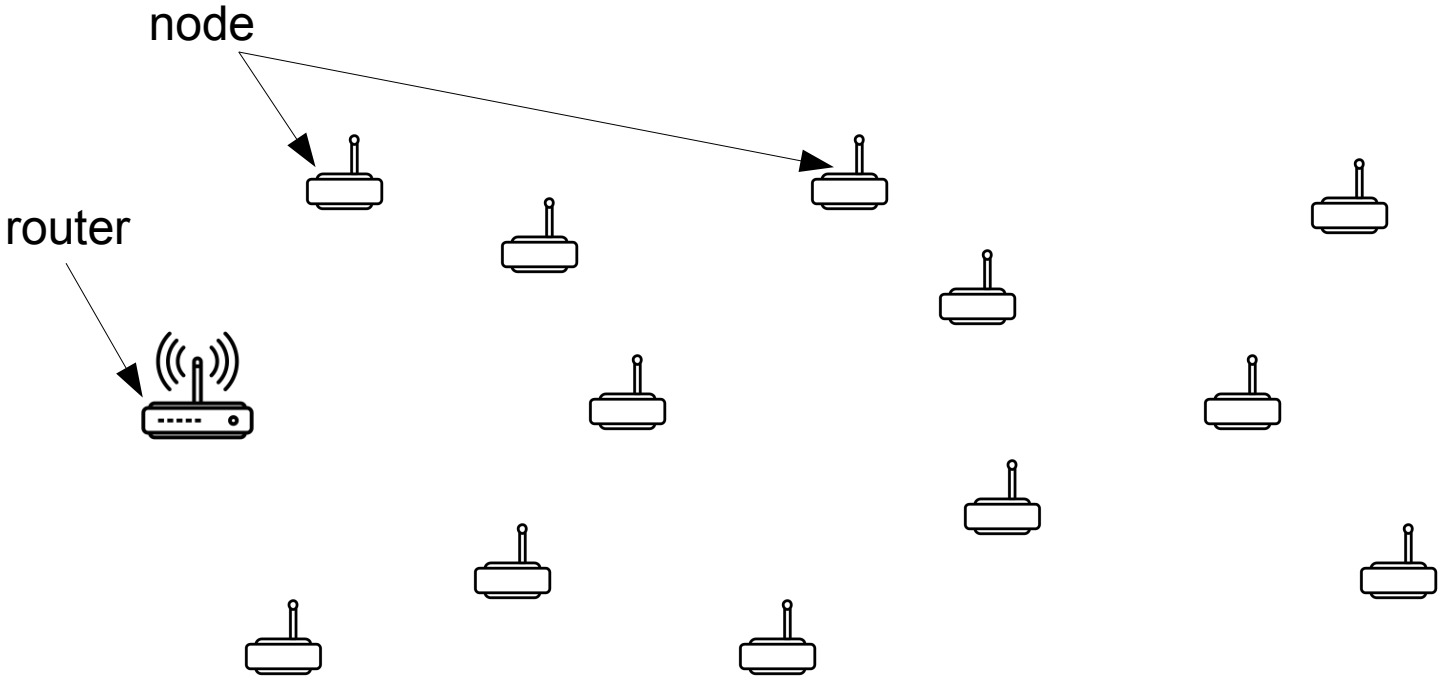
On Designing Provably Correct DODAG Formation Criteria for the IPv6 Routing Protocol for Low-Power and Lossy Networks (RPL)

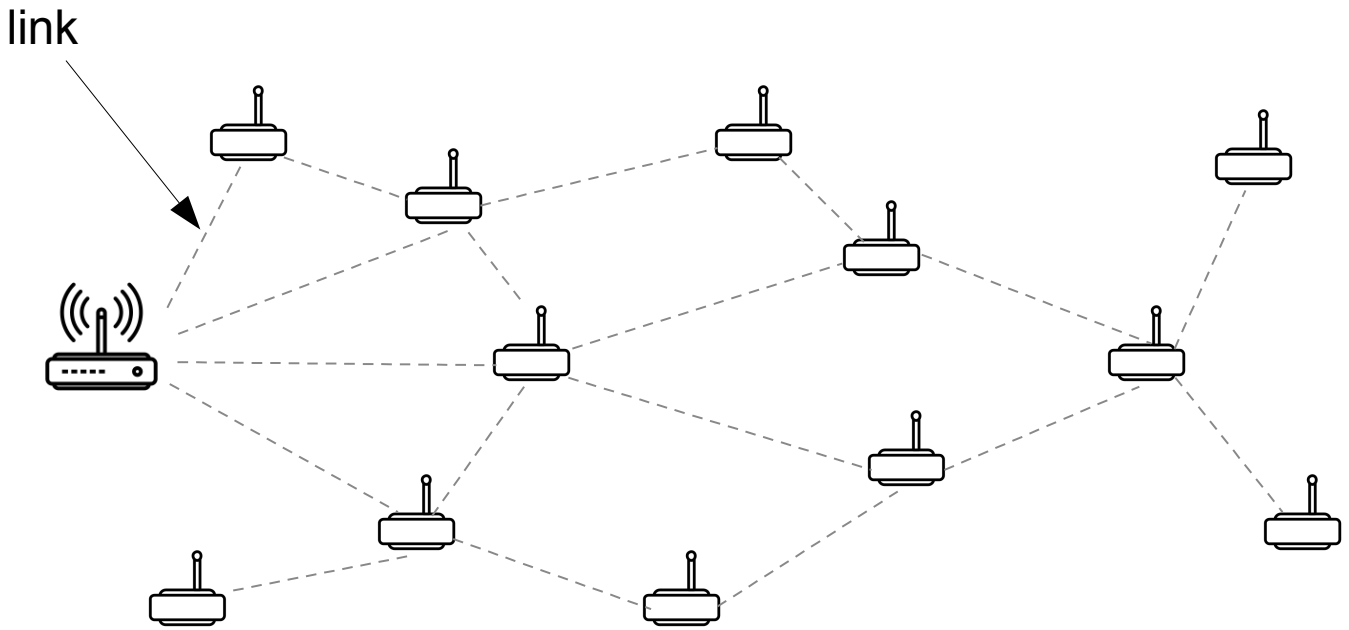
Agnieszka Paszkowska, Konrad Iwanicki
University of Warsaw

DCOSS 2018, New York, NY, USA, June 18, 2018

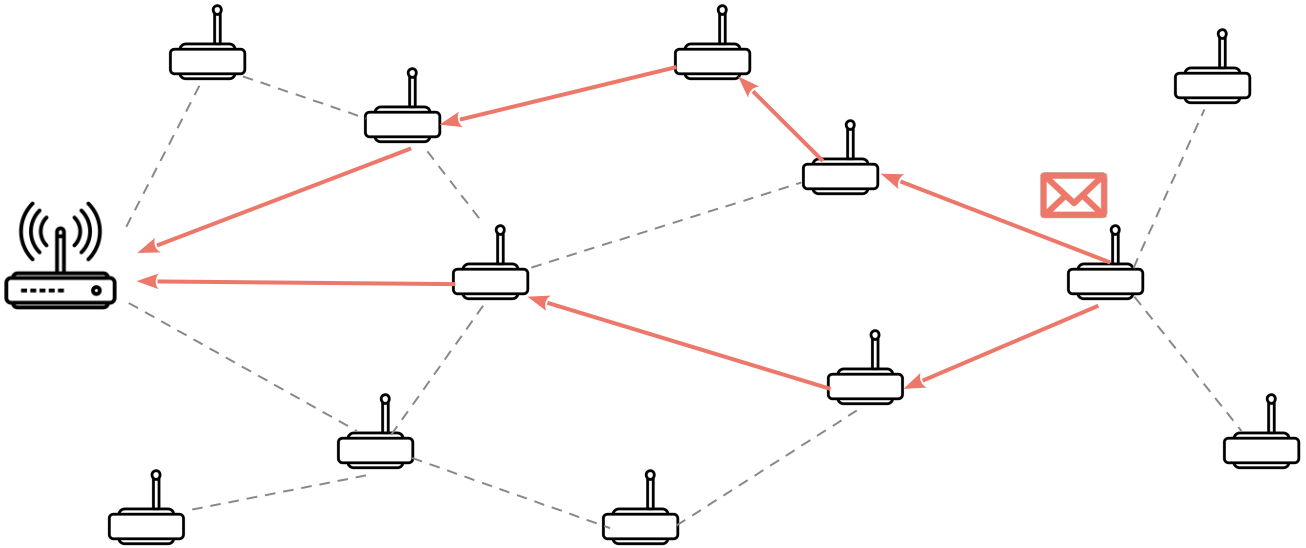
Low Power and Lossy Network



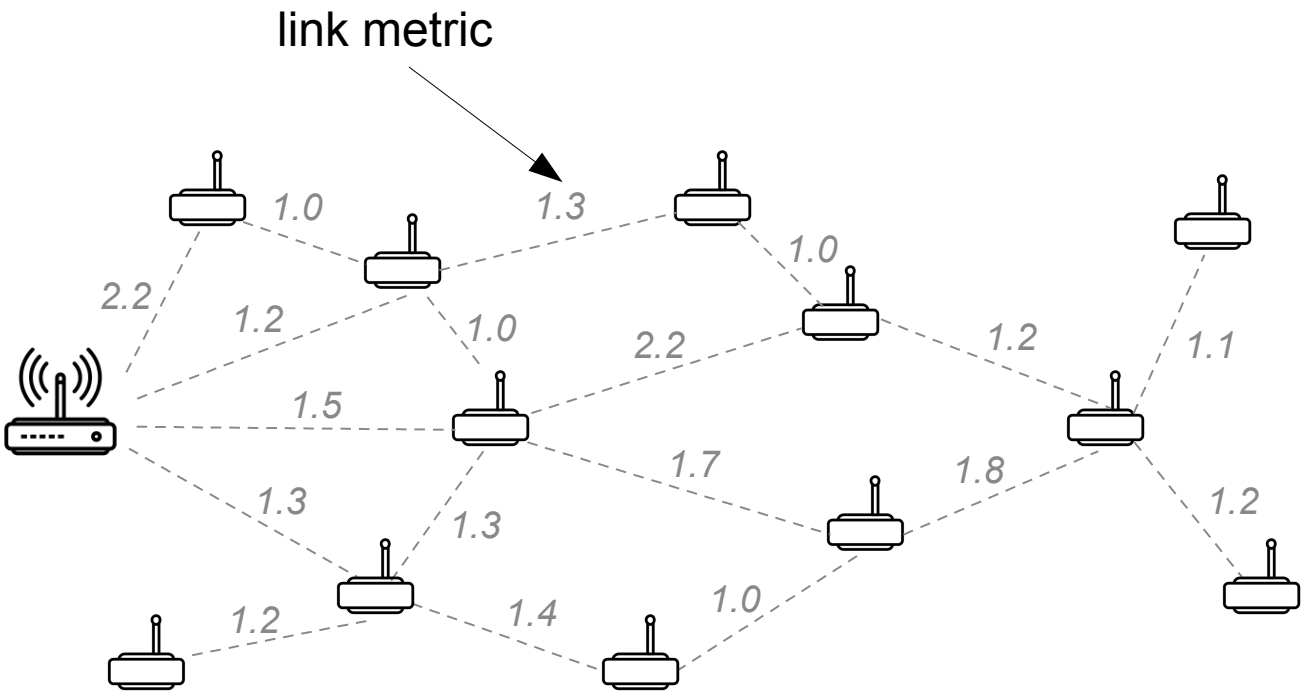
Low Power and Lossy Network



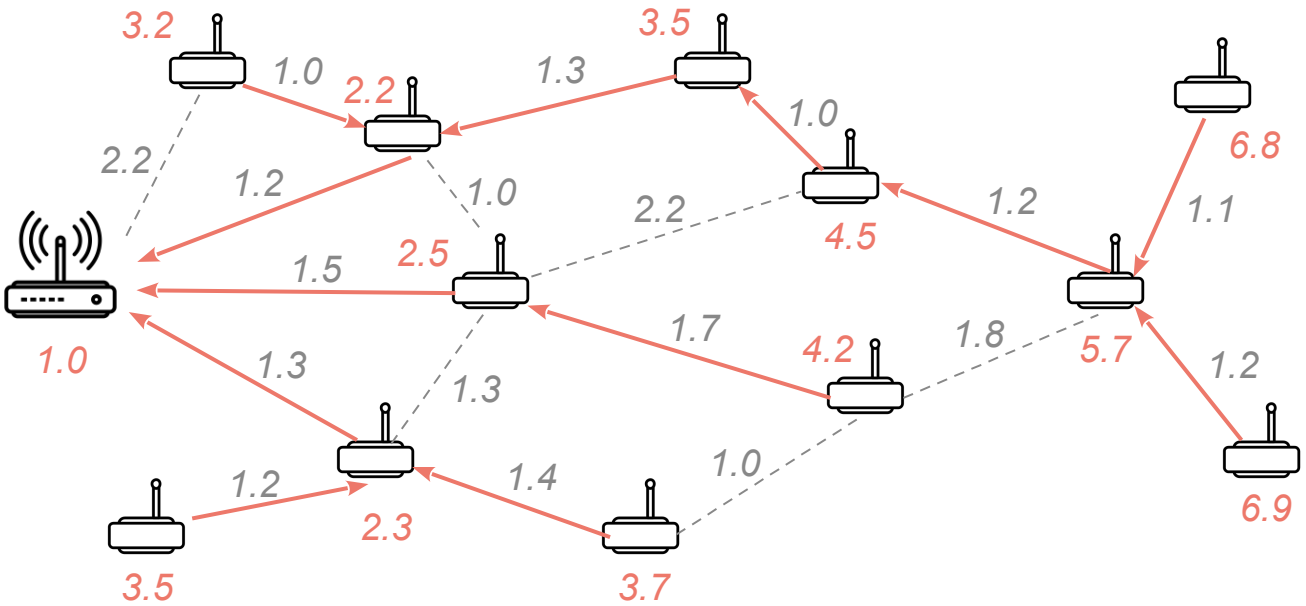
Routing



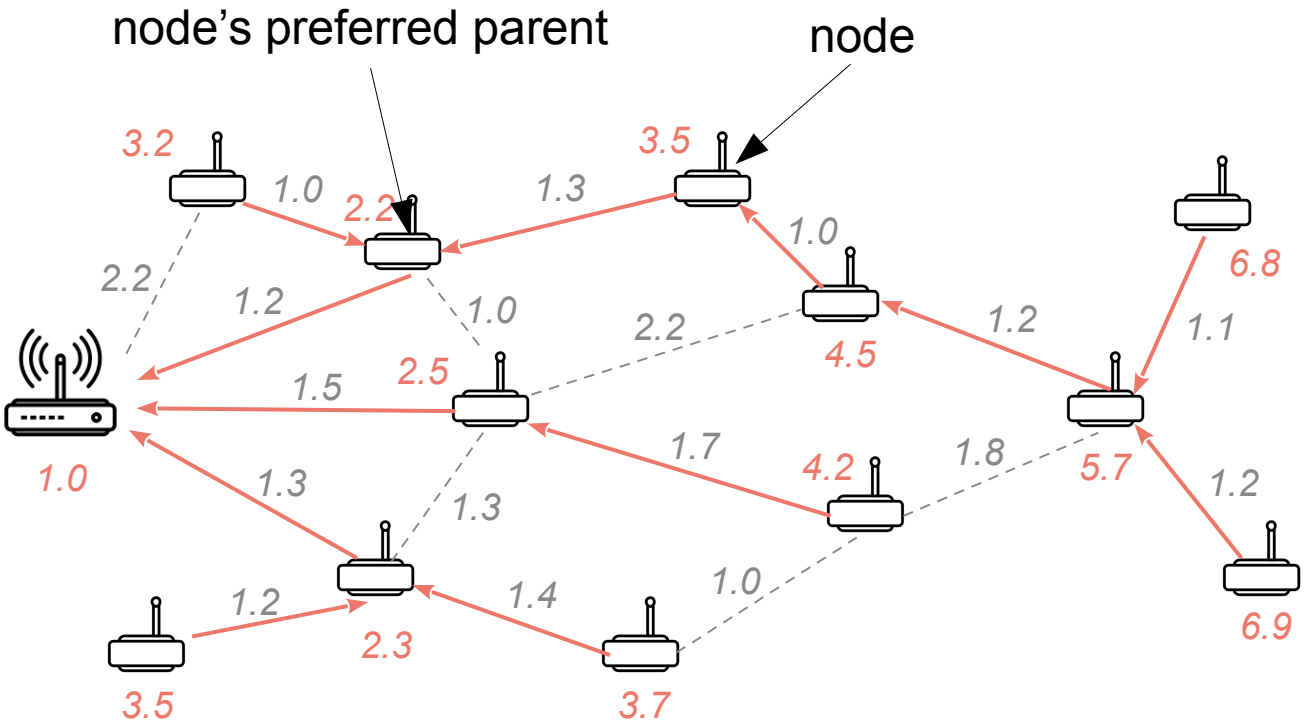
Link Metrics



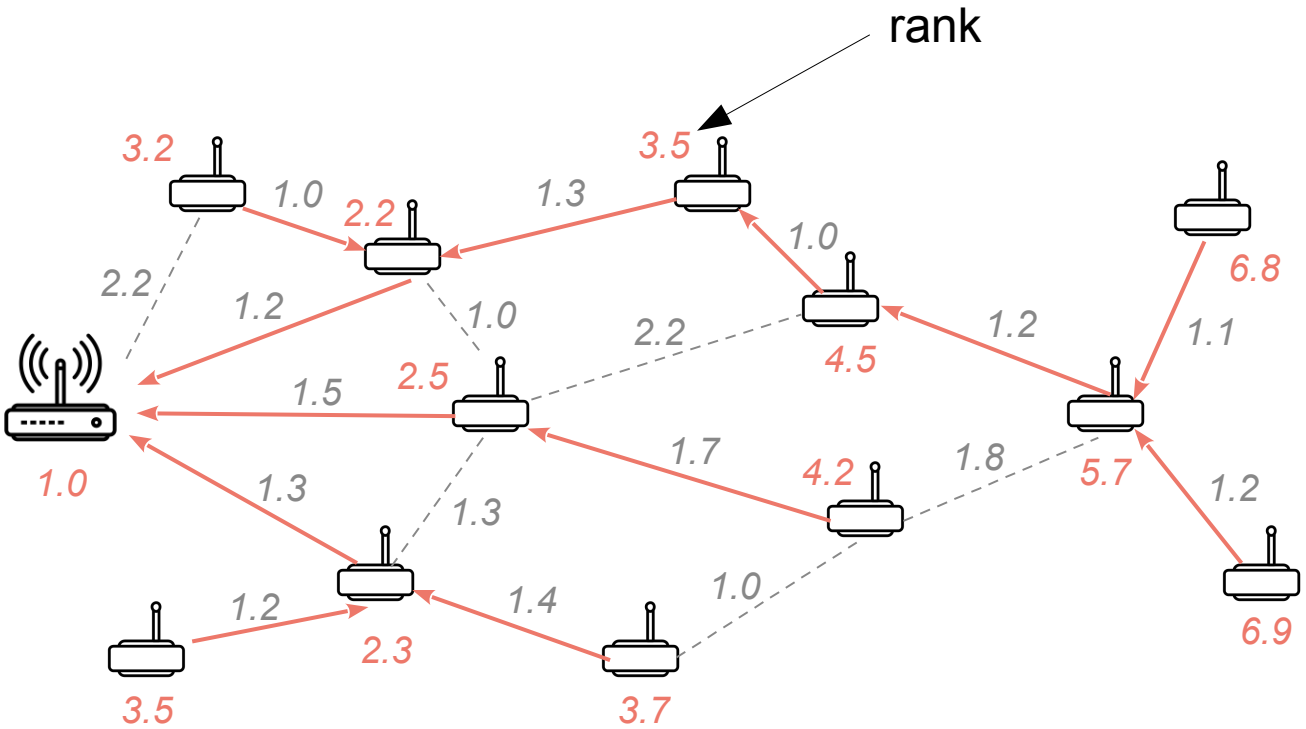
DODAG


3.2
1.0
2.2
3.5
1.3
1.0
6.8
2.2
1.2
1.0
2.2
1.2
1.1
1.5
2.5
4.5
1.0
1.7
4.2
1.8
5.7
1.3
1.3
1.2
1.2
1.4
1.0
2.3
6.9
3.5
3.7

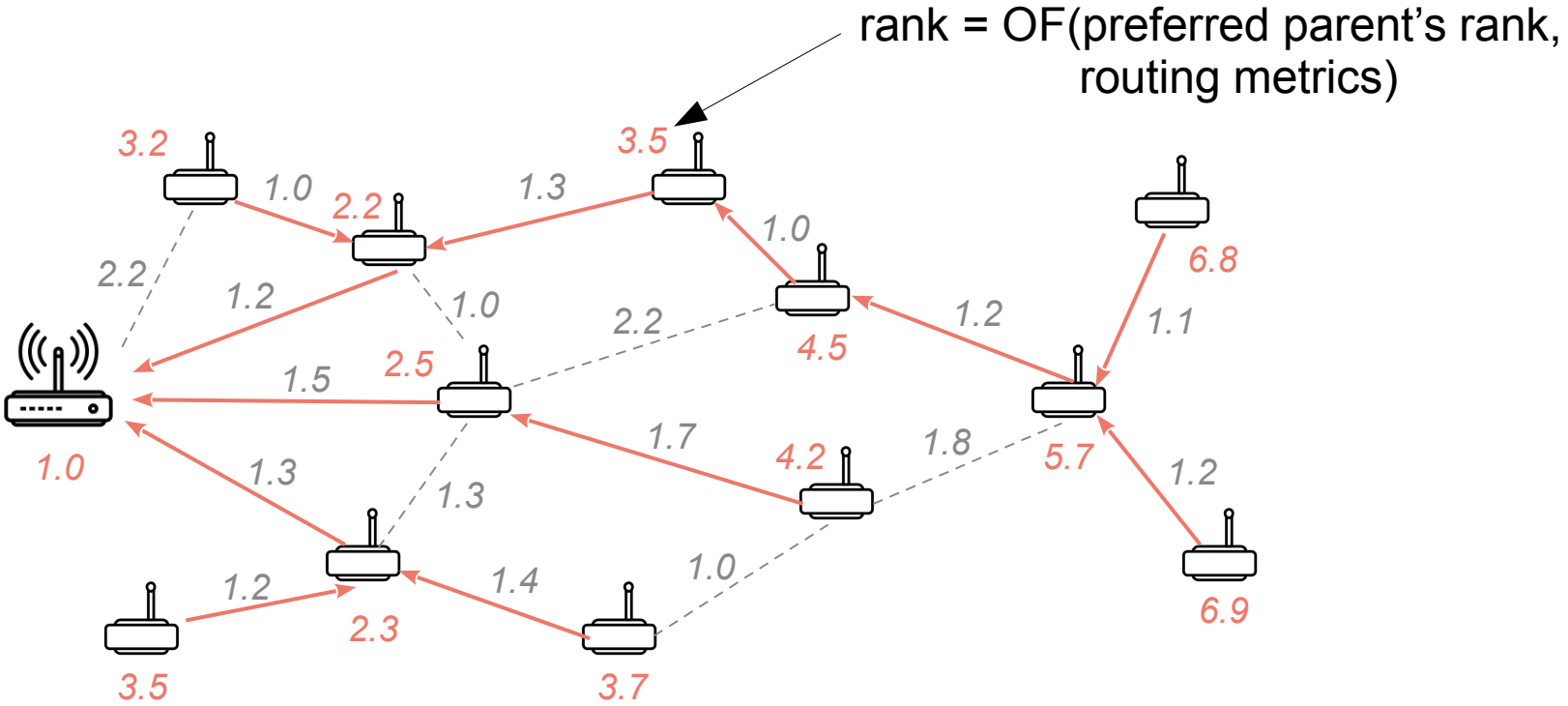
DODAG


node's preferred parent
node
3.2
1.0
2.2
1.3
3.5
1.0
6.8
2.2
1.2
1.0
2.2
4.5
1.2
1.1
1.5
2.5
1.7
1.8
1.0
4.2
5.7
1.3
1.3
1.2
1.2
1.4
1.0
2.3
3.5
3.7
6.9

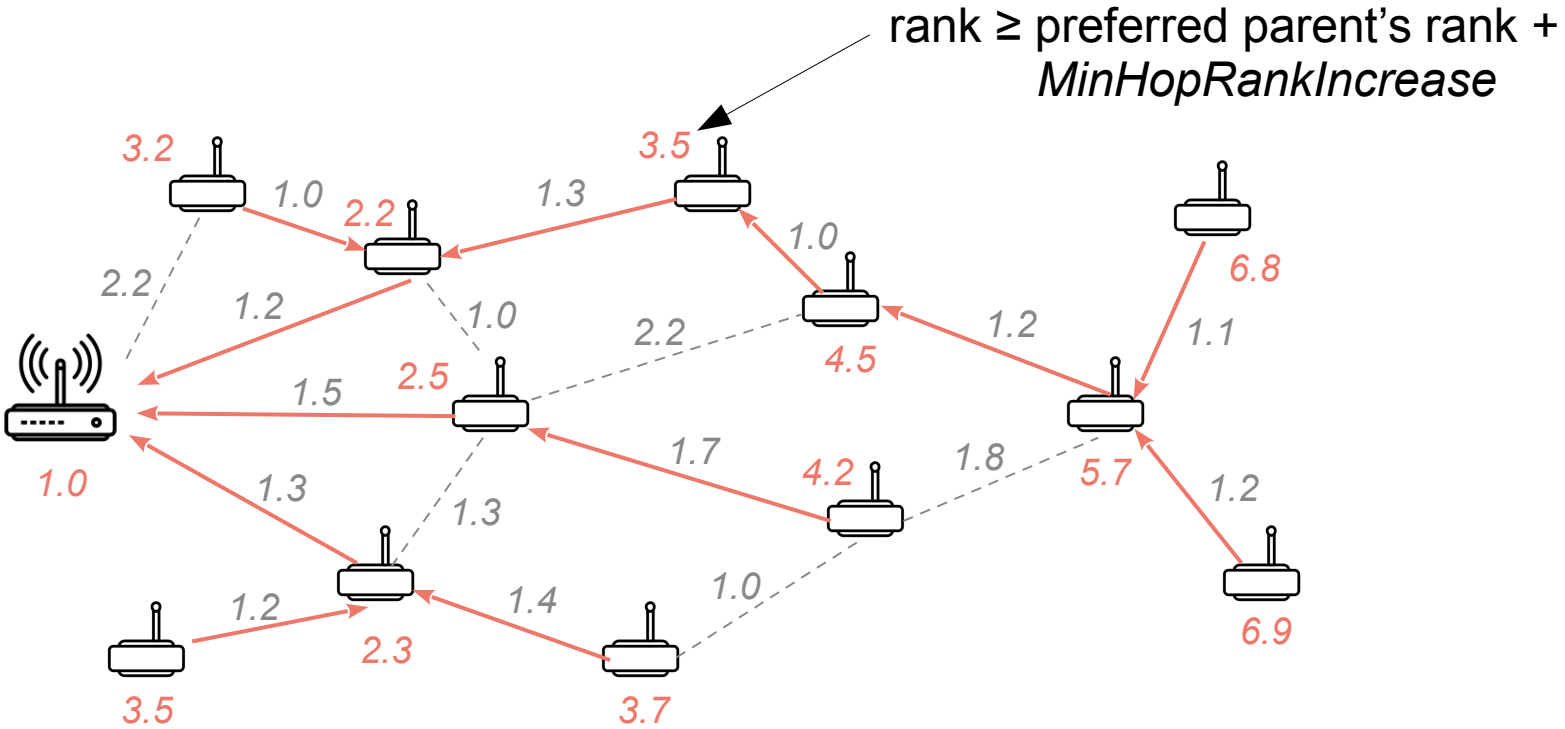
DODAG


rank
3.2
1.0
2.2
3.5
1.3
1.0
6.8
2.2
1.2
1.0
2.2
1.2
1.1
1.5
2.5
4.5
1.0
1.7
1.8
1.3
4.2
5.7
1.2
1.3
1.2
1.4
1.0
2.3
3.5
3.7
6.9

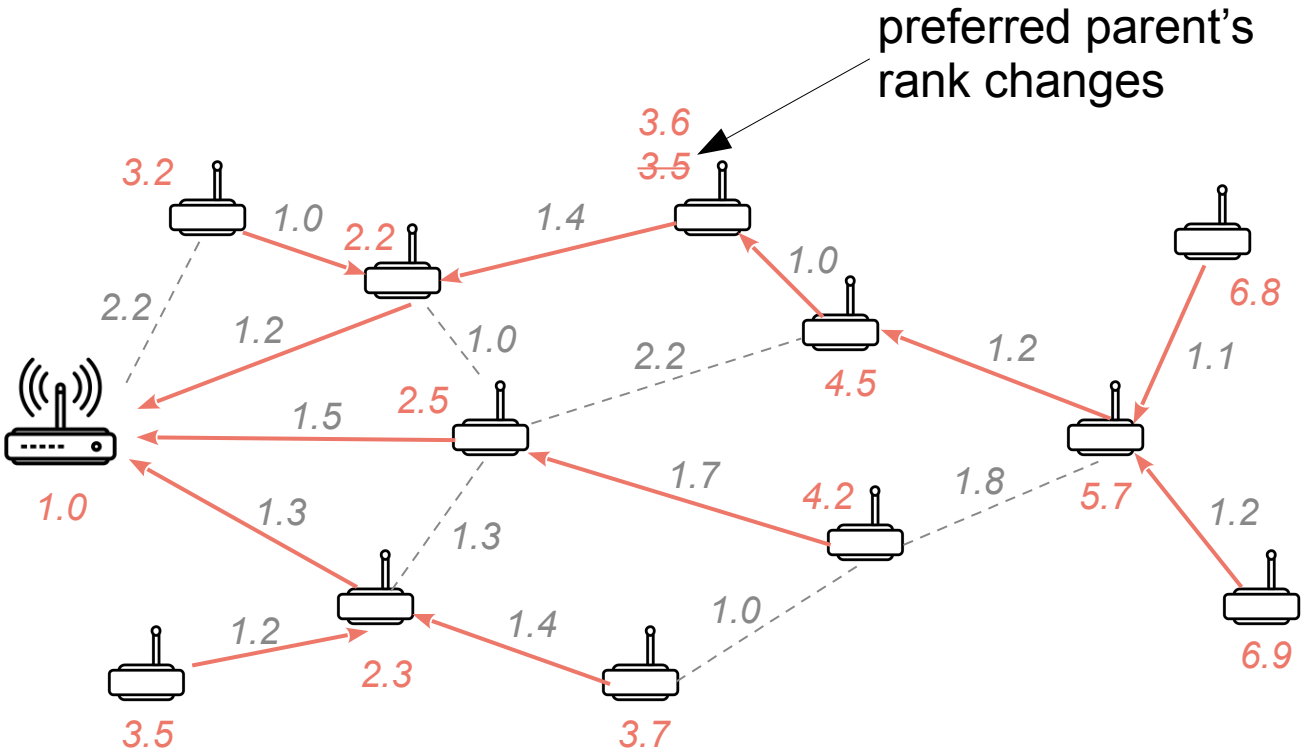
Objective Function



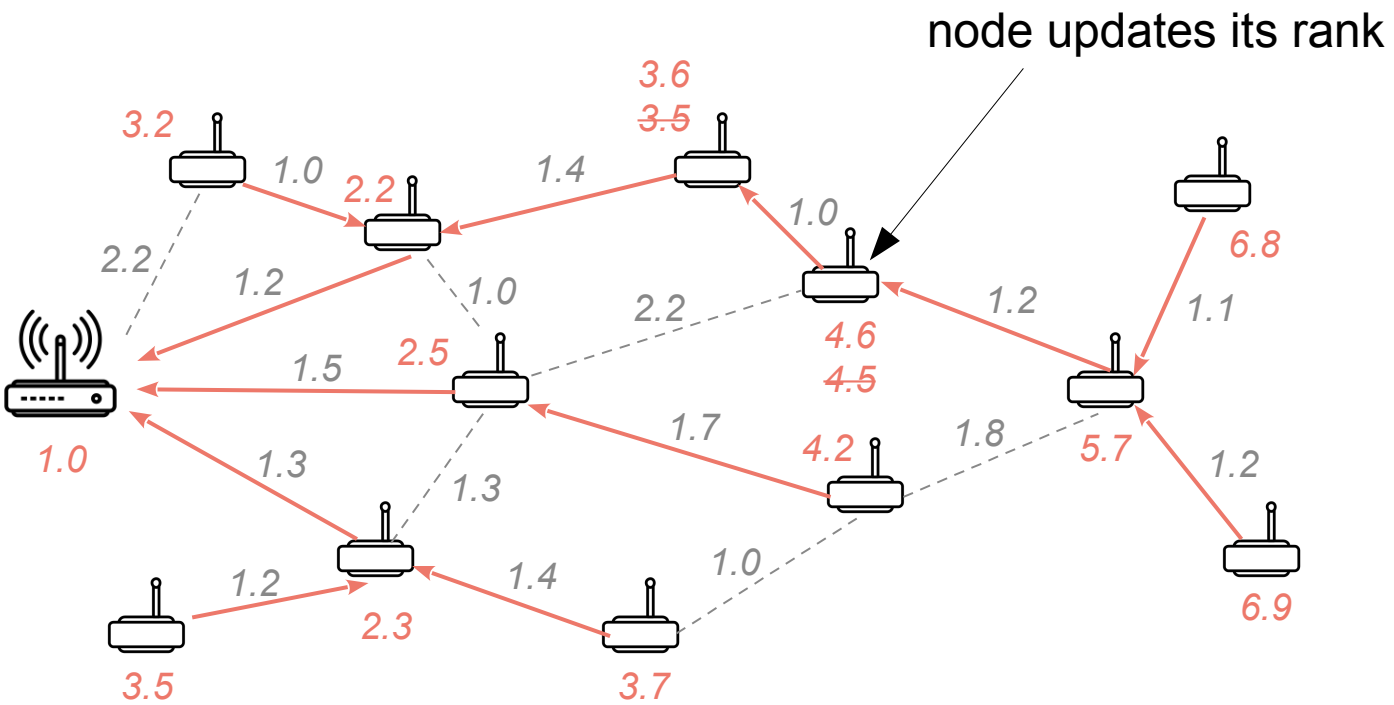
MinHopRankIncrease



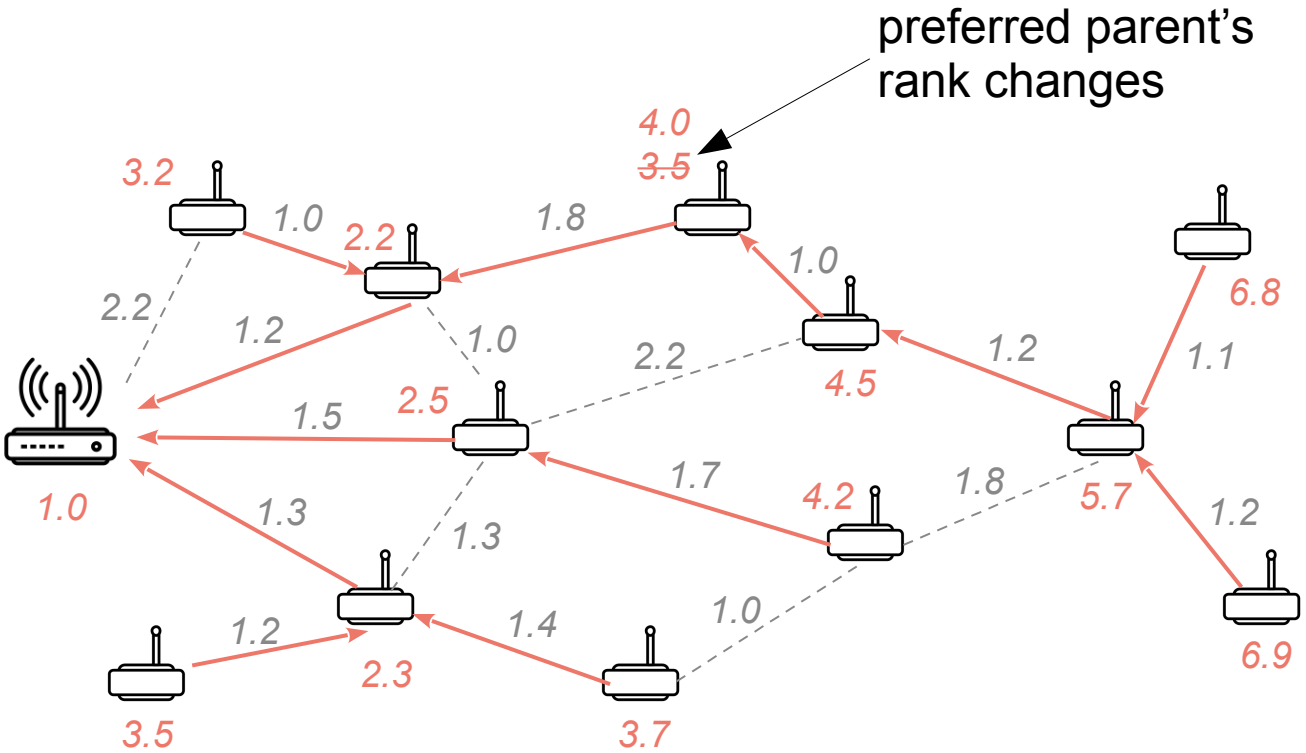
Rank Change


preferred parent's
rank changes
3.6
3.5
3.2
1.0
2.2
1.4
1.0
6.8
2.2
1.2
1.0
2.2
1.2
1.1
4.5
2.5
1.5
1.7
1.8
1.0
1.3
1.3
4.2
5.7
1.2
1.2
1.4
1.0
2.3
6.9
3.5
3.7

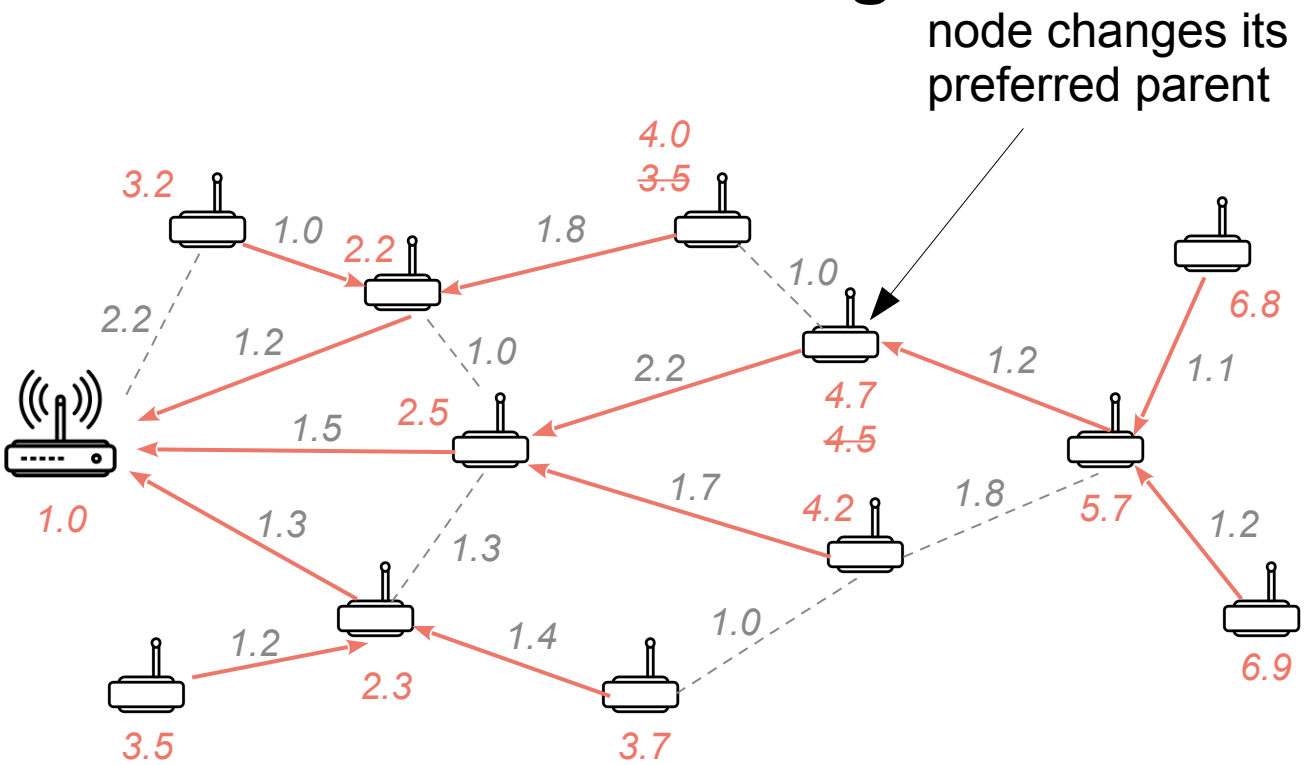
Rank Change


node updates its rank
3.6
3.5
3.2
1.0
2.2
1.4
1.0
2.2
1.2
1.0
2.2
6.8
1.1
1.2
4.6
4.5
1.5
2.5
1.7
1.8
4.2
5.7
1.0
1.3
1.3
1.2
1.2
1.4
1.0
2.3
3.5
3.7
6.9

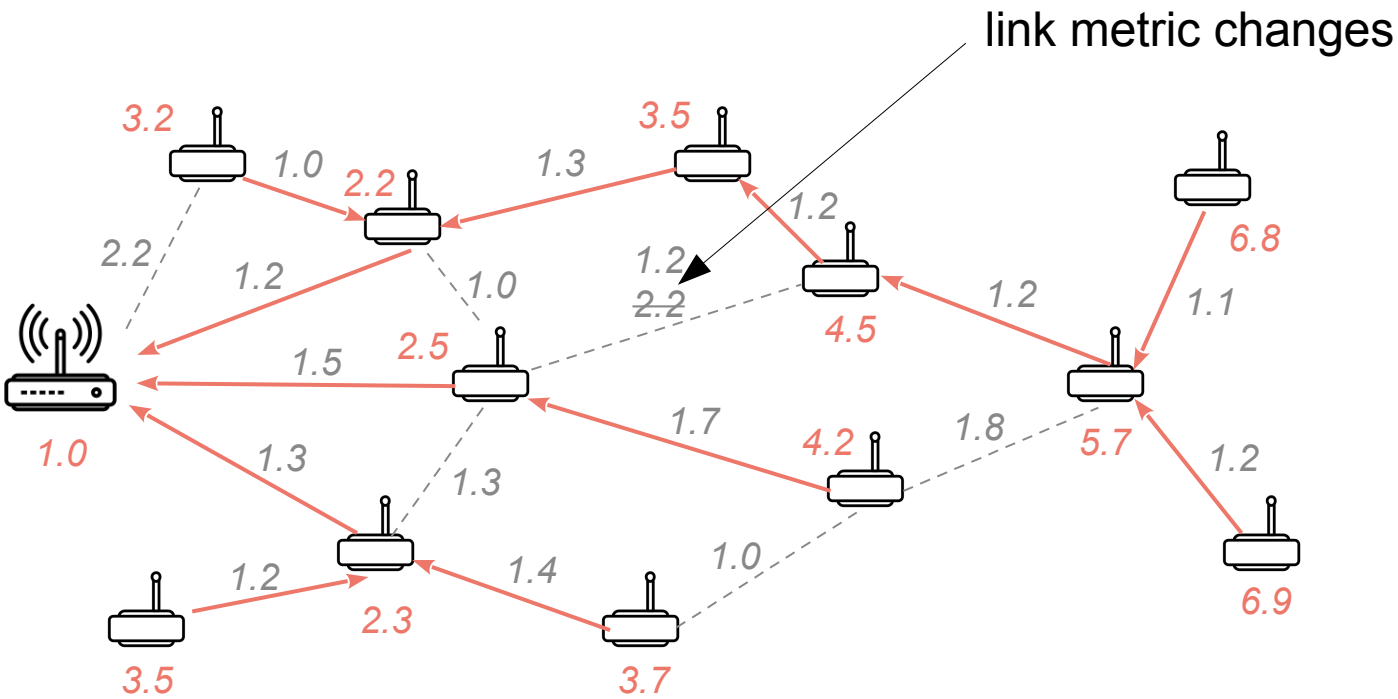
Rank Change


preferred parent's
rank changes
4.0
3.5
3.2
1.0
2.2
1.8
1.0
6.8
2.2
1.2
1.0
2.2
1.2
1.1
4.5
1.5
2.5
1.7
1.0
1.3
1.3
4.2
1.8
5.7
1.2
1.2
1.4
1.0
2.3
6.9
3.5
3.7

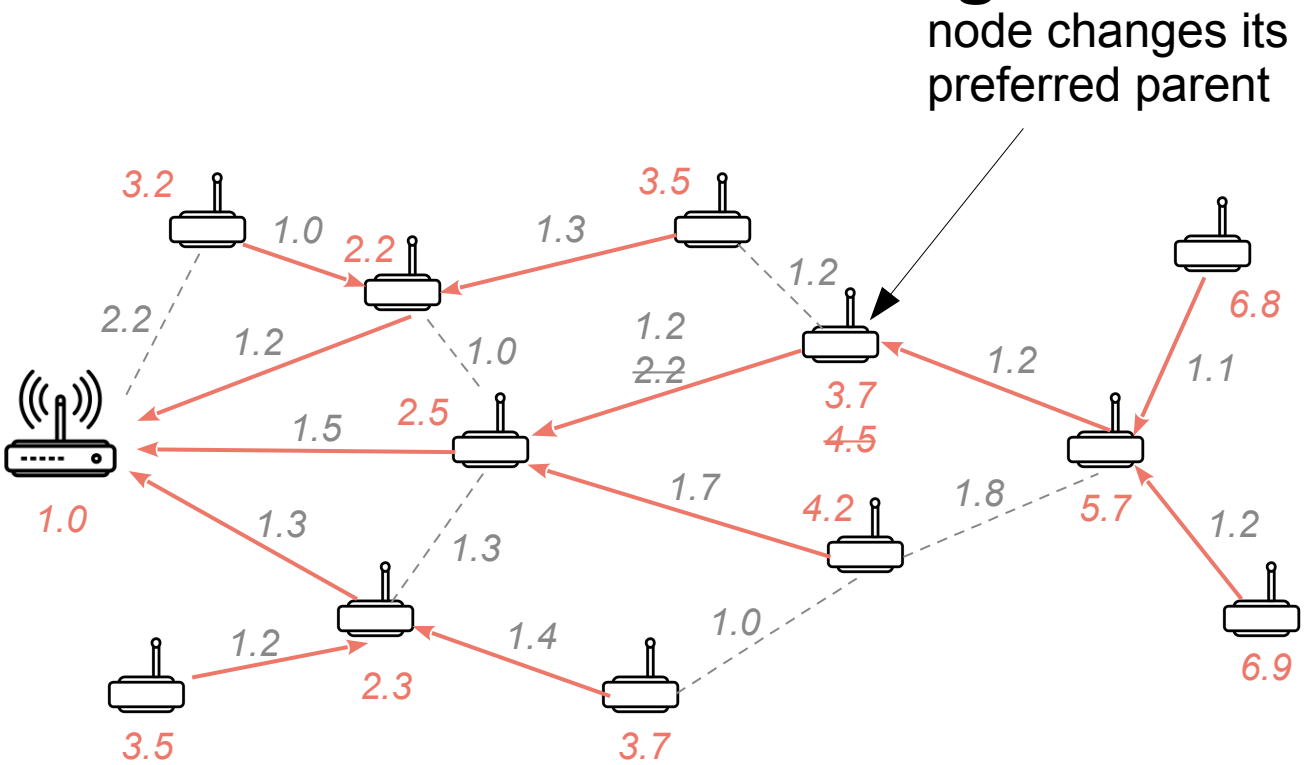
Rank Change


node changes its
preferred parent
4.0
3.5
3.2
1.0
2.2
1.8
1.0
6.8
2.2
1.2
1.0
2.2
1.2
1.1
4.7
4.5
2.5
1.5
1.0
1.7
1.8
4.2
5.7
1.3
1.3
1.2
1.2
1.4
1.0
2.3
6.9
3.5
3.7

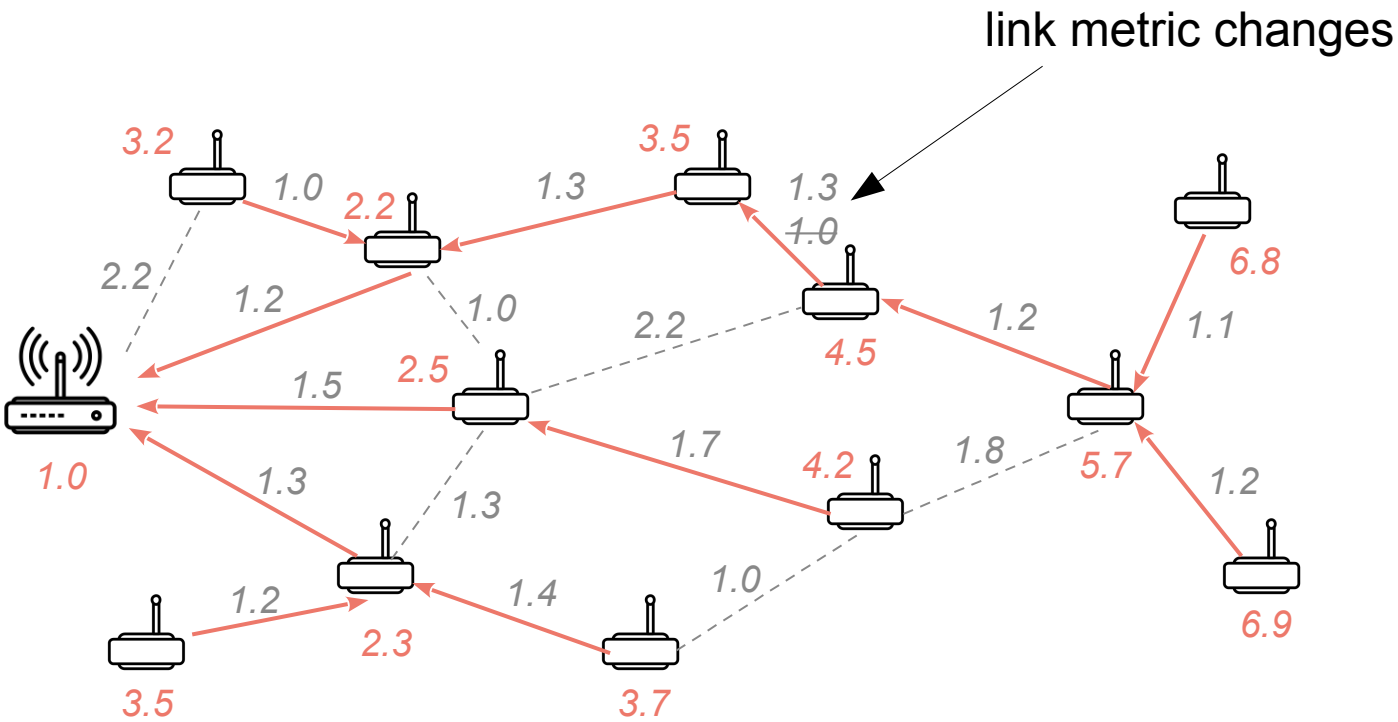
Link Metric Change



Link Metric Change



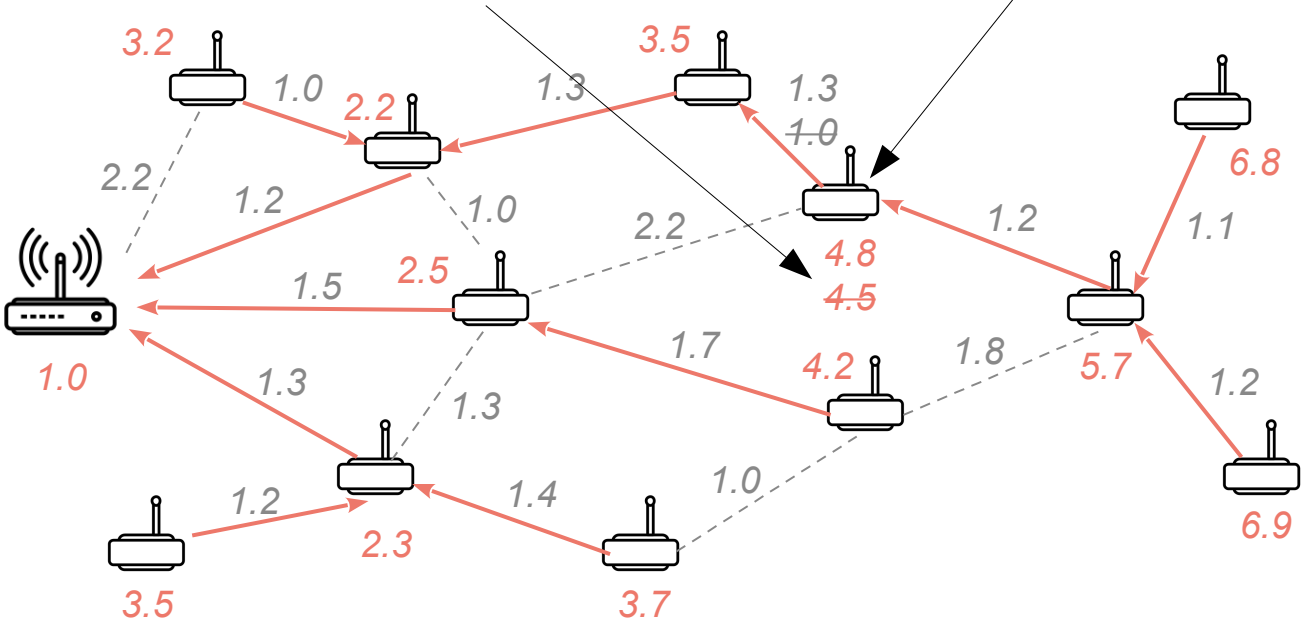
Hysteresis


link metric changes
3.2
1.0
2.2
1.3
3.5
1.3
1.0
6.8
2.2
1.2
1.0
2.2
1.2
1.1
4.5
1.5
2.5
1.7
1.8
1.0
4.2
5.7
1.3
1.3
1.2
1.2
1.4
1.0
2.3
6.9
3.5
3.7

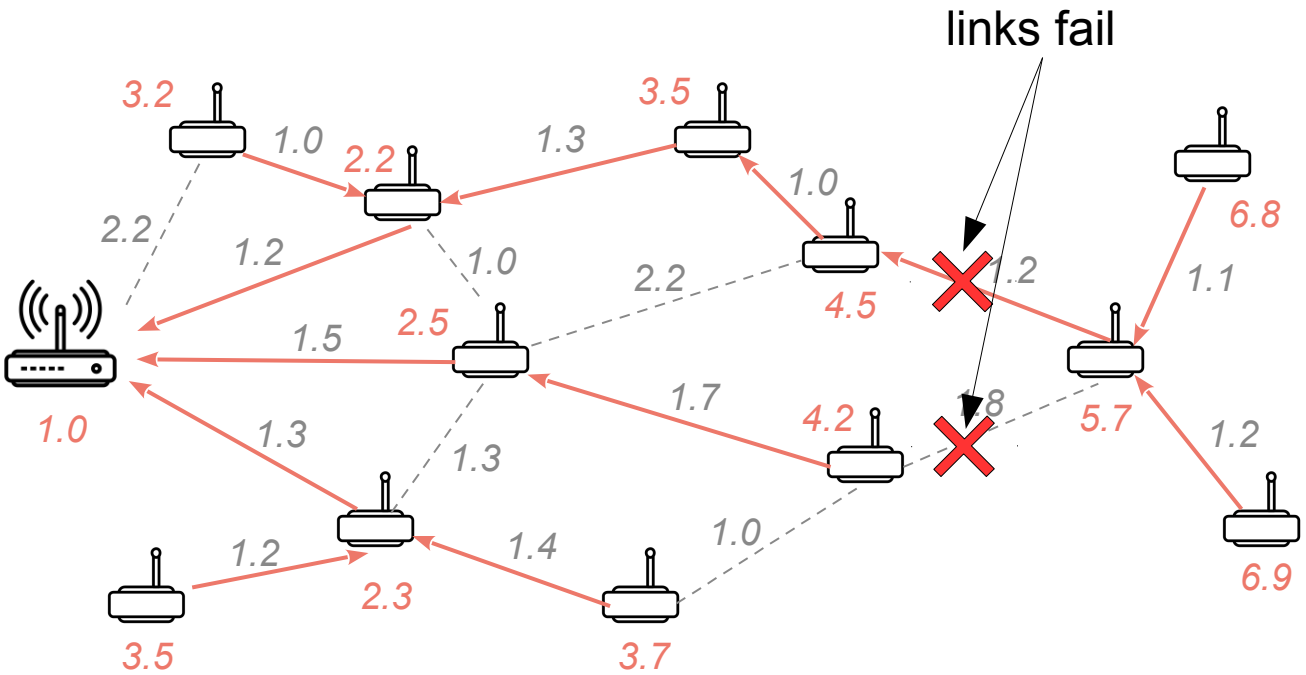
Hysteresis

node's new rank is worse from the best possible rank by < *ParentSwitchThr*

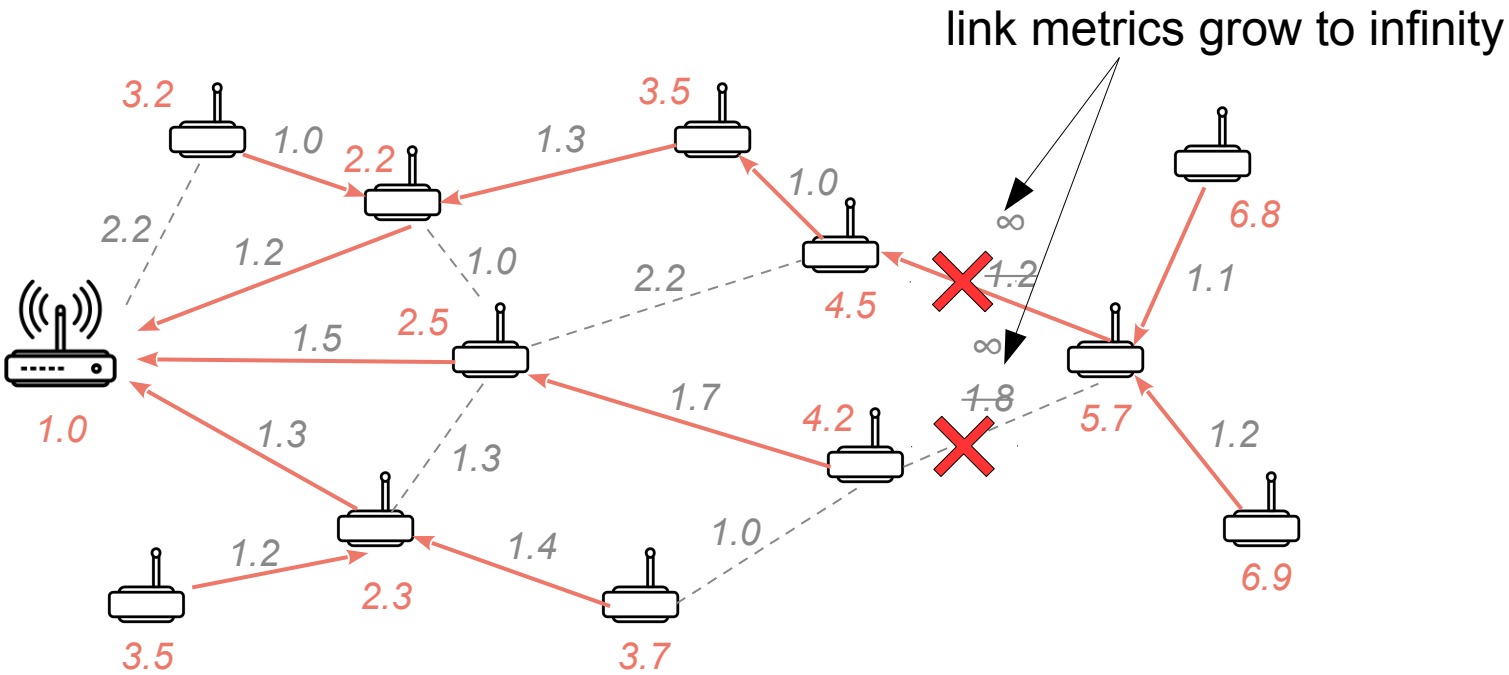
node does not change its preferred parent



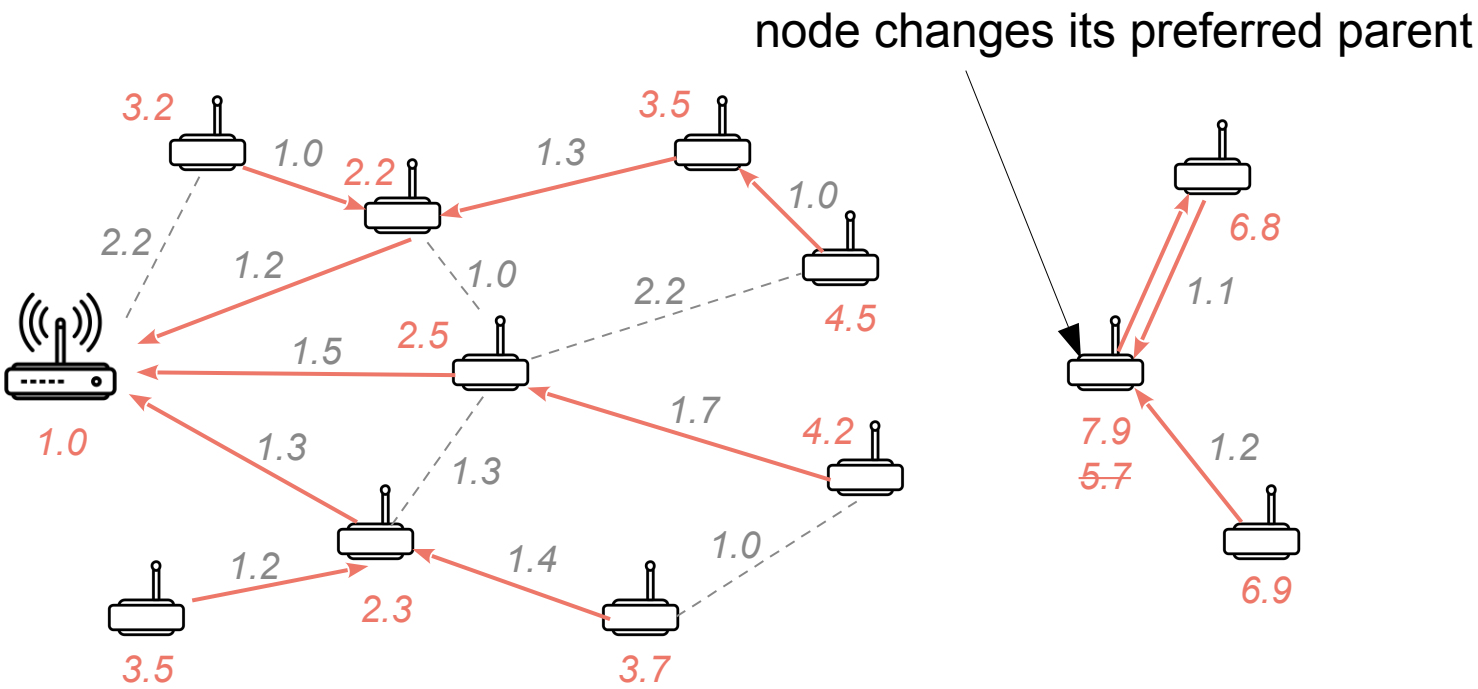
Network Partition



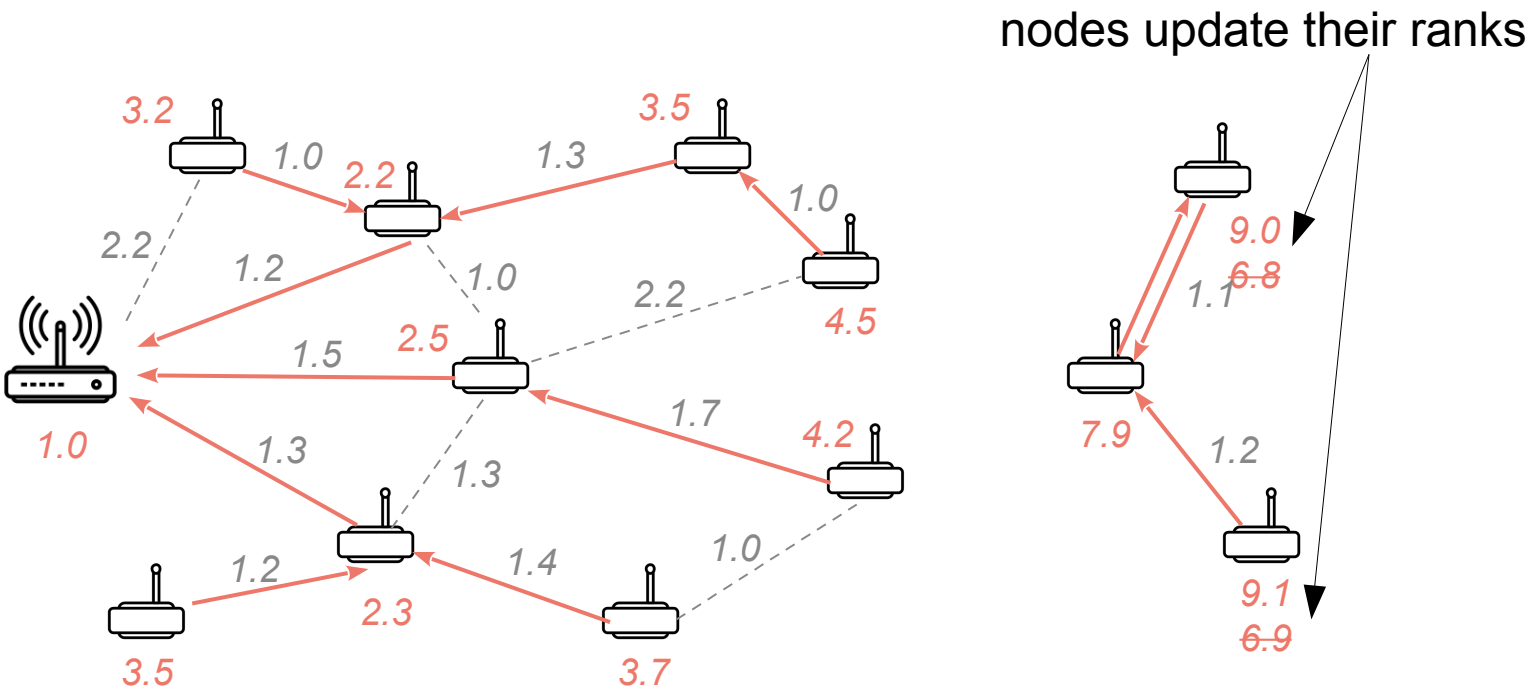
Network Partition



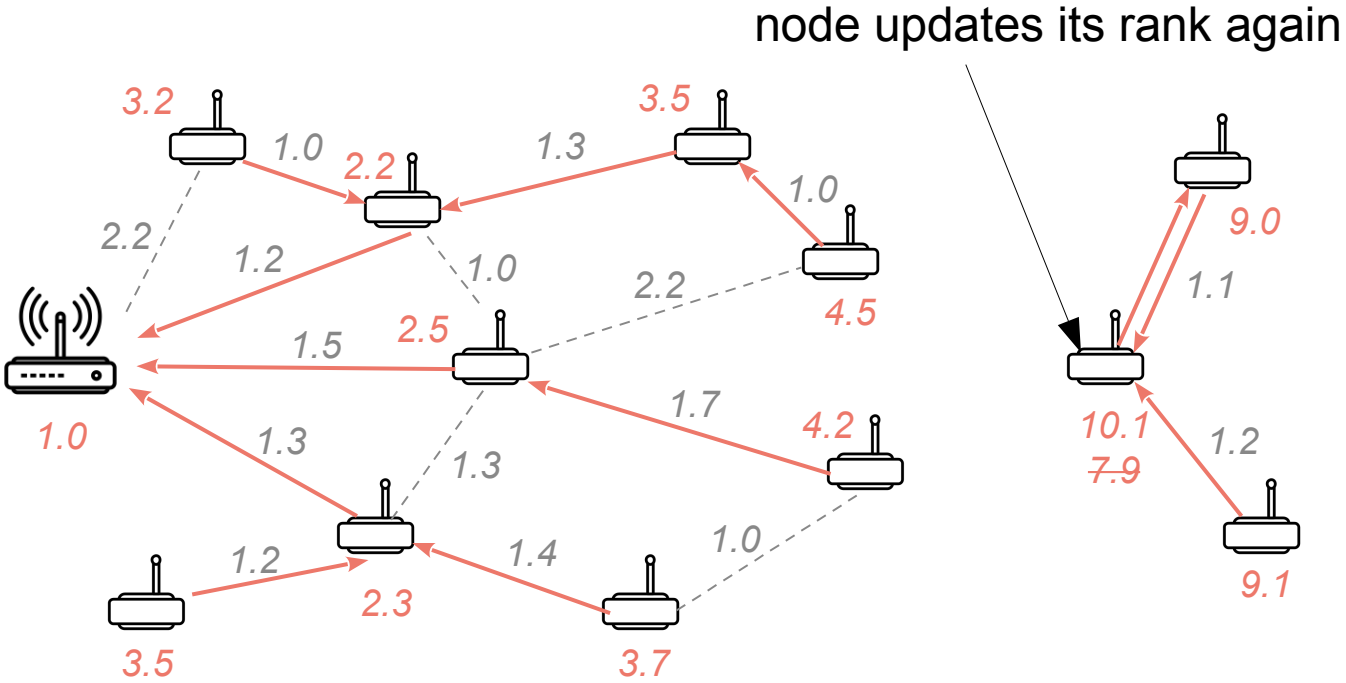
Network Partition



Network Partition

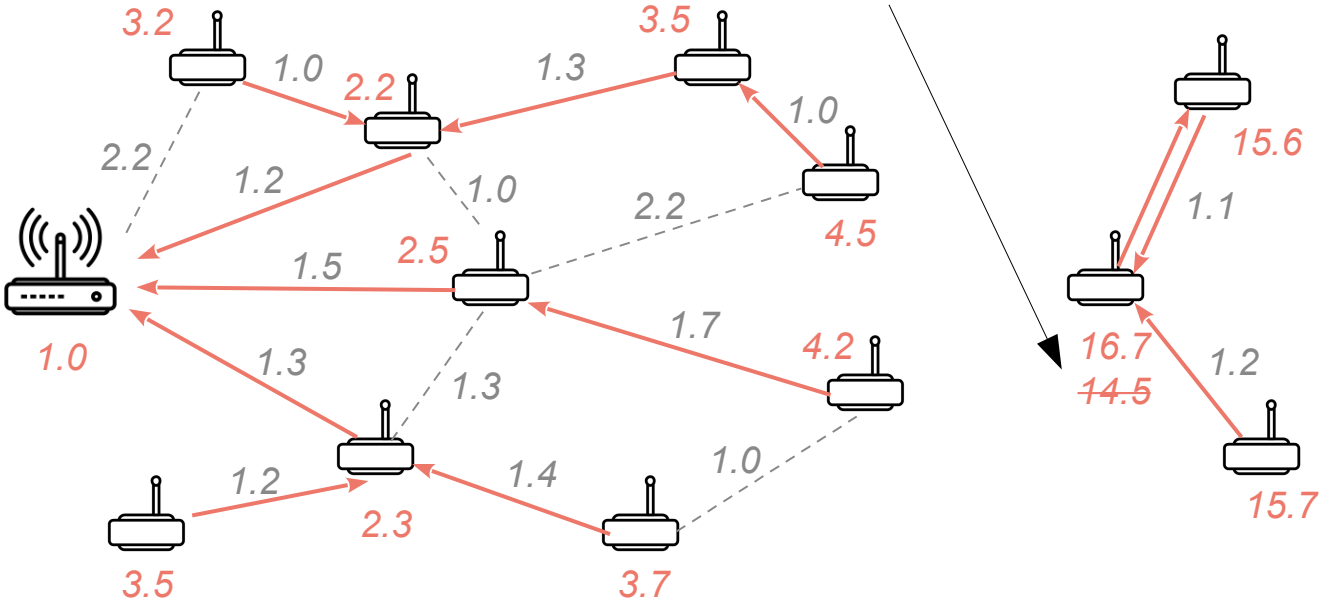


Network Partition



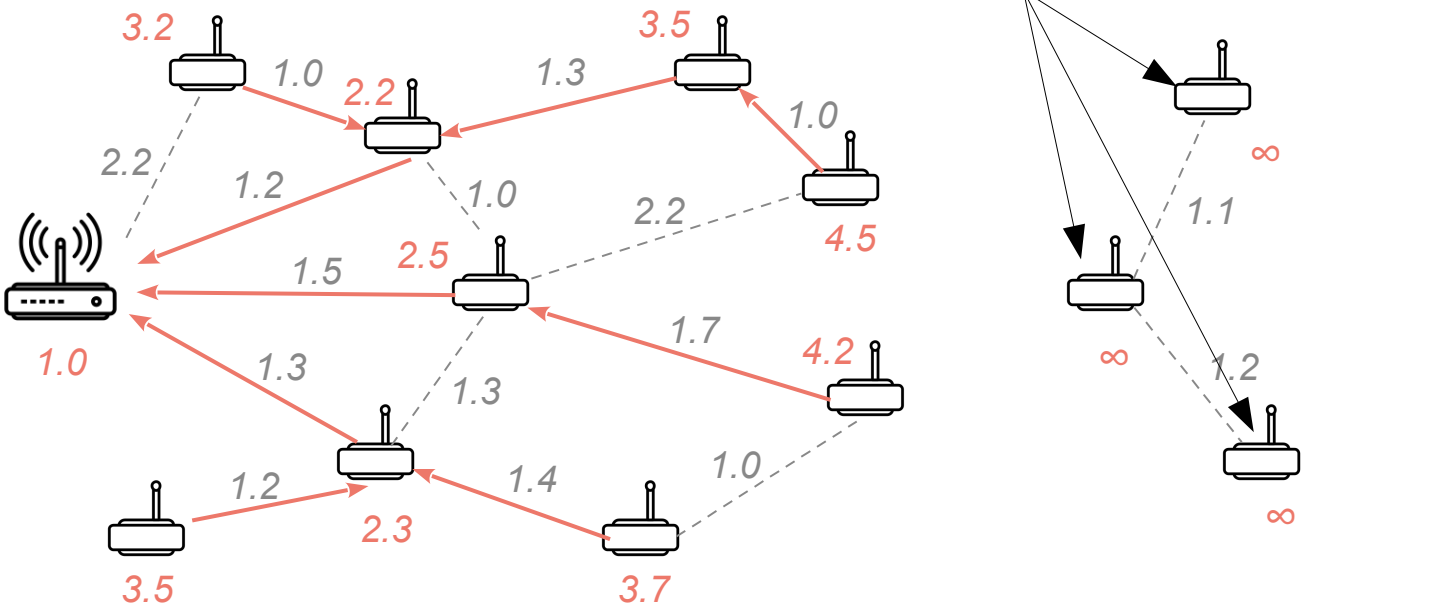
MaxRankIncrease

node's new rank exceeds its smallest value ever by > *MaxRankIncrease*



MaxRankIncrease

nodes discard their preferred parents and adopt infinite ranks



Summary

- **Routing metric:** value that is used for rank computation and is optimized in the network
- **Objective function:** component responsible for choosing preferred parents and computing ranks
- RPL's configuration parameters:
  - ***MinHopRankIncrease:*** minimal difference between a node's rank and its preferred parent's rank
  - ***MaxRankIncrease:*** maximal allowed increase in a node's rank before concluding that the DODAG is broken
  - ***ParentSwitchThr:*** maximal allowed difference between a node's rank and the best rank offered by any of the node's neighbors; a constant that defines hysteresis

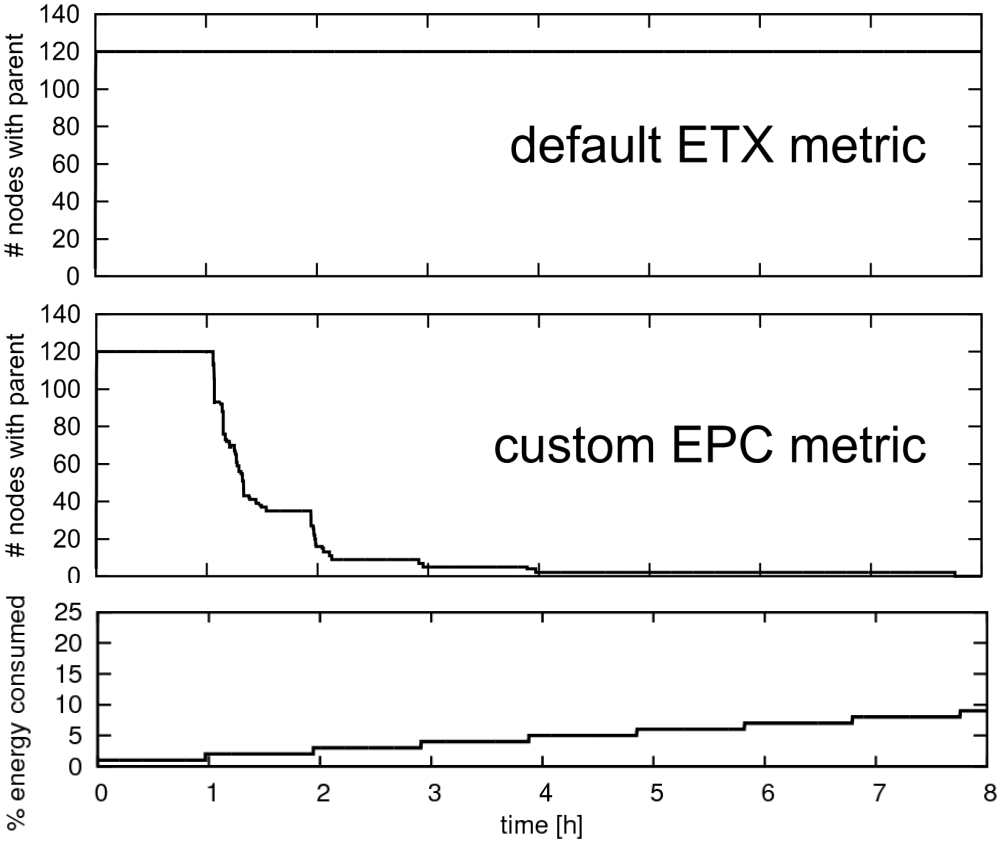
Observation

- RPL allows for various routing metrics and routing formation criteria
- **Advantage:** it is easy to customize the protocol to fit individual requirements of each application
- **Risk:** route formation may fail

Case Study

- Based on a real deployment
- Multi-hop network of battery-powered nodes
- **Goal:** prolong the network lifetime
- **Routing metric:** the percentage of a node's energy budget already consumed (**EPC**)
- **Objective function:** rank = preferred parent's rank + node's own EPC

Case Study


140
120
100
80
60
40
20
0
nodes with parent
default ETX metric
140
120
100
80
60
40
20
0
nodes with parent
custom EPC metric
25
20
15
10
5
0
% energy consumed
0
1
2
3
4
5
6
7
8
time [h]

Case Study

Conclusion: despite having 91% of their energy available, nodes are unable to route any data

Problem

A **bug** in the implementation or a **feature** of RPL's design itself?

Analyzing RPL

- We analyzed RPL's **route formation process**
- **Abstract** objective function and routing metrics
- Goal: **derive conditions** for objective functions and routing metrics under which DODAG construction is **provably correct**

Analyzing RPL

We adapted a model from EWSN 2018:

- Based on **RPL's specification**
- **LTL**-like formalism
- We **abstracted out** an objective function and routing metrics
- We added **dynamically changing** routing metrics

Analyzing RPL

Using the model we formulated a **hypothesis** and subsequently attempted to **prove** it.

Hypothesis 1. All nodes eventually always have their ranks finite.

Proof

1. Common approach: **induction** on a node's **hop count** from the DODAG root

Proof

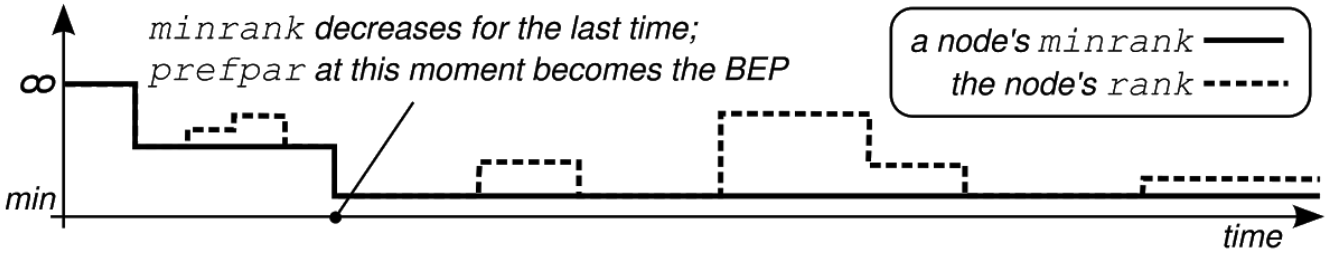
1. Common approach: **induction** on a node's **hop count** from the DODAG root
   - It did not work due to **dynamically changing** routing metrics

Proof

1. Common approach: **induction** on a node's **hop count** from the DODAG root
   - It did not work due to **dynamically changing** routing metrics
2. We were looking for an alternative approach for a long time…

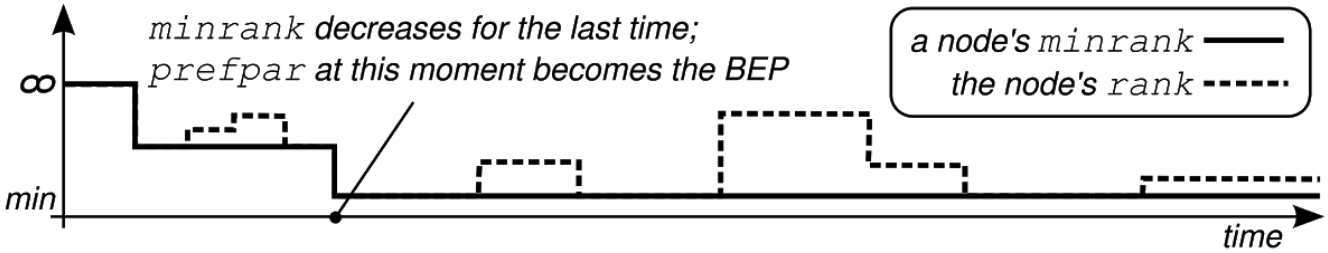
Proof

1. Common approach: **induction** on a node's **hop count** from the DODAG root
   - It did not work due to **dynamically changing** routing metrics
2. We were looking for an alternative approach for a long time…
3. **Best ever parent** (**BEP**): a node's current or former preferred parent, whose selection gave the node the smallest rank ever for the first time



Proof

1. Common approach: **induction** on a node's **hop count** from the DODAG root
   - It did not work due to **dynamically changing** routing metrics
2. We were looking for an alternative approach for a long time…
3. **Best ever parent** (**BEP**): a node's current or former preferred parent, whose selection gave the node the smallest rank ever for the first time



4. Links to BEPs form a **tree** – can be used for induction!

Result

Hypothesis 1. All nodes eventually always have their ranks finite.

Theorem 1. Hypothesis 1 is true.

Result

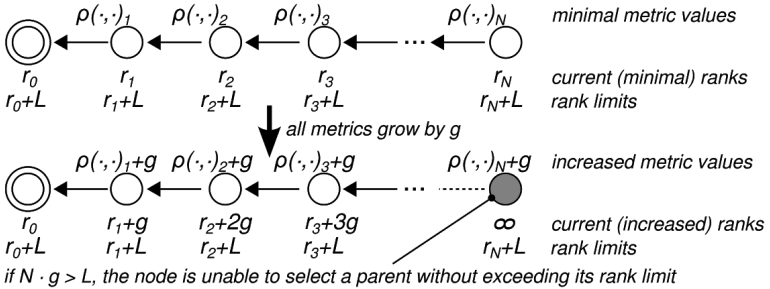
- **C3.** For each entry *n* in a node's *neighborset*, *ρ(n.metrics, self.metrics)* will eventually always be equal to its smallest value ever.

Result

- **C3.** For each entry *n* in a node's *neighborset*, *ρ(n.metrics, self.metrics)* will eventually always be equal to its smallest value ever.
- We allow for the **bounded** growth of metrics

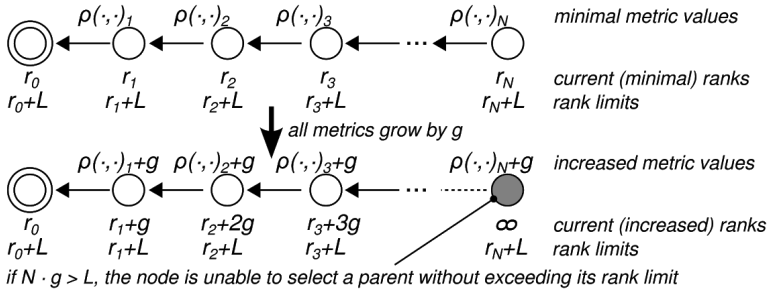
Result

- **C3.** For each entry *n* in a node's *neighborset*, *ρ(n.metrics, self.metrics)* will eventually always be equal to its smallest value ever.
- We allow for the **bounded** growth of metrics



Result

- **C3.** For each entry *n* in a node's *neighborset*, *ρ(n.metrics, self.metrics)* will eventually always be equal to its smallest value ever.
- We allow for the **bounded** growth of metrics



- **C3'.** For each entry *n* in a node's *neighborset*, *ρ(n.metrics, self.metrics)* eventually always does not exceed its smallest ever value plus *MaxRankIncrease / D*, where *D* is the depth of the BEP tree.

Result

- But a node does not need to choose a neighbor that offers it the best rank…

Result

- But a node does not need to choose a neighbor that offers it the best rank…
- **C3”.** For each entry *n* in a node’s *neighborset*, *ρ(n.metrics, self.metrics)* eventually always does not exceed its smallest ever value plus *max(0, MaxRankIncrease / D − ParentSwitchThr)*.

Result

- But a node does not need to choose a neighbor that offers it the best rank…
- **C3".** For each entry *n* in a node's *neighborset*, *ρ(n.metrics, self.metrics)* eventually always does not exceed its smallest ever value plus *max(0, MaxRankIncrease / D − ParentSwitchThr)*.

Proposition 1. If the value of RPL's *MaxRankIncrease* does not depend on the number of nodes in the network, then RPL may not be able to form a connected DODAG.

Result

C1. If *rank* is finite, then *rank* + *MinHopRankIncrease* ≤ $\mathscr{R}$(*rank*, •, •) < *infinity*.

Result

C1. If *rank* is finite, then *rank* + *MinHopRankIncrease* ≤ $\mathscr{R}$*(rank, •, •)* < *infinity*.

C2. $\mathscr{R}$*(n.rank, n.metrics, self.metrics)* = *n.rank* + *ρ(n.metrics, self.metrics)*, where *ρ* depends solely on *n.metrics* and *self.metrics*.

Result

C1. If *rank* is finite, then *rank* + *MinHopRankIncrease* ≤ $\mathscr{R}$*(rank, •, •)* < *infinity*.

C2. $\mathscr{R}$*(n.rank, n.metrics, self.metrics)* = *n.rank* + *ρ(n.metrics, self.metrics)*, where *ρ* depends solely on *n.metrics* and *self.metrics*.

C3". For each entry *n* in a node's *neighborset*, *ρ(n.metrics, self.metrics)* eventually always does not exceed its smallest ever value plus *max(0, MaxRankIncrease / D − ParentSwitchThr)*.

Result

C1. If *rank* is finite, then *rank* + *MinHopRankIncrease* ≤ $\mathscr{R}$*(rank, •, •)* < *infinity*.

C2. $\mathscr{R}$*(n.rank, n.metrics, self.metrics)* = *n.rank* + *ρ(n.metrics, self.metrics)*, where *ρ* depends solely on *n.metrics* and *self.metrics*.

C3". For each entry *n* in a node's *neighborset*, *ρ(n.metrics, self.metrics)* eventually always does not exceed its smallest ever value plus *max(0, MaxRankIncrease / D − ParentSwitchThr)*.

C4. If a node selects a finite *rank* and non-null *prefpar* then there does not exist another candidate for the node's *prefpar* that offers the node a potential rank lower than the selected rank by more than *ParentSwitchThr*.

Recommendations

1. Use a **simple** function for potential rank computation

Recommendations

1. Use a **simple** function for potential rank computation
2. Try to keep routing metric values **bounded**

Recommendations

1. Use a **simple** function for potential rank computation
2. Try to keep routing metric values **bounded**
3. **Explicitly configure** the tolerated rank growth

Recommendations

1. Use a **simple** function for potential rank computation
2. Try to keep routing metric values **bounded**
3. **Explicitly configure** the tolerated rank growth
4. Watch out for **hysteresis** in objective functions

Recommendations

1. Use a **simple** function for potential rank computation
2. Try to keep routing metric values **bounded**
3. **Explicitly configure** the tolerated rank growth
4. Watch out for **hysteresis** in objective functions
5. Stay in control of the **growth** of routing metric values

Conclusions

- We showed that RPL's DODAG formation process is **correct** with the four additional conditions added

Conclusions

- We showed that RPL's DODAG formation process is **correct** with the four additional conditions added
- We presented a new **technique** for proving properties in a DODAG – **induction on BEPs**

Conclusions

- We showed that RPL's DODAG formation process is **correct** with the four additional conditions added
- We presented a new **technique** for proving properties in a DODAG – **induction on BEPs**
- We came up with **practical recommendations** for configuring RPL's route formation

Thank you

Questions?

The presented research was supported by the National Center for Research and Development (NCBR) in Poland under grant no. LIDER/434/L-6/14/NCBR/2015.

Analyzing RPL

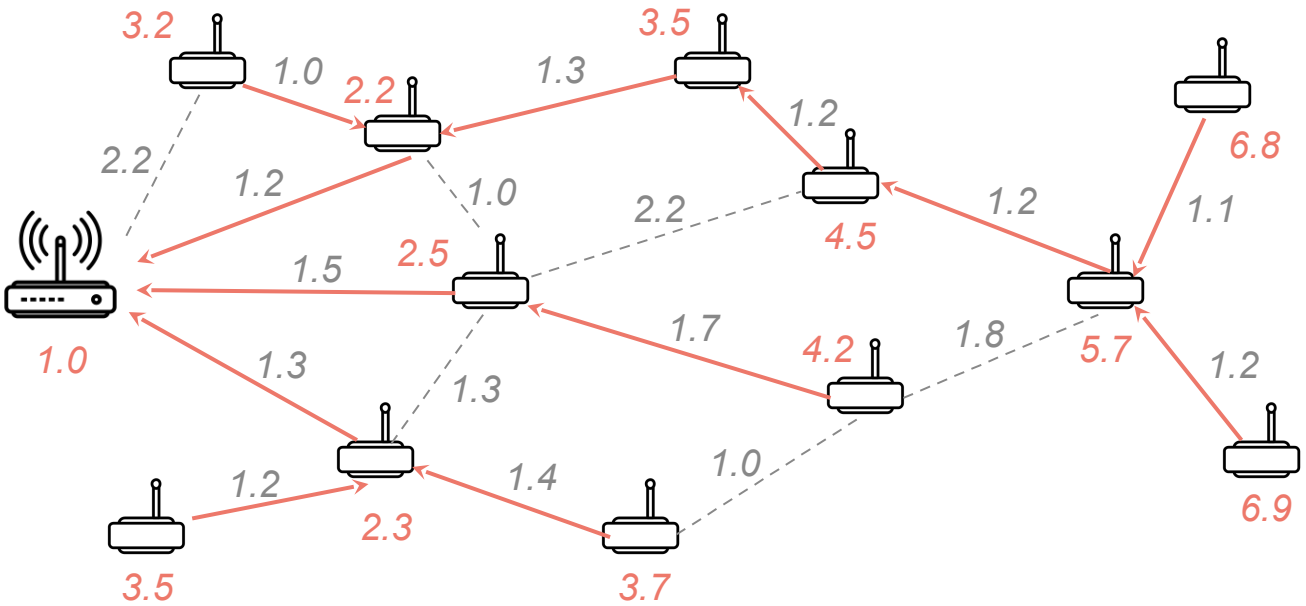
Examples of properties:

OF4. Always, when reselecting its *prefpar* and *rank*, a non-root node adopts *null* as *prefpar* iff it also adopts infinite *rank*. These are also the initial values of the two variables at the non-root node when the node starts.

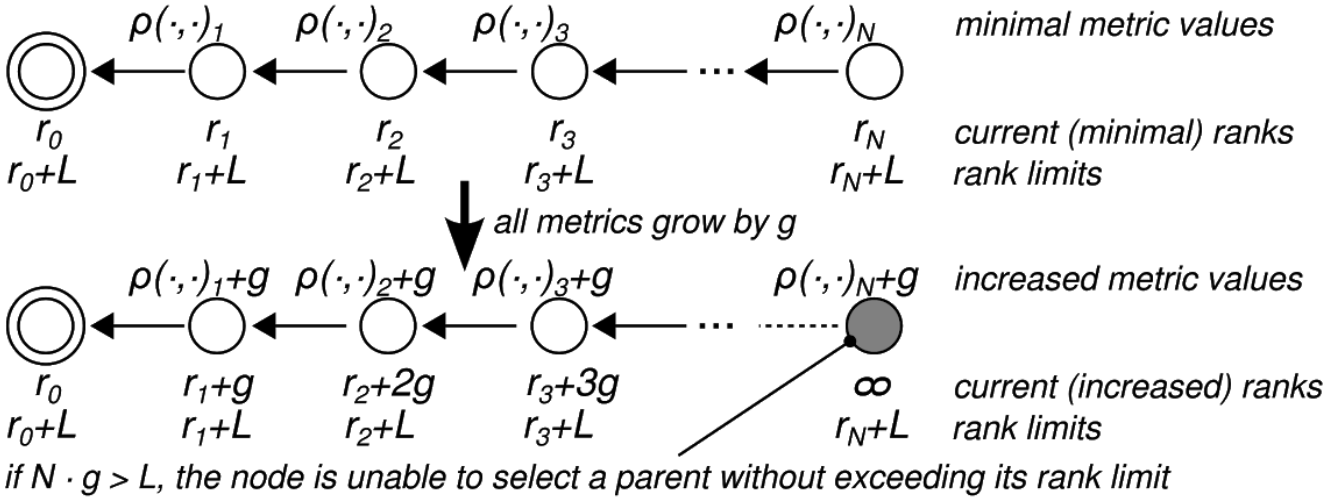
DIO2. A node always eventually receives a DIO message from each of its neighbors.

RA1. Always, if a node's *neighborset* changes, then the node eventually reselects its *prefpar* and *rank*.

Introduction



Proof


ρ(·,·)1
ρ(·,·)2
ρ(·,·)3
ρ(·,·)N
minimal metric values
r0
r1
r2
r3
rN
current (minimal) ranks
r0+L
r1+L
r2+L
r3+L
rN+L
rank limits
all metrics grow by g
ρ(·,·)1+g
ρ(·,·)2+g
ρ(·,·)3+g
ρ(·,·)N+g
increased metric values
r0
r1+g
r2+2g
r3+3g
∞
current (increased) ranks
r0+L
r1+L
r2+L
r3+L
rN+L
rank limits
if N · g > L, the node is unable to select a parent without exceeding its rank limit

Case Study

- **Energy consumption model**
  - total battery lifetime of 96 h
  - battery consumption: 1% approximately every hour
- **Application**
  - typical data collection application
  - nodes repeatedly send one-frame packets to the DODAG root

Low Power and Lossy Network