RPL: IPv6 Routing Protocol for Low Power and Lossy Networks
(draft-ietf-roll-rpl-04)

Jonathan Hui
RPL Author Team

ROLL WG Meeting
76th IETF Meeting
Hiroshima, Japan

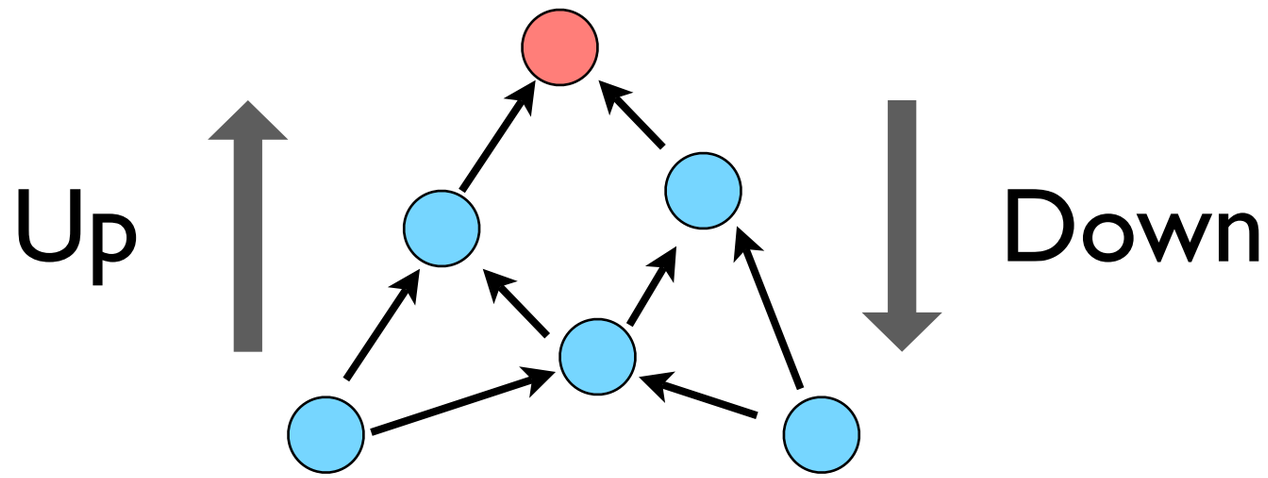
Outline

- Basic Approach
- Mechanism Details (draft-03 vs. draft-04)
- Open Issues
- Next Steps

Basic Approach

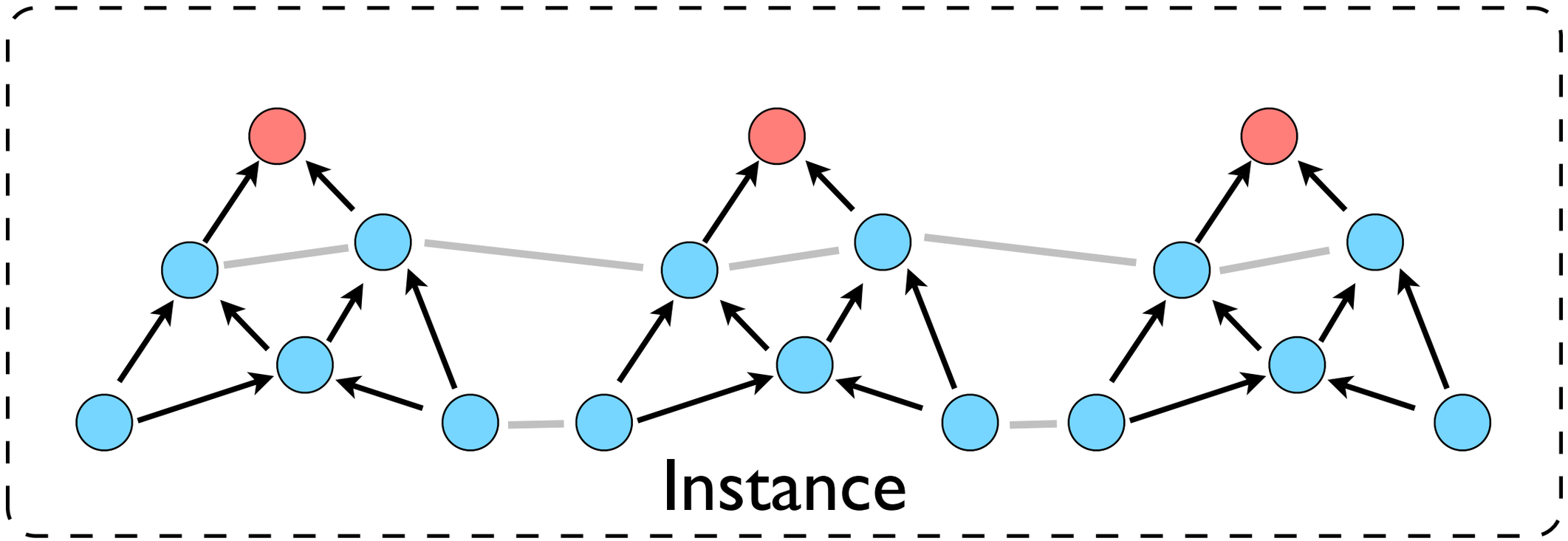
Overview

- Assumption: most traffic flows through few nodes
  - many-to-one
  - one-to-many
  - baseline required by all requirements drafts
- Approach: build DAG(s) rooted at these nodes
  - Up towards the DAG root for many-to-one
  - Down away from the DAG root for one-to-many
  - Use the DAG to detect and avoid loops
  - Allow point-to-point via up* down*



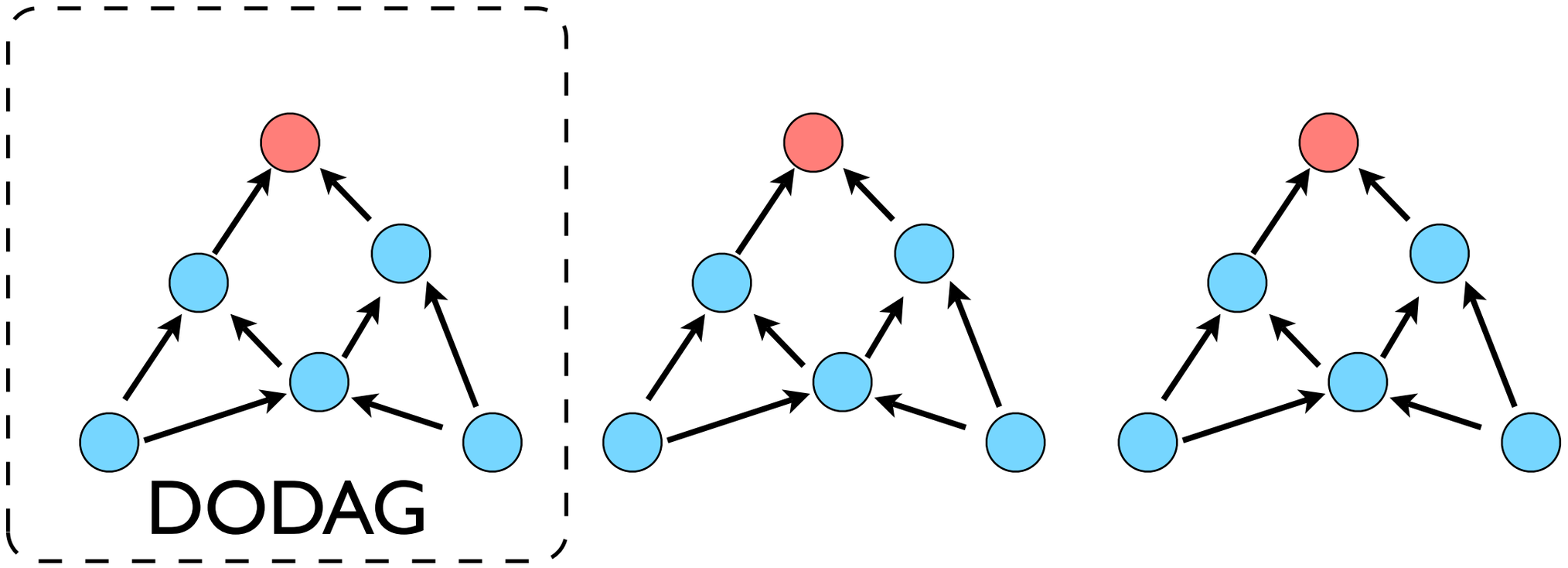
Definitions

- Instance
  - Defines the optimization objective when forming paths towards roots
  - Link properties: (Reliability, Latency), Node properties: (Powered or not)
  - Objective: optimize paths based on one or more metrics
  - Scope: RPL network
  - Composed of one or more disjoint DODAGs



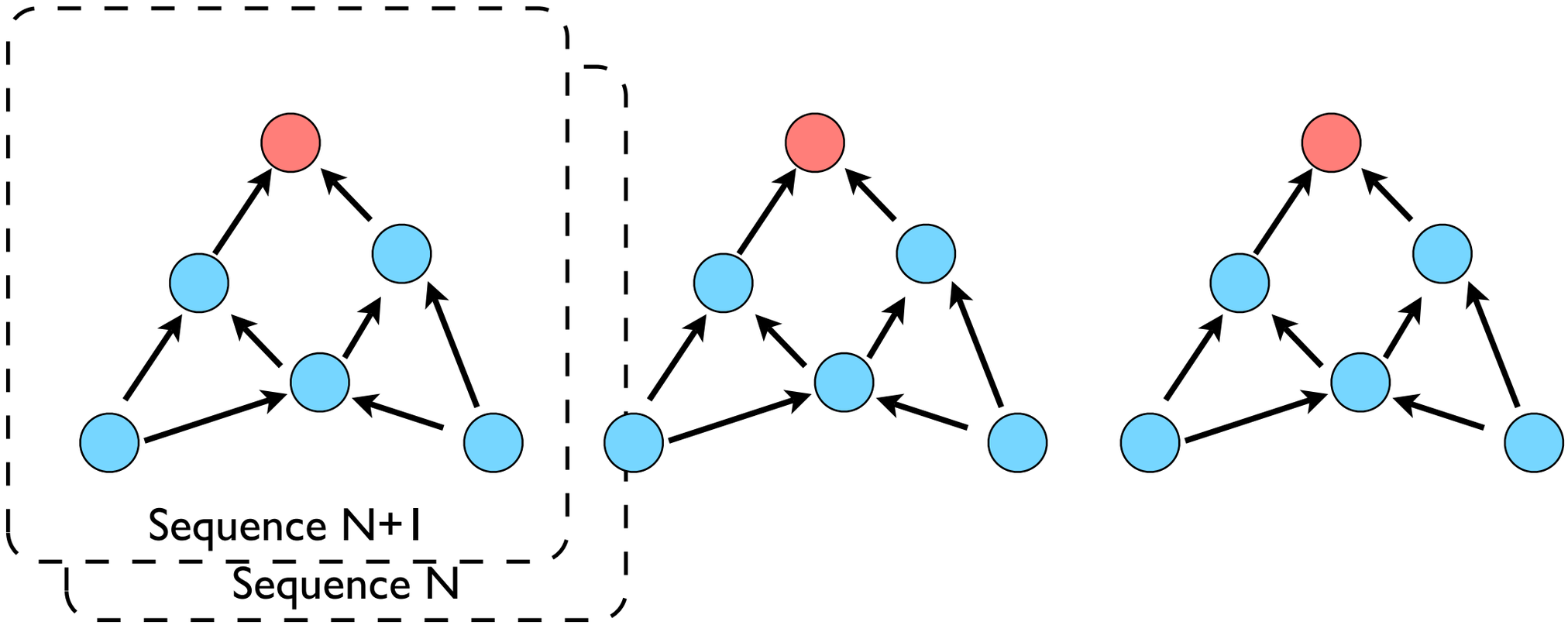
Definitions

- DODAG
  - Defines a DAG that forms paths to a single logical root
  - Scope: within an Instance



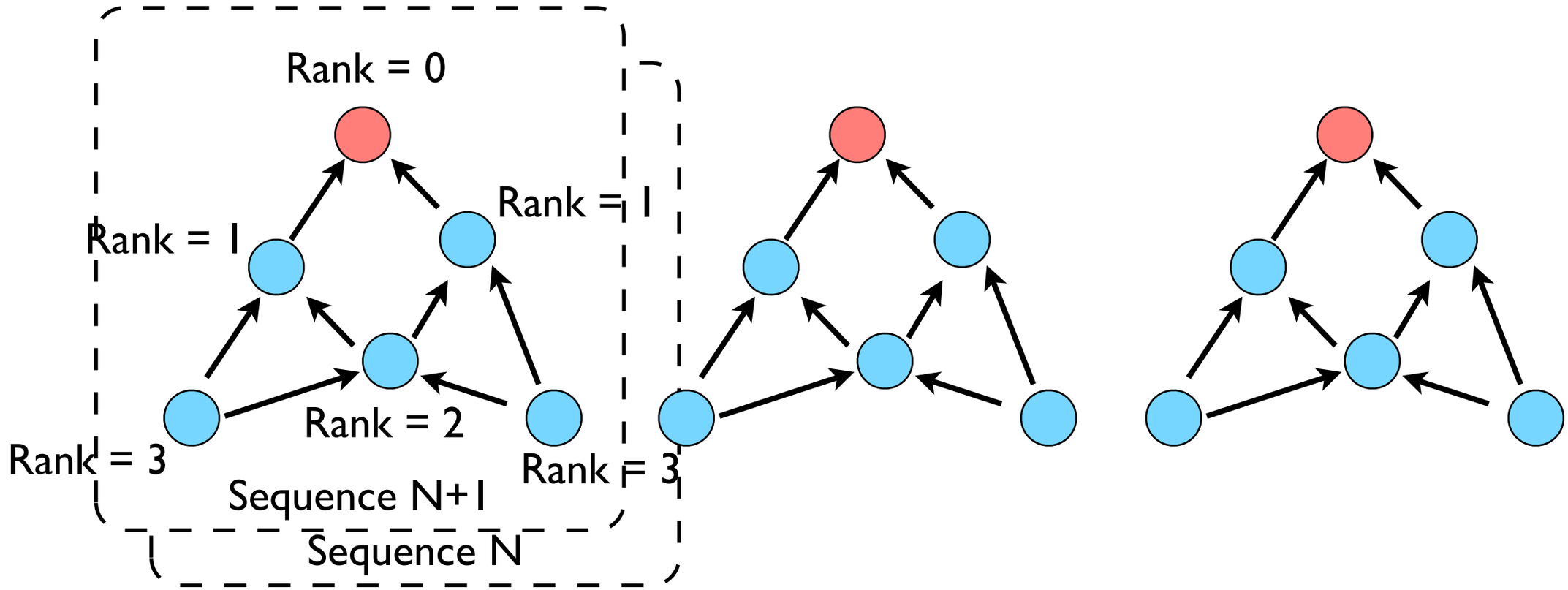
Definitions

- DODAG Iteration
  - A DODAG constructed using a particular sequence
  - Scope: within a DODAG



Definitions

- Node Rank
  - Defines a node's relative position within a DODAG
  - Scope: within a DODAG Iteration



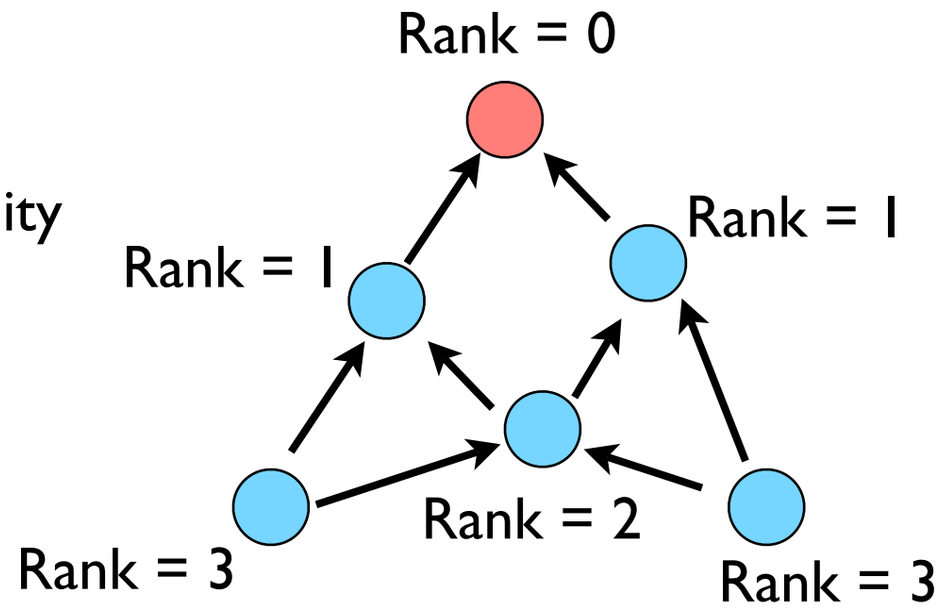
Definitions

- Instance: defines optimization objective for the network
- DODAG: a disjoint DAG within an Instance
- DAG Iteration: a DODAG built with a particular sequence number
- Rank: position within a DODAG Iteration

- Objective Function: identifies metrics, constraints, and objectives
- Objective Code Point: identifies Objective Function

DAG Construction

- Distance-Vector
  - advertise path cost to root
  - choose parents that minimize path cost
  - but be careful about loops & count-to-infinity



- Assign every node a Rank
  - Rank strictly decreasing towards root

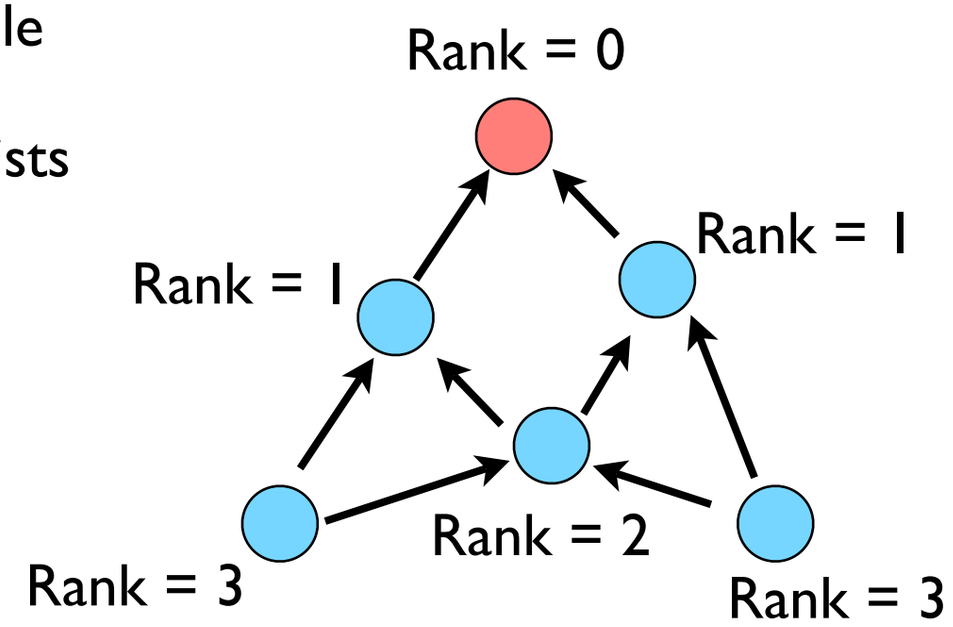
Route Construction

- Up routes towards nodes of decreasing rank
  - DAG parents

- Down routes towards nodes of increasing rank
  - Nodes inform parents of their presence and reachability to descendants.
  - Record route/source route for nodes that cannot maintain any down routes.

Forwarding Rules

- All routes: up*down* along DODAG
- When going up
  - always forward to lower rank when possible
  - may forward to sibling if no lower rank exists
- When going down
  - forward based on down routes



Metric vs. Rank

- Metric is used to achieve an optimization goal
- Rank: path calculation according to objective metric
  - Scalar that represents relative position within a DAG
  - Strictly increasing from the root
  - Topological constraint to avoid and detect loops
  - Coarse granularity allows siblings (in addition to parents, children)
  - Common language if we want to utilize different OCPs in a DAG

Protocol Mechanisms

Protocol Mechanisms

- Control Messages
  - Conveyance
  - Loop Avoidance
  - Route Flapping Avoidance
- Loop Detection & Repair

- Present proposals from Draft 03 & Draft 04
- WG Feedback on 03: SIMPLIFY

Draft 03 to 04 Summary

- Remove binding between RPL and IPv6 ND
- Remove detach/float/reattach local repair
- Remove Hold-Up/Down Timer
- Specify data-path loop detection/repair

Conveyance (draft-03)

- Bind to IPv6 ND
- Router Solicitation
- Router Advertisement
  - DAG Information Option (form the DAG)
- Neighbor Advertisement
  - Destination Adv Option (form down routes)

Conveyance (draft-04)

- Create new ICMPv6 type for RPL
  - Use ICMPv6 Code to identify RPL message

- DIS: DAG Information Solicitation
- DIO: DAG Information Object
- DAO: Destination Advertisement Object

Loop Avoidance (draft-03)

- Loops may occur when node increases Rank
- Global repair
  - Create new DAG Iteration (use sequence number to rebuild DAG)
  - Sequence number establishes event-horizon
- Local repair
  - Detach/float/merge within a DAG instance
  - Use DAG Hop Timer to color sub-DAG and reduce advertisements

Loop Avoidance (draft-04)

- Only global repair for simplicity
  - Must wait for new instance to increase rank
  - Maybe too simple?

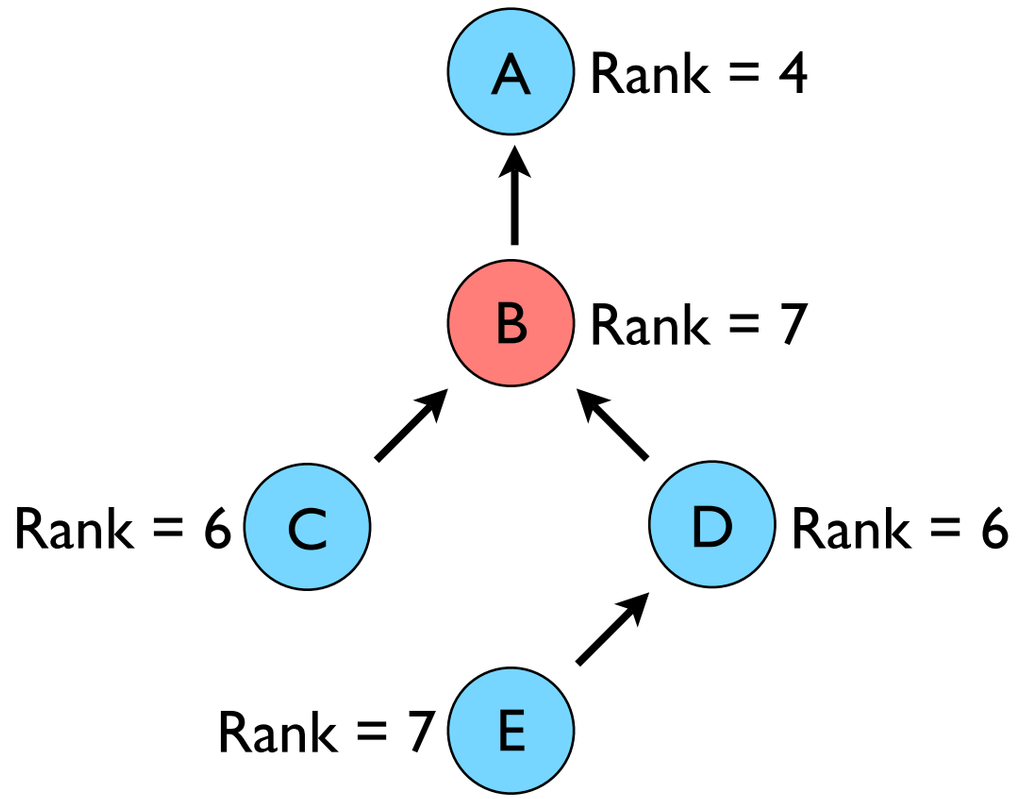
- If no parents exist with lower rank, MUST poison route by advertising infinite cost

Loop Detection & Repair (draft-03)

- Not addressed (listed as open issue)

Loop Detection & Repair (draft-04)

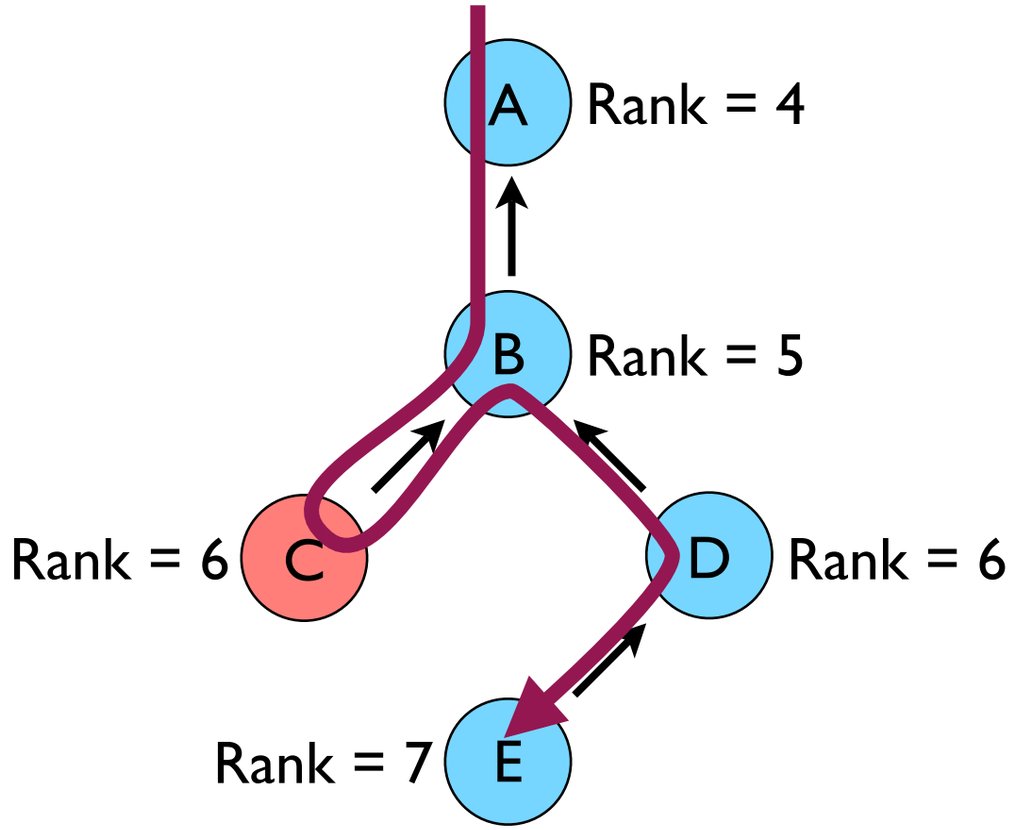
- Up routes must strictly decrease in rank
- Down routes must strictly increase in rank



- Generalized to inconsistency detection & repair

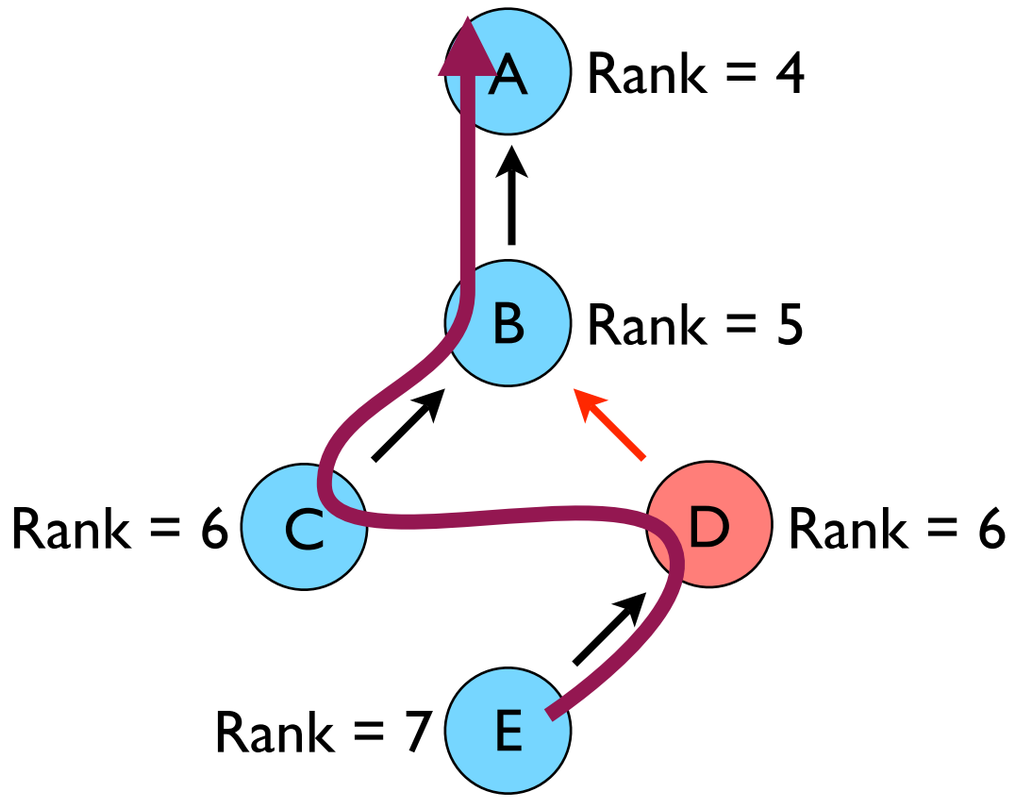
Inconsistency Detection/Repair (draft-04)

- Pass back to parent if no down route exists
- Cleanup stale down routes if dgram is passed back



Loop Detection & Repair (draft-04)

- Parent routes fail, use siblings (same rank)
- Allow at most one (?) sibling hop at a time



Loop Detection & Repair (draft-04)

- Include routing info in data path to validate DAG
  - Instance ID identifies instance to route along
  - Up/Down Bit to identify up vs. down
  - Sender's Rank to assert Rank invariant
  - Rank-Error Bit to tolerate single rank error
  - DAO-Error Bit to back-track and cleanup state
  - Sibling Bit to allow at most one sibling hop

Route Flapping Avoidance (draft-03)

- Use a Hold-Down Timer to delay parent eviction

Route Flapping Avoidance (draft-04)

- Removed: unclear if needed given other mechanisms

Draft 03 to 04 Summary

- Remove binding between RPL and IPv6 ND
- Remove detach/float/reattach local repair
- Remove Hold-Up/Down Timer
- Specify data-path loop detection/repair

- 93 page ➜ 82 pages... and counting

Open Issues

Open Issues

- Behavior when OF is not supported
- Local repair
- Use of siblings
- Minimum Viable Protocol

Open Issues: OF Not Supported

- Node wants to join a network, but does not understand OFx
- Join and extend network using "default" OCP
  - requires all nodes to implement OF0
- Join as leaf and log issue
  - allows connectivity but does not extend the network
- Strong consensus on "join as leaf", will fix in draft-05 and close this issue

Open Issues: Local Repair

- Global repair
  - requires nodes to wait for sequence number
  - delay to repair depends on sequence number refresh rate
- Local repair
  - nodes can move down in the DAG within an instance
  - risks creating loops and count-to-infinity issues
  - general consensus that important cases exist
- One proposed mechanism
  - Use DAG Hop Timer to wait for poisoning to occur
- Another proposed mechanism
  - nodes can move down at most X ranks within a sequence
  - if things are really bad, must wait for new sequence

Open Issues: Utilize Siblings?

- Only parents
  - Only route to nodes with lower rank
- Directional sibling links
  - Equal rank may only route one way
  - More receiver diversity
- Bi-directional sibling links
  - Equal rank nodes may route through each other
  - Even more receiver diversity
  - Loop detection and/or error reporting when siblings have no parents

Open Issues: MVP

- General consensus: draft 03 is far too complex.
- Draft 04 removes many mechanisms/knobs/hooks
  - hold-up/down timers, detach/float/reattach local repair
- Should we simplify more?
  - Nearly orthogonal Instance and DAG concepts
  - Backtracking on DAO errors
  - Utilization of sibling links
  - Objective Control Point generality
- What is missing?
  - Local repair, Address/Header Compression
  - Tradeoff between defining a feature-limited base architecture and working "well" out-of-the-box.

Supporting Drafts

- Metrics
- Applicability Statements
- Objective Function specifications

- Source Routing?
- Address/Header Compression?

Next Steps

Next Steps

- State machine
- Clarify relationship between neighbor set and candidate parents
- Specify node operation when OF is not supported
- Focus on security
- Many editorial improvements to make the spec more clear, concise, and organized

End

Open Issues: Inconsistency Detection

- IPv6 Flow Label?
  - Not compliant with RFC 3697
  - Require end-points to set Flow Label to zero/Instance ID
  - Edge Routers then reset Flow Label to zero/Instance ID
- Define an IPv6 Hop-by-Hop Option?
  - Requires all routers to process a IPv6 HBH Option header
- Define a new IPv6 Extension Header?
  - Only processed by RPL routers
  - Requires all end-points to understand new extension header
- Currently soliciting 6man for feedback