Implementing new Topology Mapping Primitives

Guillermo Baltra

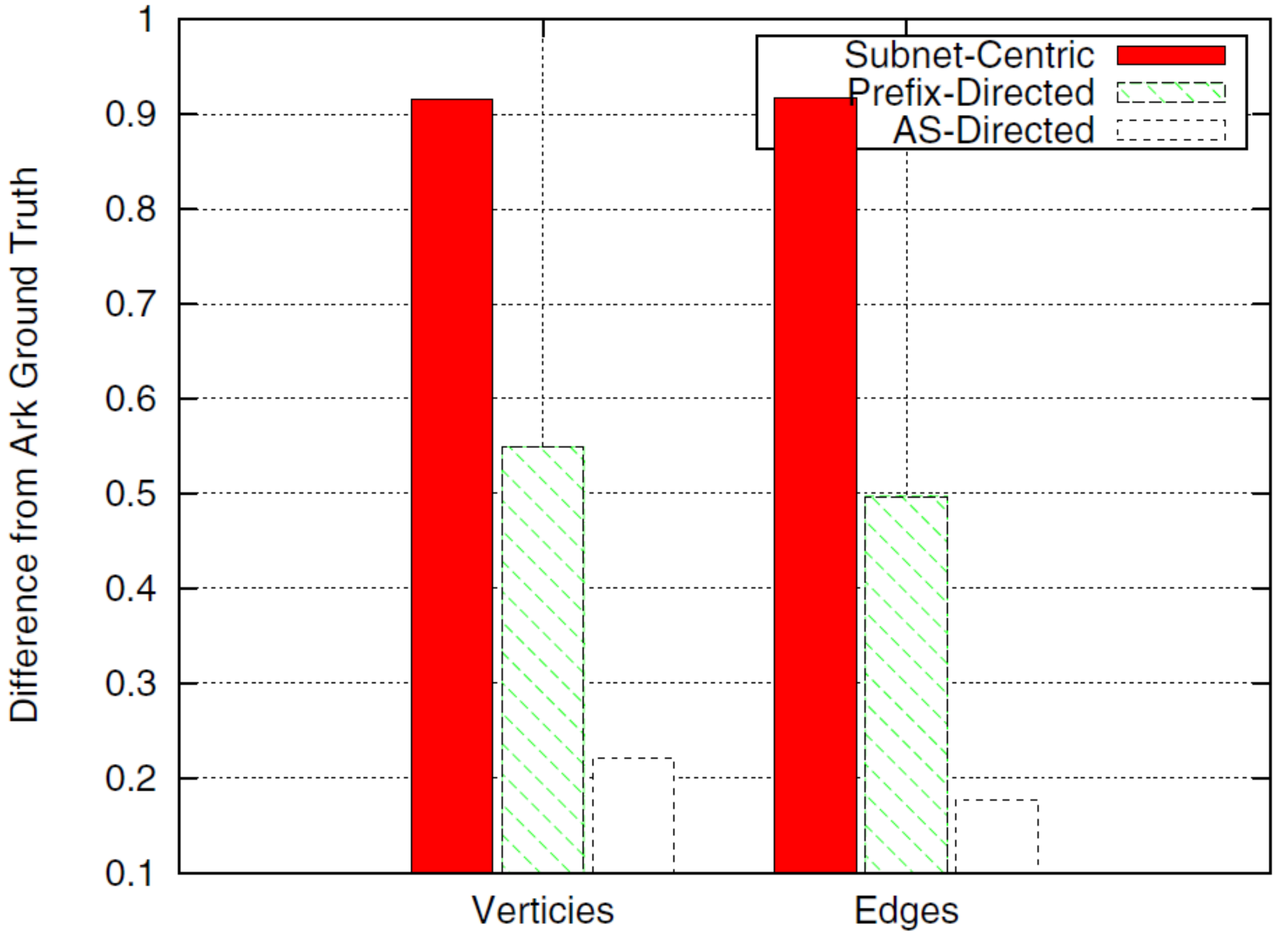
Prior Work

“Primitives for Active Internet Topology Mapping: Toward High-Frequency Characterization”, Beverly, R., Berger, A., Xie, G., IMC 2010.

- Demonstrated the ability of each primitive to generate significant probing savings. Fewer probes implies potential to:
  - Improve quality of topologies as currently inferred
  - Additional results with same probing bugdet, e.g. alias resolution .
  - Perform more complete/detailed probing
  - Increase feasible frequency (i.e. speed) of full-topology inferences

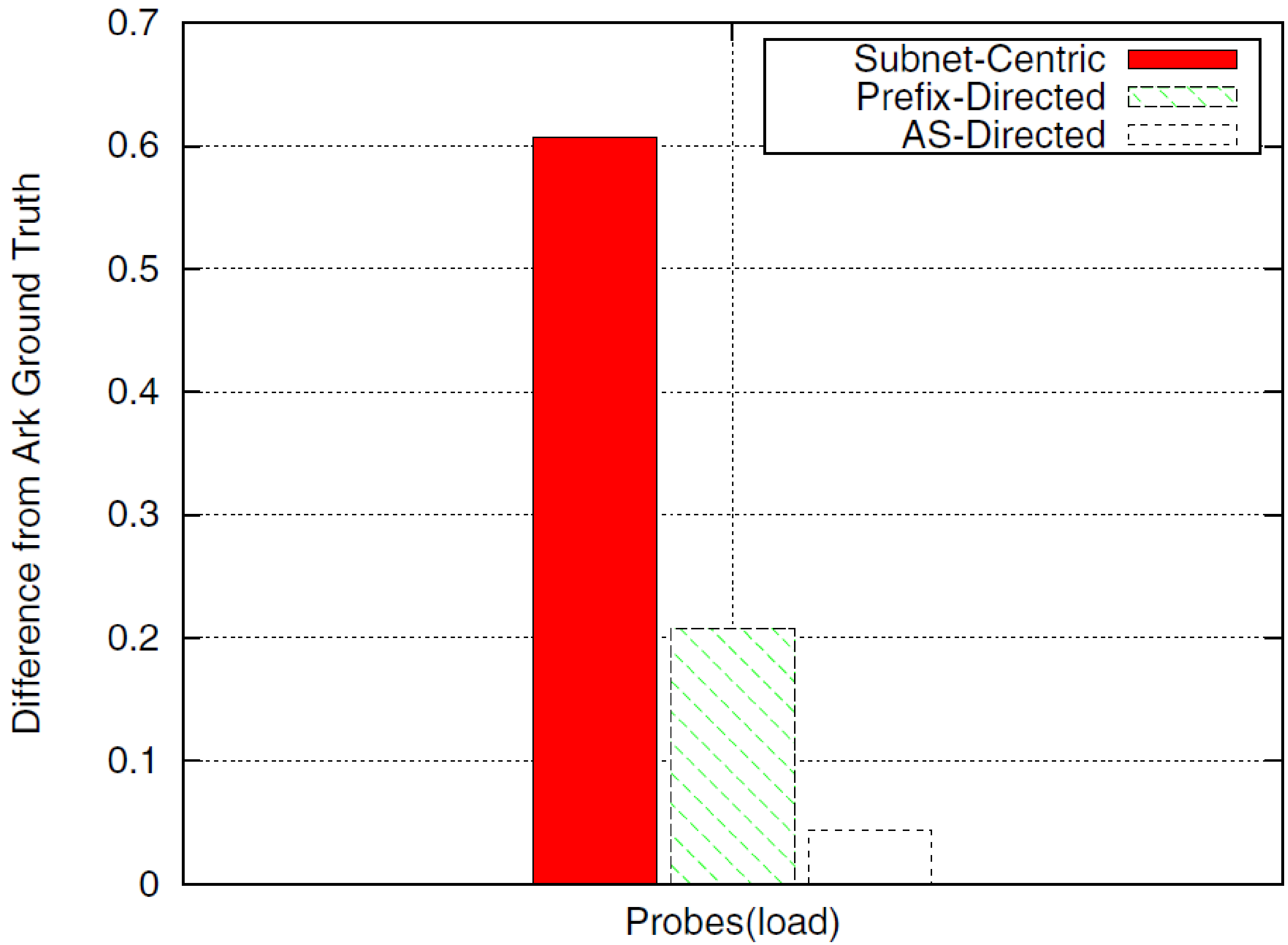
Subnet Centric Probing (SCP)

- Adapt the number of probes to the degree of subnetting to avoid wasted probing.
- Discover internal structure of networks.
- Leverage BGP as coarse structure.



(b) Model fidelity: Subnet centric probing captures $\geq 90\%$ of the vertices and edges.

Ref. [1]



(c) Induced load: Subnet centric uses $\simeq 60\%$ of the ground truth load.

Ref. [1]

Subnet Centric Probing (SCP)

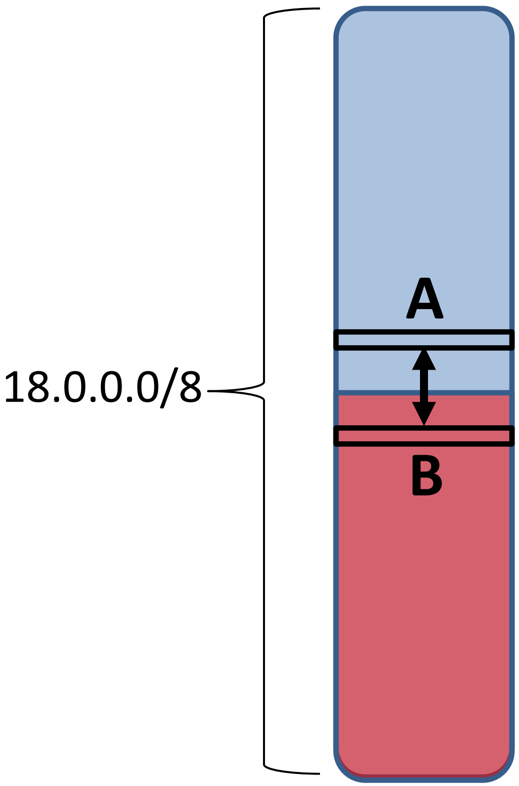
- 3 separable problems via a unified methodology:
  1. Select destinations.
  2. Select sources.
  3. Stopping criterion.

Least Common Prefix (LCP)

- Iteratively pick destinations within prefix that are maximally distant (in subnetting sense)
- 2 numerically consecutive IP addresses more likely to share paths.
- But address “distance” can be misleading: e.g. 18.255.255.100 vs 19.0.0.4 vs. 18.0.0.5

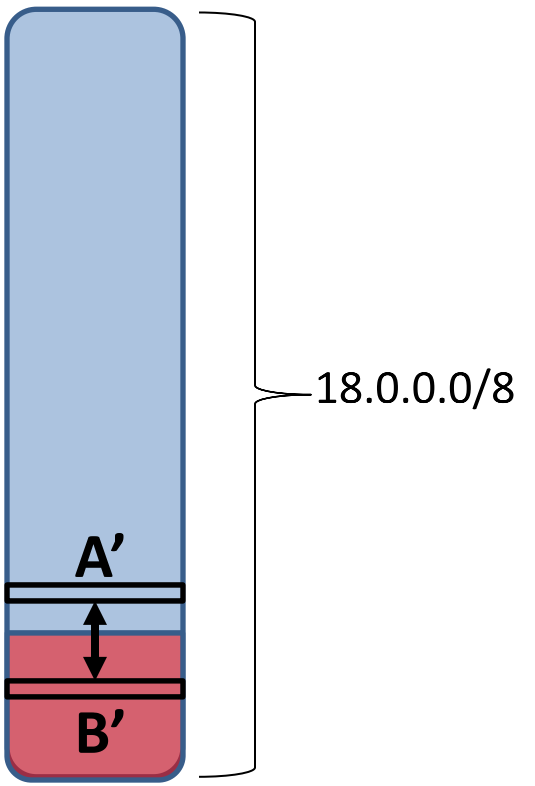
Least Common Prefix (LCP)

- Use knowledge of how networks are provisioned and subnetted.

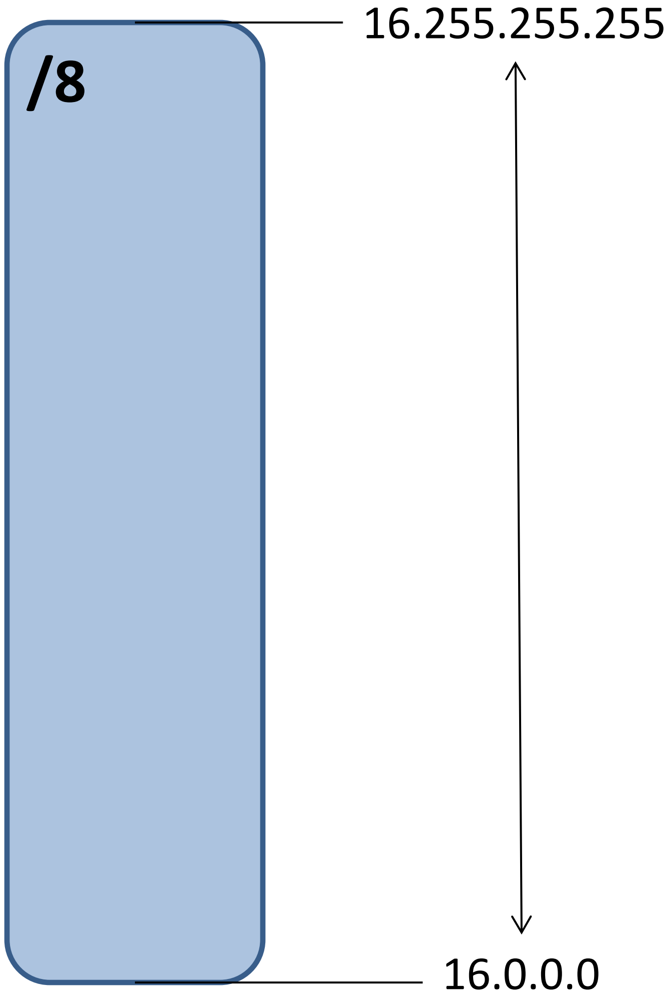


Easier to believe A and B in different subnets

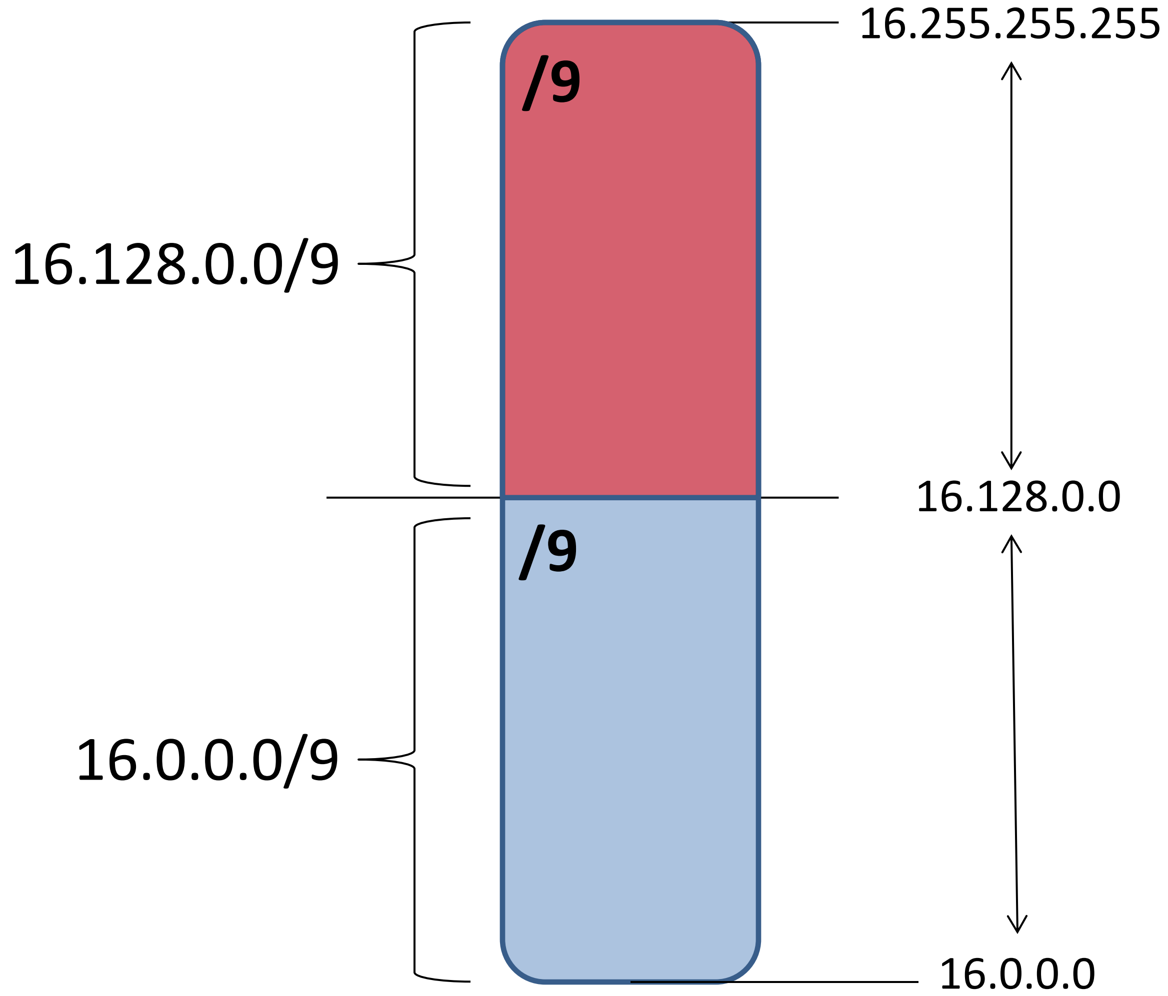
Than A’ and B’ in different subnets.



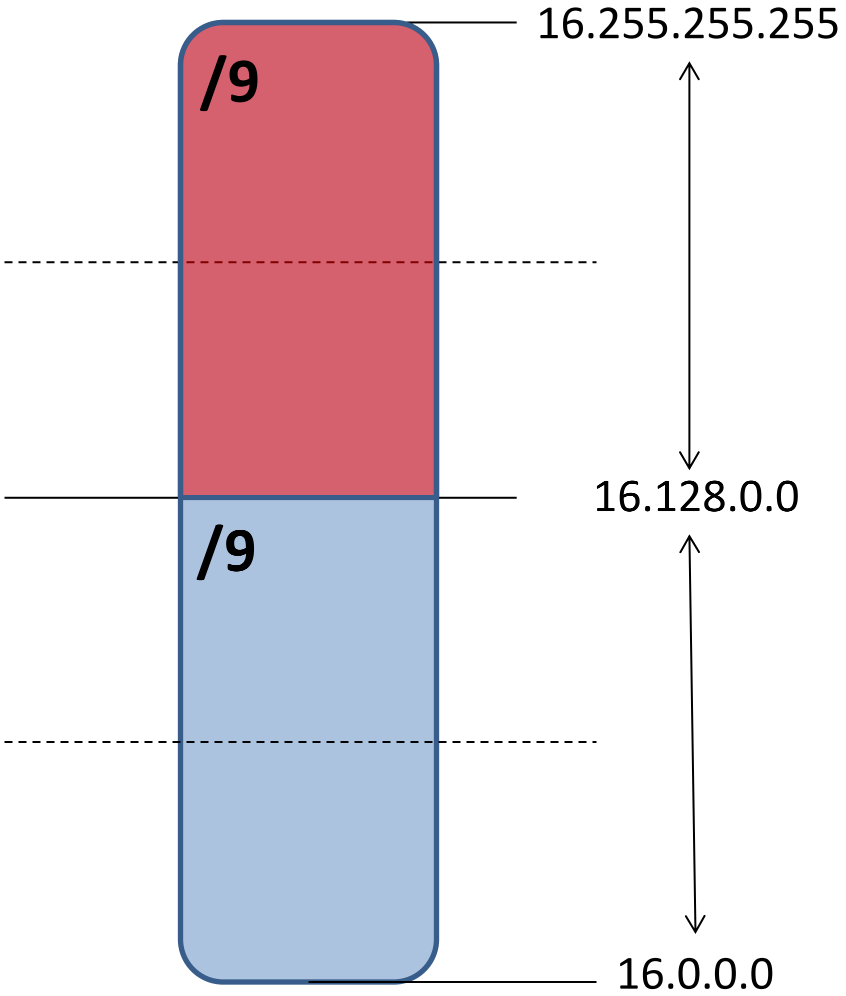
Least Common Prefix (LCP)



Least Common Prefix (LCP)

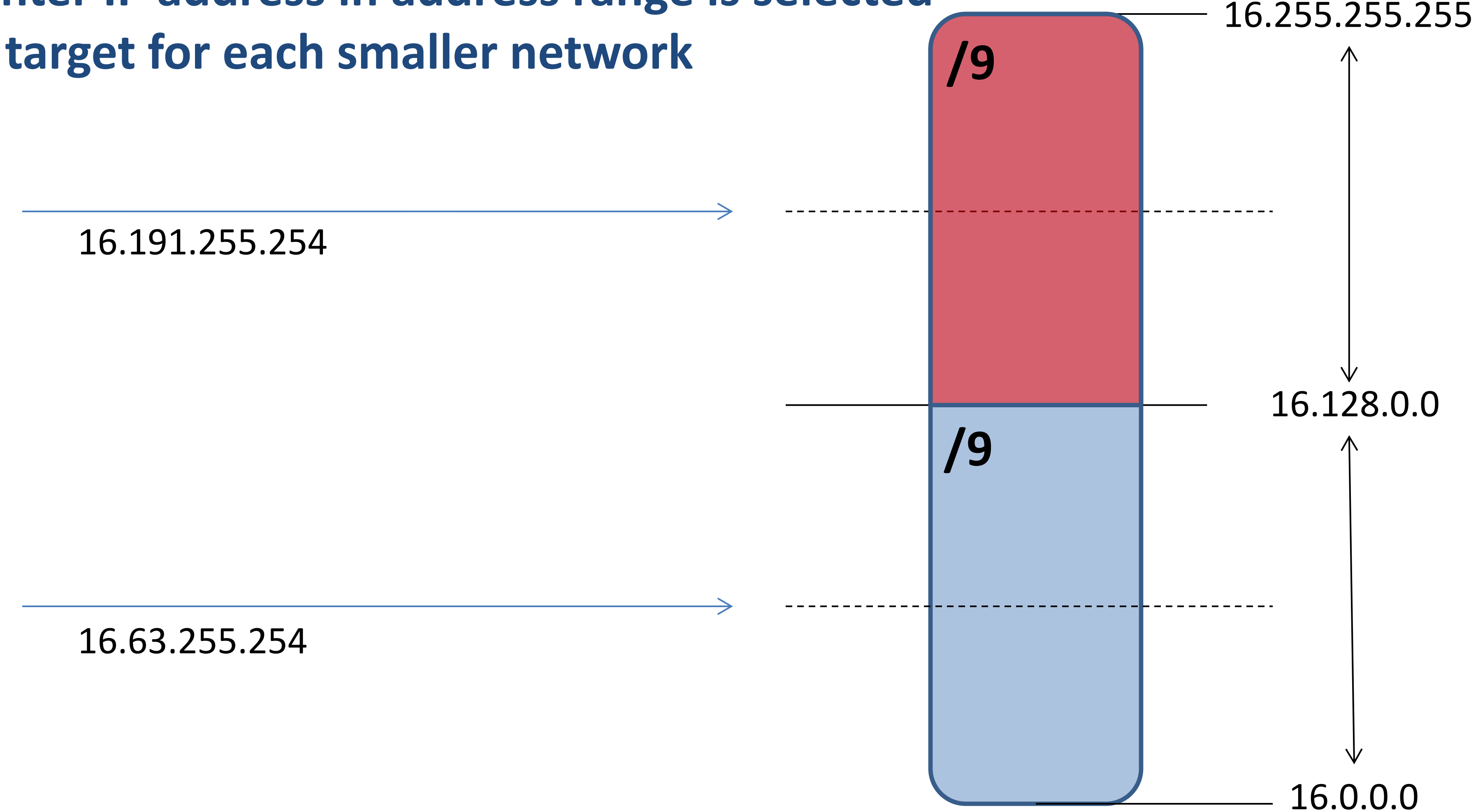


Least Common Prefix (LCP)

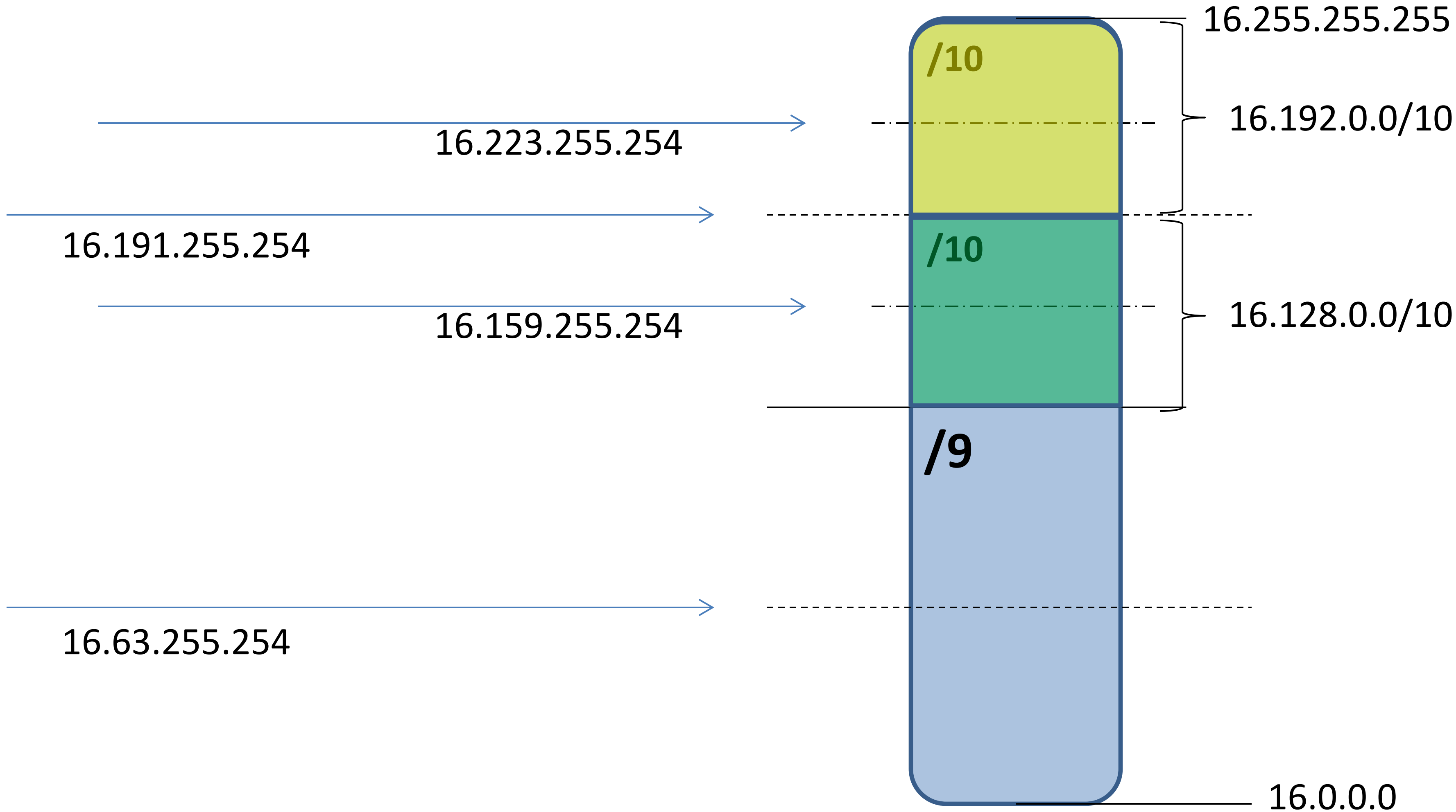


LCP and Target Determination

Center IP address in address range is selected as target for each smaller network

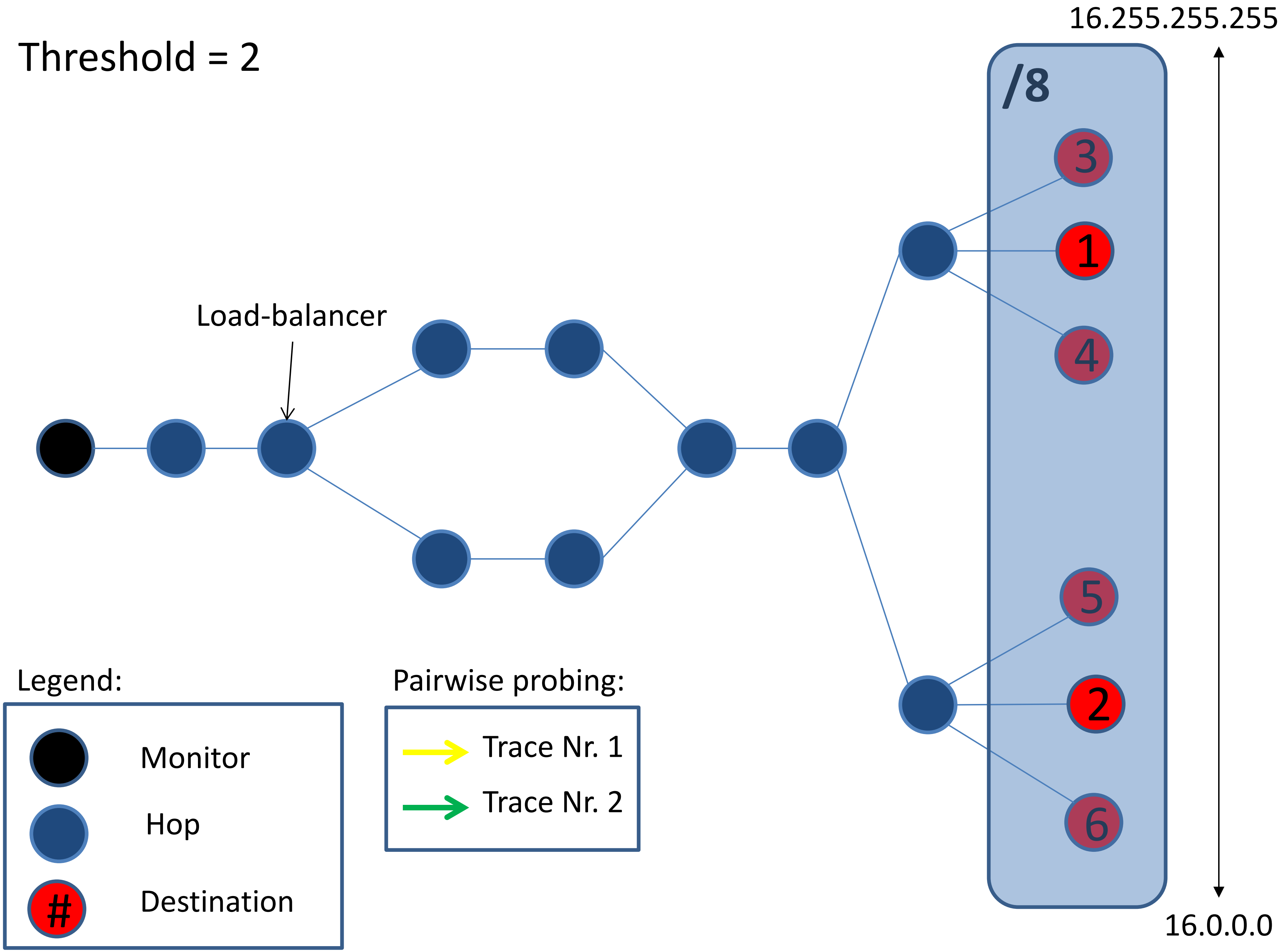


LCP and Target Determination

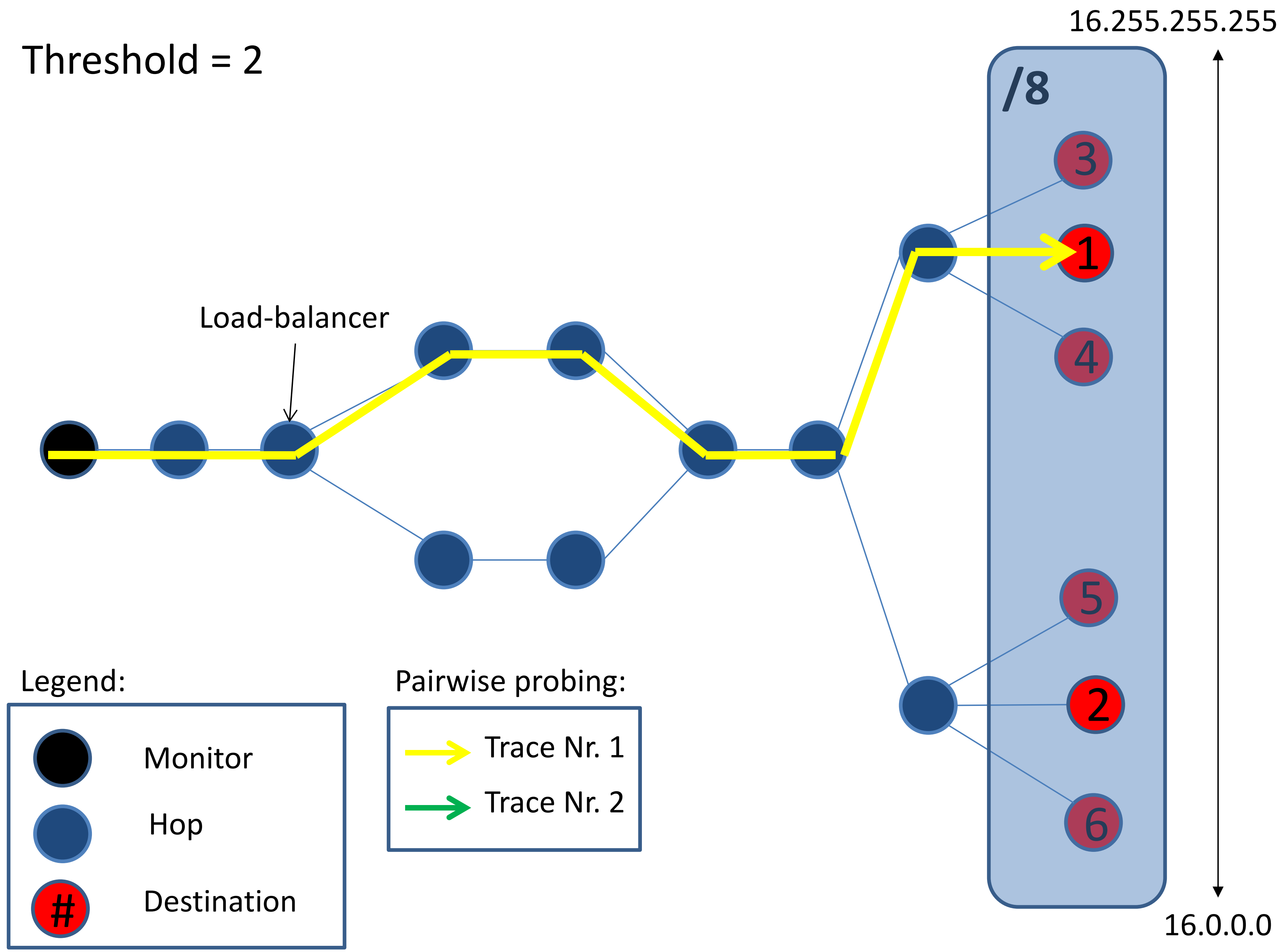


Edit Distance Metric

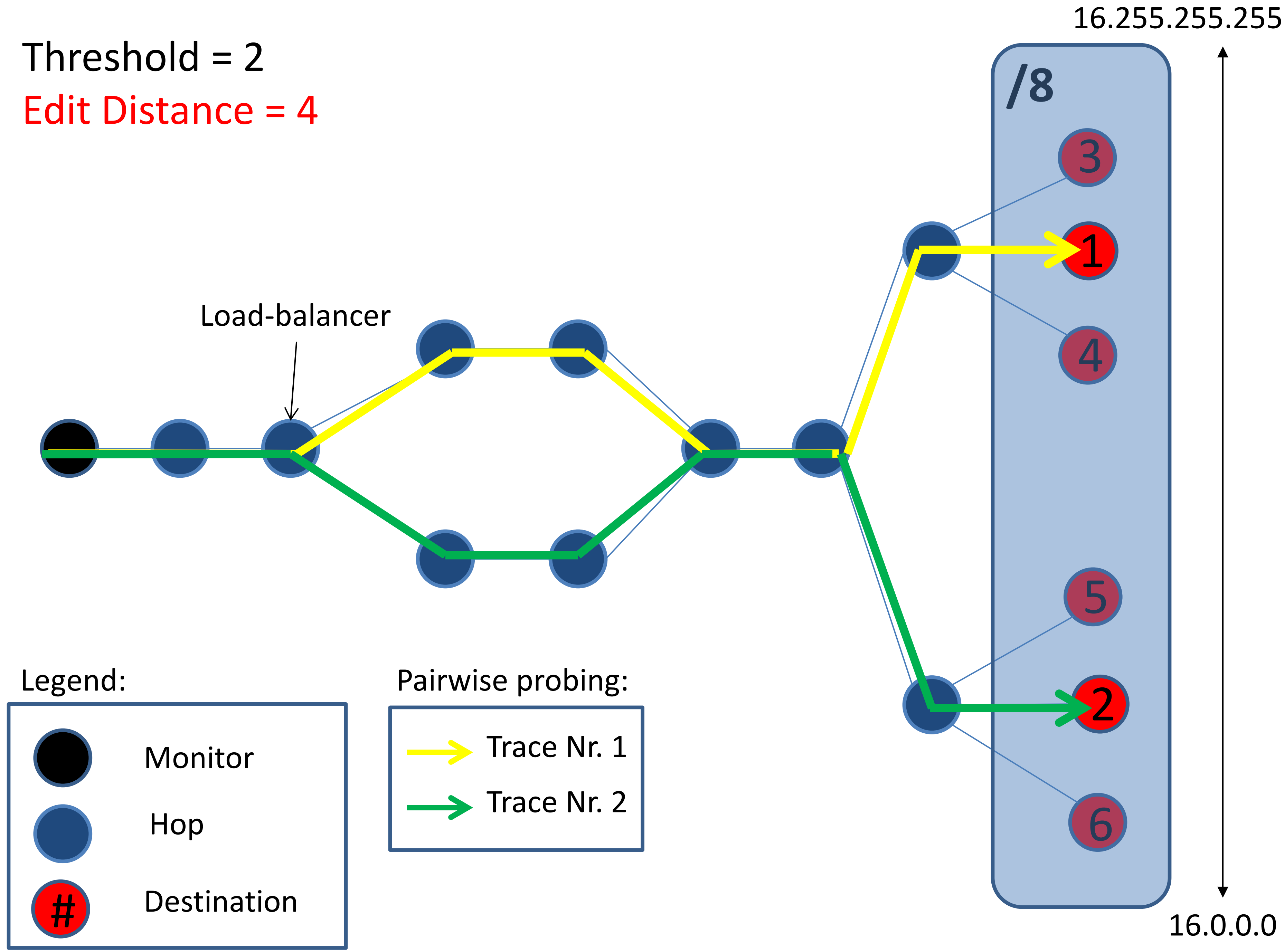
- Pairwise probing from same source.
- Compares full traces.
- Load-balancing artificially distorts ED for some paths.
- Doesn’t take benefit from previous traces to the same prefix.
- In practice: recurse all the way down to /32s.
- Note: this occurs even when using Paris-style traceroute. Paris ensures determinism over per-flow load balanced path to a given destination. SCP uses different destinations as part of its exploration algorithm.

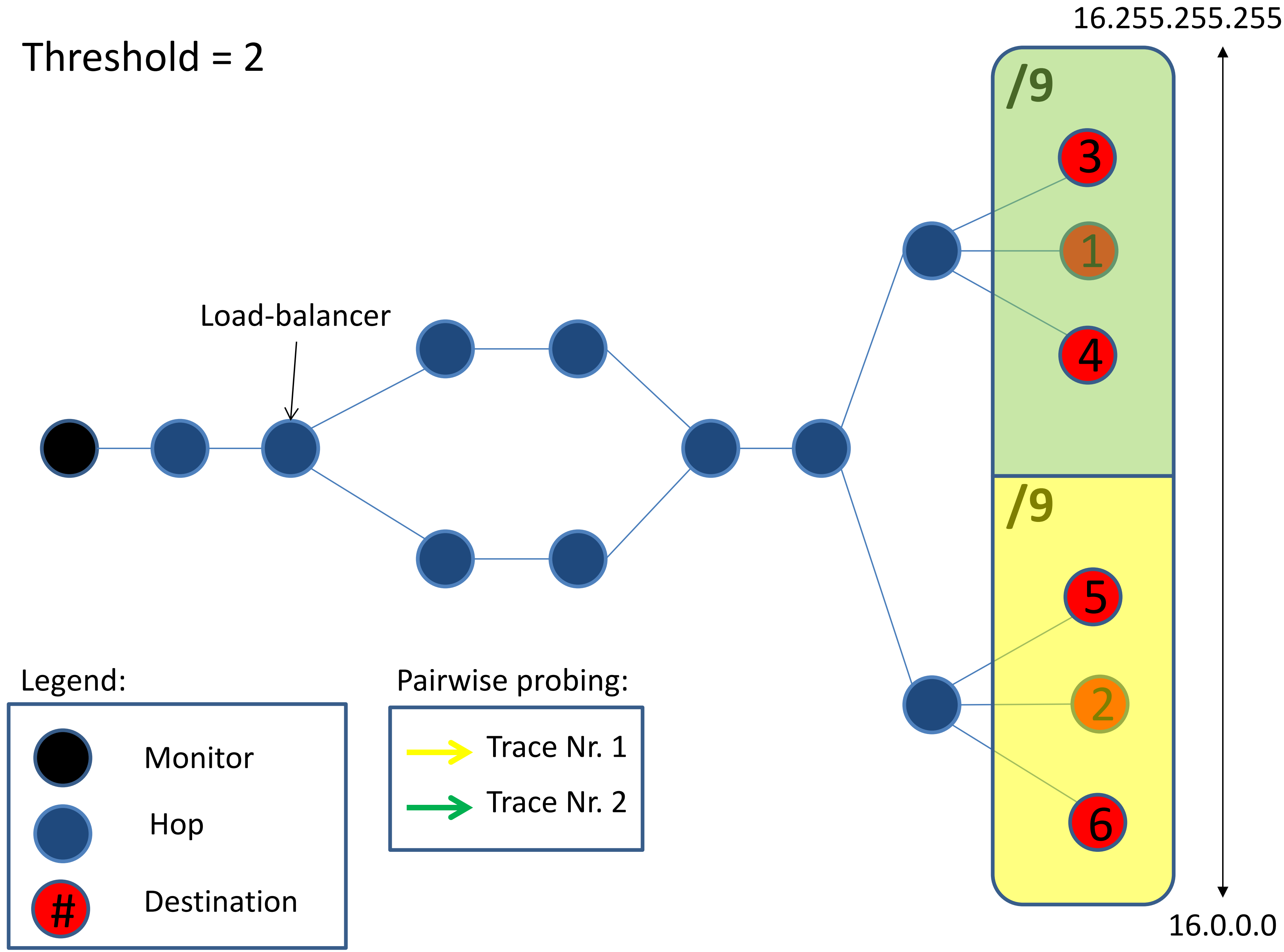


Threshold = 2
16.255.255.255
/8
3
1
4
5
2
6
16.0.0.0
Load-balancer
Legend:
Monitor
Hop
#
Destination
Pairwise probing:
Trace Nr. 1
Trace Nr. 2

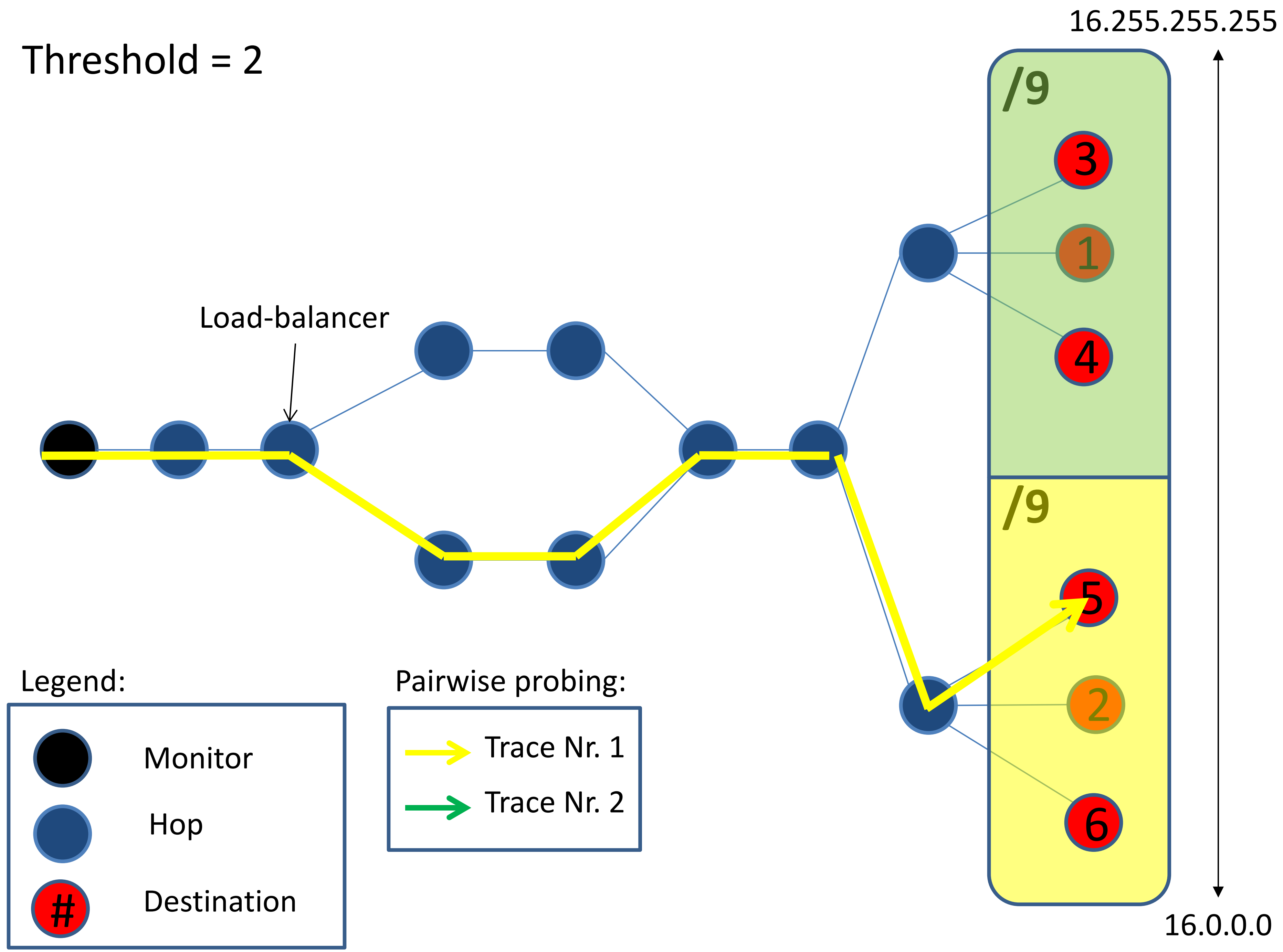


Threshold = 2
16.255.255.255
/8
3
1
4
5
2
6
16.0.0.0
Load-balancer
Legend:
Monitor
Hop
#
Destination
Pairwise probing:
Trace Nr. 1
Trace Nr. 2

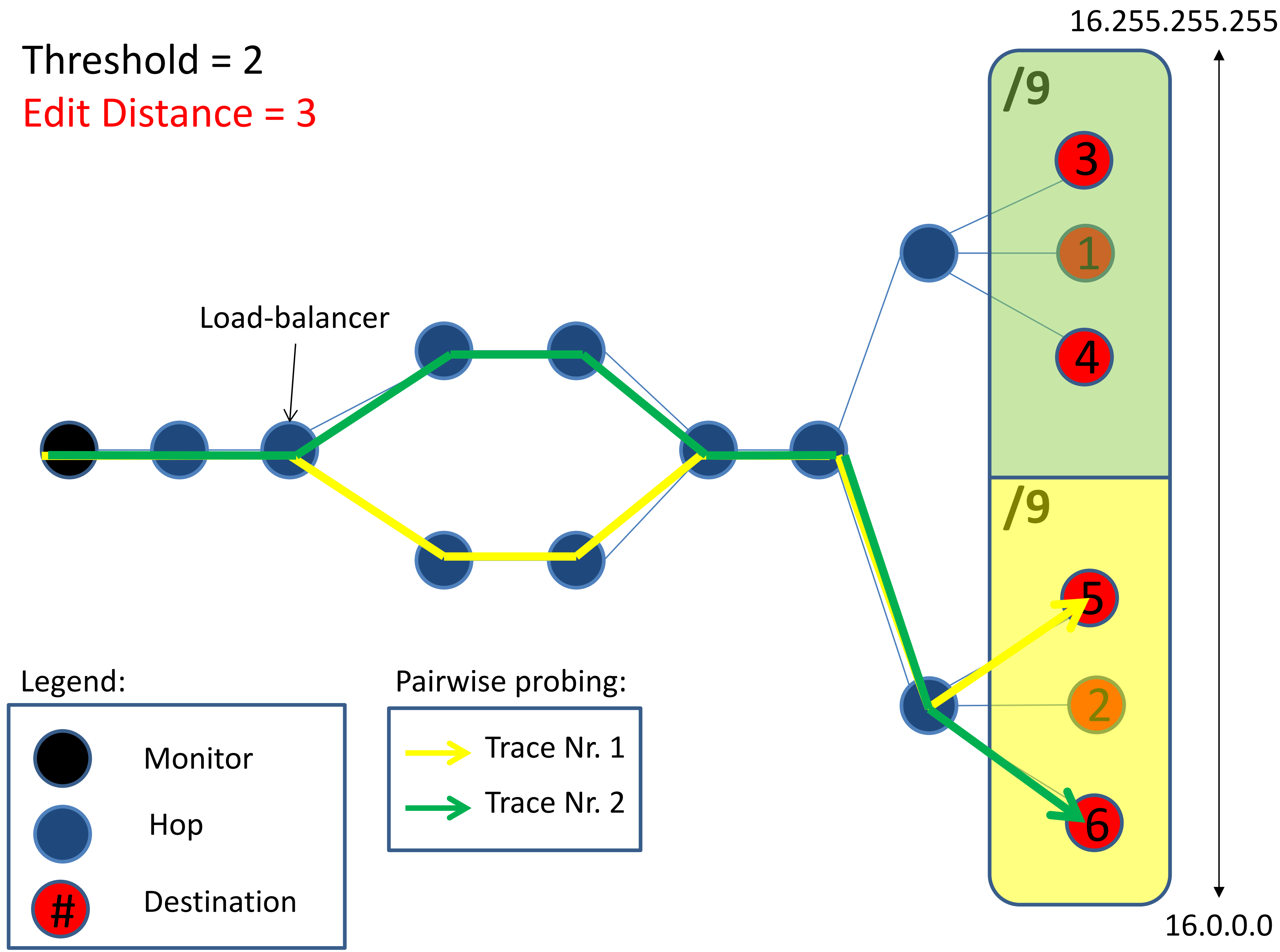

16.255.255.255
Threshold = 2
Edit Distance = 4
/8
3
1
4
Load-balancer
5
2
6
16.0.0.0
Legend:
Monitor
Hop
#
Destination
Pairwise probing:
Trace Nr. 1
Trace Nr. 2

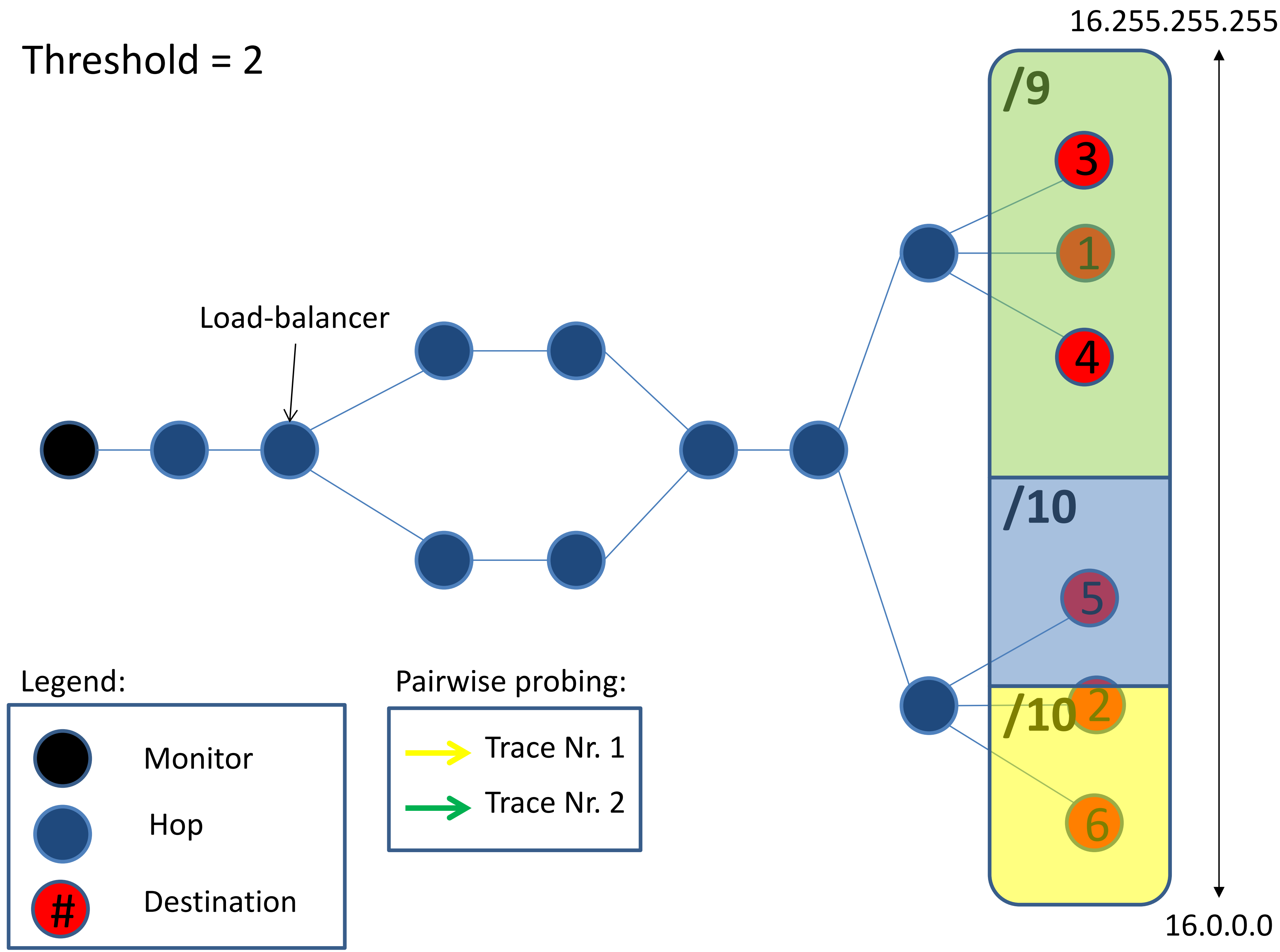


Threshold = 2
16.255.255.255
/9
3
1
4
Load-balancer
/9
5
2
6
16.0.0.0
Legend:
Monitor
Hop
#
Destination
Pairwise probing:
Trace Nr. 1
Trace Nr. 2

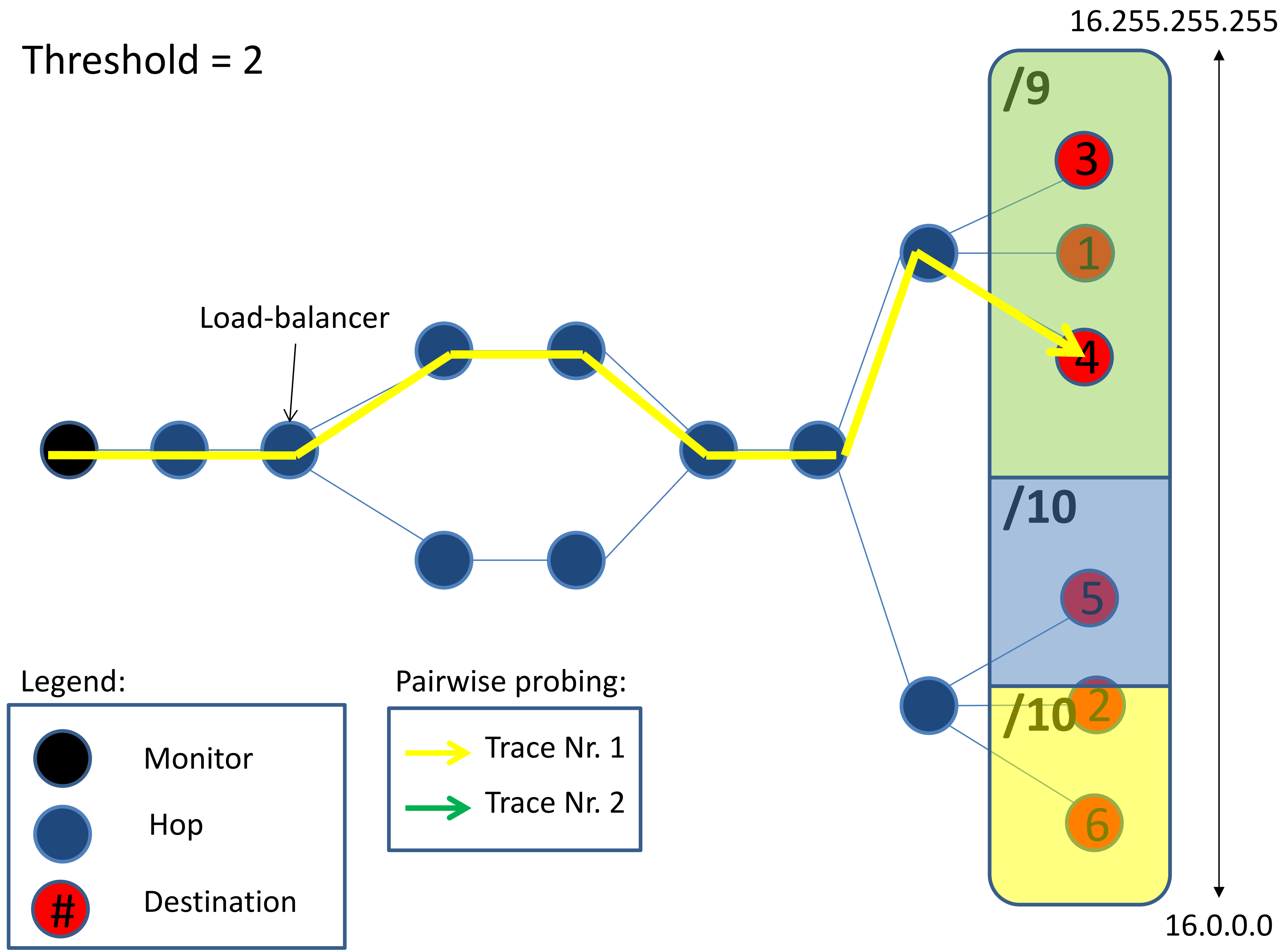


Threshold = 2
16.255.255.255
/9
3
1
4
Load-balancer
/9
5
2
6
16.0.0.0
Legend:
Monitor
Hop
#
Destination
Pairwise probing:
Trace Nr. 1
Trace Nr. 2

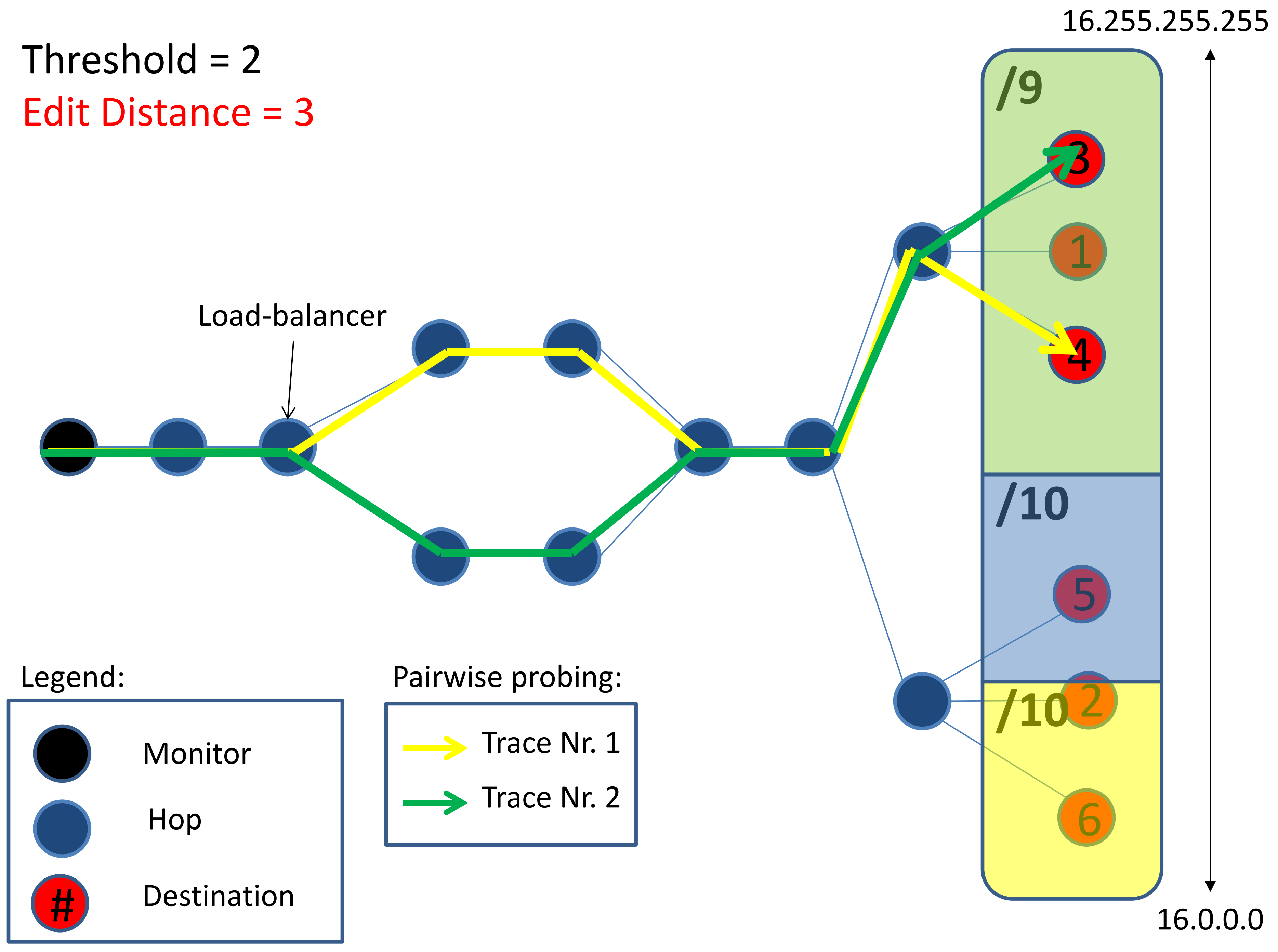

Threshold = 2
Edit Distance = 3
16.255.255.255
/9
3
1
4
Load-balancer
/9
5
2
6
16.0.0.0
Legend:
Monitor
Hop
#
Destination
Pairwise probing:
Trace Nr. 1
Trace Nr. 2



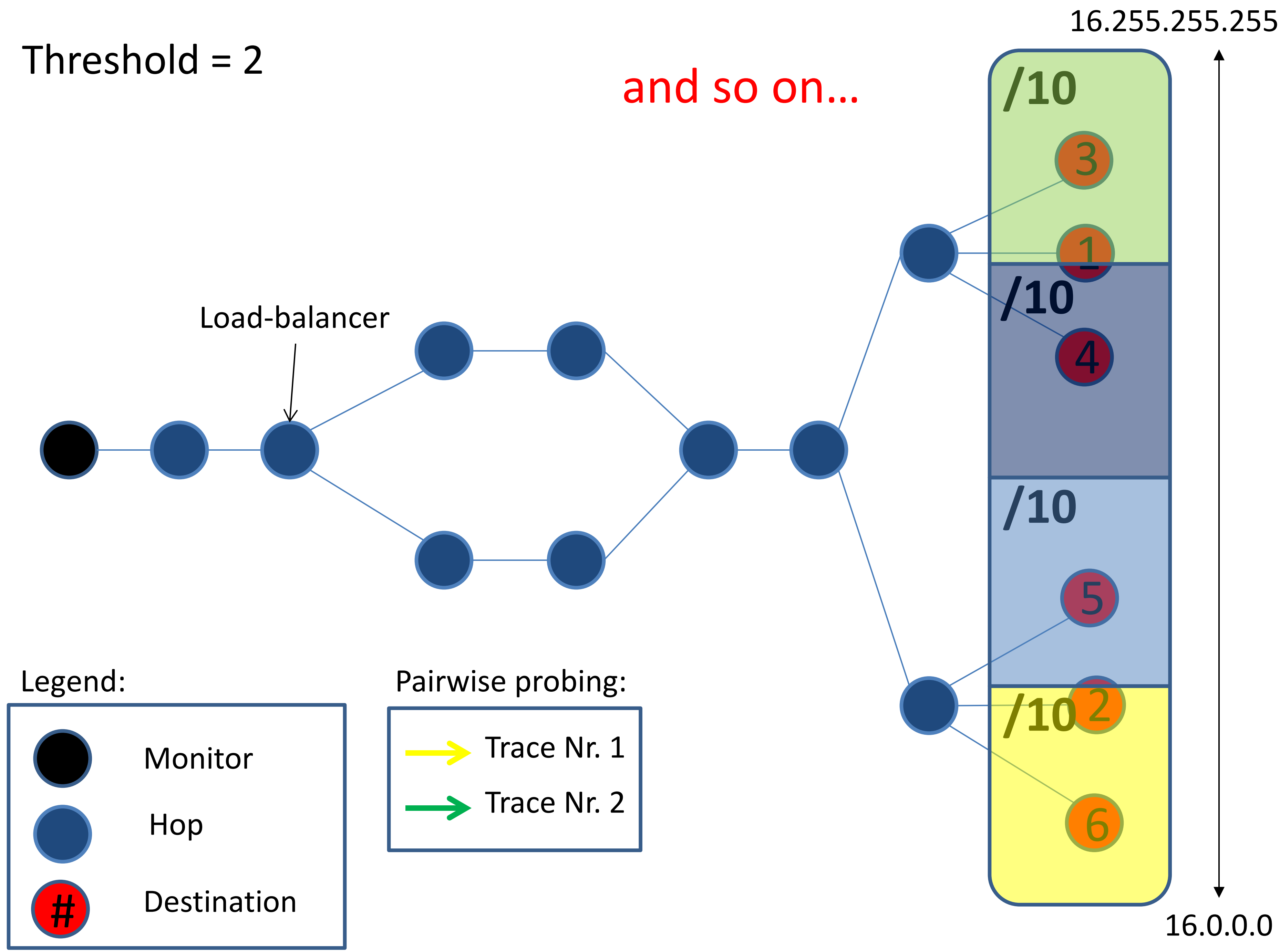
Threshold = 2
16.255.255.255
/9
3
1
4
Load-balancer
/10
5
/10
2
6
16.0.0.0
Legend:
Monitor
Hop
#
Destination
Pairwise probing:
Trace Nr. 1
Trace Nr. 2



Threshold = 2
16.255.255.255
/9
3
1
4
Load-balancer
/10
5
/10
2
6
16.0.0.0
Legend:
Monitor
Hop
#
Destination
Pairwise probing:
Trace Nr. 1
Trace Nr. 2



Threshold = 2
Edit Distance = 3
Load-balancer
16.255.255.255
/9
3
1
4
/10
5
/10
2
6
16.0.0.0
Legend:
Monitor
Hop
Destination
#
Pairwise probing:
Trace Nr. 1
Trace Nr. 2

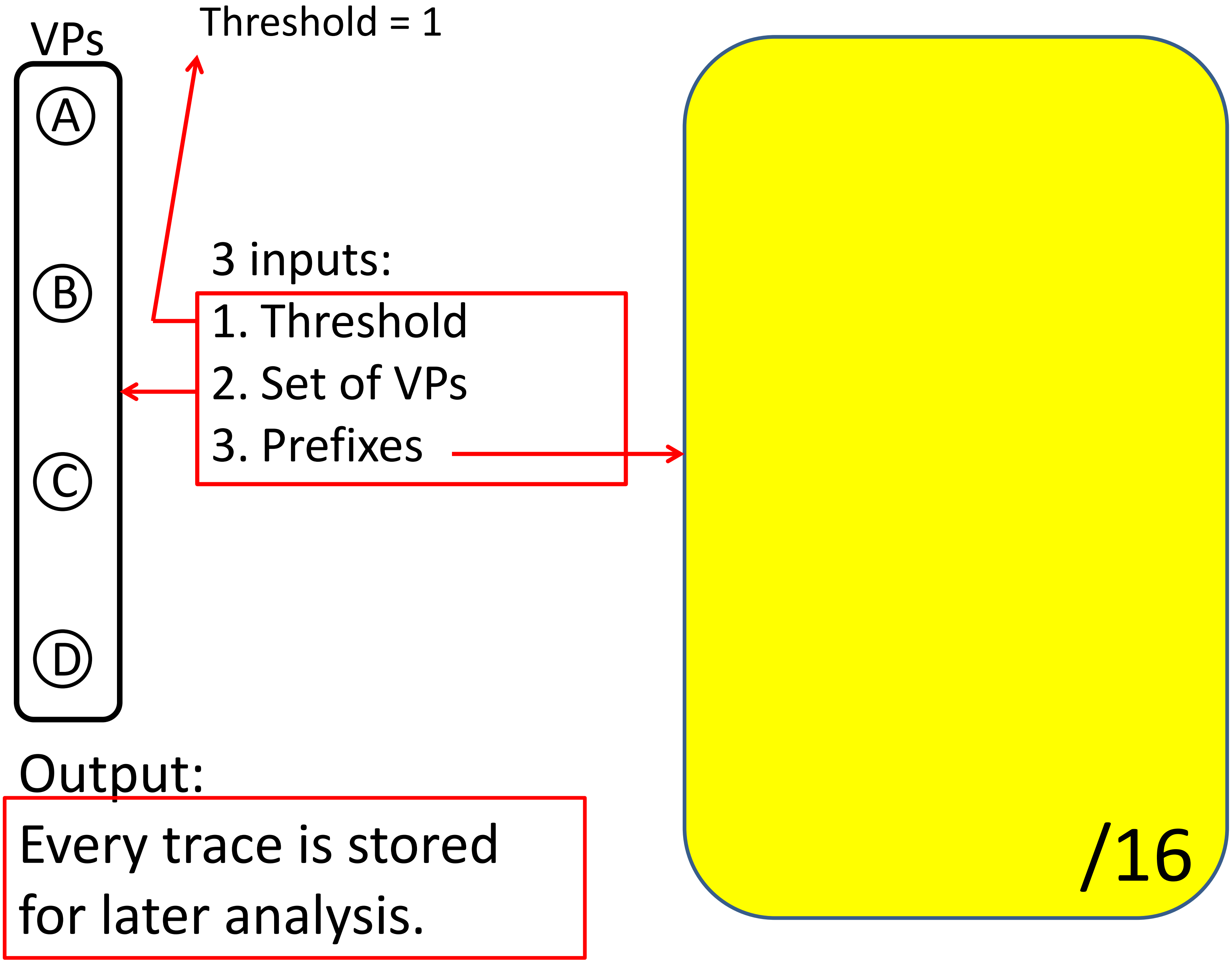

Threshold = 2
and so on...
16.255.255.255
/10
3
1
/10
4
/10
5
/10
2
6
16.0.0.0
Load-balancer
Legend:
Monitor
Hop
#
Destination
Pairwise probing:
Trace Nr. 1
Trace Nr. 2

New Interface Discovery (NID)

- Focus on destination AS:
  - SCP's objective is to discover structure within the destination AS. SCP should base its operation on new structure (edges, vertices) discovered in target AS.
- Hops inside destination AS are added into a set to which future traces will be compared.
- Number of new hops discovered inside prefix is compared to a threshold.
- Traces sent to original prefix get all their hops inside the prefix counted as newly discovered.
- Not affected by load balancing as a stopping criterion.
- Benefits from load balancing in terms of vertices and edges learning.

New Interface Discovery (NID)

- Source Distribution:
  - By focusing on target AS, we can distribute the source of each probe, as opposed to pairwise probing. Using multiple vantage points as part of SCP naturally helps discover AS ingress points.
- Number of traces per prefix:
  - For a pairwise probing, 2 traces need to be sent per prefix. This new method allows to sent as many probes as needed.
- Maintain State:
  - SCP's recursive stopping criterion should consider all traces to a destination prefix, rather than just being pair-wise.
- Future Improvements:
  - Consider number of new edges.


VPs
Threshold = 1
A
B
C
D
3 inputs:
1. Threshold
2. Set of VPs
3. Prefixes
/16
Output:
Every trace is stored
for later analysis.

Threshold = 1

VPs

Ⓐ

Ⓑ

Ⓒ

Ⓓ

Monitors/Destinations Pairs Queue:

Discovered Prefix-Vertices List:

/16

Threshold = 1

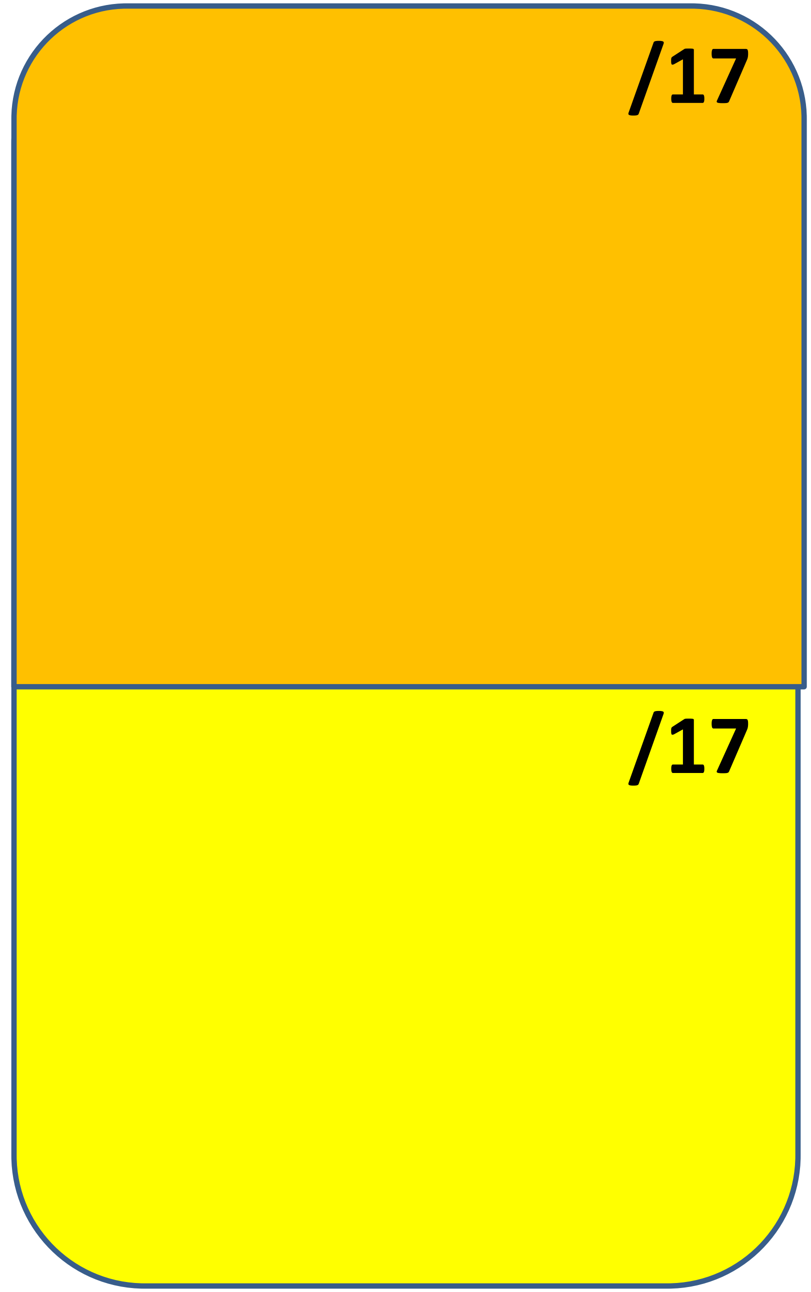
VPs

Ⓐ

Ⓑ

Ⓒ

Ⓓ



Monitors/Destinations Pairs Queue:

Discovered Prefix-Vertices List:

Threshold = 1

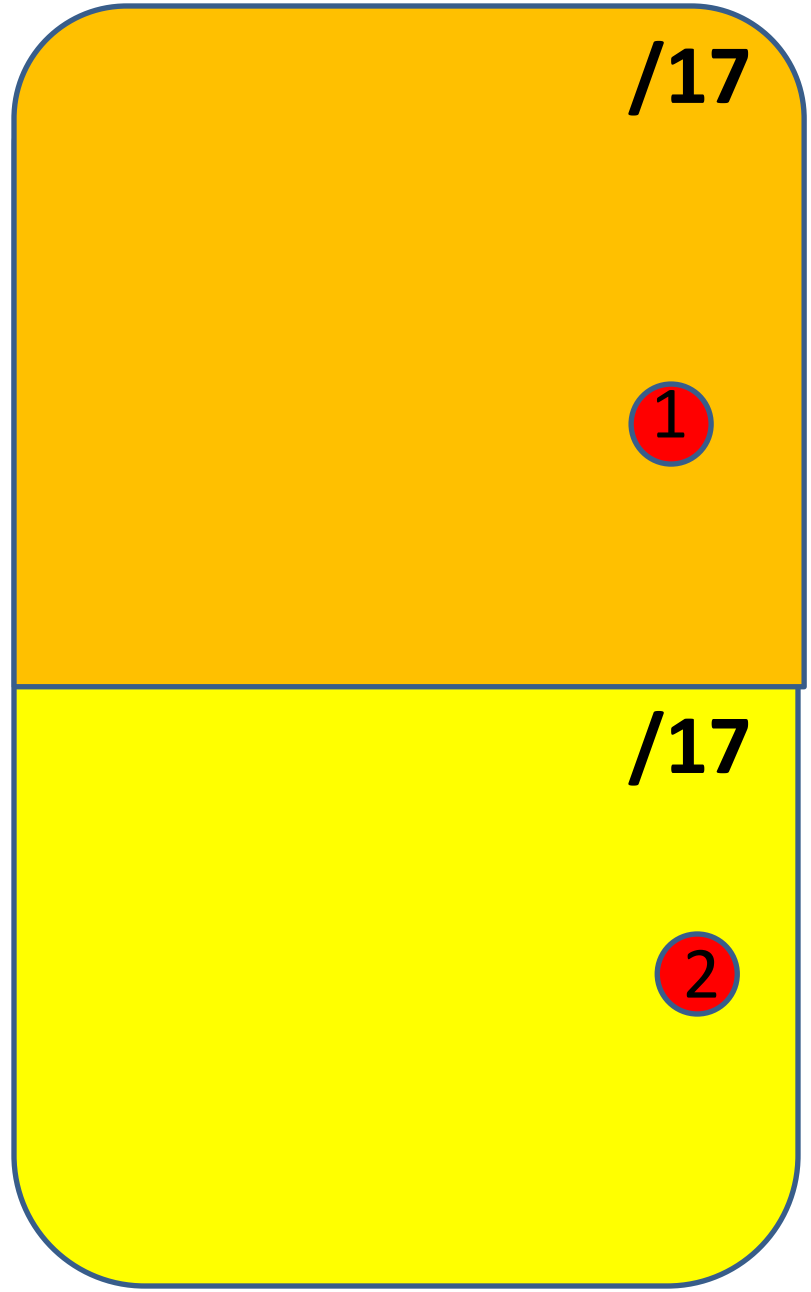
VPs

Ⓐ

Ⓑ

Ⓒ

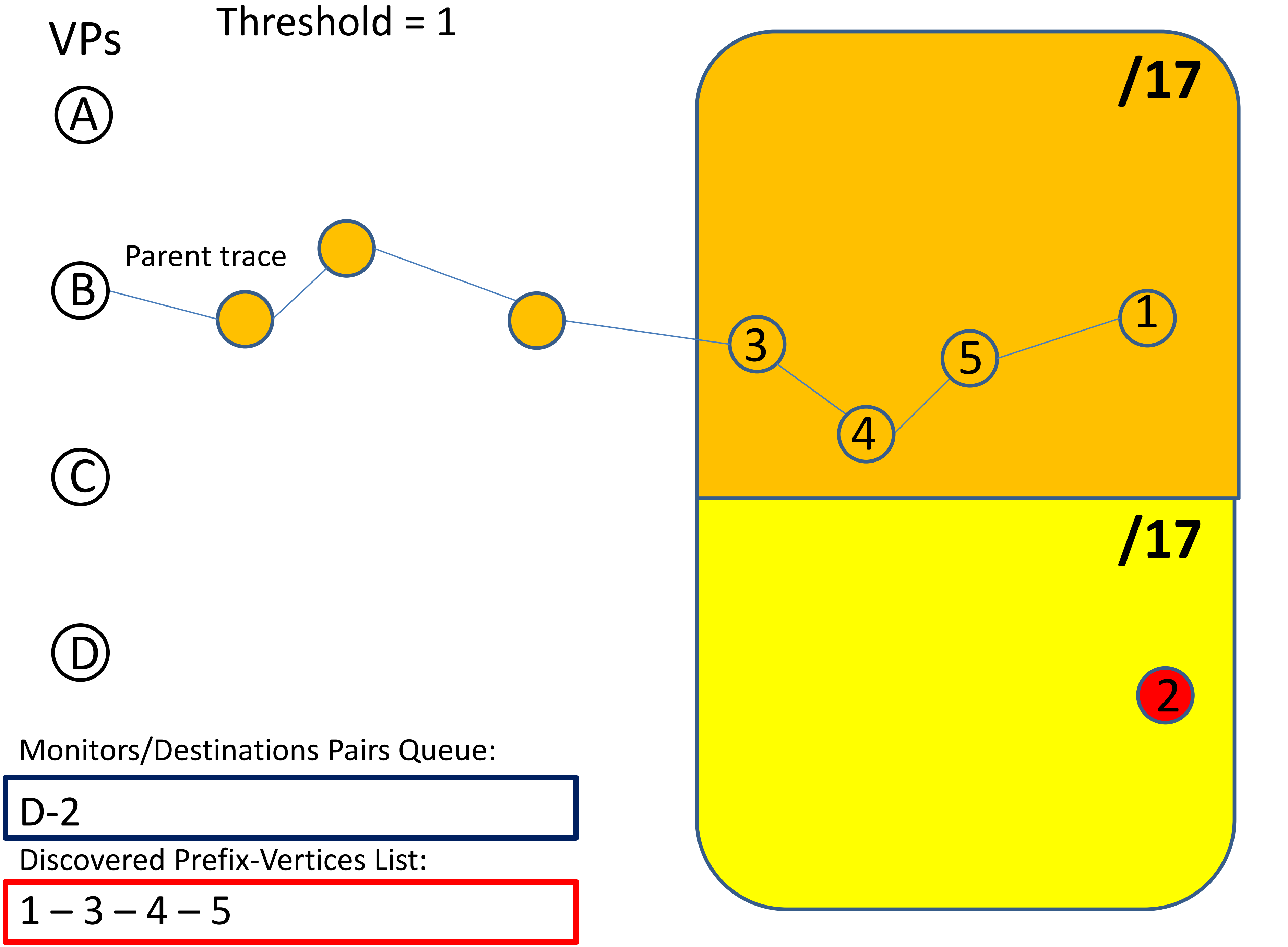
Ⓓ



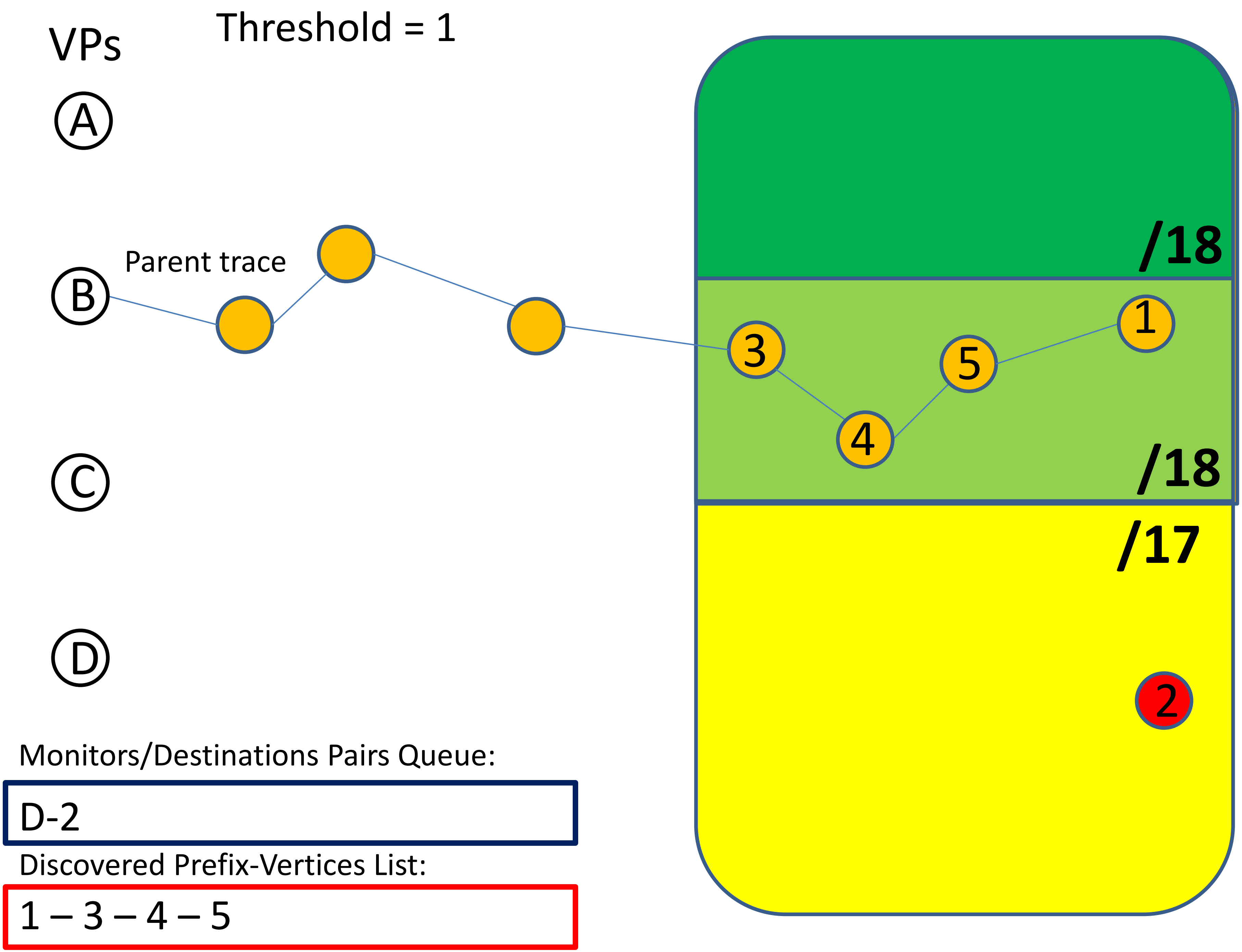
Monitors/Destinations Pairs Queue:

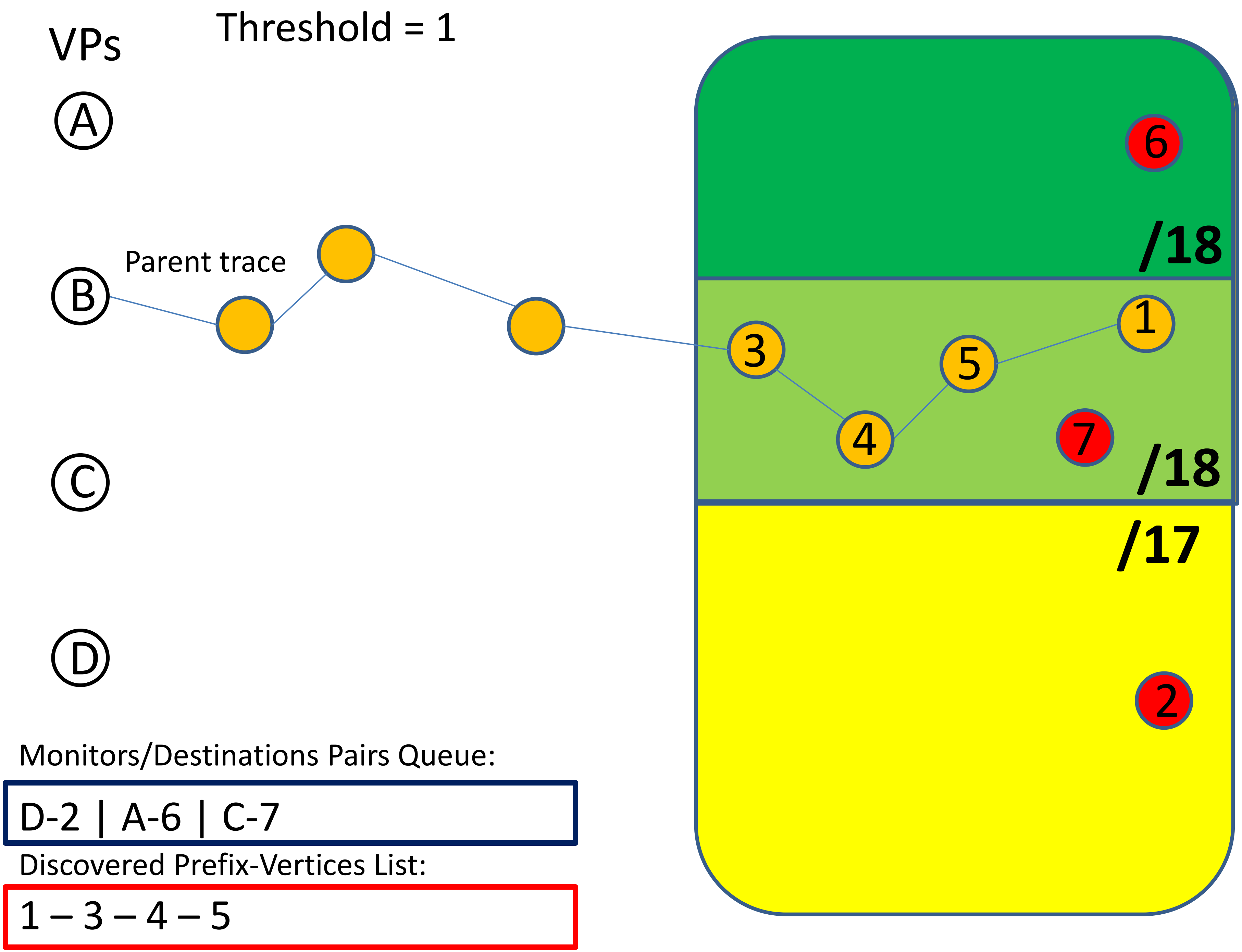
B-1 | D-2

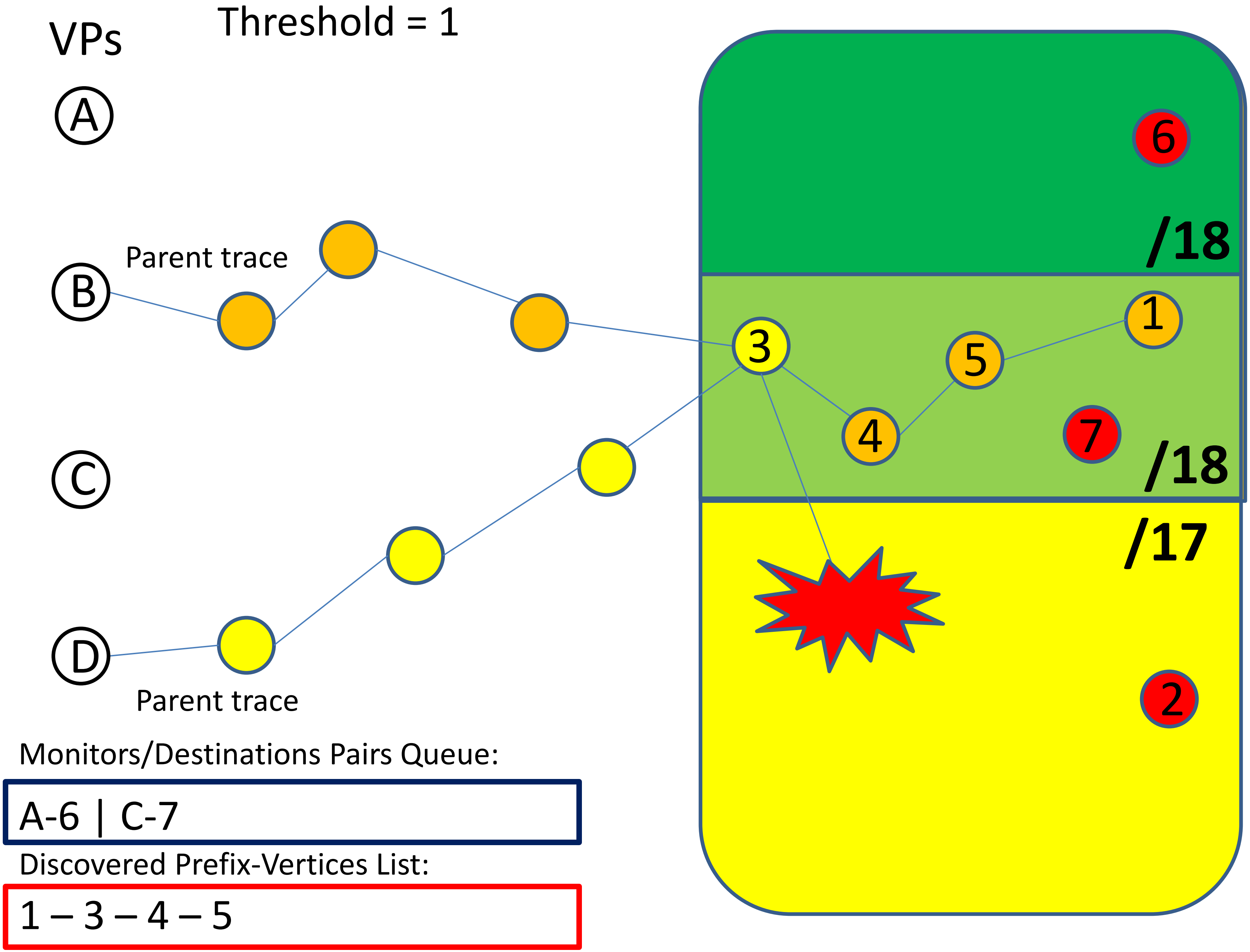
Discovered Prefix-Vertices List:



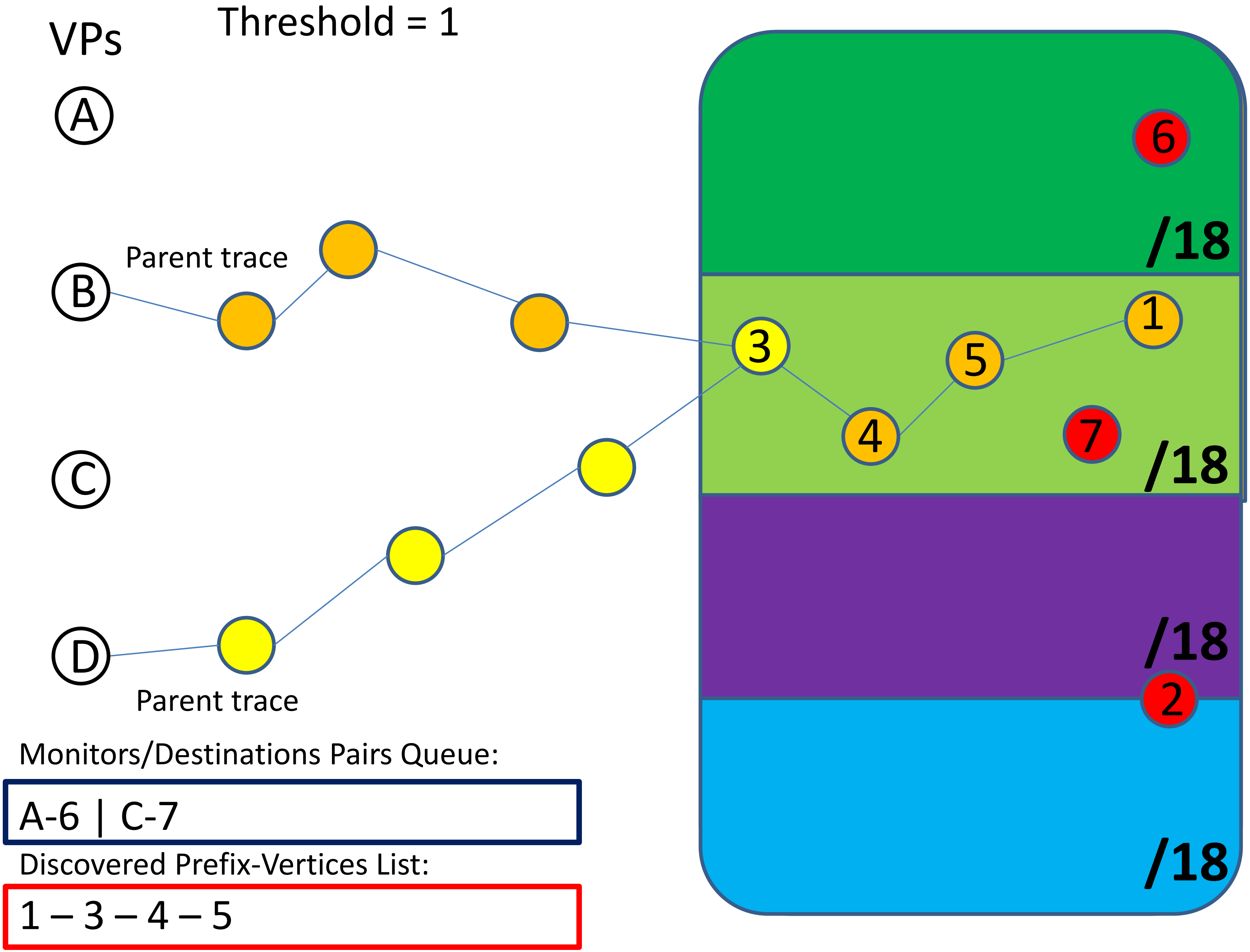
VPs
Threshold = 1
A
B
C
D
Parent trace
/17
3
4
5
1
/17
2
Monitors/Destinations Pairs Queue:
D-2
Discovered Prefix-Vertices List:
1 – 3 – 4 – 5

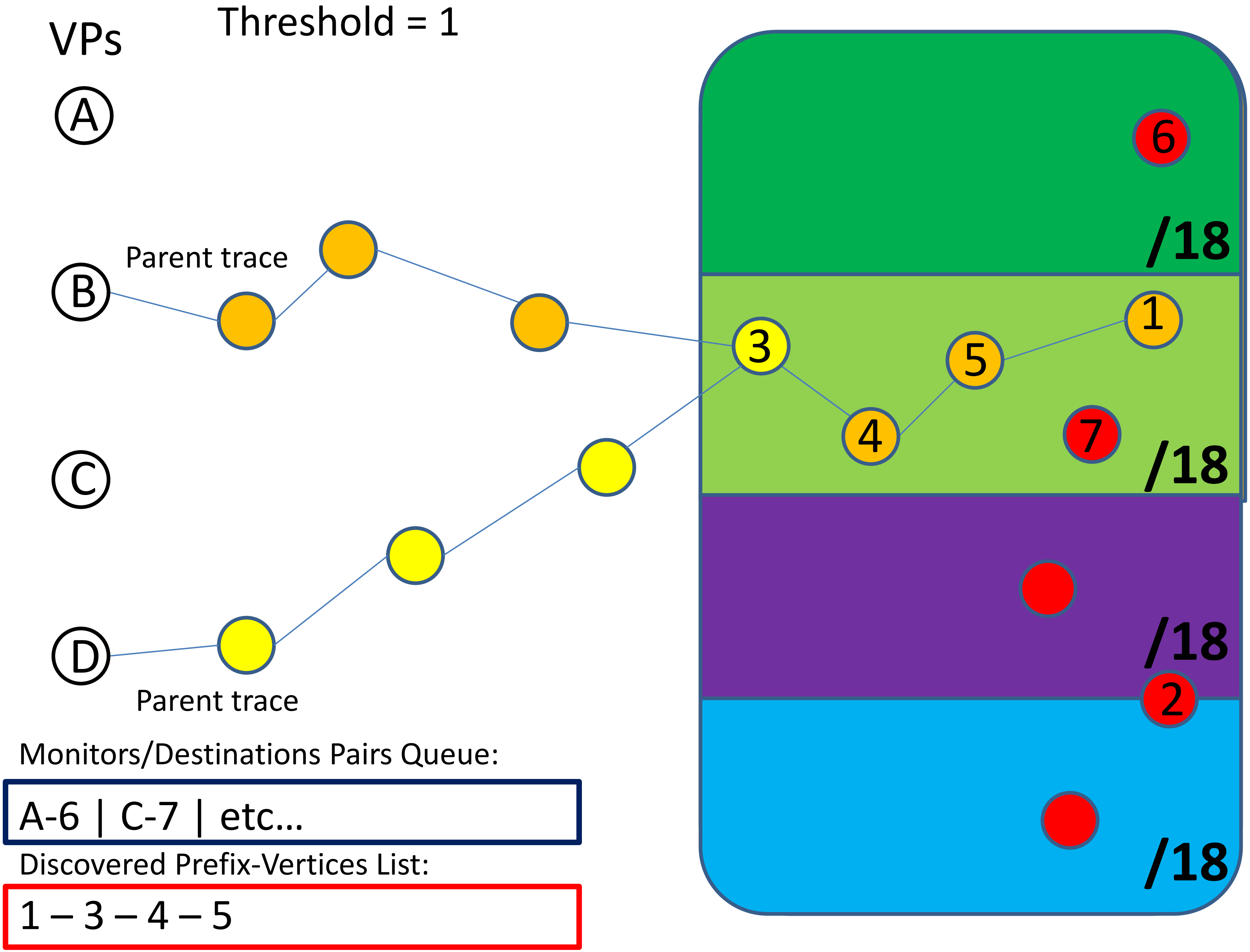

VPs
Threshold = 1
A
B
C
D
Parent trace
/18
/18
/17
3
4
5
1
2
Monitors/Destinations Pairs Queue:
D-2
Discovered Prefix-Vertices List:
1 – 3 – 4 – 5


Threshold = 1
VPs
A
B
C
D
Parent trace
/18
/18
/17
1
2
3
4
5
6
7
Monitors/Destinations Pairs Queue:
D-2 | A-6 | C-7
Discovered Prefix-Vertices List:
1 – 3 – 4 – 5

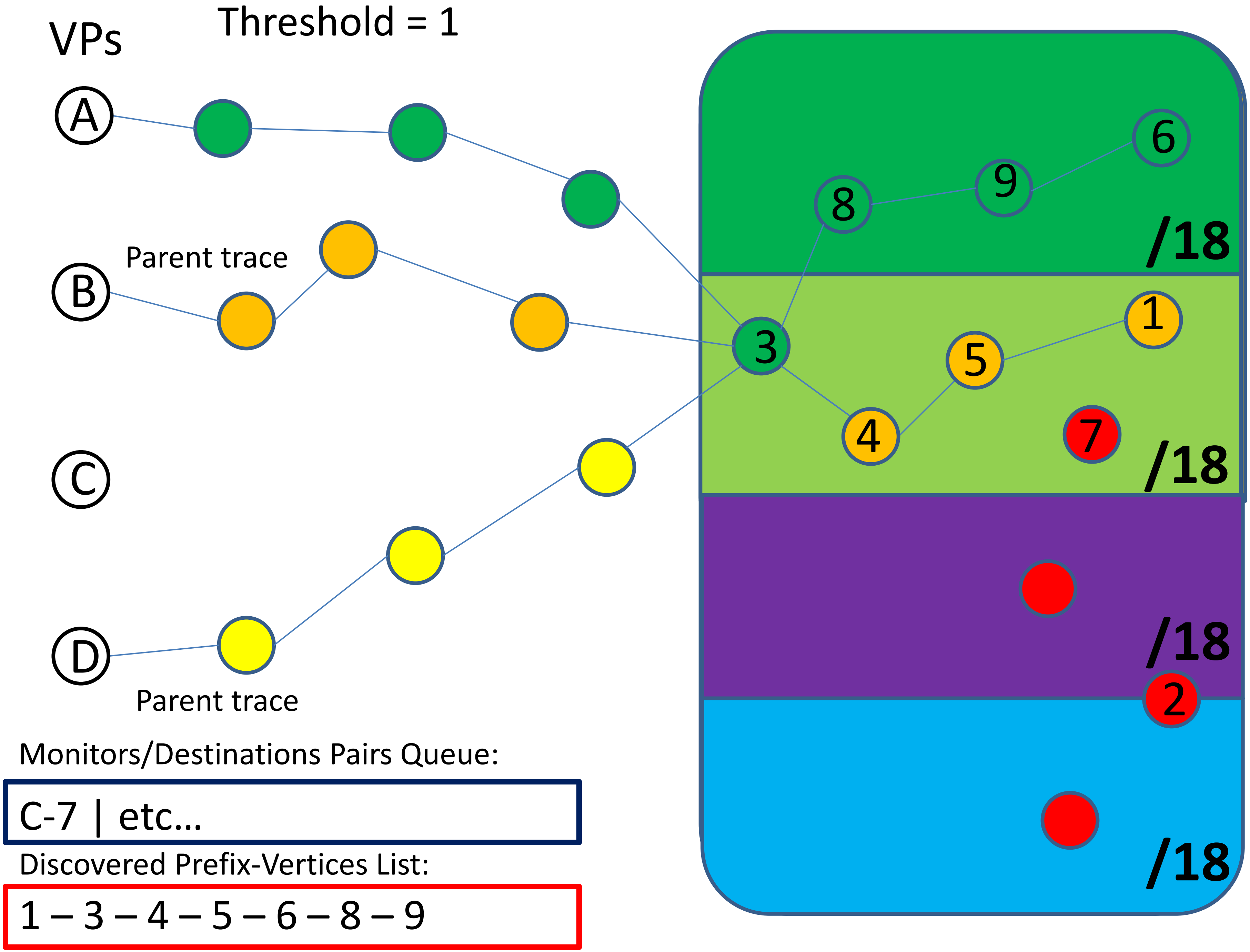


VPs
Threshold = 1
A
B
C
D
Parent trace
Parent trace
6
/18
1
3
5
4
7
/18
/17
2
Monitors/Destinations Pairs Queue:
A-6 | C-7
Discovered Prefix-Vertices List:
1 – 3 – 4 – 5

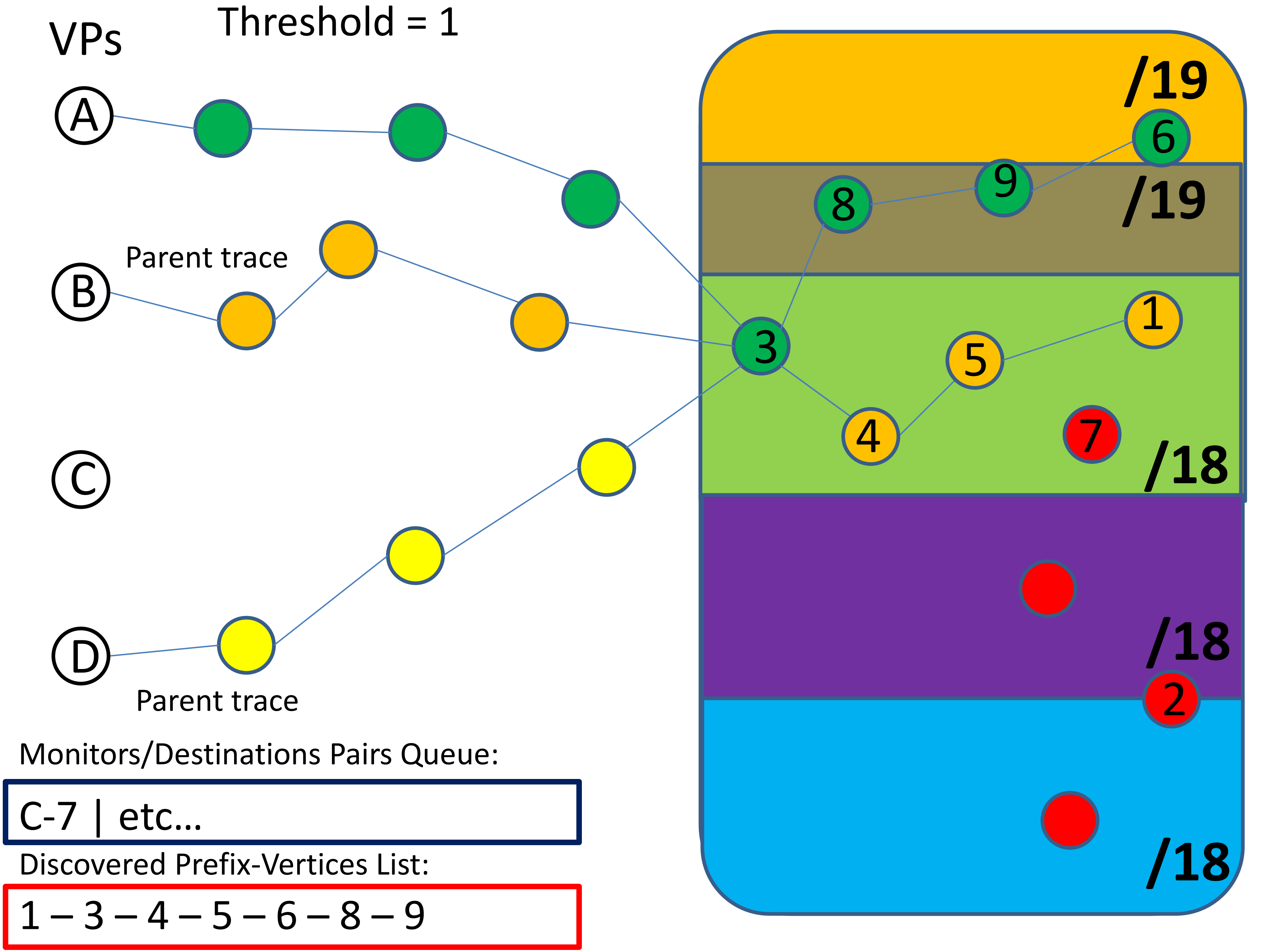

VPs
Threshold = 1
A
B
C
D
Parent trace
Parent trace
6
/18
1
3
5
4
7
/18
/18
2
/18
Monitors/Destinations Pairs Queue:
A-6 | C-7
Discovered Prefix-Vertices List:
1 – 3 – 4 – 5



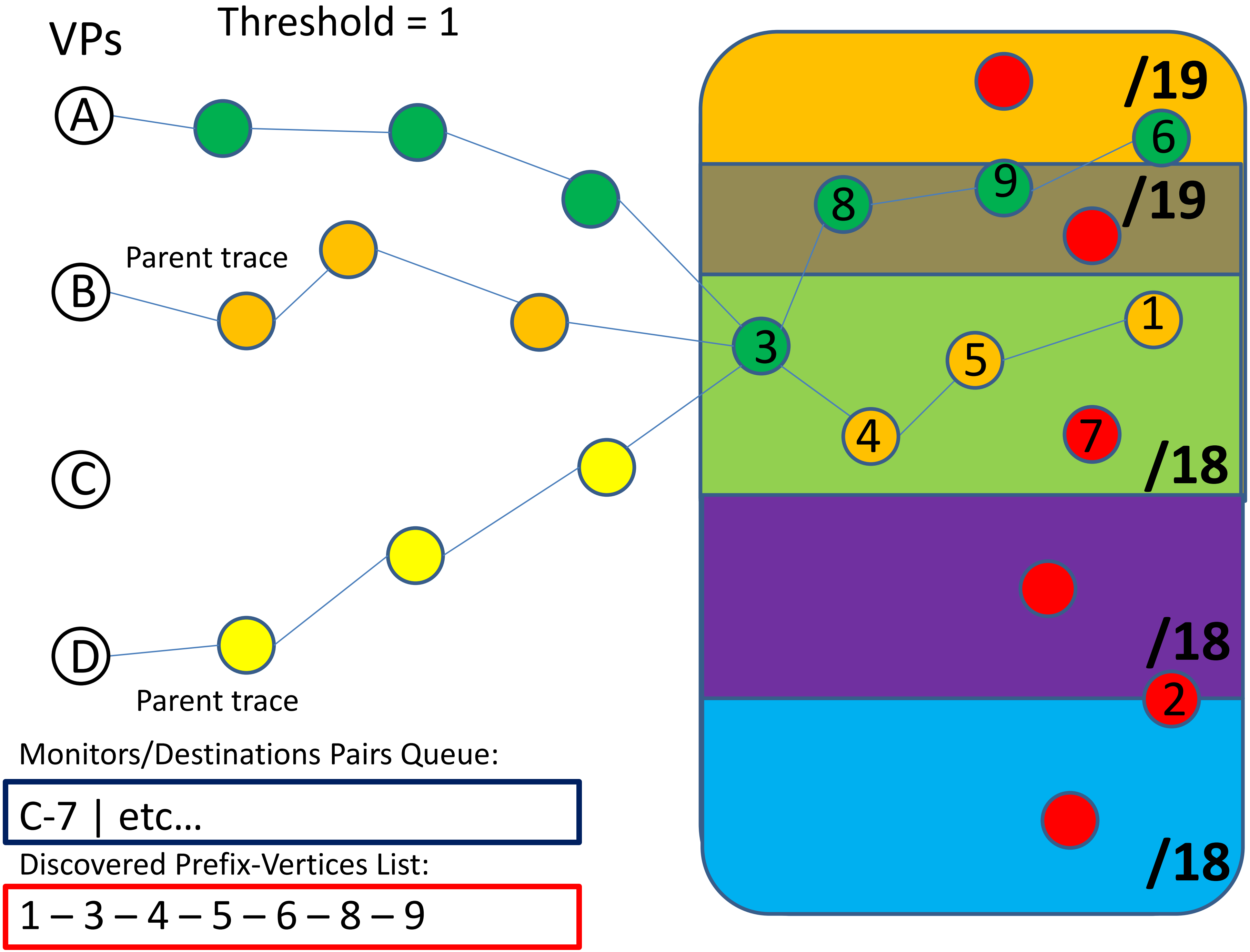
VPs
Threshold = 1
A
B
C
D
Parent trace
Parent trace
6
/18
3
5
1
4
7
/18
/18
2
/18
Monitors/Destinations Pairs Queue:
A-6 | C-7 | etc...
Discovered Prefix-Vertices List:
1 – 3 – 4 – 5

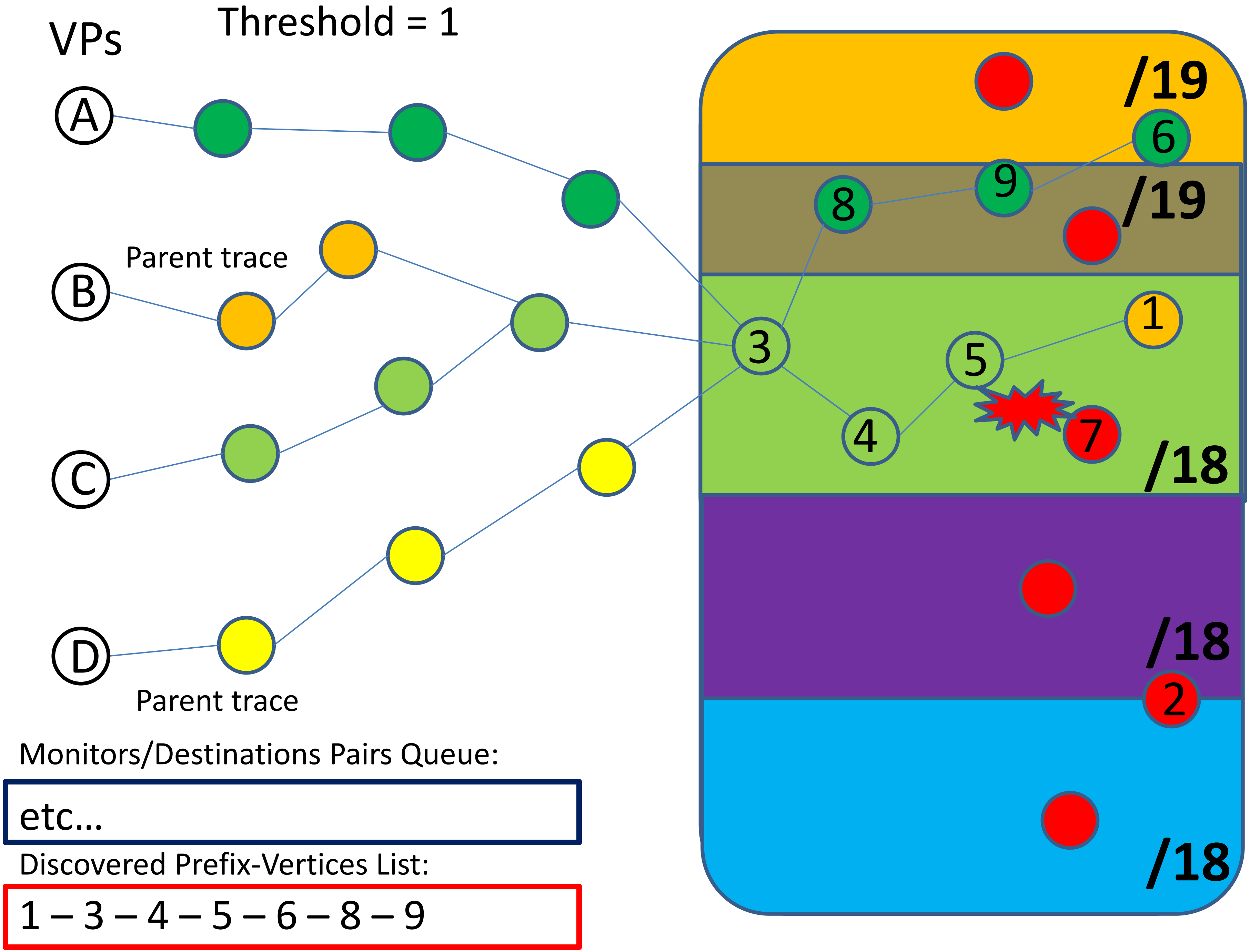


VPs
Threshold = 1
A
B
C
D
Parent trace
Parent trace
3
8
9
6
1
5
4
7
2
/18
/18
/18
/18
Monitors/Destinations Pairs Queue:
C-7 | etc…
Discovered Prefix-Vertices List:
1 – 3 – 4 – 5 – 6 – 8 – 9

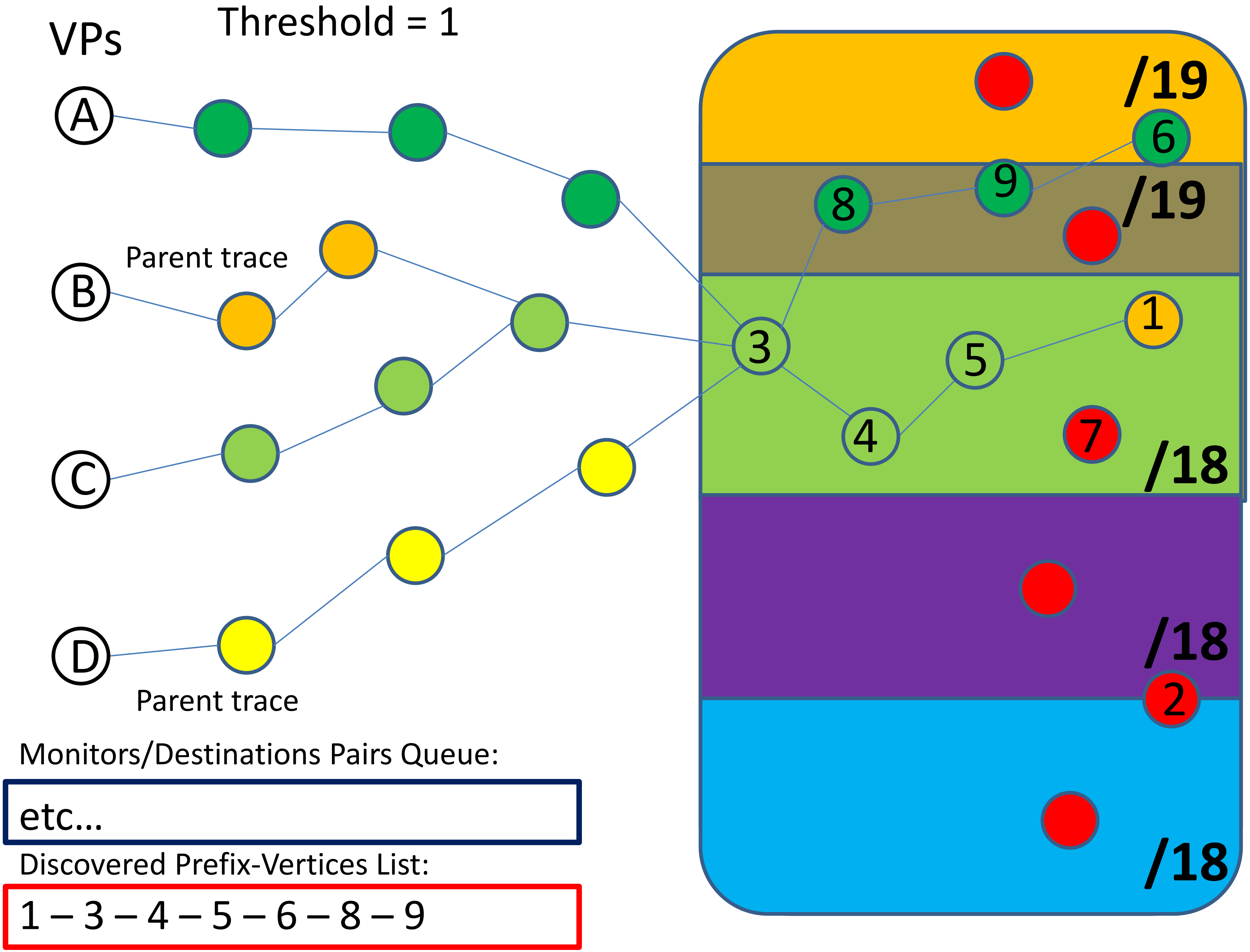


VPs
Threshold = 1
A
B
C
D
Parent trace
Parent trace
/19
/19
/18
/18
/18
1
2
3
4
5
6
7
8
9
Monitors/Destinations Pairs Queue:
C-7 | etc…
Discovered Prefix-Vertices List:
1 – 3 – 4 – 5 – 6 – 8 – 9


VPs
Threshold = 1
A
B
C
D
Parent trace
Parent trace
/19
/19
/18
/18
/18
1
2
3
4
5
6
7
8
9
Monitors/Destinations Pairs Queue:
C-7 | etc...
Discovered Prefix-Vertices List:
1 – 3 – 4 – 5 – 6 – 8 – 9



VPs
Threshold = 1
A
B
C
D
Parent trace
Parent trace
/19
/19
/18
/18
/18
1
2
3
4
5
6
7
8
9
Monitors/Destinations Pairs Queue:
etc...
Discovered Prefix-Vertices List:
1 – 3 – 4 – 5 – 6 – 8 – 9


Threshold = 1
VPs
A
B
C
D
Parent trace
Parent trace
/19
/19
/18
/18
/18
1
2
3
4
5
6
7
8
9
Monitors/Destinations Pairs Queue:
etc...
Discovered Prefix-Vertices List:
1 – 3 – 4 – 5 – 6 – 8 – 9

NID Pseudo Code

- Input:
  - p/m: Destination prefix / mask
  - M: Set of monitors
  - τ: Threshold of new interfaces
- Output:
  - T: Set of path traces; initially empty

Algorithm 1: NID(p/m, M, τ)

```
1: I = empty set   // global variable
2: T = empty set // global variable
3: SCP(p/m, M, τ)
```

Algorithm 2: SCP(p/m, M, τ)

A = ASN(p/m)

(d_1, d_2) = determine_targets(p/m) // LCP algorithm

(m_1, m_2) = assign_monitors(M, p/m) // Random, VPS, IPS, Max.

t_1 = trace(m_1, d_1)

t_2 = trace(m_2, d_2)

$T = T \cup (t_1 \cup t_2)$

$t_1' = \{$interface *i* in $t_1 \mid$ ASN(*i*) = A$\}$

$t_2' = \{$interface *i* in $t_2 \mid$ ASN(*i*) = A$\}$

for j = 1 **to** 2 **do**

  if t_j' is parent_trace **then**

    if length$(t_j') > 0$ **then**

      $I = I \cup t_j'$

      SCP(p/(m + 1), M, τ)

  else:

    if $\left|t_j' - I\right| \geq \tau$ **then**

      $I = I \cup t_j'$

      SCP(p/(m + 1), M, τ)

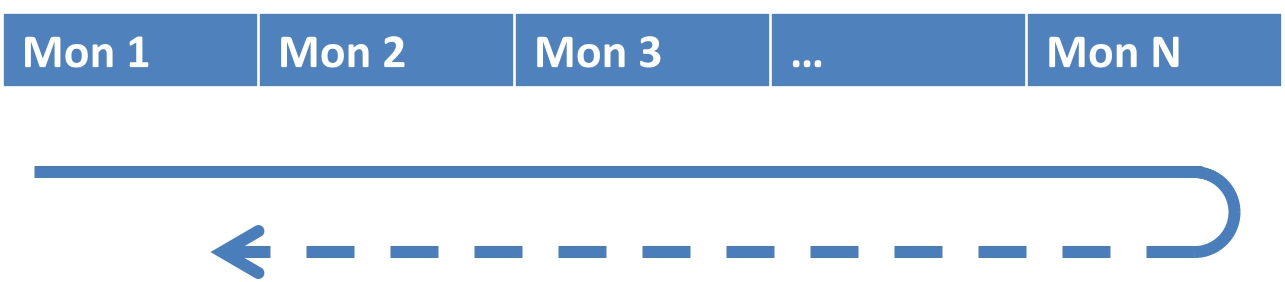
Random Monitor Assignment

- Each destinations gets a monitor assigned randomly.
- Uniform Distribution.
- Random assignment of destinations to vantage points is wasteful.
  - E.g. empirically, the 16 /24’s in a /20 prefix are hit on average by 12 unique VPs.
  - 2 paths from 2 different monitors towards the same prefix might be too similar, and therefore it is inconvenient to assign the monitors consecutively.

Vantage Point Spreading

- New destinations are assigned to VPs not yet used for probing the original BGP prefix
- Uniform distribution of VPs when more destinations than VPs.
- Implementation: build list of all monitors in random order for each prefix. Round robin for assignment.

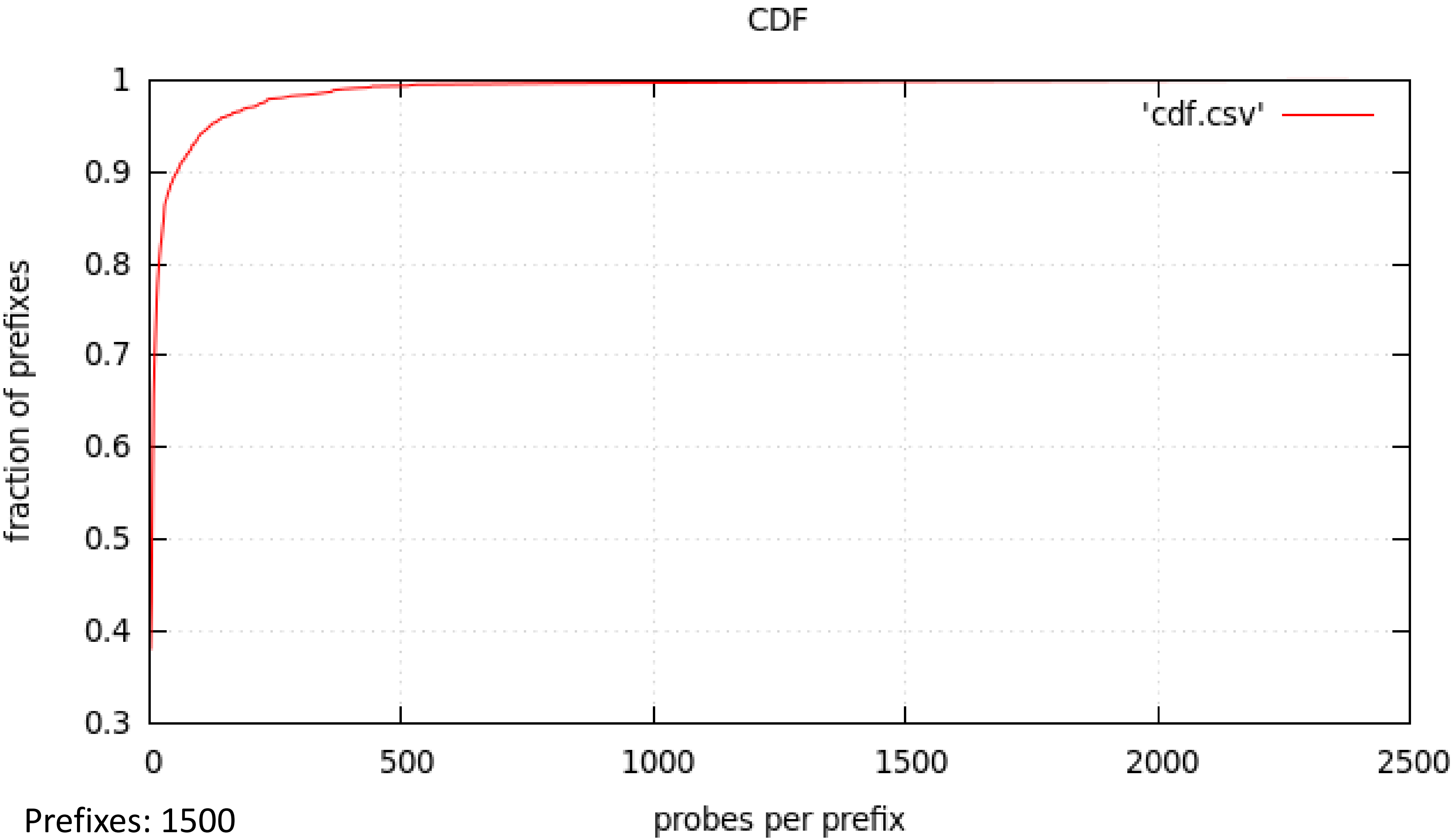
e.g.
Some /16 prefix:



Vantage Point Spreading

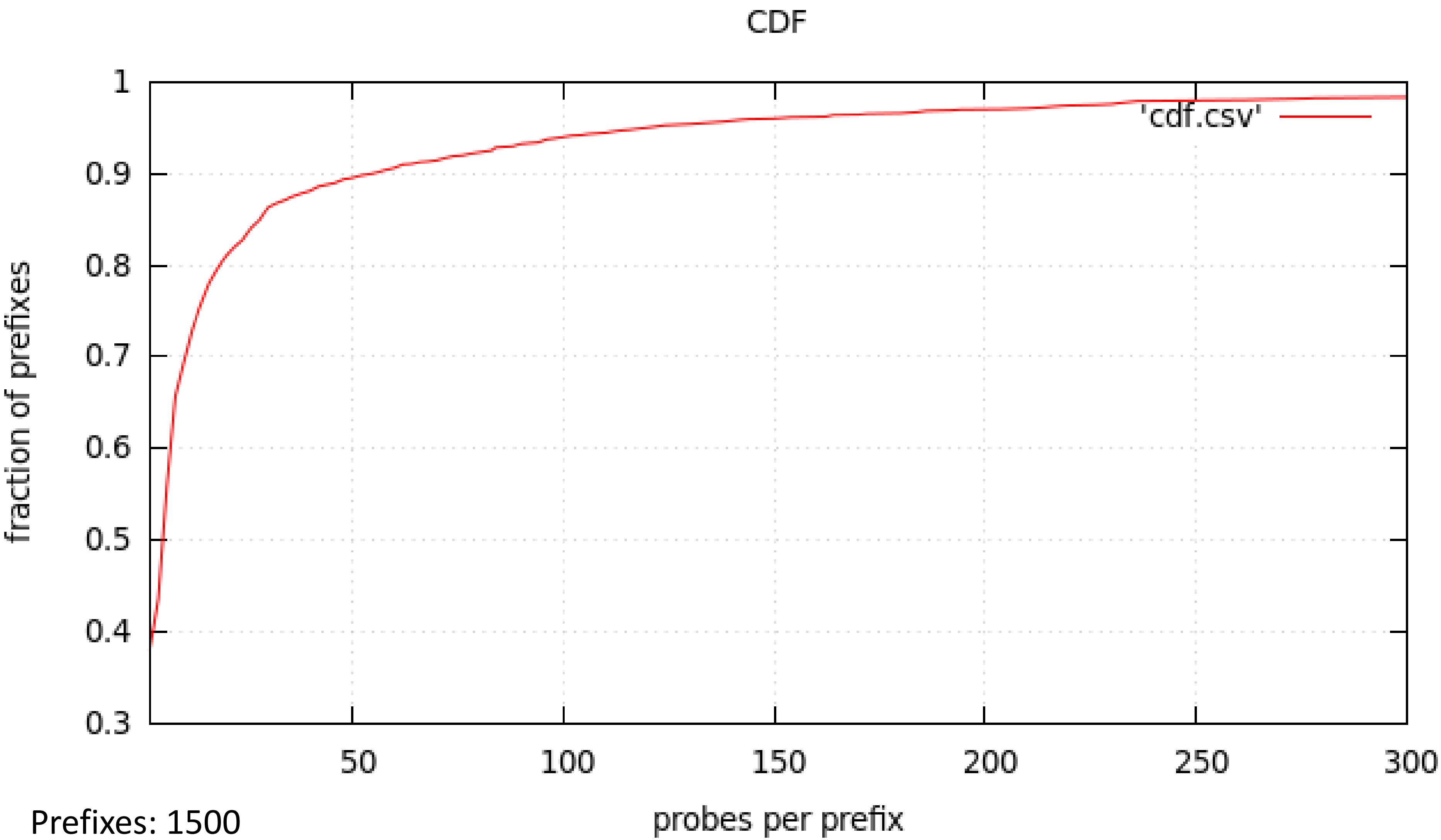
- Issues:
  - Early termination.
  - Doesn’t maximize the benefit of each probe in terms of new vertices learnt.
  - 90% of prefixes send less probes than the number of available monitors.

Number of Monitors used per Prefix



Prefixes: 1500
Monitors: 60
Method: Random

Zooming in ….

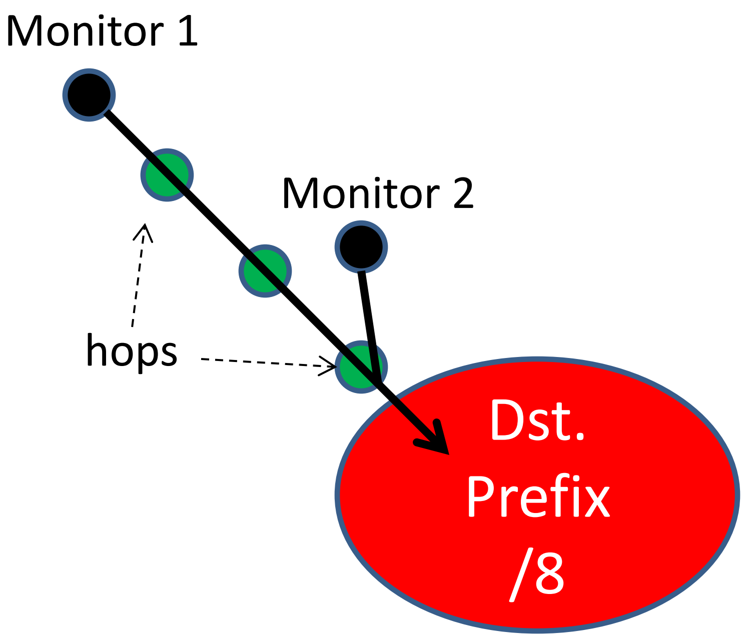


Prefixes: 1500
Monitors: 60
Method: Random

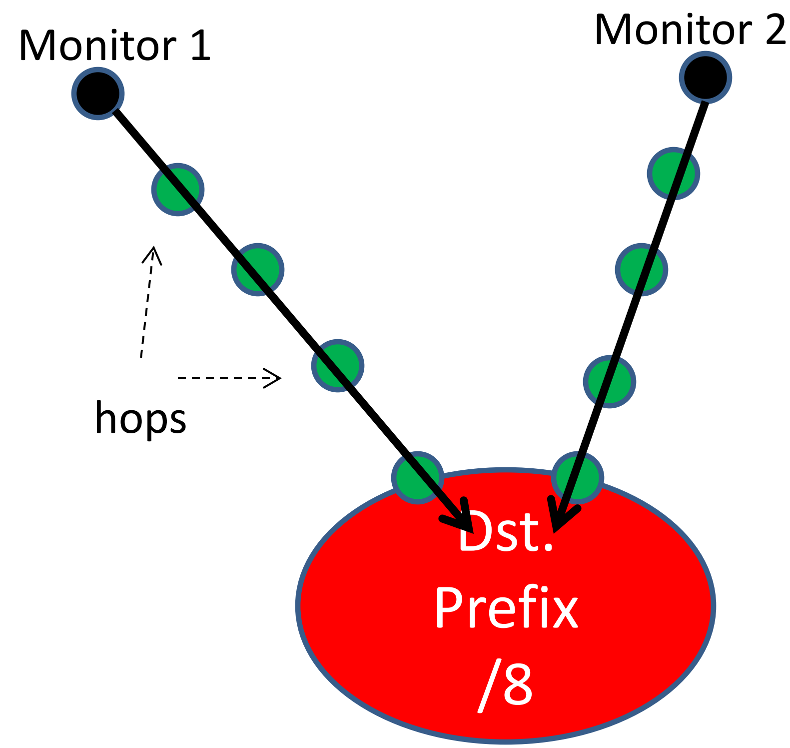
Vantage Point Spreading

- Proposed Solution: Rank Ordering Methods
  - Maximum Coverage and Ingress Point Spreading.
  - By carefully deciding the order in which monitors are assigned, the amount of new information learnt can be enhanced.
  - Pre-probing or previous data required as bootstrap.

Max Coverage: Intuition



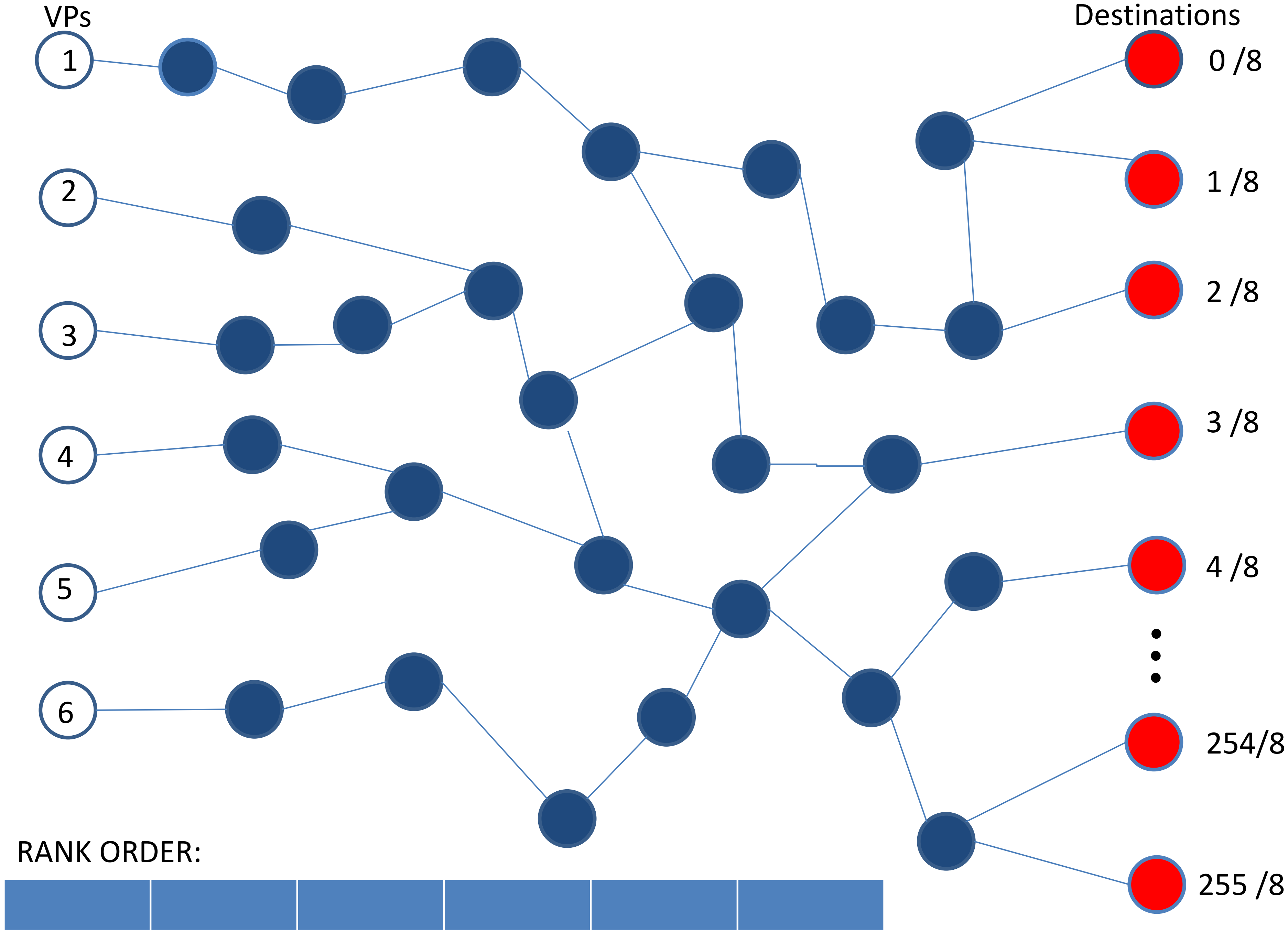
- 2 consecutive traces might share many hops,
- thus forcing stop criterion too early.
- The path might be too short and therefore not maximizing the benefit of that particular probe.

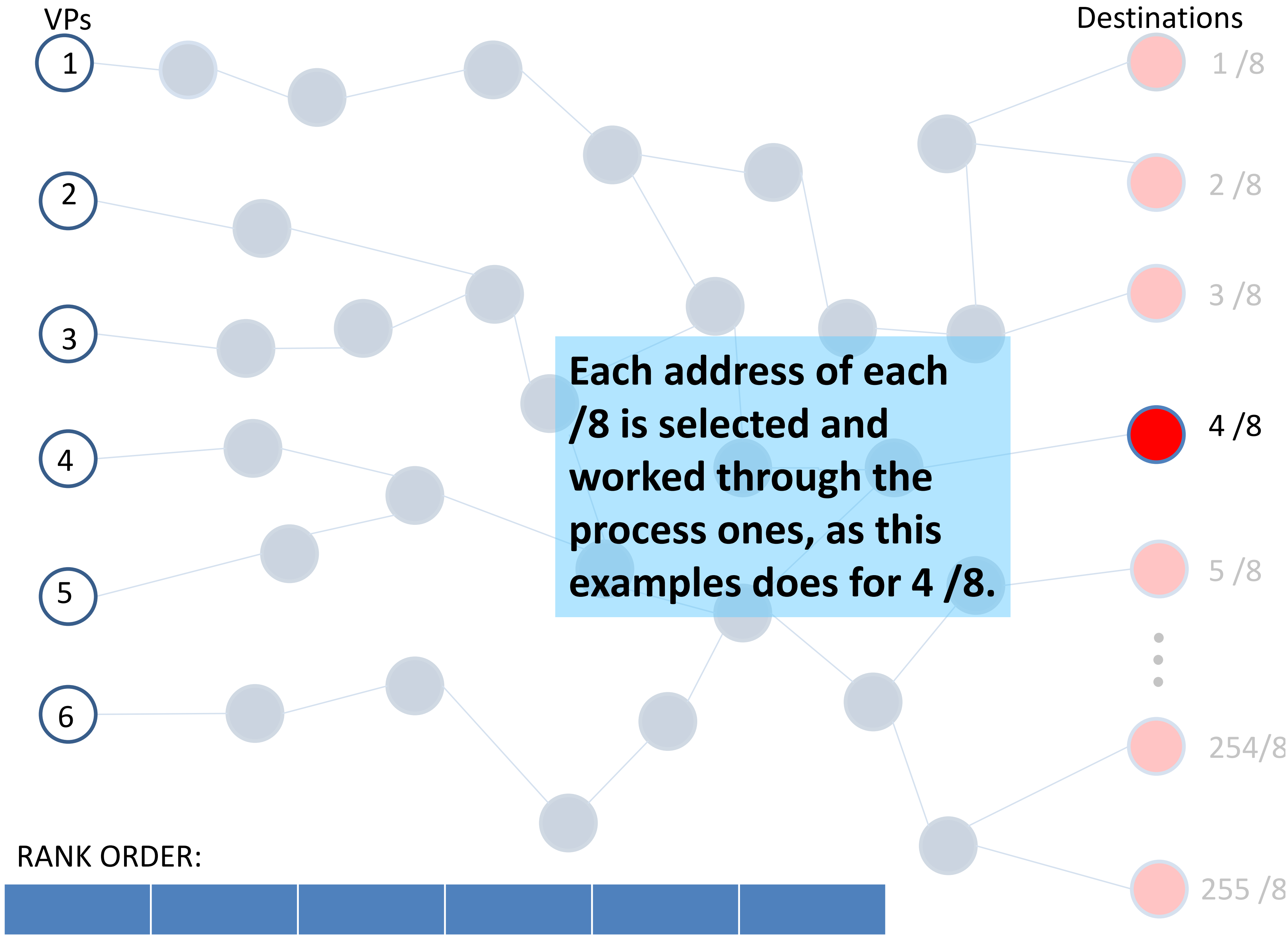


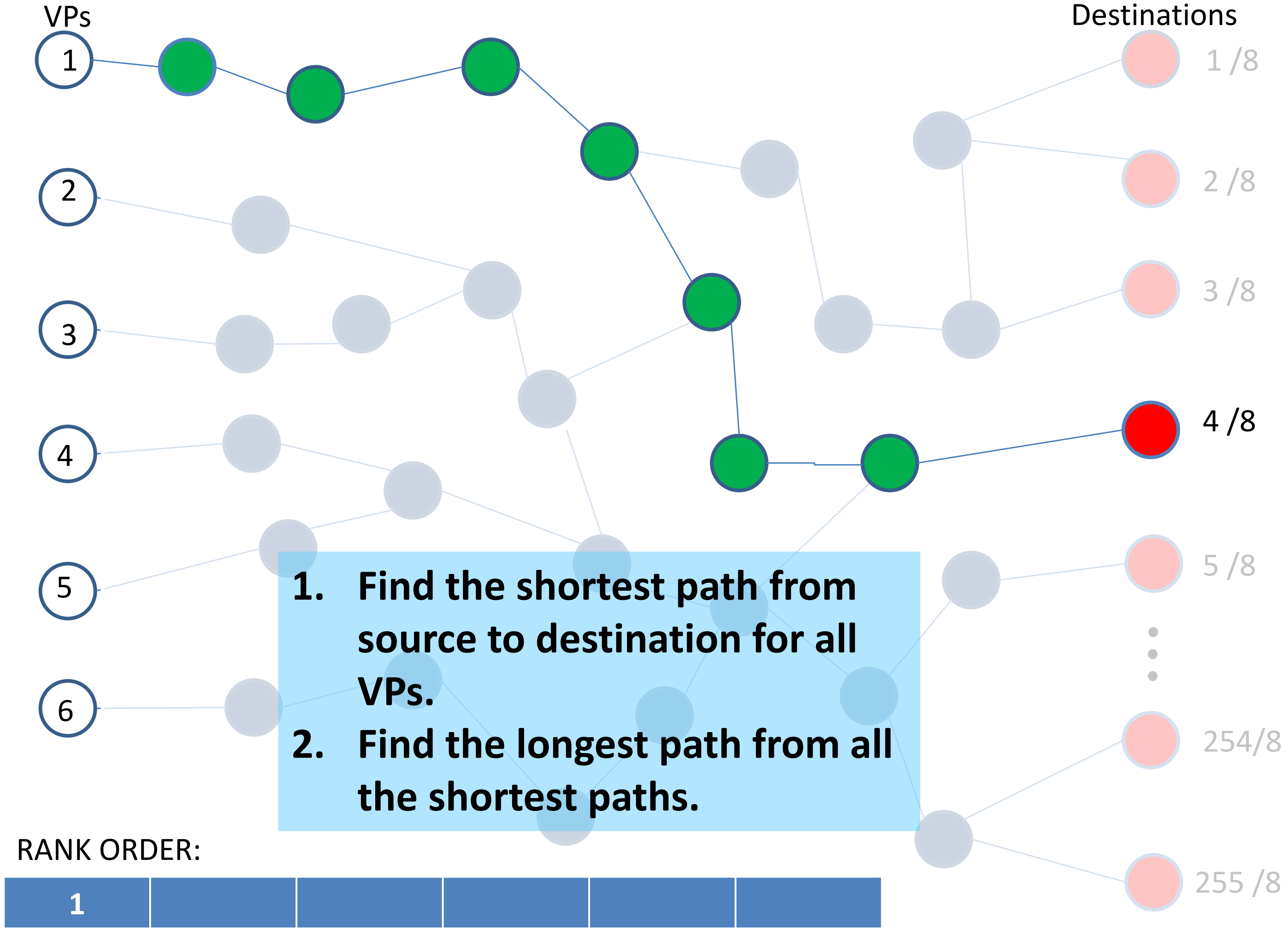
- Long path maximizes the number of vertices learnt.
- Greater chances of finding different ingress routers.

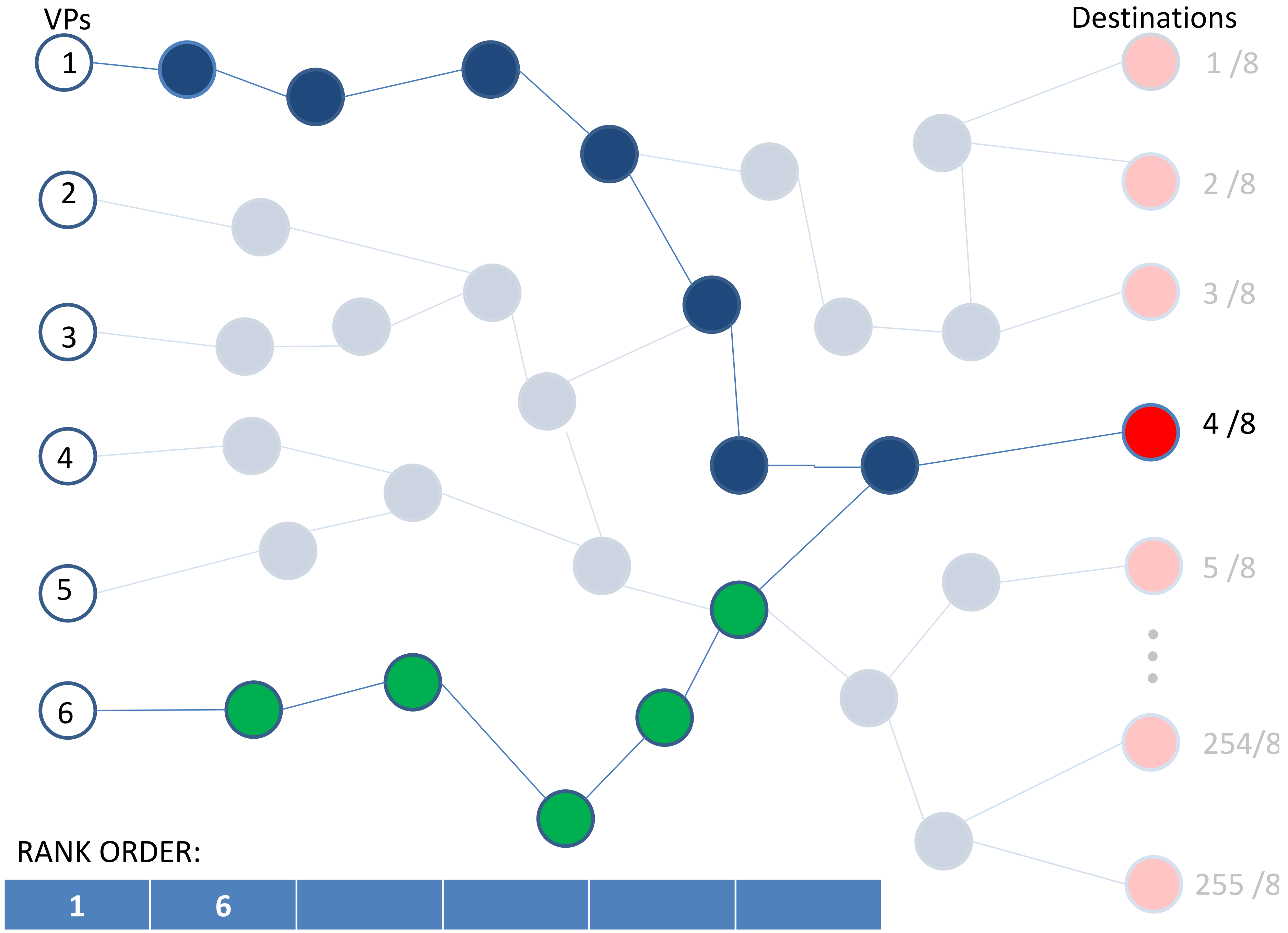
Max Coverage

- IANA: /8 allocations by region.
- Focuses in maximizing vertices outside destination-AS.
- Max hop count first vs. Min hop count first.
- Abstract example:
  - One IP address per each /8 prefix randomly selected.
  - Every monitor probes every destination address.
  - Traces are connected with each other forming the following network…

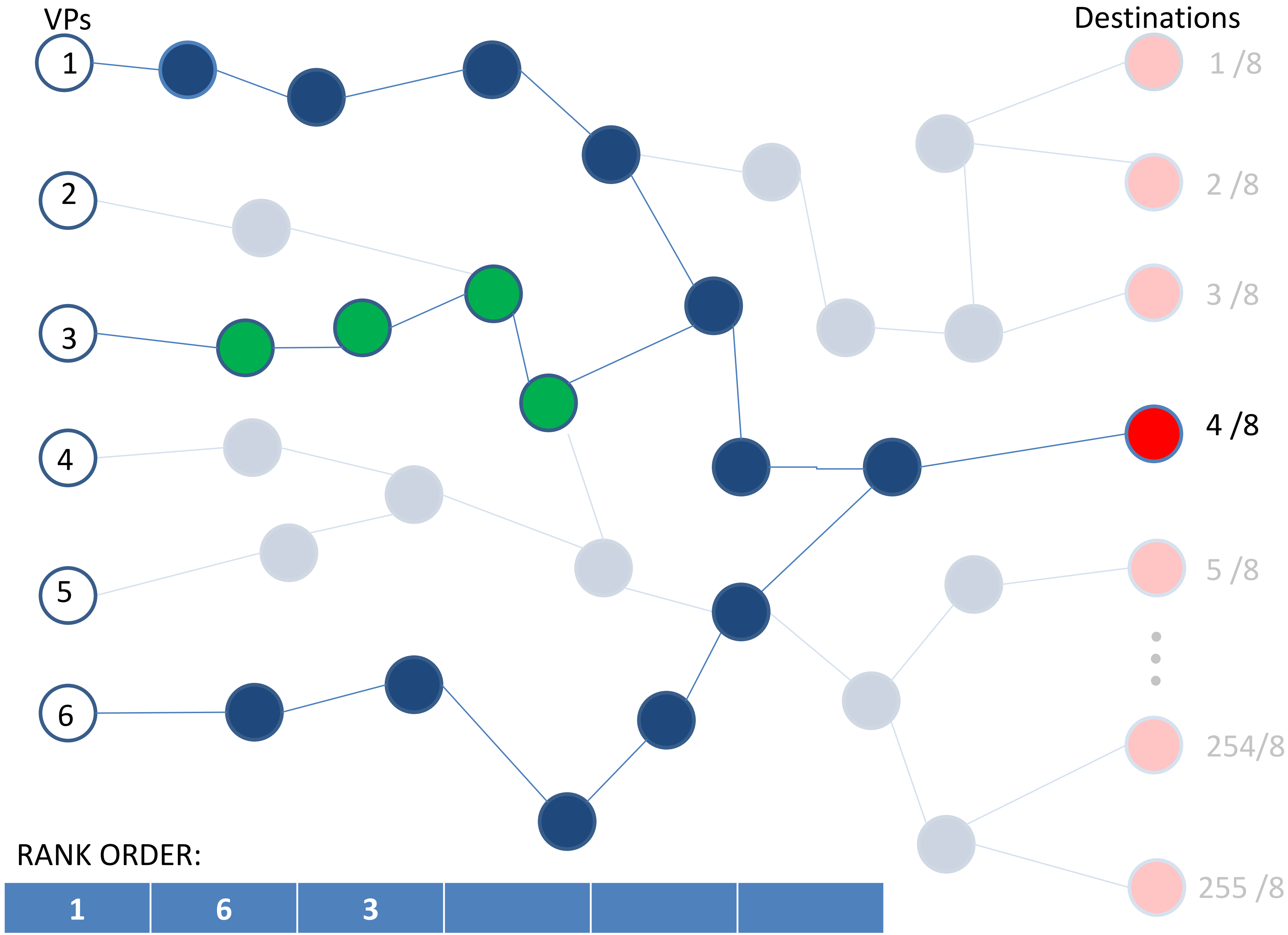

VPs
1
2
3
4
5
6
Destinations
0 /8
1 /8
2 /8
3 /8
4 /8
254/8
255 /8
RANK ORDER:


VPs
Destinations
1
2
3
4
5
6
1 /8
2 /8
3 /8
4 /8
5 /8
254/8
255 /8
Each address of each /8 is selected and worked through the process ones, as this examples does for 4 /8.
RANK ORDER:

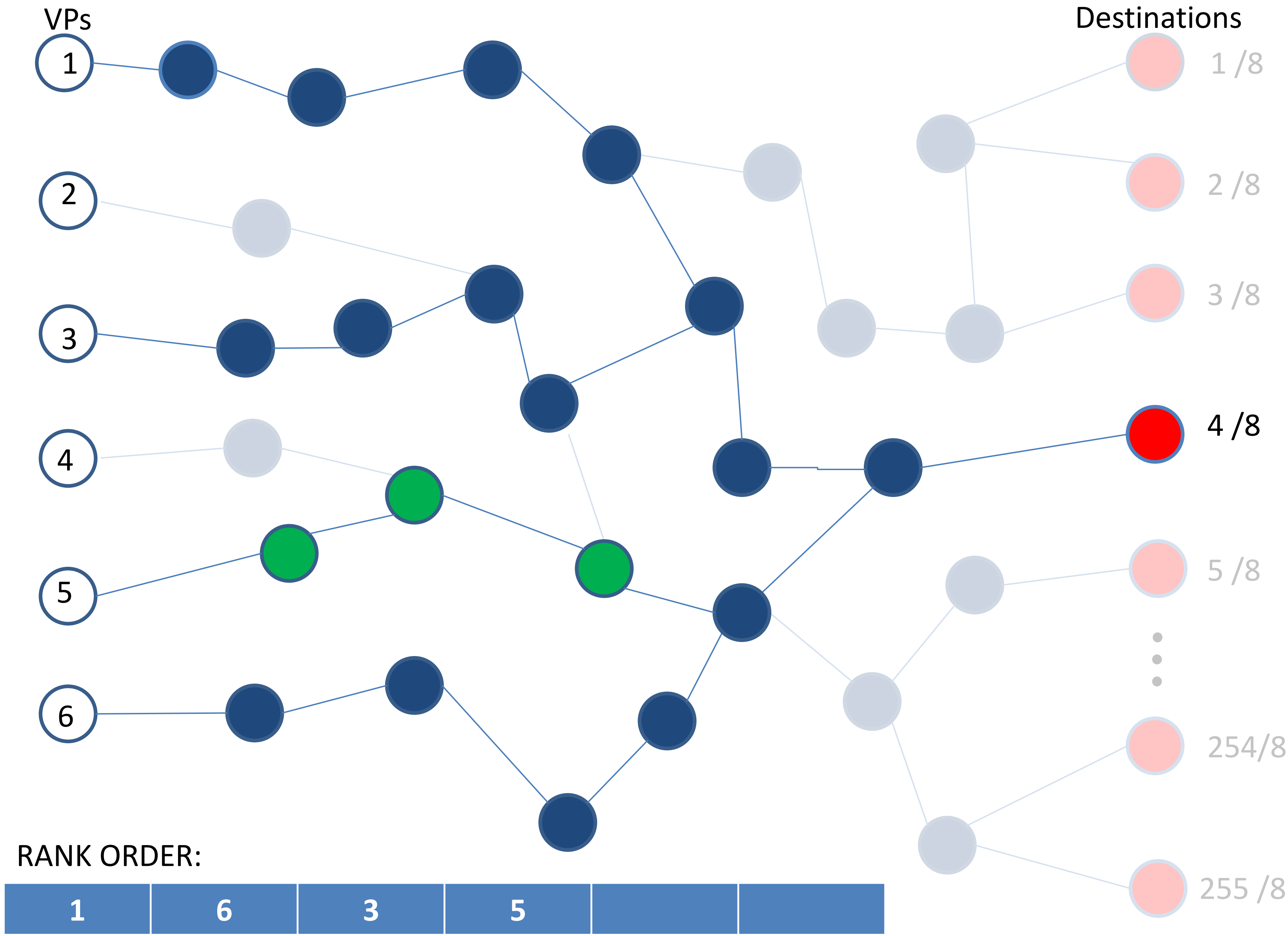

VPs
1
2
3
4
5
6
Destinations
1 /8
2 /8
3 /8
4 /8
5 /8
254/8
255 /8
1. Find the shortest path from source to destination for all VPs.
2. Find the longest path from all the shortest paths.
RANK ORDER:
1

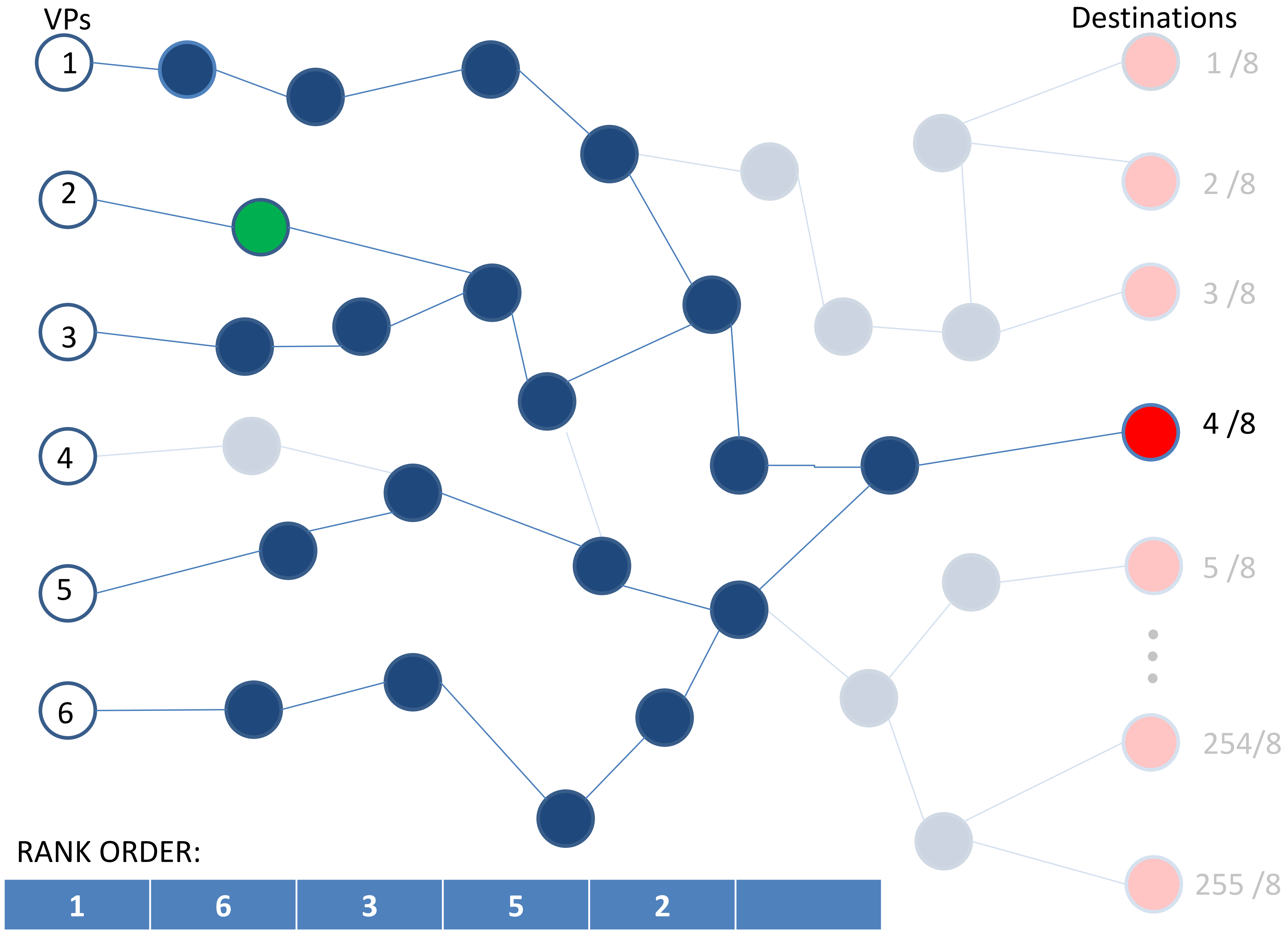


VPs
Destinations
1
2
3
4
5
6
1 /8
2 /8
3 /8
4 /8
5 /8
254/8
255 /8
RANK ORDER:
1
6

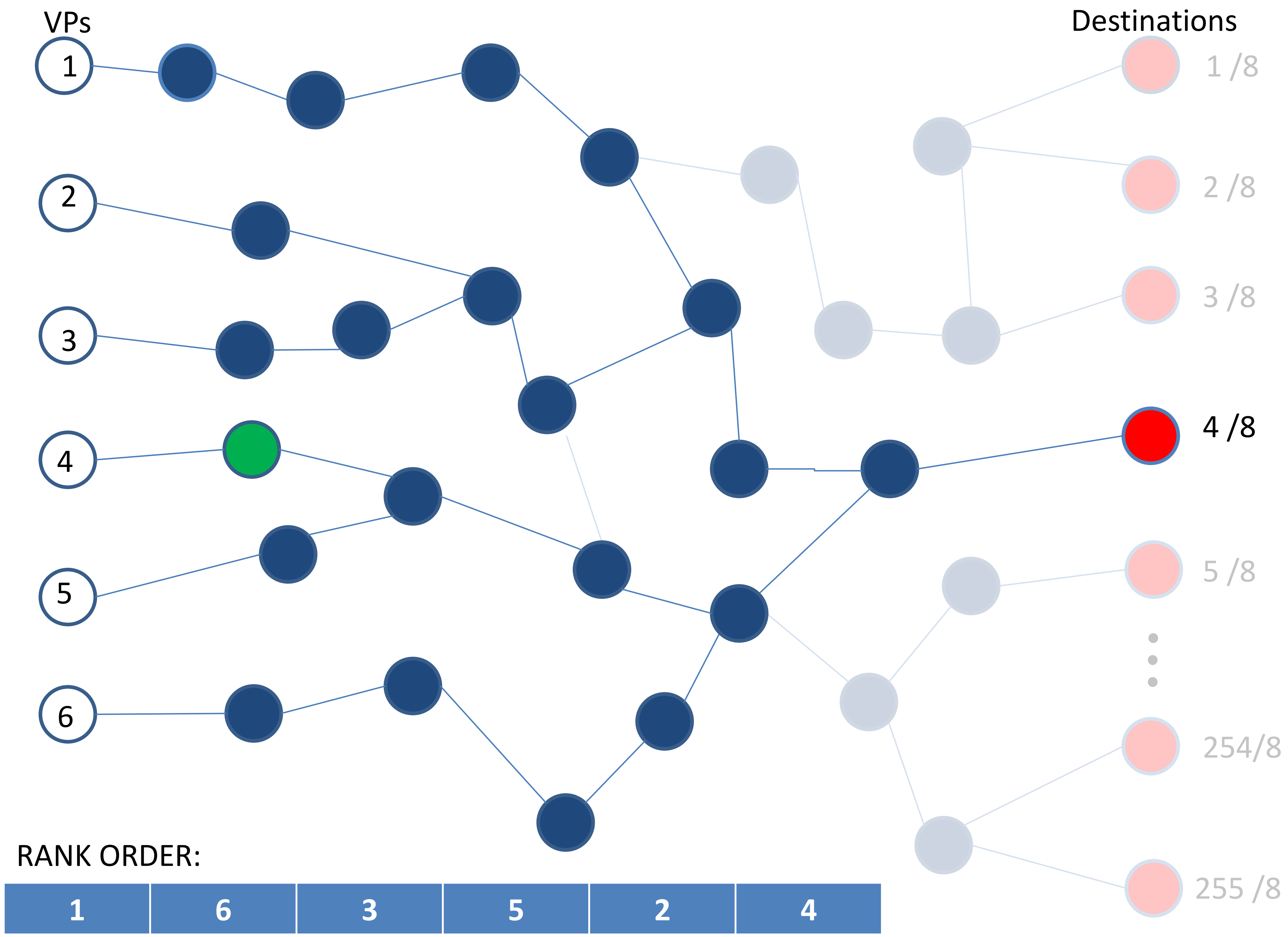


VPs
Destinations
1
2
3
4
5
6
1 /8
2 /8
3 /8
4 /8
5 /8
254/8
255 /8
RANK ORDER:
1
6
3


VPs
1
2
3
4
5
6
Destinations
1 /8
2 /8
3 /8
4 /8
5 /8
254/8
255 /8
RANK ORDER:
1
6
3
5



VPs
1
2
3
4
5
6
Destinations
1 /8
2 /8
3 /8
4 /8
5 /8
254/8
255 /8
RANK ORDER:
1
6
3
5
2



VPs
1
2
3
4
5
6
Destinations
1 /8
2 /8
3 /8
4 /8
5 /8
254/8
255 /8
RANK ORDER:
1
6
3
5
2
4

MAX

- Input:
  - G: Graph built by pre-probed traces
  - M: Set of monitors
  - P: Set of each /8 prefix

- Output:
  - R: Set of ranked monitor lists per /8 prefix

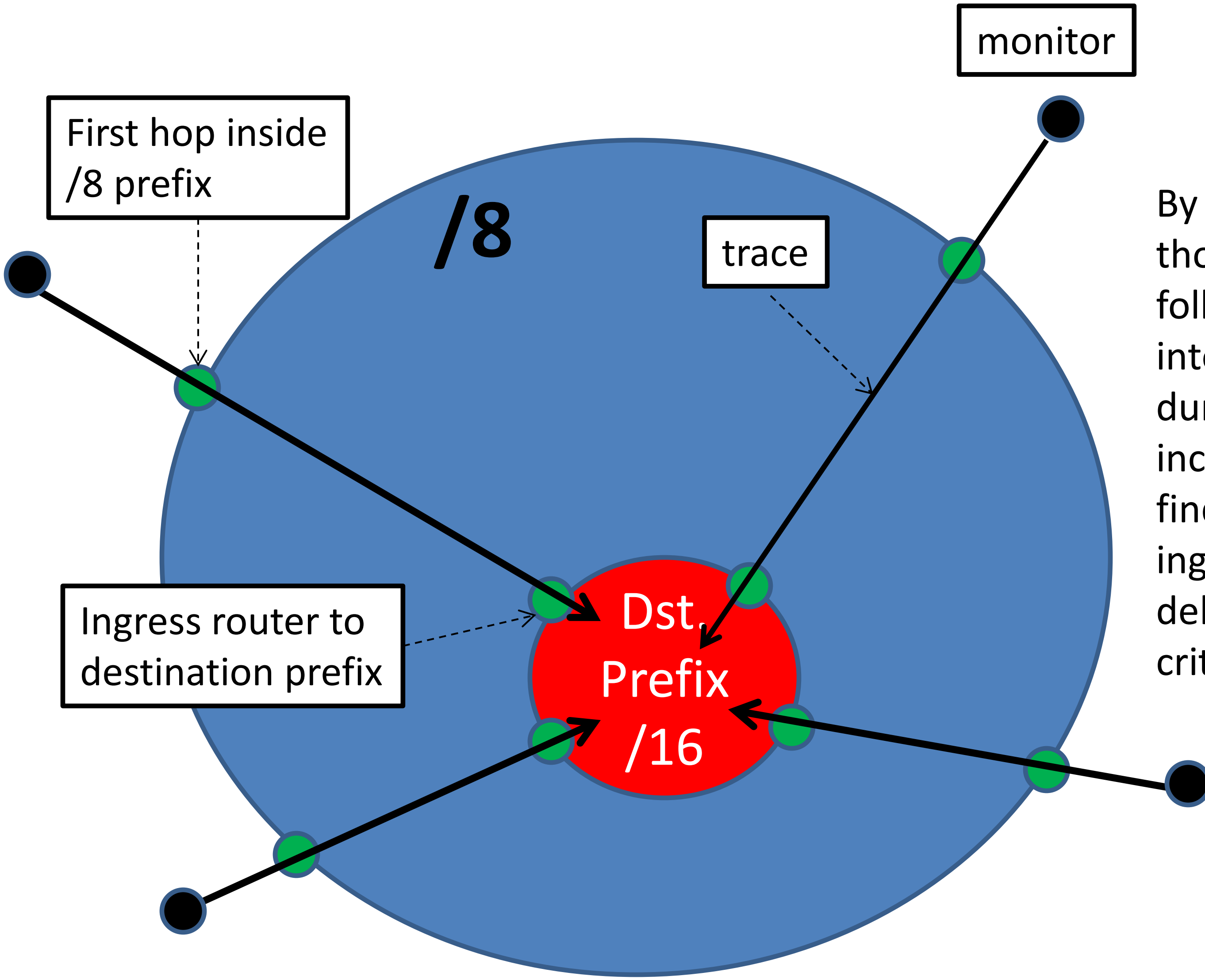
Algorithm 2: MAX(G, M, P)

for each prefix **in** P **do**
    Y = Ø (Set of shortest paths from monitors to selected destination)
    $dest_{prefix}$ = **select** one destination in P from G.
    for each monitor **in** M **do**
        t = shortest_path(monitor, $dest_{prefix}$, G)
        Y = Y ∪ t
    for each monitor **in** M **do**
        m = max(Y)
        append(m) to r_{prefix}
        delete all hops in m from all traces in Y
    R = R ∪ r_{prefix}

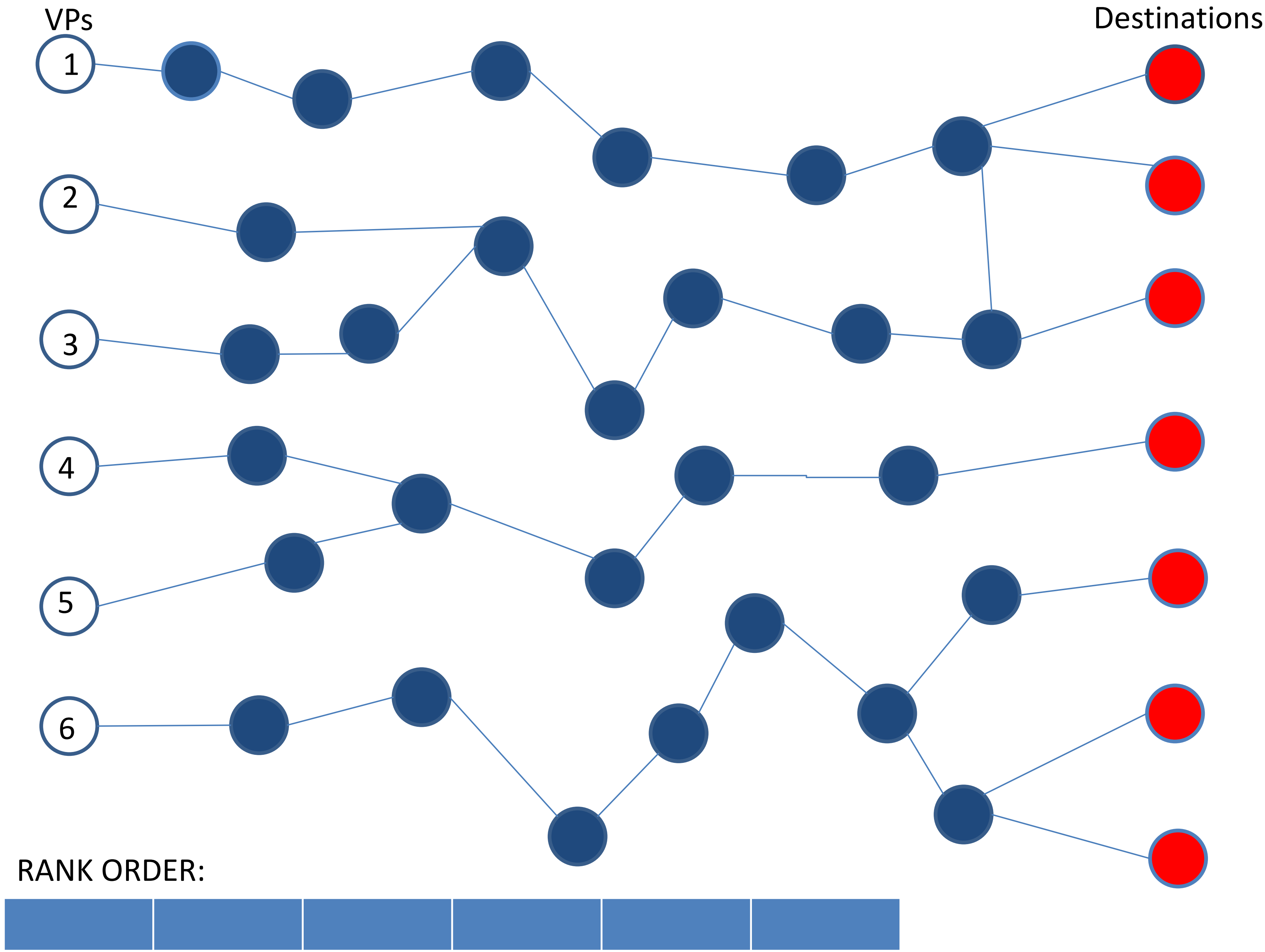
Ingress Point Spreading (IPS)

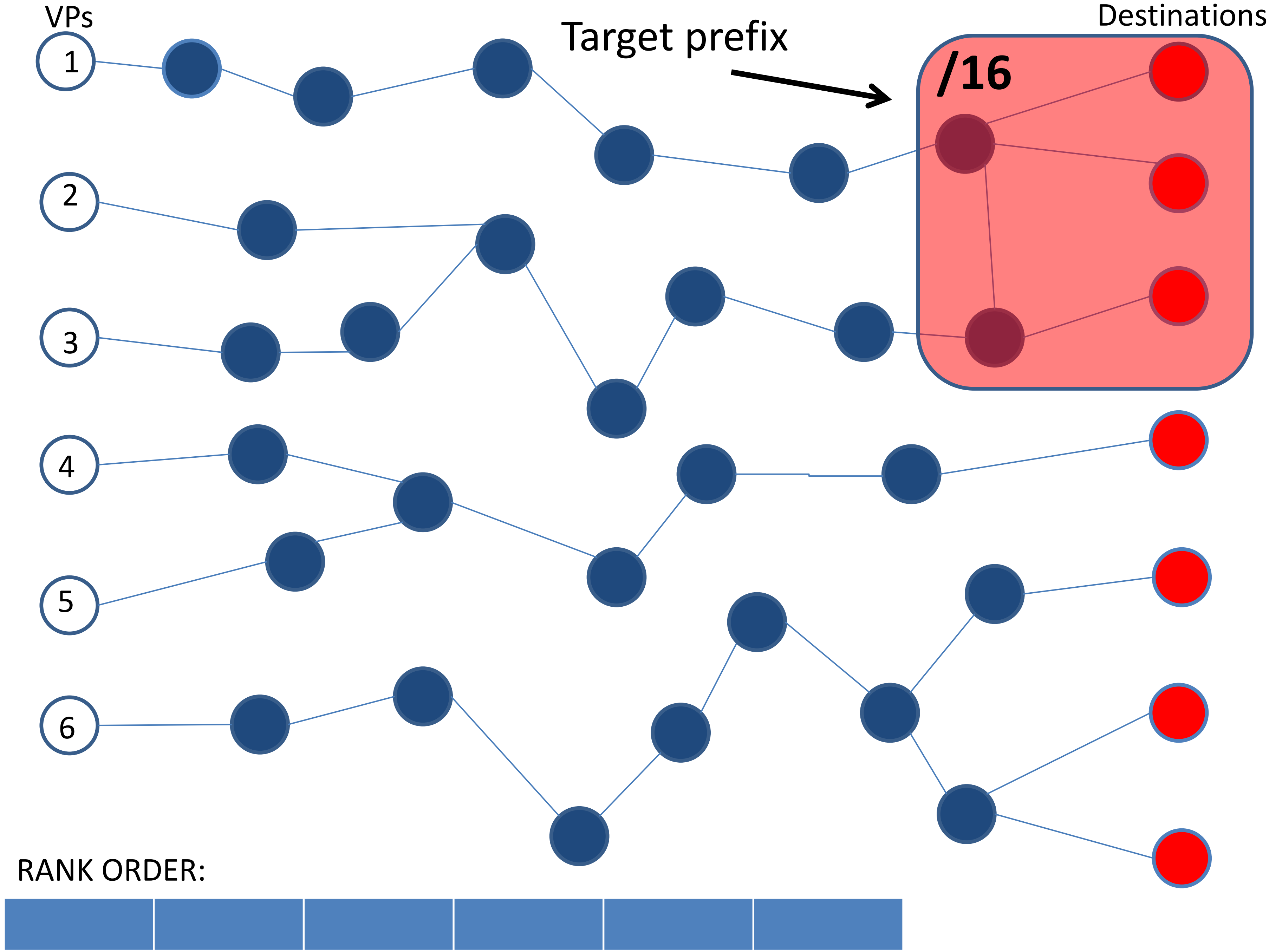
- Focuses in finding destination prefix ingress routers.
- Builds a list of ranked monitors for each prefix from data gathered by pre-probing.
- Relates 'monitor – first hop inside /8 prefix – destination' for all traces in database.
- First monitors in list are chosen from those that have a unique first hop inside /8 prefix towards destination prefix.
- Rest of list is filled with monitors that have a trace towards the expansion of the original prefix, e.g. if the original prefix is /16, the expansion would be /15, /14,…, /8.
- Needs more data than the Max Coverage method.

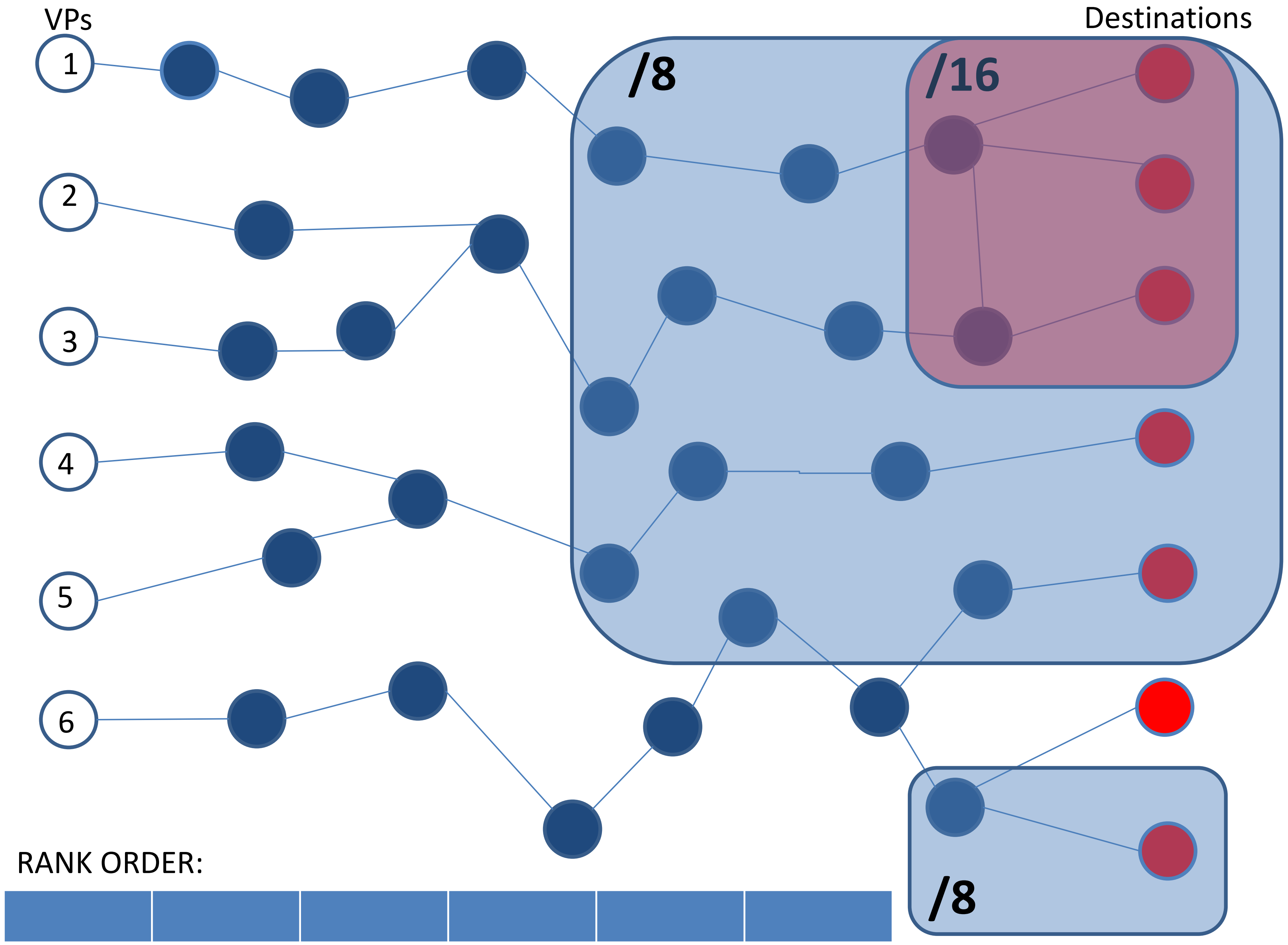
IPS: Intuition

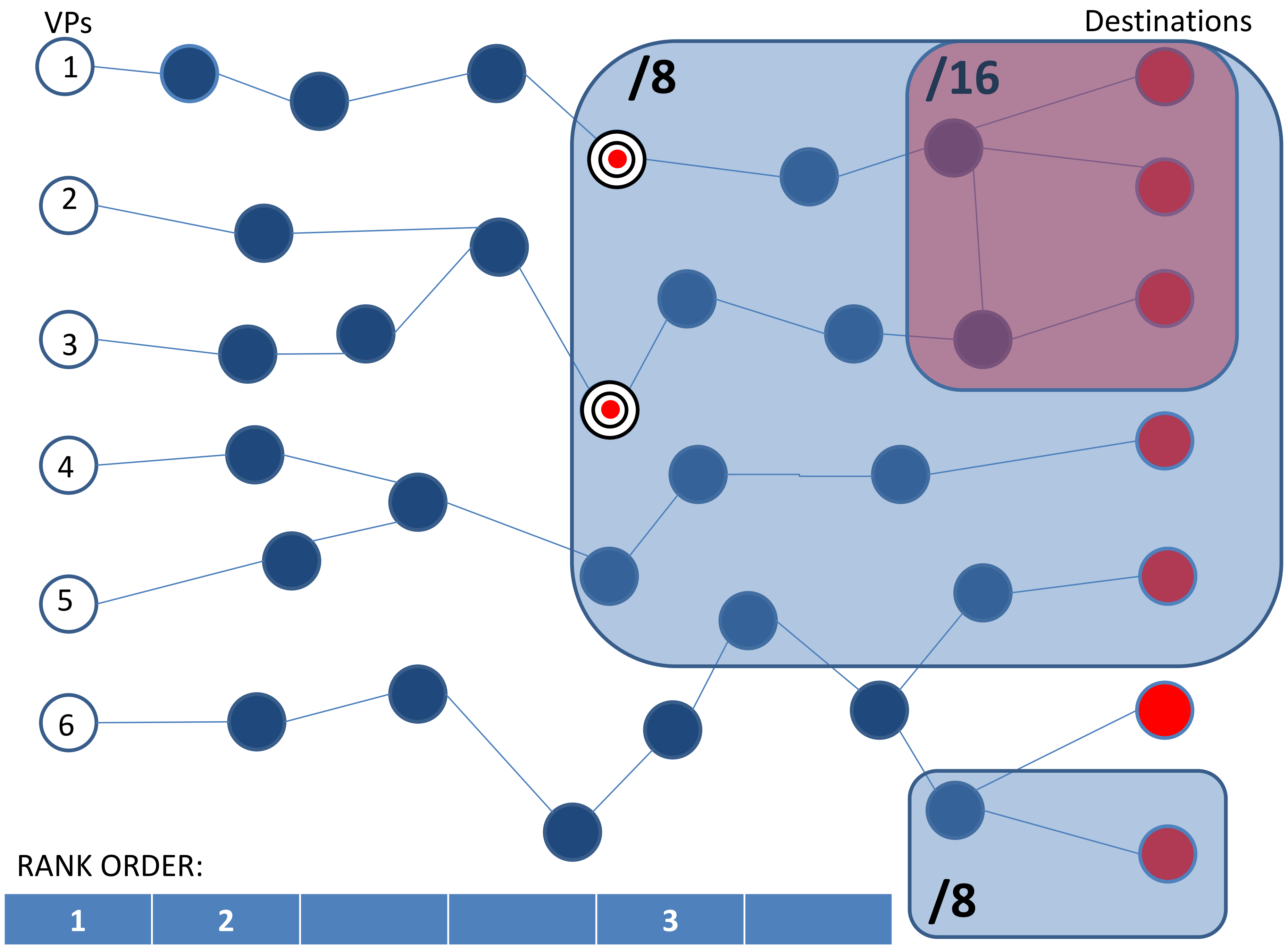


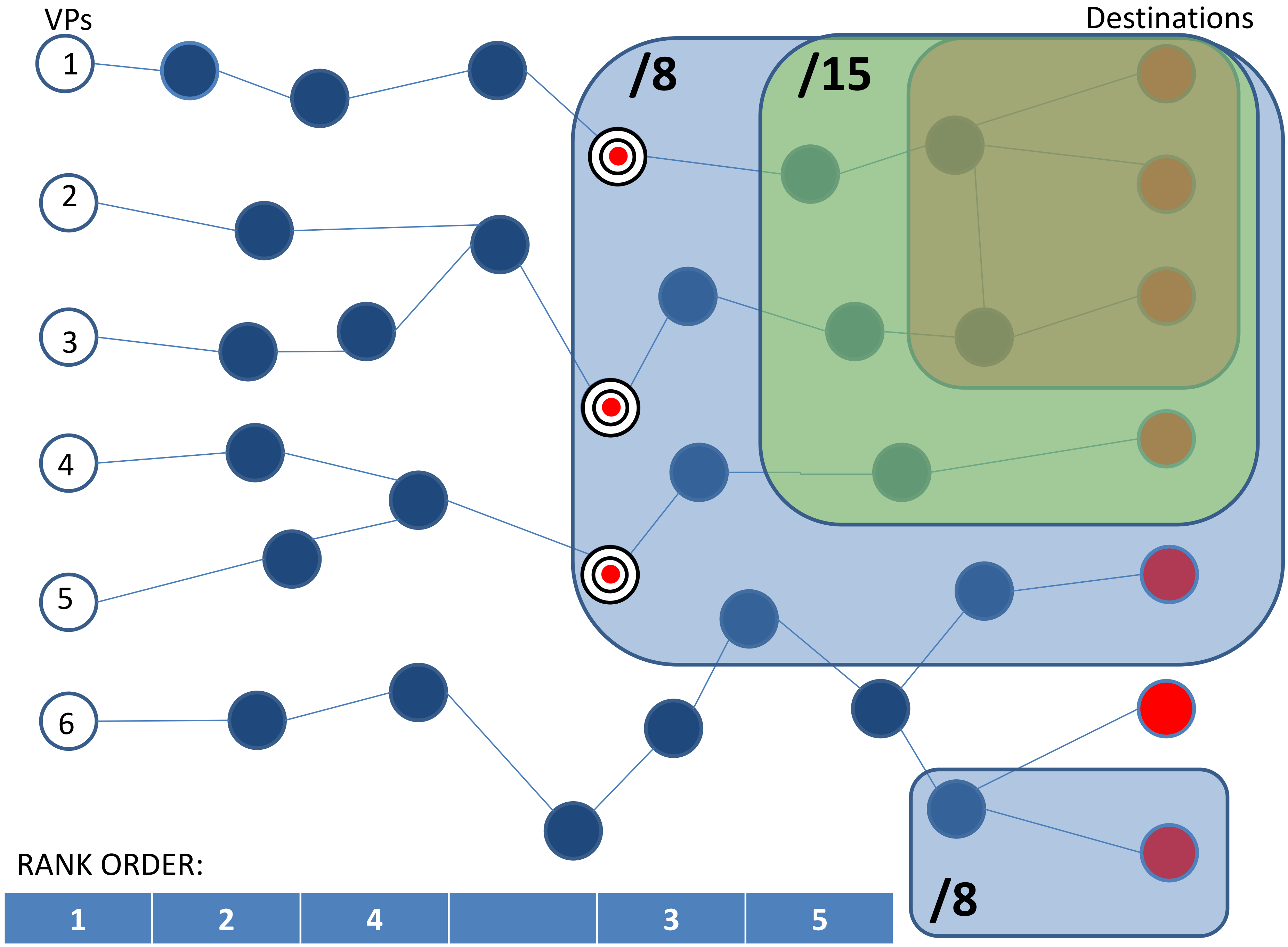
By starting to probe with those monitors that have followed different paths into the destination prefix during pre-probing, increases the chances of finding destination prefix ingress router and thus delaying meeting stop criterion.


VPs
Destinations
1
2
3
4
5
6
RANK ORDER:

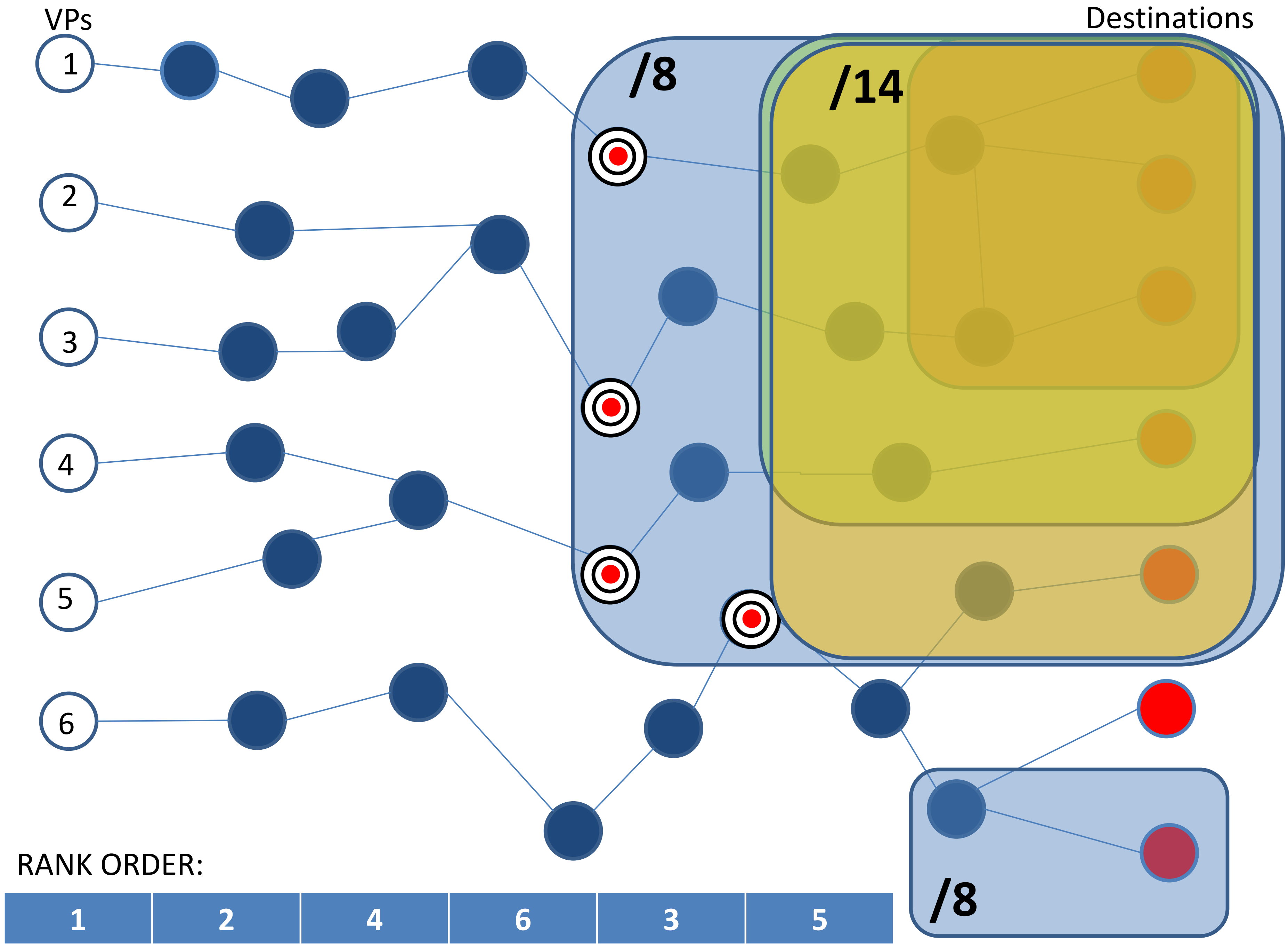

VPs
1
2
3
4
5
6
Target prefix
/16
Destinations
RANK ORDER:


VPs
Destinations
1
2
3
4
5
6
/8
/16
/8
RANK ORDER:


VPs
Destinations
/8
/16
1
2
3
4
5
6
/8
RANK ORDER:
1
2
3



VPs
Destinations
1
2
3
4
5
6
/8
/15
/8
RANK ORDER:
1
2
4
3
5



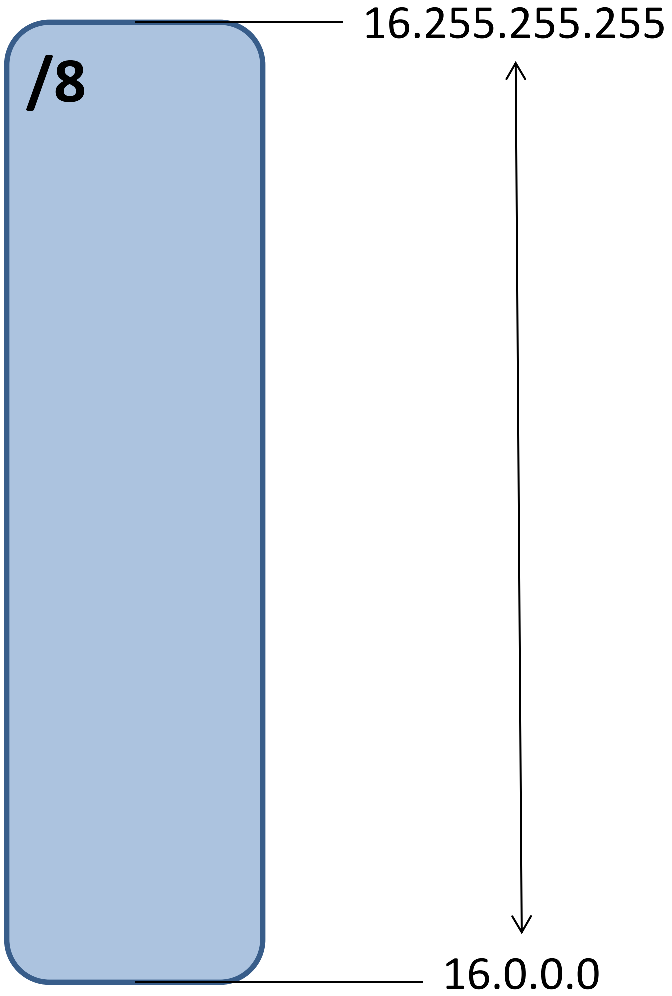
VPs
Destinations
1
2
3
4
5
6
/8
/14
/8
RANK ORDER:
1
2
4
6
3
5

IPS++

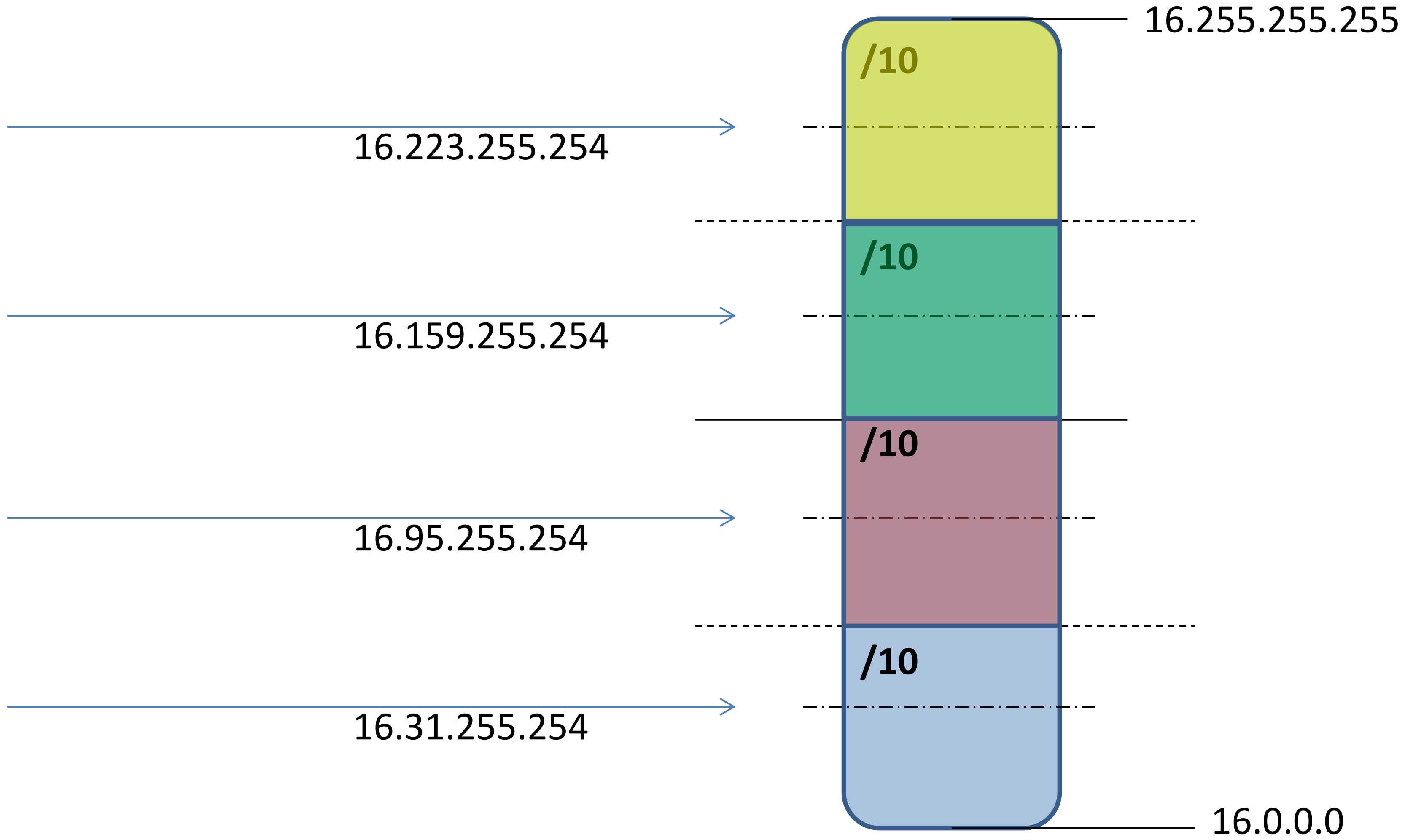
- Avoid IPS early termination.
- Ensure use of all ingress routers, and thus possible paths.
- Number of ingress routers approximated to the closest power of 2.

IPS++

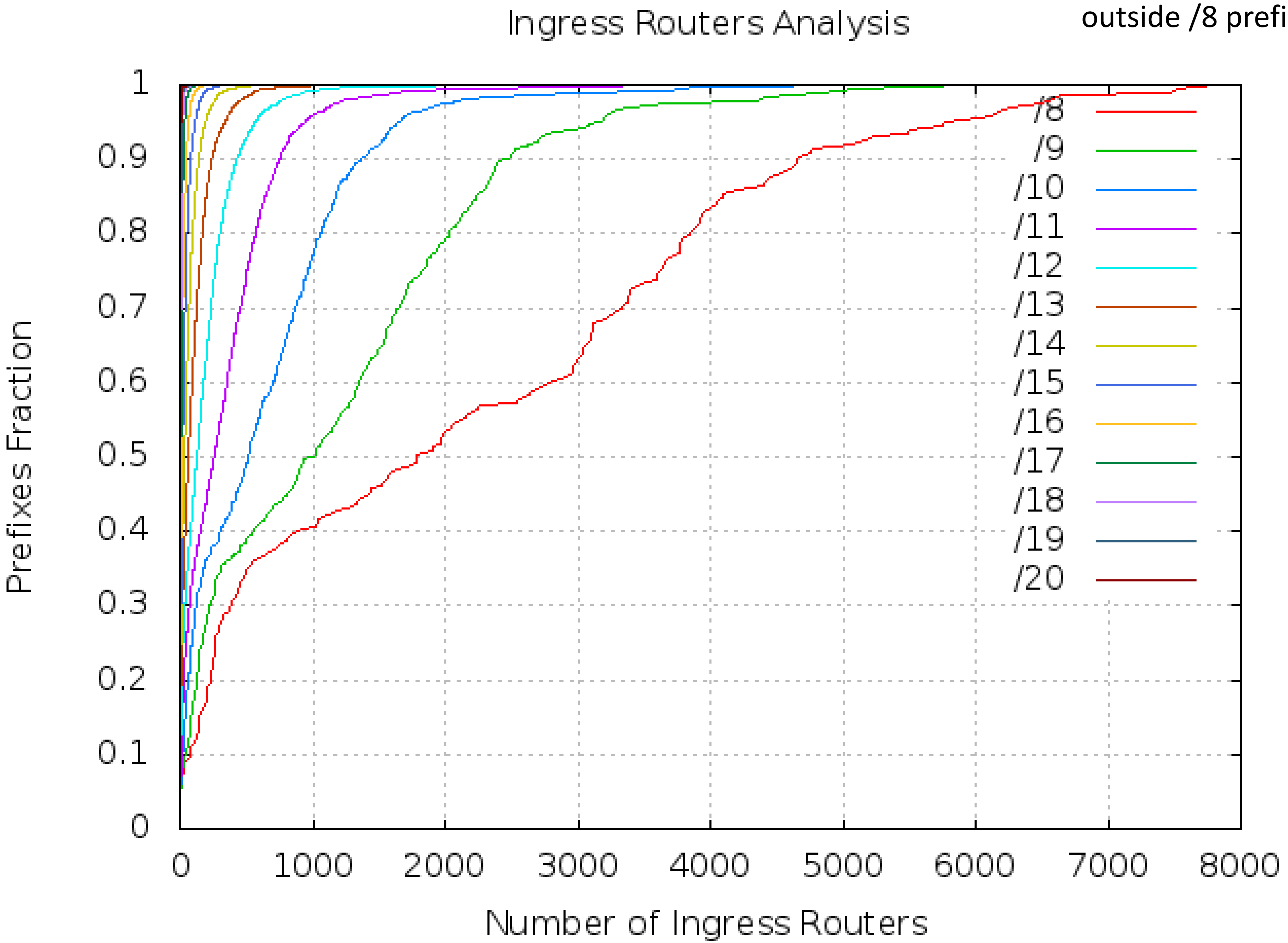
Suppose current prefix has 3 ingress routers, then the closest power 2 is 4.

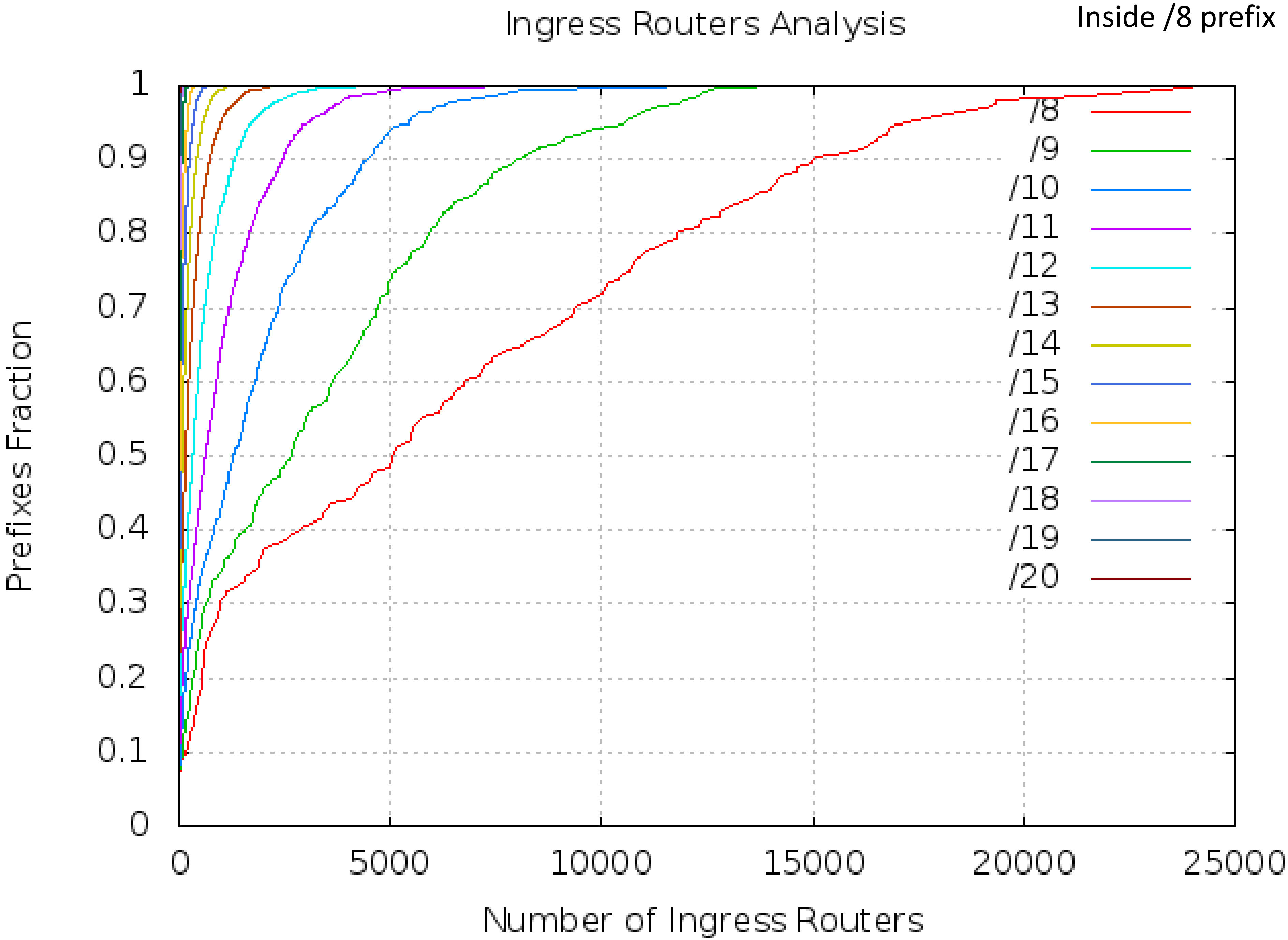


IPS++


16.255.255.255
/10
16.223.255.254
/10
16.159.255.254
/10
16.95.255.254
/10
16.31.255.254
16.0.0.0

outside /8 prefix




Inside /8 prefix
Ingress Routers Analysis
Prefixes Fraction
Number of Ingress Routers
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
0
5000
10000
15000
20000
25000
/8
/9
/10
/11
/12
/13
/14
/15
/16
/17
/18
/19
/20

Ingress Point Spreading (IPS)

- Input:
  - T: Set of pre-probed traces
  - M: Set of monitors
  - p/m: Set of prefixes / masks

- Output:
  - R: Set of ranked monitor lists per prefix / mask

Algorithm 3: IPS(T, M, p/m)

{I} =⊥ (set of tuples formed by (monitor, first hop in destinations /8 prefix and destination) from each trace)

{J} = ⊥ (set of tuples formed by monitor, first hop in /8 prefix from traces with destination inside p/m and mask)

for each trace **in** T **do**

(src, dst) = source and destination of trace

D = /8 prefix of dst

hop = **find** first hop in trace inside D

I = I ∪ (src, hop, dst)

end for

for each prefix **in** p/m **do**

r_{prefix} = ⊥ (array of tuples containing monitor and first hop)

for each monitor **in** M **do**

flag = False

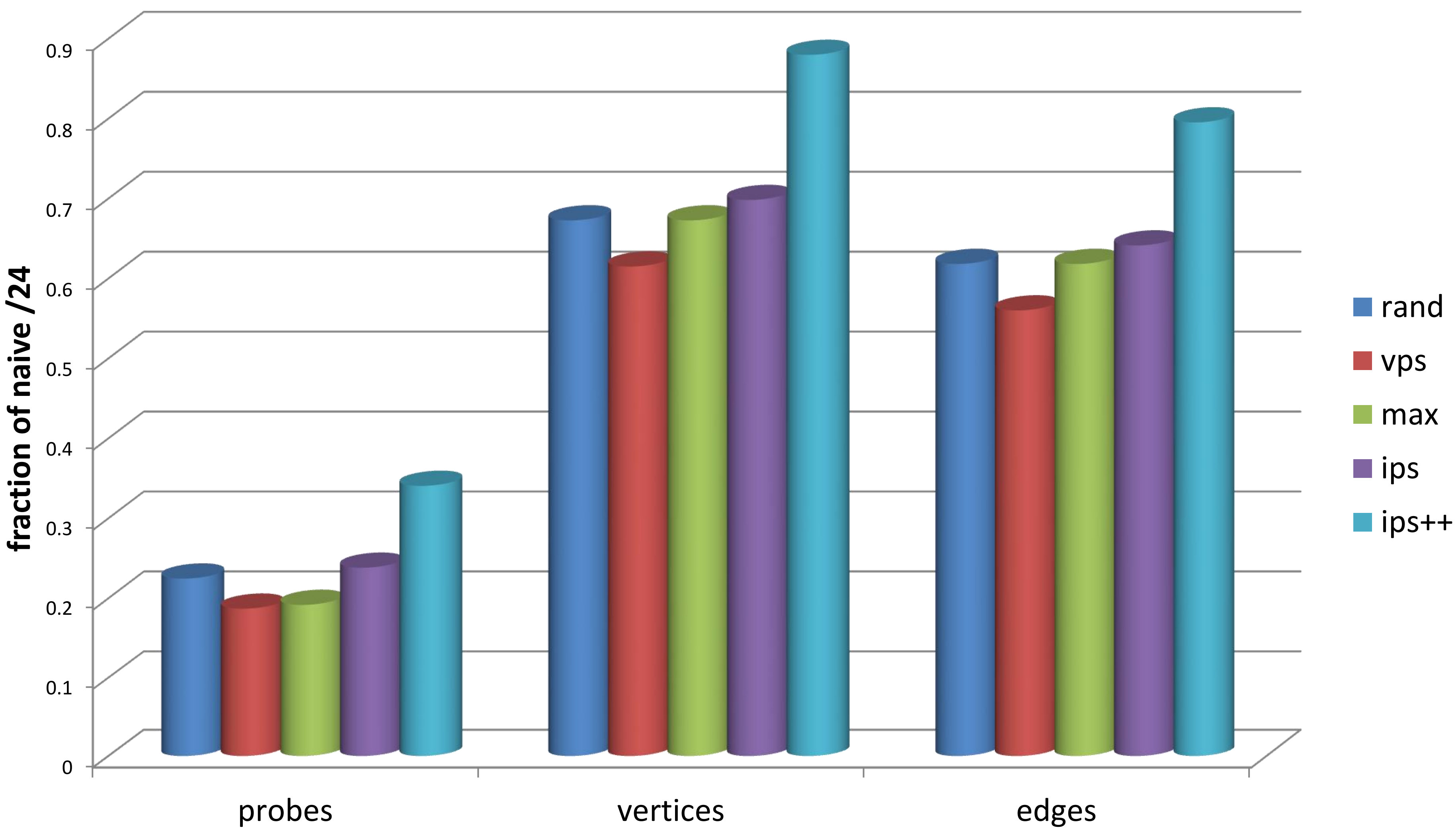
for mask = m **to** 8 in steps = -1 **do**

if flag **then**

break // continue with next monitor

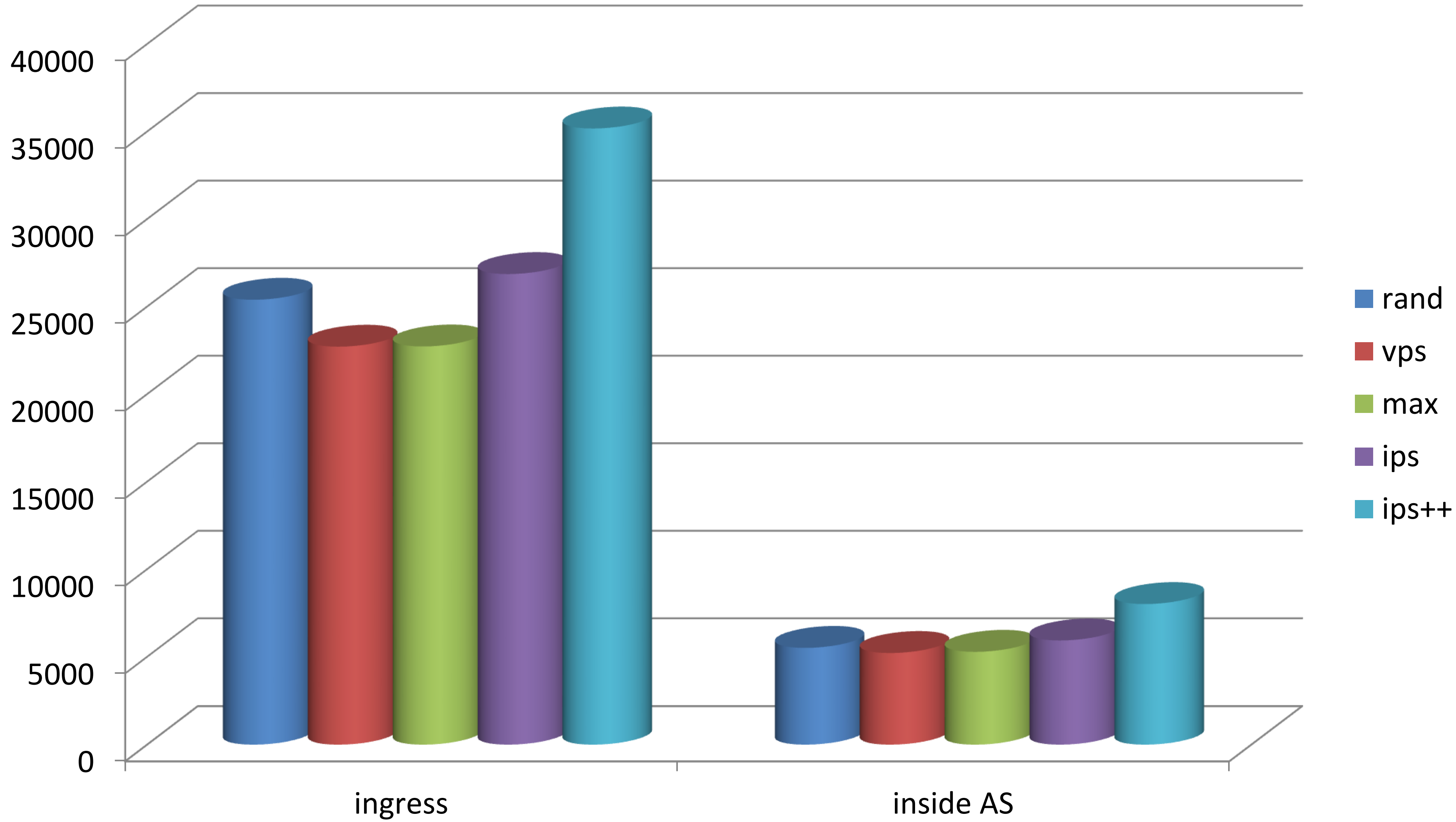
```
        for each tuple in I that contains monitor do
          D = /mask prefix of dst in tuple
          P = /mask prefix of p
          if D == P then
            append (src, hop, mask) to r_prefix
            flag = True
        end for
      end for
    end for
    delete tuples from r_prefix with repeated monitors and first hops
    sort(r_prefix) // by mask from higher to lower
    J = J ∪ r_prefix
  end for
```

Results



fraction of naive /24
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
probes
vertices
edges
rand
vps
max
ips
ips++

Results


40000
35000
30000
25000
20000
15000
10000
5000
0
ingress
inside AS
rand
vps
max
ips
ips++

References:

1. Beverly, R., Berger, A., Xie, G.: Primitives for Active Internet Topology Mapping: Toward High-Frequency Characterization. In: Proc. IMC (2010).