Labeling and Routing

Réseaux & Communication

Alain Cournier   Stéphane Devismes

Université de Picardie Jules Verne

November 30, 2025

Roadmap

Roadmap

Preamble

This lesson is mainly based on Chapter 4 of the book entitled

"Introduction to Distributed Algorithms,"

by Gerard Tel [4].

Routing

In a network, a node can send packets of information directly only to a subset of nodes: its neighbors.

Routing: decision procedure by which a node selects one (or, sometimes, more) of its neighbors to forward a packet on its way to an ultimate destination.

Routing Algorithm: a decision-making procedure to perform routing and guaranteeing delivery of each packet.

Criteria for “good” routing

Criteria for “good” routing

1. **Correctness:** each packet should be eventually delivery to its ultimate destination.

Criteria for “good” routing

1. **Correctness:** each packet should be eventually delivery to its ultimate destination.
2. **Efficiency:** each packet should be routed through “good” paths. (*n.b.*, more detail in the next slide)

Criteria for "good" routing

1. **Correctness:** each packet should be eventually delivery to its ultimate destination.
2. **Efficiency:** each packet should be routed through "good" paths. (*n.b.*, more detail in the next slide)
3. **Complexity:** cost in terms of messages (control and data packets), volume of exchanged data, time, storage . . . .

Criteria for “good” routing

1. **Correctness:** each packet should be eventually delivery to its ultimate destination.
2. **Efficiency:** each packet should be routed through “good” paths. (*n.b.*, more detail in the next slide)
3. **Complexity:** cost in terms of messages (control and data packets), volume of exchanged data, time, storage . . . .
4. **Message ordering:** is the message sending order between a source and a destination preserved upon receipt (FIFO)?

Criteria for "good" routing

1. **Correctness:** each packet should be eventually delivery to its ultimate destination.
2. **Efficiency:** each packet should be routed through "good" paths. (*n.b.*, more detail in the next slide)
3. **Complexity:** cost in terms of messages (control and data packets), volume of exchanged data, time, storage . . . .
4. **Message ordering:** is the message sending order between a source and a destination preserved upon receipt (FIFO)?
5. **Robustness:** ability of handling topological changes.

Criteria for "good" routing

1. **Correctness:** each packet should be eventually delivery to its ultimate destination.
2. **Efficiency:** each packet should be routed through "good" paths. (*n.b.*, more detail in the next slide)
3. **Complexity:** cost in terms of messages (control and data packets), volume of exchanged data, time, storage . . . .
4. **Message ordering:** is the message sending order between a source and a destination preserved upon receipt (FIFO)?
5. **Robustness:** ability of handling topological changes.
6. **Adaptiveness:** load-balancing at channels and nodes.

Criteria for “good” routing

1. **Correctness:** each packet should be eventually delivery to its ultimate destination.
2. **Efficiency:** each packet should be routed through “good” paths. (*n.b.*, more detail in the next slide)
3. **Complexity:** cost in terms of messages (control and data packets), volume of exchanged data, time, storage . . . .
4. **Message ordering:** is the message sending order between a source and a destination preserved upon receipt (FIFO)?
5. **Robustness:** ability of handling topological changes.
6. **Adaptiveness:** load-balancing at channels and nodes.
7. **Fairness:** ability to provide service to every user in the same degree.

Criteria for "good" routing

1. **Correctness:** each packet should be eventually delivery to its ultimate destination.
2. **Efficiency:** each packet should be routed through "good" paths. (*n.b.*, more detail in the next slide)
3. **Complexity:** cost in terms of messages (control and data packets), volume of exchanged data, time, storage . . . .
4. **Message ordering:** is the message sending order between a source and a destination preserved upon receipt (FIFO)?
5. **Robustness:** ability of handling topological changes.
6. **Adaptiveness:** load-balancing at channels and nodes.
7. **Fairness:** ability to provide service to every user in the same degree.

Remark

These criteria are often conflicting: most of algorithms perform well only *w.r.t.* a subset of them.

A illustrative example will be proposed later.

Main optimization criteria

1. **Minimum hop:** minimizing the number traversed edges.
2. **Shortest path:** a (non-negative) weight is statically assigned to each channel.

   Minimizing the sum of the weights of the traversed edges.
3. **Minimum delay:** a (non-negative) weight is dynamically assigned to each channel

   (weights are periodically revised depending on the traffic).

   Minimizing the sum of the weights of the traversed edges.

Labeling

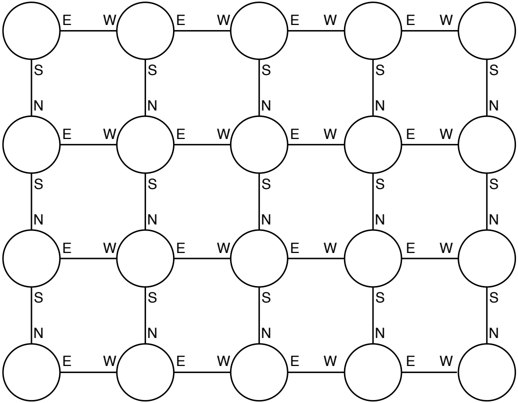
Labeling consists of assigning (or re-assigning) labels to nodes and/or channels.

Usually,

- node labels are unique in the network,
- while channel labels are unique only at the incident node.

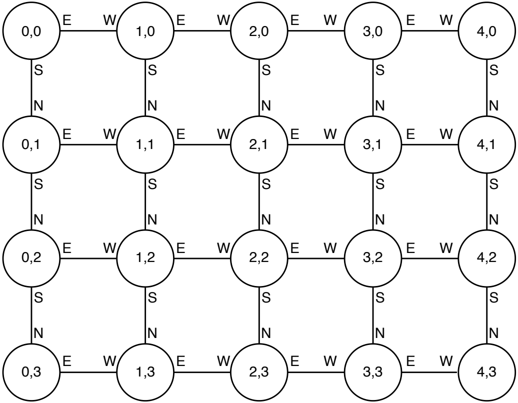
Illustrative Example (channel labeling)

N-S-E-W sense of direction in a $(\ell \times L)$-grid with $\ell > 1$ and $L > 1$

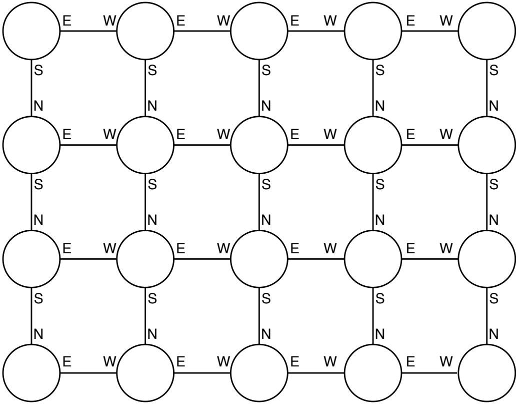


Illustrative Example

From N-S-E-W sense of direction to Coordinated System (Node Labeling)



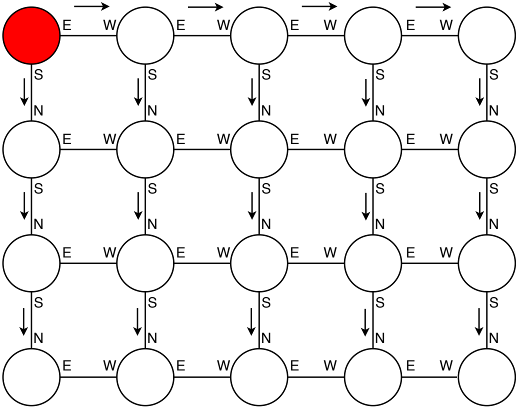
How?



Illustrative Example

From N-S-E-W sense of direction to Coordinated System (Node Labeling)

How?



Node labeling, code for node p: the Algorithm

Inputs

$\ell, L \in \mathbb{N}$: length and width of the grid
Labels $\subseteq \{N, S, E, W\}$: labels of channels incident to p

Variables

$x, y \in \mathbb{N}$

Initialization ▷ All Initiators

if *Labels* $= \{E, S\}$ **then** ▷ Top-Left Corner
    $(x, y) \leftarrow (0, 0)$
    `Send` $\langle x, y \rangle$ `to` $\{S, E\}$
end if

Receipt of $\langle a, b \rangle$ **from** N ▷ Along a column

$(x, y) \leftarrow (a, b + 1)$
if $S \in$ *Labels* **then**
    `Send` $\langle x, y \rangle$ `to` S
end if

Receipt of $\langle a, b \rangle$ **from** W ▷ Along the uppermost row

$(x, y) \leftarrow (a + 1, b)$
if $E \in$ *Labels* **then**
    `Send` $\langle x, y \rangle$ `to` E
end if
`Send` $\langle x, y \rangle$ `to` S

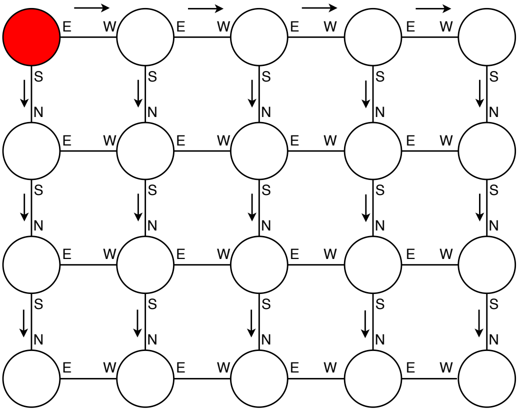
Observations

1. From all-initiators to multi-initiators: wake-up the leader using flooding
   (*cf.*, distributed computing courses)
2. Time complexity: $\ell + L$, optimal

   N.b., in case $\ell = L$, $\sqrt{n}$, where n is the number of nodes
3. Message complexity: $\ell \times L$, optimal
4. Message size: $O(\log \ell + \log L)$ bits per message
5. Memory requirement: $O(\log \ell + \log L)$ bits per node
6. **termination detection is missing**

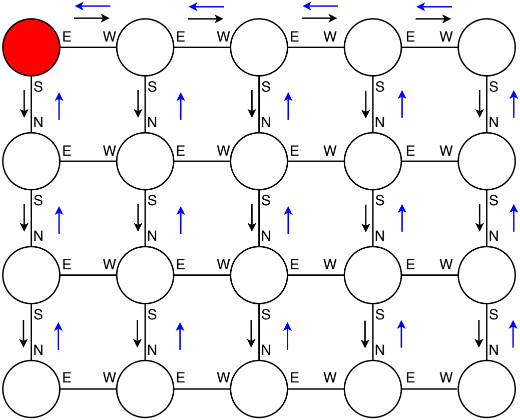
Adding Termination Detection at $(0, 0)$

How?



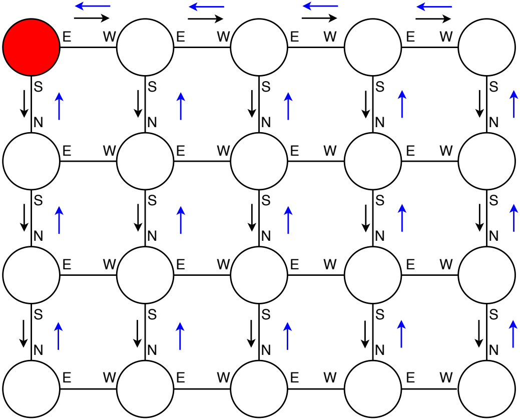
Adding Termination Detection at $(0, 0)$

How?



Adding Termination Detection at $(0, 0)$

How?



Remark: global termination detection requires an additional flooding initiated by $(0, 0)$

Adding Termination Detection at $(0,0)$, the algorithm

Add variable Cpt initialized to 0

Receipt of $\langle a, b \rangle$ **from** N

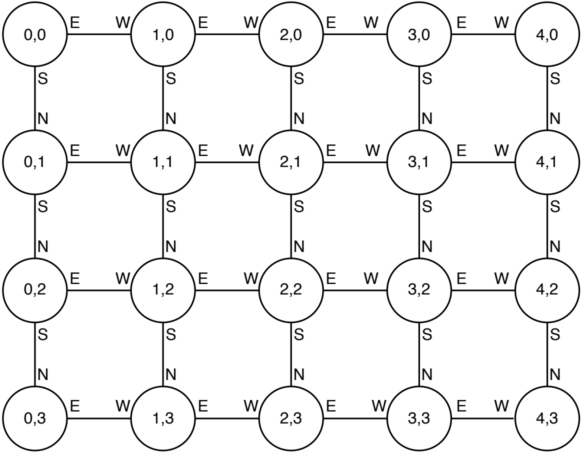
$(x, y) \leftarrow (a, b+1)$
if $S \in Labels$ **then**
    `Send` $\langle x, y \rangle$ `to` S
else
    `Send` $\langle Ack \rangle$ `to` N
end if

Receipt of $\langle Ack \rangle$ **from** c

if $N \in Labels$ **then**
    `Send` $\langle Ack \rangle$ `to` N
else if $E \notin Labels$ **then**
    `Send` $\langle Ack \rangle$ `to` W
else
    $Cpt++$
    if $Cpt = 2$ **then**
        if $Labels = \{S, E\}$ **then**
            termination
        else
            `Send` $\langle Ack \rangle$ `to` W
        end if
    end if
end if

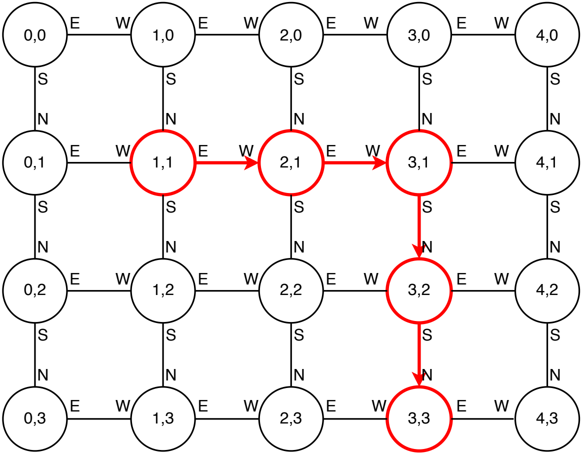
Routing in the Labeled Grid

Example: from $(1, 1)$ to $(3, 3)$?



Routing in the Labeled Grid

Example: from $(1,1)$ to $(3,3)$



Routing in the Labeled Grid

The Algorithm

Function `Latitude`(D,nx,ny)

```
if y < ny then
    return S
end if
return N
```

Function `Longitude`(D,nx,ny)

```
if x < nx then
    return E
end if
return W
```

Function `Routing`(D,nx,ny)

```
if nx = x ∧ ny = y then
    Deliver D
else if nx = x then
    Send ⟨D,nx,ny⟩ to Latitude(D,nx,ny)
else
    Send ⟨D,nx,ny⟩ to Longitude(D,nx,ny)
end if
```

Inputs

- $(x, y) \in \mathbb{N}^2$: label of the source node
- *Data*: data to transmit (initiator only)
- (dx, dy): destination label (initiator only)

Initialization

- `Routing`(*Data*,dx,dy)

Receipt of $\langle D, nx, ny\rangle$ **from** c

- `Routing`(D,nx,ny)

Pros and Cons

Pros:

- Correctness
  (if the links are reliable)
- Hop-optimal
  (from node p to node q,
  $\|p, q\| \leq \ell + L$ hops)
- Low memory usage,
  $O(\log \ell + \log L)$ bits per node
  n.b., "brute-force" routing table
  in a grid: $\Omega(\ell \times L)$ bits per node
- FIFO
- Fair

Cons:

- (Very) Particular topology
- Not robust
- Not adaptive

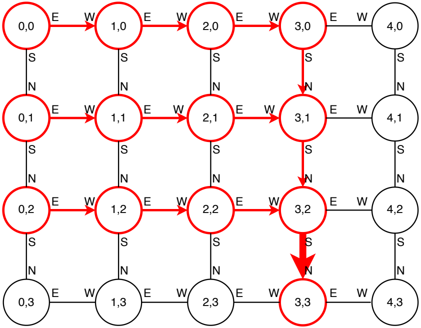
Pros and Cons

Pros:

- Correctness
  (if the links are reliable)
- Hop-optimal
  (from node p to node q,
  $\|p, q\| \leq \ell + L$ hops)
- Low memory usage,
  $O(\log \ell + \log L)$ bits per node
  n.b., “brute-force” routing table
  in a grid: $\Omega(\ell \times L)$ bits per node
- FIFO
- Fair

Cons:

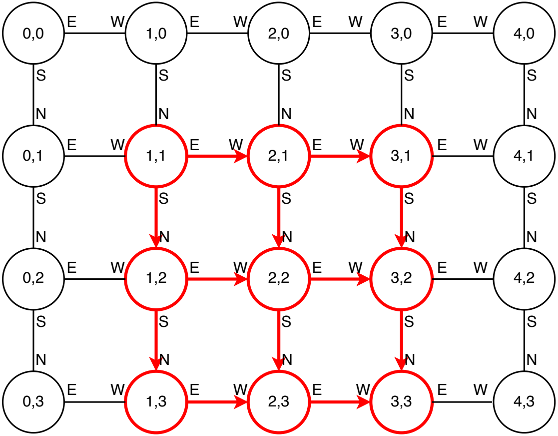
- (Very) Particular topology
- Not robust
- Not adaptive



A more adaptive solution?

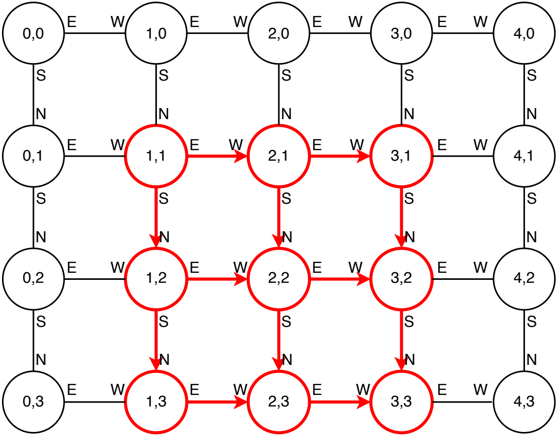
A more adaptive solution

There are several hop-optimal paths from a source to a destination.



A more adaptive solution

There are several hop-optimal paths from a source to a destination.



We can select one of them based on bandwidth.

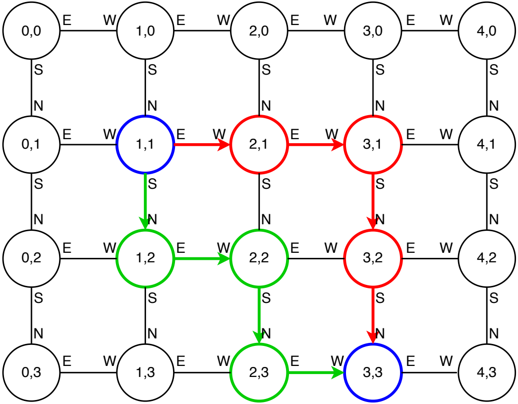
A more adaptive solution

The algorithm

Function `Routing`(D,nx,ny)

if $nx = x \wedge ny = y$ **then**
    `Deliver` D
else if $nx = x$ **then**
    `Send` $\langle D,nx,ny \rangle$ `to Latitude`(D,nx,ny)
else if $ny = y$ **then**
    `Send` $\langle D,nx,ny \rangle$ `to Longitude`(D,nx,ny)
else
    if `Bandwidth(Latitude`(D,nx,ny)`)` $>$ `Bandwidth(Longitude(Latitude`(D,nx,ny)`)))` **then**
        `Send` $\langle D,nx,ny \rangle$ `to Latitude`(D,nx,ny)
    else
        `Send` $\langle D,nx,ny \rangle$ `to Longitude`(D,nx,ny)
    end if
end if

A more adaptive solution
No more FIFO



E.g., M_A sent through the green path before M_B, sent through the red path. Yet M_B may be delivered before M_A.

A more adaptive solution

Reconstruction of the FIFO at the destination

1. A sequence number at each source.
2. The message can be tagged with the node label and the sequence number
3. Storing at the destination, the expected sequence number and a queue containing the early messages

   (only for sources that have already routed a message to the destination)

A more adaptive solution

Reconstruction of the FIFO at the destination

1. A sequence number at each source.
2. The message can be tagged with the node label and the sequence number
3. Storing at the destination, the expected sequence number and a queue containing the early messages

   (only for sources that have already routed a message to the destination)

Very costly! Worst case: $\Omega(\ell \times L \times B)$ bits, where B is the number of bits required for storing one sequence number, just for saving sequence numbers at the destination.

Bigger than the "brute-force" routing table ($\Theta(\ell \times L)$ bits per node in a grid)!

A more adaptive solution

Reconstruction of the FIFO at the destination

1. A sequence number at each source.
2. The message can be tagged with the node label and the sequence number
3. Storing at the destination, the expected sequence number and a queue containing the early messages

   (only for sources that have already routed a message to the destination)

Very costly! Worst case: $\Omega(\ell \times L \times B)$ bits, where B is the number of bits required for storing one sequence number, just for saving sequence numbers at the destination.

Bigger than the “brute-force” routing table ($\Theta(\ell \times L)$ bits per node in a grid)!

Remark

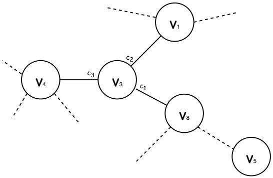
Optimization criteria for “good” routing are often conflicting: most of algorithms perform well only *w.r.t.* a subset of them.

Roadmap

Goal: compact routing tables by generalizing the grid example to arbitrary connected networks.

v_i: node label (*e.g.*, MAC address)

c_i: port number, local at the node, usually $\in [1..\delta_{v_i}]$ (δ_{v_i}: degree of v_i)



"Brute-force" routing table at v_3:

dest.	chan.
v_1	c_2
...	...
v_4	c_3
v_5	c_1
...	...
v_8	c_1
...	...

Memory requirement:
$\Omega(n \times (\log n + \log \delta_{v_i}))$ at each node v_i, where n is the total number of nodes.

"Compact" routing table at v_3:

chan.	dest.
c_1	..., v_5, ..., v_8, ...
c_2	..., v_1, ...
c_3	..., v_4, ...

Memory requirement: only δ_{v_i} cells, depends on how compactly the set of destinations for each channel can be represented.

Two ways for compacting routing tables

- **Tree-labeling Scheme**, by Santoro and Khatib [3]
- **Interval Routing**, by Leeuwen and Tan [5]

Roadmap

Principle

Goal: Considering an rooted tree network T^1 of $n > 1$ nodes, labeling of nodes from 0 to $n - 1$ in such a way that the set of destinations for each channel is a distinct interval of node labels

[1] A generalization to arbitrary connected network at the end of the subsection.

Principle

Goal: Considering an rooted tree network T^1 of $n > 1$ nodes, labeling of nodes from 0 to $n - 1$ in such a way that the set of destinations for each channel is a distinct interval of node labels

Notations:

- ring of integers modulo n: $\mathbb{Z}_n = \{0, 1, \ldots, n - 1\}$
- but integers are ordered with $<$ following $\mathbb{Z}$

[1] A generalization to arbitrary connected network at the end of the subsection.

Principle

Goal: Considering an rooted tree network T^1 of $n > 1$ nodes, labeling of nodes from 0 to $n-1$ in such a way that the set of destinations for each channel is a distinct interval of node labels

Notations:

- ring of integers modulo n: $\mathbb{Z}_n = \{0, 1, \ldots, n-1\}$
- but integers are ordered with $<$ following $\mathbb{Z}$

Remark: It is a re-labeling ! Nodes already have labels: their identifiers, which a immutable. The goal this new labeling is to compute new **small pairwise distinct labels that encodes information helping the routing**.

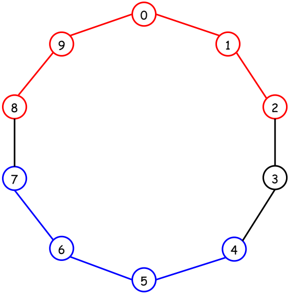
[1] A generalization to arbitrary connected network at the end of the subsection.

Cyclic Interval

A **cyclic interval** $[a, b)$ in $\mathbb{Z}_n$ in the set of integers defined by:

- $\{a, a+1, \ldots, b-1\}$ if $a < b$,
- $\{a, \ldots, n-1, 0, \ldots, b-1\}$ otherwise.

Example: let $n = 10$.



- The linear interval $[4, 8)$ is in blue
- The non-linear interval $[8, 3)$ is in red

Remarks:

- $[a, a) = \mathbb{Z}_n$
- If $a < b$, $[a, b)$ is linear,

  otherwise $[a, b)$ is non-linear.

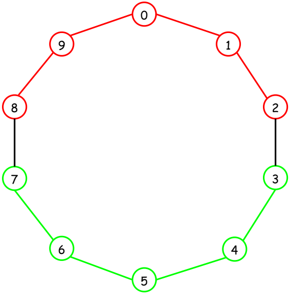
  $[a, b)$ is linear: $\forall x \in [a, b)$, $x \geq a$.
- For every $a \neq b$, the complement of $[a, b)$, *i.e.*, $\mathbb{Z}_n \setminus [a, b)$, is $[b, a)$.

Cyclic Interval

A **cyclic interval** $[a, b)$ in $\mathbb{Z}_n$ in the set of integers defined by:

- $\{a, a+1, \ldots, b-1\}$ if $a < b$,
- $\{a, \ldots, n-1, 0, \ldots, b-1\}$ otherwise.

Example: let $n = 10$.



- $[3, 8)$ is the complement of $[8, 3)$
- $[8, 3)$ is the complement of $[3, 8)$

Remarks:

- $[a, a) = \mathbb{Z}_n$
- If $a < b$, $[a, b)$ is linear,

  otherwise $[a, b)$ is non-linear.

  $[a, b)$ is linear: $\forall x \in [a, b)$, $x \geq a$.
- For every $a \neq b$, the complement of $[a, b)$, *i.e.*, $\mathbb{Z}_n \setminus [a, b)$, is $[b, a)$.

Routing using labels and cyclic intervals

Idea

For each node u of T:

- assign a (new) unique label $l_u \in \mathbb{Z}_n$ to u
- order channel from 1 to δ_u and assign a label $\alpha_i(u)$ to the ith channel outgoing from u

in such a way that for each node v:

- either $l_v = l_u$ (and so $v = u$)
- or $l_v \neq l_u$ and there exists $i \in \{1, \ldots, \delta_u\}$ such that
  - $l_v \in [\alpha_i(u), \alpha_{(i \bmod \delta_u)+1}(u))$ and
  - the link designated by $\alpha_i(u)$ is on the path from u to v

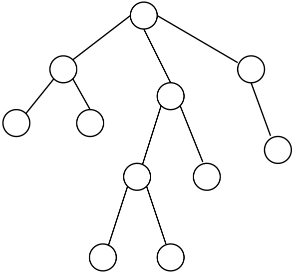
Routing using labels and cyclic intervals

The algorithm

Given a packet p with destination label d at node u.

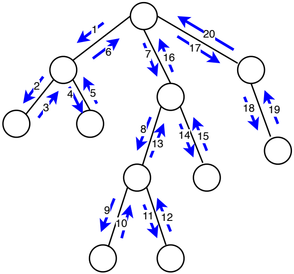
if $d = l_u$ **then**
    `deliver` p
else
    let $\alpha_i(u)$ such that $d \in [\alpha_i(u), \alpha_{i \bmod \delta_u + 1}(u))$
    send p via the channel labeled with $\alpha_i(u)$
end if

Node Labeling



A rooted tree of $n = 11$ nodes
(the uppermost node is the root)

Node Labeling

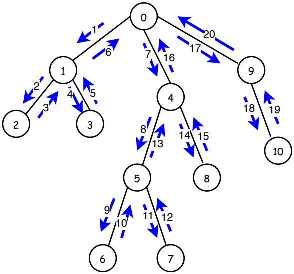


A rooted tree of $n = 11$ nodes
(the uppermost node is the root)

Preorder tree traversal

(computed by a token circulation in $2n - 2$ rounds)

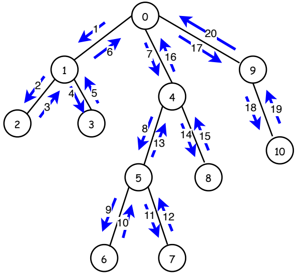
Node Labeling



A rooted tree of $n = 11$ nodes
(the uppermost node is the root)

Preorder tree traversal + node labeling

Node Labeling



A rooted tree of $n = 11$ nodes
(the uppermost node is the root)

Preorder tree traversal + node labeling

Properties:

1. $l_u + |T(u)| \leq n$

   E.g., for node with label 4, we have:
   $4 + 5 = 9 \leq 11$

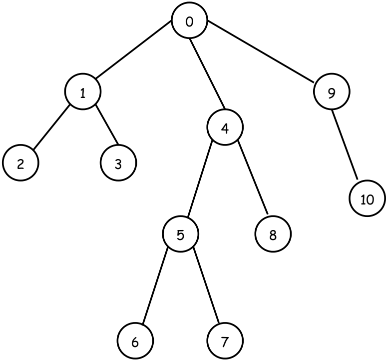
2. Labels in $T(u) : \{l_u, \dots, l_u + |T(u)| - 1\}$

   E.g., nodes in the subtree of the node with label 4 are numbered from 4 to 8 (*i.e.*, 4+5-1)

3. Let v and w be two children of u such that l_w is the label immediately greater than l_v among the labels of u's neighbors: $l_w = l_v + |T(v)|$

   E.g., consider the nodes of label 1 and 4:
   $4 = 1 + 3$

Channel Labeling

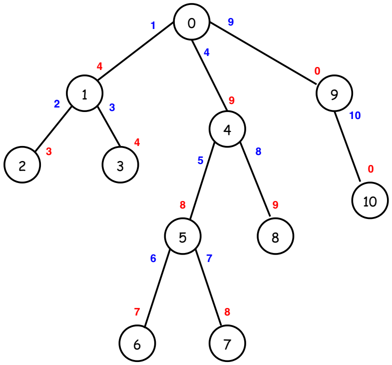


A rooted tree of $n = 11$ nodes
(the uppermost node is the root)

Labeling: Let u be a node. For every neighbor v of u, we assign the label $\alpha_v(u)$ to the channel of u outgoing to v as follows:

- $\alpha_v(u) = l_v$ if v is a child of u
- $\alpha_v(u) = (l_u + |T(u)|) \bmod n$ if v is the parent of u.

Channel Labeling

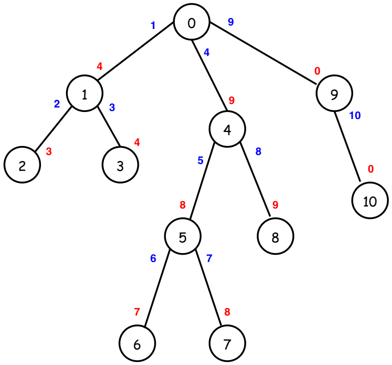


A rooted tree of $n = 11$ nodes
(the uppermost node is the root)

Labeling: Let u be a node. For every neighbor v of u, we assign the label $\alpha_v(u)$ to the channel of u outgoing to v as follows:

- $\alpha_v(u) = l_v$ if v is a child of u
- $\alpha_v(u) = (l_u + |T(u)|) \bmod n$ if v is the parent of u.

Channel Labeling



A rooted tree of $n = 11$ nodes
(the uppermost node is the root)

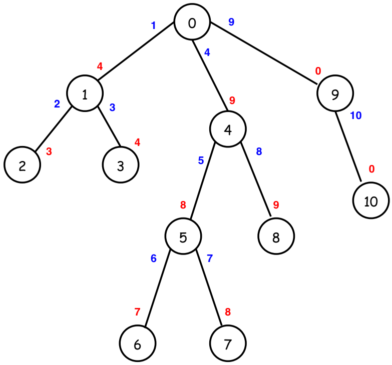
Labeling: Let u be a node. For every neighbor v of u, we assign the label $\alpha_v(u)$ to the channel of u outgoing to v as follows:

- $\alpha_v(u) = l_v$ if v is a child of u
- $\alpha_v(u) = (l_u + |T(u)|) \bmod n$ if v is the parent of u.

Let $\alpha_1(u), \ldots, \alpha_{\delta_u}(u)$ be the channel labels at u sorted in increasing order according to values $\alpha_v(u)$.

Examples:

- Let u be the node of label 4: $\alpha_1(u) = 5$, $\alpha_2(u) = 8$, $\alpha_3(u) = 9$
- Let u be the node of label 9: $\alpha_1(u) = 0$ (*i.e.*, 11 mod 11), $\alpha_2(u) = 10$



A rooted tree of $n = 11$ nodes
(the uppermost node is the root)

Labeling: Let u be a node. For every neighbor v of u, we assign the label $\alpha_v(u)$ to the channel of u outgoing to v as follows:

- $\alpha_v(u) = l_v$ if v is a child of u
- $\alpha_v(u) = (l_u + |T(u)|) \bmod n$ if v is the parent of u.

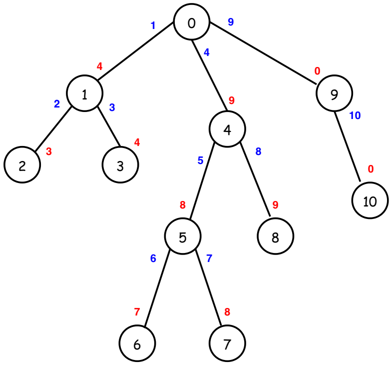
Let $\alpha_1(u), \ldots, \alpha_{\delta_u}(u)$ be the channel labels at u sorted in increasing order according to values $\alpha_v(u)$.

Examples:

- Let u be the node of label 4: $\alpha_1(u) = 5$, $\alpha_2(u) = 8$, $\alpha_3(u) = 9$
- Let u be the node of label 9: $\alpha_1(u) = 0$ (*i.e.*, 11 mod 11), $\alpha_2(u) = 10$

Properties:

1. $0 \leq \alpha_1(u) < \ldots < \alpha_{\delta_u}(u) < n$
2. $\forall i \in [1..\delta_u - 1]$, $[\alpha_i(u), \alpha_{(i \bmod \delta_u)+1}(u))$ is linear
3. $[\alpha_{\delta_u(u)}, \alpha_1(u))$ is non-linear
4. Let u be a non-root node and f its parent. The label of the channel of u outgoing to f is $\alpha_1(u)$ or $\alpha_{\delta_u}(u)$
5. Let $i < \delta_u$, v and w be two children of u whose labels are at ranks i^{th} and $i{+}1^{th}$ in the order of labels at u. $[\alpha_i(u), \alpha_{(i \bmod \delta_u)+1}(u)) = [\alpha_i(u), \alpha_{i+1}(u)) = [l_v, l_w) = [l_v, l_v + [T(v)|)$



A rooted tree of $n = 11$ nodes
(the uppermost node is the root)

Remark: Assuming n is known, the channel labeling can be also computed during the token circulation, otherwise n can computed beforehand using a PIF or a token circulation.

Labeling: Let u be a node. For every neighbor v of u, we assign the label $\alpha_v(u)$ to the channel of u outgoing to v as follows:

- $\alpha_v(u) = l_v$ if v is a child of u
- $\alpha_v(u) = (l_u + |T(u)|) \bmod n$ if v is the parent of u.

Let $\alpha_1(u), \ldots, \alpha_{\delta_u}(u)$ be the channel labels at u sorted in increasing order according to values $\alpha_v(u)$.

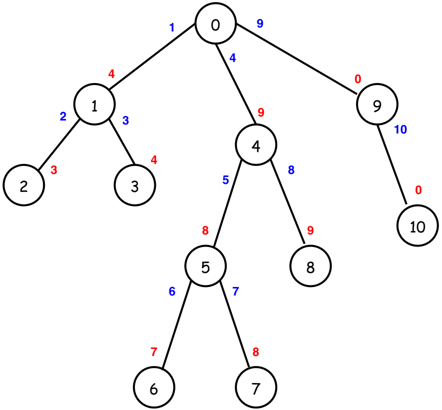
Examples:

- Let u be the node of label 4: $\alpha_1(u) = 5$, $\alpha_2(u) = 8$, $\alpha_3(u) = 9$
- Let u be the node of label 9: $\alpha_1(u) = 0$ (*i.e.*, 11 mod 11), $\alpha_2(u) = 10$

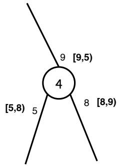
Properties:

1. $0 \leq \alpha_1(u) < \ldots < \alpha_{\delta_u}(u) < n$
2. $\forall i \in [1..\delta_u - 1]$, $[\alpha_i(u), \alpha_{(i \bmod \delta_u)+1}(u))$ is linear
3. $[\alpha_{\delta_u(u)}, \alpha_1(u))$ is non-linear
4. Let u be a non-root node and f its parent. The label of the channel of u outgoing to f is $\alpha_1(u)$ or $\alpha_{\delta_u}(u)$
5. Let $i < \delta_u$, v and w be two children of u whose labels are at ranks i^{th} and $i{+}1^{th}$ in the order of labels at u. $[\alpha_i(u), \alpha_{(i \bmod \delta_u)+1}(u)) = [\alpha_i(u), \alpha_{i+1}(u)) = [l_v, l_w) = [l_v, l_v + [T(v)|)$

Local View

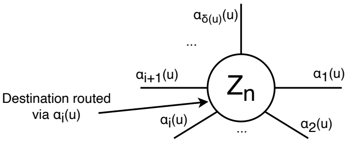


A rooted tree of $n = 11$ nodes
(the uppermost node is the root)



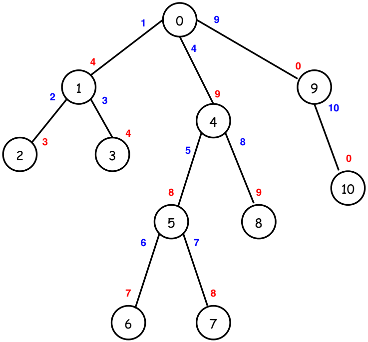
Local view at node u of label 4

- $[\alpha_1(u), \alpha_2(u)) = [5, 8) = \{5, 6, 7\}$
- $[\alpha_2(u), \alpha_3(u)) = [8, 9) = \{8\}$
- $[\alpha_3(u), \alpha_1(u)) = [9, 5) = \{9, 10, 0, 1, 2, 3, 4\}$

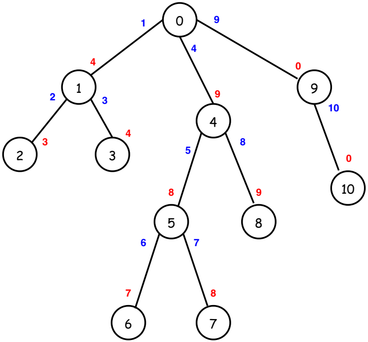


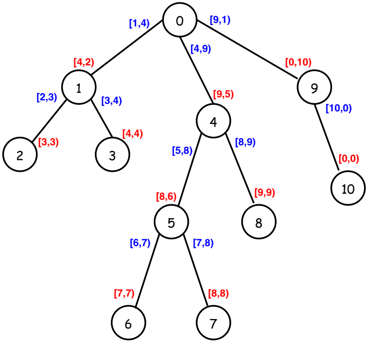
General scheme

Exercise: give all the intervals?

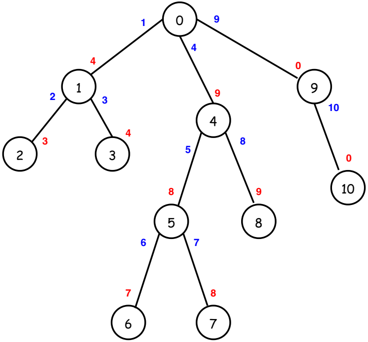


Exercise: give all the intervals?

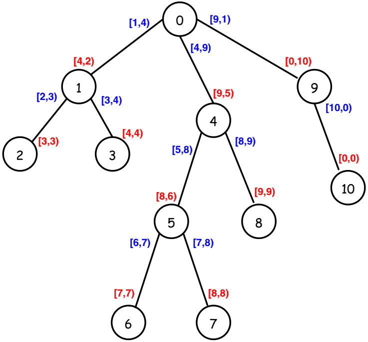




Remark: At 6, $[7,7) = \{7,8,9,10,0,1,2,3,4,5,6\} = \mathbb{Z}_{11}$



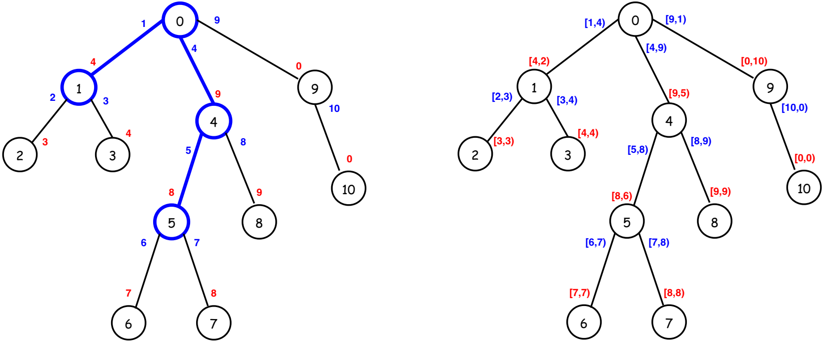
0
1
4
1
2
3
2
3
3
4
4
9
5
8
8
5
6
7
7
6
8
7
8
9
8
9
0
9
10
0
10



0
[1,4)
[9,1)
[4,9)
[4,2)
1
[2,3)
[3,4)
[3,3)
2
[4,4)
3
[9,5)
4
[5,8)
[8,9)
[8,6)
5
[6,7)
[7,8)
[7,7)
6
[8,8)
7
[9,9)
8
[0,10)
9
[10,0)
[0,0)
10

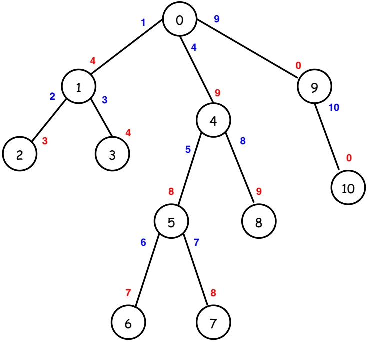
Example of routing through the labeled Tree

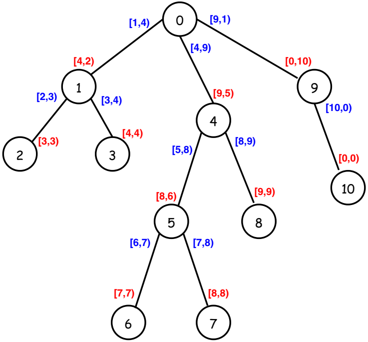
From label 5 to label 1



At 5, $1 \in [8, 6)$
At 4, $1 \in [9, 5)$
At 0, $1 \in [1, 4)$

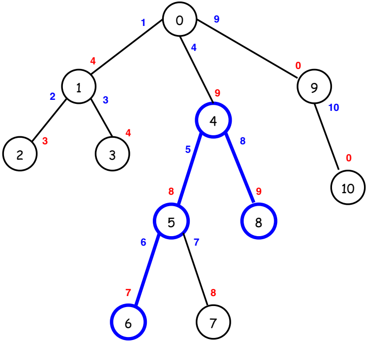
Example of routing through the labeled Tree
From label 6 to label 8

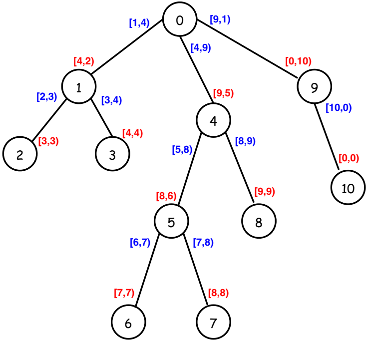




Example of routing through the labeled Tree

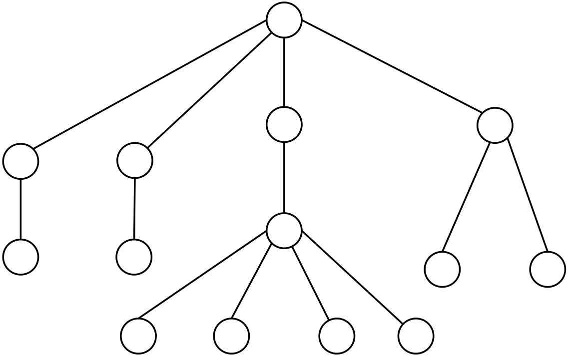
From label 6 to label 8





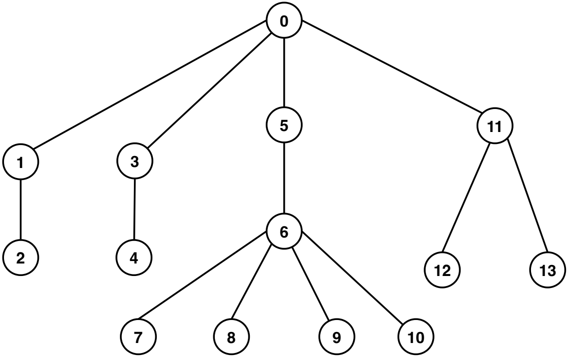
At 6, $8 \in [7,7)$
At 5, $8 \in [8,6)$
At 4, $8 \in [8,9)$

Exercise



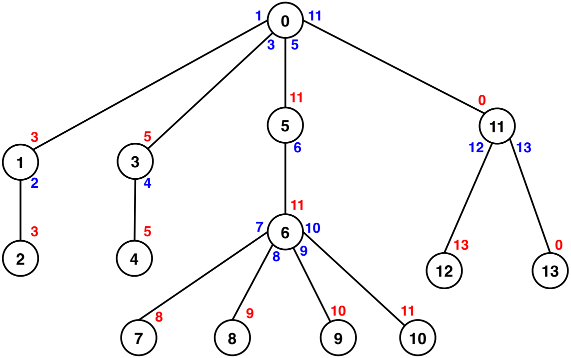
A tree (the uppermost node is the root)

Exercise



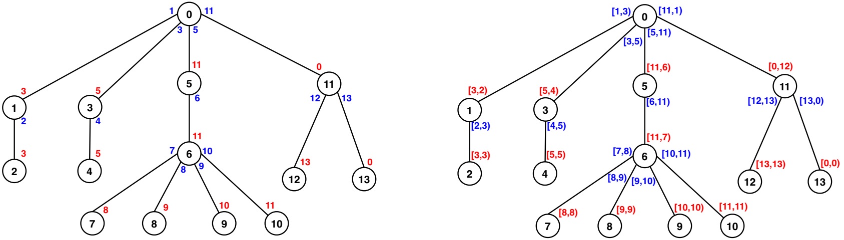
Node Labeling

Exercise



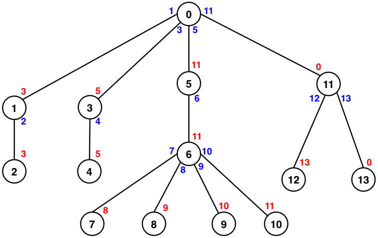
Channel Labeling

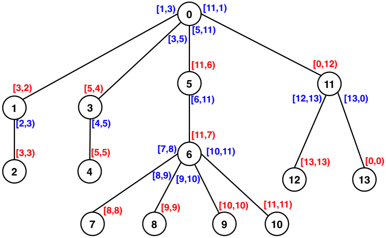
Exercise



Intervals

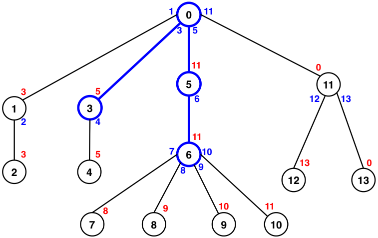
Exercise

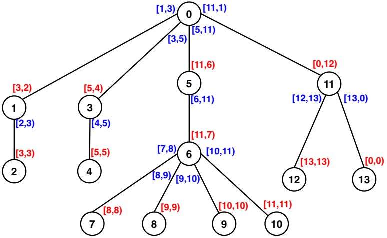




Routing path from 3 to 6?

Exercise

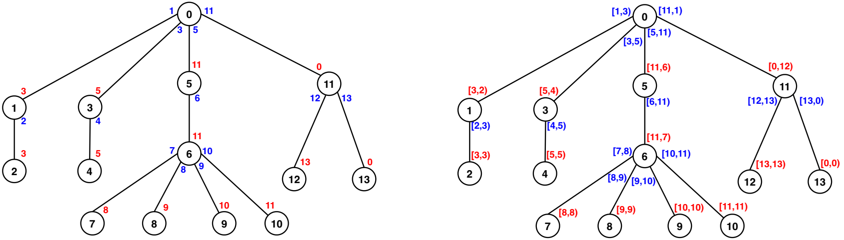




Routing path from 3 to 6:

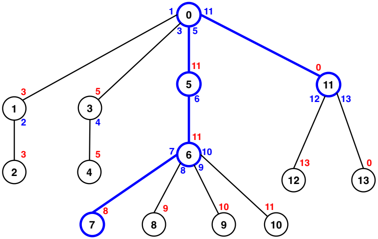
At 3, $6 \in [5, 4)$ At 0, $6 \in [5, 11)$ At 5, $6 \in [6, 11)$

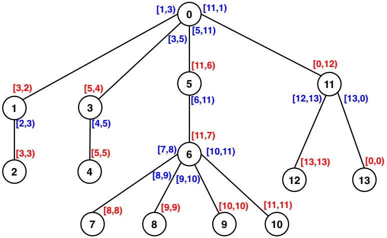
Exercise



Routing path from 11 to 7?

Exercise





Routing path from 11 to 7:

At 11, $7 \in [0, 12)$ At 0, $7 \in [5, 11)$ At 5, $7 \in [6, 11)$ At 6, $7 \in [7, 8)$

Correctness

Using the routing algorithm, each packet is eventually delivered to its final destination

Recall: The sets $S_1, \ldots, S_k$ forms a partition of the set X if the following two conditions hold

- $\bigcup_{i \in \{1,\ldots,k\}} S_i = X$ (*Union Property*)
- $\forall i, j \in \{1, \ldots, k\}$, $i \neq j \Rightarrow S_i \cap S_j = \emptyset$ (*Intersection Property*)

Correctness

Using the routing algorithm, each packet is eventually delivered to its final destination

Preliminary result:

$[\alpha_i(u), \alpha_{(i \bmod \delta_u)+1}(u))$, $i \in \{1, \ldots, \delta_u\}$ **is a partition of** $\mathbb{Z}_n$

Correctness

Using the routing algorithm, each packet is eventually delivered to its final destination

Preliminary result:

$$[\alpha_i(u), \alpha_{(i \bmod \delta_u)+1}(u)),\ i \in \{1, \ldots, \delta_u\}$$ **is a partition of** $\mathbb{Z}_n$

Proof. Two cases:

- $\delta_u = 1$:

- $\delta_u > 1$:

□

Correctness

Using the routing algorithm, each packet is eventually delivered to its final destination

Preliminary result:

$$[\alpha_i(u), \alpha_{(i \bmod \delta_u)+1}(u)),\ i \in \{1, \ldots, \delta_u\} \textbf{ is a partition of } \mathbb{Z}_n$$

Proof. Two cases:

- $\delta_u = 1$:
  The intersection property trivially holds.
  Then, $\bigcup_{i \in \{1,\ldots,\delta_u\}} [\alpha_i(u), \alpha_{(i \bmod \delta_u)+1}(u)) = [\alpha_1(u), \alpha_1(u)) = \mathbb{Z}_n$
  That is, the union property also holds
- $\delta_u > 1$:

□

Correctness

Using the routing algorithm, each packet is eventually delivered to its final destination

Preliminary result:

$$[\alpha_i(u), \alpha_{(i \bmod \delta_u)+1}(u)),\ i \in \{1, \ldots, \delta_u\} \text{ is a partition of } \mathbb{Z}_n$$

Proof. Two cases:

- $\delta_u = 1$:

  The intersection property trivially holds.

  Then, $\bigcup_{i \in \{1,\ldots,\delta_u\}} [\alpha_i(u), \alpha_{(i \bmod \delta_u)+1}(u)) = [\alpha_1(u), \alpha_1(u)) = \mathbb{Z}_n$

  That is, the union property also holds

- $\delta_u > 1$:

  By definition, we have the following property

  1. $[\alpha_{\delta_u}(u), \alpha_1(u))$ is the complement of $[\alpha_1(u), \alpha_{\delta_u}(u))$

  So, $\bigcup_{i \in \{1,\ldots,\delta_u\}} [\alpha_i(u), \alpha_{(i \bmod \delta_u)+1}(u)) = [\alpha_1(u), \alpha_{\delta_u}(u)) \cup [\alpha_{\delta_u}(u), \alpha_1(u)) = \mathbb{Z}_n$: the union property holds.

  The intersection property also holds, as shown in the next slide.

□

Correctness

Partition: Intersection Property

Let $x \in \mathbb{Z}_n$.

Assume, by contradiction, that

- $x \in [\alpha_i(u), \alpha_{(i \bmod \delta_u)+1}(u))$ and
- $x \in [\alpha_j(u), \alpha_{(j \bmod \delta_u)+1}(u))$

with $i, j \in \{1, \ldots, \delta_u\}$ and $i < j$ (so $\delta_u > 1$).

Correctness

Partition: Intersection Property

Let $x \in \mathbb{Z}_n$.

Assume, by contradiction, that

- $x \in [\alpha_i(u), \alpha_{(i \bmod \delta_u)+1}(u))$ and
- $x \in [\alpha_j(u), \alpha_{(j \bmod \delta_u)+1}(u))$

with $i, j \in \{1, \ldots, \delta_u\}$ and $i < j$ (so $\delta_u > 1$).

Since $i < \delta_u$, $[\alpha_i(u), \alpha_{(i \bmod \delta_u)+1}(u)) \subseteq [\alpha_1(u), \alpha_{\delta_u}(u))$, which implies $j < \delta_u$ by Property 1.

Correctness

Partition: Intersection Property

Let $x \in \mathbb{Z}_n$.

Assume, by contradiction, that

- $x \in [\alpha_i(u), \alpha_{(i \bmod \delta_u)+1}(u))$ and
- $x \in [\alpha_j(u), \alpha_{(j \bmod \delta_u)+1}(u))$

with $i, j \in \{1, \ldots, \delta_u\}$ and $i < j$ (so $\delta_u > 1$).

Since $i < \delta_u$, $[\alpha_i(u), \alpha_{(i \bmod \delta_u)+1}(u)) \subseteq [\alpha_1(u), \alpha_{\delta_u}(u))$, which implies $j < \delta_u$ by Property 1.

So, $i < \delta_u - 1$ and $x \in [\alpha_i(u), \alpha_{(i \bmod \delta_u)+1}(u))$ implies
$x < \alpha_{(i \bmod \delta_u)+1}(u) = \alpha_{i+1}(u) \leq \alpha_j(u)$.

Let $x \in \mathbb{Z}_n$.

Assume, by contradiction, that

- $x \in [\alpha_i(u), \alpha_{(i \bmod \delta_u)+1}(u))$ and
- $x \in [\alpha_j(u), \alpha_{(j \bmod \delta_u)+1}(u))$

with $i, j \in \{1, \dots, \delta_u\}$ and $i < j$ (so $\delta_u > 1$).

Since $i < \delta_u$, $[\alpha_i(u), \alpha_{(i \bmod \delta_u)+1}(u)) \subseteq [\alpha_1(u), \alpha_{\delta_u}(u))$, which implies $j < \delta_u$ by Property 1.

So, $i < \delta_u - 1$ and $x \in [\alpha_i(u), \alpha_{(i \bmod \delta_u)+1}(u))$ implies
$x < \alpha_{(i \bmod \delta_u)+1}(u) = \alpha_{i+1}(u) \leq \alpha_j(u)$.

Now, since $j < \delta_u$, $\forall y \in [\alpha_j(u), \alpha_{(j \bmod \delta_u)+1}(u))$, $y \geq \alpha_j(u)$ (this interval is linear).

Correctness

Partition: Intersection Property

Let $x \in \mathbb{Z}_n$.

Assume, by contradiction, that

- $x \in [\alpha_i(u), \alpha_{(i \bmod \delta_u)+1}(u))$ and
- $x \in [\alpha_j(u), \alpha_{(j \bmod \delta_u)+1}(u))$

with $i, j \in \{1, \ldots, \delta_u\}$ and $i < j$ (so $\delta_u > 1$).

Since $i < \delta_u$, $[\alpha_i(u), \alpha_{(i \bmod \delta_u)+1}(u)) \subseteq [\alpha_1(u), \alpha_{\delta_u}(u))$, which implies $j < \delta_u$ by Property 1.

So, $i < \delta_u - 1$ and $x \in [\alpha_i(u), \alpha_{(i \bmod \delta_u)+1}(u))$ implies $x < \alpha_{(i \bmod \delta_u)+1}(u) = \alpha_{i+1}(u) \leq \alpha_j(u)$.

Now, since $j < \delta_u$, $\forall y \in [\alpha_j(u), \alpha_{(j \bmod \delta_u)+1}(u))$, $y \geq \alpha_j(u)$ (this interval is linear).

Thus, $x \notin [\alpha_j(u), \alpha_{(j \bmod \delta_u)+1}(u))$, a contradiction.

The result follows. $\square$

Correctness

Thanks to the previous property: when u has a packet for $v \neq u$, u finds a unique destination w for the next hop.

Correctness

Thanks to the previous property: when u has a packet for $v \neq u$, u finds a unique destination w for the next hop.

The correctness is based on the following two properties:

1. If $v \notin T(u)$, then w is the parent of u.
2. If $v \in T(u)$, then w is the child of u such that $v \in T(w)$.

Correctness

Thanks to the previous property: when u has a packet for $v \neq u$, u finds a unique destination w for the next hop.

The correctness is based on the following two properties:

1. If $v \notin T(u)$, then w is the parent of u.
2. If $v \in T(u)$, then w is the child of u such that $v \in T(w)$.

Proof. Let H be the height of T.

- By Property 1, $v \in T(u)$ after at most H hops.
- By Property 2, if $v \in T(u)$, then $v \in T(w)$: the property is invariant.
- By Property 2, once $v \in T(u)$, the packet is deliver to its destination within at most H hops.

$\square$

Proof of Property 1

If $v \notin T(u)$, then w is the parent of u

Proof: u is not the root. Let f be the parent of u.

If $\delta_u = 1$, then the property trivially holds.

□

Proof of Property 1

If $v \notin T(u)$, then w is the parent of u

Proof: u is not the root. Let f be the parent of u.

If $\delta_u = 1$, then the property trivially holds.

Assume now that $\delta_u > 1$.

$v \notin T(u)$: $l_v < l_u$ or $l_v \geq l_u + |T(u)|$

$\square$

Proof of Property 1

If $v \notin T(u)$, then w is the parent of u

Proof: u is not the root. Let f be the parent of u.

If $\delta_u = 1$, then the property trivially holds.

Assume now that $\delta_u > 1$.

$v \notin T(u)$: $l_v < l_u$ or $l_v \geq l_u + |T(u)|$

Two cases:

1. The label of f is $\alpha_1(u)$

2. The label of f is $\alpha_{\delta_u}(u)$

□

Proof of Property 1

If $v \notin T(u)$, then w is the parent of u

Proof: u is not the root. Let f be the parent of u.

If $\delta_u = 1$, then the property trivially holds.

Assume now that $\delta_u > 1$.

$v \notin T(u)$: $l_v < l_u$ or $l_v \geq l_u + |T(u)|$

Two cases:

1. The label of f is $\alpha_1(u)$

   $\alpha_1(u) = (l_u + |T(u)|) \bmod n < \alpha_2(u) = l_u + 1$

   So $\alpha_1(u) = 0$, which implies that $l_u + |T(u)| = n$: $0 \leq l_v < l_u$[2]

   Now, $[\alpha_1(u), \alpha_2(u)) = [0, l_u + 1) = \{0, \ldots, l_u\}$

   So, $l_v \in [\alpha_1(u), \alpha_2(u))$: $w = f$

2. The label of f is $\alpha_{\delta_u}(u)$

□

[2] Actually, in this case, $\forall x \notin T(u)$, $l_x < l_u$

Proof of Property 1

If $v \notin T(u)$, then w is the parent of u

Proof: u is not the root. Let f be the parent of u.

If $\delta_u = 1$, then the property trivially holds.

Assume now that $\delta_u > 1$.

$v \notin T(u)$: $l_v < l_u$ or $l_v \geq l_u + |T(u)|$

Two cases:

1. The label of f is $\alpha_1(u)$

   $\alpha_1(u) = (l_u + |T(u)|) \bmod n < \alpha_2(u) = l_u + 1$

   So $\alpha_1(u) = 0$, which implies that $l_u + |T(u)| = n$: $0 \leq l_v < l_u$[2]

   Now, $[\alpha_1(u), \alpha_2(u)) = [0, l_u + 1) = \{0, \ldots, l_u\}$

   So, $l_v \in [\alpha_1(u), \alpha_2(u))$: $w = f$

2. The label of f is $\alpha_{\delta_u}(u)$

   $\alpha_{\delta_u}(u) = (l_u + |T(u)|) \bmod n = l_u + |T(u)|$ and $[\alpha_{\delta_u}(u), \alpha_1(u))$ is non-linear since $\alpha_1(u) < \alpha_{\delta_u}(u)$

   $[\alpha_{\delta_u}(u), \alpha_1(u)) = [l_u + |T(u)|, l_u + 1)$

   If $l_v < l_u$, $l_v \in [\alpha_{\delta_u}(u), \alpha_1(u))$ since $\{0, \ldots, l_u\} \subseteq [l_u + |T(u)|, l_u + 1)$

   If $l_v \geq l_u + [T(u)|$, $l_v \in [\alpha_{\delta_u}(u), \alpha_1(u))$ since $\{l_u + |T(u)|, \ldots, n-1\} \subseteq [l_u + |T(u)|, l_u + 1)$

   Hence, $l_v \in [\alpha_{\delta_u}(u), \alpha_1(u))$: $w = f$

$\square$

[2] Actually, in this case, $\forall x \notin T(u)$, $l_x < l_u$

Proof of Property 2

If $v \in T(u)$, then w is the child of u such that $v \in T(w)$

Proof: $v \in T(u)$: $l_v \in \{l_u, \ldots, l_u + |T(u)| - 1\}$

$\square$

Proof of Property 2

If $v \in T(u)$, then w is the child of u such that $v \in T(w)$

Proof: $v \in T(u)$: $l_v \in \{l_u, \ldots, l_u + |T(u)| - 1\}$

$v \neq u$: $v \in T(x)$ with x a child of u
$l_v \in \{l_x, \ldots, l_x + |T(x)| - 1\}$

□

Proof of Property 2

If $v \in T(u)$, then w is the child of u such that $v \in T(w)$

Proof: $v \in T(u)$: $l_v \in \{l_u, \dots, l_u + |T(u)| - 1\}$

$v \neq u$: $v \in T(x)$ with x a child of u

$l_v \in \{l_x, \dots, l_x + |T(x)| - 1\}$

Let i be the rank of $\alpha_x(u)$ in the order of labels at u

$\alpha_i(u) = l_x$: $l_v \geq \alpha_i(u)$

□

Proof of Property 2

If $v \in T(u)$, then w is the child of u such that $v \in T(w)$

Proof: $v \in T(u)$: $l_v \in \{l_u, \dots, l_u + |T(u)| - 1\}$

$v \neq u$: $v \in T(x)$ with x a child of u

$l_v \in \{l_x, \dots, l_x + |T(x)| - 1\}$

Let i be the rank of $\alpha_x(u)$ in the order of labels at u

$\alpha_i(u) = l_x$: $l_v \geq \alpha_i(u)$

2 cases:

1. $i = \delta_u$:

2. $i < \delta_u$:

□

Proof of Property 2

If $v \in T(u)$, then w is the child of u such that $v \in T(w)$

Proof: $v \in T(u)$: $l_v \in \{l_u, \dots, l_u + |T(u)| - 1\}$

$v \neq u$: $v \in T(x)$ with x a child of u

$l_v \in \{l_x, \dots, l_x + |T(x)| - 1\}$

Let i be the rank of $\alpha_x(u)$ in the order of labels at u

$\alpha_i(u) = l_x$: $l_v \geq \alpha_i(u)$

2 cases:

1. $i = \delta_u$: $[\alpha_{\delta_u}(u), \alpha_1(u))$ is non-linear
   So, $\{\alpha_{\delta_u}(u), \dots, n-1\} \subseteq [\alpha_{\delta_u}(u), \alpha_1(u))$
   Thus, $l_v \in [\alpha_{\delta_u}(u), \alpha_1(u))$: $w = x$

2. $i < \delta_u$:

□

Proof of Property 2

If $v \in T(u)$, then w is the child of u such that $v \in T(w)$

Proof: $v \in T(u)$: $l_v \in \{l_u, \ldots, l_u + |T(u)| - 1\}$

$v \neq u$: $v \in T(x)$ with x a child of u
$l_v \in \{l_x, \ldots, l_x + |T(x)| - 1\}$
Let i be the rank of $\alpha_x(u)$ in the order of labels at u
$\alpha_i(u) = l_x$: $l_v \geq \alpha_i(u)$

2 cases:

1. $i = \delta_u$: $[\alpha_{\delta_u}(u), \alpha_1(u))$ is non-linear
   So, $\{\alpha_{\delta_u}(u), \ldots, n-1\} \subseteq [\alpha_{\delta_u}(u), \alpha_1(u))$
   Thus, $l_v \in [\alpha_{\delta_u}(u), \alpha_1(u))$: $w = x$

2. $i < \delta_u$: $\delta_u > 1$ and 2 subcases
   - u is not the root and $i + 1$ is the rank of the label $\alpha_f(u)$ of the parent f of u.
   - $\alpha_{i+1}(u) = l_y$ where $\alpha_y(u)$ is the $i + 1^{th}$ label in the order of labels at u.

□

Proof of Property 2

If $v \in T(u)$, then w is the child of u such that $v \in T(w)$

Proof: $v \in T(u)$: $l_v \in \{l_u, \dots, l_u + |T(u)| - 1\}$

$v \neq u$: $v \in T(x)$ with x a child of u
$l_v \in \{l_x, \dots, l_x + |T(x)| - 1\}$
Let i be the rank of $\alpha_x(u)$ in the order of labels at u
$\alpha_i(u) = l_x$: $l_v \geq \alpha_i(u)$

2 cases:

1. $i = \delta_u$: $[\alpha_{\delta_u}(u), \alpha_1(u))$ is non-linear
   So, $\{\alpha_{\delta_u}(u), \dots, n-1\} \subseteq [\alpha_{\delta_u}(u), \alpha_1(u))$
   Thus, $l_v \in [\alpha_{\delta_u}(u), \alpha_1(u))$: $w = x$

2. $i < \delta_u$: $\delta_u > 1$ and 2 subcases
   - u is not the root and $i+1$ is the rank of the label $\alpha_f(u)$ of the parent f of u.
     $\alpha_{i+1}(u) = l_u + |T(u)| \bmod n = l_u + |T(u)|$ since $[\alpha_i(u), \alpha_{i+1}(u))$ is linear:
     $[\alpha_i(u), \alpha_{i+1}(u)) = \{l_x, \dots, l_u + |T(u)| - 1\}$
   - $\alpha_{i+1}(u) = l_y$ where $\alpha_y(u)$ is the $i + 1^{th}$ label in the order of labels at u.

□

Proof of Property 2

If $v \in T(u)$, then w is the child of u such that $v \in T(w)$

Proof: $v \in T(u)$: $l_v \in \{l_u, \dots, l_u + |T(u)| - 1\}$

$v \neq u$: $v \in T(x)$ with x a child of u

$l_v \in \{l_x, \dots, l_x + |T(x)| - 1\}$

Let i be the rank of $\alpha_x(u)$ in the order of labels at u

$\alpha_i(u) = l_x$: $l_v \geq \alpha_i(u)$

2 cases:

1. $i = \delta_u$: $[\alpha_{\delta_u}(u), \alpha_1(u))$ is non-linear
   So, $\{\alpha_{\delta_u}(u), \dots, n-1\} \subseteq [\alpha_{\delta_u}(u), \alpha_1(u))$
   Thus, $l_v \in [\alpha_{\delta_u}(u), \alpha_1(u))$: $w = x$

2. $i < \delta_u$: $\delta_u > 1$ and 2 subcases
   - u is not the root and $i+1$ is the rank of the label $\alpha_f(u)$ of the parent f of u.
     $\alpha_{i+1}(u) = l_u + |T(u)| \bmod n = l_u + |T(u)|$ since $[\alpha_i(u), \alpha_{i+1}(u))$ is linear:
     $[\alpha_i(u), \alpha_{i+1}(u)) = \{l_x, \dots, l_u + |T(u)| - 1\}$
   - $\alpha_{i+1}(u) = l_y$ where $\alpha_y(u)$ is the $i+1^{th}$ label in the order of labels at u.
     $\alpha_{i+1}(u) = l_x + |T(x)|$: $[\alpha_i(u), \alpha_{i+1}(u)) = \{l_x, \dots, l_x + |T(x)| - 1\}$

□

Proof of Property 2

If $v \in T(u)$, then w is the child of u such that $v \in T(w)$

Proof: $v \in T(u)$: $l_v \in \{l_u, \ldots, l_u + |T(u)| - 1\}$

$v \neq u$: $v \in T(x)$ with x a child of u

$l_v \in \{l_x, \ldots, l_x + |T(x)| - 1\}$

Let i be the rank of $\alpha_x(u)$ in the order of labels at u

$\alpha_i(u) = l_x$: $l_v \geq \alpha_i(u)$

2 cases:

1. $i = \delta_u$: $[\alpha_{\delta_u}(u), \alpha_1(u))$ is non-linear
   So, $\{\alpha_{\delta_u}(u), \ldots, n-1\} \subseteq [\alpha_{\delta_u}(u), \alpha_1(u))$
   Thus, $l_v \in [\alpha_{\delta_u}(u), \alpha_1(u))$: $w = x$

2. $i < \delta_u$: $\delta_u > 1$ and 2 subcases
   - u is not the root and $i+1$ is the rank of the label $\alpha_f(u)$ of the parent f of u.
     $\alpha_{i+1}(u) = l_u + |T(u)| \bmod n = l_u + |T(u)|$ since $[\alpha_i(u), \alpha_{i+1}(u))$ is linear:
     $[\alpha_i(u), \alpha_{i+1}(u)) = \{l_x, \ldots, l_u + |T(u)| - 1\}$
   - $\alpha_{i+1}(u) = l_y$ where $\alpha_y(u)$ is the $i+1^{th}$ label in the order of labels at u.
     $\alpha_{i+1}(u) = l_x + |T(x)|$: $[\alpha_i(u), \alpha_{i+1}(u)) = \{l_x, \ldots, l_x + |T(x)| - 1\}$

   In both subcases, $l_v \in [\alpha_i(u), \alpha_{i+1}(u))$: $w = x$

$\square$

Complexity Analysis

- Distributed Computation of the Labeling (using a token circulation):
  - $O(n)$ rounds / messages
  - message length: $O(\log n)$ bits per message
- Memory Usage: $\delta_u + 1$ labels for node u, *i.e.*, $(\delta_u + 1) \times \lceil \log n \rceil$ bits
- Routing from u to v: $\|u, v\|$ hops (hop-optimal)

Generalization to arbitrary connected networks

Leader election + spanning tree (with initialization and term. detect. at leader), token circulation in the tree

($O(mn)$ messages, $O(m)$ rounds, and $O(\delta_u + B)$ bits, where B the number of bits to store an identifier)

(*cf.*, distributed computing courses)

Pros.

- Correctness
- Time complexity: a packet is routed in at most $\min(n-1, 2H)$ hops where $H < n$ is the height of the tree.

  If the tree is *BFS*, at most $\min(n-1, 2D)$ hops where D is the network diameter.
- Memory Usage: at most δ_u+1 labels for node u, *i.e.*, at most $(\delta_u+1) \times \lceil \log n \rceil$ bits

Generalization to arbitrary connected networks

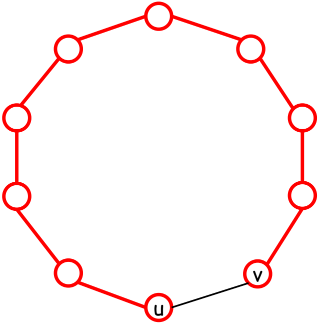
Leader election + spanning tree (with initialization and term. detect. at leader), token circulation in the tree ($O(mn)$ messages, $O(m)$ rounds, and $O(\delta_u + B)$ bits, where B the number of bits to store an identifier)
(*cf.*, distributed computing courses)

Pros.

- Correctness
- Time complexity: a packet is routed in at most $\min(n-1, 2H)$ hops where $H < n$ is the height of the tree.
  If the tree is *BFS*, at most $\min(n-1, 2D)$ hops where D is the network diameter.
- Memory Usage: at most $\delta_u + 1$ labels for node u, *i.e.*, at most $(\delta_u + 1) \times \lceil \log n \rceil$ bits

Cons.

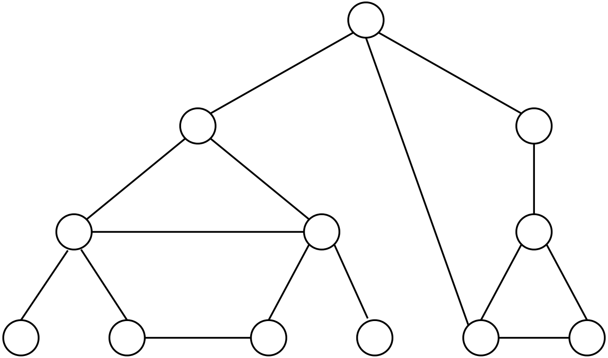
- A packet may be routed from u to v in drastically more than $\|u, v\|$ hops.
  E.g., in a ring, the two leaves are 1-hop away but any packet is routed from one to the other in $n-2$ hops.
- Only $n-1$ links are used while the network may contain $\Theta(n^2)$ links: this may lead to congestion and a single link failure partitions the network (this approach is then not robust)



This latter drawback is addressed by the interval routing

Generalization to arbitrary connected networks

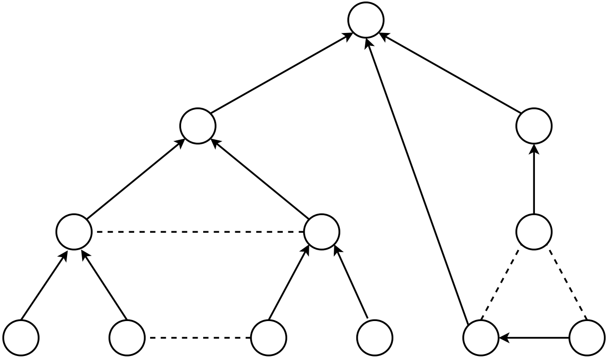
Example



A network of $n = 12$ nodes

Generalization to arbitrary connected networks

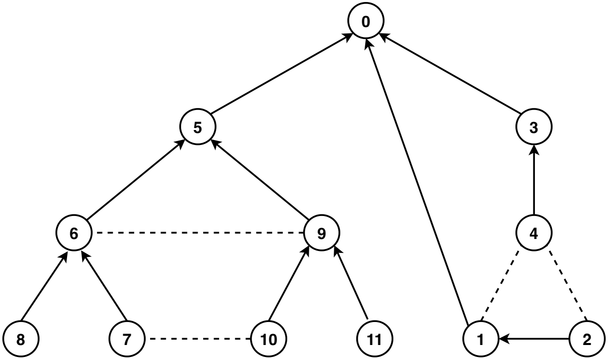
Example



A network of $n = 12$ nodes with a (BFS) spanning tree
(the uppermost node is the root)

Generalization to arbitrary connected networks

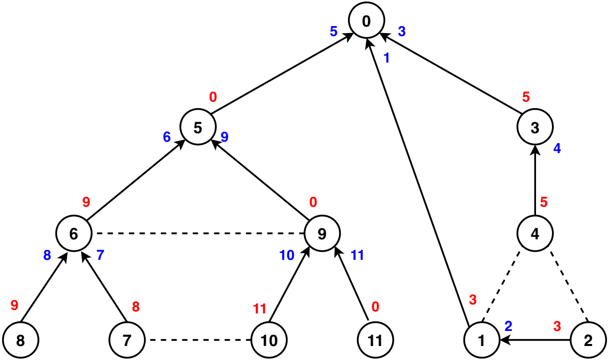
Example



A network of $n = 12$ nodes with the node labeling
(the uppermost node is the root)

Generalization to arbitrary connected networks

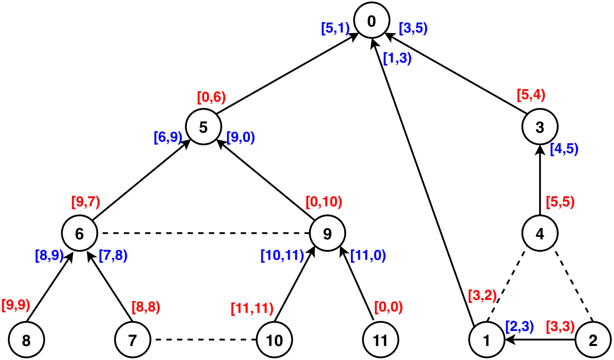
Example



A network of $n = 12$ nodes with the channel labeling

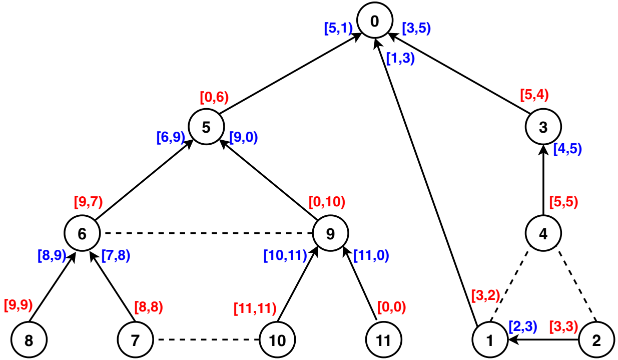
Generalization to arbitrary connected networks

Example



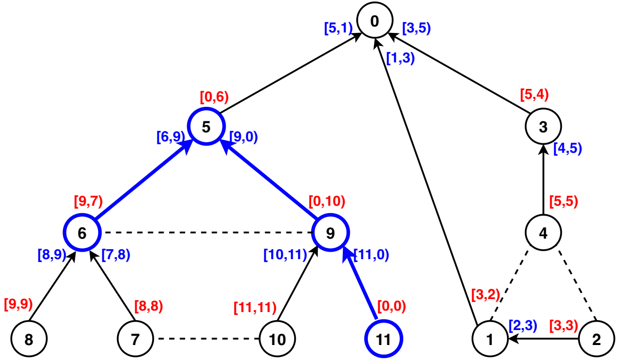
A network of $n = 12$ nodes nodes with intervals

Generalization to arbitrary connected networks
Example



A network of $n = 12$ nodes nodes with intervals

Routing from 6 to 11?



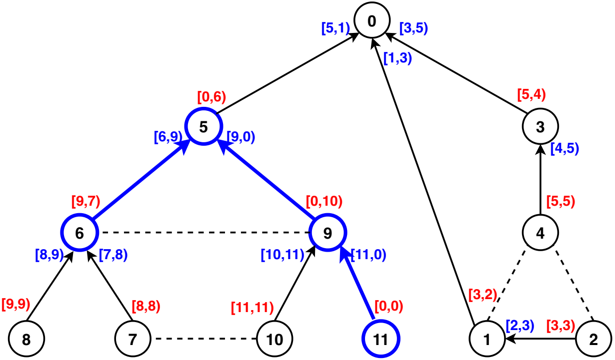
A network of $n = 12$ nodes nodes with intervals

Routing from 6 to 11:

At 6, $11 \in [9,7)$ At 5, $11 \in [9,0)$ At 9, $11 \in [11,0)$

Generalization to arbitrary connected networks

Example



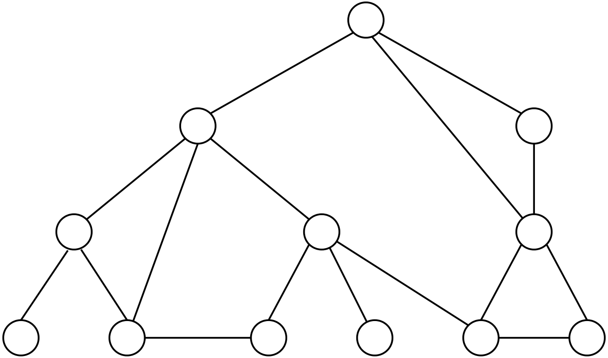
A network of $n = 12$ nodes nodes with intervals

Routing from 6 to 11:

At 6, $11 \in [9, 7)$ At 5, $11 \in [9, 0)$ At 9, $11 \in [11, 0)$

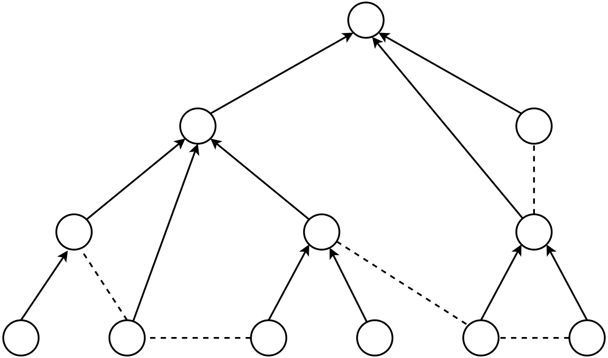
This is not the shortest path: 6,9,11 is better!

Exercise



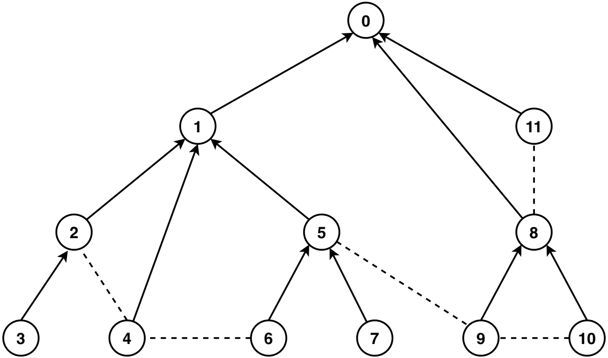
The network

Exercise



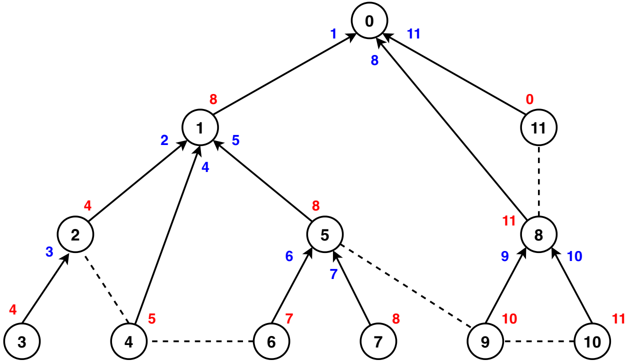
The network with a (BFS) spanning tree

Exercise



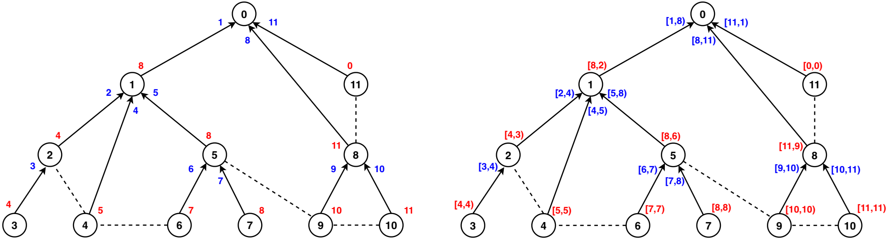
Node Labeling

Exercise



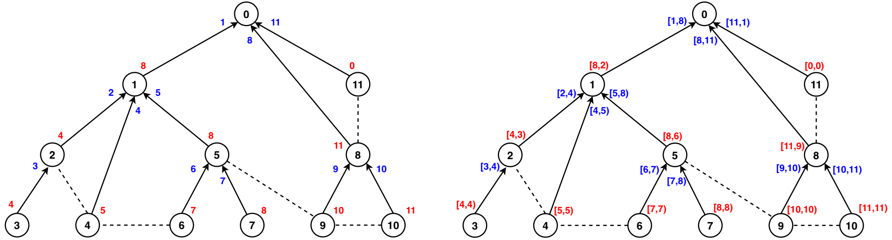
Channel Labeling

Exercise



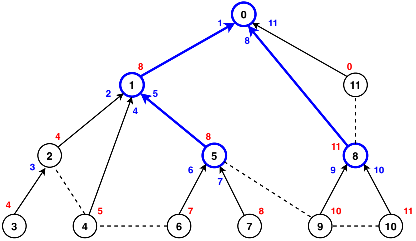
Intervals

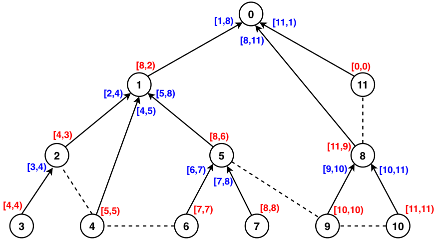
Exercise



Routing path from 5 to 8?

Exercise

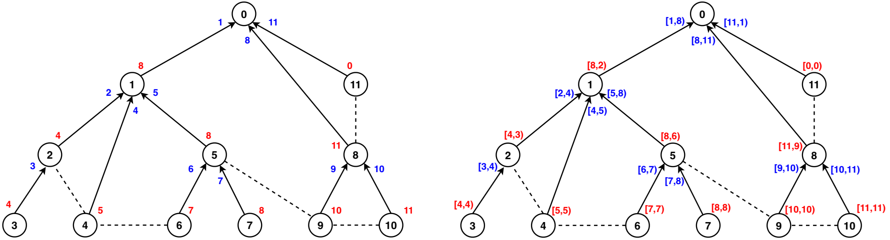




Routing path from 5 to 8:

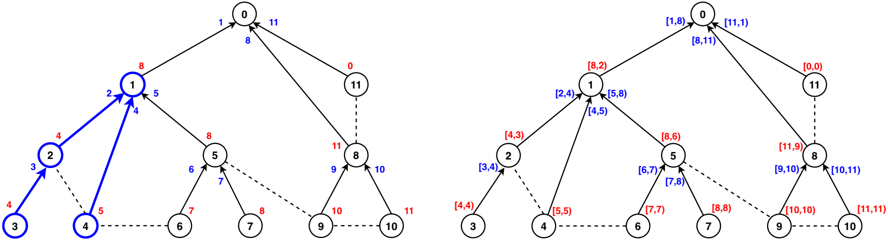
At 5, $8 \in [8,6)$ At 1, $8 \in [8,2)$ At 0, $8 \in [8,11)$

Exercise



Routing path from 4 to 3?

Exercise



Routing path from 4 to 3:

At 4, $3 \in [5, 5)$ At 1, $3 \in [2, 4)$ At 2, $3 \in [3, 4)$

Roadmap

Definition

An **interval labeling scheme (ILS)** for a network G of n nodes is

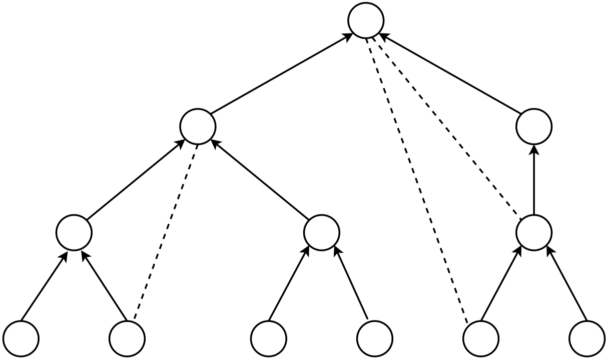
1. An assignment of different labels l_u from $\mathbb{Z}_n$ to the nodes u of G, and
2. and for each node u, an assignment of pairwise distinct labels $\alpha_i(u)$, $i = 1, \ldots, \delta_u$, from $\mathbb{Z}_n$ to all channels of u.

The **interval routing algorithm** assumes a ILS is given and forwards packets as in the tree-labeling scheme routing algorithm.

An **ILS is valid** if all packets forwarded using the interval routing algorithm eventually reach their final destination.

A valid ILS for arbitrary connected networks

Tool: Depth-First Search (DFS) spanning tree T (The root is the uppermost node)

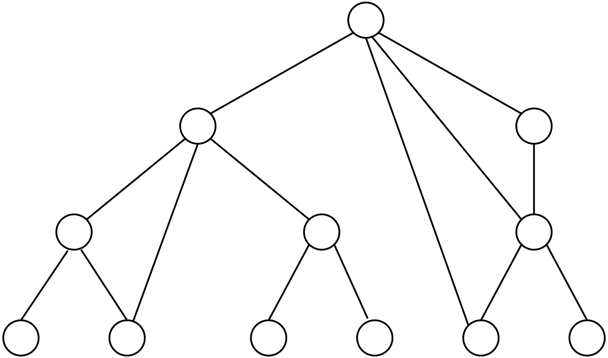


Property: **For every two neighbors u and v in G, either $u \in T(v)$, or $v \in T(u)$.**

Distributed Construction: Leader election + token circulation

($O(mn)$ messages, $O(m)$ rounds, and $O(\delta_u + B)$ bits, where B the number of bits to store an identifier, *cf.*, distributed computing courses)

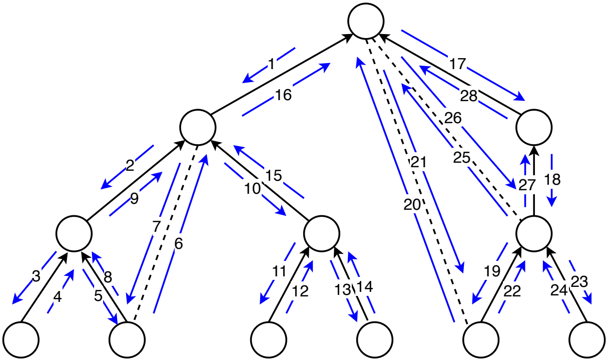
A valid ILS for arbitrary connected networks
Node labeling



A network of $n = 12$ nodes

A valid ILS for arbitrary connected networks

Node labeling



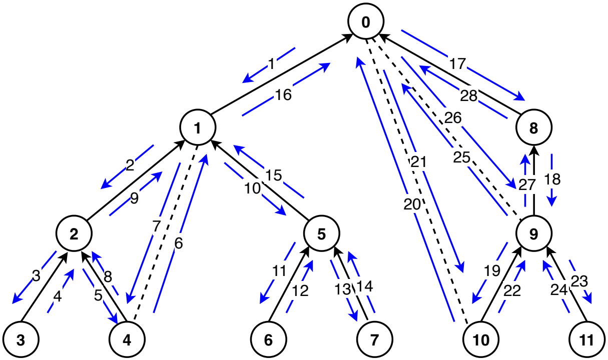
A network of $n = 12$ nodes

Preorder DFS traversal

(computed by a token circulation in $2m$ rounds)

A valid ILS for arbitrary connected networks

Node labeling



A network of $n = 12$ nodes

Preorder DFS traversal + node labeling

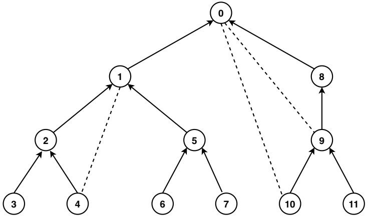
A valid ILS for arbitrary connected networks

Channel labeling

Like for the tree-labeling scheme, for every node u, for every neighbor v of u, we assign the label $\alpha_v(u)$ to the channel of u outgoing to v.

Yet, $\alpha_v(u)$ is set as follows:

1. if $\{u, v\}$ is a non-tree edge, $\alpha_v(u) = l_v$
2. If v is a child of u, $\alpha_v(u) = l_v$
3. If v is the parent of u, $\alpha_v(u) = (l_u + |T(u)|) \bmod n$ unless $l_u + |T(u)| = n$ and u has a non-tree edge to the root[3]
4. If v is the parent of u, $l_u + |T(u)| = n$, and u has a non-tree edge to the root, $\alpha_v(u) = l_v$



12 nodes

[3] In this case, the non-tree edge is labeled 0 at u by the rule 1, so assigning $\alpha_v(u)$ to $(l_u + |T(u)|) \bmod n$ would lead to two channels at u with the same label!

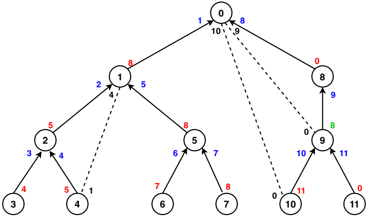
A valid ILS for arbitrary connected networks

Channel labeling

Like for the tree-labeling scheme, for every node u, for every neighbor v of u, we assign the label $\alpha_v(u)$ to the channel of u outgoing to v.

Yet, $\alpha_v(u)$ is set as follows:

1. if $\{u, v\}$ is a non-tree edge, $\alpha_v(u) = l_v$
2. If v is a child of u, $\alpha_v(u) = l_v$
3. If v is the parent of u, $\alpha_v(u) = (l_u + |T(u)|) \bmod n$ unless $l_u + |T(u)| = n$ and u has a non-tree edge to the root[3]
4. If v is the parent of u, $l_u + |T(u)| = n$, and u has a non-tree edge to the root, $\alpha_v(u) = l_v$



12 nodes
DFS spanning tree computed together with the node labeling

[3] In this case, the non-tree edge is labeled 0 at u by the rule 1, so assigning $\alpha_v(u)$ to $(l_u + |T(u)|) \bmod n$ would lead to two channels at u with the same label!

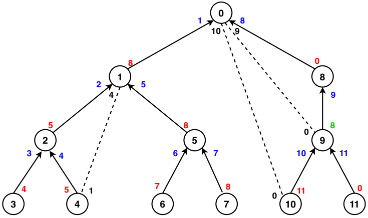
A valid ILS for arbitrary connected networks

Channel labeling

Like for the tree-labeling scheme, for every node u, for every neighbor v of u, we assign the label $\alpha_v(u)$ to the channel of u outgoing to v.

Yet, $\alpha_v(u)$ is set as follows:

1. if $\{u, v\}$ is a non-tree edge, $\alpha_v(u) = l_v$
2. If v is a child of u, $\alpha_v(u) = l_v$
3. If v is the parent of u, $\alpha_v(u) = (l_u + |T(u)|) \bmod n$ unless $l_u + |T(u)| = n$ and u has a non-tree edge to the root[3]
4. If v is the parent of u, $l_u + |T(u)| = n$, and u has a non-tree edge to the root, $\alpha_v(u) = l_v$



12 nodes
DFS spanning tree computed together with the node labeling

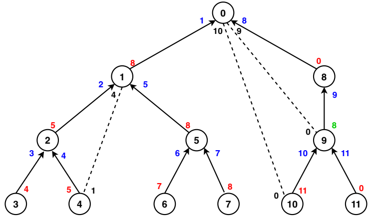
Like for the tree-labeling scheme, we let $\alpha_1(u), \ldots, \alpha_{\delta_u}(u)$ be the channel label at u sorted in increasing order according to values $\alpha_v(u)$.

[3] In this case, the non-tree edge is labeled 0 at u by the rule 1, so assigning $\alpha_v(u)$ to $(l_u + |T(u)|) \bmod n$ would lead to two channels at u with the same label!

Like for the tree-labeling scheme, for every node u, for every neighbor v of u, we assign the label $\alpha_v(u)$ to the channel of u outgoing to v.

Yet, $\alpha_v(u)$ is set as follows:

1. if $\{u, v\}$ is a non-tree edge, $\alpha_v(u) = l_v$
2. If v is a child of u, $\alpha_v(u) = l_v$
3. If v is the parent of u, $\alpha_v(u) = (l_u + |T(u)|) \bmod n$ unless $l_u + |T(u)| = n$ and u has a non-tree edge to the root[3]
4. If v is the parent of u, $l_u + |T(u)| = n$, and u has a non-tree edge to the root, $\alpha_v(u) = l_v$



12 nodes
DFS spanning tree computed together with the node labeling

Like for the tree-labeling scheme, we let $\alpha_1(u), \ldots, \alpha_{\delta_u}(u)$ be the channel label at u sorted in increasing order according to values $\alpha_v(u)$.

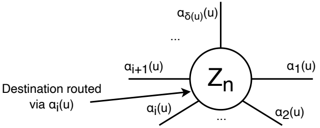
Generalization:
if G is a tree, G is labeled as with the tree-labeling scheme (Rules 1 and 4 are not used).

[3] In this case, the non-tree edge is labeled 0 at u by the rule 1, so assigning $\alpha_v(u)$ to $(l_u + |T(u)|) \bmod n$ would lead to two channels at u with the same label!

Like for the tree-labeling scheme, for every node u, for every neighbor v of u, we assign the label $\alpha_v(u)$ to the channel of u outgoing to v.

Yet, $\alpha_v(u)$ is set as follows:

1. if $\{u, v\}$ is a non-tree edge, $\alpha_v(u) = l_v$
2. If v is a child of u, $\alpha_v(u) = l_v$
3. If v is the parent of u, $\alpha_v(u) = (l_u + |T(u)|) \bmod n$ unless $l_u + |T(u)| = n$ and u has a non-tree edge to the root[3]
4. If v is the parent of u, $l_u + |T(u)| = n$, and u has a non-tree edge to the root, $\alpha_v(u) = l_v$



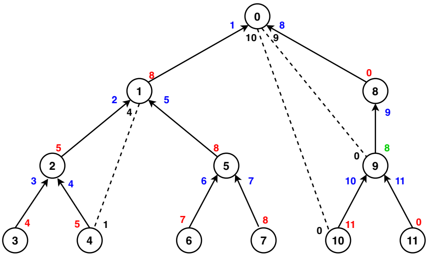
Like for the tree-labeling scheme, we let $\alpha_1(u), \ldots, \alpha_{\delta_u}(u)$ be the channel label at u sorted in increasing order according to values $\alpha_v(u)$.

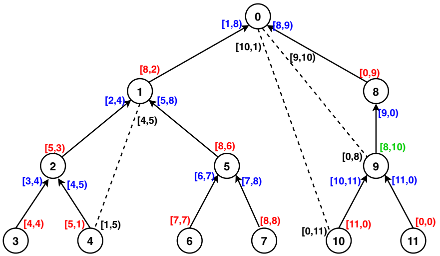
Generalization:
if G is a tree, G is labeled as with the tree-labeling scheme (Rules 1 and 4 are not used).

[3] In this case, the non-tree edge is labeled 0 at u by the rule 1, so assigning $\alpha_v(u)$ to $(l_u + |T(u)|) \bmod n$ would lead to two channels at u with the same label!

Example of Interval Routing

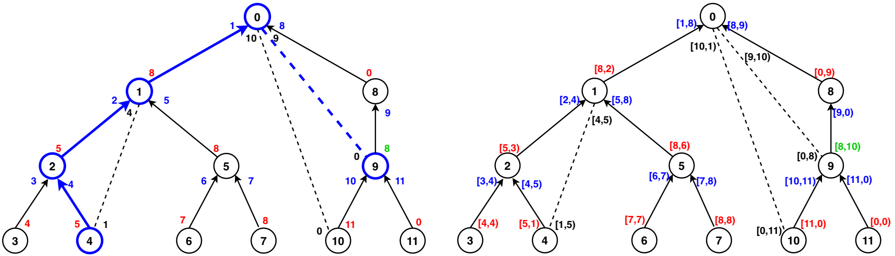
From label 4 to label 9





Example of Interval Routing

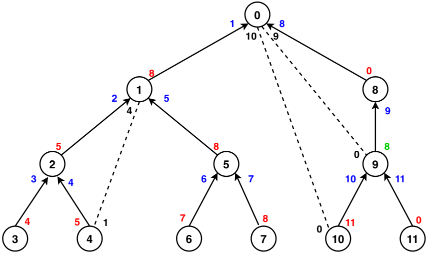
From label 4 to label 9

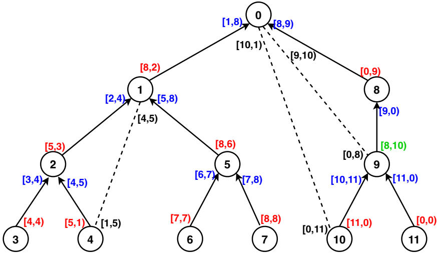


At 4, $9 \in [5, 1)$
At 2, $9 \in [5, 3)$
At 1, $9 \in [8, 2)$
At 0, $9 \in [9, 10)$

Example of Interval Routing

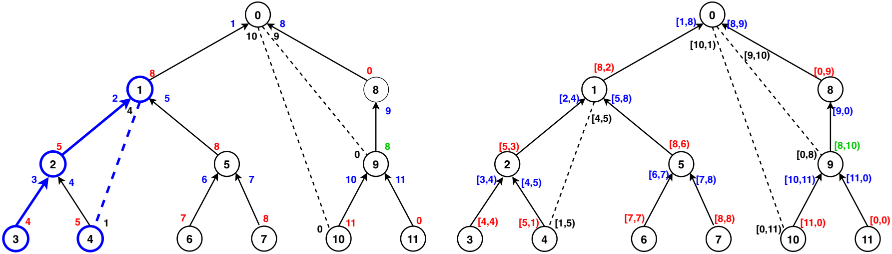
From label 4 to label 3





Example of Interval Routing

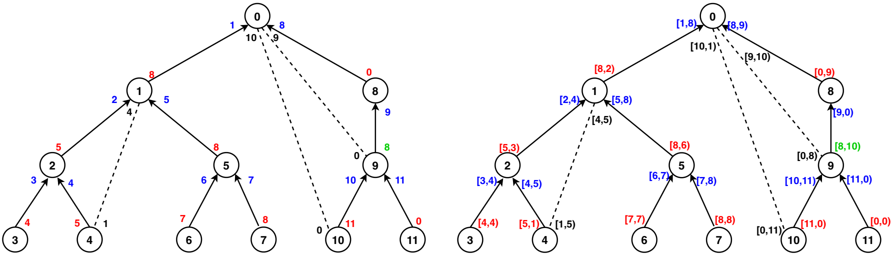
From label 4 to label 3

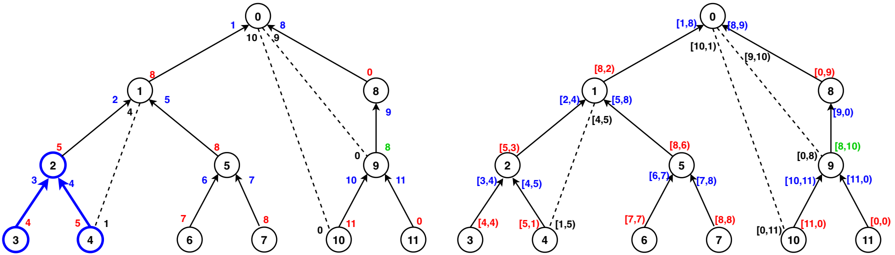


At 4, $3 \in [1, 5)$
At 1, $3 \in [2, 4)$
At 2, $3 \in [3, 4)$

Example of Interval Routing

From label 3 to label 4





At 3, $4 \in [4, 4)$

At 2, $4 \in [4, 5)$

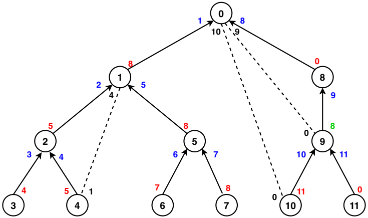
The interval routing is not necessarily symmetric!

A valid ILS for arbitrary connected networks

Properties

1. *Locally at each node, the intervals form a partition of $\mathbb{Z}_n$*

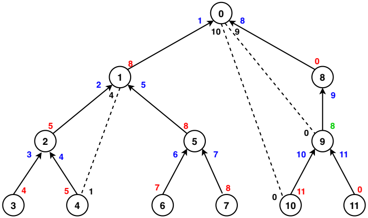
   (the proof is identical to the one for the tree-labeling scheme)



1. *Locally at each node, the intervals form a partition of* $\mathbb{Z}_n$

   (the proof is identical to the one for the tree-labeling scheme)

So, when u has a packet for $v \neq u$, u finds a unique destination w for the next hop.

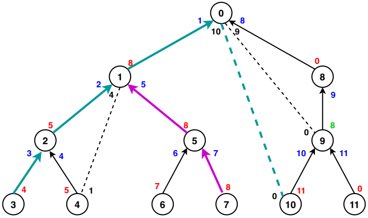


1. *Locally at each node, the intervals form a partition of $\mathbb{Z}_n$*

   (the proof is identical to the one for the tree-labeling scheme)

So, when u has a packet for $v \neq u$, u finds a unique destination w for the next hop.

2. *If $l_u > l_v$, $l_w < l_u$*



Path 10,0,1,2,3: $l_u = 10 > l_v = 3$ and $l_w = 0 < l_u = 10$

Path 7,5,1: $l_u = 7 > l_v = 1$ and $l_w = 5 < l_u = 7$

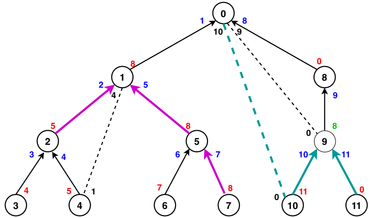
At the next hop, $l_u = 5 > l_v = 1$ and $l_w = 1 < l_u = 5$

1. *Locally at each node, the intervals form a partition of $\mathbb{Z}_n$*

   (the proof is identical to the one for the tree-labeling scheme)

So, when u has a packet for $v \neq u$, u finds a unique destination w for the next hop.

2. *If $l_u > l_v$, $l_w < l_u$*
3. *If $l_u < l_v$, $l_w \leq l_v$*



See the paths 0,10,9,11 and 2,1,5,7

A valid ILS for arbitrary connected networks

Properties

1. *Locally at each node, the intervals form a partition of $\mathbb{Z}_n$*

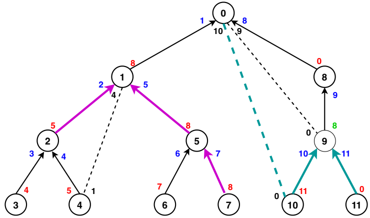
   (the proof is identical to the one for the tree-labeling scheme)

So, when u has a packet for $v \neq u$, u finds a unique destination w for the next hop.

2. *If $l_u > l_v$, $l_w < l_u$*
3. *If $l_u < l_v$, $l_w \leq l_v$*

Let $lca(x, y)$ be the label of the lowest common ancestor of x and y and $f_y(x) = (-lca(x, y), l_x)$.

4. *If $l_u < l_v$, $f_v(w) < f_v(u)$*[4]



In Path 0,10,9,11, take $l_u = 0$, $l_w = 10$, and $l_v = 11$: $f_v(w) = (-9, 10) < f_v(u) = (0, 0)$

In Path 2,1,5,7, take $l_u = 2$, $l_w = 1$, and $l_v = 7$: $f_v(w) = (-1, 1) < f_v(u) = (-1, 2)$

[4] $(a, b) < (c, d) \equiv [a < c \lor (a = c \land b < d)]$

Proof of Property 2

If $l_u > l_v$, $l_w < l_u$

Proof:

If $\alpha_w(u) \leq l_v$.

- First, w is not a proper descendent of u since otherwise $\alpha_w(u) = l_w > l_u > l_v$.
- So, w is a proper ancestor of u: $l_w < l_u$.

$\square$

Proof of Property 2

If $l_u > l_v$, $l_w < l_u$

Proof:

If $\alpha_w(u) \leq l_v$.

- First, w is not a proper descendent of u since otherwise $\alpha_w(u) = l_w > l_u > l_v$.
- So, w is a proper ancestor of u: $l_w < l_u$.

Otherwise, every label α at u satisfies $\alpha > l_v$ and $\alpha_w(u)$ is the larger label at u.

- u is not the root since $l_u > l_v \geq 0$.
- Let f be the parent of u. Since every label α at u satisfies $\alpha > l_v \geq 0$, $\alpha_f(u) = (l_u + |T(u)|) \bmod n$. Again, $\alpha_f(u) \neq 0$ since $\alpha_f(u) > l_v \geq 0$. Thus, $\alpha_f(u) = l_u + |T(u)|$ is the largest channel label at u and so $w = f$.

  Indeed, the label at u of any channel from u to any of its proper ancestor $w' \neq f$ is $l_{w'} < l_u$ and the label at u of any channel from u to any of its proper descendent w' is $l_{w'} < l_u + |T(u)|$.

As w is the parent of u, we have $l_w < l_u$. $\square$

Proof:

Two cases:

1. $v \in T(u)$.
2. $v \notin T(u)$.

Proof of Property 3

If $l_u < l_v$, $l_w \leq l_v$

Proof:

Case $v \in T(u)$.

Proof of Property 3

If $l_u < l_v$, $l_w \leq l_v$

Proof:

Case $v \in T(u)$.

Let w' be the child of u such that $v \in T(w')$. (recall that $u \neq v$)

$l_{w'} \leq l_v < l_{w'} + |T(w')|$

Proof of Property 3

If $l_u < l_v$, $l_w \leq l_v$

Proof:

Case $v \in T(u)$.

Let w' be the child of u such that $v \in T(w')$. (recall that $u \neq v$)

$l_{w'} \leq l_v < l_{w'} + |T(w')|$

Since w' is a child of u: $l_{w'} > 0$ and $\alpha_{w'}(u) = l_{w'}$, which implies that $0 < \alpha_{w'}(u) \leq l_v$.

Proof of Property 3

If $l_u < l_v$, $l_w \leq l_v$

Proof:

Case $v \in T(u)$.

Let w' be the child of u such that $v \in T(w')$. (recall that $u \neq v$)

$l_{w'} \leq l_v < l_{w'} + |T(w')|$

Since w' is a child of u: $l_{w'} > 0$ and $\alpha_{w'}(u) = l_{w'}$, which implies that $0 < \alpha_{w'}(u) \leq l_v$.

So, $\alpha_w(u)$ is the greatest label α at u such that $\alpha \leq l_v$: $0 < \alpha_{w'}(u) \leq \alpha_w(u) \leq l_v$.

Proof:

Case $v \in T(u)$.

Let w' be the child of u such that $v \in T(w')$. (recall that $u \neq v$)

$l_{w'} \leq l_v < l_{w'} + |T(w')|$

Since w' is a child of u: $l_{w'} > 0$ and $\alpha_{w'}(u) = l_{w'}$, which implies that $0 < \alpha_{w'}(u) \leq l_v$.

So, $\alpha_w(u)$ is the greatest label α at u such that $\alpha \leq l_v$: $0 < \alpha_{w'}(u) \leq \alpha_w(u) \leq l_v$.

Overall

$$0 < l_{w'} \leq \alpha_w(u) \leq l_v < l_{w'} + |T(w')|$$

Proof of Property 3

If $l_u < l_v$, $l_w \leq l_v$

Proof:

Case $v \in T(u)$.

Let w' be the child of u such that $v \in T(w')$. (recall that $u \neq v$)

$l_{w'} \leq l_v < l_{w'} + |T(w')|$

Since w' is a child of u: $l_{w'} > 0$ and $\alpha_{w'}(u) = l_{w'}$, which implies that $0 < \alpha_{w'}(u) \leq l_v$.

So, $\alpha_w(u)$ is the greatest label α at u such that $\alpha \leq l_v$: $0 < \alpha_{w'}(u) \leq \alpha_w(u) \leq l_v$.

Overall

$$0 < l_{w'} \leq \alpha_w(u) \leq l_v < l_{w'} + |T(w')|$$

Thus, w is not the parent f of u. Indeed,

- either $\alpha_f(u) = l_f < l_u < l_{w'} \leq \alpha_w(u)$,
- or $\alpha_f(u) = 0 < \alpha_w(u)$,
- or $\alpha_f(u) = l_u + |T(u)| \geq l_{w'} + |T(w')| > \alpha_w(u)$.

  ($l_u + |T(u)| \geq l_{w'} + |T(w')|$ since w' is a child of u)

Proof of Property 3

If $l_u < l_v$, $l_w \leq l_v$

Proof:

Case $v \in T(u)$.

Let w' be the child of u such that $v \in T(w')$. (recall that $u \neq v$)

$l_{w'} \leq l_v < l_{w'} + |T(w')|$

Since w' is a child of u: $l_{w'} > 0$ and $\alpha_{w'}(u) = l_{w'}$, which implies that $0 < \alpha_{w'}(u) \leq l_v$.

So, $\alpha_w(u)$ is the greatest label α at u such that $\alpha \leq l_v$: $0 < \alpha_{w'}(u) \leq \alpha_w(u) \leq l_v$.

Overall

$$0 < l_{w'} \leq \alpha_w(u) \leq l_v < l_{w'} + |T(w')|$$

Thus, w is not the parent f of u. Indeed,

- either $\alpha_f(u) = l_f < l_u < l_{w'} \leq \alpha_w(u)$,
- or $\alpha_f(u) = 0 < \alpha_w(u)$,
- or $\alpha_f(u) = l_u + |T(u)| \geq l_{w'} + |T(w')| > \alpha_w(u)$.

  ($l_u + |T(u)| \geq l_{w'} + |T(w')|$ since w' is a child of u)

Hence $l_w = \alpha_w(u) \leq l_v$.

Proof of Property 3

If $l_u < l_v$, $l_w \leq l_v$

Proof:

Case $v \notin T(u)$.

$\square$

Proof of Property 3

If $l_u < l_v$, $l_w \leq l_v$

Proof:

Case $v \notin T(u)$.

As $l_v > l_u$, we also have $l_v \geq l_u + |T(u)|$.

$\square$

Proof of Property 3

If $l_u < l_v$, $l_w \leq l_v$

Proof:

Case $v \notin T(u)$.

As $l_v > l_u$, we also have $l_v \geq l_u + |T(u)|$.

- Since $l_u + |T(u)| \leq l_v \leq n - 1$, the label of channel from u to its parent is $l_u + |T(u)|$.

$\square$

Proof of Property 3

If $l_u < l_v$, $l_w \leq l_v$

Proof:

Case $v \notin T(u)$.

As $l_v > l_u$, we also have $l_v \geq l_u + |T(u)|$.

- Since $l_u + |T(u)| \leq l_v \leq n - 1$, the label of channel from u to its parent is $l_u + |T(u)|$.
- The channel from u to one of its proper descendent w' is labeled at u with $l_{w'} < l_u + |T(u)|$.

□

Proof of Property 3

If $l_u < l_v$, $l_w \leq l_v$

Proof:

Case $v \notin T(u)$.

As $l_v > l_u$, we also have $l_v \geq l_u + |T(u)|$.

- Since $l_u + |T(u)| \leq l_v \leq n - 1$, the label of channel from u to its parent is $l_u + |T(u)|$.
- The channel from u to one of its proper descendent w' is labeled at u with $l_{w'} < l_u + |T(u)|$.
- The channel from u to one of its non-parent proper ancestor w' is labeled at u with $l_{w'} < l_u < l_u + |T(u)|$.

$\square$

Proof of Property 3

If $l_u < l_v$, $l_w \leq l_v$

Proof:

Case $v \notin T(u)$.

As $l_v > l_u$, we also have $l_v \geq l_u + |T(u)|$.

- Since $l_u + |T(u)| \leq l_v \leq n - 1$, the label of channel from u to its parent is $l_u + |T(u)|$.
- The channel from u to one of its proper descendent w' is labeled at u with $l_{w'} < l_u + |T(u)|$.
- The channel from u to one of its non-parent proper ancestor w' is labeled at u with $l_{w'} < l_u < l_u + |T(u)|$.

So, w is the parent of u and $l_w < l_u < l_v$.

$\square$

Proof of Property 4

If $l_u < l_v$, $f_v(w) < f_v(u)$

Proof:

If $v \in T(u)$, $lca(u, v) = l_u$.

Let w' the child of u such that $v \in T(w')$.

As in the proof of Property 3, we have $l_{w'} \leq l_w < l_{w'} + |T(w')|$.

Thus, $w \in T(w')$ and so $lca(w, v) \geq l_{w'} > l_u = lca(u, v)$.

Hence, $f_v(w) < f_v(u)$.

$\square$

Proof of Property 4

If $l_u < l_v$, $f_v(w) < f_v(u)$

Proof:

If $v \in T(u)$, $lca(u, v) = l_u$.

Let w' the child of u such that $v \in T(w')$.

As in the proof of Property 3, we have $l_{w'} \leq l_w < l_{w'} + |T(w')|$.

Thus, $w \in T(w')$ and so $lca(w, v) \geq l_{w'} > l_u = lca(u, v)$.

Hence, $f_v(w) < f_v(u)$.

Otherwise $v \notin T(u)$ and $l_v \geq l_u + |T(u)|$ since $l_v > l_u$.

As in the proof of Property 3, w is the parent of u and so $l_w < l_u$.

Now, $v \notin T(u)$ implies $lca(w, v) = lca(u, v)$. Hence, $f_v(w) < f_v(u)$. $\square$

Correctness:

- By Property 2 (if $l_u > l_v$, $l_w < l_u$), after a finite number of hops, the packet reaches a node u such that $l_u \leq l_v$
- By Property 3 (if $l_u < l_v$, $l_w \leq l_v$), the property $l_u \leq l_v$ is invariant
- By Property 4 (if $l_u < l_v$, $f_v(w) < f_v(u)$), the packet is deliver to its destination within a finite number of hops after the property $l_u \leq l_v$ becomes true

Correctness:

- By Property 2 (if $l_u > l_v$, $l_w < l_u$), after a finite number of hops, the packet reaches a node u such that $l_u \leq l_v$
- By Property 3 (if $l_u < l_v$, $l_w \leq l_v$), the property $l_u \leq l_v$ is invariant
- By Property 4 (if $l_u < l_v$, $f_v(w) < f_v(u)$), the packet is deliver to its destination within a finite number of hops after the property $l_u \leq l_v$ becomes true

Complexity: At most $n - 1$ hops

The correctness implies the absence of cycles:

- Correctness implies the routing path is finite
- The algorithm is memoryless and deterministic, so the routing path is elementary

Elementary $\Rightarrow$ length of the routing path $\leq n - 1$

A valid ILS for arbitrary connected networks

Pros and Cons

Pros:

1. Load-balancing

   (every link is used by at least one route)

2. Memory Usage: $\delta_u + 1$ labels for node u,
   i.e., $(\delta_u + 1) \times \lceil \log n \rceil$ bits

3. More robust than tree-labeling scheme

A valid ILS for arbitrary connected networks

Pros and Cons

Pros:

1. Load-balancing

   (every link is used by at least one route)

2. Memory Usage: $\delta_u + 1$ labels for node u, *i.e.*, $(\delta_u + 1) \times \lceil \log n \rceil$ bits

3. More robust than tree-labeling scheme

Cons:

1. Robustness: in case of topological changes, the DFS spanning tree may have to be totally recomputed.

   A more robust solution: prefix routing (presented in the next section)

2. Efficiency: in arbitrary connected networks, the route length can be greater than the distance between the source and the destination.

   In the first example: nodes of labels 4 and 2 are neighbors but the route from 4 to 2 go through the node of label 1!

   Lower bound: in the worst case the interval routing algorithm chooses a route of length at least $\frac{3}{2}$ of the network diameter [2]

A valid ILS for arbitrary connected networks

Pros and Cons

Pros:

1. Load-balancing

   (every link is used by at least one route)

2. Memory Usage: $\delta_u + 1$ labels for node u, *i.e.*, $(\delta_u + 1) \times \lceil \log n \rceil$ bits

3. More robust than tree-labeling scheme

Cons:

1. Robustness: in case of topological changes, the DFS spanning tree may have to be totally recomputed.

   A more robust solution: prefix routing (presented in the next section)

2. Efficiency: in arbitrary connected networks, the route length can be greater than the distance between the source and the destination.

   In the first example: nodes of labels 4 and 2 are neighbors but the route from 4 to 2 go through the node of label 1!

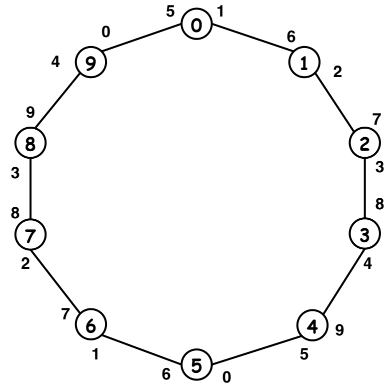
   Lower bound: in the worst case the interval routing algorithm chooses a route of length at least $\frac{3}{2}$ of the network diameter [2]

However, hop-optimal in many topologies with regular structure, *e.g.*, rings and $L \times L$-grids

A hop-optimal valid ILS for rings

Labeling:

1. Nodes are labeled from 0 to $n-1$ in clockwise order
2. For each node labeled i, the clockwise channel is labeled $(i+1) \bmod n$
3. For each node labeled i, the anticlockwise channel is labeled $(i + \lceil \frac{n}{2} \rceil) \bmod n$



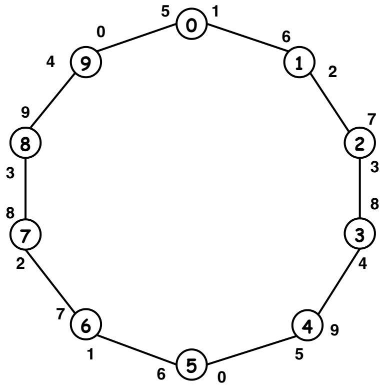
A hop-optimal valid ILS for rings

Labeling:

1. Nodes are labeled from 0 to $n-1$ in clockwise order
2. For each node labeled i, the clockwise channel is labeled $(i+1) \bmod n$
3. For each node labeled i, the anticlockwise channel is labeled $(i+\lceil \frac{n}{2} \rceil) \bmod n$

Routing:

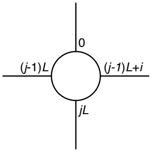
1. Packets for nodes $i+1, \ldots, (i+\lceil \frac{n}{2} \rceil)-1$ routed *via* the clockwise channel
2. Packets for nodes $(i+\lceil \frac{n}{2} \rceil), \ldots, i-1$ routed *via* the anticlockwise channel

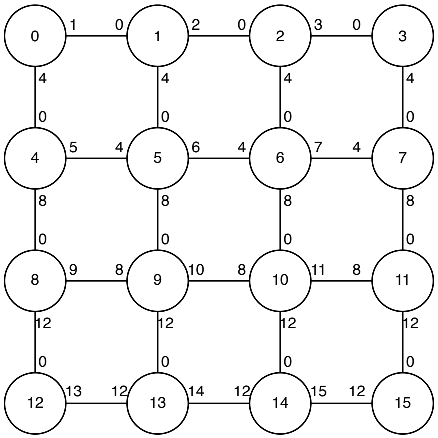


A hop-optimal valid ILS for $(L \times L)$-grids $(n = L \times L)$

Labeling:

1. The node at the ith column and jth row is labeled $(j-1)L + (i-1)$
2. The channels of the node at the ith column and the jth row are labeled as follows

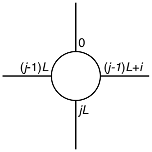


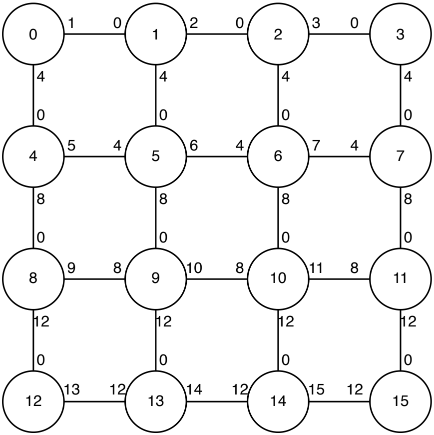


A hop-optimal valid ILS for $(L \times L)$-grids $(n = L \times L)$

Labeling:

1. The node at the ith column and jth row is labeled $(j-1)L + (i-1)$
2. The channels of the node at the ith column and the jth row are labeled as follows

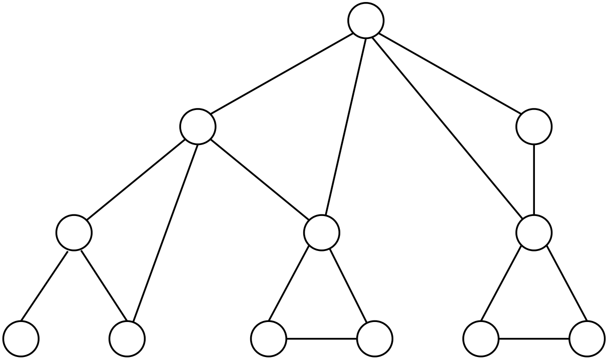




Routing:

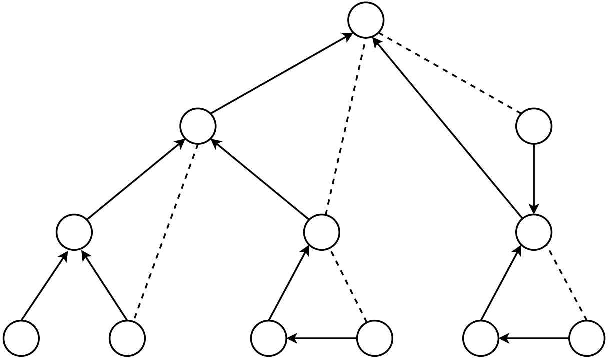
1. If v is in a row higher that u, u sends the packet up
2. If v is in a row lower that u, u sends the packet down
3. If v is in the same row as u but to the left, u sends the packet to the left
4. If v is in the same row as u but to the right, u sends the packet to the right

Exercise (1/2)



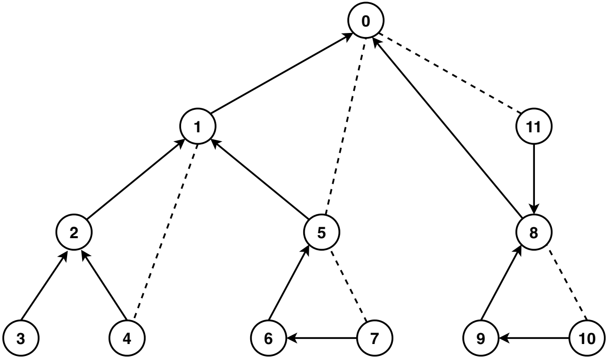
The network

Exercise (1/2)



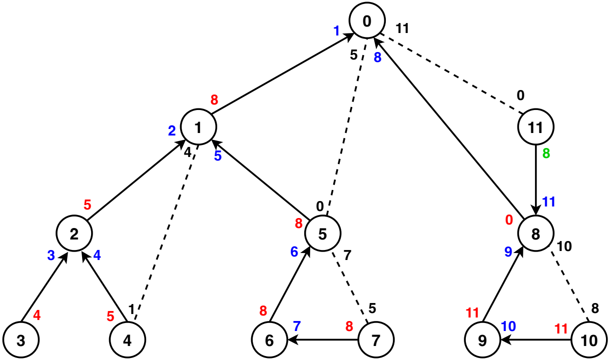
The network with a DFS spanning tree

Exercise (1/2)



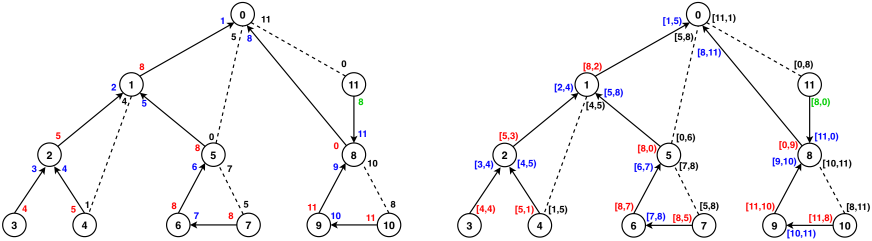
Node Labeling

Exercise (1/2)

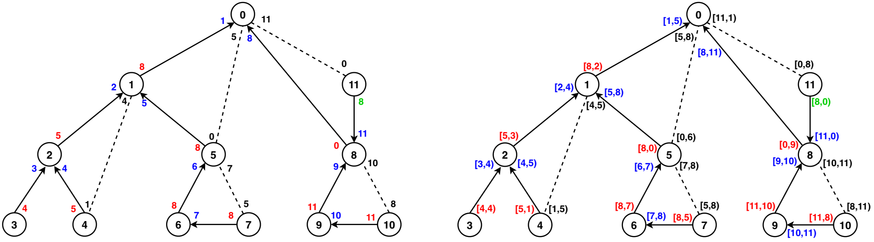


Channel Labeling

Exercise (1/2)

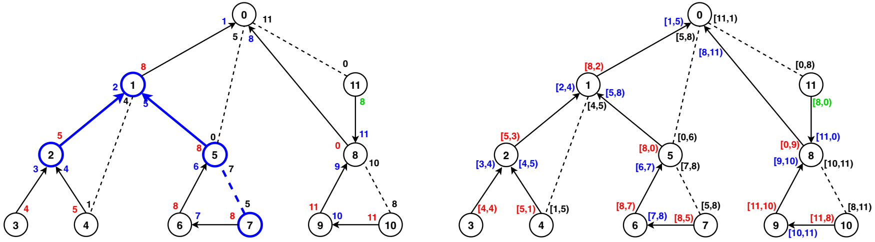


Intervals



Routing path from 2 to 7?

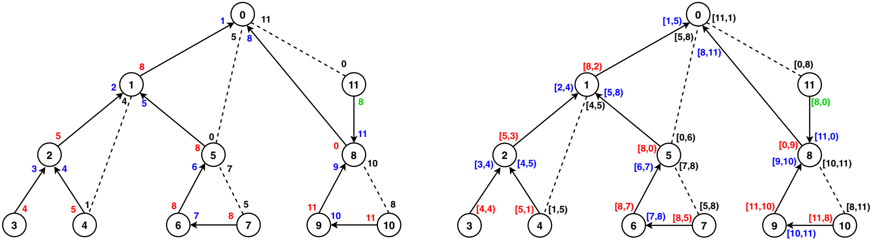
Exercise (1/2)



Routing path from 2 to 7:

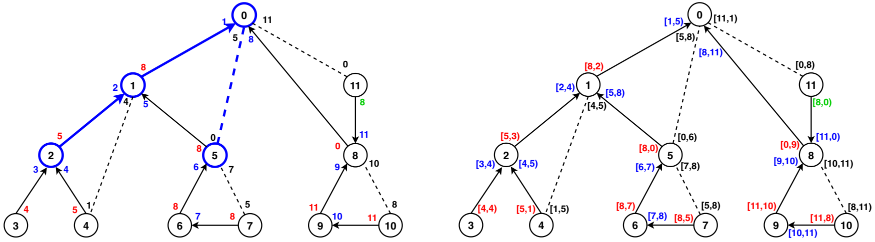
At 2, $7 \in [5, 3)$ At 1, $7 \in [5, 8)$ At 5, $7 \in [7, 8)$

It is the shortest path!



Routing path from 5 to 2?

Exercise (1/2)

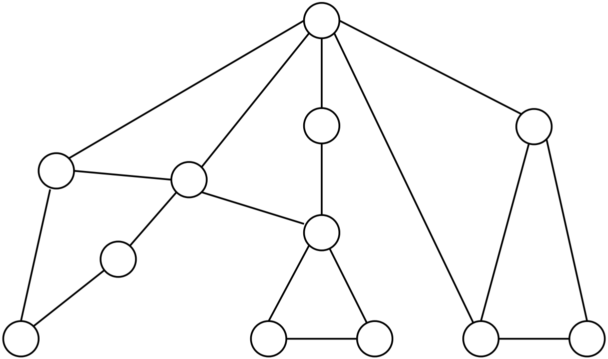


Routing path from 5 to 2:

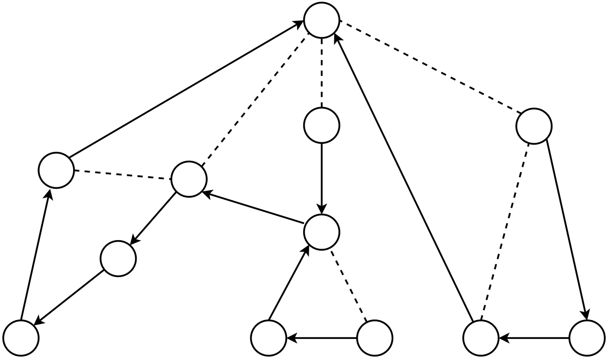
At 5, $2 \in [0, 6)$ At 0, $2 \in [1, 5)$ At 1, $2 \in [2, 4)$

It is NOT the shortest path!

Exercise (2/2)

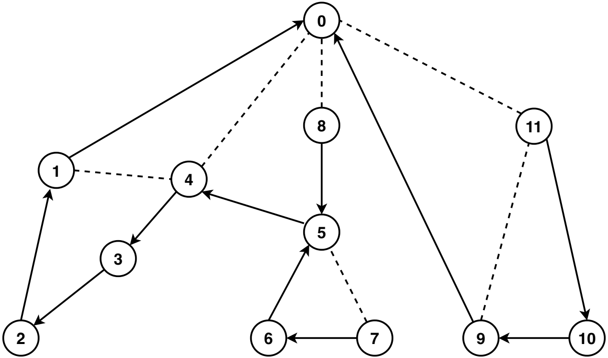


The network



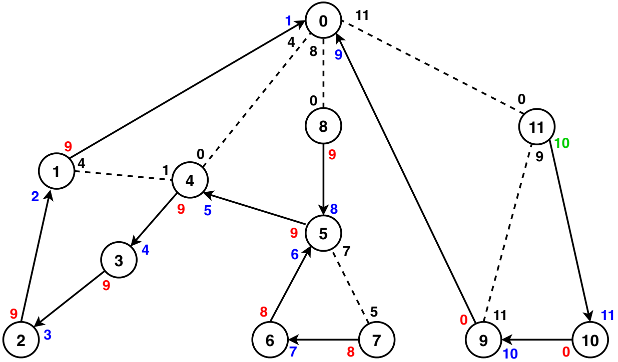
The network with a DFS spanning tree

Exercise (2/2)



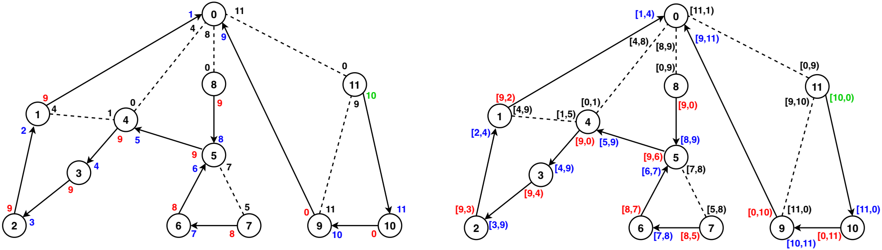
Node Labeling

Exercise (2/2)



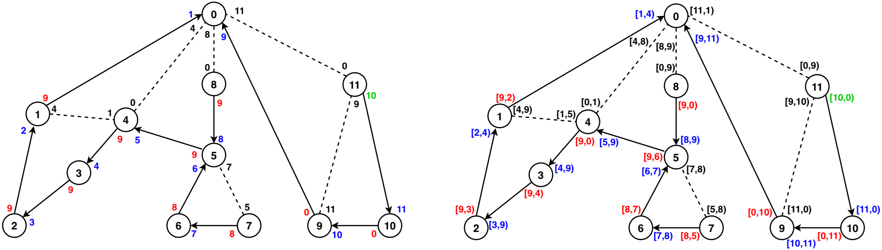
Channel Labeling

Exercise (2/2)



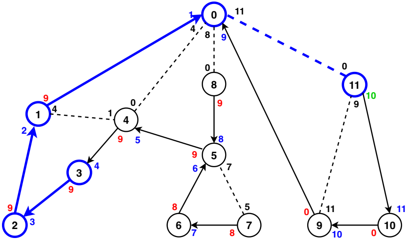
Intervals

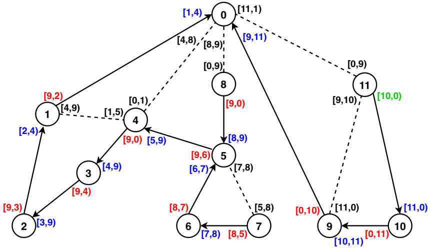
Exercise (2/2)



Routing path from 3 to 11?

Exercise (2/2)



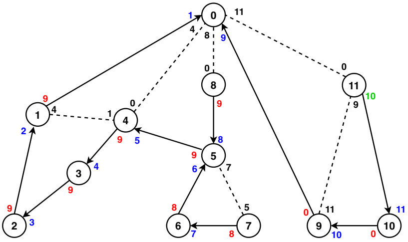


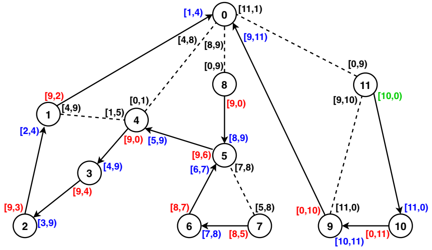
Routing path from 3 to 11:

At 3, $11 \in [9,4)$ At 2, $11 \in [9,3)$ At 1, $11 \in [9,2)$ At 0, $11 \in [11,1)$

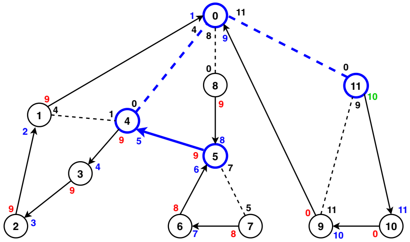
It is NOT the shortest path!

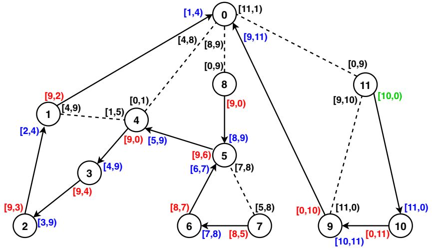
Exercise (2/2)





Routing path from 11 to 5?





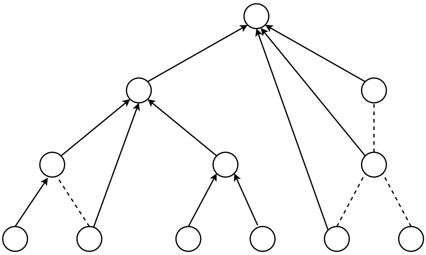
Routing path from 11 to 5:

At 11, $5 \in [0, 9)$ At 0, $5 \in [4, 8)$ At 4, $5 \in [5, 9)$

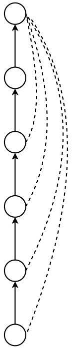
It is the shortest path!

Rule 4: how many times?

Rule 4: how many times?



0 times



$n-2$ times

Roadmap

Idea [1]

Based on an **arbitrary** spanning tree T of the network to increase robustness:

1. If a link is added between two nodes, the spanning tree remains a spanning tree and the new link is a non-tree edge

2. If a new node is added together with new links connecting it to existing nodes, the spanning tree is extended using one of the links,[5] the other are non-tree edges

[5] *e.g.*, the one with the extremity that is closest to the root

Idea [1]

Based on an **arbitrary** spanning tree T of the network to increase robustness:

1. If a link is added between two nodes, the spanning tree remains a spanning tree and the new link is a non-tree edge

2. If a new node is added together with new links connecting it to existing nodes, the spanning tree is extended using one of the links,[5] the other are non-tree edges

Efficiency can be improved starting from a BFS spanning tree

[5] *e.g.*, the one with the extremity that is closest to the root

Principle

1. Node and channels labels: strings on some alphabet Σ (*e.g.*, port numbers)
2. Σ^*: set of all strings over Σ
3. ϵ: the empty string
4. $\alpha \lhd \beta$: α is a prefix of β

Principle

1. Node and channels labels: strings on some alphabet Σ (*e.g.*, port numbers)
2. Σ^*: set of all strings over Σ
3. ϵ: the empty string
4. $\alpha \lhd \beta$: α is a prefix of β

Packet forwarding

Consider all channels whose label is prefix of the destination label and select the longest one.

Principle

1. Node and channels labels: strings on some alphabet Σ (*e.g.*, port numbers)
2. Σ^*: set of all strings over Σ
3. ϵ: the empty string
4. $\alpha \lhd \beta$: α is a prefix of β

Packet forwarding

Consider all channels whose label is prefix of the destination label and select the longest one.

Example: If the destination label is aabbc and the current node has channel labels: aabb, abba, aab, aabc, aa.

aabb, aab, aa are prefix and the channel labeled **aabb** is selected for the next hop.

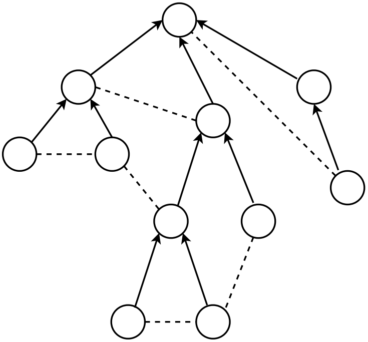
Routing Algorithm

Given a packet p with destination label d at node u.

if $d = l_u$ **then**
    `deliver` p
else
    let $\alpha_i(u) :=$ the longest channel label such that $\alpha_i(u) \lhd d$
    send p via the channel labeled with $\alpha_i(u)$
end if

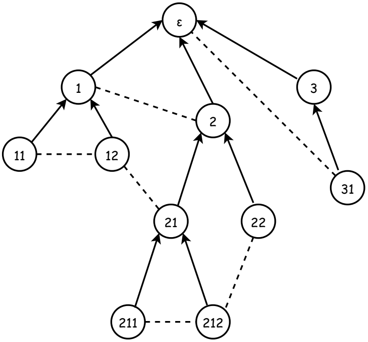
Node Labeling

1. If u is the root, u is labeled with $l_u = \epsilon$
2. If w is the child of u, l_w extends l_u by one letter: if $u_1, \ldots, u_k$ are the children of u, then $l_{u_i} = l_u.a_i$, where $a_1, \ldots, a_k$ are k distinct letters from Σ



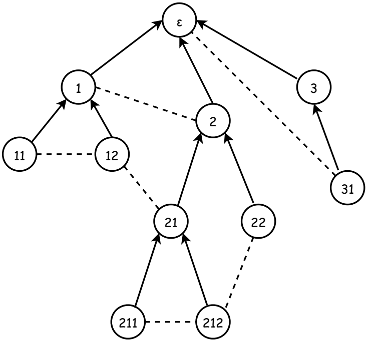
Node Labeling

1. If u is the root, u is labeled with $l_u = \epsilon$
2. If w is the child of u, l_w extends l_u by one letter: if u_1, ..., u_k are the children of u, then $l_{u_i} = l_u.a_i$, where a_1, ..., a_k are k distinct letters from Σ



Node Labeling

1. If u is the root, u is labeled with $l_u = \epsilon$
2. If w is the child of u, l_w extends l_u by one letter: if $u_1, \ldots, u_k$ are the children of u, then $l_{u_i} = l_u.a_i$, where $a_1, \ldots, a_k$ are k distinct letters from Σ



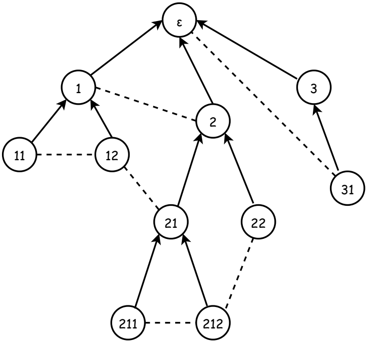
Remark: It may be distributedly computed using (BFS) spanning tree construction (with initialization and termination detection at the root) ($O(H)$ rounds, $O(n.m)$ messages of $O(H.\log|\Sigma|)$ bits, and $O(\log\Delta + H.\log|\Sigma|)$ bits per node)

(H is the height of the tree)

(If we use port numbers as alphabet, $|\Sigma| = O(\Delta)$ where Δ is the degree of the network)

Channel Labeling

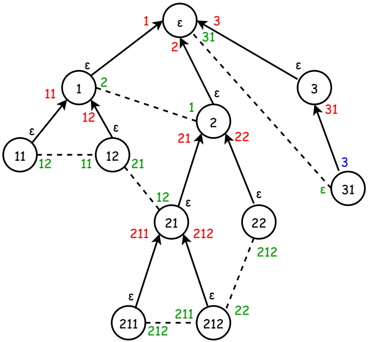
1. If $\{u, v\}$ is a non-tree edge, $\alpha_v(u) = l_v$
2. If v is a child of u, $\alpha_v(u) = l_v$
3. If v is the parent of u and u has no non-tree edge to the root,[6] $\alpha_v(u) = \epsilon$
4. If v is the parent of u and u has a non-tree edge to the root, $\alpha_v(u) = l_v$



[6] Otherwise, the non-tree edge is labeled 0 at u by the rule 1, so assigning $\alpha_v(u)$ to ϵ would lead to two channels at u with the same label!

Channel Labeling

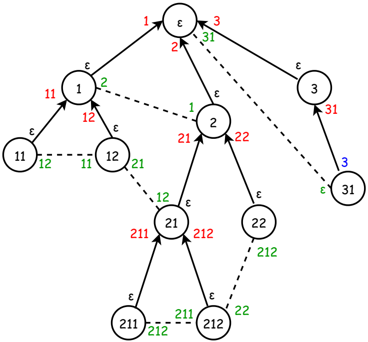
1. If $\{u, v\}$ is a non-tree edge, $\alpha_v(u) = l_v$
2. If v is a child of u, $\alpha_v(u) = l_v$
3. If v is the parent of u and u has no non-tree edge to the root,[6] $\alpha_v(u) = \epsilon$
4. If v is the parent of u and u has a non-tree edge to the root, $\alpha_v(u) = l_v$



[6] Otherwise, the non-tree edge is labeled 0 at u by the rule 1, so assigning $\alpha_v(u)$ to ϵ would lead to two channels at u with the same label!

1. If $\{u, v\}$ is a non-tree edge, $\alpha_v(u) = l_v$
2. If v is a child of u, $\alpha_v(u) = l_v$
3. If v is the parent of u and u has no non-tree edge to the root,[6] $\alpha_v(u) = \epsilon$
4. If v is the parent of u and u has a non-tree edge to the root, $\alpha_v(u) = l_v$

Property: v is an ancestor of u if and only if $l_v \lhd l_u$



[6] Otherwise, the non-tree edge is labeled 0 at u by the rule 1, so assigning $\alpha_v(u)$ to ϵ would lead to two channels at u with the same label!

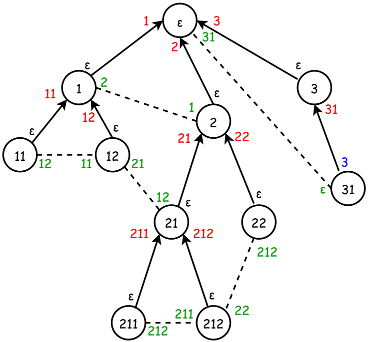
Channel Labeling

1. If $\{u, v\}$ is a non-tree edge, $\alpha_v(u) = l_v$
2. If v is a child of u, $\alpha_v(u) = l_v$
3. If v is the parent of u and u has no non-tree edge to the root,[6] $\alpha_v(u) = \epsilon$
4. If v is the parent of u and u has a non-tree edge to the root, $\alpha_v(u) = l_v$

Property: v is an ancestor of u if and only if $l_v \lhd l_u$

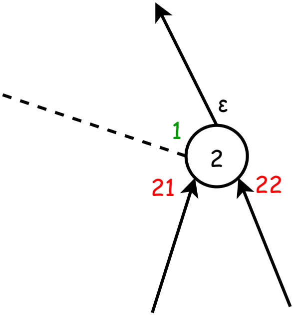
Remark: It may be distributedly computed using PIF in the tree ($O(H)$ rounds, $O(n)$ messages, and $O(\Delta.H.\log|\Sigma|)$ bits per node

(If we use port numbers as alphabet, $|\Sigma| = O(\Delta)$) and so we have $O(\Delta.H.\log\Delta)$ bits per node

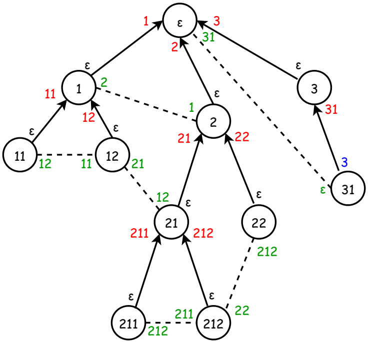


[6] Otherwise, the non-tree edge is labeled 0 at u by the rule 1, so assigning $\alpha_v(u)$ to ϵ would lead to two channels at u with the same label!

Local View

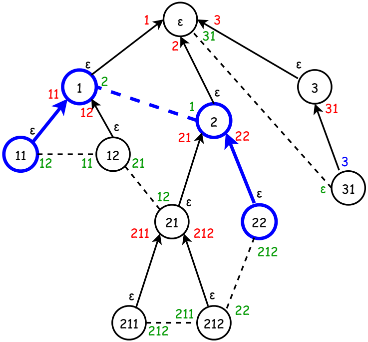


Example of Prefix Routing



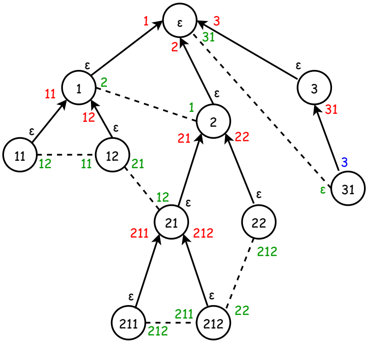
Routing from 11 to 22?

Example of Prefix Routing



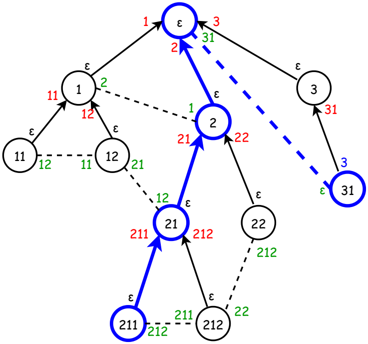
Routing from 11 to 22

Example of Prefix Routing



Routing from 211 to 31?

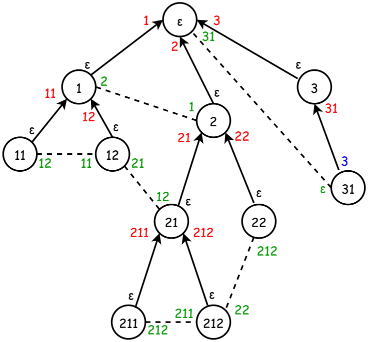
Example of Prefix Routing



Routing from 211 to 31

Properties

1. *For all nodes u and v such that $u \neq v$, there is a channel at u labeled with a prefix of l_v*



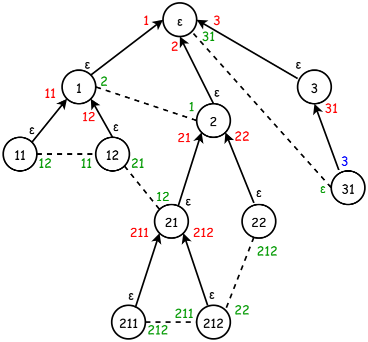
Properties

1. *For all nodes u and v such that $u \neq v$, there is a channel at u labeled with a prefix of l_v*

So, when u has a packet for $v \neq u$, u uniquely determines a destination w for the next hop

(channel labels are unique at u, and so is the one that is the longest prefix of l_v)

2. *If $u \in T(v)$, w is an ancestor of u*
   See, *e.g.*, Paths $211, 21, 2$ and $31, \epsilon$



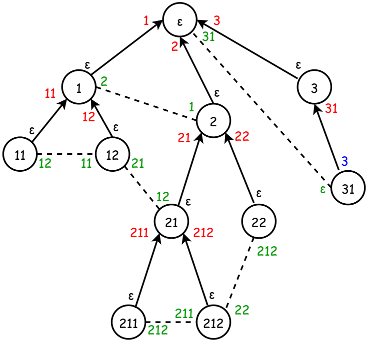
1. *For all nodes u and v such that $u \neq v$, there is a channel at u labeled with a prefix of l_v*

So, when u has a packet for $v \neq u$, u uniquely determines a destination w for the next hop

(channel labels are unique at u, and so is the one that is the longest prefix of l_v)

2. *If $u \in T(v)$, w is an ancestor of u*
   See, *e.g.*, Paths $211, 21, 2$ and $31, \epsilon$

3. *If u is an ancestor of v, w is an ancestor of v closer to v than u*
   See, *e.g.*, Path $2, 21, 212$



Properties

1. *For all nodes u and v such that $u \neq v$, there is a channel at u labeled with a prefix of l_v*

So, when u has a packet for $v \neq u$, u uniquely determines a destination w for the next hop

(channel labels are unique at u, and so is the one that is the longest prefix of l_v)

2. *If $u \in T(v)$, w is an ancestor of u*

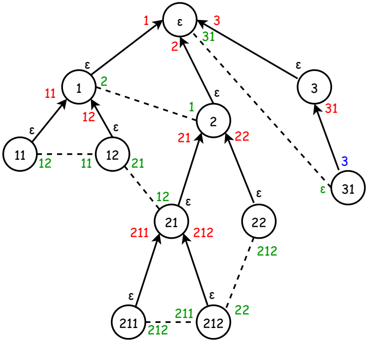
   See, *e.g.*, Paths $211, 21, 2$ and $31, \epsilon$

3. *If u is an ancestor of v, w is an ancestor of v closer to v than u*

   See, *e.g.*, Path $2, 21, 212$

4. *If $u \notin T(v)$, w is an ancestor of v or w is the parent of u*

   See, *e.g.*, Paths $11, 1, 2$ and $12, 1, 2, 22$



Proof of Property 1

For all nodes u and v such that $u \neq v$, there is a channel at u labeled with a prefix of l_v

If u is not the root, u has a channel ϵ which is a prefix of l_v.

Otherwise, u is the root, $v \in T(u)$, and has a child w such that $v \in T(w)$. By construction, $\alpha_w(u) = l_w \lhd l_v$. $\square$

Proof of Property 2

If $u \in T(v)$, w is an ancestor of u

If $\alpha_w(u) = \epsilon$, w is an ancestor of u
(w is either the parent of u or the root).

Otherwise, $l_w = \alpha_w(u) \lhd l_v \lhd l_u$ and so w is an ancestor of u.

$\square$

Proof of Property 3

If u is an ancestor of v, w is an ancestor of v closer to v than u

Let w' be the child of u such that $v \in T(w')$. $\alpha_{w'}(u) = l_{w'}$ is a non-empty prefix of l_v. As $\alpha_w(u)$ is the longest prefix of l_v at u, we have

$$l_u \lhd \alpha_{w'}(u) = l_{w'} \lhd \alpha_w(u) = l_w \lhd l_v$$

That is, w is an ancestor of v below u.

$\square$

Proof of Property 4

If $u \notin T(v)$, w is an ancestor of v or w is the parent of u

If $\alpha_w(u) = \epsilon$, w is the parent of u or the root. Now, the root is an ancestor of v.

Otherwise, $\alpha_w(u) = l_w \lhd l_v$: w is an ancestor of v. $\square$

Correctness & Complexity

Assume u sends a packet to v

Correctness & Complexity

Assume u sends a packet to v

1. If u is an ancestor of v, v is reached within at most H hops by Property 3 (if u is an ancestor of v, w is an ancestor of v closer to v than u)

Correctness & Complexity

Assume u sends a packet to v

1. If u is an ancestor of v, v is reached within at most H hops by Property 3 (if u is an ancestor of v, w is an ancestor of v closer to v than u)
2. If u is a descendent of v, an ancestor of v is reached within at most H hops by Property 2 (if $u \in T(v)$, w is an ancestor of u); then v is reached within at most H hops by Property 3 (if u is an ancestor of v, w is an ancestor of v closer to v than u)

Correctness & Complexity

Assume u sends a packet to v

1. If u is an ancestor of v, v is reached within at most H hops by Property 3 (if u is an ancestor of v, w is an ancestor of v closer to v than u)
2. If u is a descendent of v, an ancestor of v is reached within at most H hops by Property 2 (if $u \in T(v)$, w is an ancestor of u); then v is reached within at most H hops by Property 3 (if u is an ancestor of v, w is an ancestor of v closer to v than u)
3. If u is neither an ancestor nor a descendent of v, the packet reaches an ancestor of v in at most H hops by Property 4 (if $u \notin T(v)$, w is an ancestor of v or w is the parent of u)[7] and then at most H additional hops are required to reach v by Property 3 (if u is an ancestor of v, w is an ancestor of v closer to v than u)

[7] In case w is the parent of u, we have $w \notin T(v)$

Correctness & Complexity

Assume u sends a packet to v

1. If u is an ancestor of v, v is reached within at most H hops by Property 3 (if u is an ancestor of v, w is an ancestor of v closer to v than u)
2. If u is a descendent of v, an ancestor of v is reached within at most H hops by Property 2 (if $u \in T(v)$, w is an ancestor of u); then v is reached within at most H hops by Property 3 (if u is an ancestor of v, w is an ancestor of v closer to v than u)
3. If u is neither an ancestor nor a descendent of v, the packet reaches an ancestor of v in at most H hops by Property 4 (if $u \notin T(v)$, w is an ancestor of v or w is the parent of u)[7] and then at most H additional hops are required to reach v by Property 3 (if u is an ancestor of v, w is an ancestor of v closer to v than u)

Overall, a packet for v initiated at u reaches v within at most $2H$ hops.

[7] In case w is the parent of u, we have $w \notin T(v)$

Pros. and Cons.

Pros.

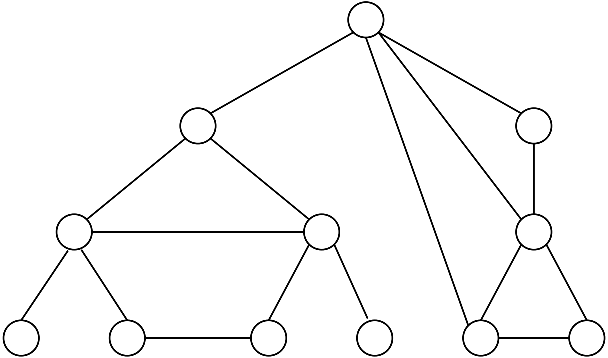
- Correct
- Robust
- Load-balancing (every link is used by at least one route)

Cons.

- Memory usage: $O(\Delta.H.\log|\Sigma|)$ bits per node

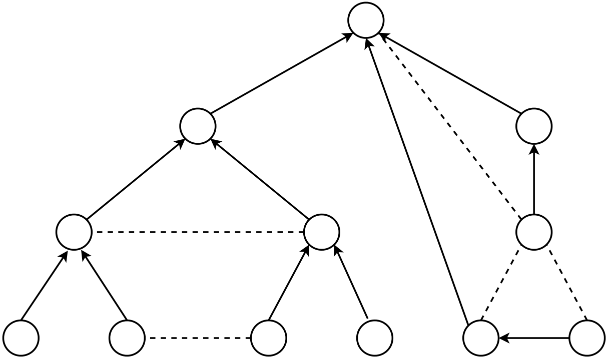
  ($O(\Delta.H.\log\Delta)$ bits per node if we use port numbers)
- Efficiency

Exercise (1/2)



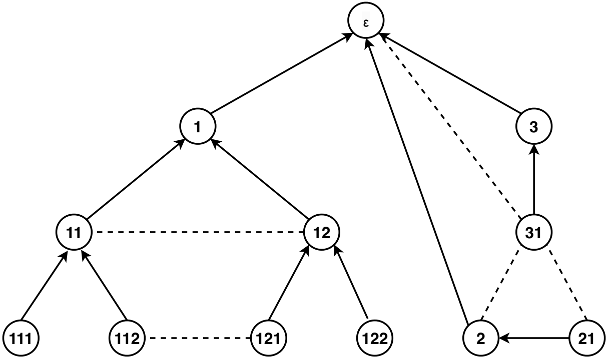
The network

Exercise (1/2)



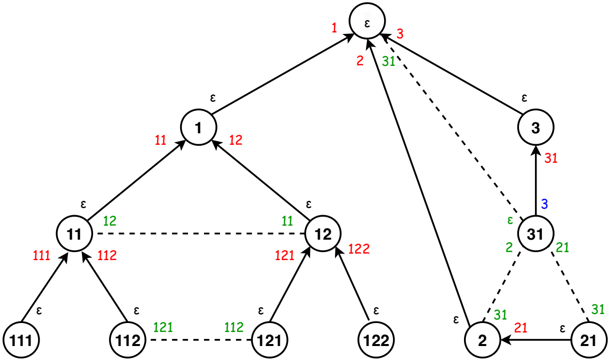
The network with an arbitrary spanning tree

Exercise (1/2)



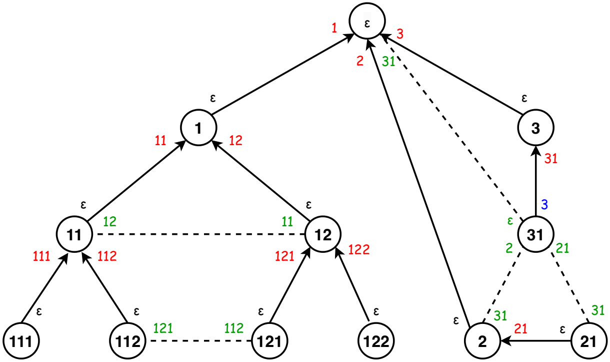
Node Labeling

Exercise (1/2)



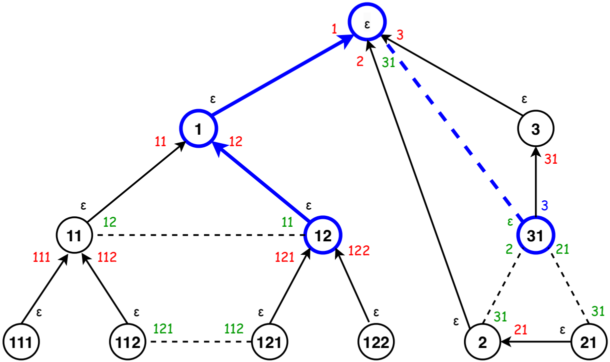
Channel Labeling

Exercise (1/2)



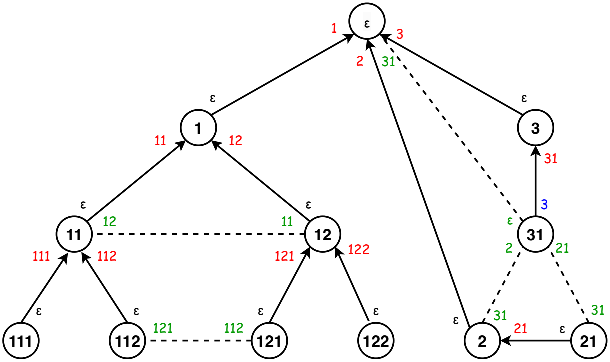
Routing from 12 to 31?

Exercise (1/2)



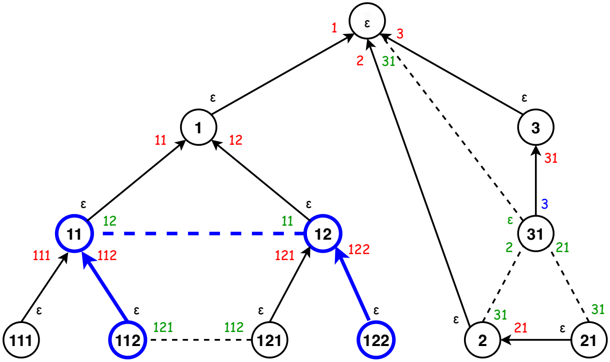
Routing from 12 to 31

Exercise (1/2)



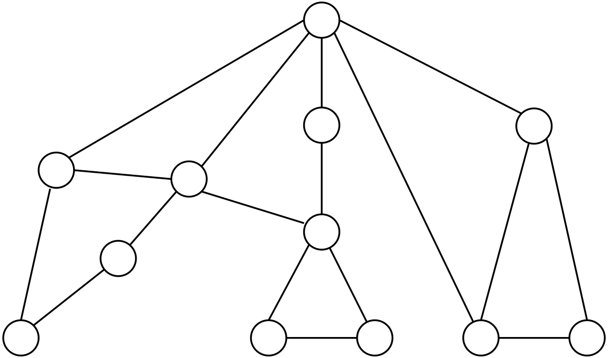
Routing from 112 to 122?

Exercise (1/2)



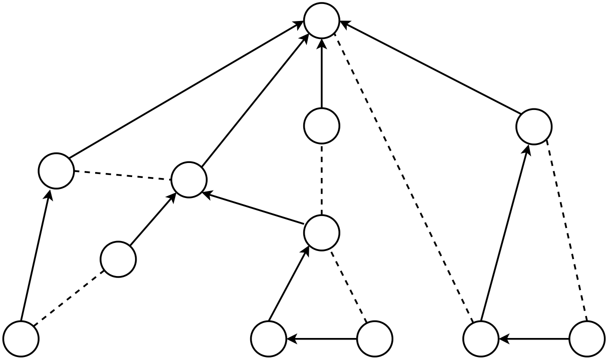
Routing from 112 to 122

Exercise (2/2)



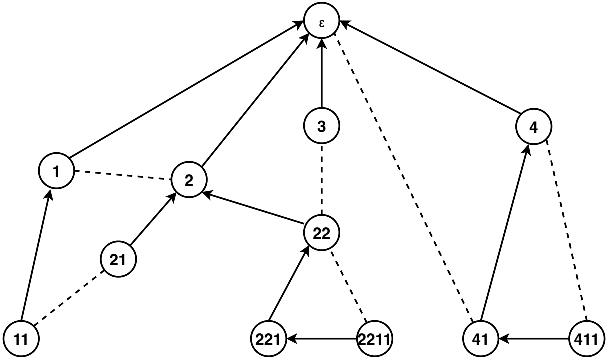
The network

Exercise (2/2)



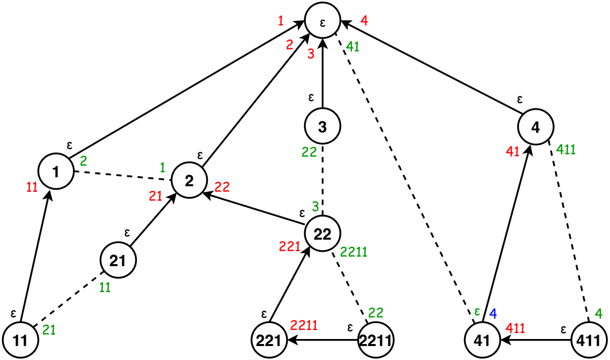
The network with an arbitrary spanning tree

Exercise (2/2)



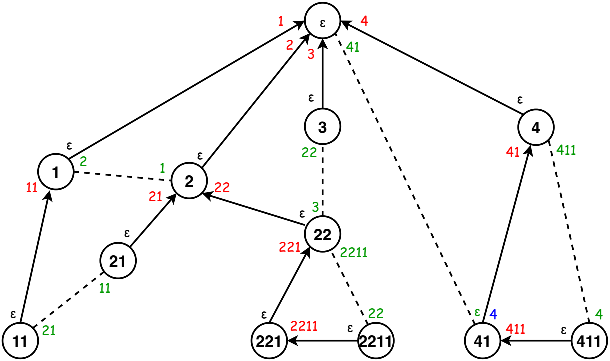
Node Labeling

Exercise (2/2)



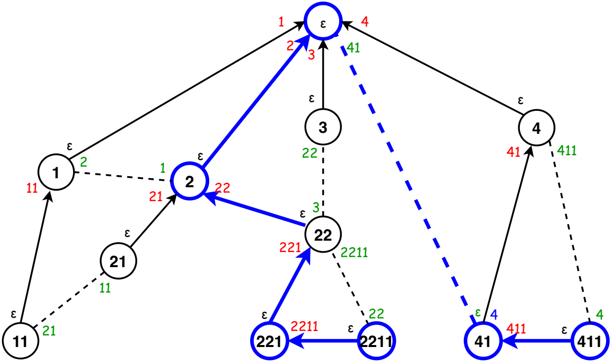
Channel Labeling

Exercise (2/2)



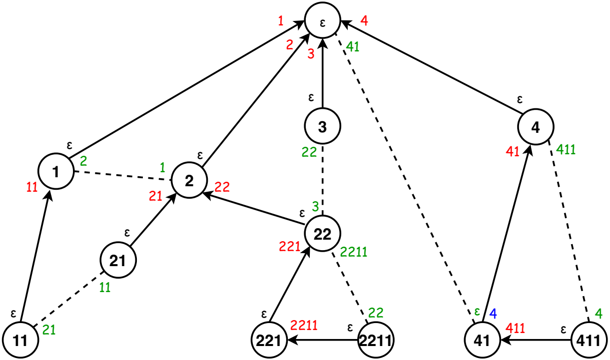
Routing from 2211 to 411?

Exercise (2/2)



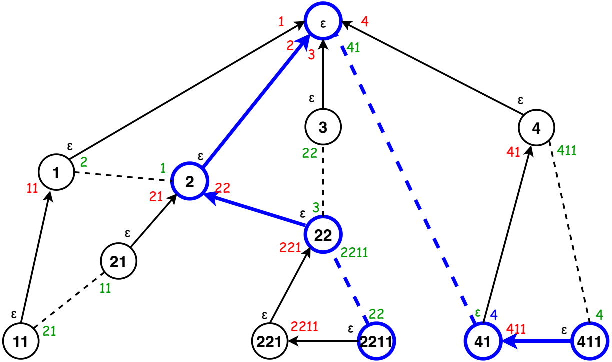
Routing from 2211 to 411

Exercise (2/2)



Routing from 411 to 2211?

Exercise (2/2)

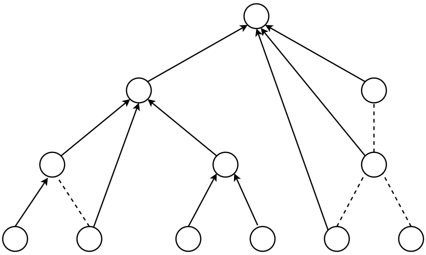


Routing from 411 to 2211

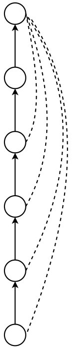
The prefix routing is not necessarily symmetric!

Rule 4: how many times?

Rule 4: how many times?



0 times



$n - 2$ times

Roadmap

Conclusion

1. A good labeling allows to save space in routing algorithms.

2. No optimal solution

   Optimization criteria for “good” routing are often conflicting: most of algorithms perform well only *w.r.t.* a subset of them.

Roadmap

References

[1] D. Medhi and K. Ramasamy.
Chapter 14 - ip address lookup algorithms.
In D. Medhi and K. Ramasamy, editors, *Network Routing (Second Edition)*, The Morgan Kaufmann Series in Networking, pages 454–499. Morgan Kaufmann, Boston, second edition edition, 2018.

[2] P. Ruzicka.
On efficiency of interval routing algorithms.
In M. Chytil, L. Janiga, and V. Koubek, editors, *Mathematical Foundations of Computer Science 1988, MFCS'88, Carlsbad, Czechoslovakia, August 29 - September 2, 1988, Proceedings*, volume 324 of *Lecture Notes in Computer Science*, pages 492–500. Springer, 1988.

[3] N. Santoro and R. Khatib.
Labelling and implicit routing in networks.
Comput. J., 28(1):5–8, 1985.

[4] G. Tel.
Introduction to Distributed Algorithms.
Cambridge University Press, USA, 2nd edition, 2001.

[5] J. van Leeuwen and R. B. Tan.
Interval routing.
Comput. J., 30(4):298–307, 1987.