Labeling and Routing

Réseaux & Communication

Alain Cournier   Stéphane Devismes

Université de Picardie Jules Verne

November 19, 2024

Roadmap

Roadmap

This lesson is mainly based on Chapter 4 of the book entitled **"Introduction to Distributed Algorithms,"** by Gerard Tel [3].

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.

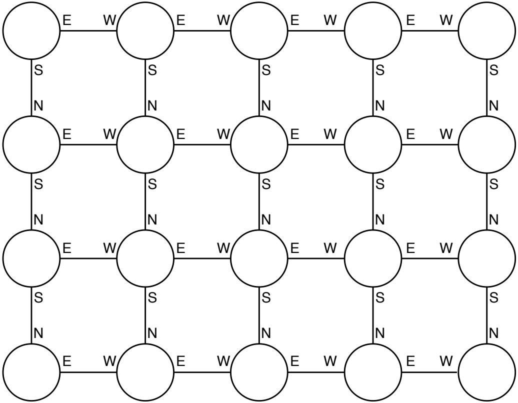
Labeling

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.

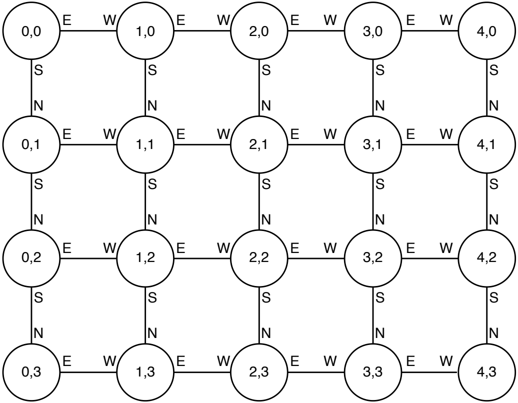
Illustrative Example

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



Illustrative Example

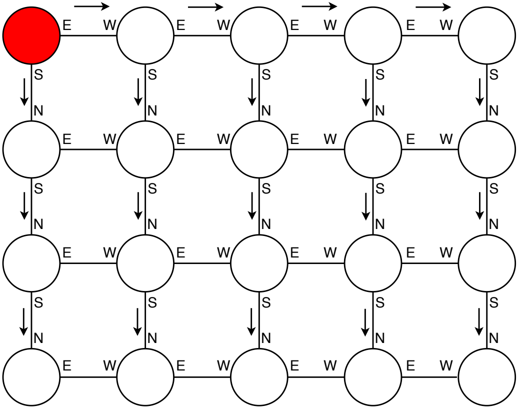
From N-S-E-W sense of direction to Coordinated System (Node Labeling)



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

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

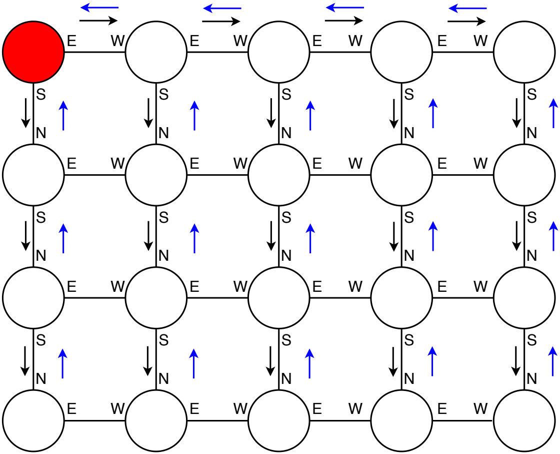
Receipt of $\langle a, b \rangle$ **from** W

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

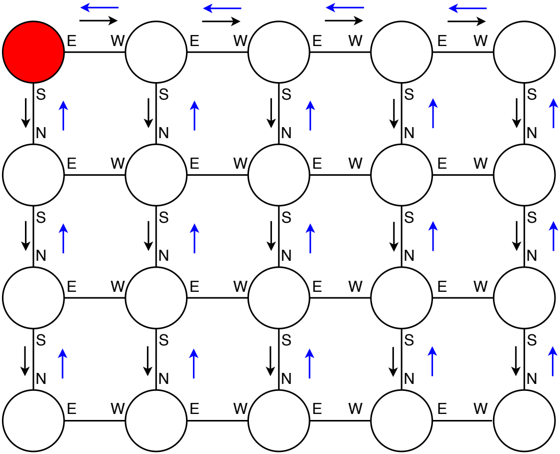
Observations

1. From all-initiators to multi-initiators: wake-up the leader using flooding
(*cf.*, distributed computing courses)
2. Time complexity: $\ell + L$, optimal (in case $\ell = L$, $\sqrt{\ell}$)
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 $(0, 0)$



Adding Termination Detection at $(0, 0)$



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

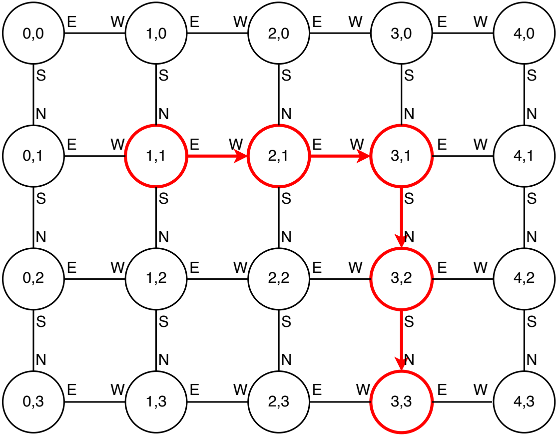
```
(x, y) ← (a, b + 1)
if S ∈ Labels then
    Send ⟨x, y⟩ to S
else
    Send ⟨Ack⟩ to N
end if
```

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

```
if N ∈ Labels then
    Send ⟨Ack⟩ to N
else if E ∉ Labels then
    Send ⟨Ack⟩ to W
else
    Cpt++
    if Cpt = 2 then
        if Labels = {S, E} then
            termination
        else
            Send ⟨Ack⟩ 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
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 $nx = x \wedge ny = y$ **then**
    `Deliver` D
else if $nx = x$ **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

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)

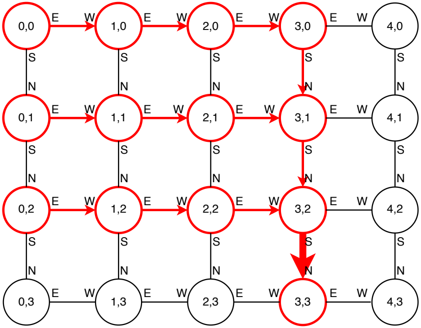
Pros and Cons

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

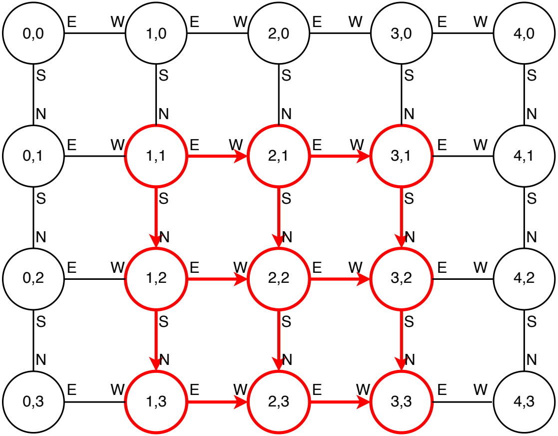
Cons:

- Not robust
- Not adaptive



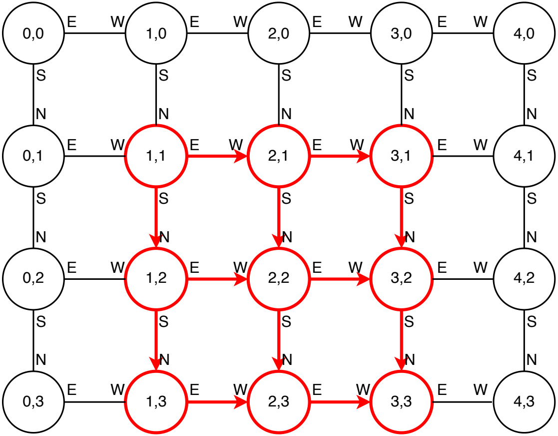
A more adaptive solution

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.



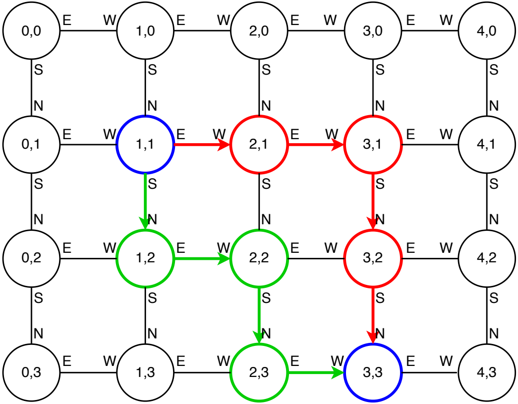
We can select one of them based on bandwidth.

Function `Routing`(D, nx, ny)

if $nx = x \land ny = y$ **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

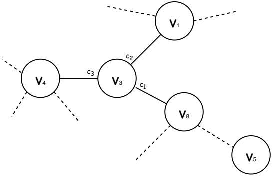
Optimization criteria for "good" routing are often conflicting: most of algorithms perform well only *w.r.t.* a subset of them.

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 [2]
- **Interval Routing**, by Leeuwen and Tan [4]

Roadmap

Principle

In-tree network T[1]

Goal: Considering a connected network 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 $\mathbb{Z}$

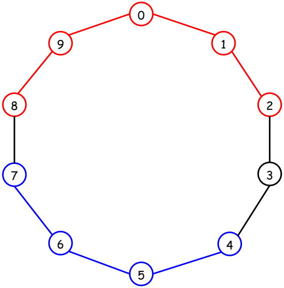
[1] A generalization to arbitrary connected network at the end of the subsection.

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 $n = 10$.



- $[4, 8)$ is in blue
- $[8, 3)$ is in red

Remarks:

- $[a, a) = \mathbb{Z}_n$
- 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:

- assign a 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

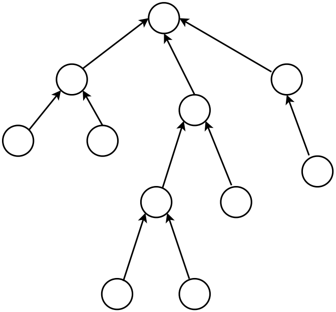
Routing using labels and cyclic intervals

The algorithm

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

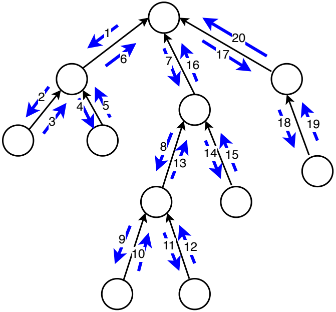
if $d = l_u$ **then**
  `deliver` p
else
  let $\alpha_i(u)$ such that $d \in [\alpha_i(u), \alpha_{i+1}(u))$
  send p via the channel labeled with $\alpha_i(u)$
end if

Node Labeling



A tree of $n = 11$ nodes

Node Labeling

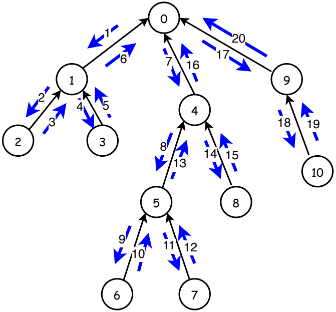


A tree of $n = 11$ nodes

Preorder tree traversal

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

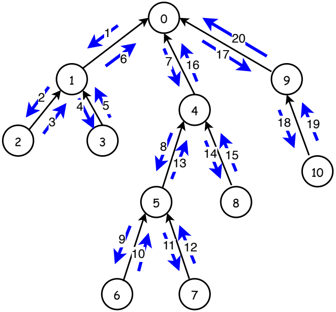
Node Labeling



A tree of $n = 11$ nodes

Preorder tree traversal + node labeling

Node Labeling



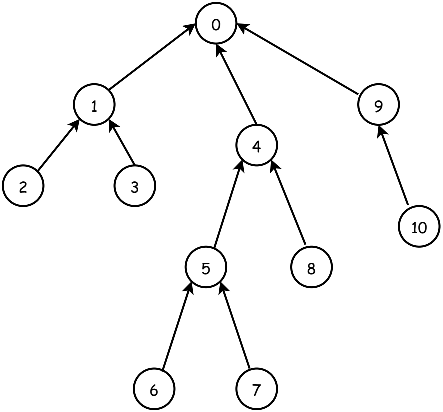
A tree of $n = 11$ nodes

Preorder tree traversal + node labeling

Property:
Labels in $T(u) : \{l_u, \ldots, l_u + |T(u)| - 1\}$

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

Channel Labeling



A tree of $n = 11$ nodes

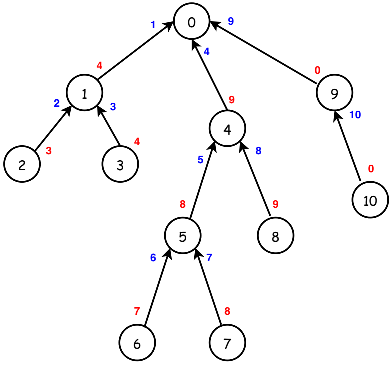
Labeling: let u be a node. For every neighbor v of u, we assign the label $\alpha_v(u) = A_v(u) \bmod n$ to the channel of u outgoing to v, where $A_v(u)$ is set as follows:

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

Remark: If v is a child of u, then $\alpha_v(u) = l_v$ since $l_v < n$.

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

Channel Labeling



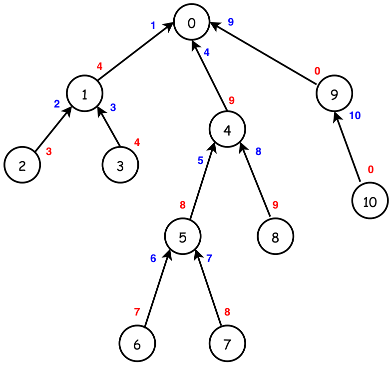
A tree of $n = 11$ nodes

Labeling: let u be a node. For every neighbor v of u, we assign the label $\alpha_v(u) = A_v(u) \bmod n$ to the channel of u outgoing to v, where $A_v(u)$ is set as follows:

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

Remark: If v is a child of u, then $\alpha_v(u) = l_v$ since $l_v < n$.

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



A tree of $n = 11$ nodes

Labeling: let u be a node. For every neighbor v of u, we assign the label $\alpha_v(u) = A_v(u) \bmod n$ to the channel of u outgoing to v, where $A_v(u)$ is set as follows:

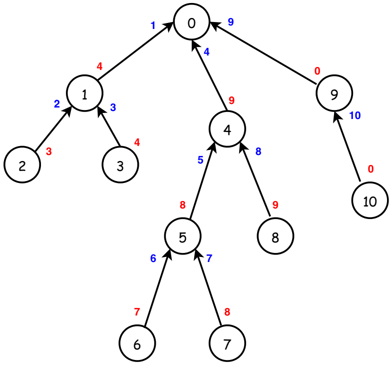
- $A_v(u) = l_v$ if v is a child of u
- $A_v(u) = l_u + |T(u)|$ if v is the parent of u.

Remark: If v is a child of u, then $\alpha_v(u) = l_v$ since $l_v < n$.

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

Examples:

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



A tree of $n = 11$ nodes

Let nbc_u be the number of children of u.

Properties:

1. If u is not the root, $\delta_u = nbc_u + 1$ and $\alpha_{\delta_u}(u)$ is the label of u outgoing to its parent, otherwise $\delta_u = nbc_u$.

   We order u's children: the ith child of u, with $i \in \{1, \ldots, nbc_u\}$, is the one, say v, of label $\alpha_i(u)$ (*n.b.*, $\alpha_i(u){=}l_v$ since $l_v{<}n$)

2. $0 < \alpha_1(u) < \ldots < \alpha_{nbc_u}(u) < n$

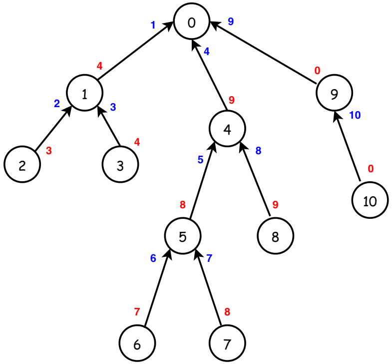
3. $\forall i \in \{1, \ldots, nbc_u\}$,
   $\forall x \in [\alpha_i(u), \alpha_{i+1}(u))$, $x \geq \alpha_i(u)$

4. Let $i \in \{1, \ldots, nbc_u - 1\}$. Let v and w be the ith and $(i+1)$th child of u, resp.

   Labels in $T(v) : \{l_v, \ldots, l_w - 1\} = [l_v, l_w) = [\alpha_i(u), \alpha_{i+1}(u))$

   E.g., labels in the subtree of the 1st child of 4 (label 5) range from 5 to 7 (*i.e.*, the label of the 2nd child of 4, 8, minus 1)

Channel Labeling



A tree of $n = 11$ nodes

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.

Let nbc_u be the number of children of u.

Properties:

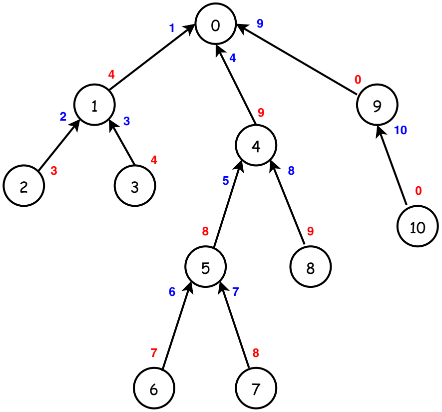
1. If u is not the root, $\delta_u = nbc_u + 1$ and $\alpha_{\delta_u}(u)$ is the label of u outgoing to its parent, otherwise $\delta_u = nbc_u$.

   We order u's children: the ith child of u, with $i \in \{1, \ldots, nbc_u\}$, is the one, say v, of label $\alpha_i(u)$ (*n.b.*, $\alpha_i(u) = l_v$ since $l_v < n$)
2. $0 < \alpha_1(u) < \ldots < \alpha_{nbc_u}(u) < n$
3. $\forall i \in \{1, \ldots, nbc_u\}$,
   $\forall x \in [\alpha_i(u), \alpha_{i+1}(u)), x \geq \alpha_i(u)$
4. Let $i \in \{1, \ldots, nbc_u - 1\}$. Let v and w be the ith and $(i+1)$th child of u, resp.

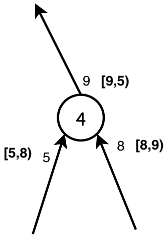
   Labels in $T(v) : \{l_v, \ldots, l_w - 1\} = [l_v, l_w) = [\alpha_i(u), \alpha_{i+1}(u))$

   E.g., labels in the subtree of the 1st child of 4 (label 5) range from 5 to 7 (*i.e.*, the label of the 2nd child of 4, 8, minus 1)

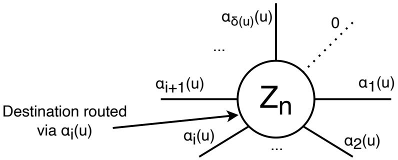
Local View



A tree of $n = 11$ nodes



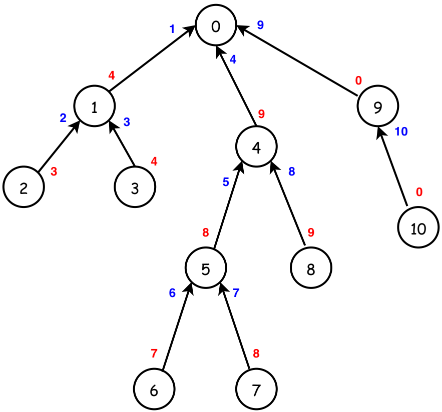
Local view at node with label 4



General scheme

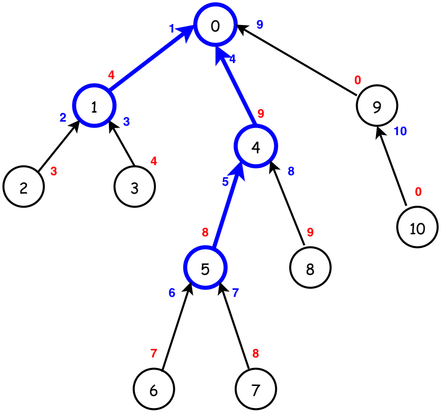
Example of routing through the labeled Tree

From label 5 to label 1



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, $1 \in [1, 4)$

Correctness

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

Preliminary result:

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

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.

1. By definition, if $\delta_u > 1$, then $[\alpha_{\delta_u}(u), \alpha_1(u))$ is the complement of $[\alpha_1(u), \alpha_{\delta_u}(u))$

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.

1. By definition, if $\delta_u > 1$, then $[\alpha_{\delta_u}(u), \alpha_1(u))$ is the complement of $[\alpha_1(u), \alpha_{\delta_u}(u))$
2. $\bigcup_{i \in \{1,\ldots,\delta_u\}} [\alpha_i(u), \alpha_{(i \bmod \delta_u)+1}(u)) = [\alpha_1(u), \alpha_{\delta_u}(u)) \cup [\alpha_{\delta_u}(u), \alpha_1(u)) = \mathbb{Z}_n$.

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.

1. By definition, if $\delta_u > 1$, then $[\alpha_{\delta_u}(u), \alpha_1(u))$ is the complement of $[\alpha_1(u), \alpha_{\delta_u}(u))$

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

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

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.

1. By definition, if $\delta_u > 1$, then $[\alpha_{\delta_u}(u), \alpha_1(u))$ is the complement of $[\alpha_1(u), \alpha_{\delta_u}(u))$
2. $\bigcup_{i\in\{1,\ldots,\delta_u\}}[\alpha_i(u), \alpha_{(i \bmod \delta_u)+1}(u)) = [\alpha_1(u), \alpha_{\delta_u}(u)) \cup [\alpha_{\delta_u}(u), \alpha_1(u)) = \mathbb{Z}_n$.

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 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.

1. By definition, if $\delta_u > 1$, then $[\alpha_{\delta_u}(u), \alpha_1(u))$ is the complement of $[\alpha_1(u), \alpha_{\delta_u}(u))$
2. $\bigcup_{i\in\{1,\ldots,\delta_u\}}[\alpha_i(u), \alpha_{(i \bmod \delta_u)+1}(u)) = [\alpha_1(u), \alpha_{\delta_u}(u)) \cup [\alpha_{\delta_u}(u), \alpha_1(u)) = \mathbb{Z}_n$.

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 1.

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

Correctness

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

Preliminary result:

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

Proof.

1. By definition, if $\delta_u > 1$, then $[\alpha_{\delta_u}(u), \alpha_1(u))$ is the complement of $[\alpha_1(u), \alpha_{\delta_u}(u))$
2. $\bigcup_{i \in \{1,\dots,\delta_u\}} [\alpha_i(u), \alpha_{(i \bmod \delta_u)+1}(u)) = [\alpha_1(u), \alpha_{\delta_u}(u)) \cup [\alpha_{\delta_u}(u), \alpha_1(u)) = \mathbb{Z}_n$.

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 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 $j < \delta_u$, $\forall y \in [\alpha_j(u), \alpha_{(j \bmod \delta_u)+1}(u))$, $y \geq \alpha_j(u)$.

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.

1. By definition, if $\delta_u > 1$, then $[\alpha_{\delta_u}(u), \alpha_1(u))$ is the complement of $[\alpha_1(u), \alpha_{\delta_u}(u))$
2. $\bigcup_{i \in \{1,\ldots,\delta_u\}} [\alpha_i(u), \alpha_{(i \bmod \delta_u)+1}(u)) = [\alpha_1(u), \alpha_{\delta_u}(u)) \cup [\alpha_{\delta_u}(u), \alpha_1(u)) = \mathbb{Z}_n$.

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 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 $j < \delta_u$, $\forall y \in [\alpha_j(u), \alpha_{(j \bmod \delta_u)+1}(u))$, $y \geq \alpha_j(u)$.

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

The result follows. □

Correctness

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

The previous result implies that will a packet has not reached its final destination, a channel is always uniquely determined for the next hop.

We now show that the channel chosen by the algorithm allows to get closer from the destination.

Correctness

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

Consider a packet p with destination v at node u.

Two cases: either $v \notin T(u)$ or $v \in T(u)$.

$v \notin T(u)$: Then, u is not the root and p should be forwarded via the parent link of u.

Correctness

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

Consider a packet p with destination v at node u.

Two cases: either $v \notin T(u)$ or $v \in T(u)$.

$v \notin T(u)$: Then, u is not the root and p should be forwarded via the parent link of u.

Now, $l_v \notin \{l_u, \ldots, l_u + |T(u)| - 1\}$.

Correctness

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

Consider a packet p with destination v at node u.

Two cases: either $v \notin T(u)$ or $v \in T(u)$.

$v \notin T(u)$: Then, u is not the root and p should be forwarded via the parent link of u.

Now, $l_v \notin \{l_u, \ldots, l_u + |T(u)| - 1\}$.

So, $l_v \neq l_u$ and $l_v \notin [l_u + 1, (l_u + |T(u)|) \bmod n)$, *i.e.*, $l_v \notin [\alpha_1(u), \alpha_{\delta_u}(u))$.

Correctness

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

Consider a packet p with destination v at node u.

Two cases: either $v \notin T(u)$ or $v \in T(u)$.

$v \notin T(u)$: Then, u is not the root and p should be forwarded via the parent link of u.

Now, $l_v \notin \{l_u, \ldots, l_u + |T(u)| - 1\}$.

So, $l_v \neq l_u$ and $l_v \notin [l_u + 1, (l_u + |T(u)|) \bmod n)$, *i.e.*, $l_v \notin [\alpha_1(u), \alpha_{\delta_u}(u))$.

As intervals are a partition of $\mathbb{Z}_n$, $l_v \in [\alpha_{\delta_u}(u), \alpha_1(u))$, which implies that p is forwarded via the parent link of u.

Correctness

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

$v \in T(u)$:

Correctness

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

$v \in T(u)$: If $v = u$, $l_v = l_u$ and u delivers p.

Correctness

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

$v \in T(u)$: If $v = u$, $l_v = l_u$ and u delivers p.

Assume now that $v \neq u$: $v \in T(w)$, where w is the ith child of u with $i \in \{1, \ldots, nbc_u\}$.

p should be forwarded toward w.

Correctness

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

$v \in T(u)$: If $v = u$, $l_v = l_u$ and u delivers p.

Assume now that $v \neq u$: $v \in T(w)$, where w is the ith child of u with $i \in \{1, \ldots, nbc_u\}$.

p should be forwarded toward w.

$l_v \in \{l_w, \ldots, l_w + |T(w)| - 1\}$, *i.e.*, $l_v \in [l_w, l_w + |T(w)|)$

Correctness

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

$v \in T(u)$: If $v = u$, $l_v = l_u$ and u delivers p.

Assume now that $v \neq u$: $v \in T(w)$, where w is the ith child of u with $i \in \{1, \dots, nbc_u\}$.

p should be forwarded toward w.

$l_v \in \{l_w, \dots, l_w + |T(w)| - 1\}$, *i.e.*, $l_v \in [l_w, l_w + |T(w)|)$

If $i < \delta_u$, $(i \bmod \delta_u) + 1 = i + 1$ and we consider two subcases: $i < nbc_u$ and $i = nbc_u$

Correctness

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

$v \in T(u)$: If $v = u$, $l_v = l_u$ and u delivers p.

Assume now that $v \neq u$: $v \in T(w)$, where w is the ith child of u with $i \in \{1, \ldots, nbc_u\}$.

p should be forwarded toward w.

$l_v \in \{l_w, \ldots, l_w + |T(w)| - 1\}$, *i.e.*, $l_v \in [l_w, l_w + |T(w)|)$

If $i < \delta_u$, $(i \bmod \delta_u) + 1 = i + 1$ and we consider two subcases: $i < nbc_u$ and $i = nbc_u$

- If $i < nbc_u$

  $l_w + |T(w)| < n$ is the label of the $(i+1)$th child of u

  so $l_v \in [\alpha_i(u), \alpha_{i+1}(u)) = [\alpha_i(u), \alpha_{(i \bmod \delta_u)+1}(u))$

Correctness

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

$v \in T(u)$: If $v = u$, $l_v = l_u$ and u delivers p.

Assume now that $v \neq u$: $v \in T(w)$, where w is the ith child of u with $i \in \{1, \ldots, nbc_u\}$.

p should be forwarded toward w.

$l_v \in \{l_w, \ldots, l_w + |T(w)| - 1\}$, *i.e.*, $l_v \in [l_w, l_w + |T(w)|)$

If $i < \delta_u$, $(i \bmod \delta_u) + 1 = i + 1$ and we consider two subcases: $i < nbc_u$ and $i = nbc_u$

- If $i < nbc_u$

  $l_w + |T(w)| < n$ is the label of the $(i+1)$th child of u

  so $l_v \in [\alpha_i(u), \alpha_{i+1}(u)) = [\alpha_i(u), \alpha_{(i \bmod \delta_u)+1}(u))$

- Otherwise, $i = nbc_u$

  u is not the root, $i + 1 = \delta_u$, and $(l_w + |T(w)|) \bmod n = \alpha_{\delta_u}(u)$, the label of the parent of u

Correctness

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

$v \in T(u)$: If $v = u$, $l_v = l_u$ and u delivers p.

Assume now that $v \neq u$: $v \in T(w)$, where w is the ith child of u with $i \in \{1, \dots, nbc_u\}$.

p should be forwarded toward w.

$l_v \in \{l_w, \dots, l_w + |T(w)| - 1\}$, *i.e.*, $l_v \in [l_w, l_w + |T(w)|)$

If $i < \delta_u$, $(i \bmod \delta_u) + 1 = i + 1$ and we consider two subcases: $i < nbc_u$ and $i = nbc_u$

- If $i < nbc_u$

  $l_w + |T(w)| < n$ is the label of the $(i+1)$th child of u

  so $l_v \in [\alpha_i(u), \alpha_{i+1}(u)) = [\alpha_i(u), \alpha_{(i \bmod \delta_u)+1}(u))$

- Otherwise, $i = nbc_u$

  u is not the root, $i + 1 = \delta_u$, and $(l_w + |T(w)|) \bmod n = \alpha_{\delta_u}(u)$, the label of the parent of u

  Now, $l_v \geq l_w = \alpha_i(u)$. Moreover, $l_w + |T(w)| \leq n$.

Correctness

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

$v \in T(u)$: If $v = u$, $l_v = l_u$ and u delivers p.

Assume now that $v \neq u$: $v \in T(w)$, where w is the ith child of u with $i \in \{1, \ldots, nbc_u\}$.

p should be forwarded toward w.

$l_v \in \{l_w, \ldots, l_w + |T(w)| - 1\}$, *i.e.*, $l_v \in [l_w, l_w + |T(w)|)$

If $i < \delta_u$, $(i \bmod \delta_u) + 1 = i + 1$ and we consider two subcases: $i < nbc_u$ and $i = nbc_u$

- If $i < nbc_u$

  $l_w + |T(w)| < n$ is the label of the $(i+1)$th child of u

  so $l_v \in [\alpha_i(u), \alpha_{i+1}(u)) = [\alpha_i(u), \alpha_{(i \bmod \delta_u)+1}(u))$

- Otherwise, $i = nbc_u$

  u is not the root, $i + 1 = \delta_u$, and $(l_w + |T(w)|) \bmod n = \alpha_{\delta_u}(u)$, the label of the parent of u

  Now, $l_v \geq l_w = \alpha_i(u)$. Moreover, $l_w + |T(w)| \leq n$.

  If $(l_w + |T(w)|) \bmod n = l_w + |T(w)|$, $l_v \leq l_w + |T(w)| = (l_w + |T(w)|) \bmod n = \alpha_{\delta_u}(u)$. So, $l_v \in [\alpha_i(u), \alpha_{\delta_u}(u))$.

Correctness

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

$v \in T(u)$: If $v = u$, $l_v = l_u$ and u delivers p.

Assume now that $v \neq u$: $v \in T(w)$, where w is the ith child of u with $i \in \{1, \ldots, nbc_u\}$.

p should be forwarded toward w.

$l_v \in \{l_w, \ldots, l_w + |T(w)| - 1\}$, *i.e.*, $l_v \in [l_w, l_w + |T(w)|)$

If $i < \delta_u$, $(i \bmod \delta_u) + 1 = i + 1$ and we consider two subcases: $i < nbc_u$ and $i = nbc_u$

- If $i < nbc_u$

  $l_w + |T(w)| < n$ is the label of the $(i+1)$th child of u

  so $l_v \in [\alpha_i(u), \alpha_{i+1}(u)) = [\alpha_i(u), \alpha_{(i \bmod \delta_u)+1}(u))$

- Otherwise, $i = nbc_u$

  u is not the root, $i + 1 = \delta_u$, and $(l_w + |T(w)|) \bmod n = \alpha_{\delta_u}(u)$, the label of the parent of u

  Now, $l_v \geq l_w = \alpha_i(u)$. Moreover, $l_w + |T(w)| \leq n$.

  If $(l_w + |T(w)|) \bmod n = l_w + |T(w)|$, $l_v \leq l_w + |T(w)| = (l_w + |T(w)|) \bmod n = \alpha_{\delta_u}(u)$. So, $l_v \in [\alpha_i(u), \alpha_{\delta_u}(u))$.

  Otherwise, $(l_w + |T(w)|) \bmod n = 0 = \alpha_{\delta_u}(u)$. As $l_v \leq n - 1$ and $\alpha_{\delta_u}(u) < \alpha_i(u)$, we also have $l_v \in [\alpha_i(u), \alpha_{\delta_u}(u))$.

Correctness

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

$v \in T(u)$: If $v = u$, $l_v = l_u$ and u delivers p.

Assume now that $v \neq u$: $v \in T(w)$, where w is the ith child of u with $i \in \{1, \ldots, nbc_u\}$.

p should be forwarded toward w.

$l_v \in \{l_w, \ldots, l_w + |T(w)| - 1\}$, *i.e.*, $l_v \in [l_w, l_w + |T(w)|)$

If $i < \delta_u$, $(i \bmod \delta_u) + 1 = i + 1$ and we consider two subcases: $i < nbc_u$ and $i = nbc_u$

- If $i < nbc_u$

  $l_w + |T(w)| < n$ is the label of the $(i+1)$th child of u

  so $l_v \in [\alpha_i(u), \alpha_{i+1}(u)) = [\alpha_i(u), \alpha_{(i \bmod \delta_u)+1}(u))$

- Otherwise, $i = nbc_u$

  u is not the root, $i + 1 = \delta_u$, and $(l_w + |T(w)|) \bmod n = \alpha_{\delta_u}(u)$, the label of the parent of u

  Now, $l_v \geq l_w = \alpha_i(u)$. Moreover, $l_w + |T(w)| \leq n$.

  If $(l_w + |T(w)|) \bmod n = l_w + |T(w)|$, $l_v \leq l_w + |T(w)| = (l_w + |T(w)|) \bmod n = \alpha_{\delta_u}(u)$. So, $l_v \in [\alpha_i(u), \alpha_{\delta_u}(u))$.

  Otherwise, $(l_w + |T(w)|) \bmod n = 0 = \alpha_{\delta_u}(u)$. As $l_v \leq n - 1$ and $\alpha_{\delta_u}(u) < \alpha_i(u)$, we also have $l_v \in [\alpha_i(u), \alpha_{\delta_u}(u))$.

  Thus, in either cases, $l_v \in [\alpha_i(u), \alpha_{\delta_u}(u)) = [\alpha_i(u), \alpha_{(i \bmod \delta_u)+1}(u))$.

Correctness

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

$v \in T(u)$: If $v = u$, $l_v = l_u$ and u delivers p.

Assume now that $v \neq u$: $v \in T(w)$, where w is the ith child of u with $i \in \{1, \ldots, nbc_u\}$.

p should be forwarded toward w.

$l_v \in \{l_w, \ldots, l_w + |T(w)| - 1\}$, *i.e.*, $l_v \in [l_w, l_w + |T(w)|)$

If $i < \delta_u$, $(i \bmod \delta_u) + 1 = i + 1$ and we consider two subcases: $i < nbc_u$ and $i = nbc_u$

- If $i < nbc_u$

  $l_w + |T(w)| < n$ is the label of the $(i+1)$th child of u

  so $l_v \in [\alpha_i(u), \alpha_{i+1}(u)) = [\alpha_i(u), \alpha_{(i \bmod \delta_u)+1}(u))$

- Otherwise, $i = nbc_u$

  u is not the root, $i + 1 = \delta_u$, and $(l_w + |T(w)|) \bmod n = \alpha_{\delta_u}(u)$, the label of the parent of u

  Now, $l_v \geq l_w = \alpha_i(u)$. Moreover, $l_w + |T(w)| \leq n$.

  If $(l_w + |T(w)|) \bmod n = l_w + |T(w)|$, $l_v \leq l_w + |T(w)| = (l_w + |T(w)|) \bmod n = \alpha_{\delta_u}(u)$. So, $l_v \in [\alpha_i(u), \alpha_{\delta_u}(u))$.

  Otherwise, $(l_w + |T(w)|) \bmod n = 0 = \alpha_{\delta_u}(u)$. As $l_v \leq n - 1$ and $\alpha_{\delta_u}(u) < \alpha_i(u)$, we also have $l_v \in [\alpha_i(u), \alpha_{\delta_u}(u))$.

  Thus, in either cases, $l_v \in [\alpha_i(u), \alpha_{\delta_u}(u)) = [\alpha_i(u), \alpha_{(i \bmod \delta_u)+1}(u))$.

Otherwise, $i = \delta_u$ and u is the root.

Correctness

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

$v \in T(u)$: If $v = u$, $l_v = l_u$ and u delivers p.

Assume now that $v \neq u$: $v \in T(w)$, where w is the ith child of u with $i \in \{1, \ldots, nbc_u\}$.

p should be forwarded toward w.

$l_v \in \{l_w, \ldots, l_w + |T(w)| - 1\}$, *i.e.*, $l_v \in [l_w, l_w + |T(w)|)$

If $i < \delta_u$, $(i \bmod \delta_u) + 1 = i + 1$ and we consider two subcases: $i < nbc_u$ and $i = nbc_u$

- If $i < nbc_u$

  $l_w + |T(w)| < n$ is the label of the $(i+1)$th child of u

  so $l_v \in [\alpha_i(u), \alpha_{i+1}(u)) = [\alpha_i(u), \alpha_{(i \bmod \delta_u)+1}(u))$

- Otherwise, $i = nbc_u$

  u is not the root, $i + 1 = \delta_u$, and $(l_w + |T(w)|) \bmod n = \alpha_{\delta_u}(u)$, the label of the parent of u

  Now, $l_v \geq l_w = \alpha_i(u)$. Moreover, $l_w + |T(w)| \leq n$.

  If $(l_w + |T(w)|) \bmod n = l_w + |T(w)|$, $l_v \leq l_w + |T(w)| = (l_w + |T(w)|) \bmod n = \alpha_{\delta_u}(u)$. So, $l_v \in [\alpha_i(u), \alpha_{\delta_u}(u))$.

  Otherwise, $(l_w + |T(w)|) \bmod n = 0 = \alpha_{\delta_u}(u)$. As $l_v \leq n - 1$ and $\alpha_{\delta_u}(u) < \alpha_i(u)$, we also have $l_v \in [\alpha_i(u), \alpha_{\delta_u}(u))$.

  Thus, in either cases, $l_v \in [\alpha_i(u), \alpha_{\delta_u}(u)) = [\alpha_i(u), \alpha_{(i \bmod \delta_u)+1}(u))$.

Otherwise, $i = \delta_u$ and u is the root.

$l_w \leq l_v < n$. So, $l_v \in [l_w, n) = [\alpha_i(u), n) = [\alpha_{\delta_u}(u), n)$.

Correctness

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

$v \in T(u)$: If $v = u$, $l_v = l_u$ and u delivers p.

Assume now that $v \neq u$: $v \in T(w)$, where w is the ith child of u with $i \in \{1, \ldots, nbc_u\}$.

p should be forwarded toward w.

$l_v \in \{l_w, \ldots, l_w + |T(w)| - 1\}$, *i.e.*, $l_v \in [l_w, l_w + |T(w)|)$

If $i < \delta_u$, $(i \bmod \delta_u) + 1 = i + 1$ and we consider two subcases: $i < nbc_u$ and $i = nbc_u$

- If $i < nbc_u$

  $l_w + |T(w)| < n$ is the label of the $(i+1)$th child of u

  so $l_v \in [\alpha_i(u), \alpha_{i+1}(u)) = [\alpha_i(u), \alpha_{(i \bmod \delta_u)+1}(u))$

- Otherwise, $i = nbc_u$

  u is not the root, $i + 1 = \delta_u$, and $(l_w + |T(w)|) \bmod n = \alpha_{\delta_u}(u)$, the label of the parent of u

  Now, $l_v \geq l_w = \alpha_i(u)$. Moreover, $l_w + |T(w)| \leq n$.

  If $(l_w + |T(w)|) \bmod n = l_w + |T(w)|$, $l_v \leq l_w + |T(w)| = (l_w + |T(w)|) \bmod n = \alpha_{\delta_u}(u)$. So, $l_v \in [\alpha_i(u), \alpha_{\delta_u}(u))$.

  Otherwise, $(l_w + |T(w)|) \bmod n = 0 = \alpha_{\delta_u}(u)$. As $l_v \leq n - 1$ and $\alpha_{\delta_u}(u) < \alpha_i(u)$, we also have $l_v \in [\alpha_i(u), \alpha_{\delta_u}(u))$.

  Thus, in either cases, $l_v \in [\alpha_i(u), \alpha_{\delta_u}(u)) = [\alpha_i(u), \alpha_{(i \bmod \delta_u)+1}(u))$.

Otherwise, $i = \delta_u$ and u is the root.

$l_w \leq l_v < n$. So, $l_v \in [l_w, n) = [\alpha_i(u), n) = [\alpha_{\delta_u}(u), n)$.

Since $\alpha_{\delta_u}(u) > \alpha_1(u)$, $l_v \in [\alpha_{\delta_u}(u), \alpha_1(u)) = [\alpha_i(u), \alpha_{(i \bmod \delta_u)+1}(u))$.

Correctness

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

$v \in T(u)$: If $v = u$, $l_v = l_u$ and u delivers p.

Assume now that $v \neq u$: $v \in T(w)$, where w is the ith child of u with $i \in \{1, \dots, nbc_u\}$.

p should be forwarded toward w.

$l_v \in \{l_w, \dots, l_w + |T(w)| - 1\}$, *i.e.*, $l_v \in [l_w, l_w + |T(w)|)$

If $i < \delta_u$, $(i \bmod \delta_u) + 1 = i + 1$ and we consider two subcases: $i < nbc_u$ and $i = nbc_u$

- If $i < nbc_u$

  $l_w + |T(w)| < n$ is the label of the $(i + 1)$th child of u

  so $l_v \in [\alpha_i(u), \alpha_{i+1}(u)) = [\alpha_i(u), \alpha_{(i \bmod \delta_u)+1}(u))$

- Otherwise, $i = nbc_u$

  u is not the root, $i + 1 = \delta_u$, and $(l_w + |T(w)|) \bmod n = \alpha_{\delta_u}(u)$, the label of the parent of u

  Now, $l_v \geq l_w = \alpha_i(u)$. Moreover, $l_w + |T(w)| \leq n$.

  If $(l_w + |T(w)|) \bmod n = l_w + |T(w)|$, $l_v \leq l_w + |T(w)| = (l_w + |T(w)|) \bmod n = \alpha_{\delta_u}(u)$. So, $l_v \in [\alpha_i(u), \alpha_{\delta_u}(u))$.

  Otherwise, $(l_w + |T(w)|) \bmod n = 0 = \alpha_{\delta_u}(u)$. As $l_v \leq n - 1$ and $\alpha_{\delta_u}(u) < \alpha_i(u)$, we also have $l_v \in [\alpha_i(u), \alpha_{\delta_u}(u))$.

  Thus, in either cases, $l_v \in [\alpha_i(u), \alpha_{\delta_u}(u)) = [\alpha_i(u), \alpha_{(i \bmod \delta_u)+1}(u))$.

Otherwise, $i = \delta_u$ and u is the root.

$l_w \leq l_v < n$. So, $l_v \in [l_w, n) = [\alpha_i(u), n) = [\alpha_{\delta_u}(u), n)$.

Since $\alpha_{\delta_u}(u) > \alpha_1(u)$, $l_v \in [\alpha_{\delta_u}(u), \alpha_1(u)) = [\alpha_i(u), \alpha_{(i \bmod \delta_u)+1}(u))$.

Hence, in all cases, $l_v \in [\alpha_i(u), \alpha_{(i \bmod \delta_u)+1}(u))$ and, since intervals are a partition of $\mathbb{Z}_n$, p is forwarded toward w, and we are done. $\square$

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

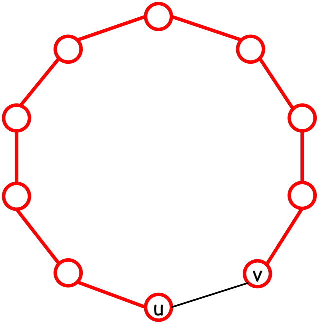
Leader election + spanning tree (with initialization and term. detect. at leader), token circulation in the tree ($O(mn)$ messages, $O(m)$ rounds, and $O(\delta_u + B)$ bits, where B the number of bits to store an identifier)
(*cf.*, distributed computing courses)

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 δ_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

Roadmap

Definition

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

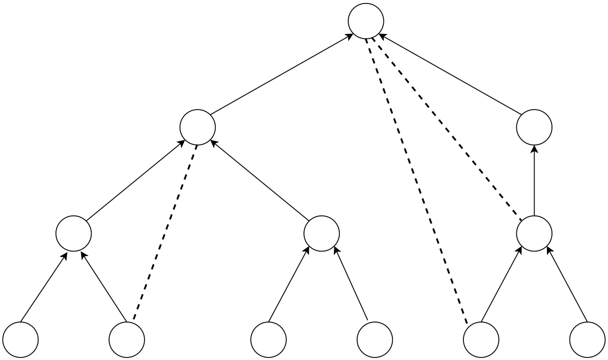
1. An assignment of different labels from $\mathbb{Z}_n$ to the nodes of G, and
2. and for each node u, an assignment of pairwise distinct labels $\alpha_i(u)$, $i = 1, \ldots, \delta_u$, to all channels of u.

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



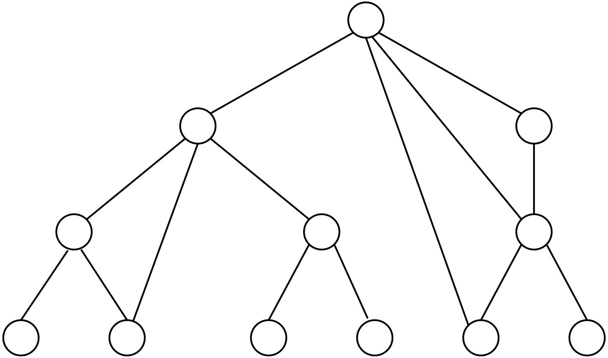
Property: **For every two neighbors u and v in G, either $u \in T(v)$, or $v \in T(u)$.**

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

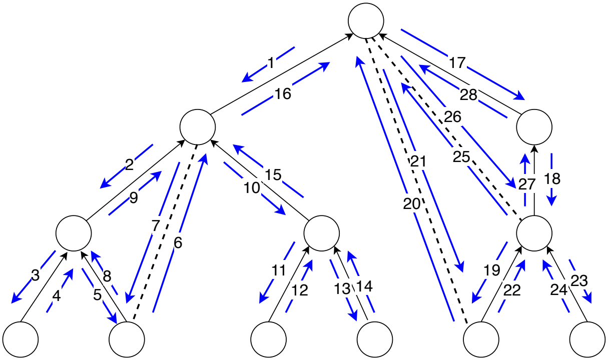
Node labeling



A network of $n = 12$ nodes

A valid ILS for arbitrary connected networks

Node labeling



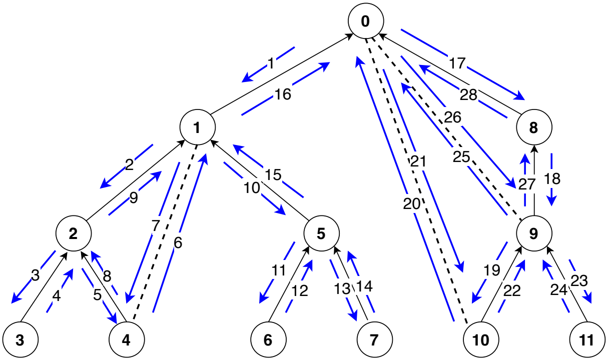
A network of $n = 12$ nodes

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

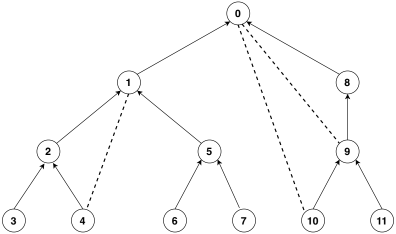
A valid ILS for arbitrary connected networks

Channel labeling

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

Yet, $A_v(u)$ is set as follows:

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



12 nodes

Remark: If v is a non-parent neighbor of u, $\alpha_v(u) = l_v$ since $l_v < n$.

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

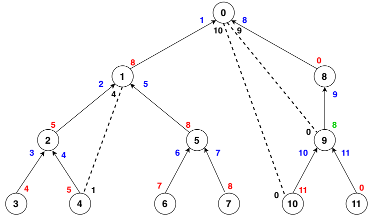
A valid ILS for arbitrary connected networks

Channel labeling

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

Yet, $A_v(u)$ is set as follows:

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



12 nodes, computed together with the node labeling

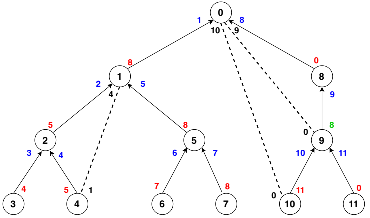
Remark: If v is a non-parent neighbor of u, $\alpha_v(u) = l_v$ since $l_v < n$.

[2] In this case, the non-tree edge is labeled 0 at u by the rule 1, so assigning $A_v(u)$ to $l_u + |T(u)|$ 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 $\alpha_v(u) = A_v(u) \bmod n$ to the channel of u outgoing to v.

Yet, $A_v(u)$ is set as follows:

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



12 nodes, computed together with the node labeling

Remark: If v is a non-parent neighbor of u, $\alpha_v(u) = l_v$ since $l_v < n$.

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 $A_v(u)$.

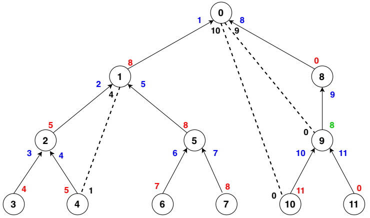
[2] In this case, the non-tree edge is labeled 0 at u by the rule 1, so assigning $A_v(u)$ to $l_u + |T(u)|$ would lead to two channels at u with the same label!

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 $\alpha_v(u) = A_v(u) \bmod n$ to the channel of u outgoing to v.

Yet, $A_v(u)$ is set as follows:

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



12 nodes, computed together with the node labeling

Remark: If v is a non-parent neighbor of u, $\alpha_v(u) = l_v$ since $l_v < n$.

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 $A_v(u)$.

Generalization: if G is a tree, G is labeled as with the tree-labeling scheme.

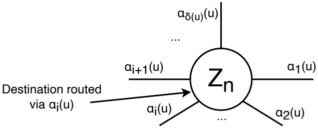
[2] In this case, the non-tree edge is labeled 0 at u by the rule 1, so assigning $A_v(u)$ to $l_u + |T(u)|$ would lead to two channels at u with the same label!

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 $\alpha_v(u) = A_v(u) \bmod n$ to the channel of u outgoing to v.

Yet, $A_v(u)$ is set as follows:

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



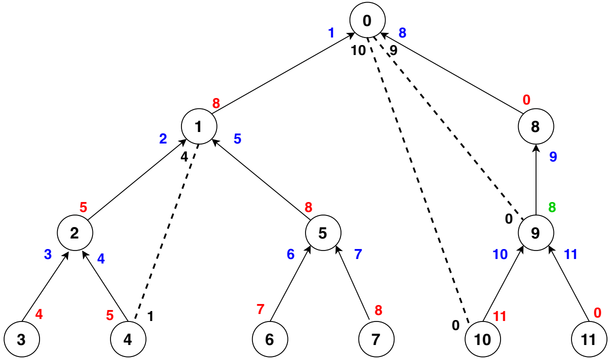
Remark: If v is a non-parent neighbor of u, $\alpha_v(u) = l_v$ since $l_v < n$.

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 $A_v(u)$.

Generalization: if G is a tree, G is labeled as with the tree-labeling scheme.

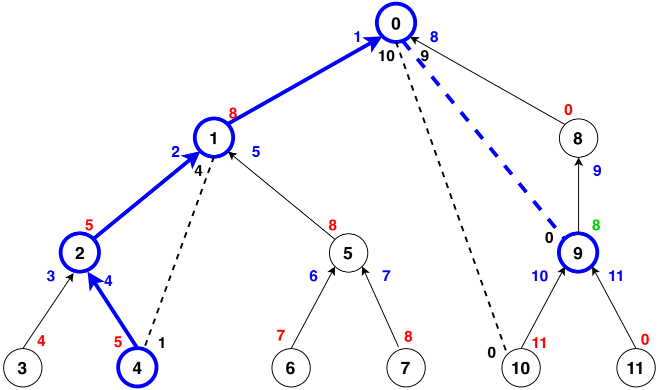
[2] In this case, the non-tree edge is labeled 0 at u by the rule 1, so assigning $A_v(u)$ to $l_u + |T(u)|$ would lead to two channels at u with the same label!

Example of Interval Routing



Routing from 4 to 9

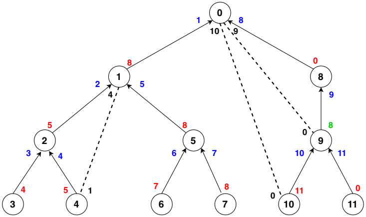
Example of Interval Routing



Routing from 4 to 9

1. *Locally at each node, the union of intervals is equal to $\mathbb{Z}_n$*

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



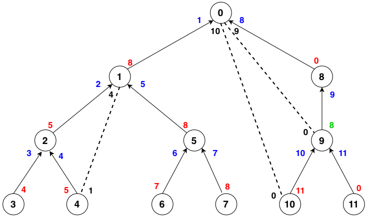
---

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

1. *Locally at each node, the union of intervals is equal to $\mathbb{Z}_n$*

   (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 destination w for the next hop.



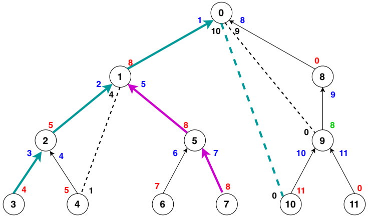
[3] $(a, b) < (c, d) \equiv [a < c \vee (a = c \wedge b < d)]$

1. *Locally at each node, the union of intervals is equal to* $\mathbb{Z}_n$

   (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 destination w for the next hop.

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



In the path 10,0,1,2,3: $l_u = 10 > l_v = 3$ and $l_w = 0 < l_u = 10$

In the 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 $l_w = 1 < l_u = 5$

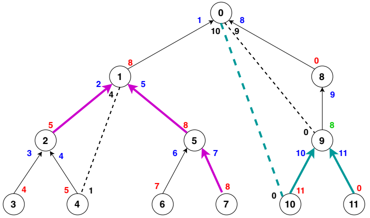
[3] $(a, b) < (c, d) \equiv [a < c \vee (a = c \wedge b < d)]$

1. *Locally at each node, the union of intervals is equal to $\mathbb{Z}_n$*

   (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 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

[3] $(a, b) < (c, d) \equiv [a < c \vee (a = c \wedge b < d)]$

A valid ILS for arbitrary connected networks

Properties

1. *Locally at each node, the union of intervals is equal to $\mathbb{Z}_n$*

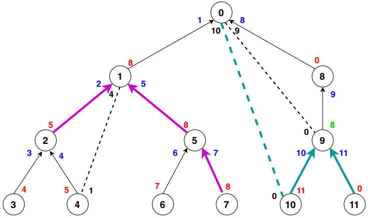
   (the proof is identical to the one for the tree-labeling scheme)

So, when u has a packet for $v \neq u$, u finds a 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 $lca(u, v)$ be the label of the lowest common ancestor of u and v and $f_v(u) = (-lca(u, v), l_u)$.[3]

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



In the path 0,10,9,11: take $l_u = 0$ and $l_w = 10$, we have $f_v(w) = (-9, 10) < f_v(u) = (0, 0)$

In the path 2,1,5,7: $l_u = 2$ and $l_w = 1$, we have $f_v(w) = (-1, 1) < f_v(u) = (-1, 2)$

[3] $(a, b) < (c, d) \equiv [a < c \vee (a = c \wedge b < d)]$

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: $l_w < l_u$.

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 father of u, we have $l_w < l_u$. □

Proof of Property 3

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

Proof:

If $v \in T(u)$, let w' be the child of u such that $v \in T(w')$.

We have $\alpha_{w'}(u) = l_{w'} \leq l_v$ and this implies that $\alpha_{w'}(u) \leq \alpha_w(u) \leq l_v < l_{w'} + |T(w')|$.

So, w is not the father f of u (indeed, either $\alpha_f(u) = l_f < l_u < l_{w'}$, $\alpha_f(u) = 0 < l_{w'}$, or $\alpha_f(u) = l_u + |T(u)| \geq l_{w'} + |T(w')|$) and $l_w = \alpha_w(u) \leq l_v$.

Proof of Property 3

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

Proof:

If $v \in T(u)$, let w' be the child of u such that $v \in T(w')$.

We have $\alpha_{w'}(u) = l_{w'} \leq l_v$ and this implies that $\alpha_{w'}(u) \leq \alpha_w(u) \leq l_v < l_{w'} + |T(w')|$.

So, w is not the father f of u (indeed, either $\alpha_f(u) = l_f < l_u < l_{w'}$, $\alpha_f(u) = 0 < l_{w'}$, or $\alpha_f(u) = l_u + |T(u)| \geq l_{w'} + |T(w')|$) and $l_w = \alpha_w(u) \leq l_v$.

Otherwise $v \notin T(u)$ and 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 father of u and $l_w < l_u < l_v$.

□

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

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 4, 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, $f_v(w) < f_v(u)$. □

- 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

A valid ILS for arbitrary connected networks

Correctness & Complexity

- 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)

A valid ILS for arbitrary connected networks

Pros and Cons

Pros:

1. More robust than tree-labeling scheme
2. Load-balancing

   (every link is used by at least one route)
3. Memory Usage: $\delta_u + 1$ labels for node u,
   i.e., $(\delta_u + 1) \times \lceil \log n \rceil$ bits

A valid ILS for arbitrary connected networks

Pros and Cons

Pros:

1. More robust than tree-labeling scheme
2. Load-balancing

   (every link is used by at least one route)
3. Memory Usage: $\delta_u + 1$ labels for node u, *i.e.*, $(\delta_u + 1) \times \lceil \log n \rceil$ bits

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 previous 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 [1]

A valid ILS for arbitrary connected networks

Pros and Cons

Pros:

1. More robust than tree-labeling scheme
2. Load-balancing

   (every link is used by at least one route)
3. Memory Usage: $\delta_u + 1$ labels for node u, *i.e.*, $(\delta_u + 1) \times \lceil \log n \rceil$ bits

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

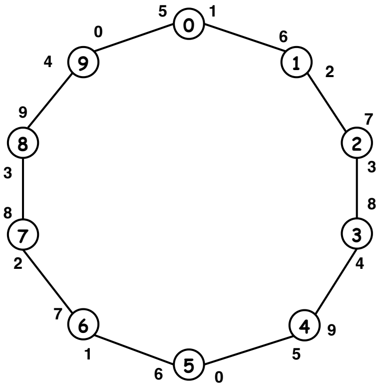
   Lower bound: in the worst case the interval routing algorithm chooses a route of length at least $\frac{3}{2}$ of the network diameter [1]

However, hop-optimal in many regular topologies, *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 $(i + \lceil \frac{n}{2} \rceil) \bmod n$



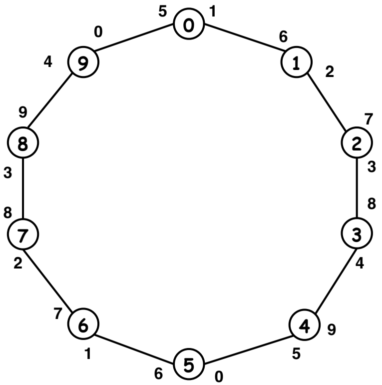
A hop-optimal valid ILS for rings

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:

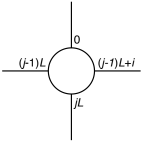
1. Packets for nodes $i+1, \ldots, (i + \lceil \frac{n}{2} \rceil) - 1$ routed *via* the clockwise channel
2. Packets for nodes $(i + \lceil \frac{n}{2} \rceil), \ldots, i-1$ routed *via* the anticlockwise channel

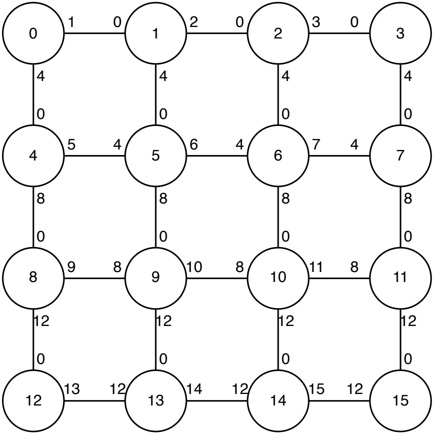


A hop-optimal valid ILS for $(L \times L)$-grids $(n = L \times L)$

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

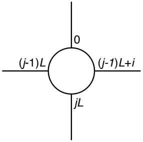




A hop-optimal valid ILS for $(L \times L)$-grids $(n = L \times L)$

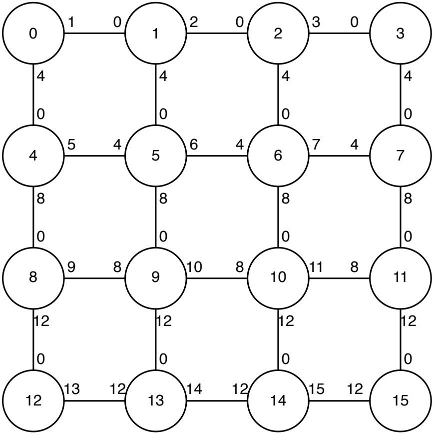
Labeling:

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



Roadmap

Idea

Based on an arbitrary spanning tree T 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,[4] the other are non-tree edges

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

Idea

Based on an arbitrary spanning tree T 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,[4] the other are non-tree edges

Efficiency can be improved starting from a BFS spanning tree

[4] *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.

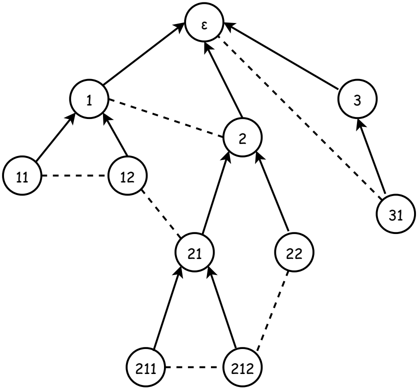
Routing Algorithm

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

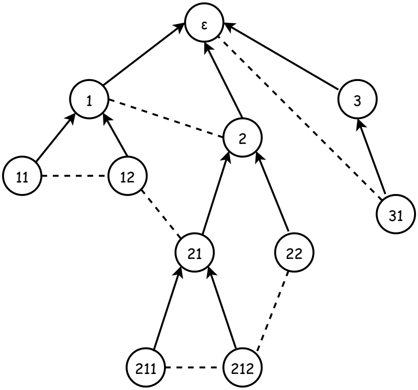
Node Labeling

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 $u_1, \ldots, l_k$ 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 $u_1, \ldots, l_k$ 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



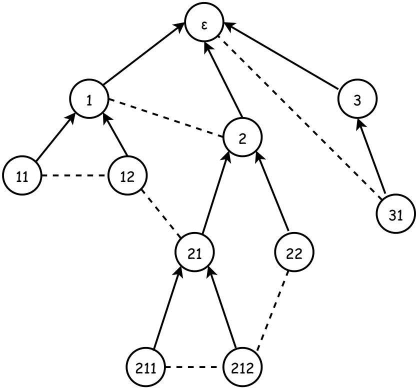
Remark: It may be distributedly computed using (BFS) spanning tree construction (with initialization and termination detection at the root) ($O(H)$ rounds, $O(n.m)$ messages of $O(H.\log|\Sigma|)$ bits, and $O(\log\Delta + H.\log|\Sigma|)$ bits per node)

(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

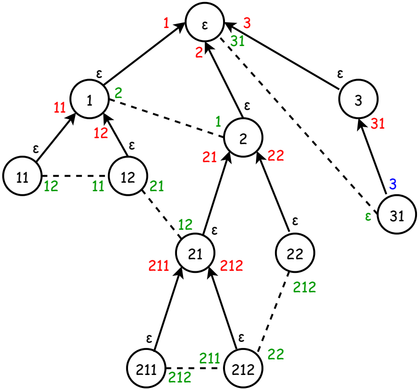
1. If $\{u, v\}$ is a non-tree edge, $\alpha_v(u) = l_v$
2. If v is a child of u, $\alpha_v(u) = l_v$
3. If v is the parent of u and u has no non-tree edge to the root,[5] $\alpha_v(u) = \epsilon$
4. If v is the parent of u and u has a non-tree edge to the root, $\alpha_v(u) = l_v$



[5] 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

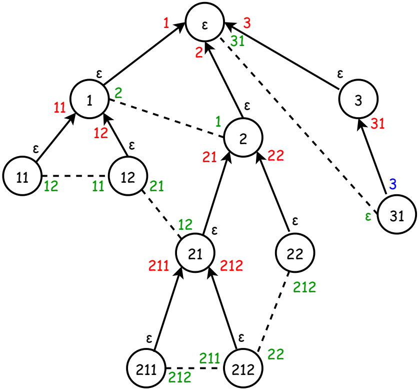
1. If $\{u, v\}$ is a non-tree edge, $\alpha_v(u) = l_v$
2. If v is a child of u, $\alpha_v(u) = l_v$
3. If v is the parent of u and u has no non-tree edge to the root,[5] $\alpha_v(u) = \epsilon$
4. If v is the parent of u and u has a non-tree edge to the root, $\alpha_v(u) = l_v$



[5] 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,[5] $\alpha_v(u) = \epsilon$
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 $l_v \lhd l_u$



[5] 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!

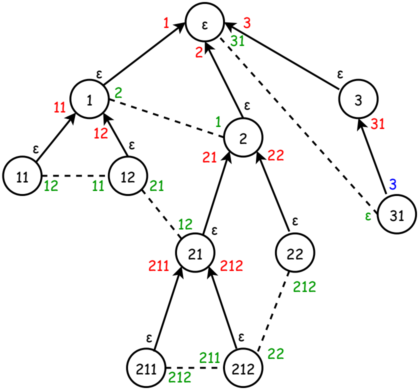
Channel Labeling

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,[5] $\alpha_v(u) = \epsilon$
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 $l_v \lhd l_u$

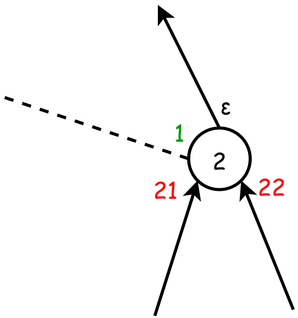
Remark: It may be distributedly computed using PIF in the tree ($O(H)$ rounds, $O(n)$ messages, and $O(\Delta.H.\log|\Sigma|)$ bits per node

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

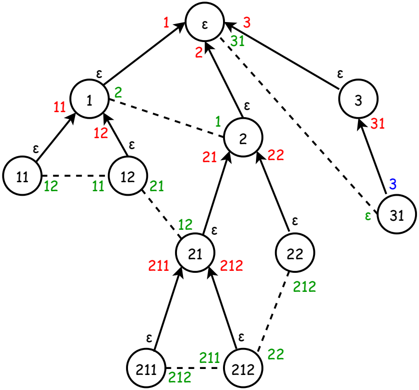


[5] Otherwise, the non-tree edge is labeled 0 at u by the rule 1, so assigning $\alpha_v(u)$ to ϵ would lead to two channels at u with the same label!

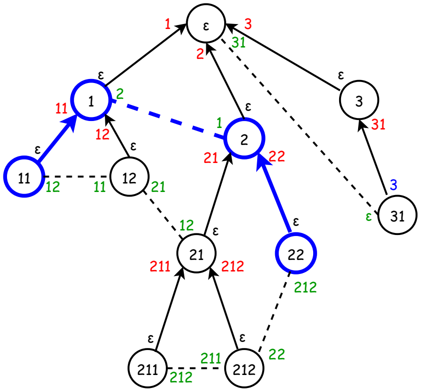
Local View



Example of Prefix Routing



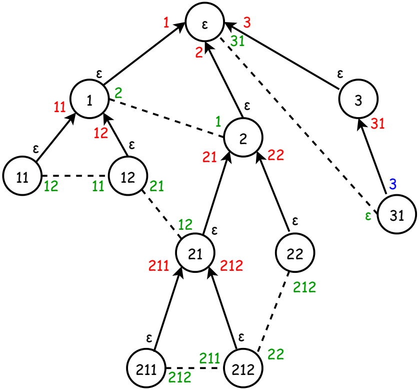
Routing from 11 to 22



Routing from 11 to 22

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

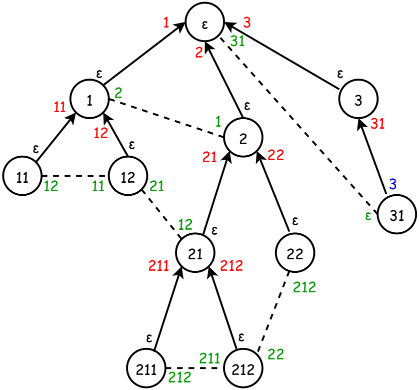
1. *For all nodes u and v such that $u \neq v$, there is a channel at u labeled with a prefix of l_v*

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.*, $211, 21, 2$ and $31, \epsilon$



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

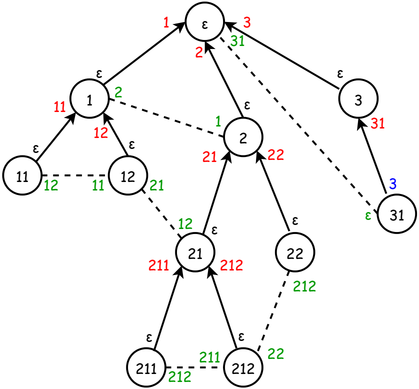
(channel labels are unique at u, and so is the one that is the longest prefix of l_v)

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

   See, *e.g.*, $211, 21, 2$ 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.*, $2, 21, 212$



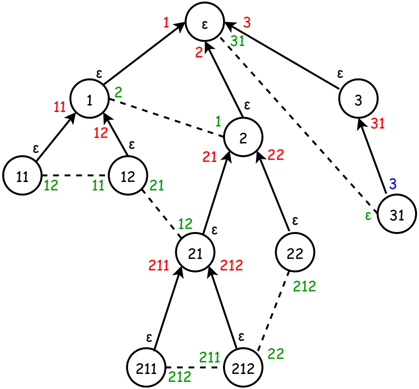
Properties

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.*, $211, 21, 2$ 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.*, $2, 21, 212$
4. *If $u \notin T(v)$, w is an ancestor of v or w is the parent of u*
   See, *e.g.*, $11, 1, 2$ 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, $\alpha_w(u) = l_w \lhd l_v$. □

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.

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 $\alpha_{w'}(u) = l_{w'} \lhd \alpha_w(u) = l_w \lhd l_v$: 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)[6] 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)

[6] 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)[6] 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.

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

Pros. and Cons.

Pros.

- Correct
- Robust

Cons.

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

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

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] 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.

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

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

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