Interval Routing

Exercise in compact routing (Santoro and Khatib, 1985). Explosive size of the routing table is a problem.

<table>
<thead>
<tr>
<th>condition</th>
<th>port number</th>
</tr>
</thead>
<tbody>
<tr>
<td>destination &gt; id</td>
<td>0</td>
</tr>
<tr>
<td>destination &lt; id</td>
<td>1</td>
</tr>
<tr>
<td>destination = id</td>
<td>(local delivery)</td>
</tr>
</tbody>
</table>

Interval routing uses this concept. To forward a message, a node finds out, to which one of a set of predefined intervals the destination id belongs.
**Defining intervals**

For a set of $N$ nodes $0 \ldots N-1$, the interval $[p, q)$ between $p$ and $q$ ($p, q < N$) is defined as follows:

If $p < q$ then $[p, q) = p, p+1, p+2, \ldots, q-2, q-1$

If $p \geq q$ then $[p, q) = p, p+1, p+2, \ldots, N, 0, \ldots, q-2, q-1$

Read the ports in the anticlockwise order, and send messages to destination $[p, q)$ via port $p$
**Interval routing on a tree**

**Step 1.** Label the root as node 0.

**Step 2.** Do a pre-order traversal of the tree, and label the successive nodes in ascending order starting from 1.

**Step 3.** For each node, label the port towards a child by the node number of the child. Then label the port towards the parent by \( L(i) + T(i) + 1 \mod N \), where \( T(i) \) is the number of nodes in the subtree under node \( i \) (including \( i \)),
**Interval routing on non-tree topology**

Construct a spanning tree and apply interval routing. But it does not utilize the non-tree edges.

---

*Example of Interval routing on a ring of processes*

The above scheme is *not optimal*. To make it optimal, label the ports of node $i$ with $i+1 \mod 8$ and $i+4 \mod 8$.

The main drawback is that it works on static topologies only. Whenever the topology changes, all labels have to be recomputed.
Prefix routing

It is more flexible than interval routing.

Prefix routing on a tree

Each label is a string from an alphabet $\sqsubseteq = (a, b, c, \ldots)$

Here, $z$ designates the empty string.

Let $X$ be the label of a destination node. A node $Y$ makes the routing decision as follows:

if $X=Y$ \[ deliver the message locally \]

$X \neq Y$ \[ find port with longest prefix of $X$ as its label; forward the message towards that port \]

fi
Chang’s Spanning tree algorithm

A quick review

Uses Dijkstra Scholten’s idea in termination detection

{for the initiator}
send probes to each neighbor; D := number of neighbors;
\[\text{do } D \neq 0 \implies \text{echo} \implies D := D - 1 \text{ od} \]

Sample run of Chang-Roberts algorithm
{for a non-initiator process i}

do

probe parent = i C=0

C:=1; sender := parent;

if i is a leaf send echo to parent;

i is not a leaf send probes to non-parent neighbors;

D:= number of non-parent neighbors

fi

echo D:= D-1

probe parent ≠ i send echo to sender;

C=1 D=0 send echo to parent; C:=0

od
**Tarry’s (Tarry 1895) traversal algorithm**

**Rule 1.** Send the token towards any neighbor at most once.

**Rule 2.** If Rule 1 cannot be used to send the token, then send the token to its parent.

A possible traversal route 0 1 2 5 3 1 4 6 2 6 4 1 3 5 2 1 0.

The edges connecting each node to its parent form a spanning tree. Why?
Graph coloring

The problem

Color the nodes of a graph using a fixed color palette, so that no two neighboring nodes have the same color.

Lemma. Any graph in which the maximum degree of a node is $D$ can be colored using $(D+1)$ colors.

But it is a bit silly (or at least an overkill) to use five colors to color the nodes of the above star graph, when only two colors are enough.
**Conversion into dag often helps**

How?

**Lemma.** Any graph in which the maximum out-degree of a node is $D$ can be colored using $(D+1)$ colors.

![Diagram of a dag]

**Six-coloring a planar graph**

Make sure you know what a planar graph is.

We will transform the planar graph $G = (V, E)$ into a dag, in which no node has an out-degree greater than five. How?
(Euler’s polyhedron formula) \( e \leq 3p - 6 \)
\[ e = |E|, \ p = |V| \]

**Lemma.** Every planar graph has at least one node with degree five or less.

Each node with degree \( \leq 5 \) will make all its edges “outgoing”. The remainder graph is also a planar graph, so the process can be recursively applied, until all edges are directed. This results in the desired dag.

So six-coloring is clearly feasible. But how will the distributed algorithm work?

Each node executes the following:

```
do
no. of undirected edges \( \leq 5 \) make all such edges outgoing
od
```

But the coloring algorithm need not wait for the completion of this phase!
program planar graph coloring;

{program for node i}

do
    outdegree(i) ≦ 5 \ j \ succ(i) : c[i] = c[j]
    c[i] := b : b \ {sc(i)}
    number of undirected edges ≦ 5 \ make all undirected edges outgoing

od