

---

---

# Routing Algorithms 2

2 June 2010  
Lecture 11

Slide credits: R. H. Mak

---

# Topics for Today

---

- Routing
  - All-pairs Shortest Path
    - Floyd-Warshall
  - Routing Algorithms
    - Toueg's Algorithm
      - Distributed version of Floyd-Warshall
    - Chandy-Misra Algorithm
  - Table Compaction
    - Tree-labeling scheme (Santoro Khatib)
    - Interval labeling scheme
    - Prefix routing

# S-paths and S-distance

---

Let  $G=(V,E,\omega)$  be a weighted graph and let  $S \subseteq V$ . A path  $\langle u_0, \dots, u_k \rangle$  is an  $S$ -path if all internal nodes belong to  $S$ , i.e.,  $u_i \in S$  for all  $0 < i < k$ . For  $u, v \in V$  the  $S$ -distance  $d^S(u,v)$  is the lowest weight of an  $S$ -path.

If  $G$  does not contain cycles with negative weight, then

$$d^S(u,u) = 0$$

$$d^0(u,v) = \begin{cases} \omega_{uv}, uv \in E \\ \infty, uv \notin E \end{cases}$$

$$d^{S \cup \{w\}}(u,v) = \min(d^S(u,v), d^S(u,w) + d^S(w,v))$$

$$d(u,v) = d^V(u,v)$$

# Floyd-Warshall

---

```
begin  $S := \{\}$ ;  
  forall  $u, v$  do  
    if  $u = v$  then  $D[u,v] := 0$ ;  
    else   if  $uv \in E$  then  $D[u,v] := \omega_{uv}$   
           else  $D[u,v] := \text{Inf}$ ;  
  
  while  $S \neq V$  do  
    (* inv. forall  $u,v : D[u,v] = d_S(u,v)$  *)  
    begin pick  $w$  from  $V \setminus S$ ;  
      forall  $u \in V$  do  
        forall  $v \in V$  do  
           $D[u,v] := \min(D[u,v], D[u,w] + D[w,v])$ ;  
         $S := S \cup \{w\}$ ;  
      end  
    end  
end
```

# So Far

---

- Routing
  - All-pairs Shortest Path
    - Floyd-Warshall
  - Routing Algorithms
    - Toueg's Algorithm
      - Distributed version of Floyd-Warshall
    - Chandy-Misra Algorithm
  - Table Compaction
    - Tree-labeling scheme (Santoro Khatib)
    - Interval labeling scheme
    - Prefix routing

# Simple Algorithm

---

```
var     $S_u$  : set of nodes;  
         $D_u$  : array of weights;  
         $Nb_u$  : array of nodes;  
  
begin   $S_u := 0$ ;  
        InitTables( $u$ );  
        while  $S_u \neq V$  do  
            begin  $w \in V \setminus S_u$ ;  
                (* All nodes must pick the same node  $w$  *)  
                UpdateTablesSimple( $u, w$ );  
                 $S_u := S_u \cup \{w\}$   
            end  
        end
```

# Simple Algorithm (cntd)

---

```
procedure InitTables (var  $u$ : node)
begin forall  $v \in V$  do
    if  $v = u$  then
        begin  $D_u[v] := 0$ ;  $Nb_u[v] := udef$  end
    else if  $v \in Neigh_u$  then
        begin  $D_u[v] := \omega_{uv}$ ;  $Nb_u[v] := v$  end
    else begin  $D_u[v] := Inf$ ;  $Nb_u[v] := udef$  end
end

procedure UpdateTablesSimple (var  $u, w$  : node)
begin if  $u = w$  then “broadcast the table  $D_w$ ”
    else “receive the table  $D_w$ ”;
    forall  $v \in V$  do
        if  $D_u[w] + D_w[v] < D_u[v]$  then
            begin  $D_u[v] := D_u[w] + D_w[v]$ ;  $Nb_u[v] := Nb_u[w]$ 
            end
end
end
```

# Broadcasting Tables

---

- The set  $S$  is called the set of pivots.
  - Let  $T_w$  be the tree rooted towards  $w$  stored in the arrays  $Nb_u$  at the beginning of iteration that adds pivot  $w$  to  $S$ 
    - $Nb_u[w] = v$  means  $v$  is the father of  $u$  in  $T_w$
  - Let  $u$  be a node whose distance to  $w$  is infinite, i.e.,  $D_u[w] = \text{Inf}$  at the beginning of the iteration in which pivot  $w$  is added to set  $S$ . Then
    - not  $u \in T_w$
    - the routing table  $D_u$  is not changed during that iteration.
- 
- Toueg's algorithm
    - Use  $T_w$  to broadcast  $D_u[w]$
    - This requires that each node knows its sons in  $T_w$

# Toueg's Algorithm

---

```
var     $S_u$  : set of nodes;  
         $D_u$  : array of weights;  
         $Nb_u$  : array of nodes;  
  
begin   $S_u := 0$ ;  
        InitTables( $u$ );  
        while  $S_u \neq V$  do  
            begin  $w : \in V \setminus S_u$ ;  
                MakeTree ( $w$ );  
                if  $D_u[w] < \text{Inf}$  then  
                    (* participate in pivot round *)  
                    UpdateTablesToueg ( $u, w$ );  
                     $S_u := S_u \cup \{w\}$   
                end  
        end  
  
end
```

# Toueg's algorithm (cntd)

---

```
procedure MakeTree (var  $u, w$ , : node)
var  $num\_rec$  : int;
begin forall  $x \in Neigh_u$  do
    if  $Nb_u[w] = x$ 
    then send  $\langle \mathbf{ys}, w \rangle$  to  $x$ ;
    else send  $\langle \mathbf{nys}, w \rangle$  to  $x$ ;
 $num\_rec := 0$ ;
(*  $u$  must receive  $\#Neigh_u$  messages *)
while  $num\_rec < \#Neigh_u$  do
    begin receive  $\langle \mathbf{ys}, w \rangle$  or receive  $\langle \mathbf{nys}, w \rangle$ ;
         $num\_rec := num\_rec + 1$ ;
    end
end
```

# Toueg's algorithm (cntd)

**procedure** *UpdateTablesToueg* (**var**  $u, w, :$  node)

**begin** **if**  $u \neq w$   
**then** receive  $\langle \mathbf{dtab}, w, D \rangle$  from this  $Nb_u[w]$ ;

**else**  $D := D_u$ ;

**forall**  $x \in Neigh_u$  **do**

**if** ( $\langle \mathbf{ys}, w \rangle$  received from  $x$ ) **then**

**send**  $\langle \mathbf{dtab}, w, D \rangle$  to  $x$

**forall**  $v \in V$  **do**

**if**  $D_u[w] + D[v] < D_u[v]$  **then**

**begin**  $D_u[v] := D_u[w] + D_u[v]$ ;

$Nb_u[v] := Nb_u[w]$ ;

**end**

**end**

Broadcast

# Properties of Toueg's Algorithm

---

- Complexity:  $W$  bits for an  $D_u[v]$  and  $Nb_u[v]$ 
  - $O(W)$  bits per  $\langle \mathbf{ys}, w \rangle$  or  $\langle \mathbf{nys}, w \rangle$
  - $O(|V| \times W)$  bits per  $\langle \mathbf{dtab}, w, D \rangle$
  - $O(|E|)$   $\langle \mathbf{ys}, w \rangle + \langle \mathbf{nys}, w \rangle$  messages per iteration
  - $O(|V|)$   $\langle \mathbf{dtab}, w, D \rangle$  messages per iteration
  - $O(|V|)$  iterations
  - $O(|V| \times |E|)$  messages in total
  - $O(|V|^3 \times W)$  bits in total
- Disadvantages
  - Extra buffering required if channels are not FIFO
  - Broadcast is expensive

# So Far

---

- Routing
  - All-pairs Shortest Path
    - Floyd-Warshall
  - Routing Algorithms
    - Toueg's Algorithm
      - Distributed version of Floyd-Warshall
    - Chandy-Misra Algorithm
  - Table Compaction
    - Tree-labeling scheme (Santoro Khatib)
    - Interval labeling scheme
    - Prefix routing

# Alternative distance recurrence

---

---

$$d(u, v) = \begin{cases} 0, & u = v \\ \min\{\omega_{uw} + d(w, v) \mid w \in \text{Neigh}_u\}, & \text{otherwise} \end{cases}$$

## 1. Data locality

- avoids transport of data between remote nodes
- only information from neighbor is needed
- easier to accommodate topological changes

## 2. Destination independence

- only  $d(w, v)$  needed by  $u$  to compute the distance to  $v$ . Hence for each destination a separate computation can be done.

# Chandy-Misra Algorithm

---

```
var     $D_u[v_0]$  : weight  init  $Inf$ ;  
         $Nb_u[v_0]$  : node   init  $undef$ ;
```

For node  $v_0$  only:

```
begin   $D_{v_0} := 0$ ;  
        forall  $w \in Neigh_{v_0}$  do send  $\langle \mathbf{mydist}, v_0, 0 \rangle$  to  $w$   
end
```

Processing a  $\langle \mathbf{mydist}, v_0, d \rangle$  message from neighbor  $w$  by  $u$ :

```
{ $\langle \mathbf{mydist}, v_0, d \rangle \in M_{wu}$ }  
begin  receive  $\langle \mathbf{mydist}, v_0, d \rangle$  from  $w$ ;  
        if  $d + \omega_{uw} < D_u[v_0]$  then  
            begin   $D_u[v_0] := d + \omega_{uw}$ ;  $Nb_u[v_0] := w$ ;  
                    forall  $x \in Neigh_u$  do send  $\langle \mathbf{mydist}, v_0, D_u[v_0] \rangle$  to  $x$ ;  
            end  
end
```

# Properties of Chandy-Misra algorithm

---

- Single destination algorithm
- Diffusing computation
- Full algorithm requires termination detection

# So Far

---

- Routing
  - All-pairs Shortest Path
    - Floyd-Warshall
  - Routing Algorithms
    - Toueg's Algorithm
      - Distributed version of Floyd-Warshall
    - Chandy-Misra Algorithm
  - Table Compaction
    - Tree-labeling scheme (Santoro Khatib)
    - Interval labeling scheme
    - Prefix routing

# Destination-based forwarding

---

(\* A packet with address  $d$  was received or generated at node  $u^*$ )

**if**  $d = u$

**then** deliver packet locally

**else** send the packet to  $table\_lookup_u(d)$

# Arbitrary Routing Table

---

$v$	$Nb_4[v]$
0	3
1	5
2	5
3	3
4	Local
5	7
6	3
7	7
8	3
9	5

$$Neigh_4 = \{3, 5, 7\}$$

$w$	$\{v \mid Nb_4[v] = w\}$
3	0, 3, 6, 8
5	1, 2, 9
7	5, 7

# Cyclic Interval Table

$v$	$Nb_4[v]$
0	3
1	5
2	5
3	5
4	Local
5	7
6	7
7	7
8	7
9	3

$$Neigh_4 = \{3, 5, 7\}$$

$w$	$\{v \mid Nb_4[v] = w\}$	interval
local	4	[4,5)
3	9,0	[9,1)
5	1, 2, 3	[1, 4)
7	5, 6, 7, 8	[5, 9)

# Interval forwarding

---

- $Z_N = \{0, 1, 2, 3, \dots, N - 1\}$
- $[a, b) = \{a, a + 1, \dots, b - 1\}$ , if  $a < b$
- $[a, b) = \{0, \dots, b - 1, a, \dots, N - 1\}$ , if  $a \geq b$
- at each node  $u : Z_N = \{[\alpha_i, \alpha_{i+1}) \mid 0 \leq i \leq \text{deg}(u)\}$

(\* A packet with address  $d$  was received or generated at node  $u$  \*)

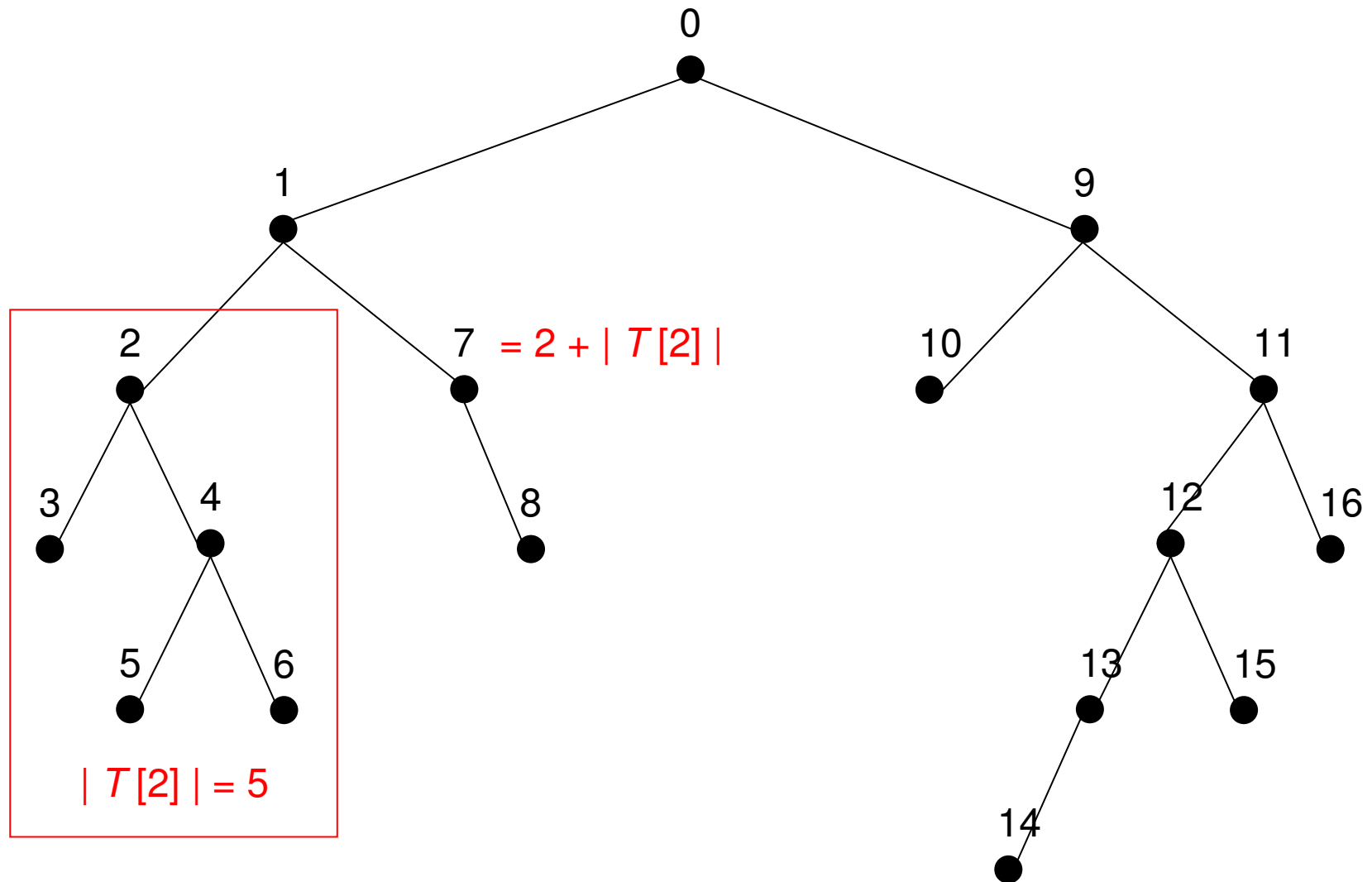
if  $d = l_u$

**then** deliver packet locally

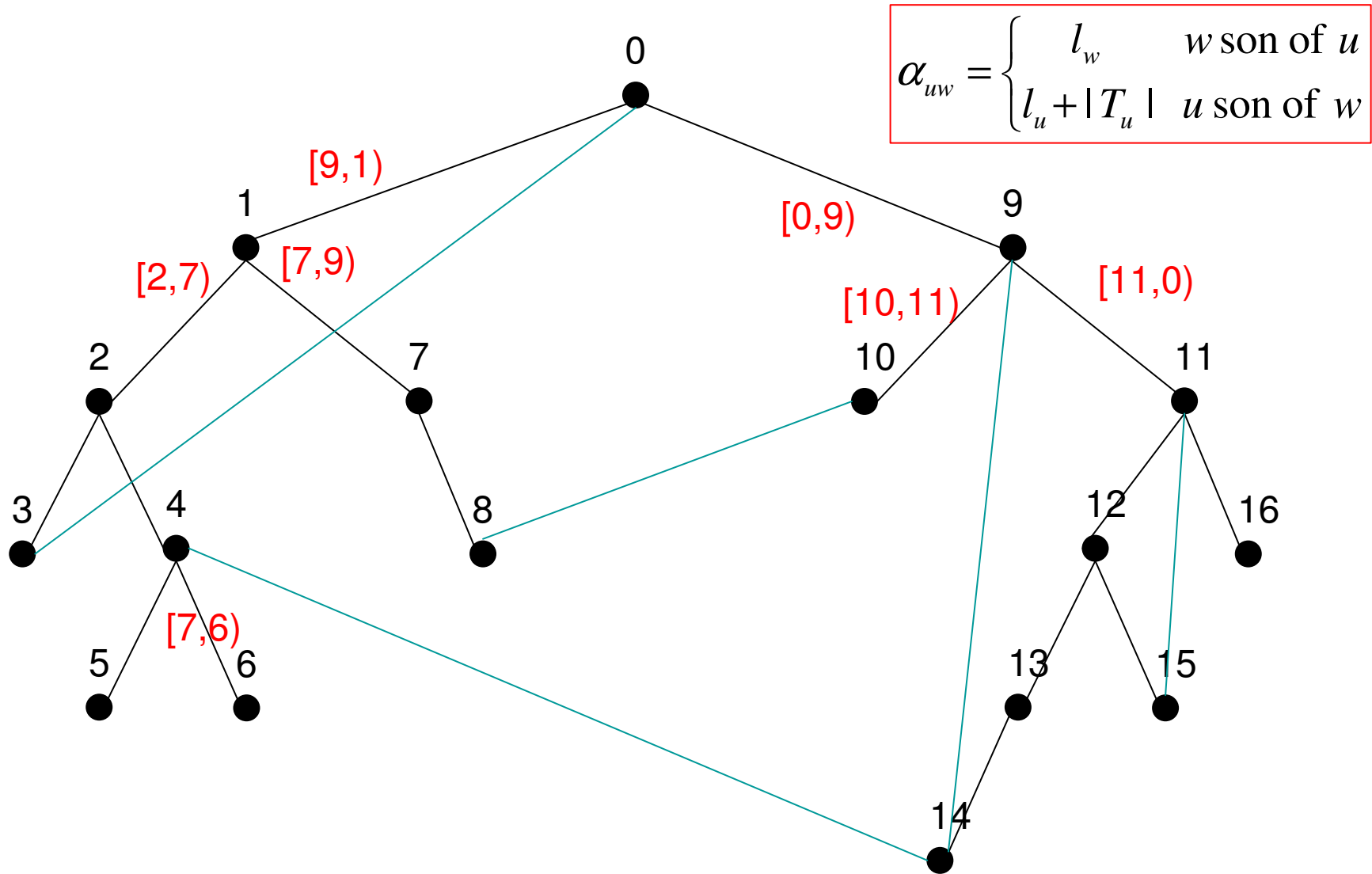
**else begin** select  $\alpha_i$  such that  $d \in [\alpha_i, \alpha_{i+1})$ ;  
send packet via the channel labeled with  $\alpha_i$

**end**

# Preorder tree labeling (Santoro Khatib)



# Tree Labeling Scheme



# Disadvantages of Santoro Khatib

---

- The ratio between the distance in the tree and the distance in network can be **proportional to the size of the network** (ex. a ring)
  - assuming symmetric channel cost, the tree can be chosen such that the distance in the tree is **at most twice the diameter of the network**
  - this scheme is only good if most communication is between **eccentric nodes** (nodes on the boundary of the network)
- Channels that are frond edges are **not used**.
- Tree may become **congested**
- Failure of a single tree edge **partitions the network**

# Interval Labeling

---

An interval labeling scheme (ILS) for a network is

1. An assignment of different labels from  $Z_N$  to the nodes of the network
2. for each node, an assignment of different labels from  $Z_N$  to the channels of that node

Note that (2) implies that every channel is used

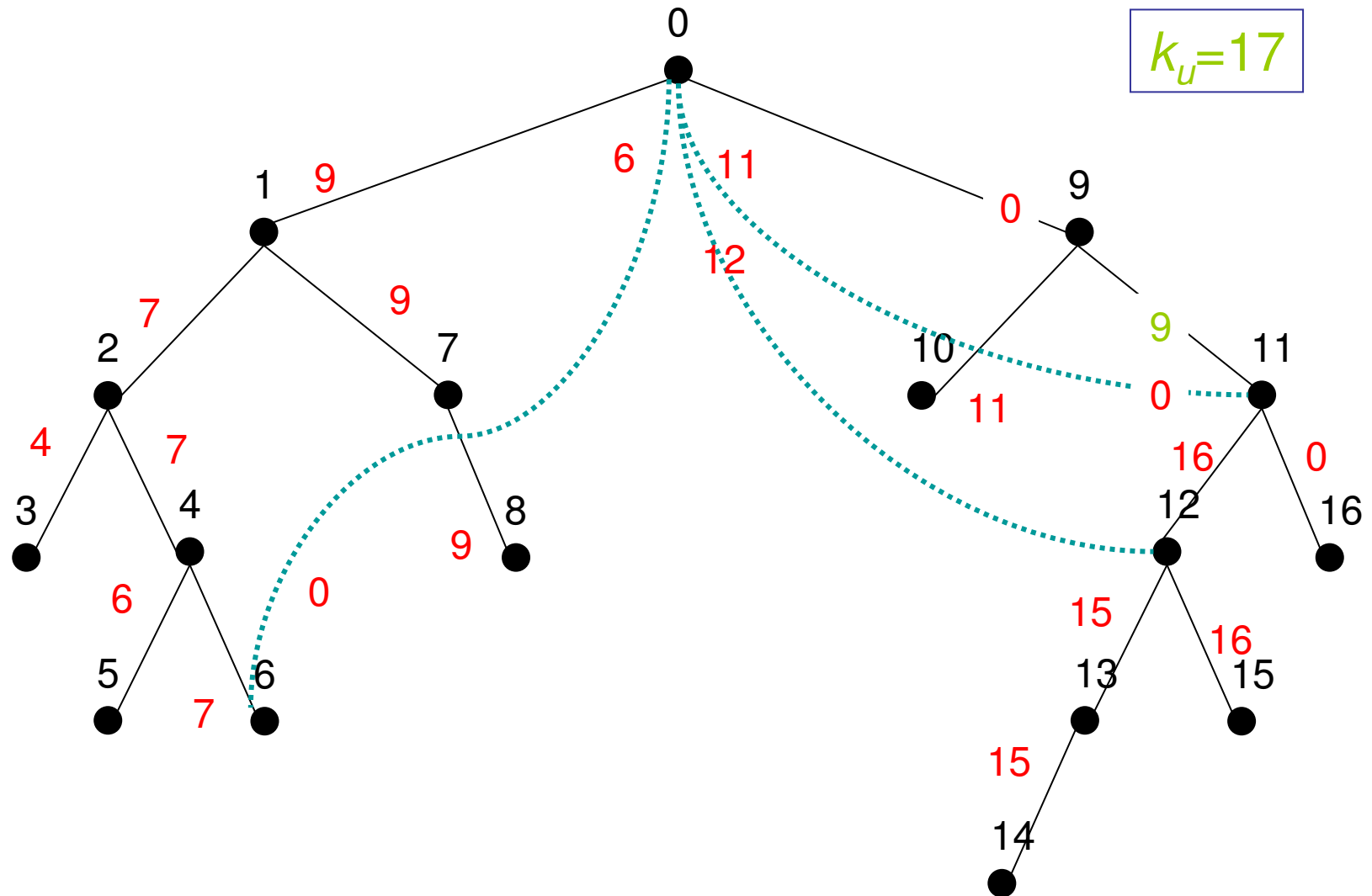
An interval labeling scheme is *valid* if all packets forwarded according to the scheme eventually reach their destination.

# Depth-First-Search ILS

---

- Construct a DFS-tree and label the nodes according the preorder scheme
  - $l_w$  the label of the root of subtree  $T[w]$
  - $k_w = l_w + |T[w]|$
- Let  $\alpha_{uw}$  be the label of edge  $uw$  at node  $u$ 
  - $uw$  a frond edge, then  $\alpha_{uw} = l_w$
  - $w = \text{son}(u)$ , then  $\alpha_{uw} = l_w$
  - $w = \text{father}(u)$  and ( $k_u = N$  and  $u$  has frond edge to the root), then  $\alpha_{uw} = l_w$
  - $w = \text{father}(u)$  and ( $k_u <> N$  or  $u$  has no frond edge to the root), then  $\alpha_{uw} = k_u$

# DFS-ILS (labels to sons not shown)



# Remarks

---

- The DFS-ILS is valid.
- There are networks for which no optimal ILS exists. In particular, there are networks with nodes  $u$  and  $v$  such that all valid ILSs route packets from  $u$  to  $v$  using  $3/2 D_G$  (*i.e.* 1.5 times the diameter of  $G$ ) hops.
- For rings, grids, hypercubes there exist optimal (w.r.t. the minimum-hop metric) ILSs.
- See Tel for a third method of table compaction, viz. prefix routing.

# Conclusion

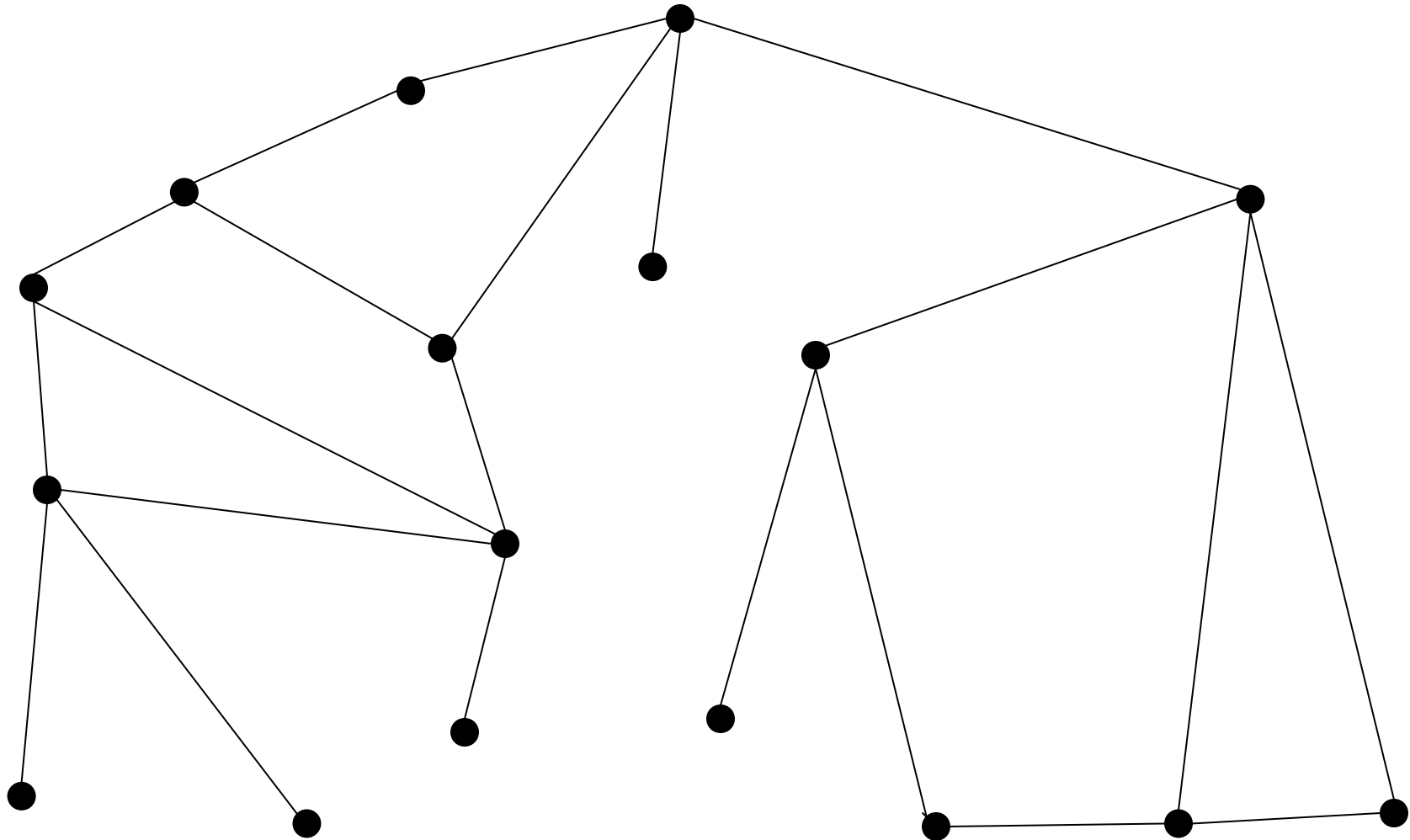
---

- Routing
  - All-pairs Shortest Path
    - Floyd-Warshall
  - Routing Algorithms
    - Toueg's Algorithm
      - Distributed version of Floyd-Warshall
    - Chandy-Misra Algorithm
  - Table Compaction
    - Tree-labeling scheme (Santoro Khatib)
    - Interval labeling scheme
    - Prefix routing

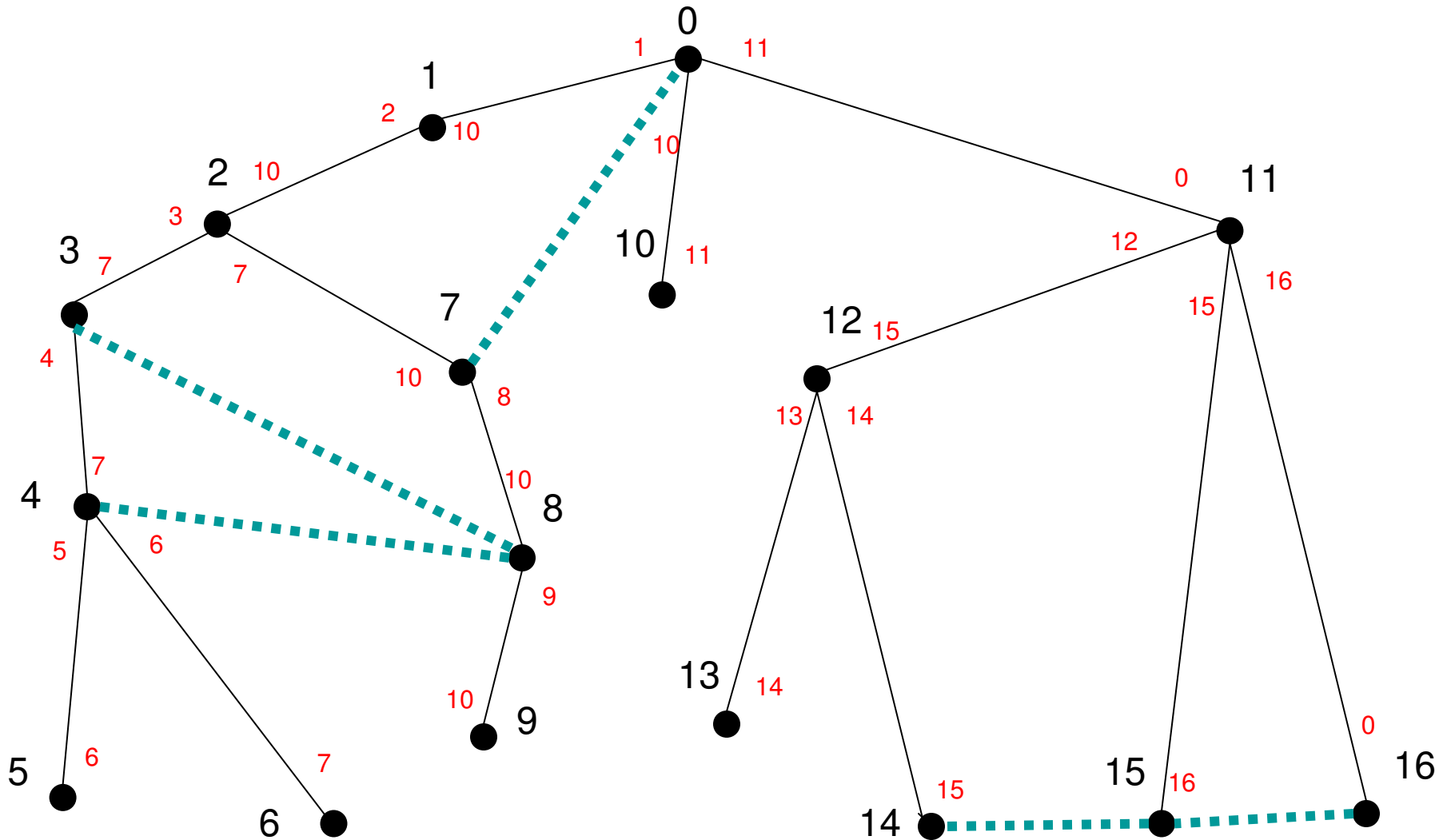
# Graph for Interval Routing

---

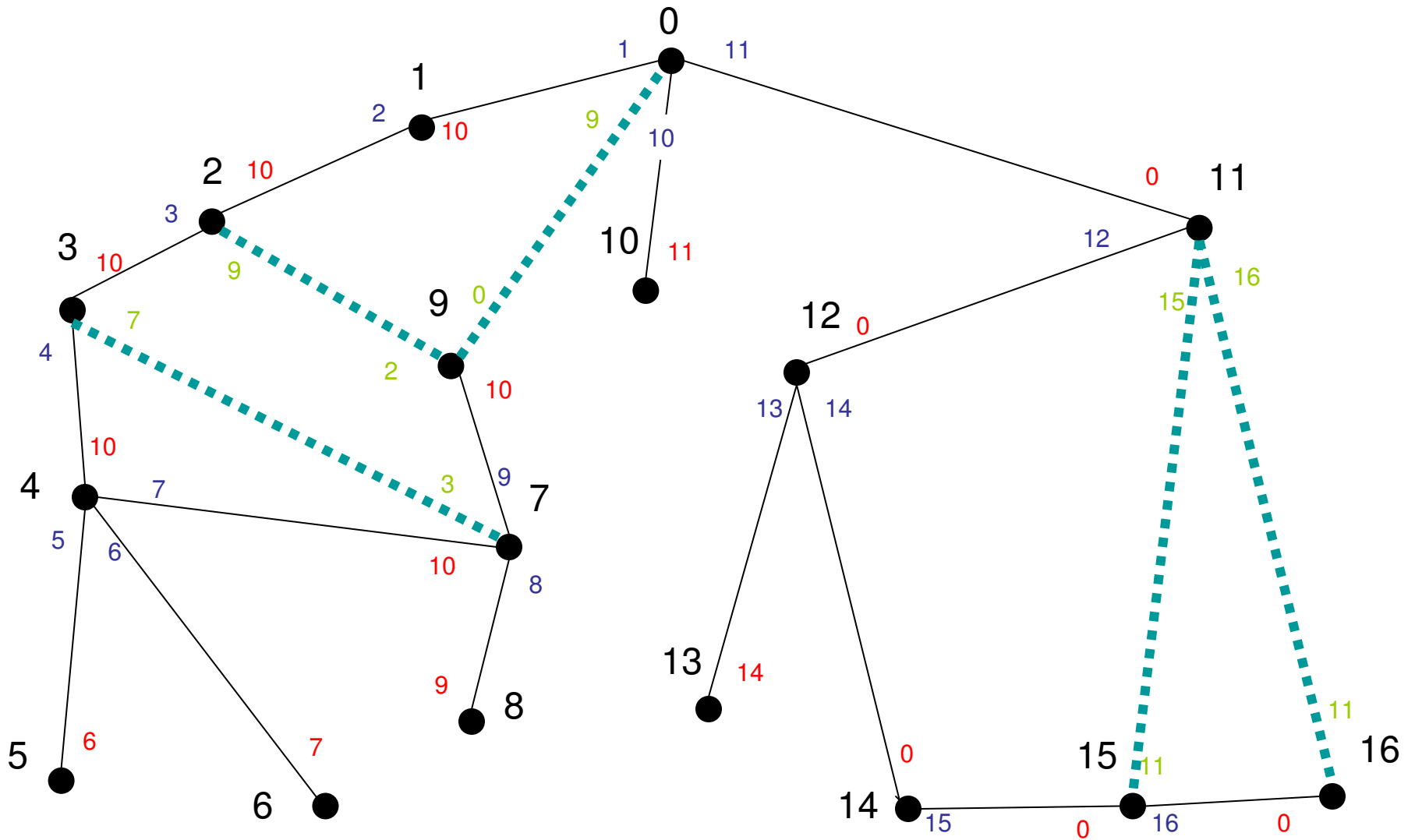
---



# Tree for Santoro Khatib



# Interval Labeling Scheme



# Interval Labeling Scheme

