FACULDADE DE CIÊNCIAS UNIVERSIDADE DO PORTO — U.PORTO — FEUP FACULDADE DE ENGENHARIA UNIVERSIDADE DO PORTO

Network Routing

Redes de Computadores

2021/22

Pedro Brandão

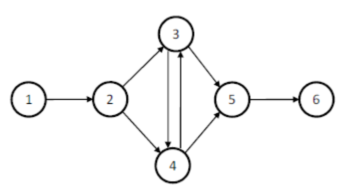
References

- These slides are from "Computer Networking: A Top Down Approach 5th edition. Jim Kurose, Keith Ross Addison-Wesley, April 2009"
- And "Computer Networks, 5th edition. Andrew S. Tanenbaum, David J. Wetherall, Prentice Hall, 2011"
- With adaptations/additions by Manuel Ricardo and Pedro Brandão

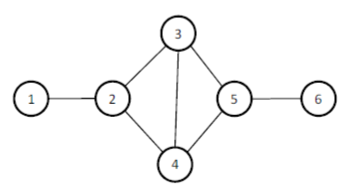
Driving questions…

- *What is a graph?*
- *What is a spanning tree?*
- *What is a shortest path tree?*
- *How are paths defined in a network?*
- *How does the Dijkstra algorithm work?*
- *How does a link state routing protocol work?*
- *How does a node learn about neighbours?*
- *How does the Bellman-Ford algorithm work?*
- *How does a distance vector work?*
- *What are the limitations of the layer 2 network of switches?*
- *How does the IEEE spanning tree protocol work?*
- *What is the maximum capacity of a flow network?*

Graph – Directed and Undirected



a) Directed graph



b) Undirected graph

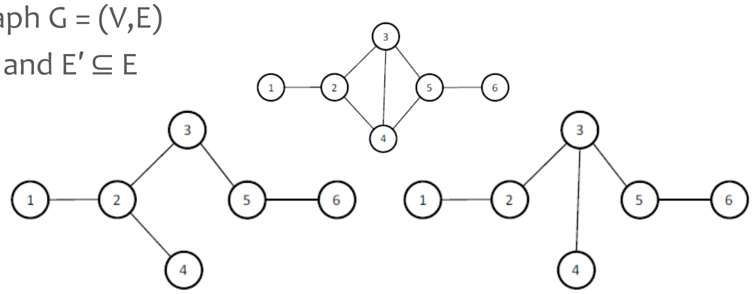
$$\begin{aligned} G =& \ (V,E) & \\ V =& \ \{v_1,v_2,v_3,v_4,v_5,v_6\}, & |V| = 6 \\ E =& \ \{(v_1,v_2),(v_2,v_3),(v_2,v_4),(v_3,v_4), & \\ & \ (v_4,v_3),(v_3,v_5),(v_4,v_5),(v_5,v_6)\}, & |E| = 8 \end{aligned}$$

$$\begin{aligned} G =& \ (V,E) & \\ V =& \ \{v_1,v_2,v_3,v_4,v_5,v_6\}, & |V| = 6 \\ E =& \ \{(v_1,v_2),(v_2,v_3),(v_2,v_4),(v_3,v_4), & \\ & \ (v_3,v_5),(v_4,v_5),(v_5,v_6)\}, & |E| = 7 \end{aligned}$$

RCom 21/22 - Routing

Tree

- Trees T = (V,E)
  - graph with **no cycles**
  - number of edges |E| = |V |−1
  - any two vertices of the tree are connected by exactly **one** path
- A tree T is said to span a graph (spanning tree) if:
  - Given graph G = (V,E)
  - T = (V,E’) and E’ ⊆ E

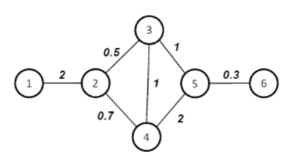


5

RCom 21/22 - Routing

Shortest Path Trees

- Graphs and Trees can be weighted
  - $G = (V, E, w)$
  - $T = (V, E', w)$
  - $w: E \rightarrow \mathbb{R}$



- Total cost of a tree T ➔ $C_{total}(T) = \sum_{i=i}^{|E|} w(e_i)$
- Minimum Spanning Tree T^* ➔ $C_{total}(T^*) = \min(C_{total}(T))$
  - algorithms used to compute MST: Prism, Kruskal
- Shortest Path Tree (SPT) Rooted at Vertex s
  - tree composed by the union of the shortest paths between s and each of other vertices of G
  - Algorithms used to compute SPT: Dijkstra, Bellman-Ford
- Computer networks use Shortest Path Trees

6

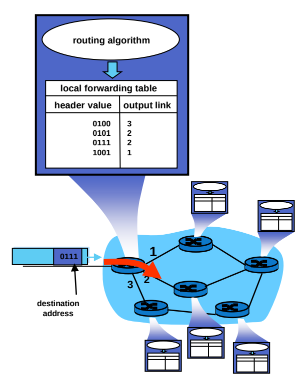
FACULDADE DE CIÊNCIAS UNIVERSIDADE DO PORTO | U.PORTO | U.PORTO FEUP FACULDADE DE ENGENHARIA UNIVERSIDADE DO PORTO

Routing in Layer 3 Networks

(Routing)

Forwarding, Routing

- Forwarding ➔ data plane
  - directing packet from input to output link
  - using a forwarding table
- Routing ➔ control plane
  - computing paths the packets will follow
  - routers exchange messages
  - each router creates its forwarding table



RCom 21/22 - Routing

U.PORTO

Importance of Routing

- End-to-end performance
  - path affects quality of service
  - delay, throughput, packet loss
- Use of network resources
  - balance traffic over routers and links
  - avoiding congestion by directing traffic to less-loaded links
- Transient disruptions
  - failures, maintenance
  - limiting packet loss and delay during changes

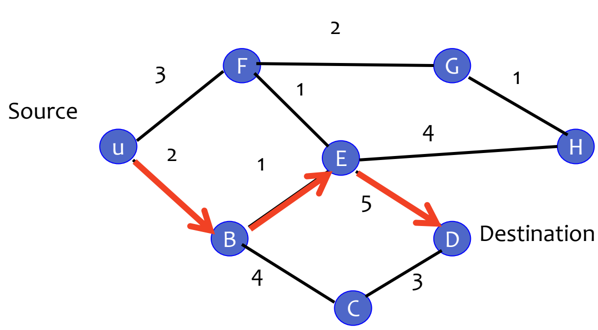
9

RCom 21/22 - Routing

U.PORTO

Shortest-Path Routing

- Path-selection model
  - Destination-based
  - Load-insensitive (e.g., static link weights)
  - Minimum hop count or minimum sum of link weights

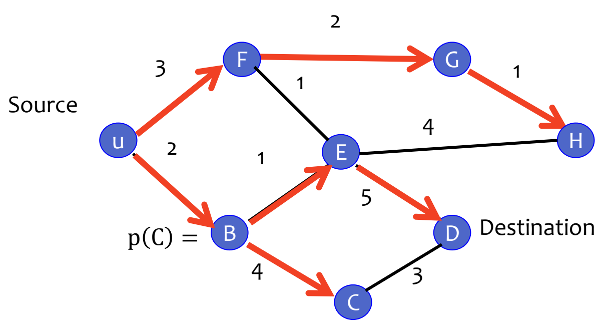


10

Shortest-Path Routing

- Given a network topology with link costs
  - c(x, y) - link cost from node x to node y
  - Infinity (∞) if x and y are not direct neighbours

- Compute the least-cost paths from source **u** to all nodes
  - p(v) - node predecessor of node v in the path to u



Dijkstra’s Shortest-Path Algorithm

- Iterative algorithm
  - After k iterations ➔ known least-cost paths to k nodes

- **S** ➔ set of nodes for which least-cost path is known
  - Initially, **`S={u}`**, where **u** is the source node
  - Add one node to **S** in each iteration

- **`D(v)`**➔ current cost of path from source to node **v**
  - Initially
    - **`D(v)=c(u, v)`** for all nodes **v** adjacent to **u**
    - **`D(v)=`** ∞ for all other nodes **v**
  - Continually update **`D(v)`** when shorter paths are learned

RCom 21/22 - Routing

Dijsktra's Algorithm

```
Initialization:
  S = {u}
  for all nodes v
   if v adjacent to u {
      D(v) = c(u,v)
   }
   else D(v) = ∞

 Loop
   find node w not in S with the smallest D(w)
      add w to S
      update D(v) for all v adjacent to w and not in S:
          D(v) = min{D(v), D(w) + c(w,v)}
  until all nodes in S
```

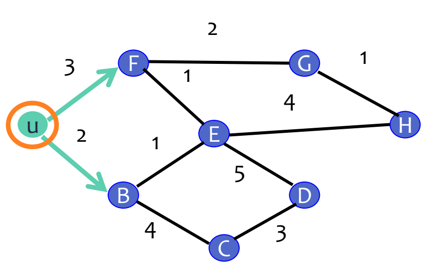
13

RCom 21/22 - Routing

Dijkstra's Algorithm – Example – 1

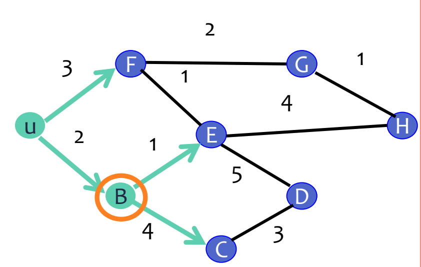
S={u}

	D(S)
u	0
B	2
C	∞
D	∞
E	∞
F	3
G	∞
H	∞



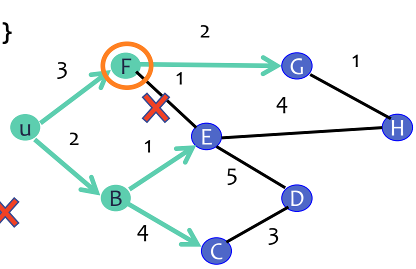
S={u,B}

	D(S)
u	0
B	2
C	6
D	∞
E	3
F	3
G	∞
H	∞



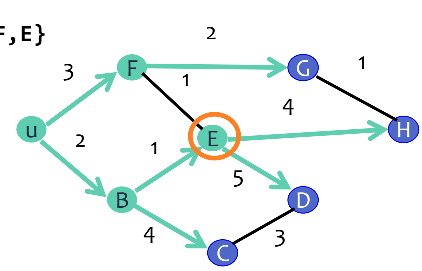
S={u,B,F}

	D(S)
u	0
B	2
C	6
D	∞
E	3
F	3
G	5
H	∞



S={u,B,F,E}

	D(S)
u	0
B	2
C	6
D	8
E	3
F	3
G	5
H	7

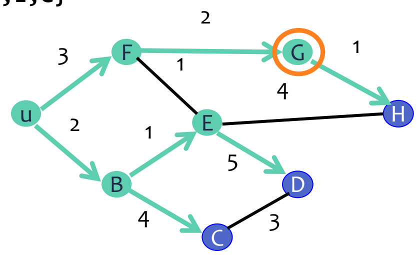


14

Dijkstra’s Algorithm – Example – 2

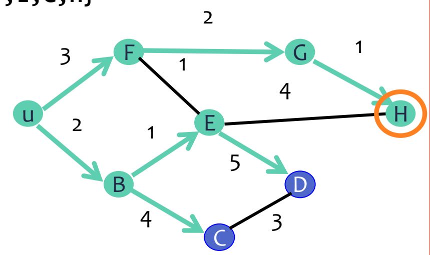
S={u,B,F,E,G}

	D(S)
u	0
B	2
C	6
D	8
E	3
F	3
G	5
H	6



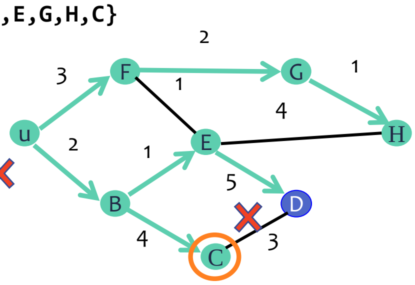
S={u,B,F,E,G,H}

	D(S)
u	0
B	2
C	6
D	8
E	3
F	3
G	5
H	6



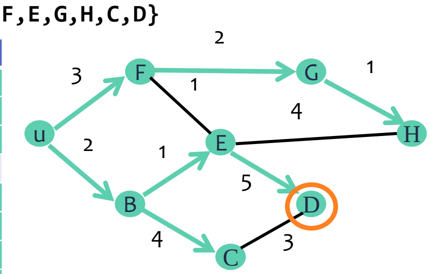
S={u,B,F,E,G,H,C}

	D(S)
u	0
B	2
C	6
D	8
E	3
F	3
G	5
H	6



S={u,B,F,E,G,H,C,D}

	D(S)
u	0
B	2
C	6
D	8
E	3
F	3
G	5
H	6

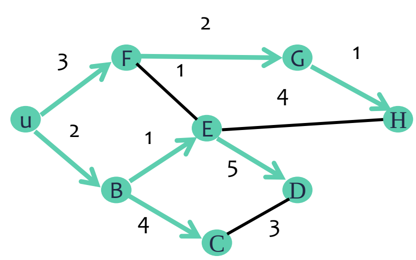


15

Dijkstra’s Algorithm – Example – 3

	D(S)	Next Hop
u	0	---
B	2	B
C	6	B
D	8	B
E	3	B
F	3	F
G	5	F
H	6	G

Forwarding table at u



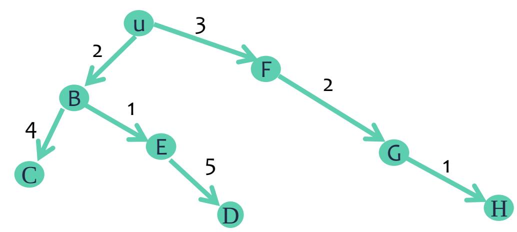
Shortest-path tree from u

16

Dijkstra's Algorithm – Example – 4

	D(S)	Next Hop
u	0	---
B	2	B
C	6	B
D	8	B
E	3	B
F	3	F
G	5	F
H	6	G

Forwarding table at u



Shortest-path tree from u

Link-State Routing

- Each router keeps track of its incident links
  - link up, link down
  - cost on the link
- Each router broadcasts link state
  - every router gets a complete view of the graph
- Each router runs Dijkstra's algorithm, to
  - compute the shortest paths
  - construct the forwarding table
- Example protocols
  - Open Shortest Path First (OSPF), STD 54, RFC 2328, OSPF Version 2
  - Intermediate System – Intermediate System (IS-IS)
    - ISO/IEC 10589:2002, Information technology — Telecommunications and information exchange between systems — Intermediate System to Intermediate System intra-domain routeing information exchange protocol for use in conjunction with the protocol for providing the connectionless-mode network service (ISO 8473)
    - RFC 1195, Use of OSI IS-IS for routing in TCP/IP and dual environments

RCom 21/22 - Routing

Detection of Topology Changes

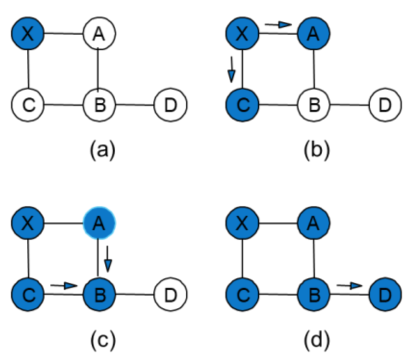
- Beacons generated by routers on links
  - Periodic “hello” messages in both directions
  - few missed “hellos” ➔ link failure

19

RCom 21/22 - Routing

Broadcasting the Link State

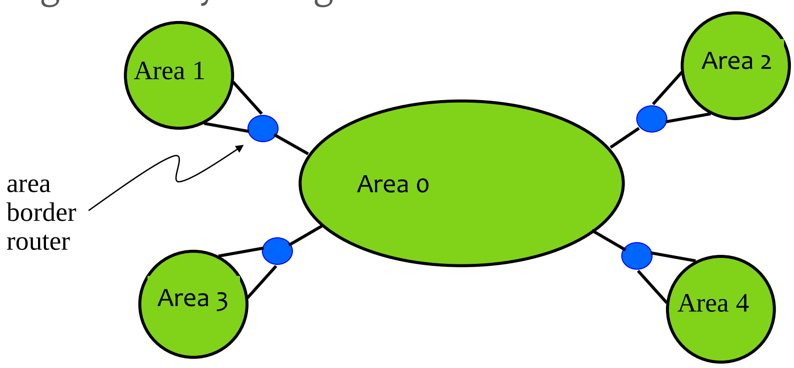
- How to Flood the link state?
  - every node sends link-state information through adjacent links
  - next nodes forward that info to all links
    - except the one where the information arrived
- When to initiate flooding?
  - Topology change
    - link or node failure/recovery
    - link cost change
  - Periodically
    - refresh link-state information
    - typically 30 minutes



20

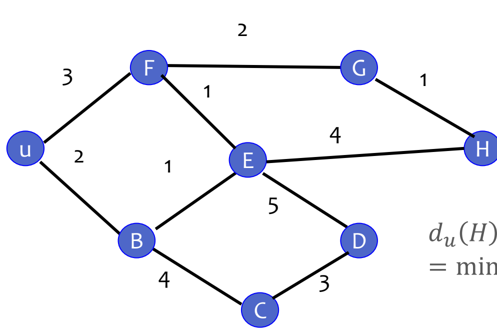
Scaling Link-State Routing

- Overhead of link-state routing
  - flooding link-state packets throughout the network
  - running Dijkstra's shortest-path algorithm
- Introducing hierarchy through "areas"



Bellman-Ford Algorithm

- Define distances at each node x
  - $d_x(y)$ = cost of least-cost path from x to y
- Update distances based on neighbours
  - $d_x(y) = \min\big(c(x,n) + d_n(y)\big)$ over all neighbours n of x



$d_u(H)$
$= \min\big(c(u,F) + d_F(H), c(u,B) + d_B(H)\big)$

Distance Vector Algorithm

- $c(x,n)$ = cost for direct link from x to n
  - node x maintains costs of direct links $c(x,n)$
  - n are x's neighbours
- $D_x(y)$ = estimate of least cost from x to y
  - node x maintains **distance vector** $\mathbf{D_x} = [D_x(y): y \in V]$
  - V is all nodes
- Node x maintains also its neighbours' distance vectors
  - for each neighbour n, x maintains $\mathbf{D_n} = [D_n(y): y \in V]$
- Each node x periodically sends D_x to its neighbours
  - and neighbours update their own distance vectors
  - $D_z(y) \leftarrow \min_x\{c(z,x) + D_x(y)\}$ for each node $y \in V$
    - z is a neighbour of x
- Over time, the distance vector D_x converges

Distance Vector Algorithm

- Iterative, asynchronous
  - each local iteration caused by:
    - local link cost change
    - distance vector update message from neighbour
- Distributed
  - node notifies neighbours
    - only when its DV changes
- Neighbours then
  - notify their neighbours,
  - if necessary

- Each node:

wait for (change in local link cost or message from neighbour)

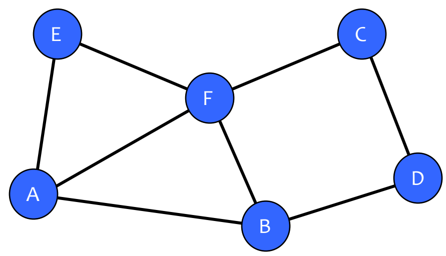
recompute estimates

if DV to any destination has changed, *notify* neighbours

Distance Vector Example - Step 0

Table for A			Table for B		
Dst	Cst	Hop	Dst	Cst	Hop
A	0	A	A	4	A
B	4	B	B	0	B
C	∞	–	C	∞	–
D	∞	–	D	3	D
E	2	E	E	∞	–
F	6	F	F	1	F

Table for C			Table for D			Table for E			Table for F		
Dst	Cst	Hop	Dst	Cst	Hop	Dst	Cst	Hop	Dst	Cst	Hop
A	∞	–	A	∞	–	A	2	A	A	6	A
B	∞	–	B	3	B	B	∞	–	B	1	B
C	0	C	C	1	C	C	∞	–	C	1	C
D	1	D	D	0	D	D	∞	–	D	∞	–
E	∞	–	E	∞	–	E	0	E	E	3	E
F	1	F	F	∞	–	F	3	F	F	0	F



U.PORTO

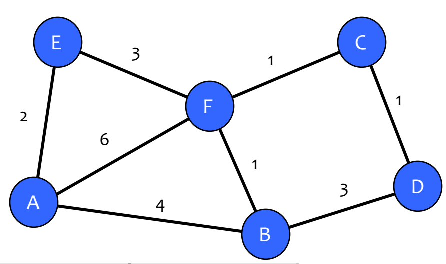
25

RCom 21/22 - Routing

Distance Vector Example - Step 1

Table for A			Table for B		
Dst	Cst	Hop	Dst	Cst	Hop
A	0	A	A	4	A
B	4	B	B	0	B
C	7	F	C	2	F
D	7	B	D	3	D
E	2	E	E	4	F
F	5	E	F	1	F

Table for C			Table for D			Table for E			Table for F		
Dst	Cst	Hop	Dst	Cst	Hop	Dst	Cst	Hop	Dst	Cst	Hop
A	7	F	A	7	B	A	2	A	A	5	B
B	2	F	B	3	B	B	4	F	B	1	B
C	0	C	C	1	C	C	4	F	C	1	C
D	1	D	D	0	D	D	∞	–	D	2	C
E	4	F	E	∞	–	E	0	E	E	3	E
F	1	F	F	2	C	F	3	F	F	0	F



U.PORTO

26

RCom 21/22 - Routing

U.PORTO

Distance Vector Example - Step 2

Table for A

Dst	Cst	Hop
A	0	A
B	4	B
C	6	E
D	7	B
E	2	E
F	5	E

Table for B

Dst	Cst	Hop
A	4	A
B	0	B
C	2	F
D	3	D
E	4	F
F	1	F

Table for C

Dst	Cst	Hop
A	6	F
B	2	F
C	0	C
D	1	D
E	4	F
F	1	F

Table for D

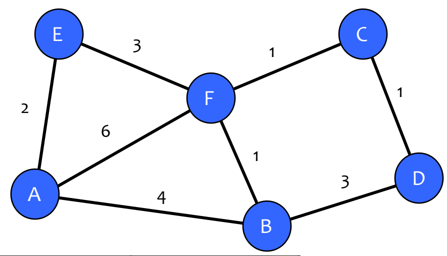
Dst	Cst	Hop
A	7	B
B	3	B
C	1	C
D	0	D
E	5	C
F	2	C

Table for E

Dst	Cst	Hop
A	2	A
B	4	F
C	4	F
D	5	F
E	0	E
F	3	F

Table for F

Dst	Cst	Hop
A	5	B
B	1	B
C	1	C
D	2	C
E	3	E
F	0	F



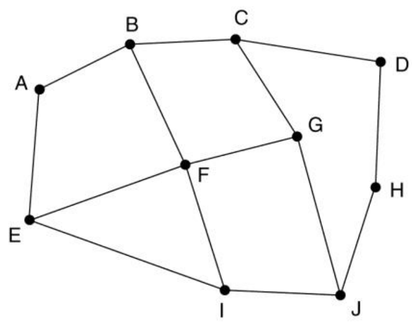
RCom 21/22 - Routing

U.PORTO

Routing Information Protocol (RIP)

- STD 56, RFC 2453, RIP Version 2
- Distance vector protocol
  - nodes send distance vectors every 30 seconds
  - or, when an update causes a change in routing
- RIP is limited to small networks

BGP – The Exterior Gateway Routing Protocol



(a)
A set of BGP routers.

Information F receives
from its neighbors about D

From B: "I use BCD"
From G: "I use GCD"
From I: "I use IFGCD"
From E: "I use EFGCD"

(b)
Information sent to F

FACULDADE DE CIÊNCIAS
UNIVERSIDADE DO PORTO

U.PORTO

U.PORTO
FEUP FACULDADE DE ENGENHARIA
UNIVERSIDADE DO PORTO

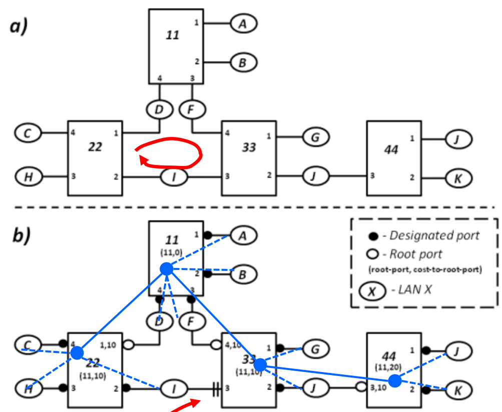
Unique Spanning Tree in Ethernet Networks

Routing

RCom 21/22 - Routing

L2 Networking - Single Tree Required

- Ethernet frame
  - No hop-count
  - Could loop forever
  - broadcast frame, mis-configuration
- Layer 2 network
  - Required to have tree topology
  - Single path between every pair of stations
- Spanning Tree Protocol (STP)
  - Running in bridges
  - Helps building the spanning tree
  - Blocks ports

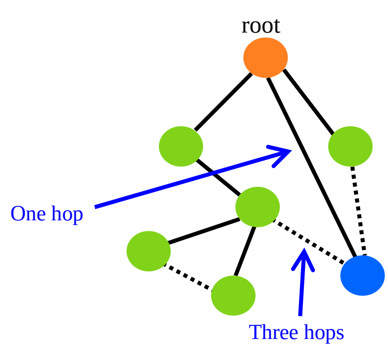


31

RCom 21/22 - Routing

Constructing a Spanning Tree

- Distributed algorithm
  - switches need to elect a "root"
    - the switch with the smallest identifier
  - each switch identifies if its interface is on the shortest path from the root
  - messages (Y, d, X)
    - from node X
    - claiming Y is the root
    - and the distance is d



32

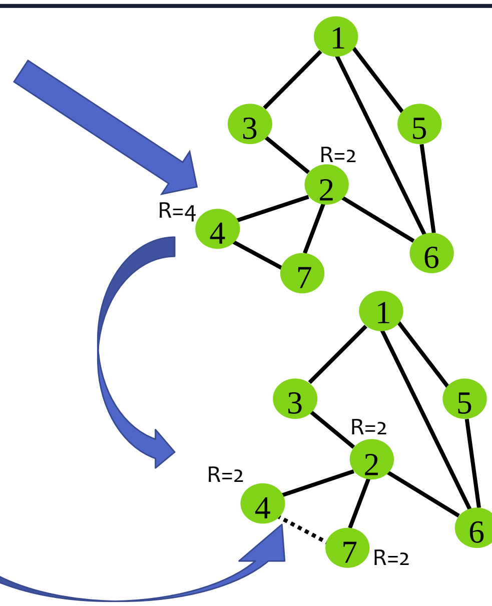
Steps in Spanning Tree Algorithm

- Initially, each switch thinks it is the root
  - switch sends a message out every interface
  - identifying itself as the root with distance `0`
  - example: switch X announces (X, `0`, X)
- Other switches update their view of the root
  - upon receiving a message, check the root id
  - if the new id is smaller, start viewing that switch as the root
- Switches compute their distance from the root
  - add 1 to the distance received from a neighbour
  - identify interfaces not on a shortest path to the root and exclude them from the spanning tree

33

Example - Switch #4's Viewpoint

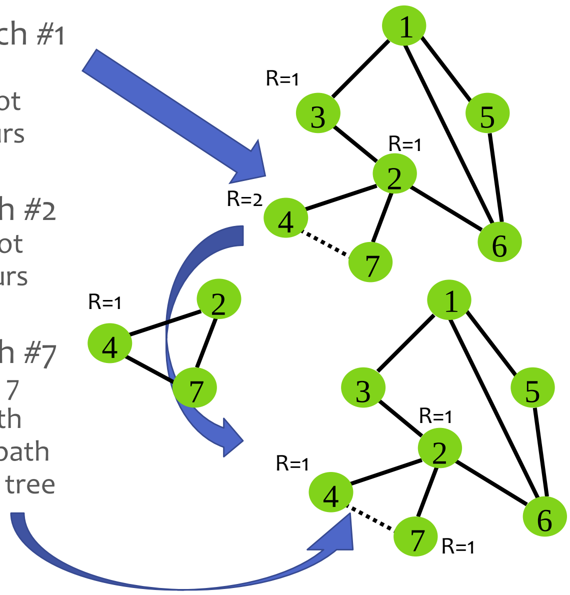
- Switch #4 thinks it is the root
  - sends (4, 0, 4) message to 2 and 7
- Then, switch #4 hears from #2
  - receives (2, 0, 2) message from 2
  - … and thinks that #2 is the root
  - and realizes it is just one hop away
- Then, switch #4 hears from #7
  - receives (2, 1, 7) from 7
  - and realizes this is a longer path
  - so, prefers its own one-hop path
  - and removes 4-7 link from the tree



34

Example - Switch #4's Viewpoint

- Switch #2 hears about switch #1
  - switch 2 hears (1, 1, 3) from 3
  - switch 2 starts treating 1 as root
  - and sends (1, 2, 2) to neighbours
- Switch #4 hears from switch #2
  - switch 4 starts treating 1 as root
  - and sends (1, 3, 4) to neighbours
- Switch #4 hears from switch #7
  - switch 4 receives (1, 3, 7) from 7
  - and realizes this is a longer path
  - so, prefers its own three-hop path
  - and removes 4-7 link from the tree



RCom 21/22 - Routing

35

FACULDADE DE CIÊNCIAS UNIVERSIDADE DO PORTO

U.PORTO

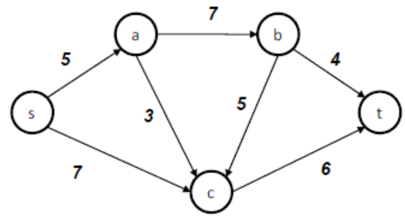
FEUP FACULDADE DE ENGENHARIA UNIVERSIDADE DO PORTO

Maximum Flow of a Network

Routing

Flow Network Model

- Flow network
  - source s
  - sink t
  - nodes a, b and c
- Edges are labelled with capacities
  - (e.g. bit/s)



- Communication networks are not flow networks
  - they are queue networks
  - flow networks enable to determine limit capacity values

Maximum Capacity of a Flow Network

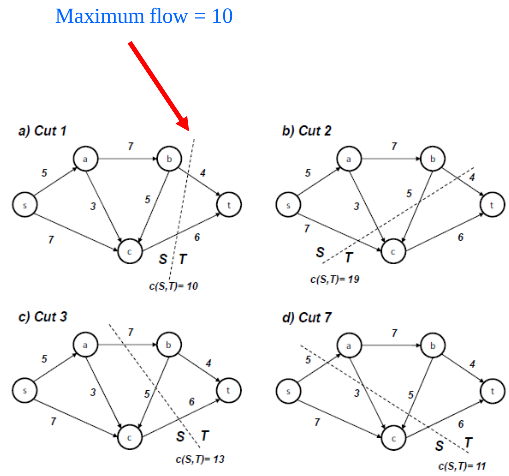
- Max-flow min-cut theorem
  - maximum amount of flow transferable through a network
  - equals minimum value among all simple cuts of the network
- Cut ➔ split of the nodes V into two disjoint sets S and T
  - $S \cup T = V$
  - there are $2^{|V|-2}$ possible cuts
- Capacity of cut (S, T): $c(S,T) = \sum_{(u,v)|u \in S, v \in T, (u,v) \in E} c(u,v)$

RCom 21/22 - Routing

Max-flow Min-cut - Example

$2^{|5|-2} = 8$ possible cuts

Cut	Vertices s	a	b	c	t	$c(S,T)$	Feasability
1	S	S	S	S	T	10	✓
2	S	S	S	T	T	19	✓
3	S	S	T	S	T	13	✓
4	S	S	T	T	T	17	✓
5	S	T	S	S	T	-	×
6	S	T	S	T	T	-	×
7	S	T	T	S	T	11	✓
8	S	T	T	T	T	12	✓



39

RCom 21/22 - Routing

Homework

1. Review slides

2. Read from Tanenbaum
   - Section 5.2 – Routing algorithms
   - Section 4.8.3 Spanning Tree Bridges

3. Answer questions at Moodle

40

FACULDADE DE CIÊNCIAS
UNIVERSIDADE DO PORTO

U.PORTO

U.PORTO
FEUP FACULDADE DE ENGENHARIA
UNIVERSIDADE DO PORTO

End of Routing