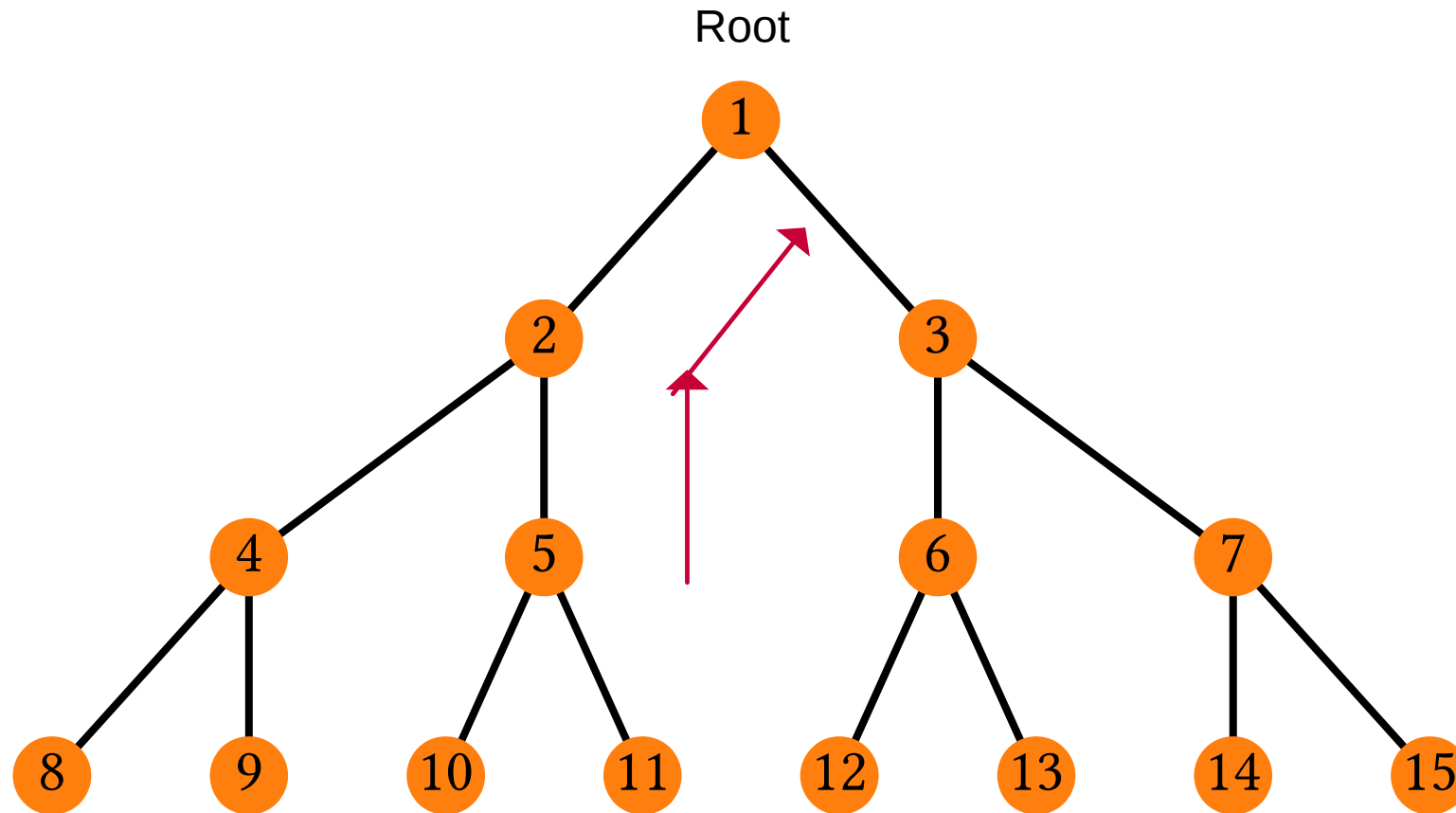


# Source Routing for Downward Data Traffic

WSN Lab final project, 2017-2018

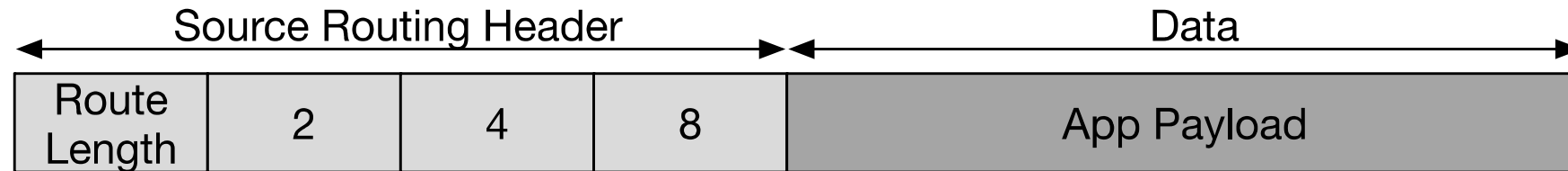
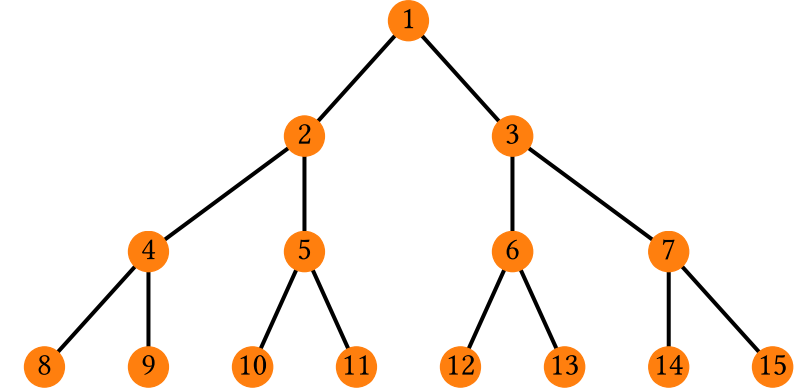
# Upward traffic



Many-to-one, up the tree

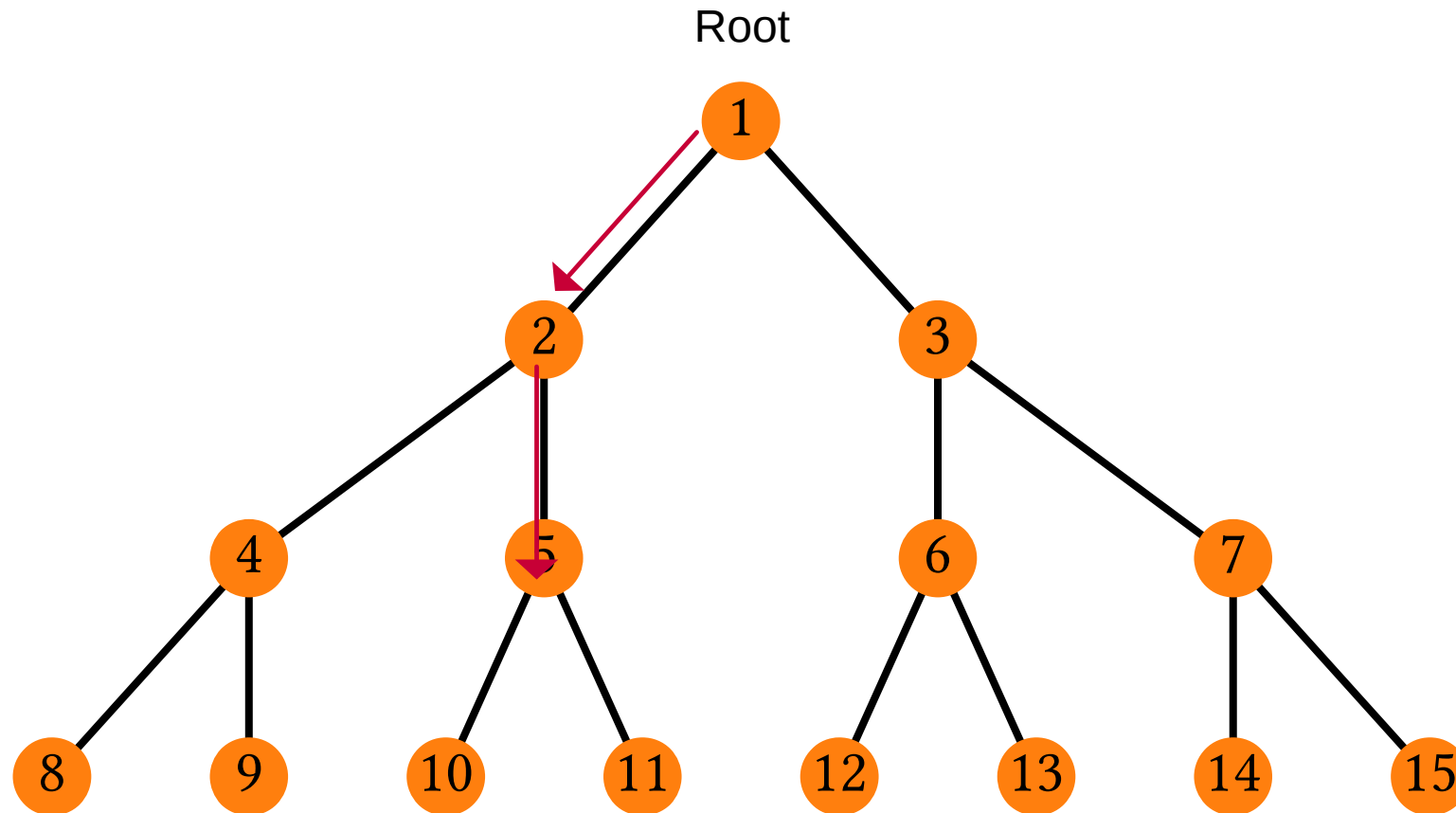
# Source Routing – overview

- Source specifies the entire packet route: complete path from source to destination
- Computed by the root node only



- Intermediate nodes just forward to specific next hop:  
#2 would look at path in header and forward to #4, and so on until the packet reaches the destination node **D = 8**

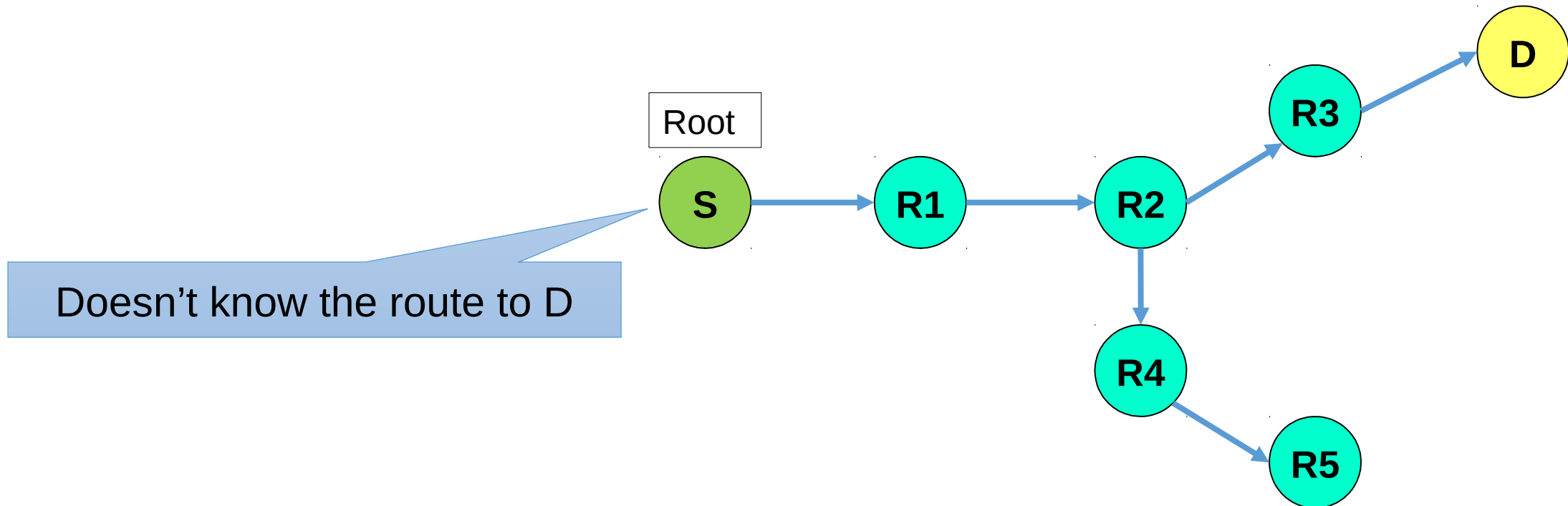
# Downwards Source Routing – overview



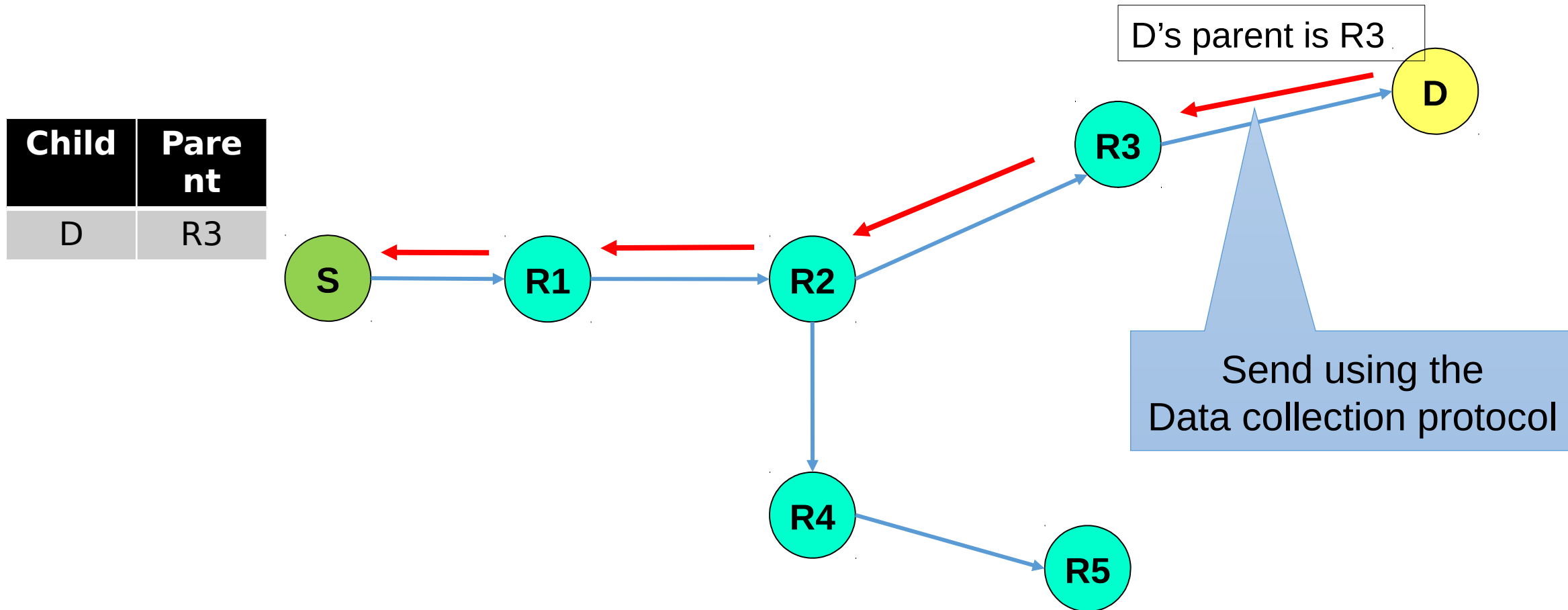
Enable the sink to send data down the tree

# To begin with:

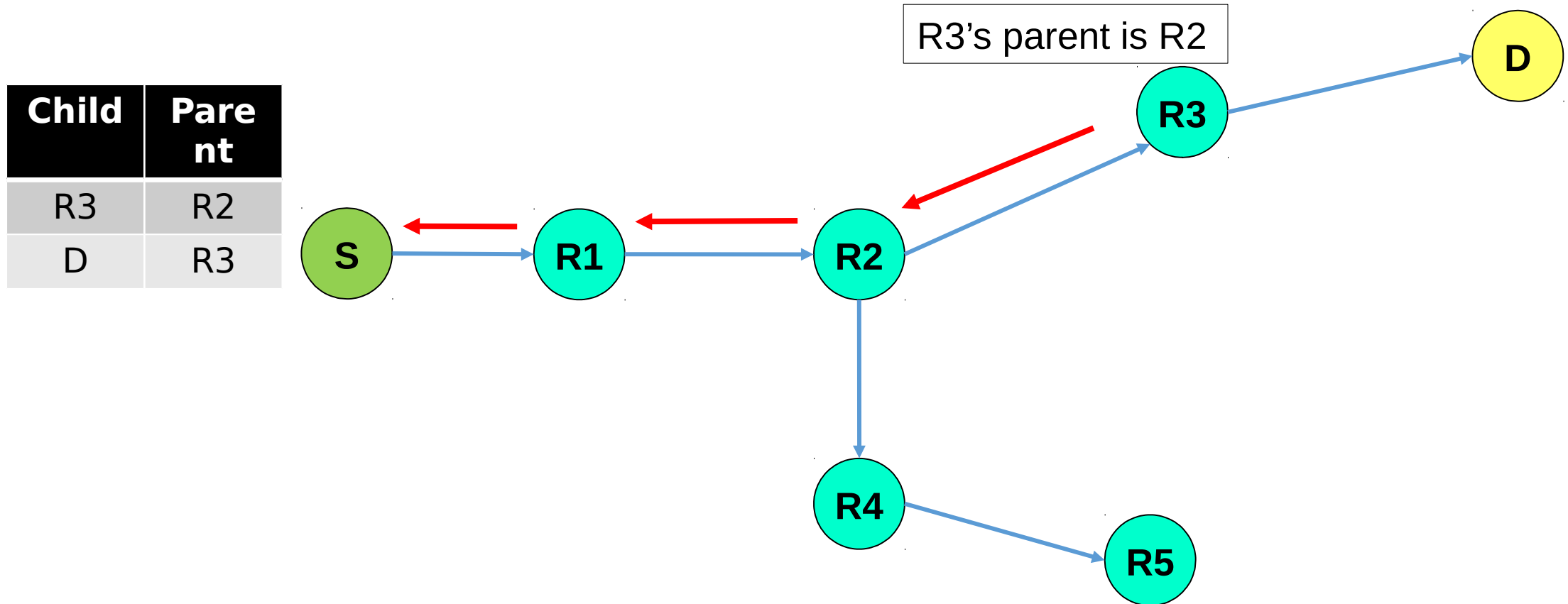
- Objective: Node S wants to send data to node D



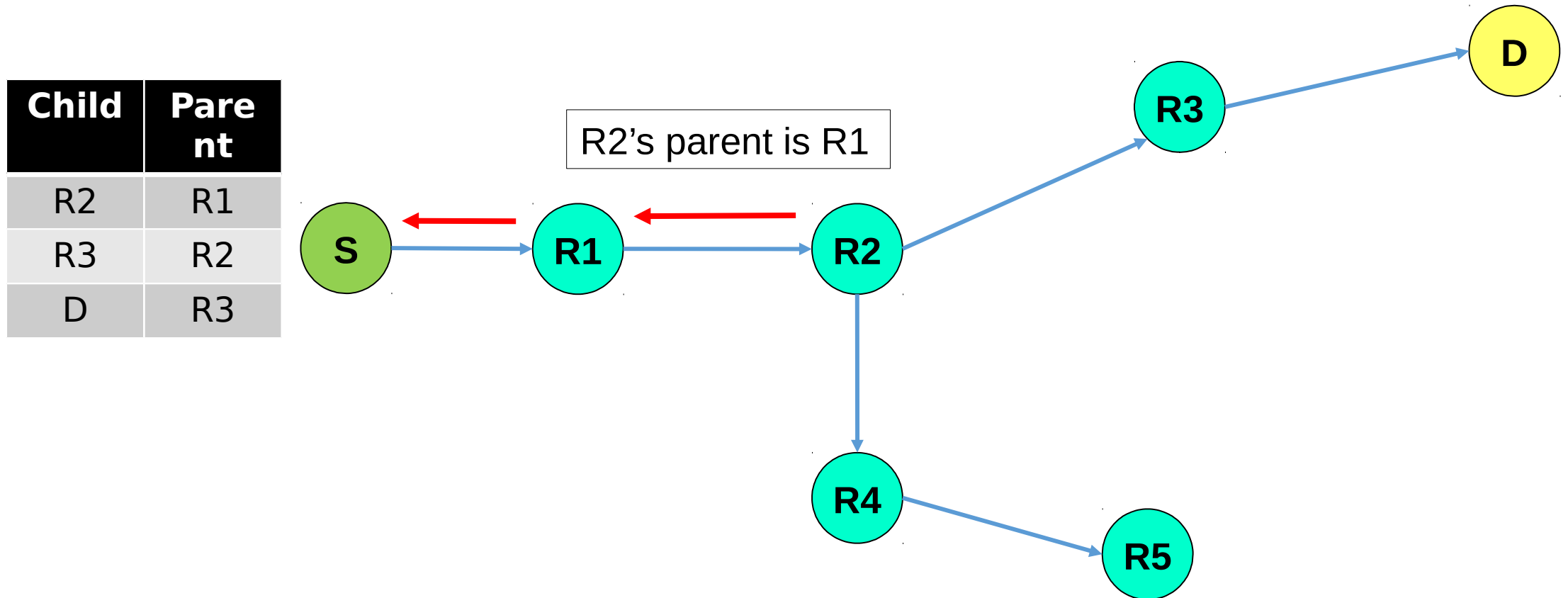
# Phase 1: Collecting the routing information



# Phase 1: Collecting the routing information



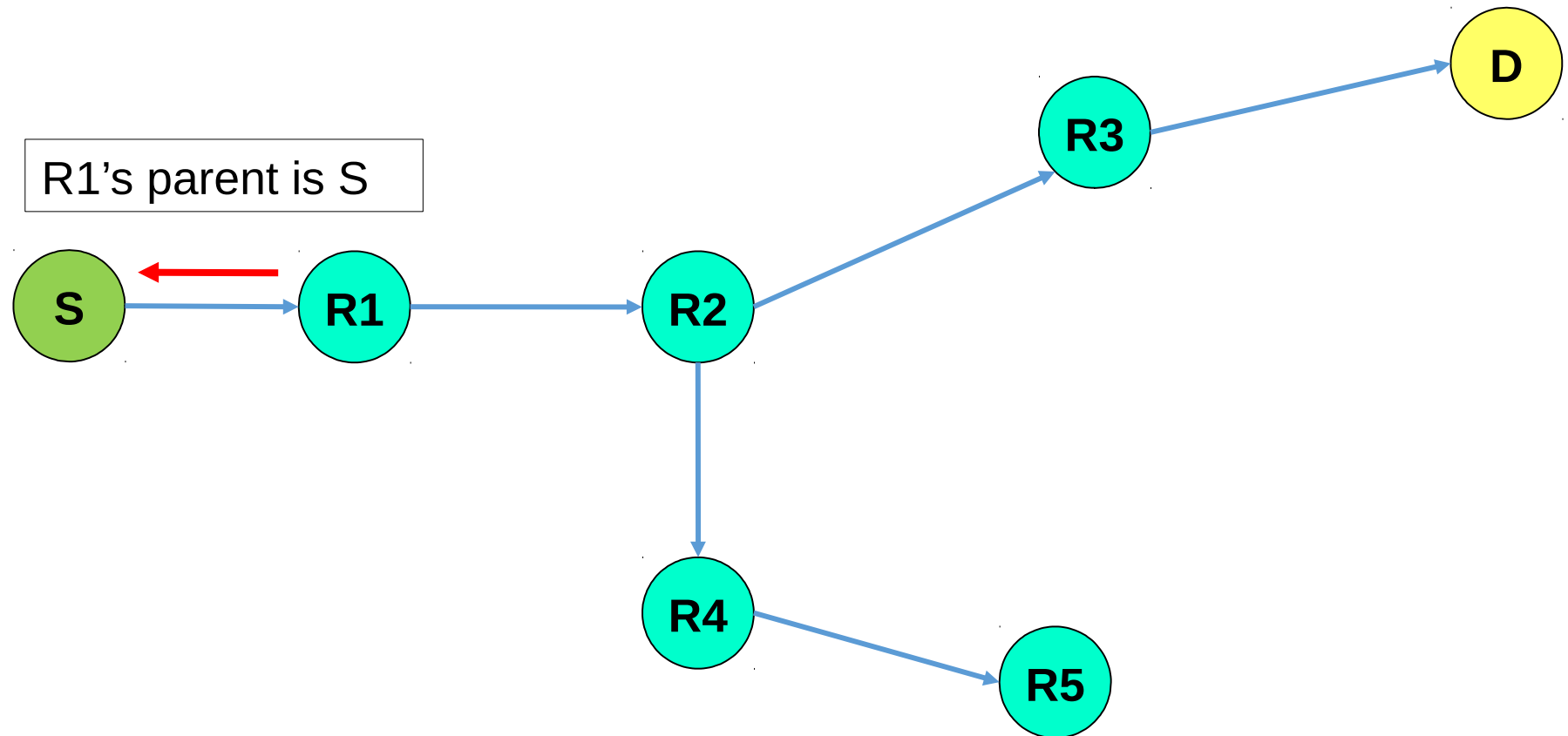
# Phase 1: Collecting the routing information



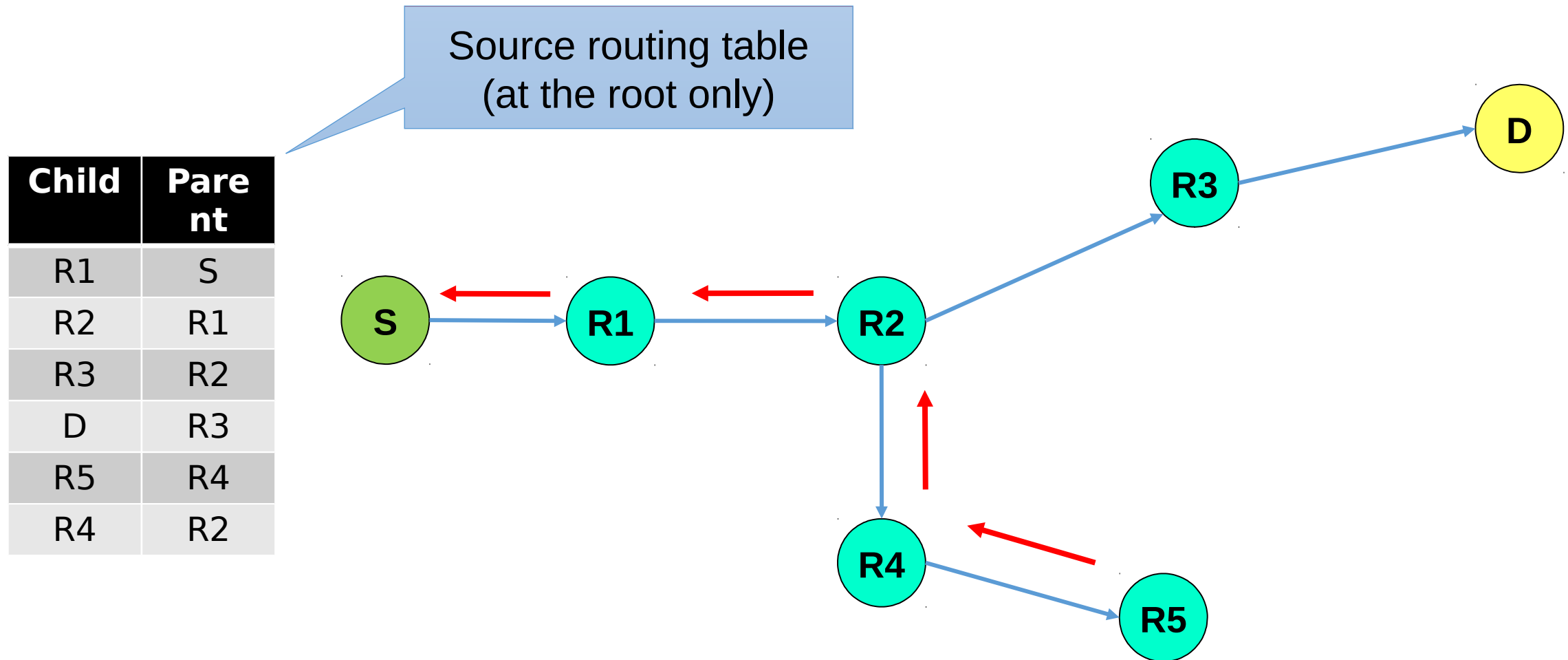


# Phase 1: Collecting the routing information

Child	Parent
R1	S
R2	R1
R3	R2
D	R3



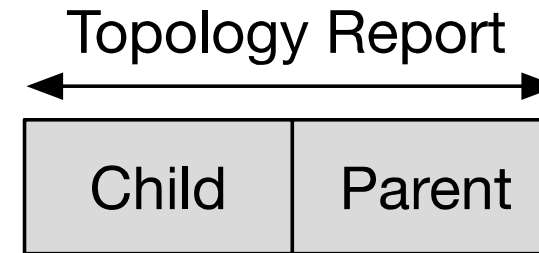
# Phase 1: Collecting the routing information



# Topology Reports

- **Dedicated Topology Reports:**

- Dedicated control traffic to inform the sink of the parent of each node



Child	Parent
R1	S
R2	R1
R3	R2
D	R3
R5	R4
R4	R2

- **Piggybacking:**

attach topology reports to data collection packets

- When the application sends a packet, e.g., using: **my\_collect\_send()**
- Include a header with the parent of the sender



# When Sending Packets

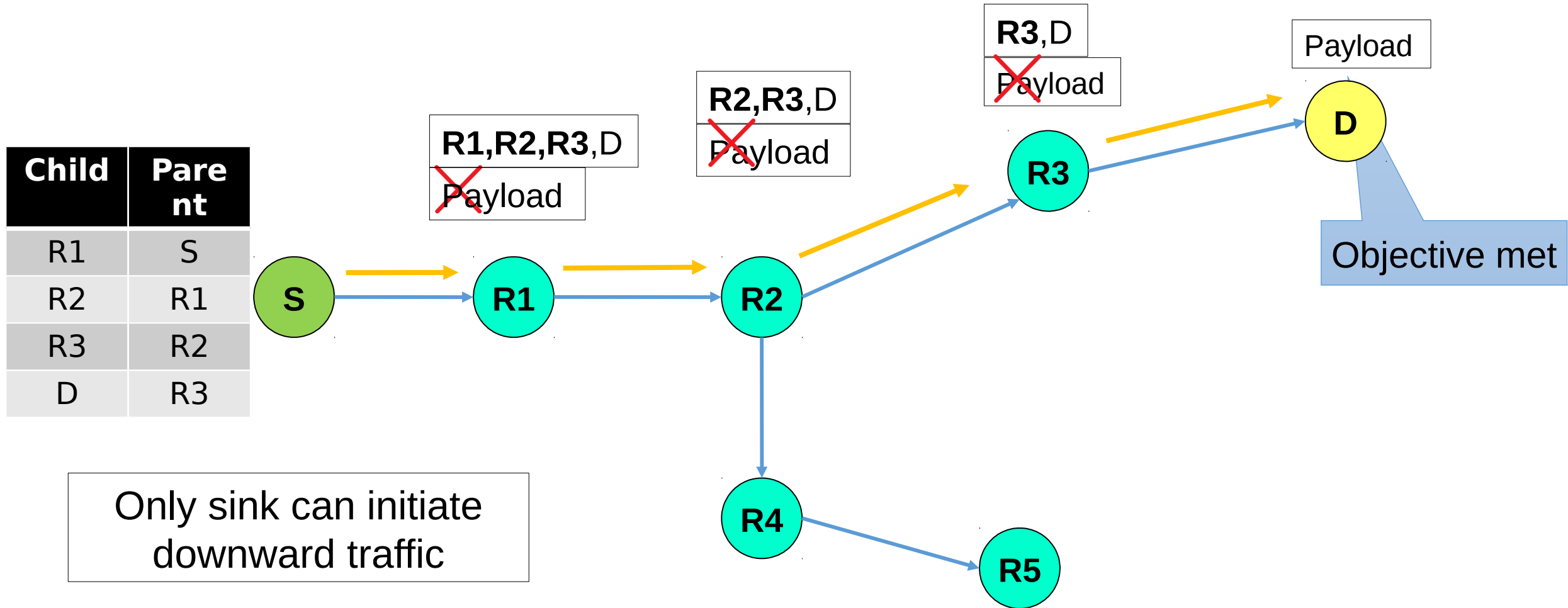
- When node **S** sends a data packet to **D**, it checks a source routing table. If route exists, then entire route is included in the packet header
  - Hence the name **source routing**

## Algorithm:

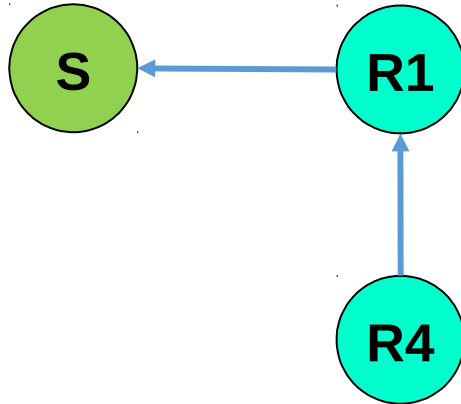
1. Assign  $N := D$
2. Search for node  $N$  in the table to find  $N$ 's parent  $P$
3. If  $N$  is not found or a loop is detected, drop the packet
4. If  $P == \text{root}$ , transmit the packet to next-hop node  $N$
5. Else add  $N$  to the source routing list of the packet, assign  $N := P$ , go to step 2

Child	Parent
R1	S
R2	R1
R3	R2
D	R3
R5	R4
R4	R2

# Source Routing - Phase 2: Data Delivery

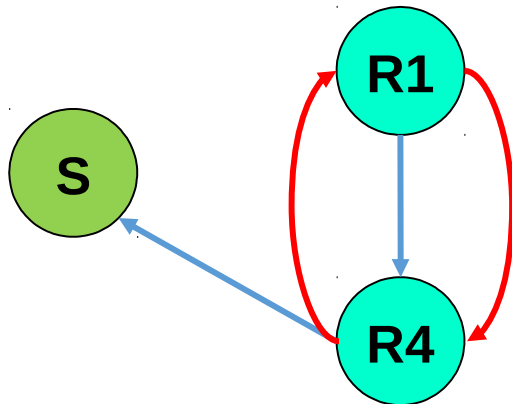


# Source Routing – Loop detection



Child	Parent
R1	S
R4	R1

If the message “**R4’s parent is S**” is lost, we have a loop:



Child	Parent
R1	R4
R4	S

Intended record

Child	Parent
R1	R4
R4	R1

Stale record

# Program Structure

- Your program should enable:
  - **Many-to-one data collection** (you already have)
  - **One-to-many data delivery** from the sink to whichever network node
- The **one-to-many interface** should provide two main functions:
  - **Send Function:**  
int **sr\_send**(struct my\_collect\_conn \*c, const linkaddr\_t \*dest);
  - **Recv Callback:**  
void **sr\_rcv**(struct my\_collect\_conn \*c, uint8\_t hops);
- The top-level application should use these functions