# Network Layer Addressing and DV Routing

COMPUTER NETWORKS A.A. 24/25

Leonardo Maccari, DAIS: Ca' Foscari University of Venice,
leonardo.maccari@unive.it

Venice, fall 2024

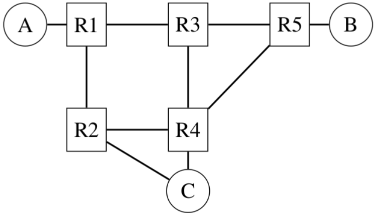
## License

## Sect. 1 The Network Layer

# Extending the Link with the Network Layer

- The Datalink layer takes care of providing connectivity from every host to the other physically connected hosts
- The network layer instead enables the transmission of information between hosts that are not part of the same physical network through intermediate routers.

- The Network layer takes care of sending data from host A to host B, through the intermediate routers (the routers)
- Routers have more than network interface card (NIC) so they have to take a decision on forwarding

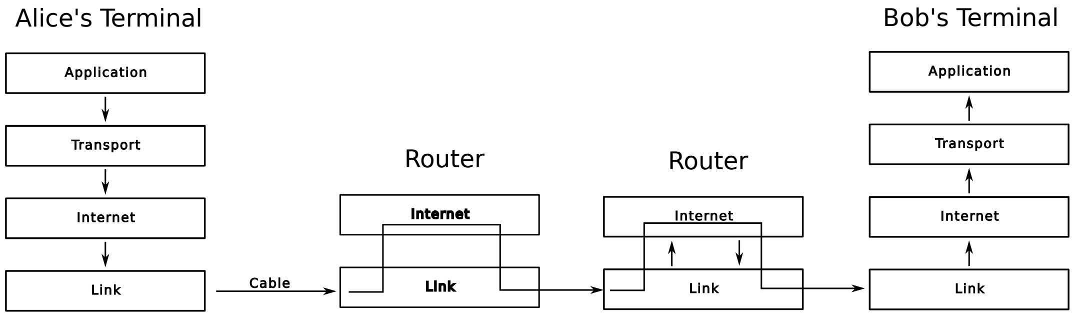


## Assumptions on the Datalink Layer:

The Network Layer assumes that:

- The Datalink layer is *almost reliable*, it means that packet loss can happen but it is expected to be infrequent
- The Datalink layer splits the data in frames. At the Network Layer these frames are called *packets*. A packet contains the data, plus a network layer header. It will be *encapsulated* into the payload of a Datalink layer frame.

## In the TCP/IP Stack



## Types of Network Layers

1. Datagram-oriented organization: the one used on the Internet
2. Circuit-oriented organization.

## The Network Layer

↳1.1 Datagram-oriented Network Layer

## Key Features of Datagram-Oriented Networks:

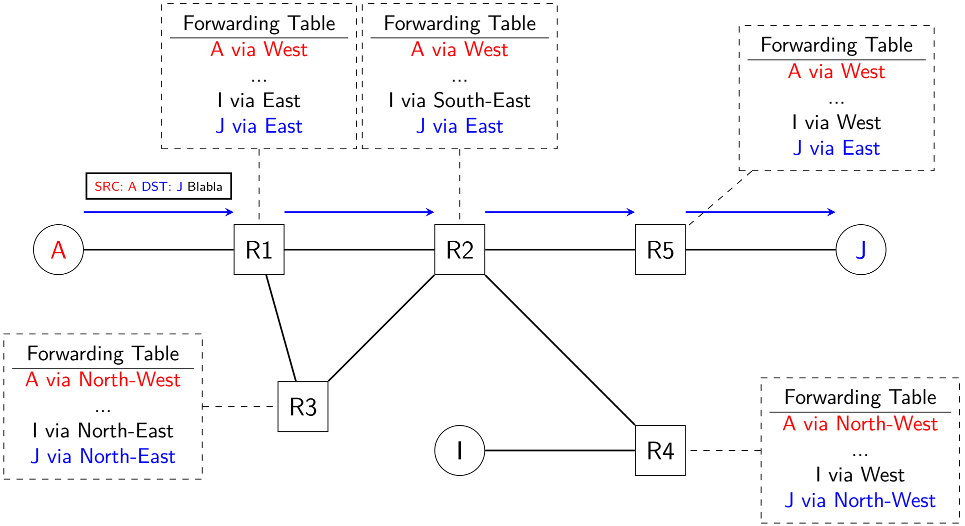
- Each host is identified by a network layer address.
- To send information to a remote host, a host creates a packet that contains:
  - the network layer address of the destination host
  - its own network layer address
  - the information to be sent
- routers use hop-by-hop forwarding: This means that when a router receives a packet that is not destined to itself, it takes a decision on who to send the packet next, this decision is a taken by the router itself (i.e.: there is not a global coordinator).

## Forwarding Tables

- The decision is based on the content of a *forwarding table*.

**Forwarding table:** A data structure that maps each destination address (or set of destination addresses) to the outgoing interface over which a packet destined to this address must be forwarded to reach its final destination. The router consults its forwarding table to forward each packet that it handles.

## An Example Network



## Forwarding Tables Consistency

- Forwarding Tables must be consistent, i.e. there must be a valid path from any host to any other host
- Two errors might happen

## Routing Errors

**Black Holes:** A black hole is the situation in which a router should forward a packet but it does not have an entry in the forwarding table towards the destination. It will drop the packet.

**Routing Loops:** To reach host J, router R1 sends the packet to R2, but R2 sends it back to R1, that sends it back to R2. . . Loops can be longer than just one hop and waste the capacity of the involved links

## Data Vs Control Plane

The software (or hardware) of a router is logically split in two functions:

- The Data Plane: this function takes care of forwarding the data packets, according to the content of the forwarding tables
- The Control Plane: this function takes care of creating the forwarding tables.

## Control Plane

- The easiest way to manage the Control plane is to do it manually.
- The network administrator can simply write the forwarding table by hand.
- This works in small networks, when the network grows we need protocols that take care of constructing the tables. In big networks it is unfeasible.
- Moreover, the network may change (links may fail or new hosts can be added) and the network layer should detect the changes and modify the forwarding table automatically.
- We need a system to manage the control plane automatically.

## Control Plane Packets

- To build the forwarding tables, routers need to exchange information on their local knowledge of the network: We need dedicated messages
- This means that the control plane of the routers sends and receive packets that do not contain user data, but contain information needed to populate the forwarding tables.

## Centralized Vs Distributed

- In private networks, there may be a centralized controller that monitors the whole physical network
- The controller can configure all the forwarding tables
- When something changes, the controller updates the tables
- On the Internet, there is not a single entity that manages the forwarding tables, and so we must use a distributed approach.

## The Network Layer

↳1.2 Addressing and Heterogeneity

## Flat Addressing

- A forwarding table needs to contain all the information to reach every destination
- We mentioned that every host and router has a different address (and even more than one)
- In the flat addressing scheme, these addresses are totally unrelated one to the other.
- For instance, they depend on a hardware component, so their address is decided by the manufacturer

## Flat addresses Pros and Cons

- This makes it easier to set-up the network, because there is no configuration to be done.
- However, it doesn't scale. The internet has billions of addresses, which means that every forwarding table should have an entry for every address.
- This produces a performance degradation, because at each hop, routers need to find the destination in a table made of billions of entries.

# Router Performance

## Product overview

Part of the Cisco® ASR 9000 Series, the Cisco ASR 9001 Router (Figure 1) is a compact high-capacity Provider Edge (PE) router that delivers 120 Gbps of nonblocking, full-duplex fabric capacity in a Two-Rack-Unit (2RU) form factor. Based on the same Cisco IOS® XR software image as the other routers in the Cisco ASR 9000 Series, the Cisco ASR 9001 Router delivers the features and services found on the ASR 9000 Series platforms, allowing customers to standardize on the same Cisco IOS XR image. The Cisco ASR 9001 Router has an Integrated Route Switch Processor (RSP) and two modular bays that support 1 GE, 10 GE, and 40 GE Modular Port Adapters (MPAs). The base chassis has four integrated 10 GE Enhanced Small Form-Factor Pluggable (SFP+) ports, a GPS input for stratum-1 clocking, Building Integrated Timing Supply (BITS) ports, and management ports.

## Router Performance

- 120 Gbps means that if a packet length is 1500B (the largest size allowed) it will take 70 ns to be forwarded.
- It is extremely important that a lookup in the forwarding table does not add delay to this very small value.
- If you spent 70k€ for a router that has gigabit line-speed, you don't want to slow it down by a lookup in a table.

## Hierarchical Addresses

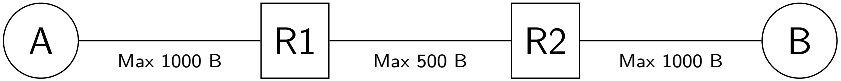
- In this case, the addresses are grouped into blocks
- For instance, each router is assigned one (or more than one) specific block
- The forwarding table does not need to contain all the addresses of all hosts, but only the addresses of a block
- A block can contain thousands/millions of addresses, so the size of the forwarding table is reduced by orders of magnitude.
- Moreover, the forwarding table occupies less memory in the router, that can use a cheaper hardware.

## Hierarchical Addresses

- However, if a host changes its network, it also changes its address
- This happens when you move your laptop from home to the university
- The host is the same, the address changes because the network changes
- So we need:
  - A way for a host to receive an address when it enters a network (easy, will see in future lessons)
  - A way for a host to maintain all its active communication when it enters a new network (hard, not always possible)

# Technology Heterogeneity

- When we expand a single link, we can not assume that the Datalink layer will be the same for all the links
- One specific feature of the Datalink layer is the maximum size of a Datalink frame.
- The situation below is technically possible on the Internet



## Fragmentation

- Host A generates a packet destined to host B, that is encapsulated in a Datalink frame of size 1000B.
- Host A sends it to R1
- R1 can not just send it to R2 because it is larger than the maximum Datalink frame size, it can do three things.

## 1) Drop and Notify

- R1 rejects the packet and sends a control packet back to host A to indicate that it cannot forward packets longer than 500 bytes (including the packet header).
- Host A receives the control packet, and reacts by retransmitting the same information in smaller packets.

! **Not an optimal solution. The same condition can happen more than once on the path to the destination: Host A needs to buffer the packets for a long time.**

## 2) Fragment and Reassemble at Next Hop

- R1 is able to fragment a packet in two parts.
- The first part contains the beginning of the payload and the second the end.
- Both packets are transmitted to router R2. Router R2 reassembles the two packet fragments in a larger packet before transmitting them on the link towards host B.

! **Better, but it forces routers to always reassemble packets before forwarding, which introduces delay, and complexity hard to implement at line-speed.**

## 3) Fragment and Reassemble at Destination

- As in option 2, but each fragment is an independent packet.
- The network layer headers are copied in each fragment, so each fragment can reach the destination
- When they arrive at B, B will collect them all and build the original packet.

! **Best trade-off: it introduces a small delay due to fragmentation on a router, but delegates reassembly at destination.**

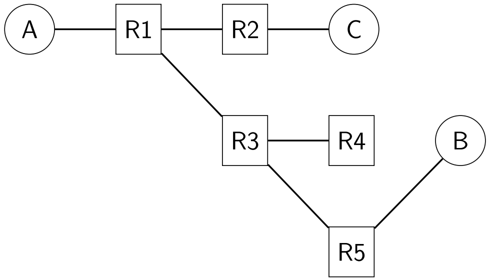
## Sect. 2 Routing Algorithm

## Routing Algorithm

↳2.1 Hot Potato

## A Simple Distributed Routing Algorithm

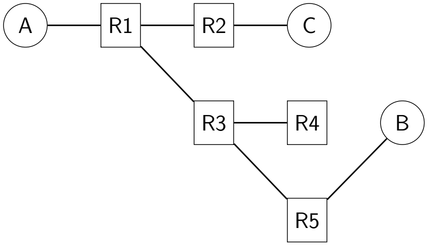
- Let's assume that the underlying network has a particular shape, a tree.
- The network boots, each host has an address, all the forwarding tables are empty.
- How can we dynamically build the forwarding tables?



## The Hot Potato

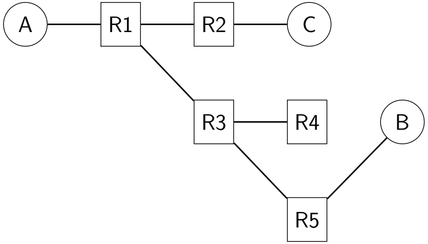
This is a classical algorithm imagined at the beginning of the Internet. Assume A wants to send a packet to B. That's what follows:

1. Host A sends a packet on its only link. The packet header contains the destination address B, and the source address A
2. When receiving this packet, R1 learns that A is reachable via its West interface: $A \rightarrow$ West. Its forwarding table is updated.
3. Since R1 it does not have an entry for destination B in its table, it forwards the packet on all its interfaces (excluding the one it came from): to R2 and R3. This is called a *broadcast*.



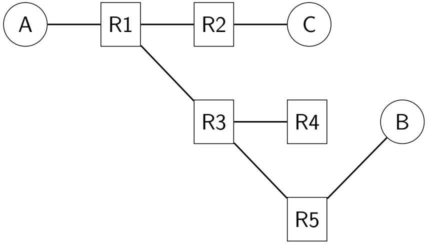
## The Hot Potato

4. When R2 receives the packet, it updates its own forwarding table with a rule $A \rightarrow$ West, and broadcasts the packet again.
5. The packet reaches C
6. Since C is not the intended recipient, C simply discards the received packet.



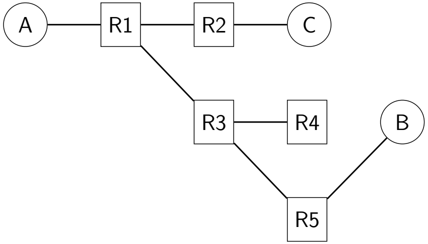
## The Hot Potato

7. Router R3 also receives the packet. It learns that A is reachable via its North-West interface and broadcasts the packet to R4 and R5.
8. R4 does nothing, as the packet destination is not one of the addresses of R4 itself
9. R5 also updates its forwarding table and finally forwards the packet to destination B



## The Hot Potato

- At the end of these steps, the packet is correctly delivered
- Note also that all the routers now have in their forwarding table a correct entry towards A
- This means that if B want to answer to A, it will send a packet to R5, that will forward to R3, then R1 then A, **without any broadcast**.

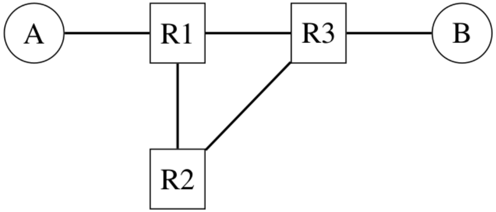


## Limits of Hot Potato

- Hot potato is used in simple, small networks (like Ethernet local area networks)
- It is fundamental that the network is a tree, or else a loop is created. . .

## Loops

- When R1 receives the packet, it broadcasts to R2 and R3
- R2 broadcasts to R3
- R3 broadcasts to B and R1. B receives the packet
- However, R1 broadcasts again to R2, that broadcasts to R3, that broadcasts to B and R1. . .
- . . . the packet will travel forever and B will receive infinite copies of it.



## More Sound Routing Approaches

- The hot potato algorithm is very simple, which we know it is a big plus for networks
- One advantage it has is that it does not require any added control packet, the routing tables are built only looking at the packet headers
- However it does not work well on complex topologies.
- We need more complicate but sound approaches.
- Before that, let me mention one completely different architecture. . .

## Routing Algorithm

↳2.2 Mentioning Virtual Circuits

## Virtual Circuits

- The analog telephone network worked creating a physical circuit from the sender to the receiver
- A similar approach can be used when a centralized controller manages the network
- For every couple of sender/receiver, the whole path is pre-determined and lives as long as there is traffic to send.
- Every packet has an header field that specifies the *virtual circuit*, and routing is done accordingly.
- This technology is used in very limited situations so we don't look at it in details.

## Routing Algorithm

↳2.3 The Control Plane

## The Control Plane

- The only routing algorithm that we mentioned so far (Hot Potato) does not need a real control plane.
- The forwarding tables are built using the data packets, and the logic is embedded with the forwarding logic.
- With real world protocols we need more complicate algorithms, and routers need to exchange some packets with the goal of populating the forwarding tables.

## Algorithms, Protocols, Daemons

**Algorithm:** An algorithm is what you have studied before: a finite sequence of non ambiguous instructions that solve a certain problem. Scientists produce algorithms, and give proof of correctness and performance, algorithms often bring the names of the inventor.

## Algorithms, Protocols, Daemons

**Routing Protocol:** A routing protocol is instead a set of specifications (packets format, state machines etc.) that is used to implement the algorithm on the routers. If the protocol is well formalized, any vendor can implement it with their own software and the routers will be interoperable. Protocols are specified in Internet Standards, open document that are the base reference for software designers (RFCs for those produced by IETF).

## Algorithms, Protocols, Daemons

**Daemon:** A routing daemon is the software running on the router that implements the routing protocol. It is the piece of logic that will create the control packets, send them on the network interface, elaborate the packets that arrive, maintain the routing table in the router memory and translate it into a forwarding table when some changes happen

## Example:

- A distance vector routing uses an algorithm known as **distributed Bellman-Ford**
- One protocol that implements it is called **Routing Information Protocol (RIP),** that was standardized (last time) in RFC 1058 `https://datatracker.ietf.org/doc/html/rfc1058`
- One open source implementation is provided by the Bird software `https://bird.network.cz/?get_doc&v=16&f=bird.html#toc6.11`

## Building Routing Tables

- The goal of the control plane or a routing protocol is to build a **routing table** (RT).
- A routing table R can be modeled as a data structure that stores, for each known destination address `d`, the following attributes :

| | |
|---|---|
| `R[d].link` | is the outgoing link that the router uses to forward packets towards destination `d` |
| `R[d].cost` | is the sum of the metrics of the links that compose the shortest path to reach destination `d` |
| `R[d].time` | is the timestamp of the last control packet that carried information about `d` |

## Routing Vs Forwarding Tables

- A routing table is a more abstract concept than a forwarding table.
- It contains more information, and there can be more than one routing table in the same router.
- For instance, one routing table can be managed by the routing daemon, another can be configured manually by the administrator
- The goal of a routing protocol is to fill the routing tables in any of the routers of the network

## Routing Vs Forwarding Tables

- In a router all the routing tables are merged into a forwarding table, that is what the router uses to take decisions packet by packet.
- In software routers the forwarding table is implemented in the operating system, but in hardware routers the forwarding table is loaded directly on programmable hardware for performance reasons

## Routing Algorithm

↳2.4 Distance-Vector Routing

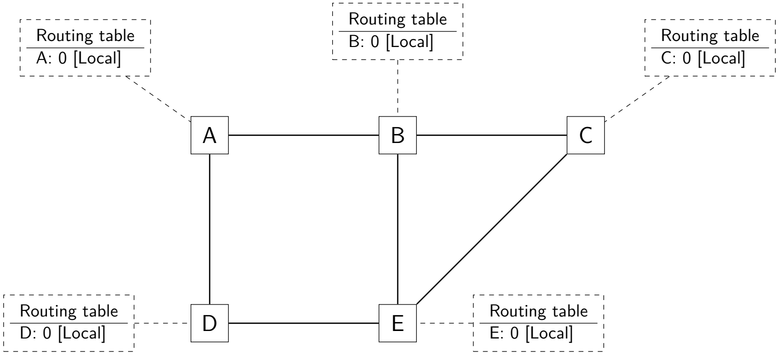
## Distance Vector (DV) Routing

- DV routing is efficient, easy to implement, reasonably lightweight, but slow to converge.
- It is used in RIP, but in a similar way also in the Border Gateway Protocol (BGP) that is the fundamental component that keeps together the whole Internet.
- Let's see how it works.

## Initialization

- We assume that every link has a *cost*.
- This must be a penalty to pass though that link.
- In the easiest case all links have the same unitary cost, but more complicate metrics can be used
- Every DV router initializes the routing table `R` with an entry for its own address, with distance zero.

## Network Initialization



## DV generation

- Periodically a router it sends its routing table (the distance vector) to its neighbor nodes
- The order in which the neighbors receive the DV is not defined, it can be any order
- Every router will run the same code when generating and receiving a DV from a neighbor

## Sending DV, Pseudo-code

```
Every N seconds:
    v = Vector()
    for d in R[]:
        # add destination d to vector
        v.add(Pair(d, R[d].cost))
    for i in interfaces
        # send vector v on this interface
        send(v, i)

```

## Receiving DV, Pseudo-code

Every router, upon receiving an advertisement, does the following:

```
# V : received Vector
# l : link over which vector is received
def received(V, l):
    # received vector from link l
    for d in V[]
        if not (d in R[]):
            # new route
            R[d].link = l
            R[d].cost = V[d].cost + l.cost
            R[d].time = now()
        else:
            # existing route, should I update ?
            if ((V[d].cost + l.cost) < R[d].cost) or (R[d].link == l):
                R[d].link = l
                R[d].cost = V[d].cost + l.cost
                R[d].time = now()

```

## DV Update: Details

line 6 If the distance vector contains a destination address that the router does not know, it inserts it inside its routing table via link l and at a distance which is the sum between the distance indicated in the distance vector and the cost associated to link l.

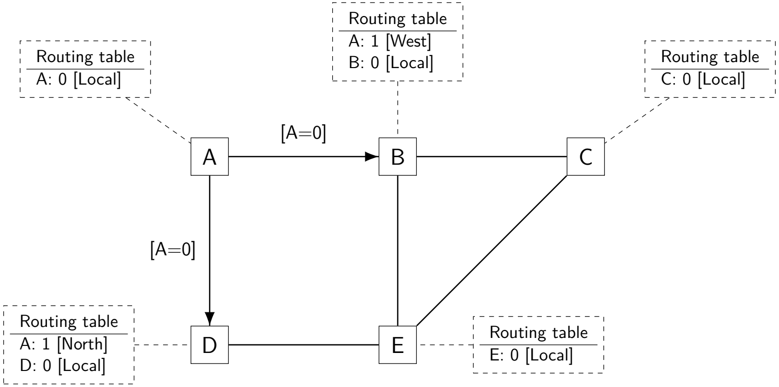
line 13 If the destination was already known by the router, it only updates the corresponding entry in its routing table if either:

- the cost of the new route is smaller than the cost of the already known route: this will make it possible to discover new routes
- the new route was learned over the same link as the current best route towards this destination: if the metric changed, then something has changed on the path to the destination, else, we just update the timestamp field.

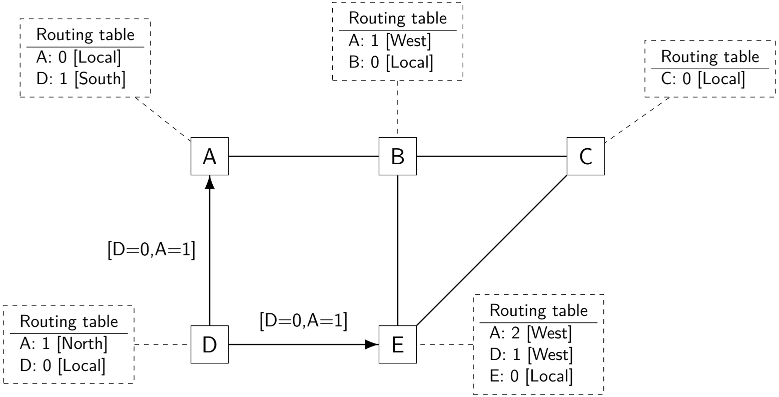
## Convergence Example

- What follows is one possible sequence of events
- We assume unitary link cost
- It can be shown that, given any sequence of events, if all packets are delivered, the convergence is obtained.
- At convergence we expect that every router knows one shortest path to any other router.
- Every diagram shows:
  - the generation of one DV from one router
  - How the RT of the neighbors are updated upon the reception of the DV

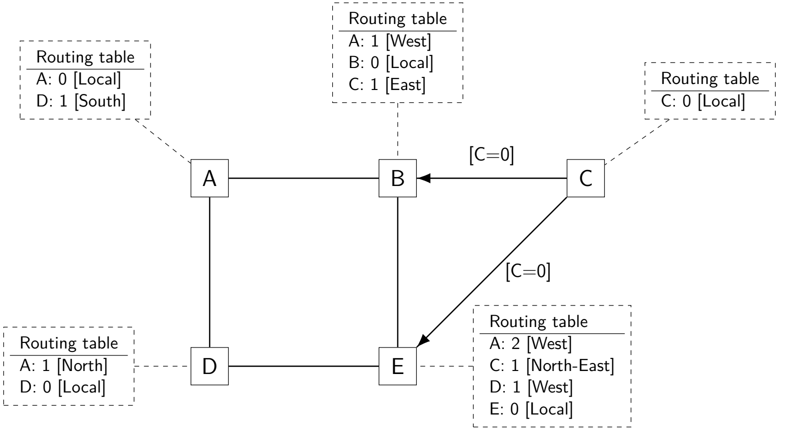
## Router A sends the DV



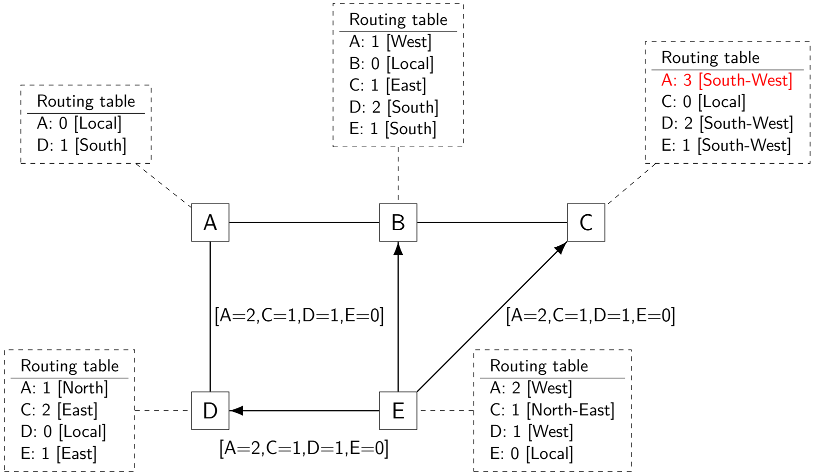
# Router D sends the DV



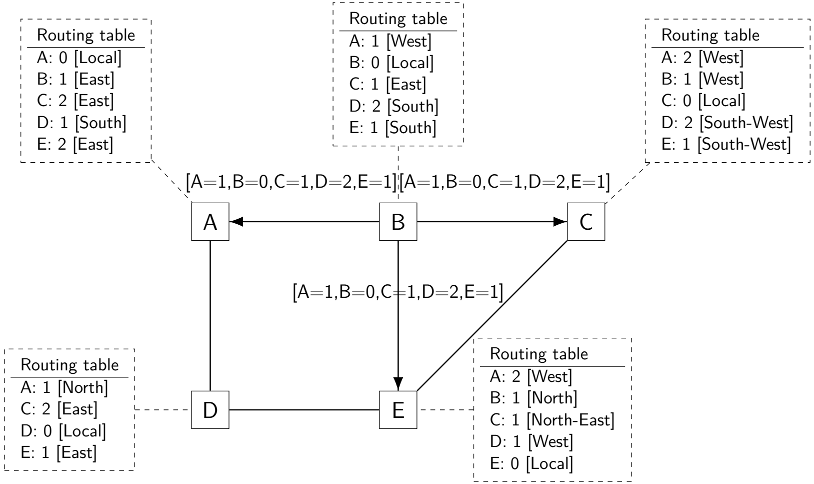
# Router C sends the DV



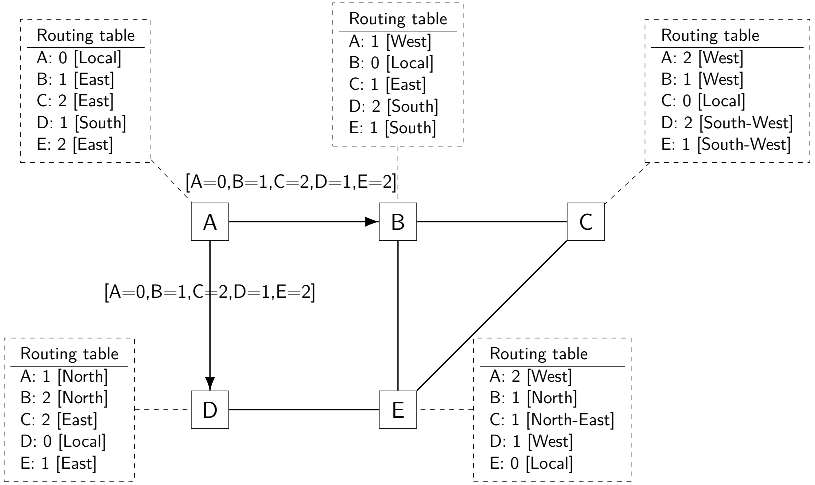
# Router E sends the DV



## Router B sends the DV



## Router A sends the DV (again): Convergence!



## Note well (1)

- Control packets are never *forwarded*: it never happens that a router broadcast a packet that it received from one interface to the neighbors.
- Every router will receive a DV, update its routing table and when its timer expires, it will send its DV to the neighbors.

## Note Well (2)

- At some point the routing tables can be sub-optimal
- Convergence to the best routing tables in a static network is guaranteed, but there can be transitory phases in which the routing tables are sub-optimal

## Note Well (3)

- There can be more than final configuration if there are multiple paths with the same weight.
- For instance in the example `B`$\rightarrow$`E`$\rightarrow$`D` and `B`$\rightarrow$`A`$\rightarrow$`D` have the same cost so `B` could use any of them
- The order of the events will make `B` pick one or the other.

## Note Well (4)

- One round of timers may not be sufficient to reach convergence, indeed in the example `A` must send two DVs before convergence is reached.
- The length of the transitory phase depends on the exact sequence of events.

## Exercise

- Consider the case in which the order of generation of the DV is strictly alphabetic (from A to E)
- Make another sequence of events and RT update, check that it reaches convergence.

## Failure Recovery

- As all routers send their distance vector every N seconds, the timestamp of each route should be regularly refreshed.
- Thus, no route should have a timestamp older than N seconds.
- Routers have a process that, every N seconds check the timestamp of all the routes stored in their routing table
- If a route is found that is older than $3 \times N$ seconds, its cost is set to $\infty$

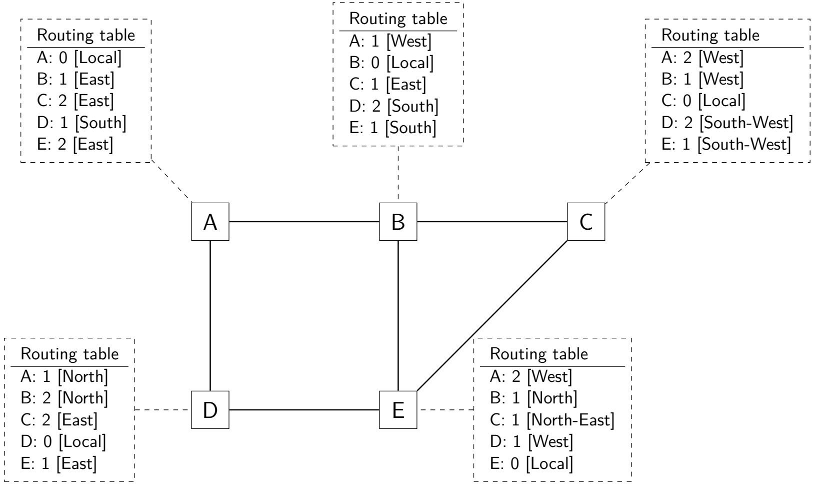
## Failure Recovery

- After antoher $3 \times N$ seconds, the route is removed from the routing table.
- Note that this gives the router enough time to eventually receive a DV from another router with a cost lower than $\infty$ and update the route, and at the same time it gives time to neighbors to install better routes coming from some other router.
- So before removing finally the route, the neighboring routers should have received the bad news.

## Failure Recovery

- The value of $3$ used as a multiplier is arbitrary, it can differ from protocol to protocol
- The point is that there should be enough time to ensure that even if some DV are lost due to some temporary transmission error, the network converges.
- Since we assume the error rate on the links is small, a small multiplier should suffice.
- Let's look at an example

# Starting point: Convergence



Routing table (A)

| Routing table |
|---|
| A: 0 [Local] |
| B: 1 [East] |
| C: 2 [East] |
| D: 1 [South] |
| E: 2 [East] |

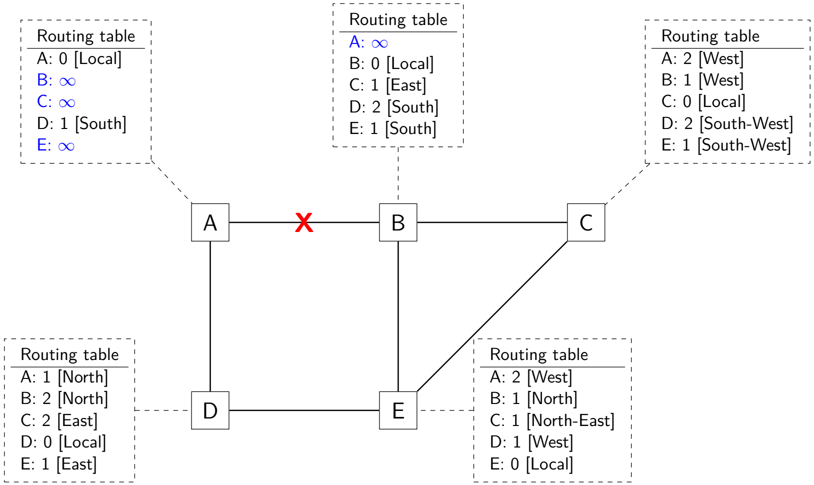
| Routing table |
|---|
| A: 1 [West] |
| B: 0 [Local] |
| C: 1 [East] |
| D: 2 [South] |
| E: 1 [South] |

| Routing table |
|---|
| A: 2 [West] |
| B: 1 [West] |
| C: 0 [Local] |
| D: 2 [South-West] |
| E: 1 [South-West] |

| Routing table |
|---|
| A: 1 [North] |
| B: 2 [North] |
| C: 2 [East] |
| D: 0 [Local] |
| E: 1 [East] |

| Routing table |
|---|
| A: 2 [West] |
| B: 1 [North] |
| C: 1 [North-East] |
| D: 1 [West] |
| E: 0 [Local] |

## Link A-B Fails



Routing table of A:

| Routing table |
|---|
| A: 0 [Local] |
| B: ∞ |
| C: ∞ |
| D: 1 [South] |
| E: ∞ |

Routing table of B:

| Routing table |
|---|
| A: ∞ |
| B: 0 [Local] |
| C: 1 [East] |
| D: 2 [South] |
| E: 1 [South] |

Routing table of C:

| Routing table |
|---|
| A: 2 [West] |
| B: 1 [West] |
| C: 0 [Local] |
| D: 2 [South-West] |
| E: 1 [South-West] |

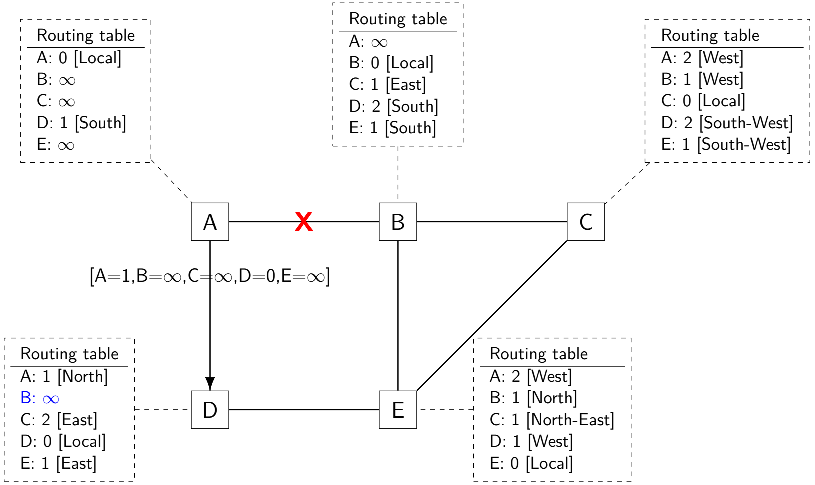
Routing table of D:

| Routing table |
|---|
| A: 1 [North] |
| B: 2 [North] |
| C: 2 [East] |
| D: 0 [Local] |
| E: 1 [East] |

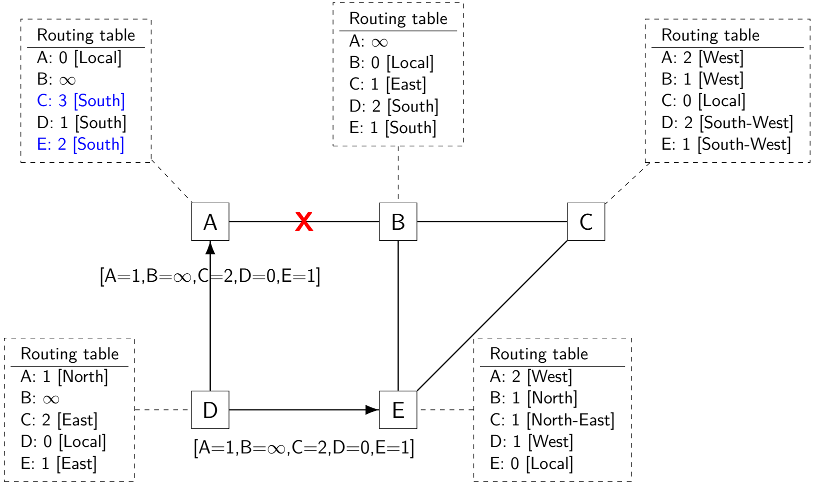
Routing table of E:

| Routing table |
|---|
| A: 2 [West] |
| B: 1 [North] |
| C: 1 [North-East] |
| D: 1 [West] |
| E: 0 [Local] |

# A Sends its DV



# D Sends it DV

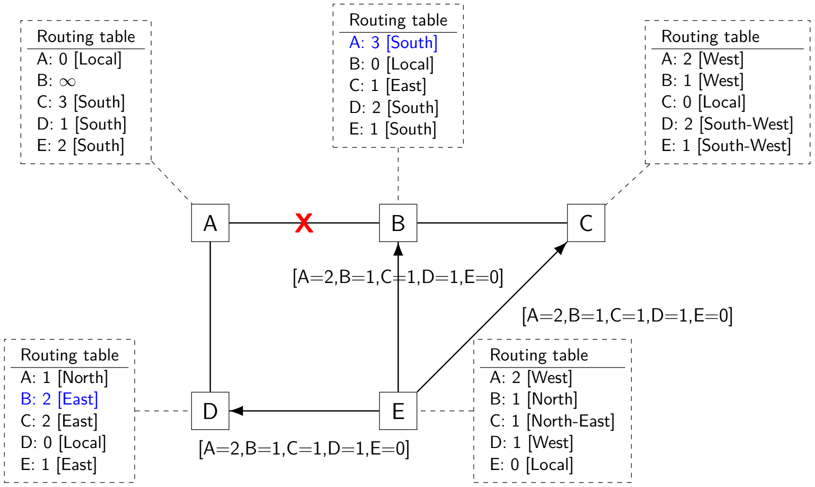


Routing table
A: 0 [Local]
B: ∞
C: 3 [South]
D: 1 [South]
E: 2 [South]
Routing table
A: ∞
B: 0 [Local]
C: 1 [East]
D: 2 [South]
E: 1 [South]
Routing table
A: 2 [West]
B: 1 [West]
C: 0 [Local]
D: 2 [South-West]
E: 1 [South-West]
A
B
C
D
E
[A=1,B=∞,C=2,D=0,E=1]
[A=1,B=∞,C=2,D=0,E=1]
Routing table
A: 1 [North]
B: ∞
C: 2 [East]
D: 0 [Local]
E: 1 [East]
Routing table
A: 2 [West]
B: 1 [North]
C: 1 [North-East]
D: 1 [West]
E: 0 [Local]

## Partial Recovery

- At this stage, A and D have only partially recovered
- Router B does not have a route to A
- However, router C will send to router B the traffic to A
- This is a **black hole!** All the traffic from C to A will reach B and will be dropped.

## E Sends it DV


Routing table
A: 0 [Local]
B: ∞
C: 3 [South]
D: 1 [South]
E: 2 [South]
Routing table
A: 3 [South]
B: 0 [Local]
C: 1 [East]
D: 2 [South]
E: 1 [South]
Routing table
A: 2 [West]
B: 1 [West]
C: 0 [Local]
D: 2 [South-West]
E: 1 [South-West]
A
B
C
X
[A=2,B=1,C=1,D=1,E=0]
[A=2,B=1,C=1,D=1,E=0]
Routing table
A: 1 [North]
B: 2 [East]
C: 2 [East]
D: 0 [Local]
E: 1 [East]
D
E
[A=2,B=1,C=1,D=1,E=0]
Routing table
A: 2 [West]
B: 1 [North]
C: 1 [North-East]
D: 1 [West]
E: 0 [Local]

## Full, sub-optimal recovery

- The blackhole will is now fixed
- However, the routing table of C is not totally correct.
- C still uses a sub-optimal path: 4 hops (C-B-E-D-A) instead of 3 (E-D-A)
- This will be fixed when B and then E send their next DV.
- Exercise: Complete the evolution of the network till convergence.