PADERBORN UNIVERSITY

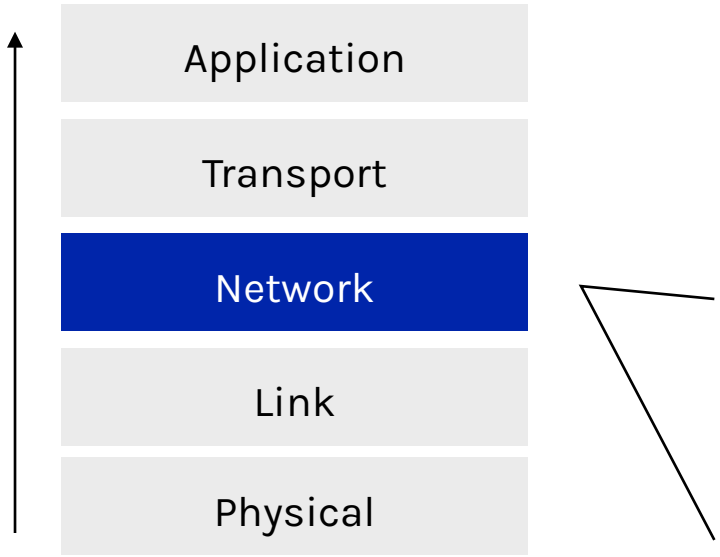
Computer Networks (WS23/24)

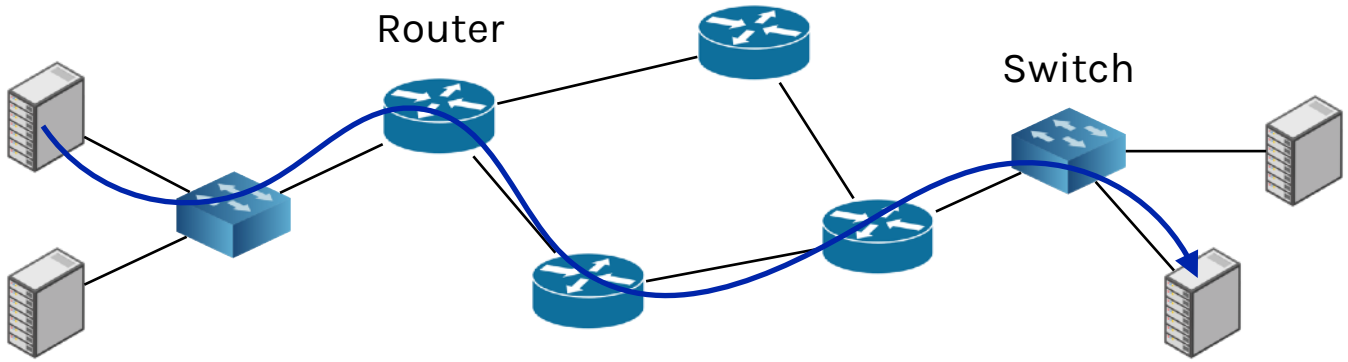
L6: The Network Layer - Part 2

Prof. Dr. Lin Wang
Computer Networks Group (PBNet)
Department of Computer Science
Paderborn University

Materials inspired by Shyam Gollakota and Laurent Vanbever

Learning objectives





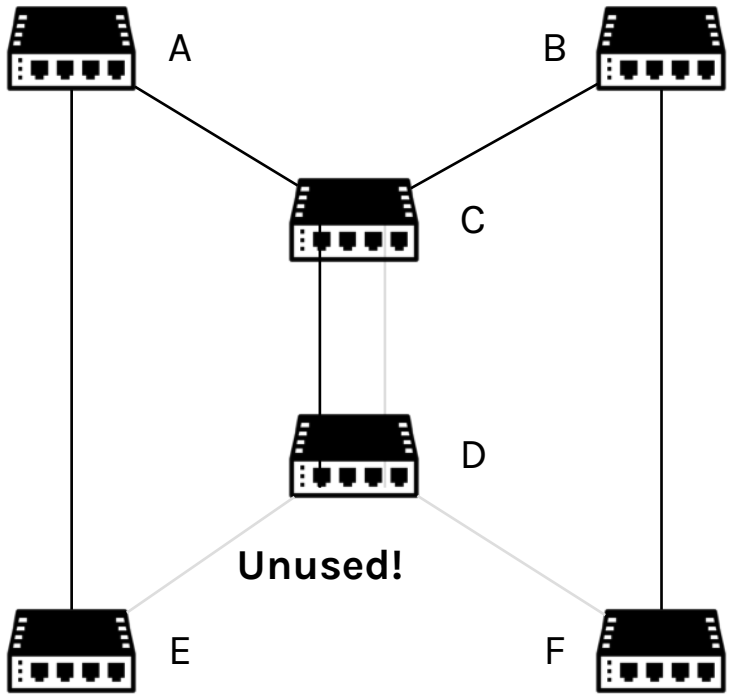
How to find a path between two entities on the Internet?

Part 2

- Routing, shortest-path algorithms
- Intra-domain routing (link-state vs. distance vector)
- Inter-domain routing (BGP)

Routing and Shortest Path Algorithms

Problems with spanning tree



Upon failures, it takes time to reconstruct the spanning tree protocol

Many links are unused (not on the spanning tree), leading to low efficiency

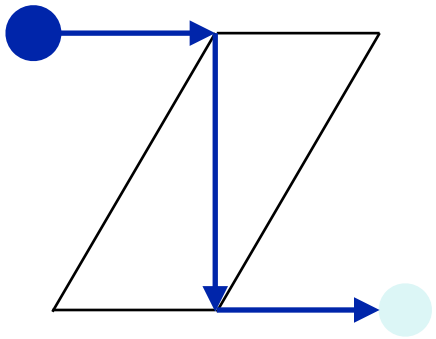
Keep all the links, apply routing to find the best path!

Bandwidth allocation

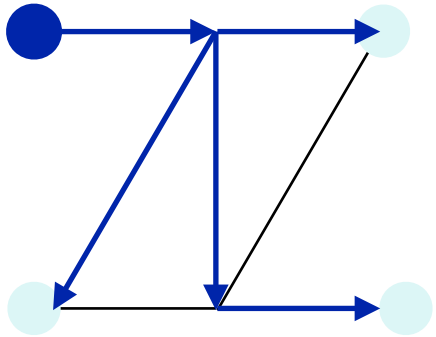
Many mechanisms, including routing, deal with bandwidth allocation

Mechanism	Timescale / adaptation
Load-sensitive routing	Seconds / traffic hotspots
Routing	Minutes / equipment failures
Traffic engineering	Hours / network load
Provisioning	Months / network customers

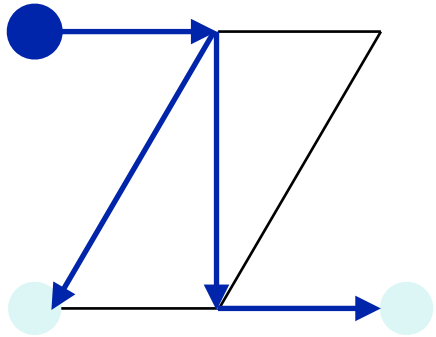
Delivery goals



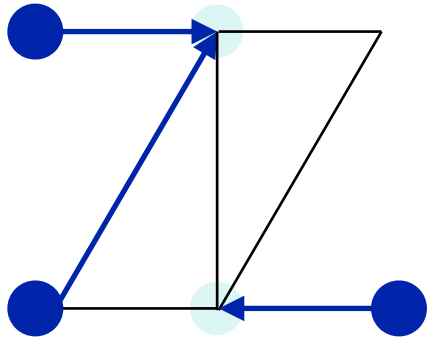
Unicast



Broadcast



Multicast



Anycast

Routing goals

Correctness

Find paths that work

Efficiency

Use network bandwidth well

Fairness

Does not starve any nodes

Fast

Recovers quickly after changes

Scalability

Works well as network grows large

Rules of routing algorithms

Decentralized, distributed setting

- All nodes are alike; no central controller
- Nodes only know what they learn by exchanging messages with neighbors
- Nodes operate concurrently
- There might be node/link/message failures

Definition of "best" paths

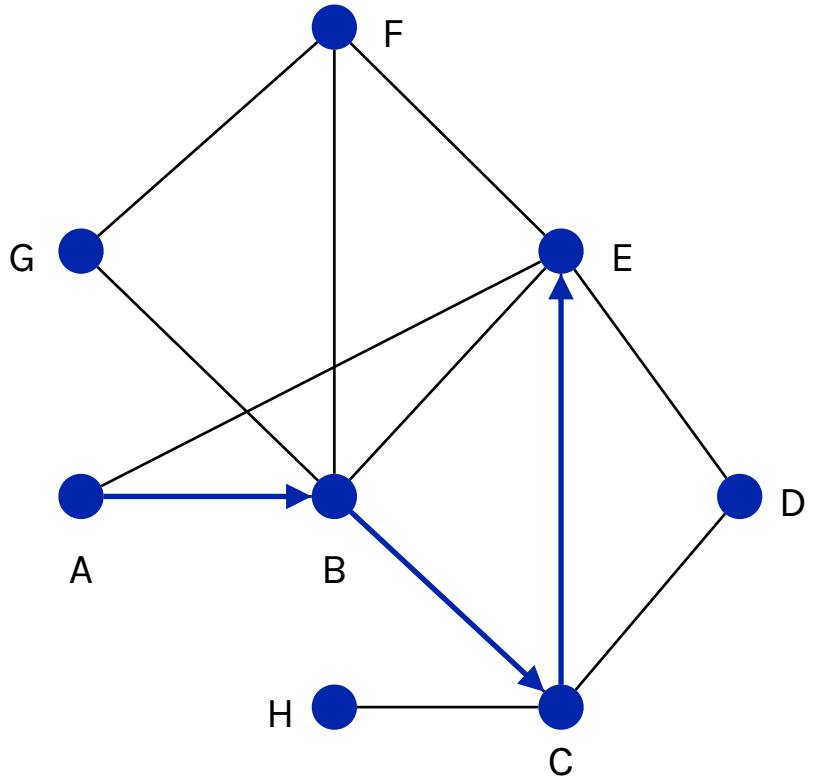
Shortest paths

In terms of…

- Latency, avoid circuitous paths
- Bandwidth, avoid slow links
- Money, avoid expensive links
- Hops, to reduce switching

Only consider topology

- Not workload like hotspots



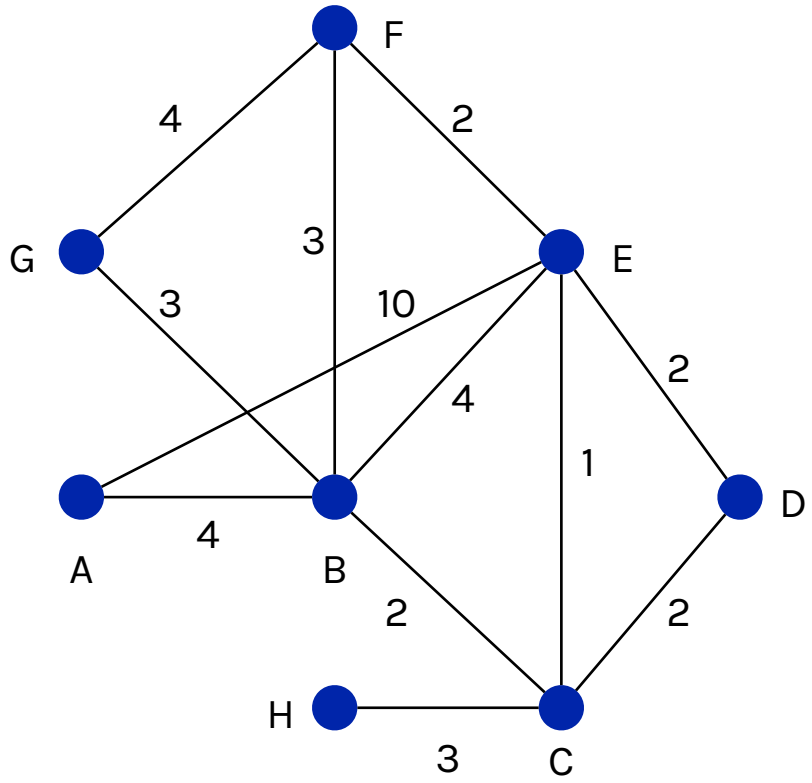
Cost function on links

Approximate "best" by a cost function that captures multiple factors

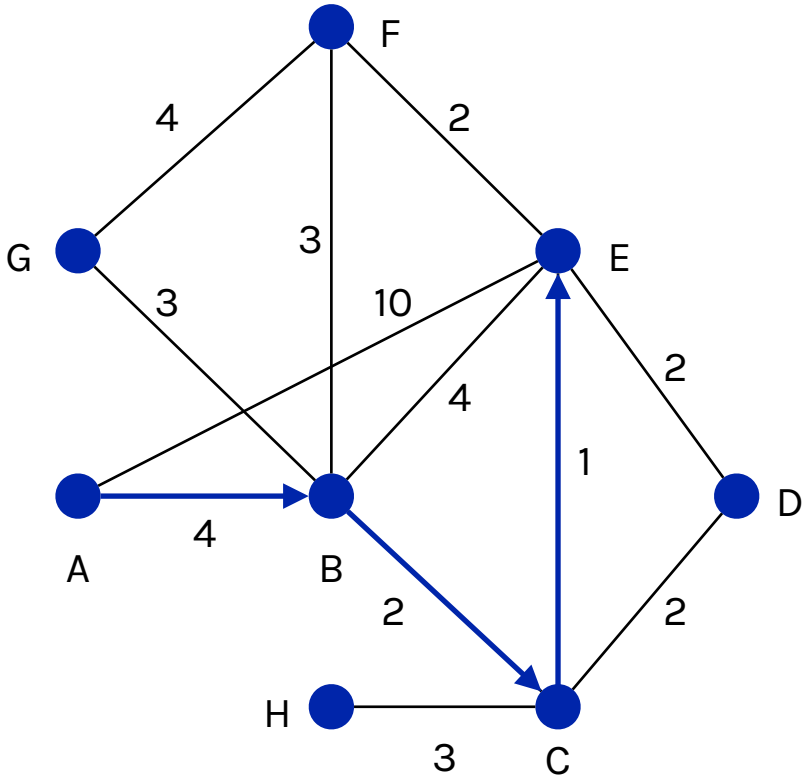
- Assign each link a cost (distance)
- Define best path between each pair of nodes as the path that has the lowest total cost (or is shortest)
- Pick randomly to break any ties

Example: best path from A to E

- All links are bidirectional (can be extended to asymetric cases)



Shortest paths



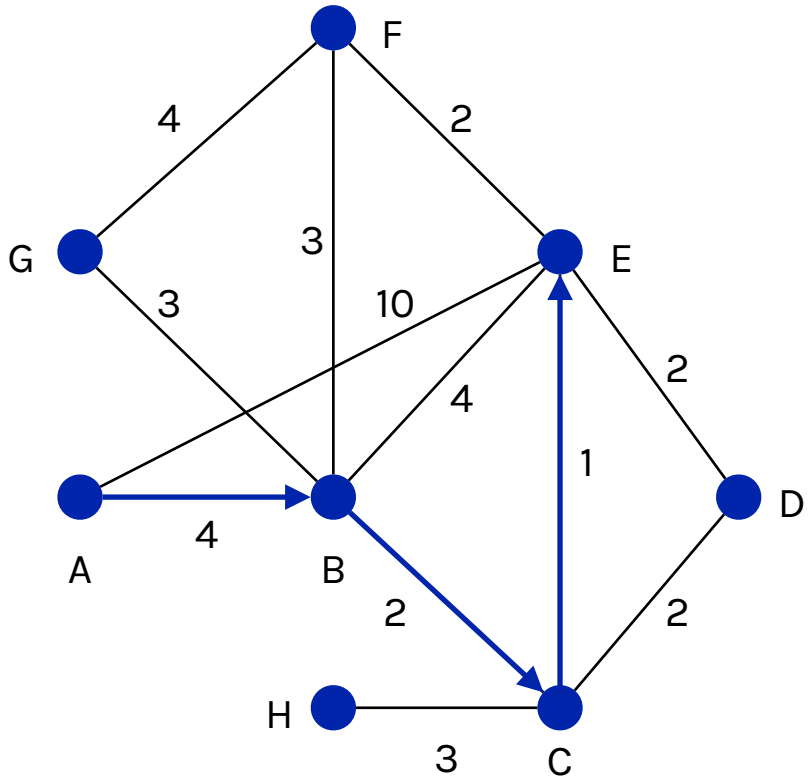
Shortest path A-E: A-B-C-E

- dist(ABCE) = 4 + 2 + 1 = 7

Other paths

- dist(ABE) = 8
- dist(ABFE) = 9
- dist(AE) = 10
- dist(ABCDE) = 10

Optimality property



Subpaths of shortest paths are also shortest paths

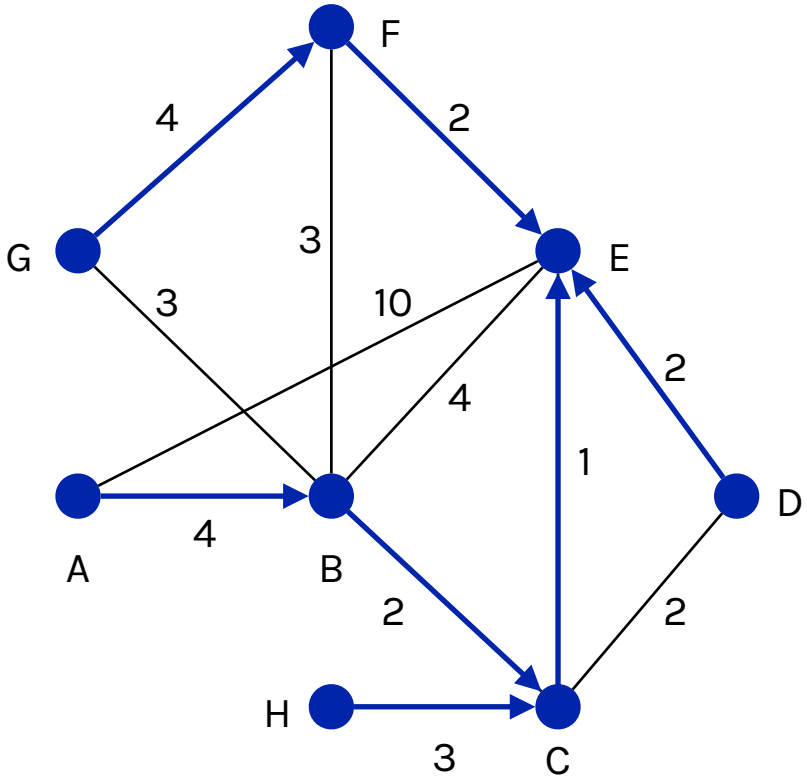
- ABCE is a shortest path
- So are ABC, AB, BCE, BC, CE

Sink (or source) trees

- The union of all shortest paths towards the destination (or from the source)

Example: find the sink tree of E

Sink tree



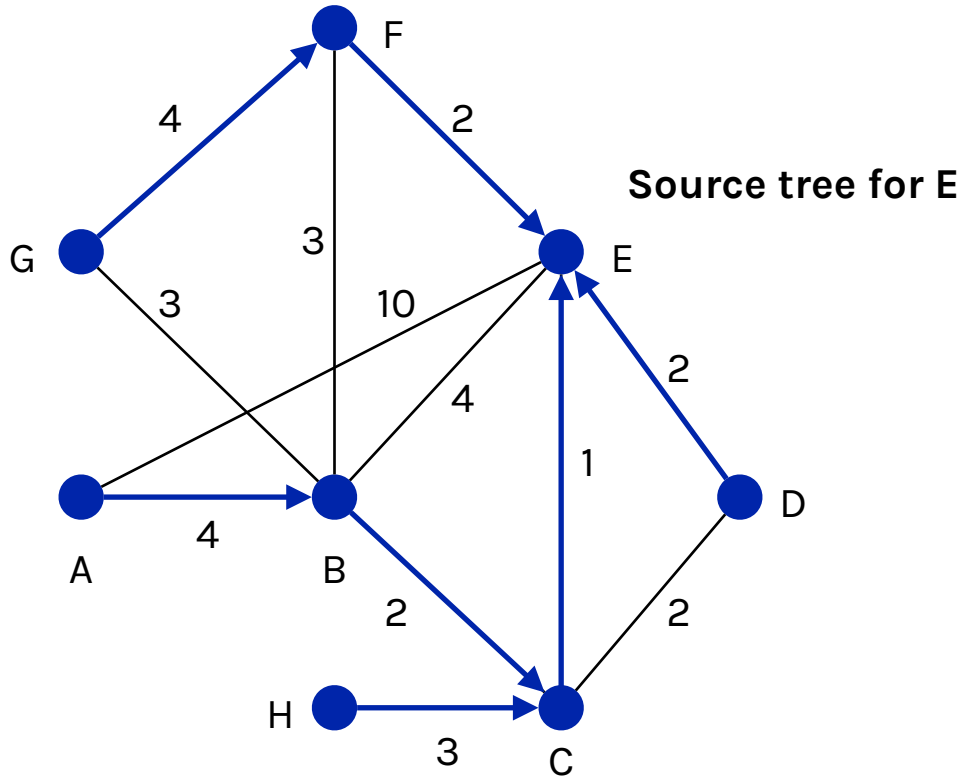
Implications

- Only need to use destination to follow shortest paths
- Each node only need to send to the next hop

Forwarding table at a node

- Lists next hop for each destination
- Routing table may know more

Shortest path calculation



How to calculate the shortest path?

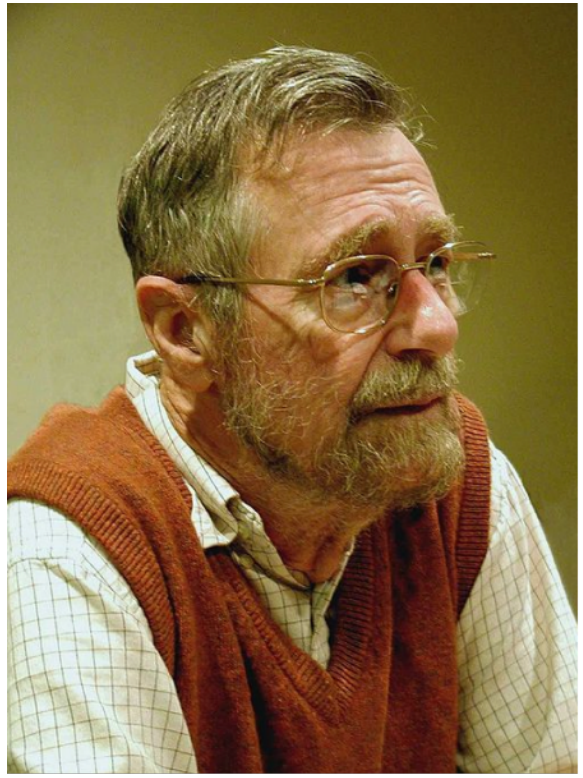
Edsger W. Dijkstra (1930-2002)

Famous computer scientiest

- Programming languages
- Distributed algorithms
- Program verification
- EWD manuscripts (nice to read)

Dijkstra's algorithm, 1969

- Single-source shortest paths, given network with non-negative link costs



https://www.cs.utexas.edu/~EWD/

The Three Golden Rules for Successful Scientific Research.

This note is devoted to three rules, the following of which is necessary if you want to be successful in scientific research. (If you manage to follow them, they will prove close to sufficient, but that is another story.) They are recorded for the benefit of those who would like to be successful in their scientific research, but fail to be so because, being unaware of these rules, they violate them. In order to avoid any misunderstanding I would like to stress, right in its first paragraph, that this note is purely pragmatic: no moral judgements are implied, and it is completely up to you to decide whether you wish to regard trying to be successful in scientific research as a noble goal in life or not. I even leave you the option of not making that decision at all.

The first rule is an "internal" one: it has nothing to do with your relations with others, it concerns you yourself in isolation. It is as follows:

> "Raise your quality standards as high as you can live with, avoid wasting your time on routine problems, and always try to work as closely as possible at the boundary of your abilities. Do this, because it is the only way of discovering how that boundary should be moved forward."

This rule tells us that the obviously possible should be shunned as well as the obviously impossible: the first would not be instructive, the second would be hopeless, and both in their own way are barren.

The second rule is an "external" one: it deals with the relation between "the scientific world" and "the real world". It is as follows:

> "We all like our work to be socially relevant and scientifically sound. If we can find a topic satisfying both desires, we are lucky; if the two targets are in conflict with each other, let the requirement of scientific soundness prevail."

The reason for this rule is obvious. If you do a piece of "perfect" work in which no one is interested, no harm is done, on the contrary: at least something "perfect" --be it irrelevant-- has been added to our culture. If, however, you offer a shaky, would-be solution to an urgent problem, you do indeed harm to the world which, in view of the urgency of the problem, will only be too

willing to apply your ineffective remedy. It is no wonder that charlatanry always flourishes in connection with incurable diseases. (Our second rule is traditionally violated by the social sciences to such an extent that one can now question if they deserve the name "sciences" at all.)

The third rule is on the scale "internal/external" somewhere in between: it deals with the relation between you and your scientific colleagues. It is as follows:

> "Never tackle a problem of which you can be pretty sure that (now or in the near future) it will be tackled by others who are, in relation to that problem, at least as competent and well-equipped as you."

Again the reason is obvious. If others will come up with as good a solution as you could obtain, the world doesn't loose a thing if you leave the problem alone. A corollary of the third rule is that one should never compete with one's collagues. If you are pretty sure that in a certain area you will do a better job than anyone else, please do it in complete devotion, but, when in doubt, abstain. The third rule ensures that your contributions --if any!-- will be unique.

* * *

I have checked the Three Golden Rules with a number of my colleagues from very different parts of the world, living and working under very different circumstances. They all agreed. And were not shocked either. The rules may strike you as a bit cruel... If so, they should, for the sooner you have discovered that the scientific world is not a soft place but --like most other worlds, for that matter-- a fairly ruthless one, the better. My blessings are with you.

Plataanstraat 5
5671 AL NUENEN
The Netherlands

prof.dr.Edsger W.Dijkstra
Burroughs Research Fellow

Dijkstra's algorithm

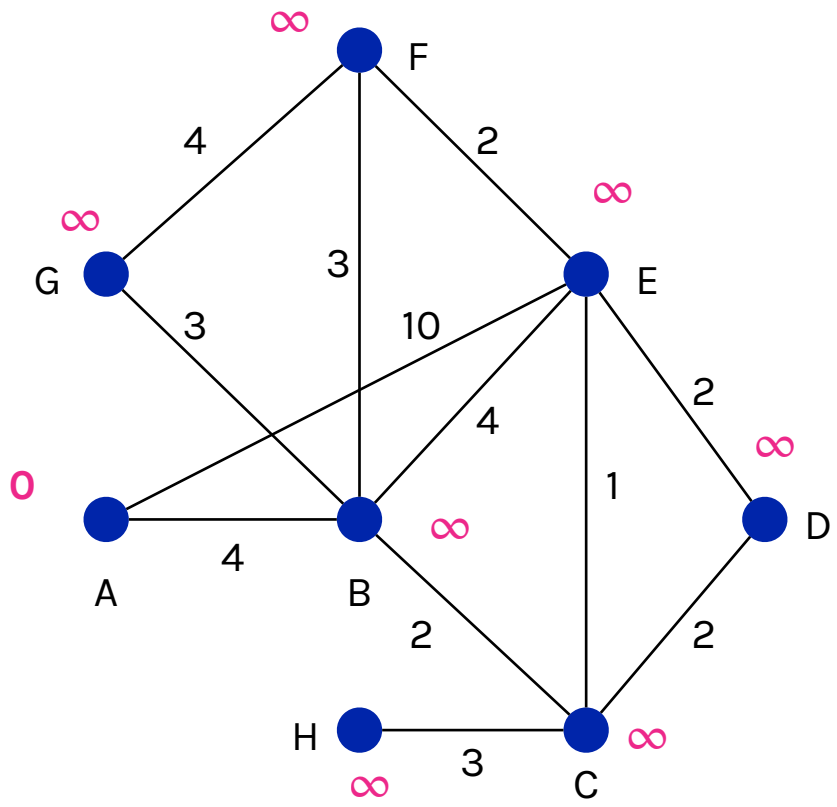
Mark all nodes tentative, set distances from source to 0 for source, and infinity for all other nodes

While tentative nodes remain

- Extract N, a node with lowest distance
- Add link to N to the shortest path tree
- Relax the distance of neighbors of N by lowering any better distance estimates

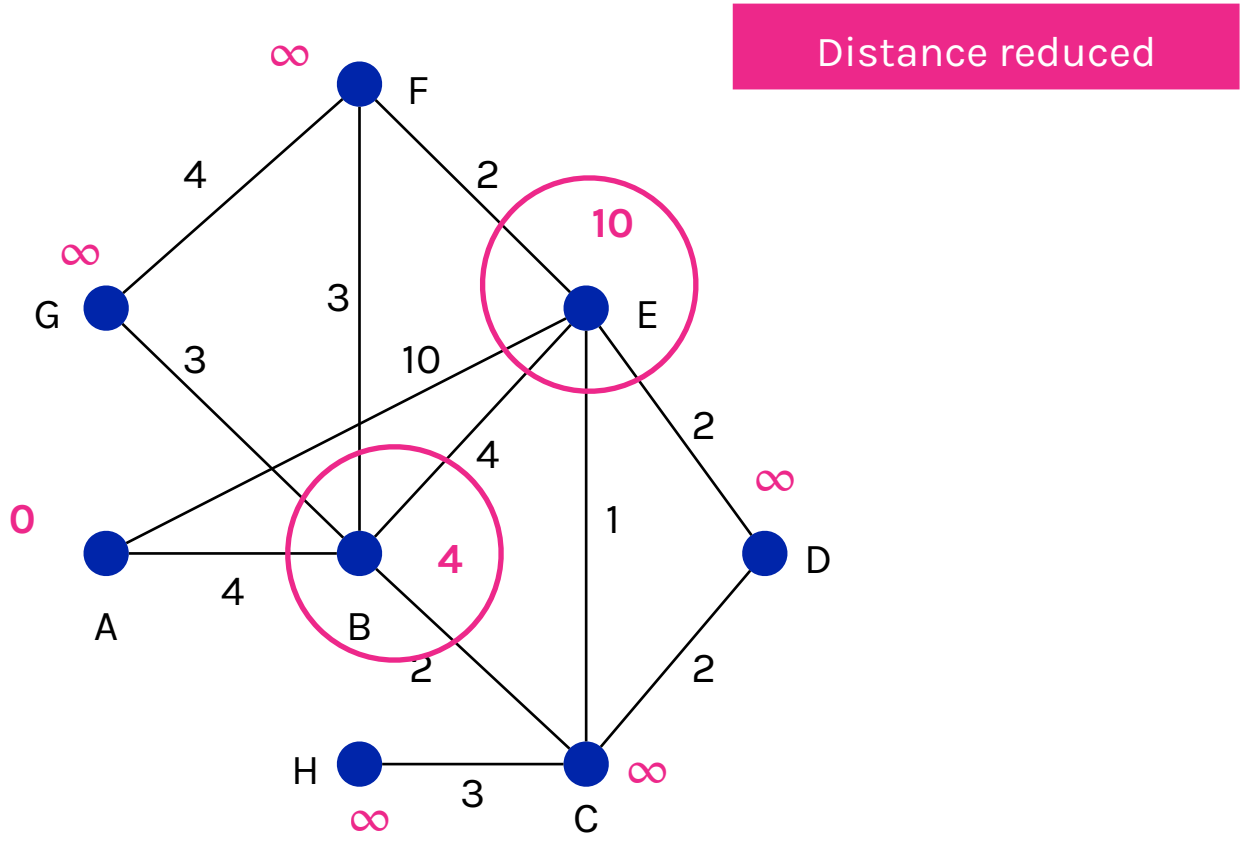
Dijkstra's algorithm example

Initialize



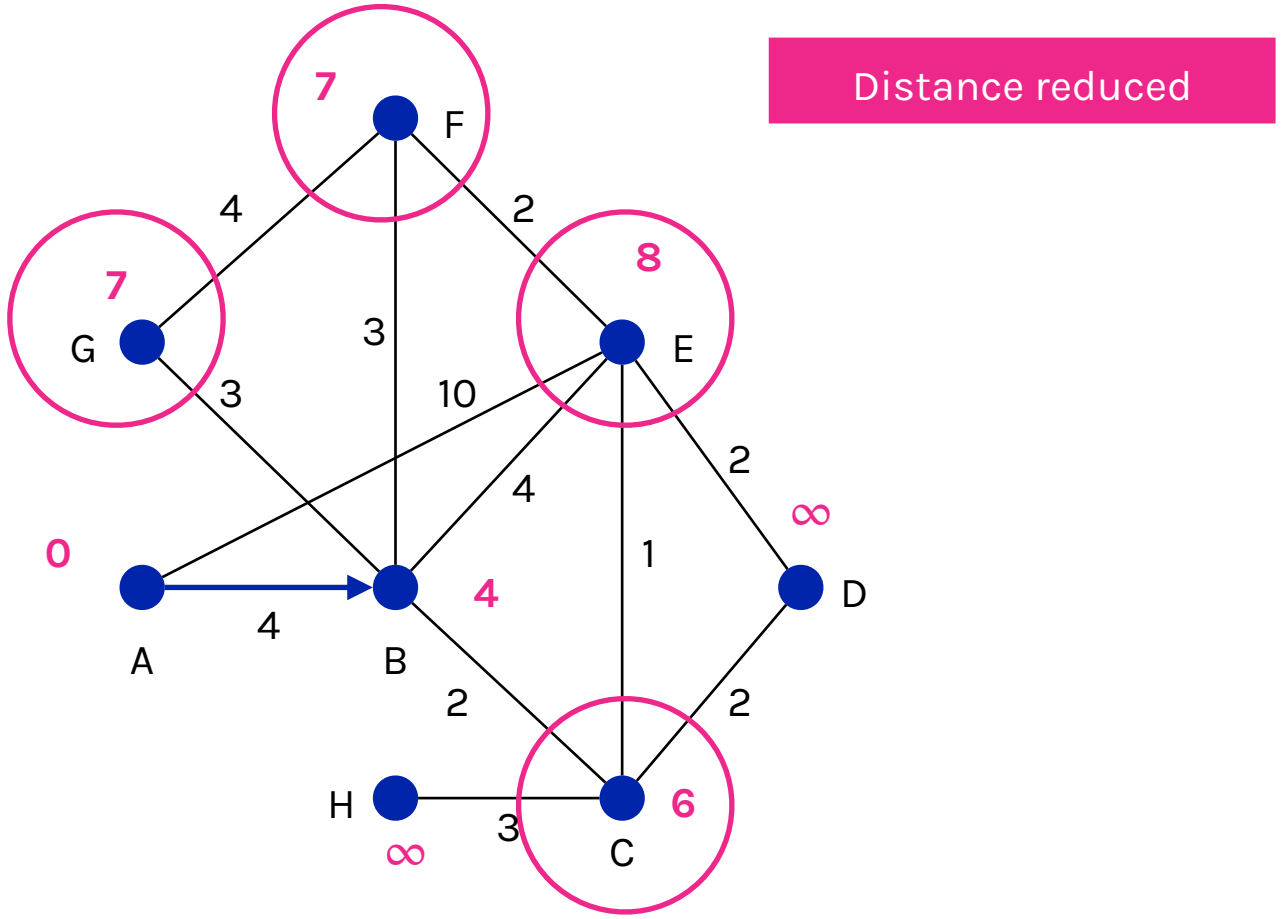
Dijkstra's algorithm example

Relax around A



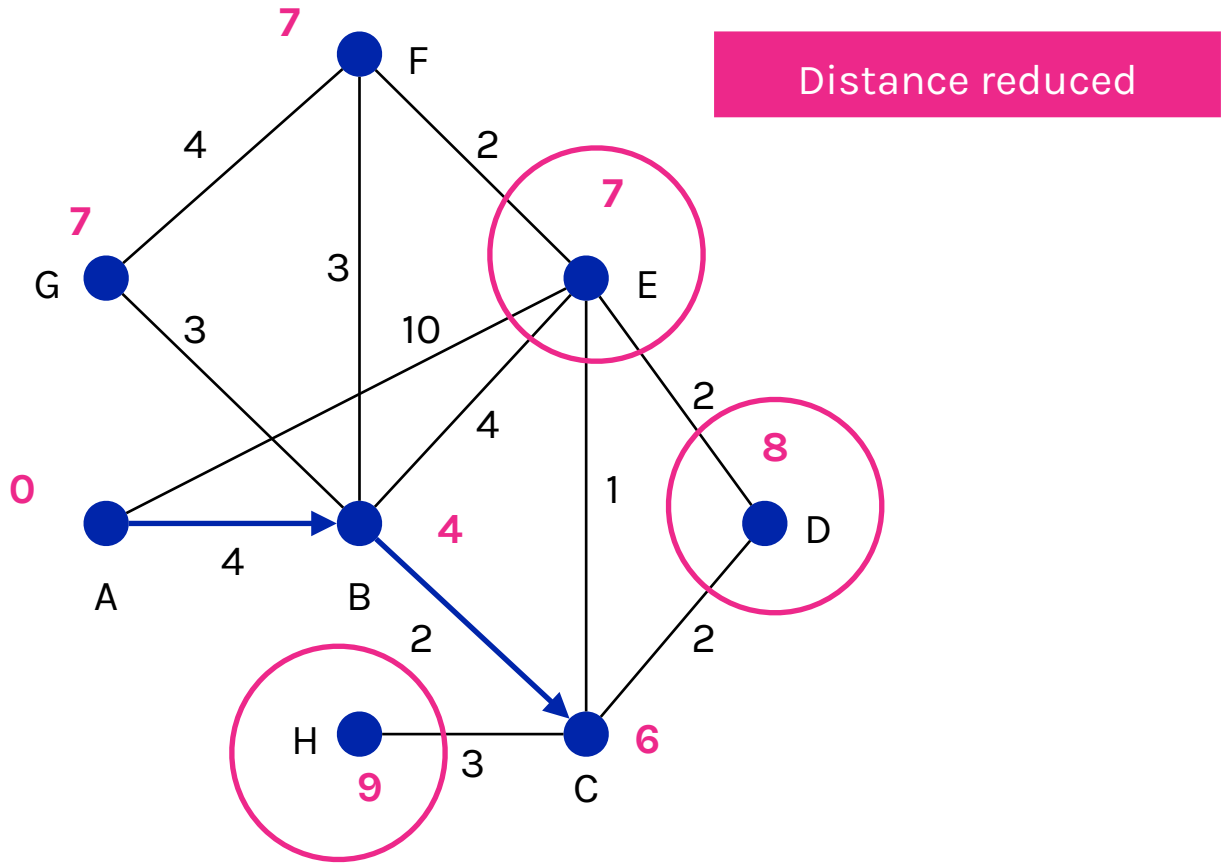
Dijkstra's algorithm example

Relax around B



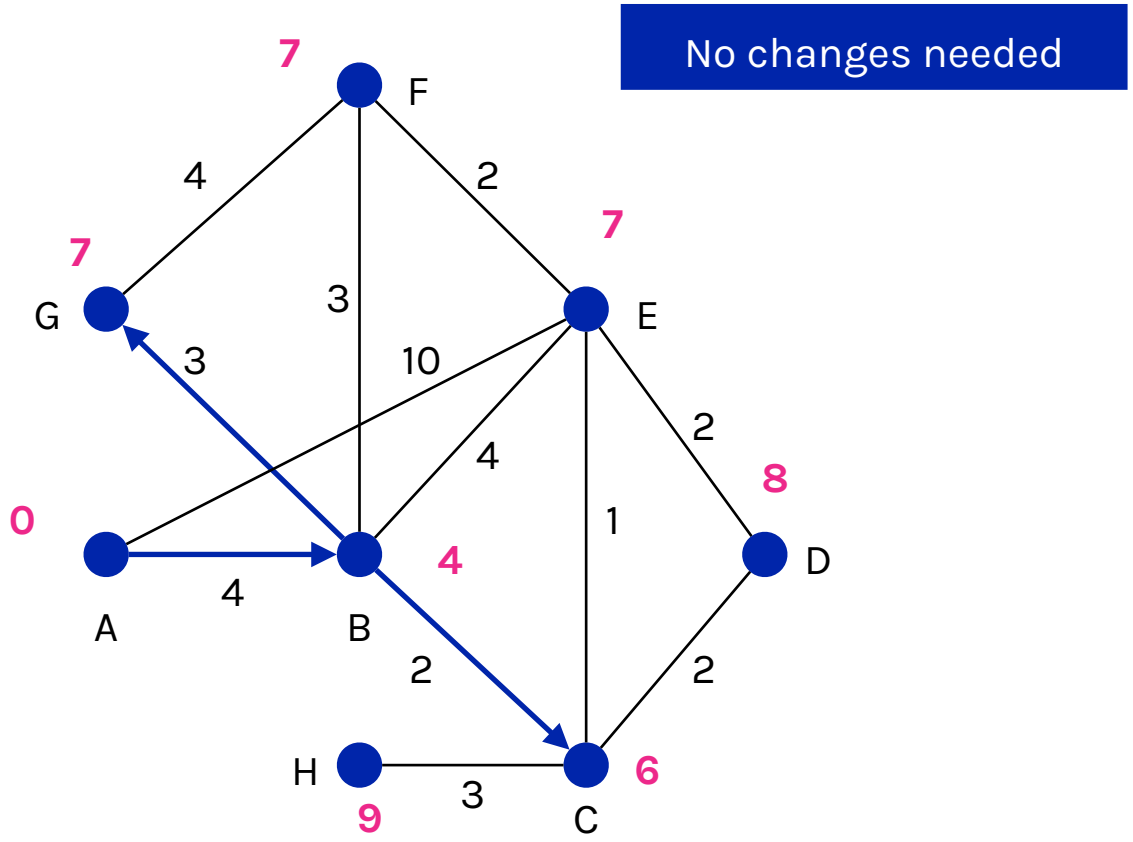
Dijkstra's algorithm example

Relax around C



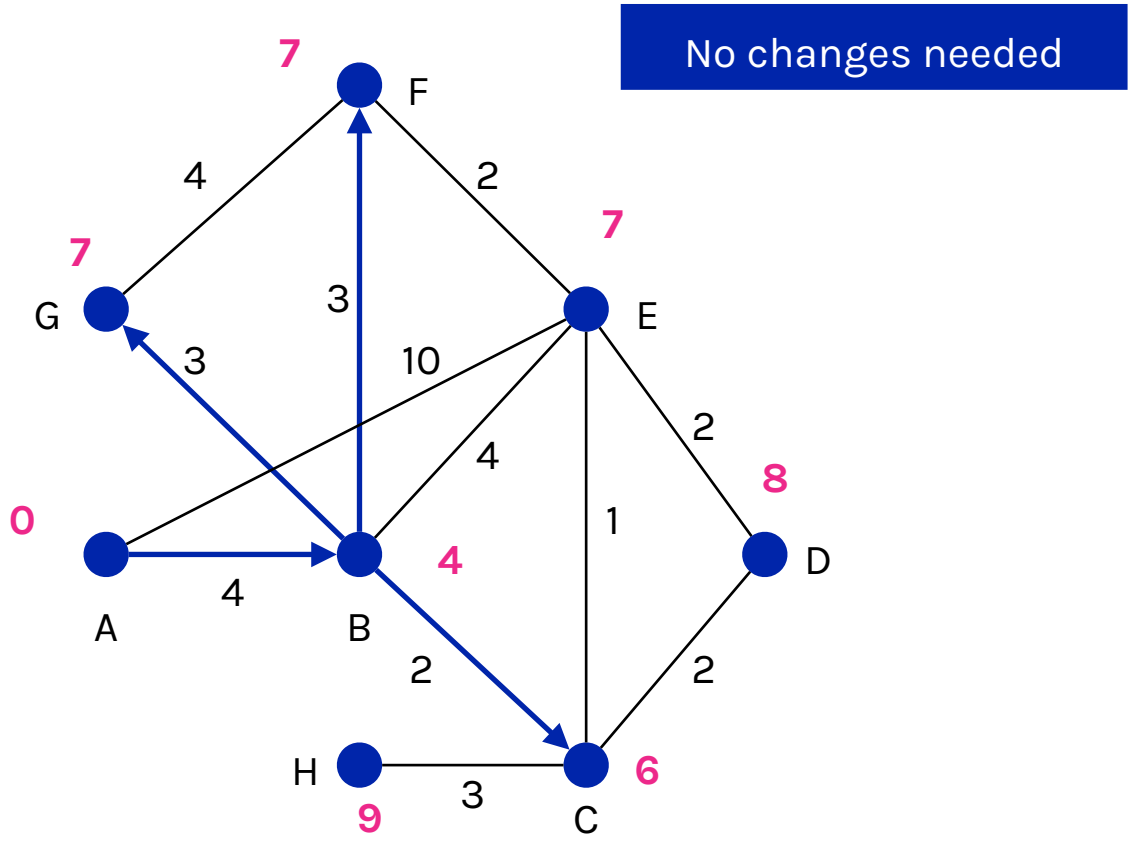
Dijkstra's algorithm example

Relax around G



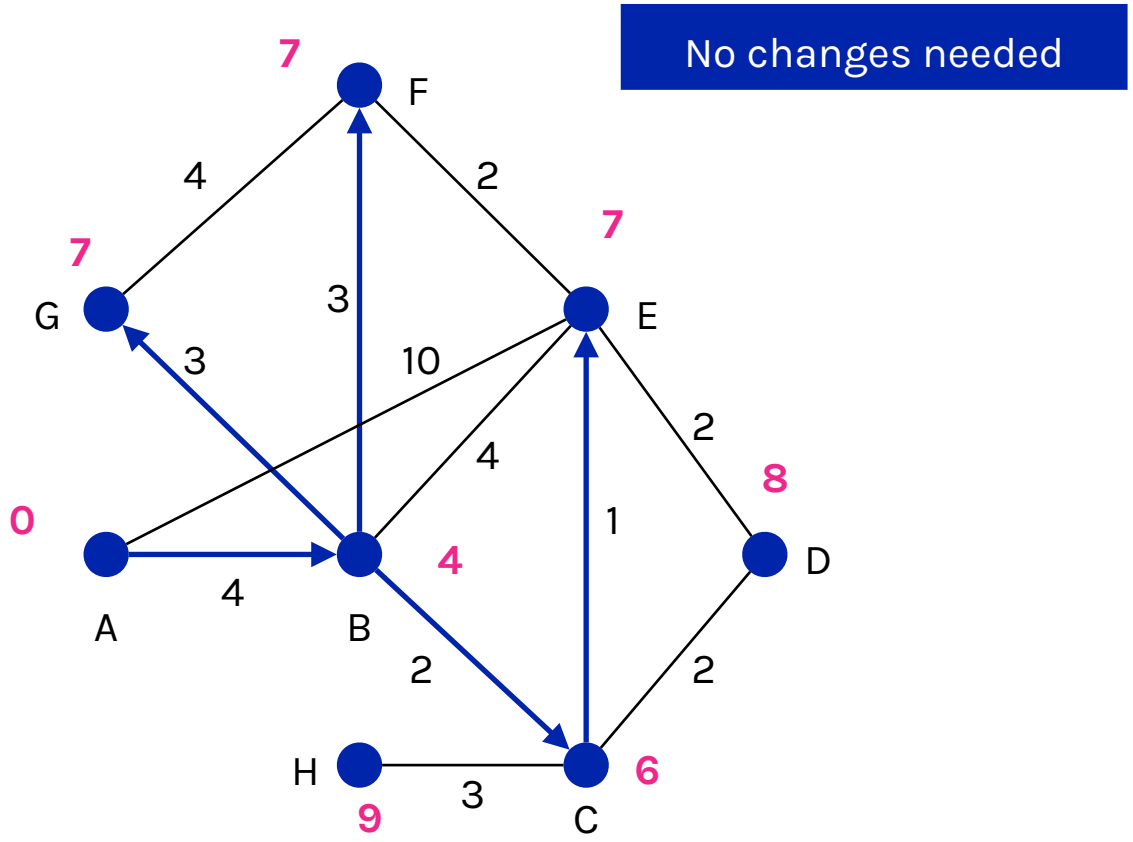
Dijkstra's algorithm example

Relax around F



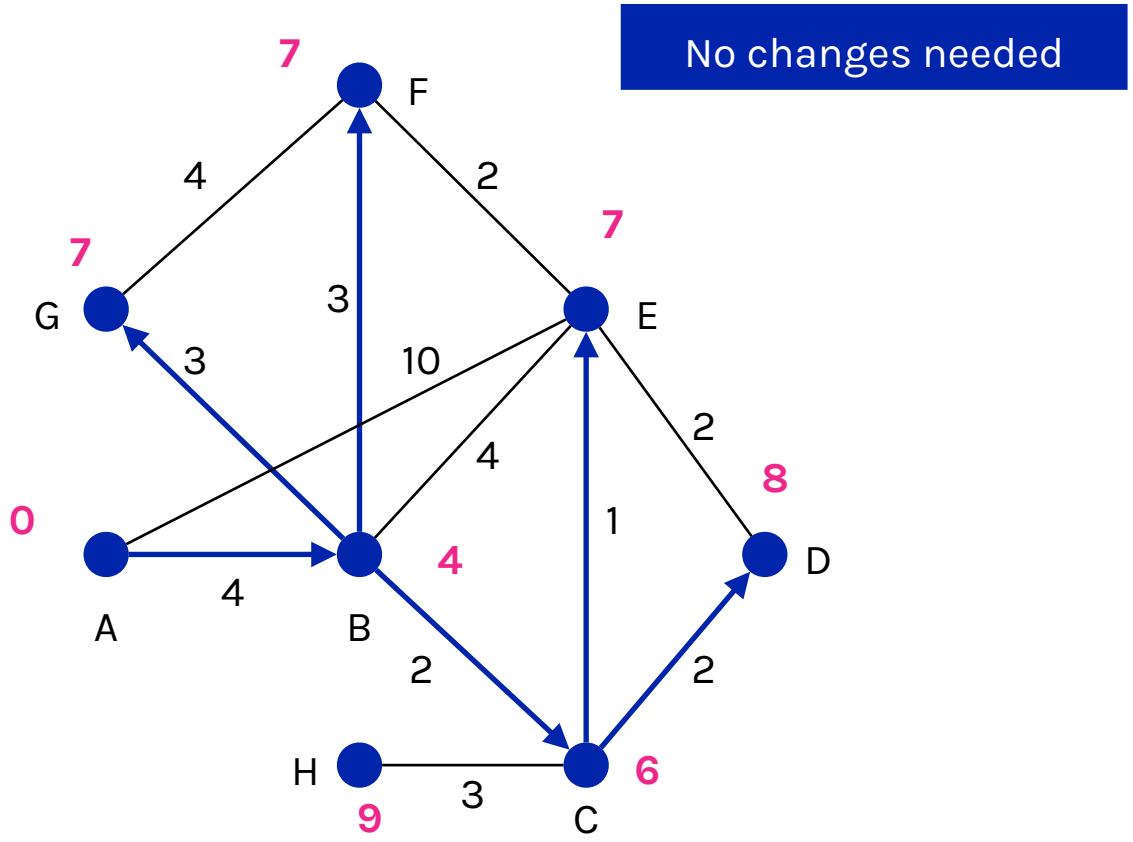
Dijkstra's algorithm example

Relax around E



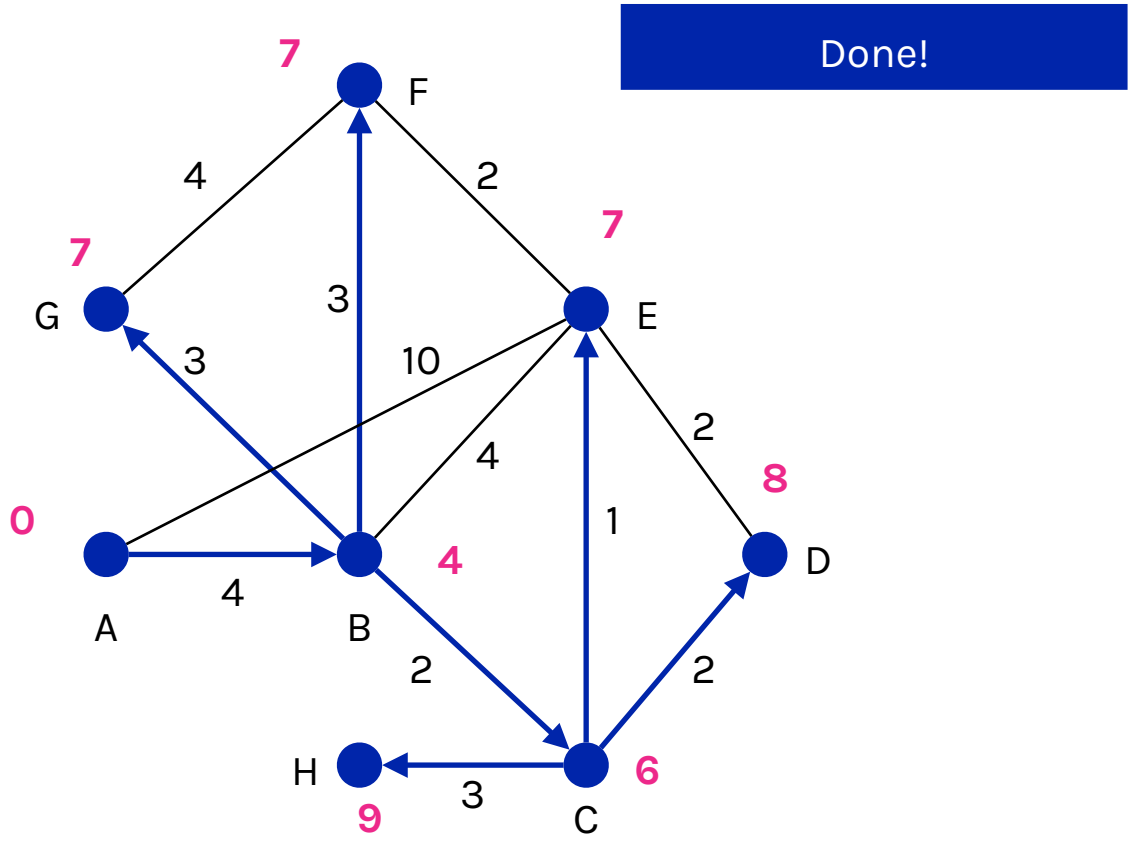
Dijkstra's algorithm example

Relax around D



Dijkstra's algorithm example

Relax around H



Dijkstra's algorithm remarks

Finds shortest paths in order of increasing distance from source

- Leverages the optimality property

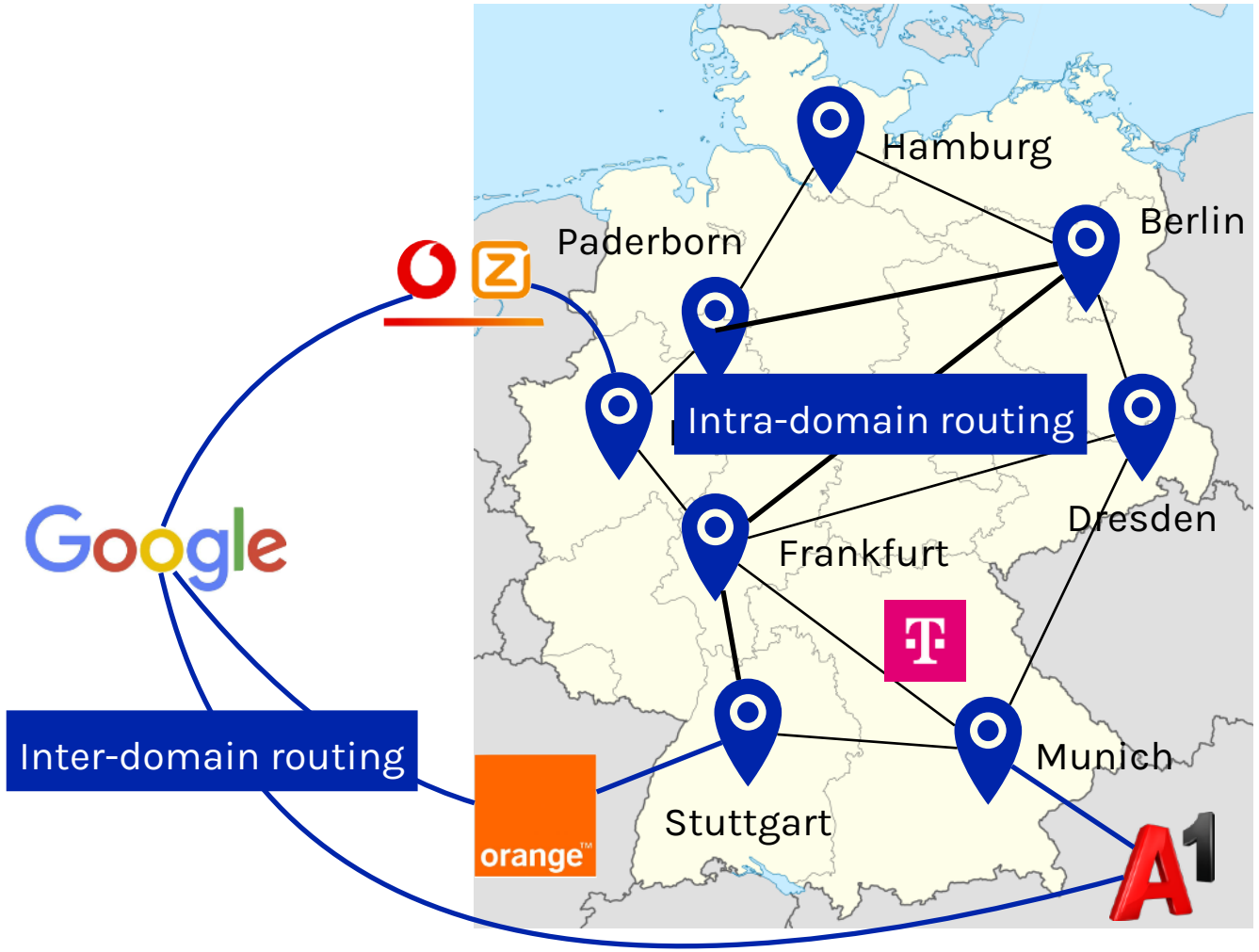
Runtime depends on efficiency of extracting min-cost node

- Superlinear in network size (grows fast), worst case $O(n^2)$ where n is the number of nodes in the network

Gives the complete source/sink tree

- More than needed for forwarding
- Requires the complete topology

Internet routing problems



Internet routing problems

```
traceroute to www.google.de (142.250.186.99), 64 hops max, 52 byte packets
 1  192.168.2.1 (192.168.2.1)  3.151 ms  3.217 ms *
 2  p3e9bf336.dip0.t-ipconnect.de (62.155.243.54)  58.827 ms  14.692 ms  4.476 ms
 3  f-ed11-i.f.de.net.dtag.de (217.5.116.94)  13.844 ms  10.749 ms  11.293 ms
 4  80.156.160.118 (80.156.160.118)  13.974 ms  13.043 ms  12.354 ms
 5  * * *
 6  142.250.214.196 (142.250.214.196)  13.960 ms
    142.251.64.180 (142.251.64.180)  11.910 ms
    142.250.236.36 (142.250.236.36)  11.192 ms
 7  108.170.252.83 (108.170.252.83)  11.742 ms  11.915 ms  11.555 ms
 8  fra24s06-in-f3.1e100.net (142.250.186.99)  11.227 ms  11.496 ms  13.420 ms
```

Intra-domain routing

Inter-domain routing

Intra-Domain Routing

Link-state protocol

Routers build a precise map of the network by flooding local views to everyone

- Each router keeps track of its incident links and cost (and up/down status)
- Each router broadcasts its own links state to give every router a complete view of the graph
- Routers run Dijkstra's algorithm on the corresponding graph to compute their shortest paths and forwarding tables

Flooding is performed as in L2 learning

- Every node sends its link state on all its links
- Next node does the same, except on the link where the information arrived

Reliability

All nodes are ensured to receive the latest version of all link states

Challenges

- Packet loss
- Out of order arrival

Solutions

- ACK and retranmissions
- Sequence number
- Time-to-live (TTL) for each link state

Flooding conditions

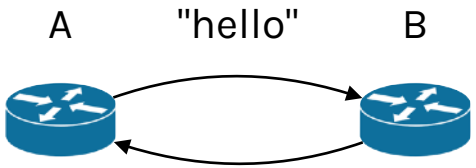
Topology change (caused by link/node failure/recovery)

Configuration change (caused by link cost change)

Periodically (to refresh the link state information)

Detecting topology changes

Using software-based beaconing



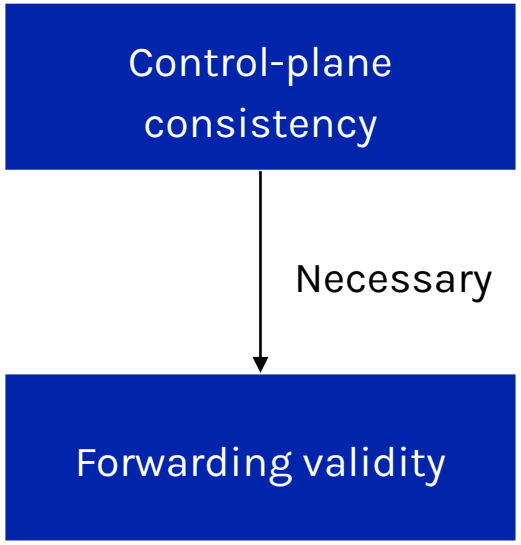
Routers periodically exchange "hello" messages in both directions (e.g., every 30s)

A failure is triggered if few "hello" messages are missed in a row (e.g., after 3 missed ones)

Tradeoffs between:

- Detection speed
- Bandwidth and CPU overhead
- False positive/negatives

Consistency is important

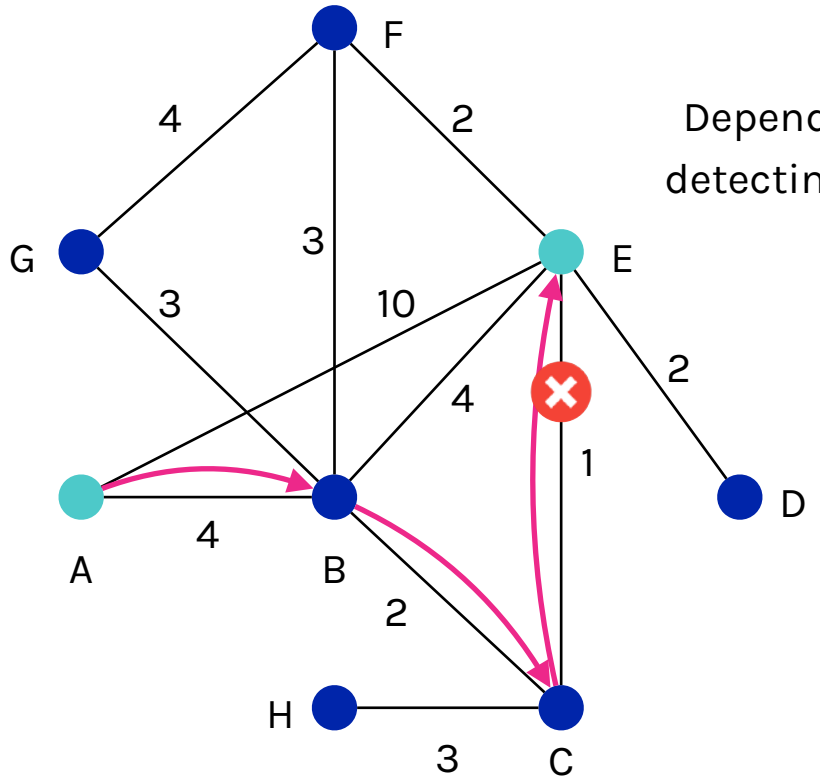


All nodes have the same link-state database

The global forwarding state directs packet to its destination

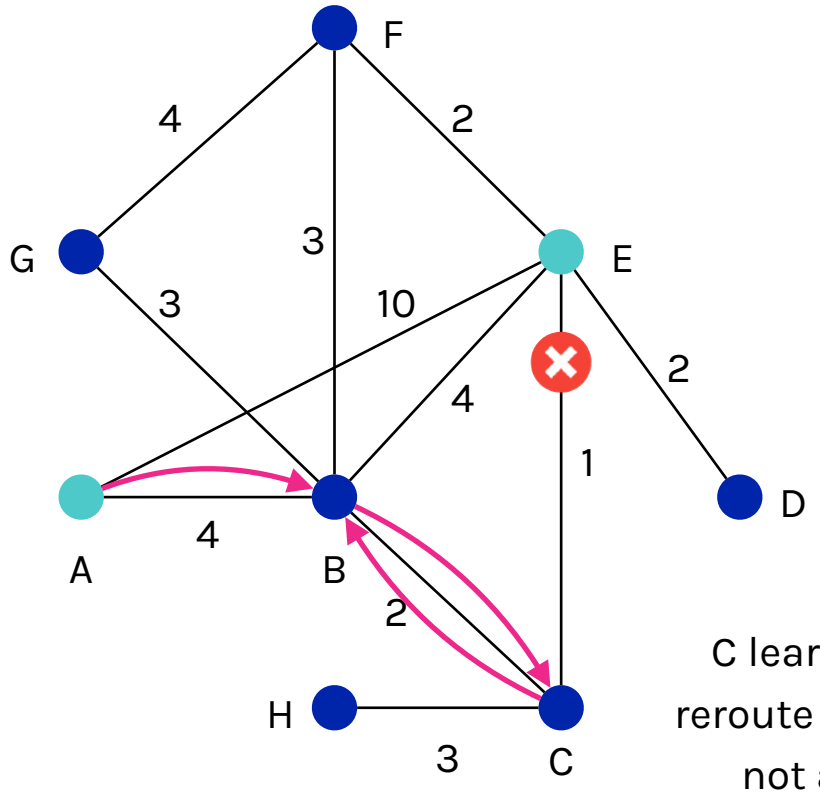
Hard to ensure consistency during network changes

Inconsistency: black holes



Depending on the timeout for detecting lost "hello" messages

Inconsistency: transient loops



C learns about the failure and reroute to B immediately, but B is not aware of the failure yet

Two link-state protocols

OSPF

Open Shortest Path First

Used in many enterprise / ISPs
Works on top of IP
Only route IPv4 by default

IS-IS

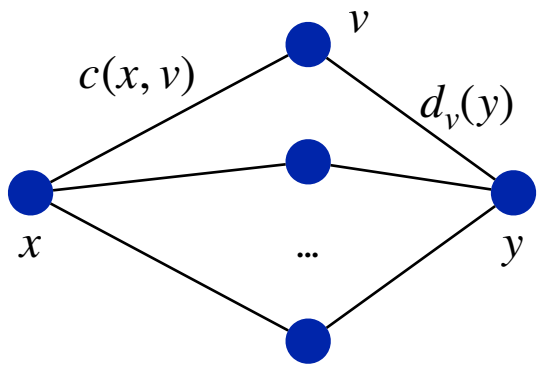
Intermediate Systems

Used mostly in large ISPs
Works on top of link layer
Network protocol agnostic

Distance-vector protocols

Based on the Bellman-Ford algorithm

- Let $d_x(y)$ be the cost of the least-cost path known by x to reach y
- Until convergence: each node bundles these distances into one message (called a vector) that it repeatedly sends to all its neighbors
- Each node updates its distances based on neighbor's vectors



$$d_x(y) = \min_v c(x,v) + d_v(y)$$

https://en.wikipedia.org/wiki/Bellman–Ford_algorithm

Conditions for sending new distance vectors

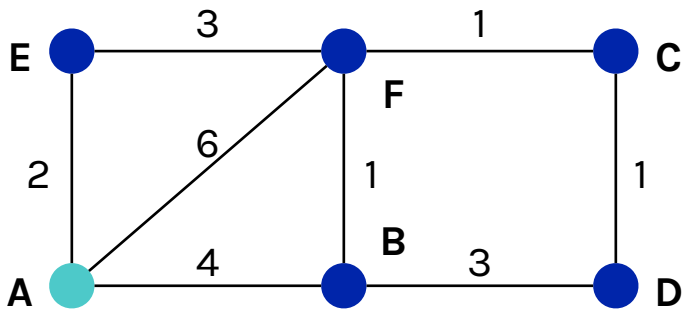
Similarly done as in the link-state protocols

Topology change (caused by link/node failure/recovery)

Configuration change (caused by link cost change)

Periodically (to refresh the link state information)

Distance-vector protocol example: one-hop



A		
Dst	**Cost**	**Next**
A	0	A
B	4	B
C	Inf	-
D	Inf	-
E	2	E
F	6	F

B		
Dst	**Cost**	**Next**
A	4	A
B	0	B
C	Inf	-
D	3	D
E	Inf	-
F	1	F

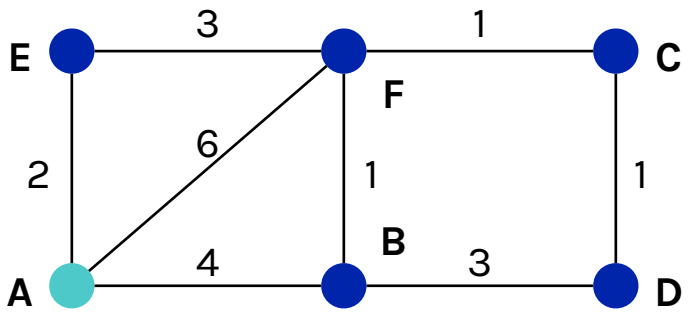
C		
Dst	**Cost**	**Next**
A	Inf	-
B	Inf	-
C	0	C
D	1	D
E	Inf	-
F	1	F

D		
Dst	**Cost**	**Next**
A	Inf	-
B	3	B
C	1	C
D	0	D
E	Inf	-
F	Inf	-

E		
Dst	**Cost**	**Next**
A	2	A
B	Inf	-
C	Inf	-
D	Inf	-
E	0	E
F	3	F

F		
Dst	**Cost**	**Next**
A	6	A
B	1	B
C	1	C
D	Inf	-
E	3	E
F	0	F

Distance-vector protocol example: two-hop



A		
Dst	**Cost**	**Next**
A	0	A
B	4	B
C	7	F
D	7	B
E	2	E
F	5	E

B		
Dst	**Cost**	**Next**
A	4	A
B	0	B
C	2	F
D	3	D
E	4	F
F	1	F

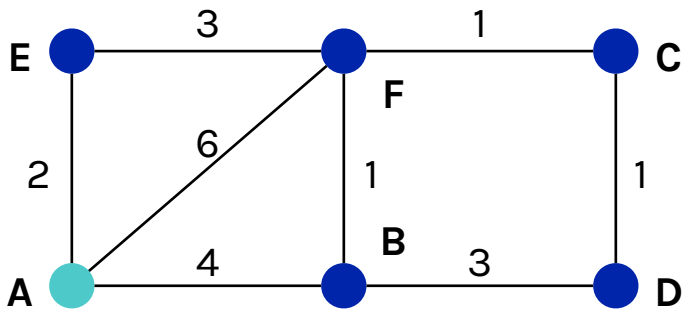
C		
Dst	**Cost**	**Next**
A	7	F
B	2	F
C	0	C
D	1	D
E	4	F
F	1	F

D		
Dst	**Cost**	**Next**
A	7	B
B	3	B
C	1	C
D	0	D
E	Inf	-
F	2	C

E		
Dst	**Cost**	**Next**
A	2	A
B	4	F
C	4	F
D	Inf	-
E	0	E
F	3	F

F		
Dst	**Cost**	**Next**
A	5	B
B	1	B
C	1	C
D	2	C
E	3	E
F	0	F

Distance-vector protocol example: three-hop



A		
Dst	**Cost**	**Next**
A	0	A
B	4	B
C	6	E
D	7	B
E	2	E
F	5	E

B		
Dst	**Cost**	**Next**
A	4	A
B	0	B
C	2	F
D	3	D
E	4	F
F	1	F

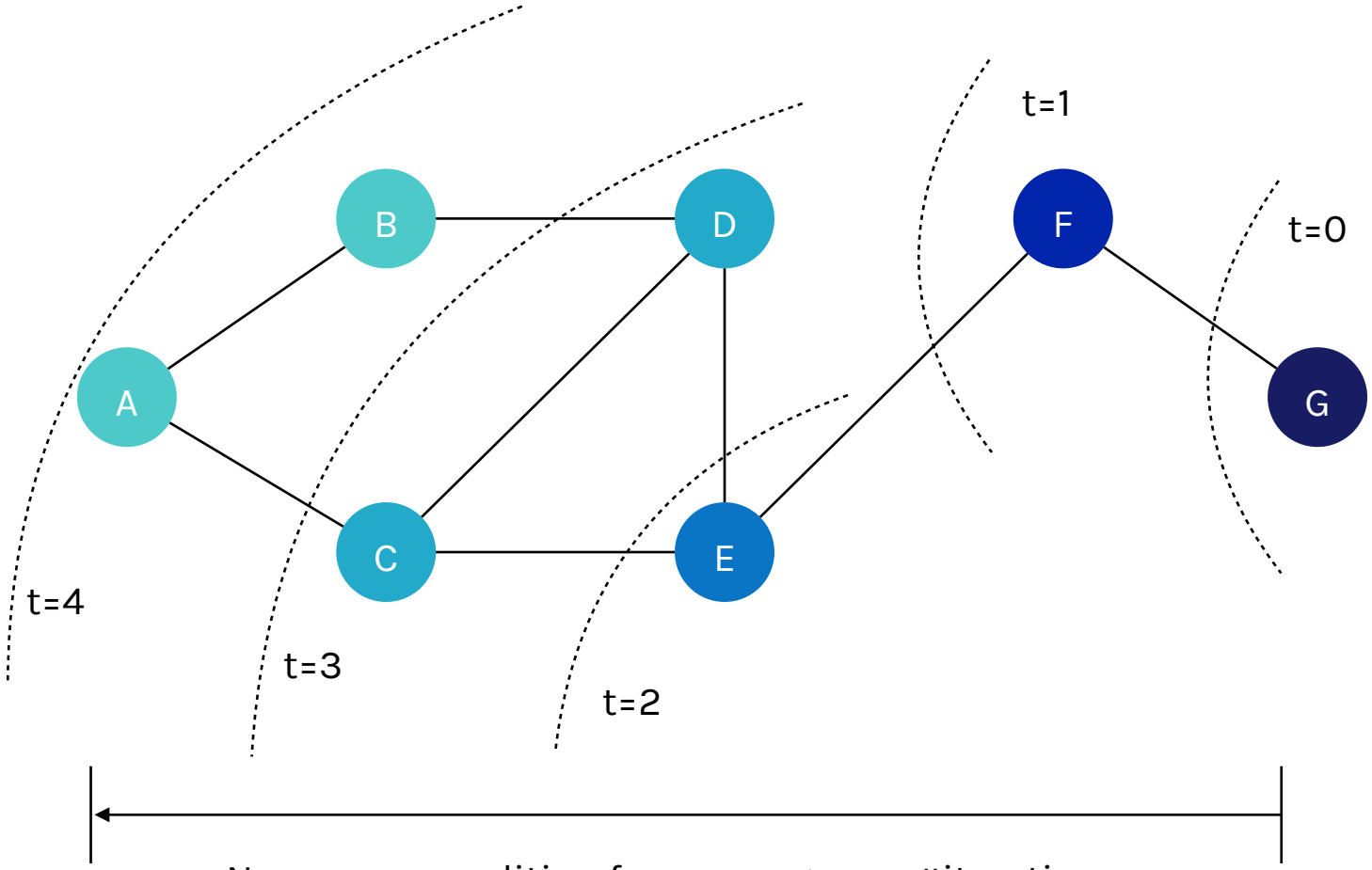
C		
Dst	**Cost**	**Next**
A	6	F
B	2	F
C	0	C
D	1	D
E	4	F
F	1	F

D		
Dst	**Cost**	**Next**
A	7	B
B	3	B
C	1	C
D	0	D
E	5	C
F	2	C

E		
Dst	**Cost**	**Next**
A	2	A
B	4	F
C	4	F
D	5	F
E	0	E
F	3	F

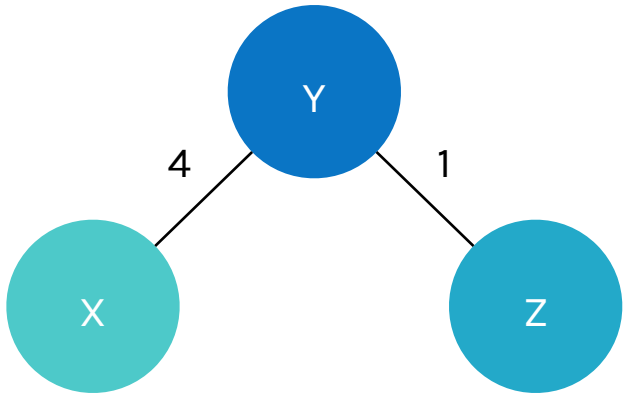
F		
Dst	**Cost**	**Next**
A	5	B
B	1	B
C	1	C
D	2	C
E	3	E
F	0	F

Convergence analysis



Necessary condition for convergence: #iterations no smaller than the diameter of the network

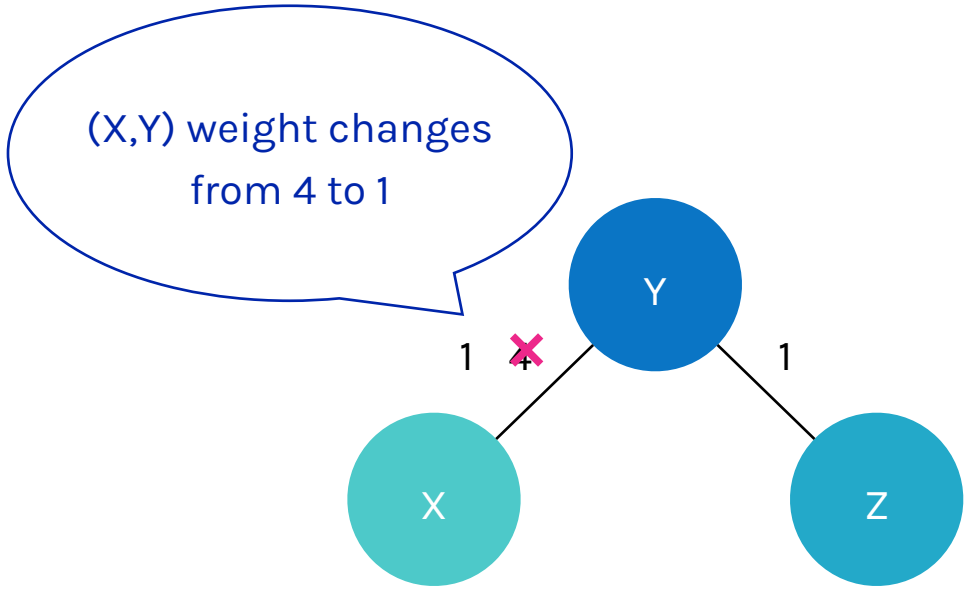
Convergence process: initial state



Vector at Y	Destination	Distance
	X	4
	Z	1

Vector at Z	Destination	Distance
	X	5
	Y	1

Convergence process: t=0

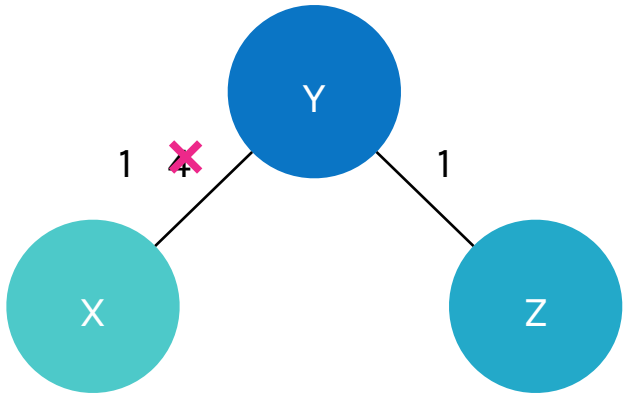


Vector at Y	Destination	Distance
	X	4
	Z	1

Vector at Z	Destination	Distance
	X	5
	Y	1

Node detects local cost change, update their vectors, and notify their neighbors if it has changed

Convergence process: t=1

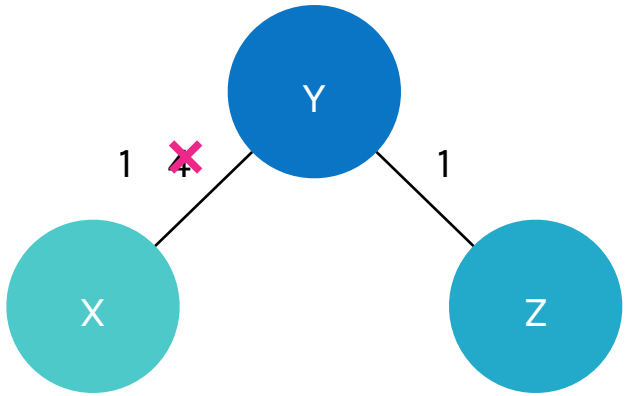


Vector at Y	Destination	Distance
	X	1
	Z	1

Y updates its vector, sends it to X and Z

Vector at Z	Destination	Distance
	X	5
	Y	1

Convergence process: t=2



Vector at Y	Destination	Distance
	X	1
	Z	1

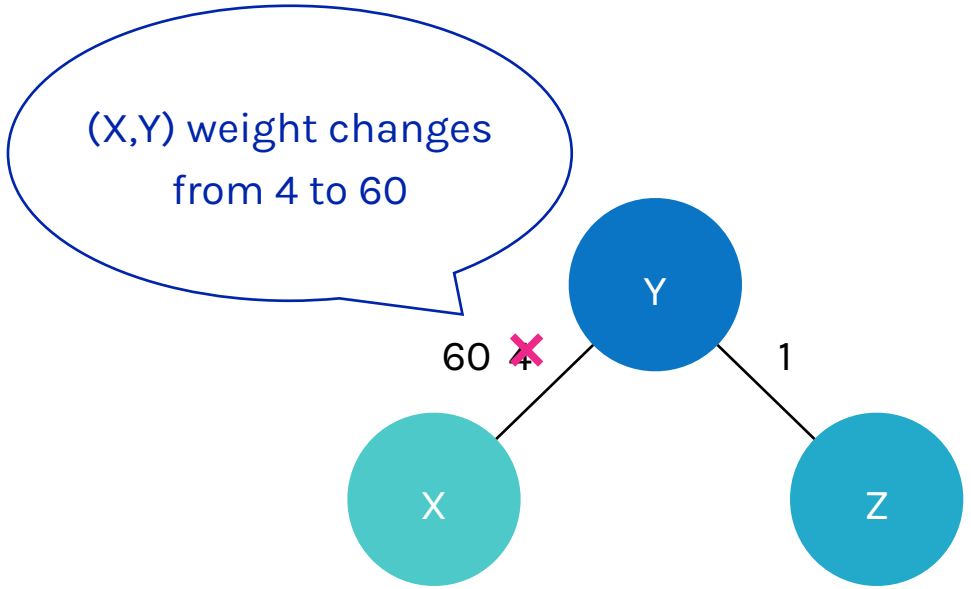
Vector at Z	Destination	Distance
	X	2
	Y	1

Z updates its vector, sends it to Y

No one moves any more, network has converged!

Good news travels fast!

Convergence process: t=0

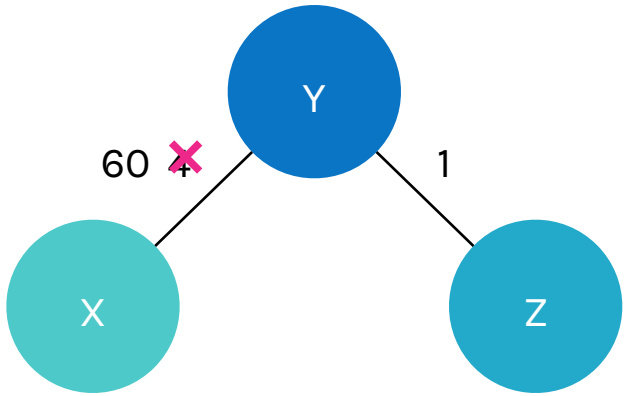


Vector at Y	Destination	Distance
	X	4
	Z	1

Vector at Z	Destination	Distance
	X	5
	Y	1

Node detects local cost change, update their vectors, and notify their neighbors if it has changed

Convergence process: t=1

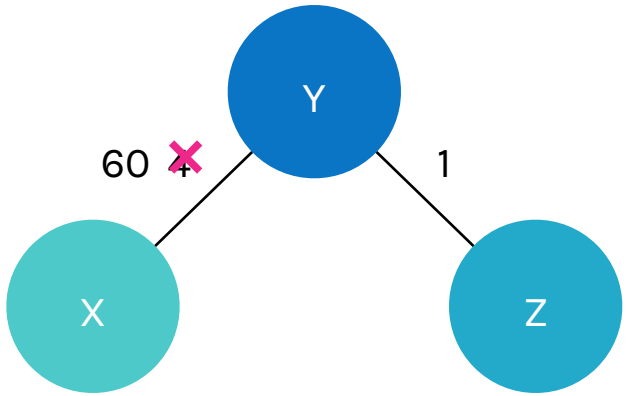


Vector at Y	Destination	Distance
	X	60
	Z	1

Y updates its vector, sends it to X and Z

Vector at Z	Destination	Distance
	X	5
	Y	1

Convergence process: t=2

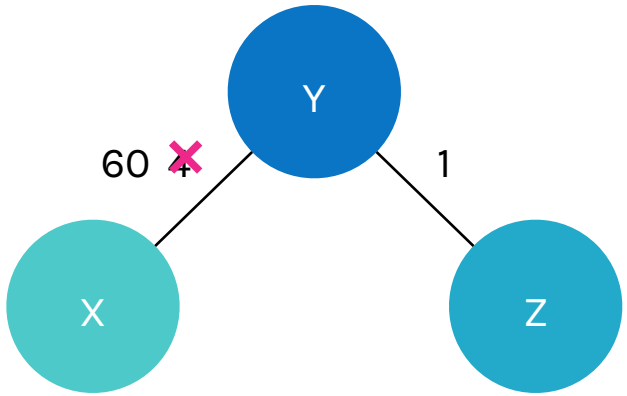


Vector at Y	Destination	Distance
	X	60
	Z	1

Vector at Z	Destination	Distance
	X	5
	Y	1

Z notices that it has a shorter path to X, send this info to Y

Convergence process: t=3



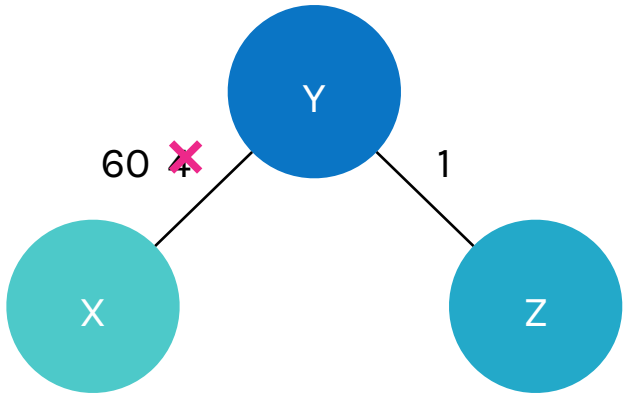
Vector at Y	Destination	Distance
	X	6
	Z	1

Y updates its vector, sends it to X and Z

Vector at Z	Destination	Distance
	X	5
	Y	1

There is no way for Y to tell if the distance vector advertised by Z is through itself

Convergence process: t=4



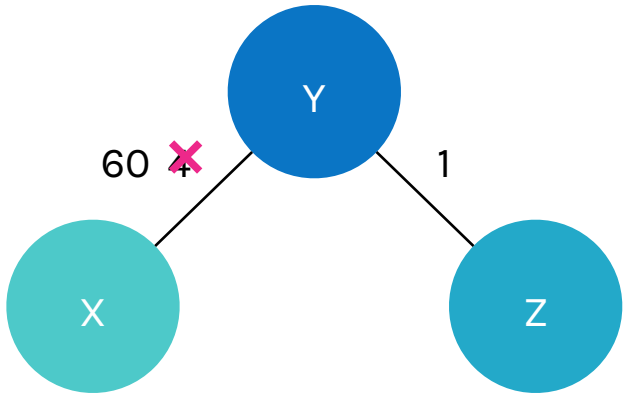
Vector at Y	Destination	Distance
	X	6
	Z	1

Vector at Z	Destination	Distance
	X	7
	Y	1

Z updates its distance vector and send it to Y

Z does not know that Y has switched to use itself

Convergence process: t=5



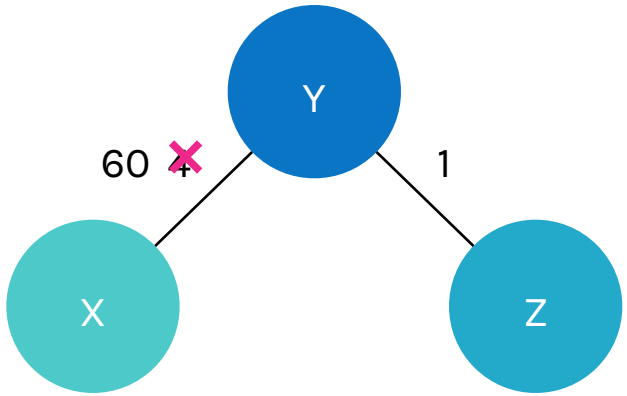
Vector at Y	Destination	Distance
	X	8
	Z	1

Y updates its vector, sends it to X and Z

Vector at Z	Destination	Distance
	X	7
	Y	1

This process will go on for quite a while until…

Convergence process: t>58



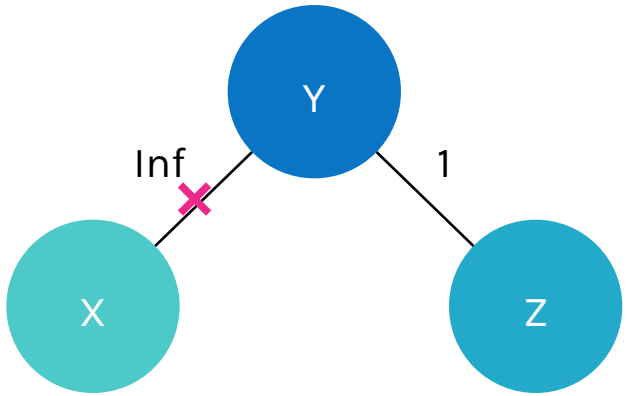
Vector at Y	Destination	Distance
	X	60
	Z	1

Vector at Z	Destination	Distance
	X	61
	Y	1

Y and Z realize that the link with increased cost is actually the (only) better option

Bad news travels slowly!

Count-to-infinity



Vector at Y	Destination	Distance
	X	Inf
	Z	1

Vector at Z	Destination	Distance
	X	Inf
	Y	1

If the link X-Y is down, the process will go on infinitely

Potential mitigations to count-to-infinity

Approximation of infinity

- For example, hop count is considered infinity when it reaches 16
- Limited to a small network

Split horizon (with poison reverse)

- A node does not send routes (or sends infinity) to nodes from which the routes were learned
- Limited to loops involving two nodes

Routing information protocol (RIP)

An implementation of distance-vector routing for inter-network connections

- First protocol used by ARPANET (the forerunner of the Internet)
- Simply, easy to implement
- A cost of 16 represents infinity (poor scalability)
- Count-to-infinity issue (rarely used nowadays)

https://datatracker.ietf.org/doc/html/rfc2453

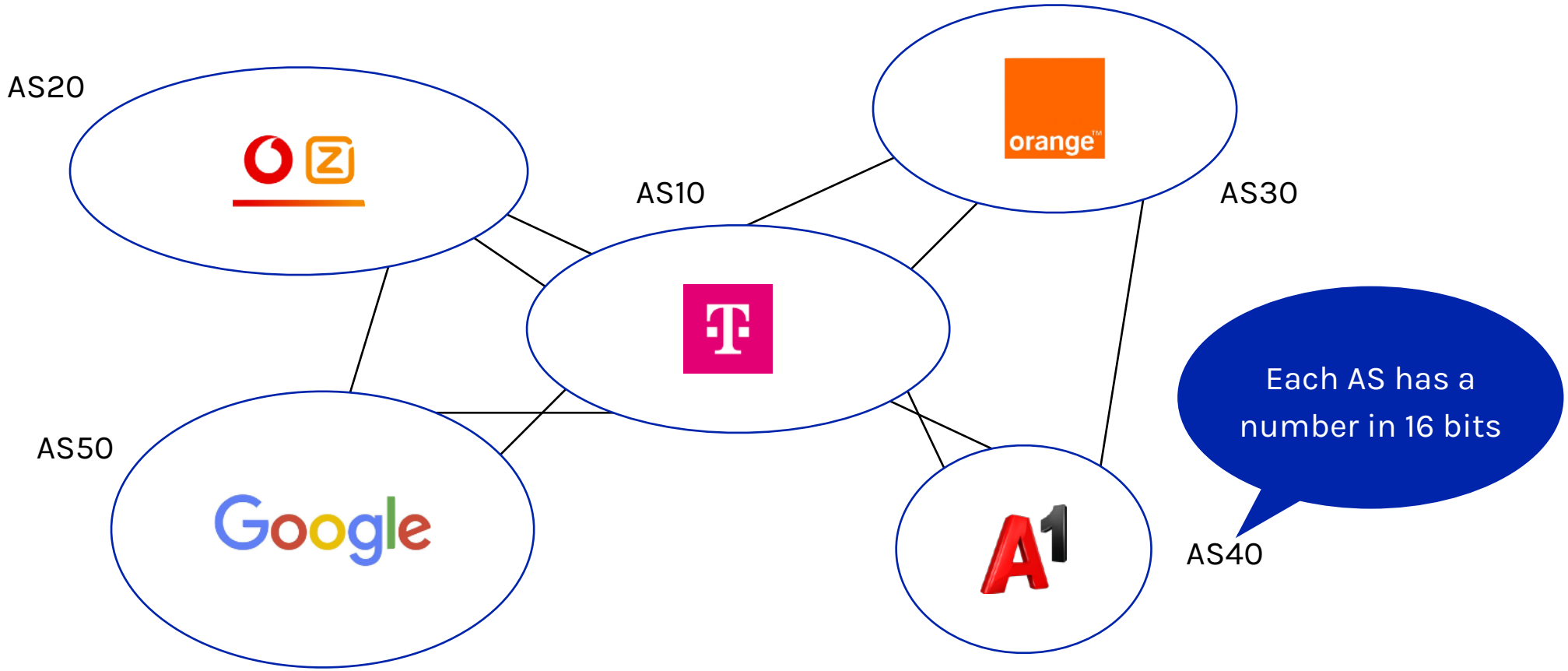
Link-state vs. distance-vector routing

	Message complexity	Convergence speed	Robustness
Link state	$O(nE)$	Relatively fast	Node can advertise incorrect link cost (but compute its own table)
Distance vector	Between neighbors	Slow	Node can advertise incorrect path cost (errors propagate)

n: number of nodes, E: number of links

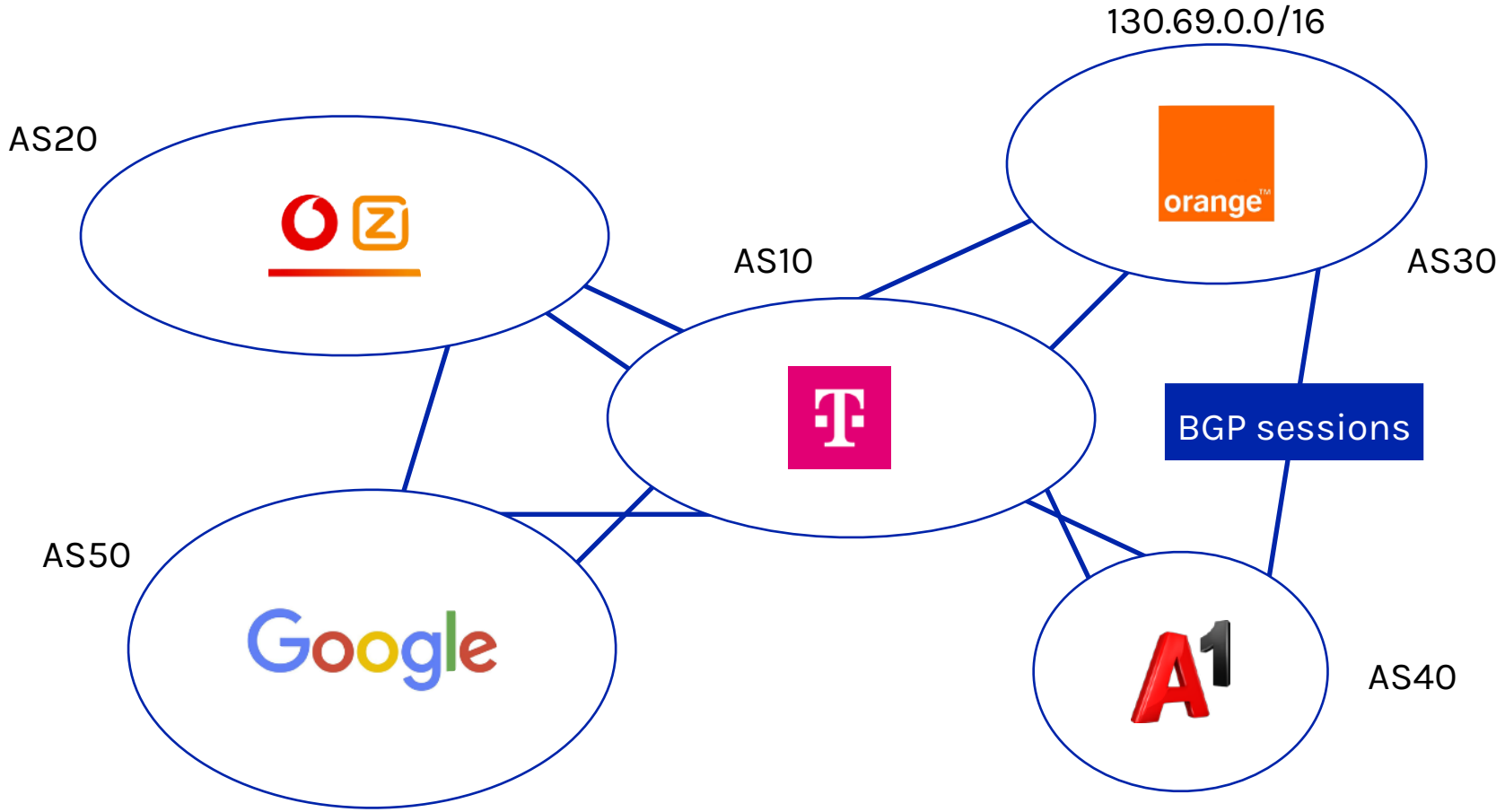
Inter-Domain Routing

Internet: a network of networks in ASes



AS: Autonomous System

Border Gateway Protocol (BGP)



ASes use BGP to exchange information about the IP prefixes they can reach, directly or indirectly

BGP challenges

High number of networks and IP prefixes

- 1M prefixes, more than 70K networks, millions of routers

Networks do not want to divulge internal topologies

- Or their business relationships

Networks need to control where to send and receive traffic

- Without an Internet-wide notion of a link cost metric

Link-state routing does not solve them

Floods topology information

- High processing overhead

Requires each node to compute the entire graph

- High processing overhead

Minimizes some notion of total distance

- Works only if the policy is shared and uniform across ASes

Distance-vector routing is on the right track

Pros

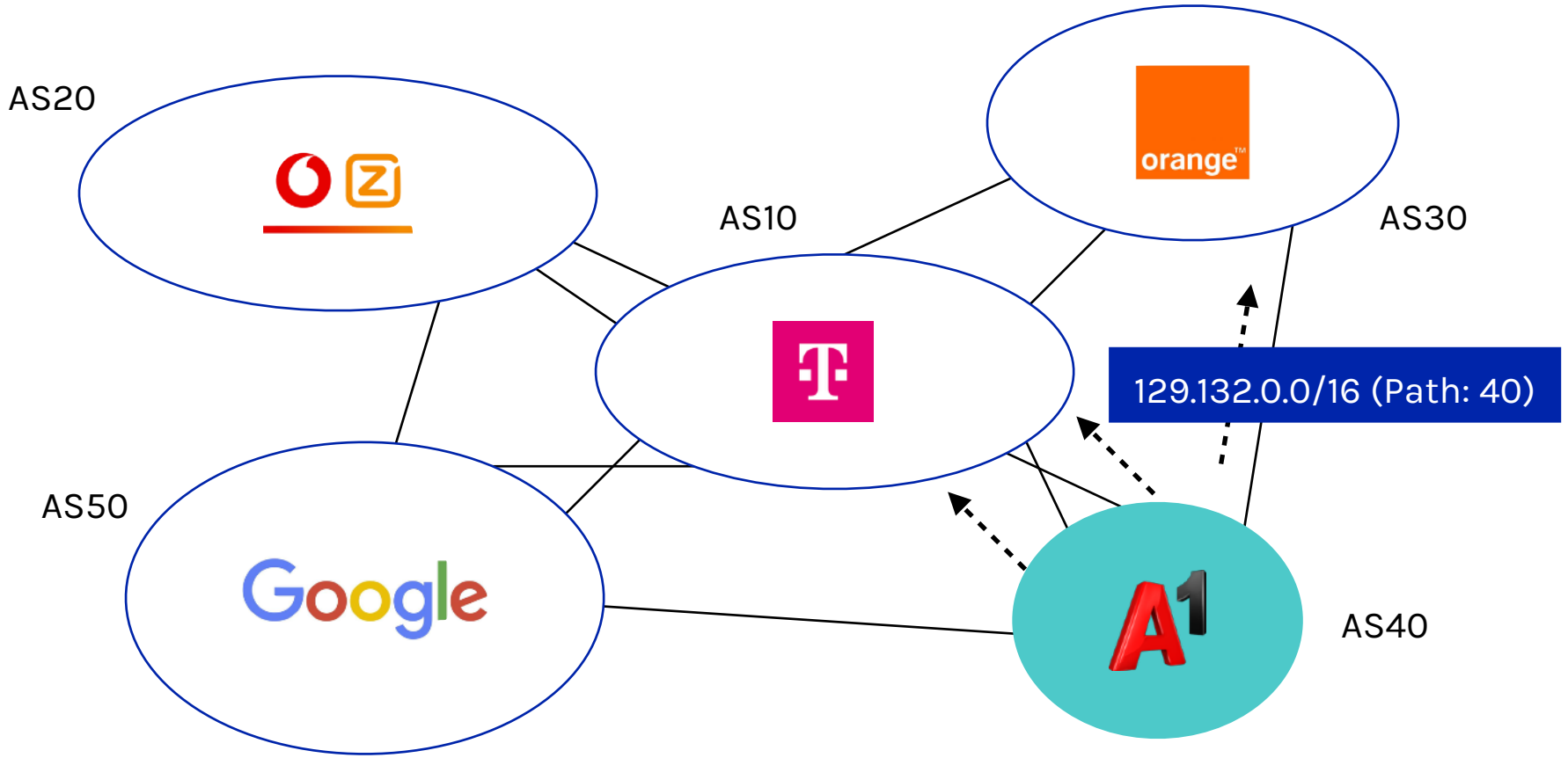
- High details of the network topology: nodes determin only "next-hop" for each destination

Cons

- Still minimizes some common distance: impossible to achieve in an inter-domain setting
- Converges slowly: count-to-infinity problem

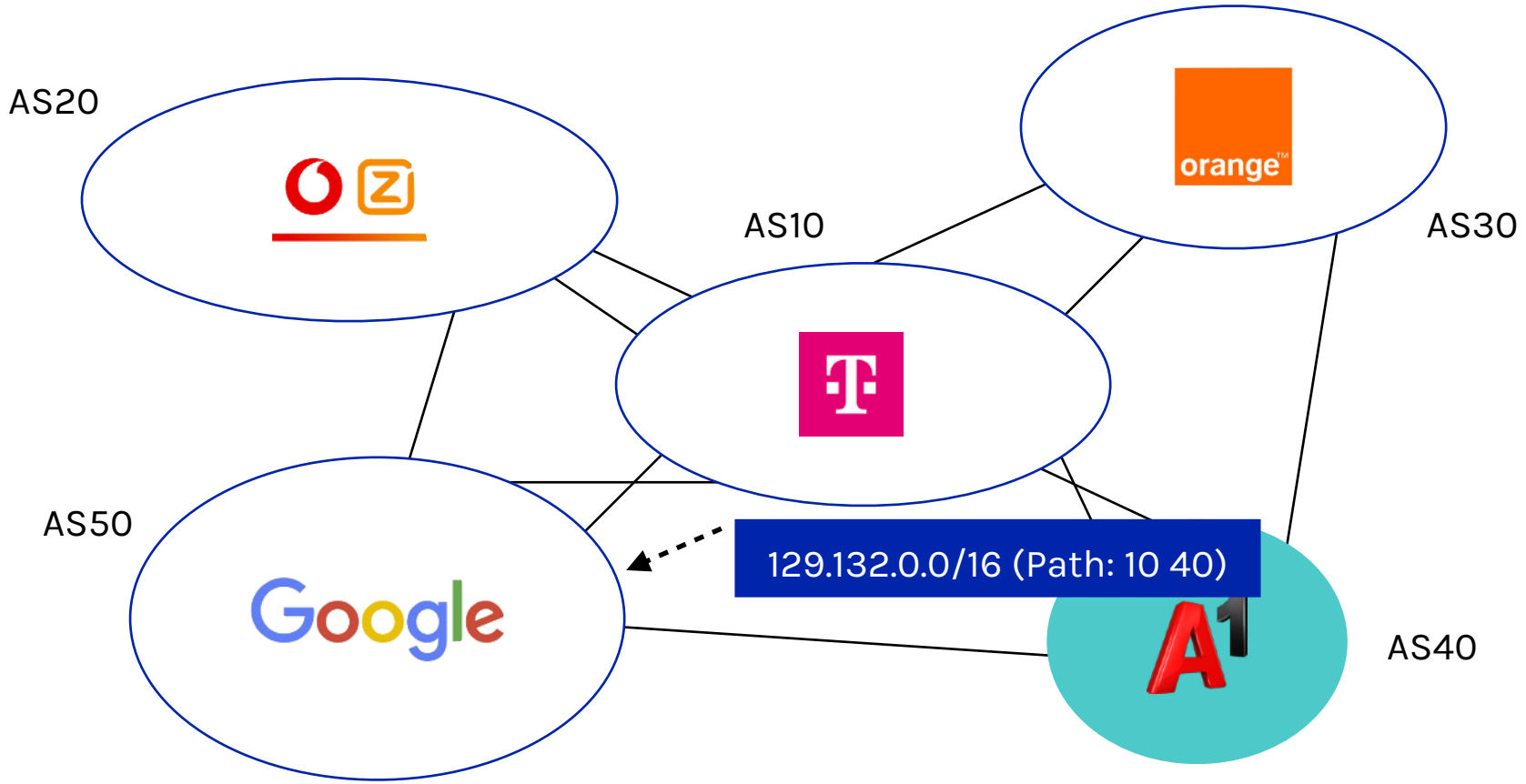
Path-vector routing in BGP

Advertise the entire path instead of distances



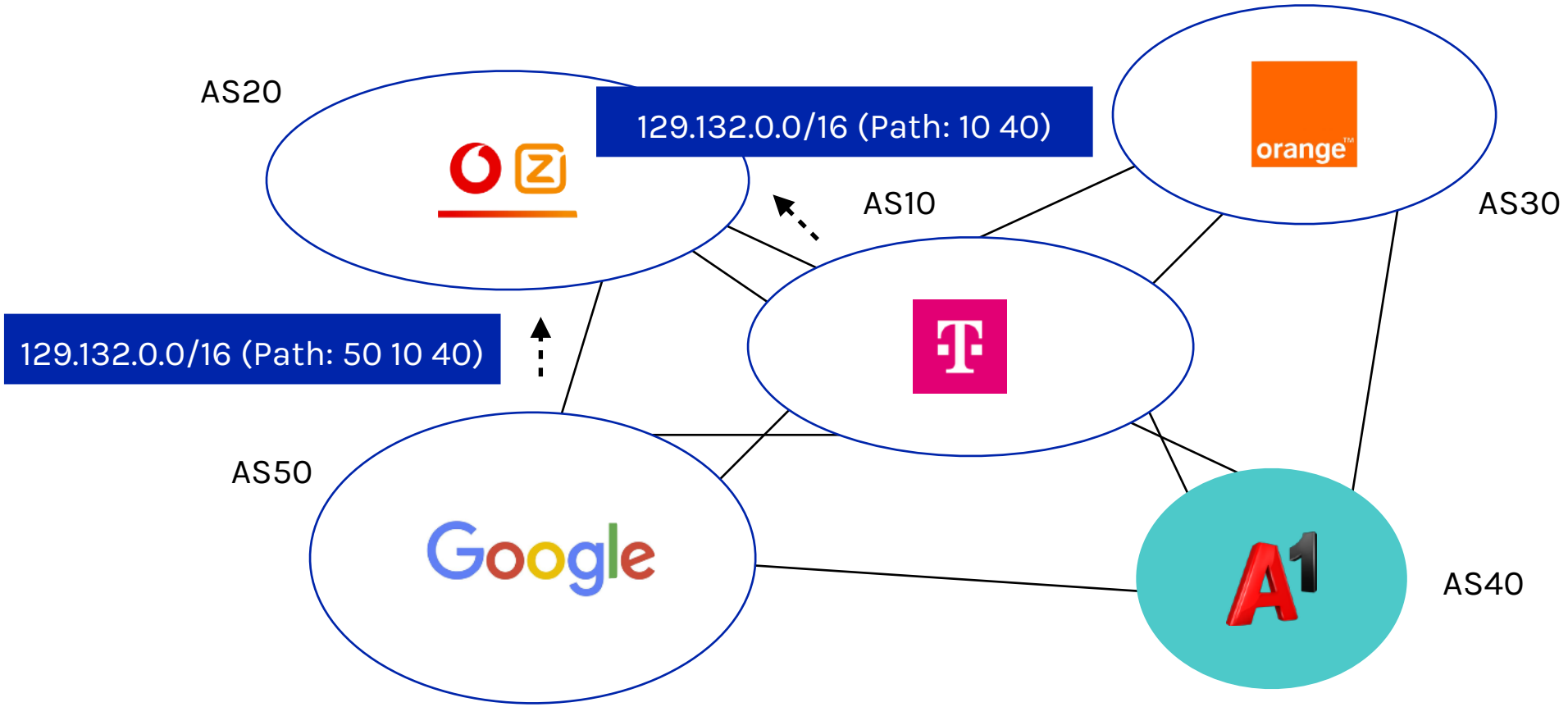
Path-vector routing in BGP

Append the AS ID to the path received when advertising



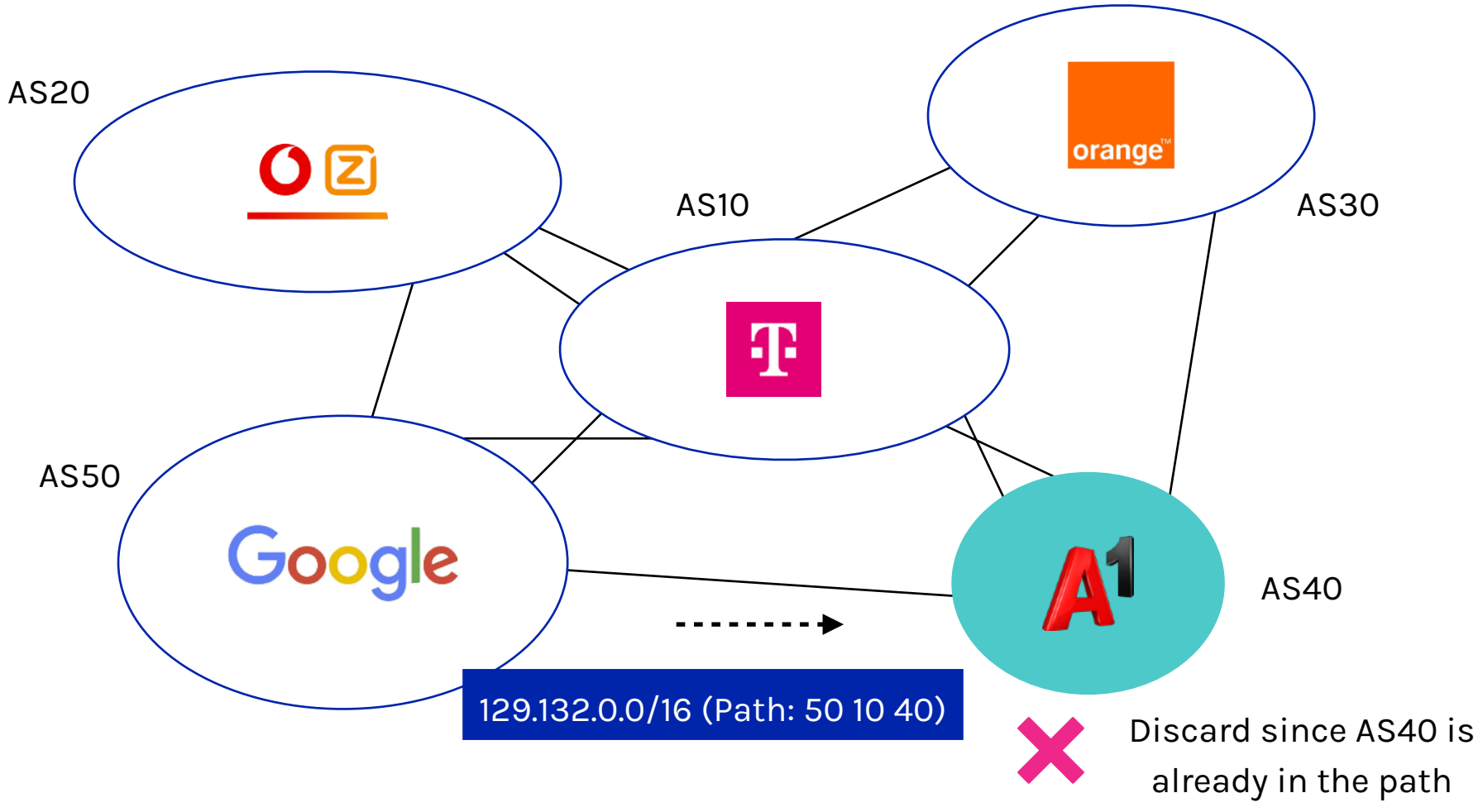
Path-vector routing in BGP

Each AS chooses (NOT) to advertise certain paths



Path-vector routing in BGP

Check for possible loops



Path-vector routing in BGP

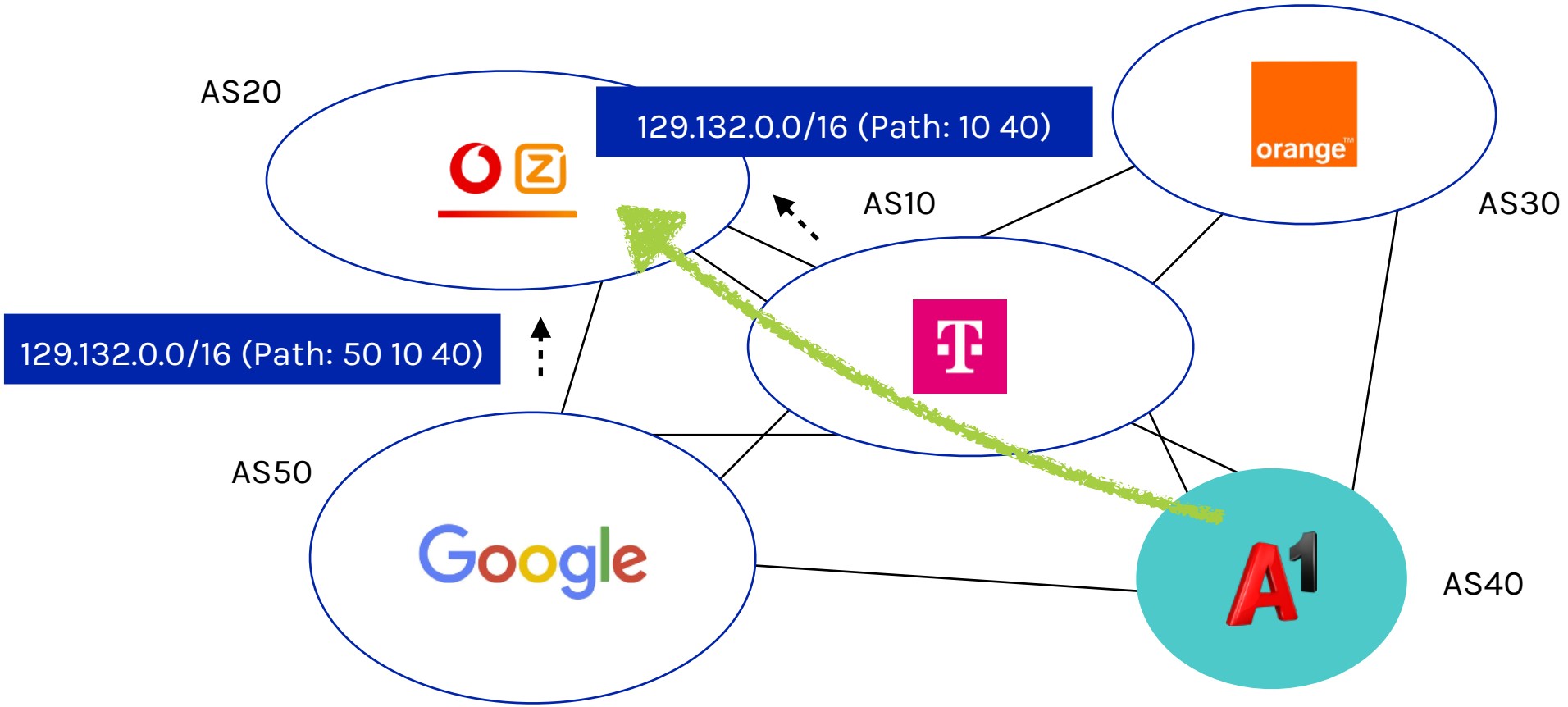
Life of BGP routers

- Receives routes from its neighbors
- Select on best route for each prefix
- Export the best route to its neighbors

Each AS can apply local routing policies and is free to

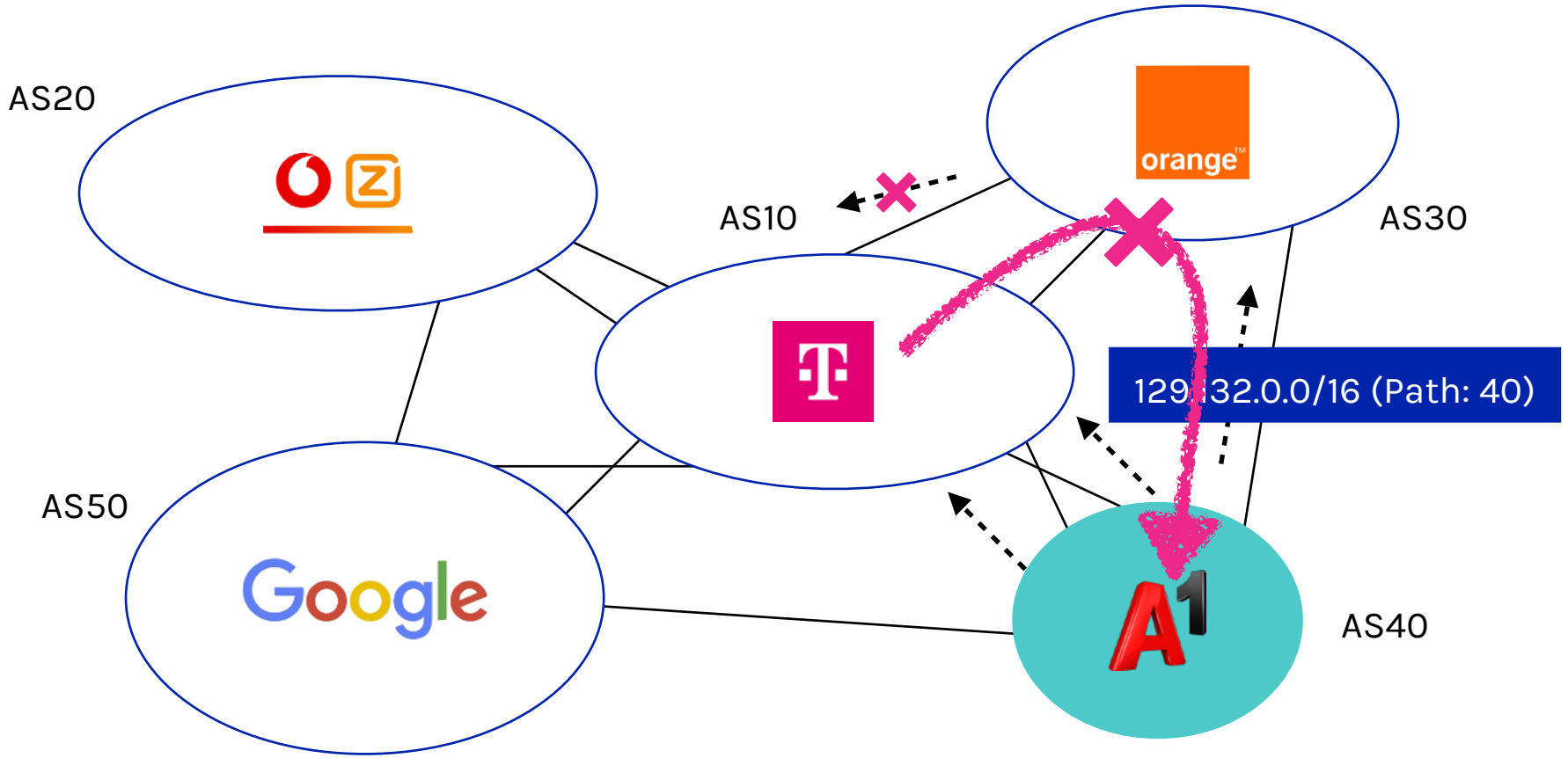
- Select and use any path (preferably, the cheapest one)
- Decide which path to export (if any) to which neighbor (preferably none to minimize carried traffic)

Path-vector routing in BGP



AS20: always prefer Deutsche Telekom over Google

Path-vector routing in BGP



AS30: do not advertise to AS10 to hide the path

Policy and economics play a bit role in BGP!

Innovations in the network layer

Many Future Internet projects 15 years back

- Named Data Networking (NDN)
- eXpressive Internet Architecture (XIA)
- MobilityFirst

Information Centric Networking (ICN)

- Still an open research topic

http://www.nets-fia.net/

Summary

Routing

- Goals and rules
- Best path definition
- Dijkstra's algorithm
- Internet routing

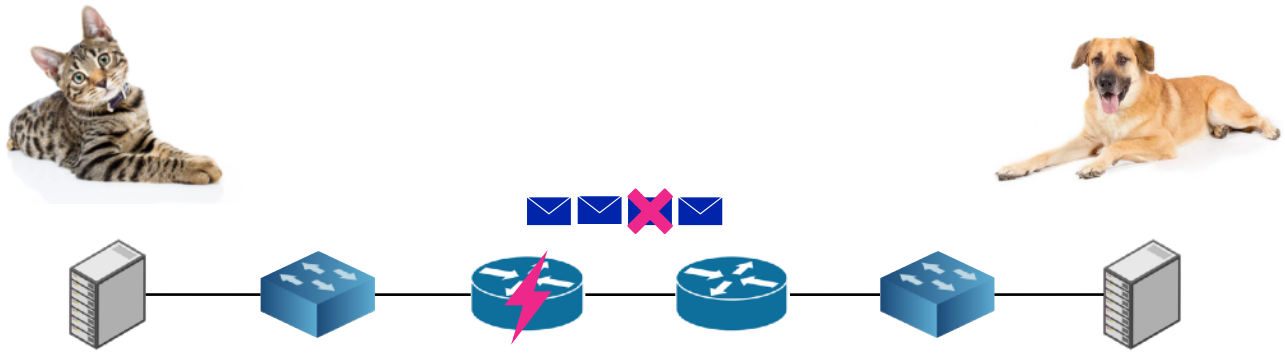
Intra-domain routing

- Link-state protocol
- Distance vector protocol

Inter-domain routing

- BGP
- Path-vector routing

Next time: transport layer



How to ensure reliability?

Further reading material

Andrew S. Tanenbaum, David J. Wetherall. Computer Networks (5th edition).

- Section 5.2: Routing Algorithms
- Section 5.6: The Network Layer in the Internet

Larry Peterson, Bruce Davie. Computer Networks: A Systems Approach.

- Section 3.4: Routing
- Section 4.1: Global Internet