

Communication Networks

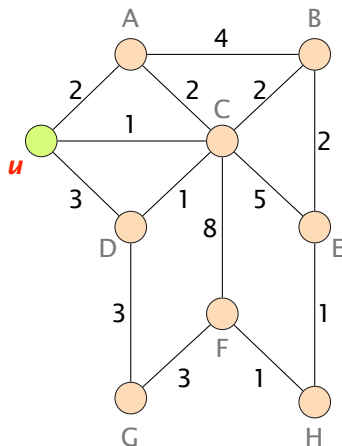
Prof. Laurent Vanbever

Solution: Exercise 2 – Routing Concepts

Routing Concepts

2.1 Dijkstra's Algorithm

The figure on the left shows a weighted graph representing a network topology with 9 nodes.



Weighted graph representing a network topology.

- a) Each of the links in the graph has an associated weight. Given that the graph represents a network, what could be the meaning of the link weights?

Solution: The "cost" of sending traffic via this link (in terms of money, delay, bandwidth, ...)

- b) Starting from node u , (i) manually compute Dijkstra's algorithm, and then (ii) list the obtained shortest-paths from u to each of the other nodes. For computing Dijkstra's algorithm, you can use the table below. The algorithm follows the one discussed in the lecture. If several nodes could next be added to node set S , select the node that comes first in the alphabet.

Solution: The shortest-paths between node u and all other nodes are listed in the following table:

Node	Path	$\Sigma(\text{weights})$
A	$u - A$	2
B	$u - C - B$	3
C	$u - C$	1
D	$u - C - D$	2
E	$u - C - B - E$	5
F	$u - C - B - E - H - F$	7
G	$u - C - D - G$	5
H	$u - C - B - E - H$	6

- c) Based on the shortest-paths from the previous task, derive the forwarding table of node u .

Solution: The following table illustrates the forwarding table of node u.

destination	next-hop
A	A
B	C
C	C
D	C
E	C
F	C
G	C
H	C

Solution:

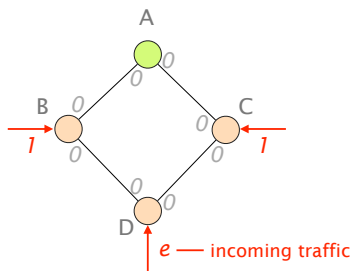
Iteration	Node Set S	D(.)								
		u	A	B	C	D	E	F	G	H
1	u	0	2	∞	1	3	∞	∞	∞	∞
2	u, C	0	2	3	1	2	6	9	∞	∞
3	u, C, A	0	2	3	1	2	6	9	∞	∞
4	u, C, A, D	0	2	3	1	2	6	9	5	∞
5	u, C, A, D, B	0	2	3	1	2	5	9	5	∞
6	u, C, A, D, B, E	0	2	3	1	2	5	9	5	6
7	u, C, A, D, B, E, G	0	2	3	1	2	5	8	5	6
8	u, C, A, D, B, E, G, H	0	2	3	1	2	5	7	5	6
9	u, C, A, D, B, E, G, H, F	0	2	3	1	2	5	7	5	6

Use this table for computing Dijkstra's algorithm in subtask b.

2.2 Changing Weights

So far, we have only seen cases in which the link weights in a network were static. However, the Internet itself is not static at all: traffic volumes change constantly; devices connect, move and disconnect. This begs the question: If the Internet is so dynamic, why should one not use dynamic weights instead?

Consider the figure on the left where B, C, D send traffic to the green destination A. (This is the only traffic in the network.) The red arrows show the incoming traffic and are labeled with its volume (1 or e). You can assume that $e \gg 1$. Unlike before, the weights on the links are bidirectional. Hence, the weight from A to B can be different to the weight from B to A.



Network topology with directional link weights.

In this network, the traffic is always forwarded along the shortest path according to the link weights. If two paths have the same cost, the path with the (alphabetically) lower next-hop is picked. Initially, e.g. when all the weights are 0, B has two paths available to reach the destination: one path via A and another one via D. According to the rule, B picks the path via A.

A specialty of this network is that the weights are dynamic and always represent the link load. Hence, if there is traffic of volume 1 being forwarded from A to B, the load of the link from A to B and therefore also the weight is 1.

In the following, we ask you to compute the forwarding state. As the link weights are dynamic, the forwarding state changes quite frequently. Therefore, you should approach the task step-by-step: at every step, consider the load on the link to be fixed to the one of the previous step. With this, compute the forwarding state, and afterwards update the load on every link.

Fill in the following table:

Solution:

	Link Load								Next Hop		
	A → B	A → C	B → A	B → D	C → A	C → D	D → B	D → C	B	C	D
0	0	0	0	0	0	0	0	0	A	A	B
1	0	0	1 + e	0	1	0	e	0	D	A	C
2	0	0	0	1	2 + e	0	0	1 + e	A	D	B
3	0	0	2 + e	0	0	1	1 + e	0	D	A	C
4	0	0	0	1	2 + e	0	0	1 + e	A	D	B
5	0	0	2 + e	0	0	1	1 + e	0	D	A	C
6	0	0	0	1	2 + e	0	0	1 + e	A	D	B
7	0	0	2 + e	0	0	1	1 + e	0	D	A	C

What is the problem with the dynamic weights?

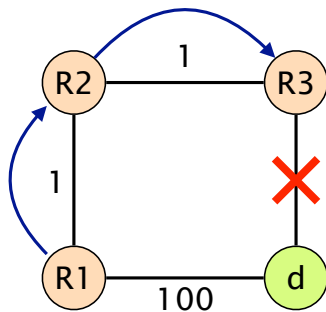
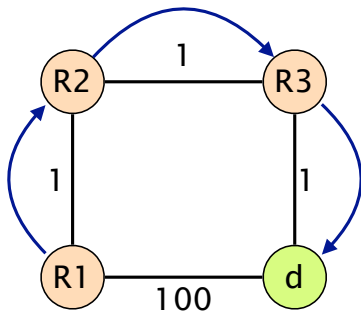
Solution: The dynamic weights lead to oscillations in the network. The forwarding state oscillates between two states: all left and then all right.

2.3 Dijkstra's Algorithm with Link Failure

The routers in the network on the top left use Dijkstra's Algorithm to find the shortest path towards destination d . You can assume that every router knows the entire network graph. In case of a failure, the routers directly affected by it inform the other routers by flooding this information in the network.

The blue arrows indicate how the routers forward the traffic: $R2$, for example, sends packets for destination d to router $R3$.

Now the link between router $R3$ and d fails (network at the bottom left) and $R3$ can no longer send packets towards destination d . As $R3$ is directly connected to the failed link, it detects the failure immediately. It starts to flood this information, such that all the routers can update their network view and recompute Dijkstra's algorithm.



Dijkstra's algorithm with a link failure

- a) What is the new shortest-path from $R3$ towards destination d ?

Solution: $R3, R2, R1, d$

- b) Assume now that the computation of the new shortest-path is *very* fast and finishes before $R3$ starts the flooding of the messages announcing the link failure. $R3$ sends a packet towards d using the new shortest-path. Will the packet reach its destination? Which path will it take?

Solution: No, $R2$ does not yet know about the link failure and did not update its shortest-path towards d . It will send the packet back towards $R3$. The packet is therefore stuck in a forwarding loop.

- c) Can you find a sequence of link failure messages and shortest-path computations such that the problem discovered in the previous task is observed between $R1$ and $R2$? The *only* link failure is still between router $R3$ and d .

Solution: $R2$ did receive a link failure message and updated its shortest-path. $R2$ then sends a packet towards d to the next-hop $R1$ before $R1$ can update its shortest-path.

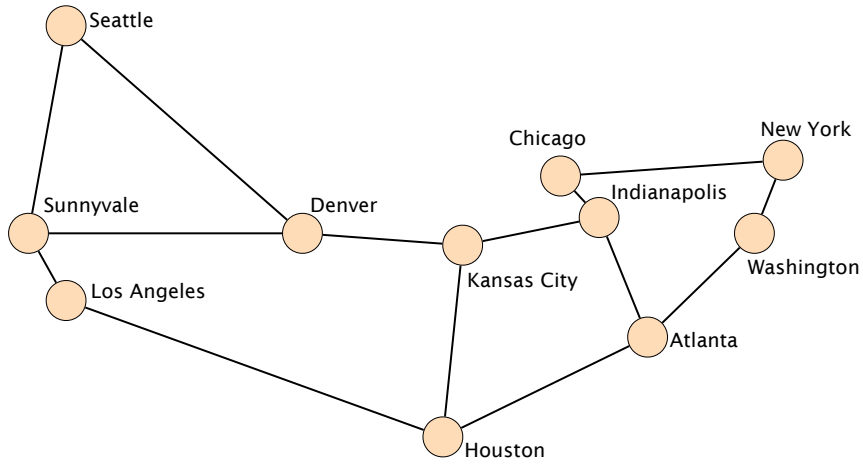
- d) As we have discovered, the order in which the link failure messages are processed and the new shortest-paths are computed is crucial for a correct forwarding behavior in the network. In which order should the router update their forwarding tables, such that the previously observed problems will not occur? Can you find a more general "rule" for a safe ordering of forwarding rule updates?

Solution: Good order: $R1, R2, R3$. To prevent forwarding loops, the routers should update their forwarding tables based on their distance to the destination. The router nearest to the destination should update its forwarding table first.

2.4 Link Weight Configuration

The Abilene network^a was a high-performance backbone network in the US. You are the network operator in charge and you have to configure the link weights in the network. Initially, all links have a weight of one and routers will always use the shortest-path available to reach a destination. For this task, assume that the weights have to be symmetric (i.e., for a given link, the weight is the same in both directions).

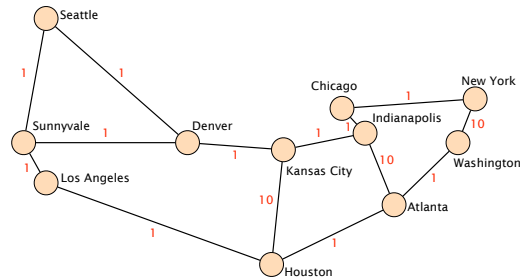
^ahttps://en.wikipedia.org/wiki/Abilene_Network



The Abilene network in the US.

- a) Is it possible to configure the link weights such that the packets sent by the router located in **Los Angeles** to the routers located in **New York** and to the ones in **Washington** take a different path? Note: the path from Los Angeles to New York and the one from Los Angeles to Washington *should not* have any link in common.

Solution:

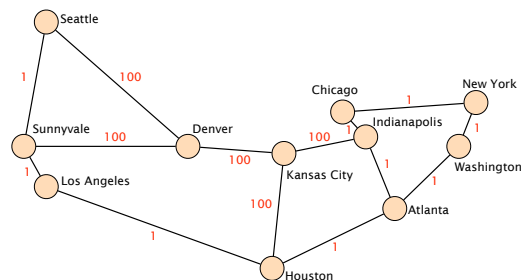


- b) Is it possible to configure the link weights such that the packets sent by the router located in **Los Angeles** to the router located in **New York** follow one path while the packets sent by the router located in **New York** to the router located in **Los Angeles** follow a *completely different* path?

Solution: Not possible in general.^a We consider only links which have the same weight in both directions. If the two routers would use different paths for the two traffic directions, the two paths would need different total weights. That implies that one path is shorter and one router is not using the shortest-path available. A contradiction to our initial assumption.

- c) Assume that the routers located in **Denver** and **Kansas City** need to exchange lots of data on the direct link. Can you configure the link weights such that the link between these two routers does not carry *any* packet sent by *any* other router in the network?

Solution:



^aIt would be possible in the following special case: there are at least two paths between New York and Los Angeles with the same minimum total weight and the two routers choose at random different minimum-total-weight paths for the two directions.

2.5 Source-and-Destination-Based Routing

As we have seen in the lecture, destination-based routing is the default in the Internet. Hence, based on the destination of a packet, the router decides where to forward an incoming packet next. However, it can also base its decision on other criteria, such as the source of a packet.

- a) Is it possible to design a routing scheme that does not rely on the destination of a packet and still produces a valid global forwarding state? Justify.

Solution: It is not possible. As we learned in the lecture, the global forwarding state is valid if it always delivers packets to the correct destination. However, when forwarding packets without looking at their destination, it is not possible to know where to send the packet next such that it reaches eventually the destination.

- b) Compare destination-routing that is solely based on the destination and source-and-destination routing that uses both the source and destination. What are the advantages and disadvantages of the two in terms of path diversity and the state required?

Solution: Destination-Routing: The path diversity is low as all the packets follow the same path at some point. The required state is also low as every device needs at most n entries if there are n different addresses in the Internet. Source-and-Destination-Routing: The path diversity is high as packets for one destination coming from one source could take a different path as the packets from all the other sources. The required state is also high as every device needs at most n^2 entries, one for each source-destination pair.