ETH zürich

Networked Systems
ETH Zürich — seit 2015

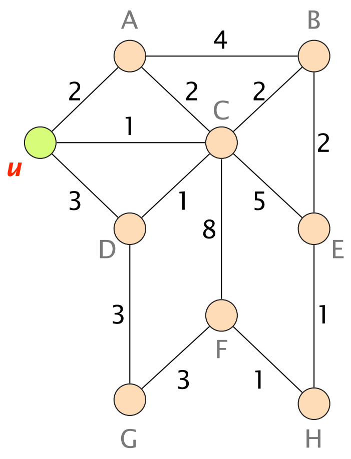
Communication Networks

Prof. Laurent Vanbever

Exercise 2 – Routing Concepts

Routing Concepts

2.1 Dijkstra's Algorithm



Weighted graph representing a network topology.

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

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?

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.

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

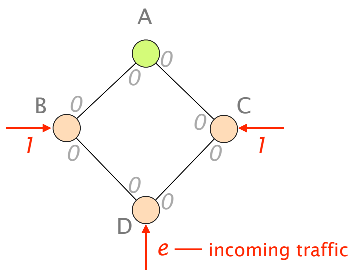
		$D(\,.\,)$								
Iteration	Node Set S	u	A	B	C	D	E	F	G	H
1	u	0	2	∞	1	3	∞	∞	∞	∞
2										
3										
4										
5										
6										
7										
8										
9										

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* >> 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:

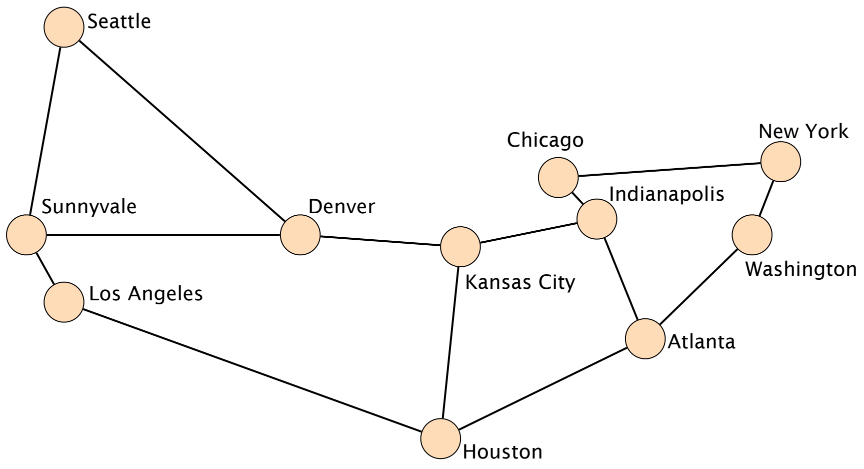
	Link Load								Next Hop		
	A → B	A → C	B → A	B → D	C → A	C → D	D → B	D → C	B	C	D
0	0	0	0	0	0	0	0	0	A	A	B
1	0	0	1 + e	0	1	0	e	0			
2											
3											
4											
5											
6											
7											

What is the problem with the dynamic weights?

2.3 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).

[a]https://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.

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?

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?

2.4 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.

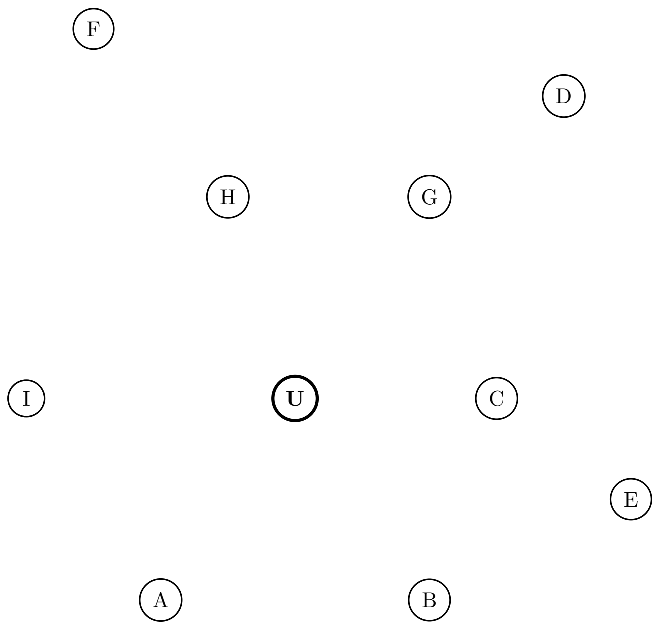
b) Compare destination-routing that is solely based on the destination and source-and-destination routing that uses both the source and destination. What are the advantages and disadvantages of the two in terms of path diversity and the state required?

2.5 Reverse Dijkstra (Exam Question 2020)

The network engineer at your company just retired and you have to take over. Unfortunately, it is unclear how the current network looks like. All you know is that it consists of 10 nodes (see below). In addition, you know that there is at most one link between two nodes and that each link has a non-negative weight. However, you neither know which links exist nor the weights configured on these links.

a) To figure out the links and the corresponding weights, you look at an output of Dijkstra's algorithm performed from node **U**. The table below shows the entire output of the algorithm. For each iteration, the table indicates the shortest path found so far towards each other node (starting from node **U**). The algorithm follows the one discussed in the lecture. If after one iteration there are multiple nodes with an equally-shortest path, the algorithm continues with the node which comes first in the alphabet.

Add all the links with their corresponding weight that you can identify based on the output from Dijkstra's algorithm.



A network consisting of 10 nodes with unknown links and link weights.

#	**U**	A	B	C	D	E	F	G	H	I
1	0	2	3	1	-	-	-	10	-	11
2	0	2	2	1	8	-	-	10	-	11
3	0	2	2	1	8	-	-	10	-	11
4	0	2	2	1	8	100	-	10	-	11
5	0	2	2	1	8	9	15	10	-	11
6	0	2	2	1	8	9	15	10	-	11
7	0	2	2	1	8	9	13	10	14	11
8	0	2	2	1	8	9	12	10	14	11
9	0	2	2	1	8	9	12	10	13	11
10	0	2	2	1	8	9	12	10	13	11

For each iteration (1 to 10) the table shows the shortest path found by Dijkstra's algorithm performed on node **U** towards all other nodes.

b) After analyzing the output from Dijkstra's algorithm, you are unsure if you really found all links in the network.

Could there be an additional link starting from node **U** which you could not identify based on the output from Dijkstra? If you think that is possible, give an example (link between node **U** and node ...) and indicate in which range the weight of this link could be. Otherwise, explain why this is not possible.

c) Could there be an additional link starting from node C which you could not identify based on the output from Dijkstra? If you think that is possible, give an example (link between node C and node ...) and indicate in which range the weight of this link could be. Otherwise, explain why this is not possible.