Shortest–path routing

Lars Staalhagen
Networks Technology and Service Platforms
DTU Fotonik Department of Photonics Engineering
larst@fotonik.dtu.dk

Version 2.0.1 — January 26, 2018

1 Introduction

The purpose of this short lecture note is to introduce the concept of shortest path routing, which is an essential part of modern communication networks. The lecture note is primarily intended to be used in course 34355 Routing in Data Networks, and is loosely based on [1] and [2].

2 Shortest path routing

The aim of shortest–path routing[1] is to find an optimal path in a network from a source node to a destination node. What constitutes an optimal path is decided by the network operator, i.e., it could be the path with the shortest end-to-end delay, the path with the highest capacity, or some other path.

The algorithms that are used to determine such an optimal path assume a graph–representation of the network, i.e., a graph, G is a tuple, $G = (V, E)$, where V is a set of the nodes in the graph and E is a set of the edges[2]. The nodes in the graph corresponds to routers in the real network, while the edges in the graph correspond to the communication links between routers. Since communication links can be either unidirectional (communication is only possible in one direction) or bi-directional (communication is possible in both directions), this must also be represented in the graph. Figure 1 shows a example of a graph with 5 nodes and 7 edges, where $V = \{A, B, C, D, E\}$ and $E = \{(A, B), (B, A), (B, D), (C, A), (C, D), (D, C), (D, E), (E, B)\}$.

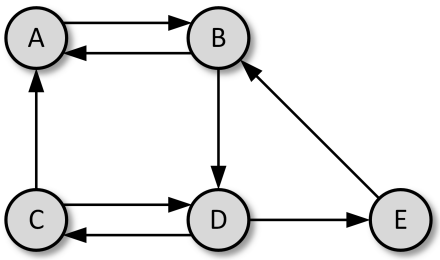

Figure 1: Sample graph.

In addition, a bi-directional link may be asymmetric in some sense, i.e., the communication in the two directions may have different characteristics, e.g., different delays. Therefore, the graph that is used to represent a network will be a *weighted directed* graph. Every edge is associated with a metric (a number), which represent some characteristic of the link in the real network. If a physical link is bi-directional, and has the same metric in both directions, it is common to draw the edge as "directionless", i.e., without an arrowhead, but it must be kept in mind, that this edge actually represents two directional edges. Part a of figure 2 illustrates a weighted directed graph, while part b shows how the two edges between nodes C and D may be drawn as one.

[1] In some textbooks, the term *least–cost* routing is used instead.
[2] Other names are *arcs* or *lines*

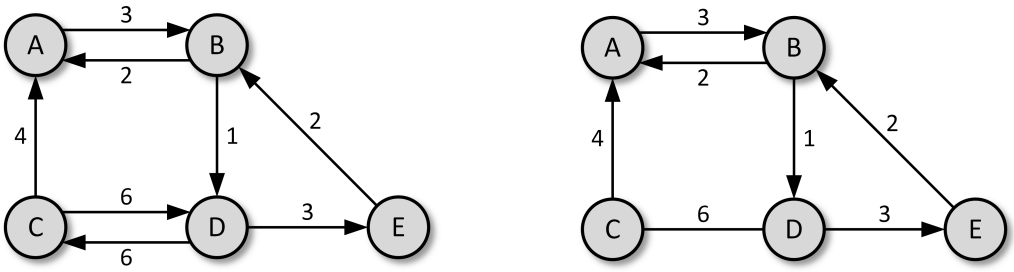

Figure 2: Collapsing edges with identical metrics into "direction less" edge (between nodes C and D).

The metrics that are associated with the edges in the graph can be thought of as the *length* of that edge. Therefore, a path from one node to another in the graph has a total length of the sum of the length of the edges that are part of the path. In this sense, the metrics are *additive*, that is, the total length of the path is the sum of the length of the different edges that are part of this path.

The purpose of shortest–path routing is then to find paths in the graph that minimizes the path–lengths. The simplest problem would be to find the shortest–path from a given source–node to a given destination–node, but the common algorithms today determine the shortest–paths from a given source node to *all* other nodes in the graph. A number of algorithms exist that can be used to determine the shortest paths, but in connection with communication networks, the most common are the *Bellman-Ford* algorithm and the *Dijkstra* algorithm. The former is used in a distributed version in connection with the RIP routing protocol, while the latter is used with the OSPF protocol.

As described earlier, the metrics associated with the edges in the graph correspond with some characteristic of the physical link in the network. Examples of these metrics include:

- Delay – the metric is (directly) proportional with the delay on the link, so that the shortest–path from the source to the destination is the path that minimizes the end–to–end delay. The link–delay may include any buffering delay at the sending node.
- Economical cost – the metric is (directly) proportional with the expenditure of transmitting information on the link. The shortest–path is then the path that minimizes the expenditure for the network operator.
- Error rate – the metric is a function of the link's error rate[3], so that a link with a high error–rate is depicted in the graph as an edge with a high cost. Finding the path in the graph that minimizes the sum of the edge metrics would then tend to favour links with lower error rates, since they would have a smaller metric.

In general, the link metrics may also be a function of (some of) the parameters of a link, i.e.

$$\text{Metric} = f(\text{delay}, \text{economical cost}, \ldots)$$

In mathematical terms, the link–metrics are here denoted by C_{ij} for the metric on the link *from* node i *to* node j. In general, $C_{ij} \neq C_{ji}$, i.e. for a bi-directional link, the metrics are generally different in the two directions, e.g., if the metrics are a function of the link delays. By definition, $C_{ii} = 0$, since (informally) the distance from node i to node i is of course 0. Furthermore, two different nodes, i and j, might not have a direct edge between them, so it is impossible to send a packet directly from node i to node j. This situation is represented in the metrics as $C_{ij} = \infty$.

The results of the shortest–path calculations are stored in the source–node's routing table. If the network uses *source–routing*, the entire paths are stored since they must be included in the packets. If the network uses *hop–by–hop* routing, the source node will only need to store the identity of the next–hop node for all destinations.

2.1 Bellman-Ford algorithm

The Bellman-Ford algorithm can be described as follows:

[3]Note that the metric and the link's error rate are not directly proportional in this case.

Definitions: s = Source node. D_j^h = The cost of shortest path from s to node j under the condition that the path contains at most h links.

Initialization: $\forall h : D_s^h = 0$ and $\forall j \neq s : D_j^0 = \infty$.

Iteration: $\forall j : D_j^{h+1} = \min_k \left(D_k^h + C_{kj} \right)$. Keep repeating the iteration until $\forall j : D_j^{h+1} = D_j^h$.

Figure 3: Bellman-Ford algorithm

Consider the graph in figure 4.

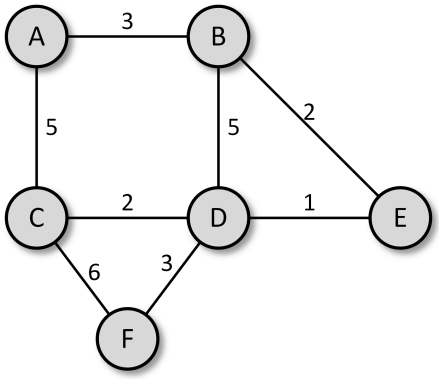

Figure 4: Example network.

If we want to calculate the shortest paths from node A to all other node in this network, we get the results, shown in table 1.

	Node B		Node C		Node D		Node E		Node F	
h	D_B^h	Path	D_C^h	Path	D_D^h	Path	D_E^h	Path	D_F^h	Path
0 (Init)	∞	—	∞	—	∞	—	∞	—	∞	—
1	3	A–B	5	A–C	∞	—	∞	—	∞	—
2	3	A–B	5	A–C	7	A–C–D	5	A–B–E	11	A–C–F
3	3	A–B	5	A–C	6	A–B–E–D	5	A–B–E	10	A–C–D–F
4	3	A–B	5	A–C	6	A–B–E–D	5	A–B–E	9	A–B–E–D–F
5	3	A–B	3	A–C	6	A–B–E–D	5	A–B–E	9	A–B–E–D–F

Table 1: Bellman-Ford calculation of shortest paths

Since $D_j^5 = D_j^4$ for all j, the algorithm has converged, and the contents of the routing table for node A will then be as shown in table 2.

At this point, the shortest–paths from the source node to all other nodes in the network have been found, but in general, we need the shortest–paths between any pair of nodes in the network. For this, it is necessary to execute the Bellman-Ford algorithm V times, using (in turn) every node in the network as the source node in the algorithm.

2.2 Distributed Bellman-Ford

The Bellman-Ford algorithm can also be implemented in a distributed version. To implement this, a node must from time to time transmit information from its entire routing table to all its neighbour[4]

[4]Two nodes are considered neighbours, if they can communicate directly, i.e., that there is a link between the two nodes

Destination	Path (source–routing)	Next–hop (hop–by–hop routing)
B	A–B	B
C	A–C	C
D	A–B–E–D	B
E	A–B–E	B
F	A–B–E–D–F	B

Table 2: Contents of routing table for node A.

nodes. It can be shown that as long as all nodes from time to time transmit this information, the optimal shortest–paths will be determined eventually.

The idea behind the distributed Bellman-Ford is that, whenever a node receives information from a neighbour node, it will determine whether the information will lead to a better (smaller cost) path than previously known. As an example, consider figure 5: Suppose that node A receives the following information from node B: "I can reach node Z with a cost of α" and from C: "I can reach node Z with a cost of β". Node A would then determine, which of these possible ways of reaching node Z would be better by determining the minimum of $C_{AB}+\alpha$ and $C_{AC}+\beta$. If $C_{AB}+\alpha < C_{AC}+\beta$ then the (currently) best path from A to Z is via node B; otherwise it is via node C.

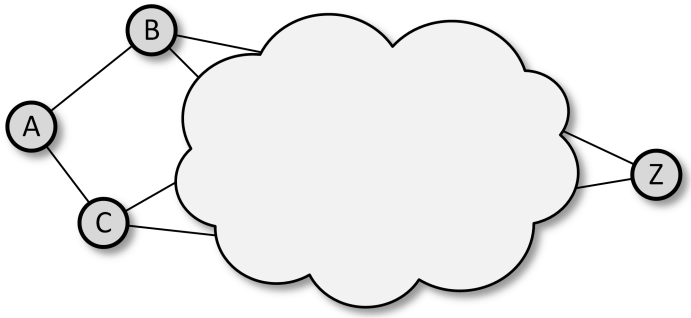

Figure 5: Example network for the distributed Bellman-Ford.

This description of the distributed version of the Bellman-Ford algorithm is very simplified. The RIP routing protocol uses the distributed Bellman-Ford algorithm, but with a number of additional facilities.

2.3 Dijktra's algorithm

Dijkstra's algorithm can be described as shown in figure 6.

Definitions: s = source node. D_j = (estimate) of the cost of the shortest path from s to (destination) node j. M = set of nodes.

Step 1: Initialize the set M with just the source node, $M = \{s\}$.

Step 2: Find the node with the smallest D_j that is not yet included in the set M and add this node to M: $D_k = \min_{j \notin M} D_j$. $M \leftarrow M \cup \{k\}$.

Step 3: For all nodes not (yet) included in M, check if a path through the just–added node (node k) would be better than the current path: $D_j = \min(D_j, D_k + C_{kj})$.

Step 4: If $V \neq M$ continue from step 2; otherwise the algorithm has terminated.

Figure 6: Dijkstra's algorithm

If we use Dijkstra's algorithm on the network in figure 4, we get table 3. The routing table of node A will of course be the same as in the Bellman-Ford example.

	Node B		Node C		Node D		Node E		Node F	
M	D_B	Path	D_C	Path	D_D	Path	D_E	Path	D_F	Path
$\{A\}$	3	A–B	5	A–C	∞	—	∞	—	∞	—
$\{A, B\}$	3	A–B	5	A–C	8	A–B–D	5	A–B–E	∞	—
$\{A, B, C\}$	3	A–B	5	A–C	7	A–C–D	5	A–B–E	11	A–C–F
$\{A, B, C, E\}$	3	A–B	5	A–C	6	A–B–E–D	5	A–B–E	11	A–C–F
$\{A, B, C, D, E\}$	3	A–B	5	A–C	6	A–B–E–D	5	A–B–E	9	A–B–E–D–F
$\{A, B, C, D, E, F\}$	3	A–B	5	A–C	6	A–B–E–D	5	A–B–E	9	A–B–E–D–F

Table 3: Dijkstra calculation of shortest paths

3 Problems

Problem 1: Suppose that all link metrics in a graph has the same value. What characterizes the paths found by a shortest–path algorithm?

Problem 2: Consider a situation where packets transmitted on a link will either arrive without errors at the other end–point of the link, or be lost. The probability of a packet loss on the link between nodes i and j is denoted P_{ij}. Determine the function for translating a link's packet loss rate into a metric, so that the shortest paths are the paths that maximizes the probability that the packets arrive at the destination, i.e., determine $f()$ in the relation: $C_{ij} = f(P_{ij})$.

Problem 3: Consider the graph in figure 7. Determine the shortest–paths from node A to all other nodes using the Bellman-Ford algorithm. Repeat the problem with Dijkstra's algorithm.

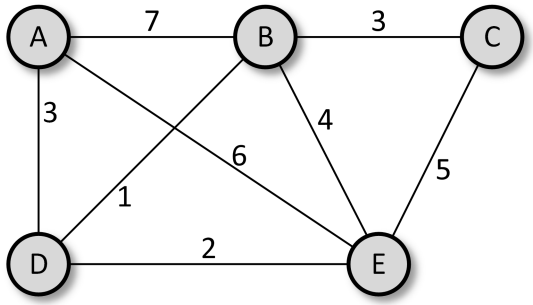

Figure 7: Graph for problems 3

Problem 4: Explain how the Bellman-Ford algorithm can be modified to find the shortest–paths to a destination node from all other nodes in the graph.

References

[1] Bertsekas, D. and Gallager, R.: "*Data Networks*", 2nd edition, 1992

[2] Cormen, T. H., Leiserson, C. E. and Rivest, R. L.: "*Introduction to Algorithms*", 1990