3F4: Data Transmission

Handout 13: Dijkstra's Routing Algorithm

Ioannis Kontoyiannis

Signal Processing and Communications Lab
Department of Engineering
i.kontoyiannis@eng.cam.ac.uk

Lent Term 2019

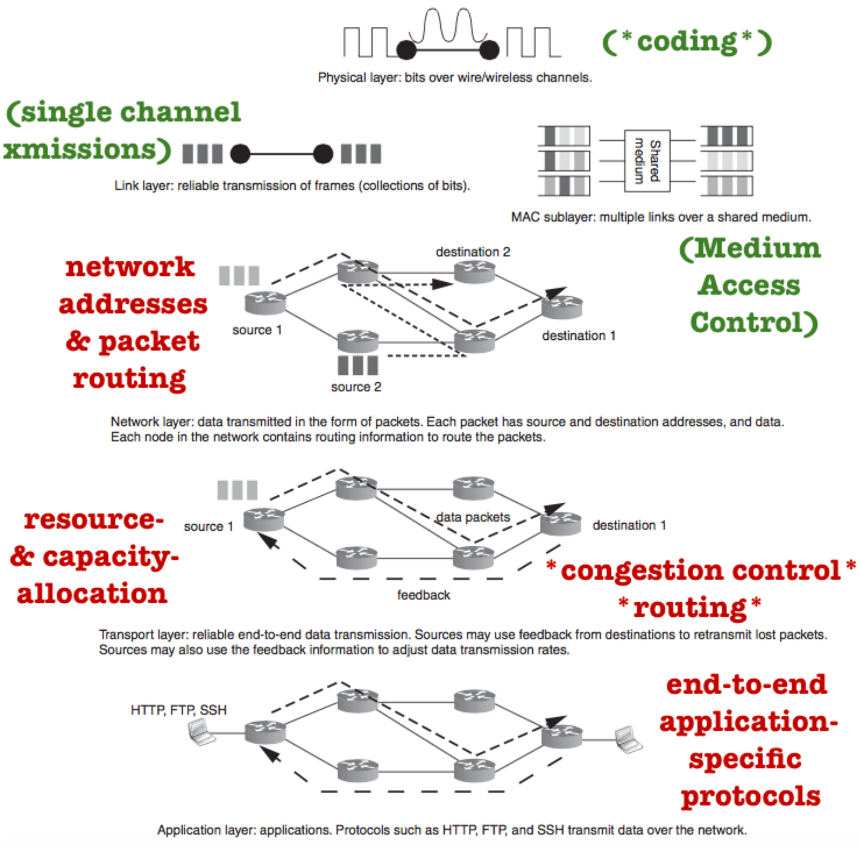


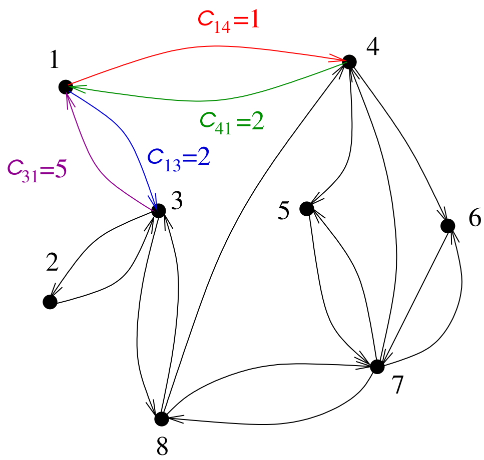
Figure 1.1 Schematic of the layered architecture of a communication network.

Image from: Srikant and Ying "Communication networks: an optimization, control, and stochastic networks perspective." Cambridge University Press, 2013

The network routing problem

- Network = directed graph $\mathcal{G} = (\mathcal{N}, \mathcal{L})$
- $\mathcal{N}$ = set of nodes
- $\mathcal{L}$ = set of directed links (i, j) for $i, j, \in \mathcal{N}$
- Cost $c_{ij} > 0$ associated with each link $(i, j) \in \mathcal{L}$

- Goal. For a given source node u and each destination node i find the route $(u, j_1, \ldots, j_m, i)$ with minimum total cost $c_{uj_1} + c_{j_1 j_2} + \cdots + c_{j_{m-1} j_m} + c_{j_m i}$
- Assume. The network topology and all link costs c_{ij} are known to each node



Dijkstra's algorithm: Initialization

Iterative algorithm that determines the paths $u \to i$ for all $i \in \mathcal{N}$, with minimum total cost ω^*_{ui}

- Fix **source node u**
- Let $\mathcal{K} \subset \mathcal{N}$ be the nodes i for which minimum-cost path $u \to i$ is already known
- Initially set $\mathcal{K} = \{u\}$
- For each node i let ω_{ui} = min-cost $u \to i$ at current iteration
- Initially set: $\omega_{ui} = c_{ui}$ if $(u, i) \in \mathcal{L}$
  $\omega_{ui} = +\infty$ if $(u, i) \notin \mathcal{L}$
- For each node i let p_{ui} = previous hop $\to i$ in current iteration min-cost path
- Initially set: $p_{ui} = u$ if $(u, i) \in \mathcal{L}$
  $p_{ui} = -1$ (unknown) if $(u, i) \notin \mathcal{L}$

Dijkstra's algorithm: Iterative step

Recall: Nodes in $\mathcal{K}$ are 'done'
Next:

- Examine the nodes *not* in $\mathcal{K}$
- Find a node $i^* \notin \mathcal{K}$ achieving the minimum current cost

$$i^* = \arg\min_{i \notin \mathcal{K}} \omega_{ui}$$

1. Add i^* to $\mathcal{K}$: Let $\mathcal{K} \mapsto \mathcal{K} \cup \{i^*\}$
2. Update ω_{ui} and p_{ui} for all other $i \notin \mathcal{K}$
   For each $i \notin \mathcal{K}$:
   - 2a. If the current path $u \to i$ is better than $u \to i^* \to i$
     i.e., if $\omega_{ui} \leq \omega_{ui^*} + c_{i^*i}$, do nothing
   - 2b. Otherwise, set $\omega_{ui} = \omega_{ui^*} + c_{i^*i}$ and $p_{ui} = i^*$

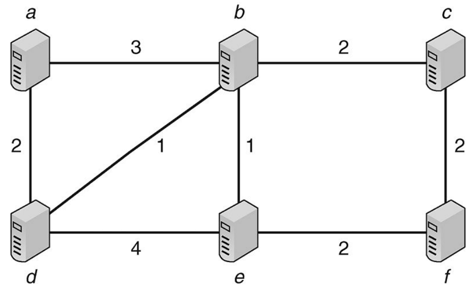
- If $\mathcal{K} = \mathcal{N}$, we are done. If not, repeat

**WHY ~~THE~~ ON EARTH
DOES THIS WORK??!**

We will prove it does

But first, an example

Example of Dijkstra's algorithm

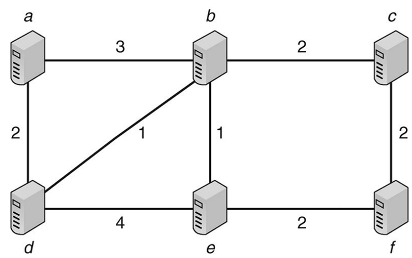


Iteration	$\mathcal{K}$	(ω_{ab}, p_{ab})	(ω_{ac}, p_{ac})	(ω_{ad}, p_{ad})	(ω_{ae}, p_{ae})	(ω_{af}, p_{af})
Initialize	$\{a\}$	$(3, a)$	$(\infty, -1)$	$(2, a)$	$(\infty, -1)$	$(\infty, -1)$
1	$\{a, d\}$	$(3, a)$	$(\infty, -1)$	$(2, a)$	$(6, d)$	$(\infty, -1)$
2	$\{a, d, b\}$	$(3, a)$	$(5, b)$		$(4, b)$	$(\infty, -1)$
3	$\{a, d, b, e\}$		$(5, b)$		$(4, b)$	$(6, e)$
4	$\{a, d, b, e, c\}$		$(5, b)$			$(6, e)$
5	$\{a, d, b, e, c, f\}$					$(6, e)$

$\rightsquigarrow$ Table contains complete information about the min-cost path from a to any other node!

Example of Dijkstra's algorithm

$(\omega^*_{ab}, p^*_{ab})$	$(\omega^*_{ac}, p^*_{ac})$	$(\omega^*_{ad}, p^*_{ad})$	$(\omega^*_{ae}, p^*_{ae})$	$(\omega^*_{af}, p^*_{af})$
$(3, a)$	$(5, b)$	$(2, a)$	$(4, b)$	$(6, e)$



E.g. Min-cost path $a \to f$?

The cost of the best path is $\omega^*_{af} = 6$

To find the actual path read the table backwards:
The last node on the path is f
the previous node is e, the one before that is b
and the one before is a

$\Rightarrow$ Best path: $a \to b \to e \to f$

Total cost: $c_{ab} + c_{be} + c_{ef} = 3 + 1 + 2 = 6 = \omega^*_{af}$

Properties of Dijkstra's algorithm

Suppose $\mathcal{G}$ contains $N = |\mathcal{N}|$ nodes
First, some obvious properties

- Length. The algorithm always terminates after $N - 1$ iterations
- Progress. In each iteration $1 \leq k \leq N - 1$ we have $|\mathcal{K}| = k + 1$
- Complexity. The worst-case running time of the algorithm is $O(N^2)$
[But there are much more accurate bounds for graphs with some structure]

Correctness of Dijkstra's algorithm

Theorem (correctness)

For each $i \in \mathcal{K}$ we have $\omega_{ui} = \omega^*_{ui}$

Proof. By induction. Assume it holds after the $(k-1)$th iteration
Consider the nodes that have the smallest true minimum cost among those not in $\mathcal{K}^{(k-1)}$:

$$\mathcal{S} = \{j : j \notin \mathcal{K}^{(k-1)} \text{ and } \omega^*_{uj} = min_{\ell \notin \mathcal{K}^{(k-1)}} \omega^*_{u\ell}\}$$

It suffices to show that in the kth iteration the algorithm:
(a) selects a node $j \in \mathcal{S}$ to add to $\mathcal{K}^{(k-1)}$ and (b) $\omega^{(k-1)}_{uj} = \omega^*_{uj}$

For any node $j \in \mathcal{S}$, the previous hop p^*_{uj} to j on a min-cost path must be in $\mathcal{K}^{(k-1)}$: O/w, the true minimum cost to node p^*_{uj} would be smaller than that to j, contradicting the fact that $j \in \mathcal{S}$

Therefore, for any node $j \in \mathcal{S}$:

$$\omega^*_{uj} \leq \omega^{(k-1)}_{uj} \leq \min_{i \in \mathcal{K}^{(k-1)}} (\omega^*_{ui} + c_{ij}) \leq \omega^*_{up^*_{uj}} + c_{p^*_{uj}j} = \omega^*_{uj}$$

The second inequality holds due to the update rule 2. and the property that $\omega_{ui} = \omega^*_{ui}$ when i is added to $\mathcal{K}^{(k-1)}$

Correctness of Dijkstra's algorithm

Theorem (correctness)

For each $i \in \mathcal{K}$ we have $\omega_{ui} = \omega^*_{ui}$

Proof continued. So for any $j \in \mathcal{S}$ we have $\omega^{(k-1)}_{uj} = \omega^*_{uj} \Rightarrow (b)$

Finally, for any node $\ell \notin \mathcal{K}^{(k-1)}$ but also $\ell \notin \mathcal{S}$:

$$\omega^{(k-1)}_{uj} = \omega^*_{uj} < \omega^*_{u\ell} \leq \omega^{(k-1)}_{u\ell}$$

so ℓ will *not* be added to $\mathcal{K}$ in the kth iteration $\Rightarrow (a)$ $\square$

Ordering properties of Dijkstra's algorithm

Observe that, since at the kth iteration we select and add to $\mathcal{K}$ the node with the best minimal cost path from u we have in fact also proved:

Corollary (ordering)

1. For any pair $i \in \mathcal{K}$ and $j \notin \mathcal{K}$ we have $\omega^*_{ui} \leq \omega^*_{uj}$
2. If in each iteration k we add node n_k to $\mathcal{K}$ with $n_0 = u$ the source node, then
$$\omega^*_{un_0} \leq \omega^*_{un_1} \leq \omega^*_{un_2} \leq \cdots \leq \omega^*_{un_{N-1}}$$

Summary: Dijkstra's algorithm

- Finds min-cost paths for a given node
  to all other nodes in the network
- Min-cost paths are found iteratively
  in order of increasing cost
- Iterative algorithm with a dynamic programming flavour
- A 'link-state routing' algorithm: The complete network
  topology and all costs must be known by everyone
- Necessary assumption: All costs $c_{ij} > 0$
- Running time complexity is obviously $O(|\mathcal{N}|^2)$
  More careful analysis shows it is in fact $O(|\mathcal{N}| \log |\mathcal{N}| + |\mathcal{L}|)$
- Next time we will see the Bellman-Ford algorithm:
  A dynamic programming method for finding min-cost paths
  A 'distance-vector routing' algorithm:
  Only requires local information