3F4: Data Transmission

Handout 14: The Bellman-Ford Routing Algorithm

Ioannis Kontoyiannis

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

Lent Term 2019

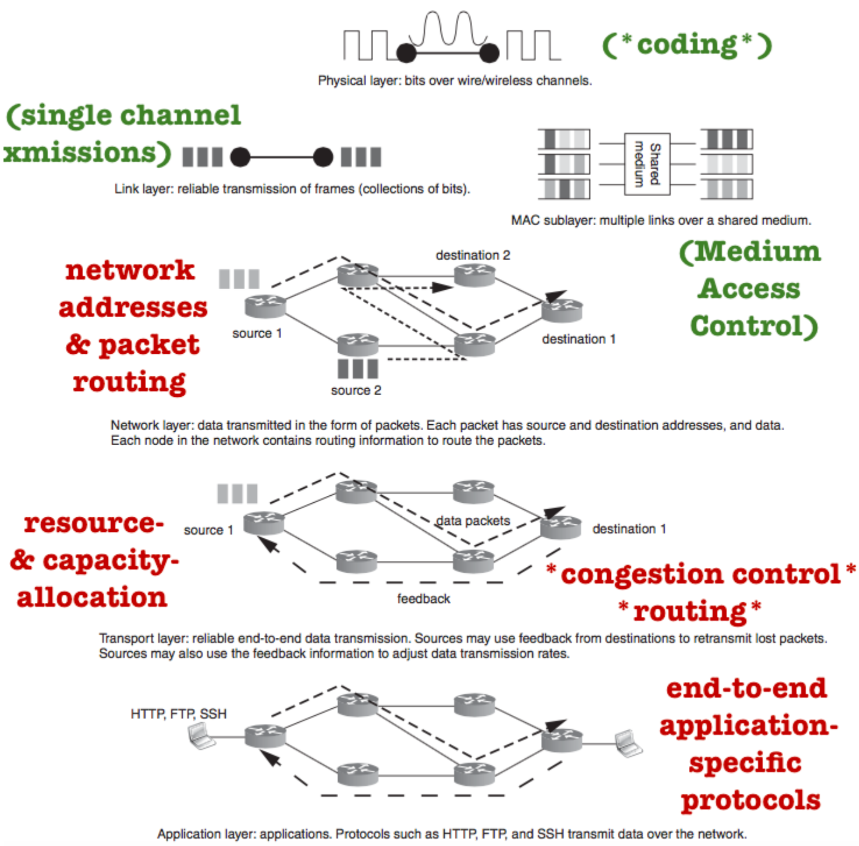


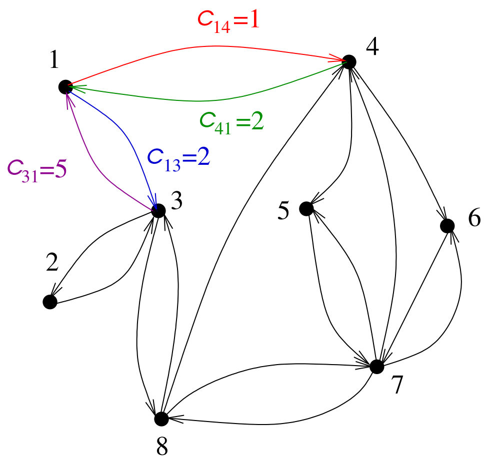
Figure 1.1 Schematic of the layered architecture of a communication network.

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 **every** source node u and **every** destination node $i \neq u$ 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 the link costs c_{ij} are **not** all known to each node



Bellman-Ford algorithm: Preliminaries

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

Each node $u \in \mathcal{N}$

- knows the cost c_{ui} for all its neighbouring nodes i
  i.e., all $i \in \mathcal{N}$ such that $(u, i) \in \mathcal{L}$
- can communicate and exchange information
  with all its neighbours
- maintains a distance vector $(\boldsymbol{\omega_{ui}}, \boldsymbol{n_{ui}})$
  for each $i \in \mathcal{N}$, where:
  - $\boldsymbol{\omega_{ui}}$ $=$ min-cost among paths $u \to i$
    at current iteration
  - $\boldsymbol{n_{ui}}$ $=$ next hop $\to i$
    in current iteration min-cost path

Bellman-Ford algorithm: Initialization

Initialization

Select a node $u \in \mathcal{N}$ and for each $i \neq u$ set:

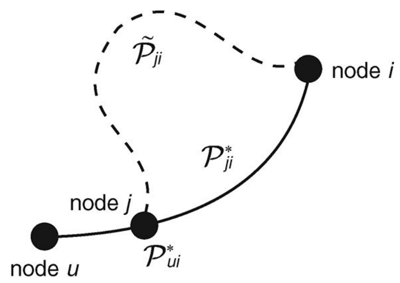
- $\omega_{ui} = c_{ui}$ if $(u, i) \in \mathcal{L}$
  $\omega_{ui} = +\infty$ if $(u, i) \notin \mathcal{L}$
- $n_{ui} = i$ if $(u, i) \in \mathcal{L}$
  $n_{ui} = -1$ (unknown) if $(u, i) \notin \mathcal{L}$

Repeat for all other nodes $u' \in \mathcal{N}$

Main idea for updates

The Bellman-Ford equation

$$\omega^*_{ui} = \min_{j:(u,j)\in\mathcal{L}} \left[c_{uj} + \omega^*_{ji}\right]$$



Bellman-Ford algorithm: Iterative updates

Iteration $t = 1, 2, \ldots$

Select a node $u \in \mathcal{N}$ and for each $i \neq u$:

$t.c.$ Send all the current values (ω_{ui}) to all the neighbours of u

Repeat for all other nodes $u' \in \mathcal{N}$

Select a node $u \in \mathcal{N}$ and for each $i \neq u$:

$t.u.$ Update ω_{ui} and n_{ui}

$t.u.1.$ If adding a first hop $u \to j$ to the current path u-to-i does not help, i.e., if $\omega_{ui} \leq \min_{j:(u,j)\in\mathcal{L}} \{c_{uj} + \omega_{ji}\}$ then do nothing

$t.u.2.$ Otherwise, set $\omega_{ui} = \min_{j:(u,j)\in\mathcal{L}} \{c_{uj} + \omega_{ji}\}$ and $n_{ui} = \arg\min_{j:(u,j)\in\mathcal{L}} \{c_{uj} + \omega_{ji}\}$

Repeat for all other nodes $u' \in \mathcal{N}$

Repeat with $t \mapsto t + 1$

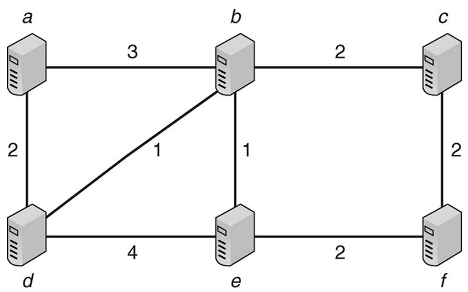
Remarks

- In practice, each iteration consists of two phases
  - all nodes send their current costs (ω_{ui}) to all their neighbours
  - all nodes update their own distance vectors (ω_{ui}, n_{ui})
- These communications and updates can be done in a synchronous or asynchronous fashion
- For the updates, each node u needs the values (ω_{ui}) and (ω_{ji}) for all $i \in \mathcal{N}$ but only for its neighbours j
- Why does this work?
  [We will 'prove' it does]
- How many iterations does the algorithm need?
  [We will see that it terminates after $|\mathcal{N}| - 1$ iterations]
- First, an example

Example of Bellman-Ford algorithm

Compute the min-cost paths from all nodes in the network *to node f*

Each node u only maintains the values of (ω_{uf}, n_{uf})

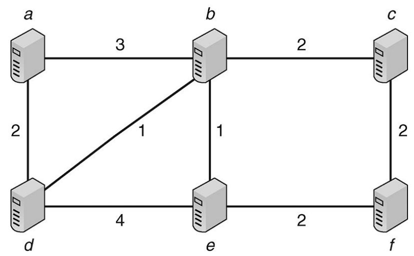


Iteration	(ω_{af}, n_{af})	(ω_{bf}, n_{bf})	(ω_{cf}, n_{cf})	(ω_{df}, n_{df})	(ω_{ef}, n_{ef})
Initialize	$(\infty, -1)$	$(\infty, -1)$	$(2, f)$	$(\infty, -1)$	$(2, f)$
1	$(\infty, -1)$	$(3, e)$	$(2, f)$	$(6, e)$	$(2, f)$
2	$(6, b)$	$(3, e)$	$(2, f)$	$(4, b)$	$(2, f)$
3	$(6, b)$	$(3, e)$	$(2, f)$	$(4, b)$	$(2, f)$

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

Example of Bellman-Ford algorithm

$(\omega^*_{af}, n^*_{af})$	$(\omega^*_{bf}, n^*_{bf})$	$(\omega^*_{cf}, n^*_{cf})$	$(\omega^*_{df}, n^*_{df})$	$(\omega^*_{ef}, n^*_{ef})$
$(6, b)$	$(3, e)$	$(2, f)$	$(4, b)$	$(2, f)$



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 forwards:

$\Rightarrow$ 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 Bellman-Ford algorithm

Consider a network with $N = |\mathcal{N}|$ nodes and $L = |\mathcal{L}|$ links

Theorem

All optimal paths of **length *k*** will be found by the kth iteration

Proof. Simple induction using the Bellman-Ford equation

$$\omega^*_{ui} = \min_{j:(u,j)\in\mathcal{L}} \left[c_{uj} + \omega^*_{ji}\right] \qquad \square$$

Corollary

All optimal paths will be found by the $(N-1)$th iteration

Corollary

The (time) complexity of the algorithm is $O(N \times L)$

Remarks and summary

- **Bellman-Ford** is a true dynamic programming algorithm
- All min-cost paths (from every node to every other node) are found in N iterations taking $O(N \times L)$ time
- Somewhat slower than **Dijkstra's algorithm** which we saw had complexity $O(N \log N + L)$
- Bellman-Ford can be implemented in an asynchronous fashion:
  Every nodes does their update in their own time
  and then sends the updated values to their neighbours
- A distributed, 'distance-vector routing' algorithm:
  Only local information used by each node
- Bellman-Ford more versatile than Dijkstra's algorithm
  e.g., can also be applied when some costs $c_{ij} < 0$!
- Main drawback: Quite sensitive to errors, link failures,
  changes in the network topology, etc.
- Popular in current practice, used by many ISPs

End of 3F4!

- Thanks for listening
- Have a good break!
- Please complete the course survey: