Network & Operation Systems Essentials

Dr Nikos Ntarmos

Joao Almeida-Domingues[*]

University of Glasgow

September 24th, 2018 – December 4th, 2019

Contents

These lecture notes were collated by me from a mixture of sources , the two main sources being the lecture notes provided by the lecturer and the content presented in-lecture. All other referenced material (if used) can be found in the *Bibliography* and *References* sections.

The primary goal of these notes is to function as a succinct but comprehensive revision aid, hence if you came by them via a search engine , please note that they're not intended to be a reflection of the quality of the materials referenced or the content lectured.

Lastly, with regards to formatting, the pdf doc was typeset in LaTeX, using a modified version of Stefano Maggiolo's class

[*]2334590D@student.gla.ac.uk

1 Routing

1.1 *Distance Vector Protocols*

Motivation
Given that the internet is by definition a network of networks , which though connected does not have a central management system which can keep track of all nodes, how are routing paths known? What's the optimal path between any two nodes?

DVPs tackle this problem by assigning a vector to each node containing the distance to every other node in the network. The main idea here is that initially each node will have access to the *cost* of sending a packet to its immediate neighbours. Then, once routing tables start being built, they are also shared amongst the nodes. Hence, farther away nodes costs are now known also via intermediary nodes. i.e if $A \to B$ and $B \to C$ then, once B shares its routing table with A, C it can now be known that $A \to B \to C$. Eventually all nodes will converge towards a *steady state*, where all tables have been computed

1.1 remark. The algorithm aims to find the shortest-cost path. Hence, routing tables can be updated at each *"iteration"*, if a shortest path is found between any two nodes.

1.2 remark. Though it makes it easier to think of the process as occurring in rounds, where each round corresponds to 1-*hop* in the network with all tables synced, this is not necessarily true

1.3 definition. Steady State when all nodes have computing their routing tables

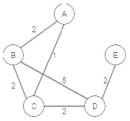
Algorithm

1.4 remark. During the process *forwarding tables* are used as an intermediate step, where triplets of the type `[Destination, Cost, NextHop]` are shared so that each node can track who sent them the info.

1. Populate each node with the vector `[Cost , NextHop]`
2. Share routing table
3. If routing tables received contain unknown nodes and/or cost to unknown is lower
   (a) Update vector cost `[DestCost + UnkCost , NextHop]`
4. Repeat from 2 until all nodes are known and periodically (topology check)

1.5 remark. Note that if any link breaks, then the process need not be restarted in full. The linked nodes simply change their vector appropriately. The affected nodes then attempt to build alternative paths by the same process

This process is usually illustrated by a 2D table



Routing table at node	Cost / Next hop to node				
	A	B	C	D	E
A	0/•	2/B	1/C	∞/-	∞/-
B	2/A	0/•	2/C	5/D	∞/-
C	1/A	2/B	0/•	2/D	∞/-
D	∞/-	5/B	2/C	0/•	2/E
E	∞/-	∞/-	∞/-	2/D	0/•

1.6 example. Take A for example. We now see that it has access to B, C tables. B, C are both connected, but their cost does not improve upon A costs to each. So A_B, A_C remain unchanged. However, A_D is currently unknown , but can be reached by both. Comparing their costs, we see that $A \to B \to D = 2 + 5$ and $A \to C \to D = 1 + 2 = 3$. Hence we update $A_D = [3, C]$, which reads essentially as *"A to D via C has a total cost of* 3*"*

To do (1)

Limitations

A major problem with DVP is *count-to-infinity* . Due to asynchronicity, in particular in larger networks, a broken link can fail to be communicated to farther away nodes in time. This means that one of the nodes involved in the broken link A, B, say B might receive an outdated table from another node C, before C is updated by B of the broken link . Hence, C will update B_A with the old path. At this point, B will interpret this a new existing path to A via C, adding the cost BCA. In the next round C will see that B has changed again, and will updated its cost appropriately, creating a feedback loop

Possible solutions:

1. Define ∞ , bounding the #hops
2. Split Horizon : prevents feedback loop by not advertising route back to the node which sent the update

Due to this, these protocols always suffer from slow convergence, since it tries to minimise state at nodes

Advantages

1. Simple to implement
2. Info at each node is minimised $\implies$ faster computation times $O(n)$
3. Good for small networks

1.2 *Link State*

It's composed of 2 main stages : *Flooding* and *Shortest Path Calculation*. The main difference here being that the shortest path computation is not done locally once and then its result is transmitted to the neighbours, but instead

nodes advertise information about all its neighbours and the link costs to them in the advertisement step. Those very same packets are then re-broadcasted to each neighbour, until all nodes have received the entire map of the network. Packed with the full map, each node is then free to compute a centralised computation* of the shortest path from itself to any other node.

**Usually by application of Djisktra's algorithm*

1.7 definition. Flooding every incoming packet is rebroadcasted through each outgoing link

1.8 definition. Link State Information links to neighbours, and the cost of using those links

Limitations

1. Complex implementation
2. Each node has to have a complete map of the network $\implies$ slower computation times $O(n^2)$

Advantages

1. Fast convergence $\implies$ good for large networks

as as
sa

To do...

- [ ] 1 (p. 3): typeset tables & adjust layout