CS 4457: Network Layer: Part 4

Brandon Chan Mesgana Dinare

February 11, 2020

1 Network Layer

1.1 Banyan Switch

The banyan switch, named after the roots of the banyan tree due to its similar appearance, is a crossover switch which connect multiple inputs to many outputs to create various paths. However, not all ports are mappable as shown in Figure 1.

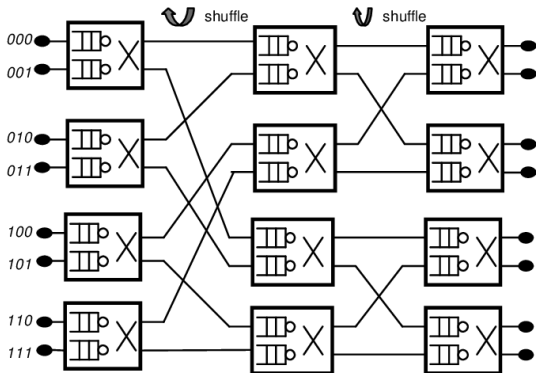


Figure 1: Banyan Switch

1.2 Bellman-Ford Algorithm

The Bellman-Ford algorithm is a single-source shortest path algorithm meaning it determines the shortest distance paths from one node to every other node in a graph. Not only can this algorithm handle negative-weight edges, but it can also detect negative-weight cycles. The run time of this algorithm is $O(|E||V|)$ where E is the number of edges and V is the number of vertices/nodes.

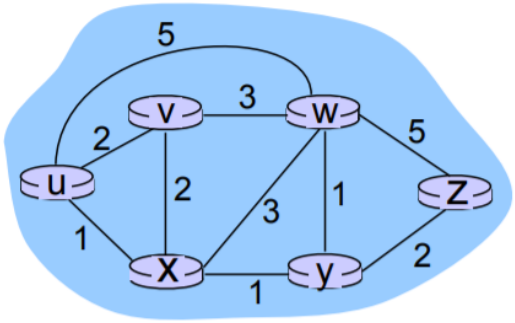


Figure 2: Bellman-Ford Example

For a simple example of how the algorithm works, we will denote the distance from one node to another node as $d_a(b)$ where a is the source and b is the destination. Additionally, $c(a, b)$ will be the cost from node a to node b (the weight of the edge between the two nodes). In Figure 2, it is observable that $d_v(z) = 5$, $d_x(z) = 3$, and $d_w(z) = 3$. If u was the source node and we wanted to find $d_u(z)$, the algorithm calculate the distance from u to z as such:

$$d_u(z) = \min[c(u, v) + d_v(z), c(u, x) + d_x(z), c(u, w) + d_w(z)]$$
$$d_u(z) = \min[2 + 5, 1 + 3, 5 + 3] = 4$$

1.3 Distance Vector Algorithm

The Distance Vector algorithm is used to have routers send packets to one another through the least costing path for quick transmission. The algorithm has the nodes (which are the routers) send estimated distance vectors containing the cost of the edges to their neighbors and the neighbors will update their own distance vectors utilizing the Bellman-Ford algorithm. If any updates are made, the process repeats to update the neighbors about the new distance vectors. In Figure 3, the adjacency matrices are shown to be local to each node and the distance vector sent is a row from the adjacency matrix. Initially, the matrices have the cost from the source node to its neighbors, a diagonal of zeros as the distance from one node to itself is 0, and the rest as infinity for unknown.

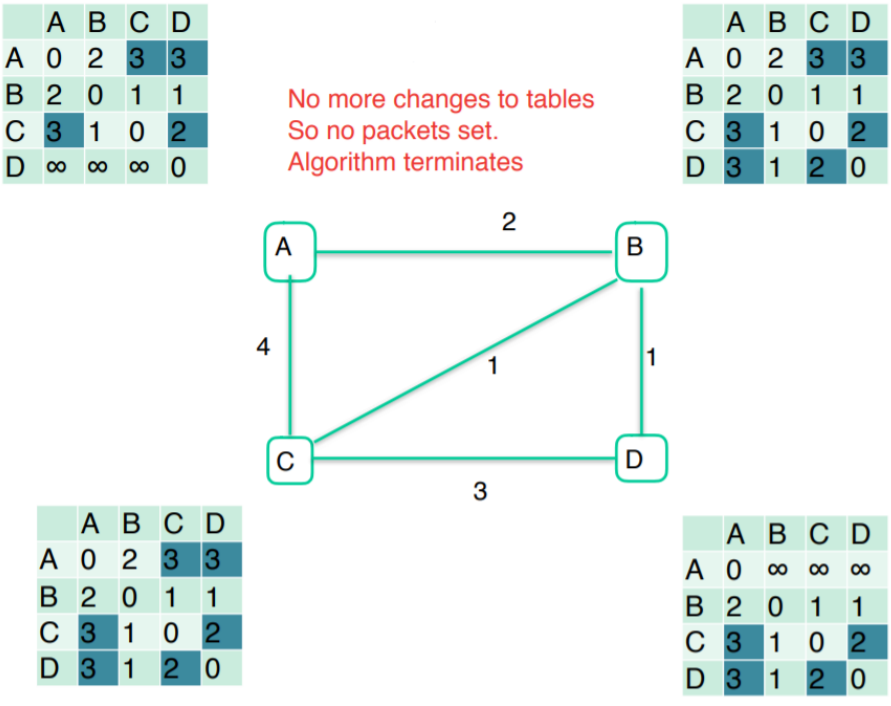


Figure 3: Distance Vector Example

The Distance Vector algorithm is iterative, asynchronous, and distributed. It is iterative and

asynchronous since each local iteration is caused by either a local edge cost change or a neighbor sent a distance vector update. It is also distributed as each node alerts its neighbors only when its distance vector changes.

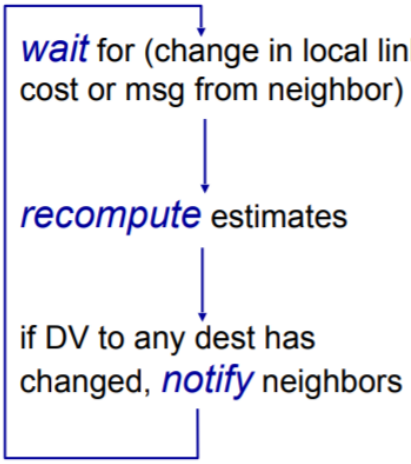


Figure 4: Distance Vector Node Process

When an edge's cost changes, the nodes connected to that edge detects the change, updates routing info, recalculates distance vector, and if there were any changes, the neighbors are notified. If the change is good (the cost of the edge is decreased), the updates happen quickly while if the change is bad (the cost of the edge is increased), the updates happen slowly which is called a "count to infinity" problem and can cause poison reverse. A "count to in nity problem" is when routers exchange information continuously as they keep updating one another to increasing values (going to infinity) due to an edge removal or large edge cost increase. Poison reverse is when a node tells its neighbors that one of the nodes is no longer connected (infinite on the matrix due to the "counting to infinity" problem). For example, in Figure 5, if Z routes through Y to get to X, Z will tell Y its distance to X which is infinite so Y won't route to X through Z.

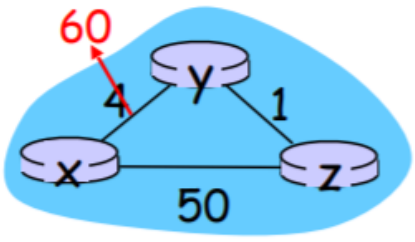


Figure 5: Edge Cost Change

The message complexity, speed of convergence, and robustness of the link state (Dijkstra's algorithm) and distance vector algorithms can be compared. For message complexity, link state has $O(VE)$ messages sent where V is the number of nodes and E is the number of edges while distance vector exchanges only between neighbors. For speed of convergence, link state is a $O(V^2)$ algorithm while distance vector has a varying convergence time due to the "count to infinity" problem. For robustness, link state nodes can alert an incorrect edge cost while distance vector nodes can alert an incorrect path cost.

1.4 Scalable Routing

The previous examples with routers are idealized with each router being identical and within a small network, but realistically, there are billions of routers. Creating matrices similar to the matrices in Figure 3 is not scalable to the real world and the internet is a network of networks where each network admin wants to control the routing in its own network. A group of routers in a region is known as an autonomous system (AS). There are two approaches to scalable routing: intra-AS routing and inter-AS routing. Inter-AS routing is external to the AS and allows one AS to send traffic to another AS.

1.4.1 Intra-AS Routing

Intra-AS routing is internal to that AS and invisible to the outside and all routers within the AS must run the *same* intra-domain protocol. It is also known as interior gateway protocols. Common intra-AS routing protocols are: RIP (Routing Information Protocol), OSPF (Open Shortest Path First), and IGRP (Interior Gateway Routing Protocol). OSPF is publicly available and uses link-state algorithms (route computation by Dijkstra's algorithm). Some features that OSPF has are security which authenticates all OSPF messages to prevent malicious intrusion. Multiple same cost paths allowed, integrated uni- and multi-cast support, and hierarchical OSPF in large domains. The hierarchy of OSPF is described in Figure 6, it is a two-level hierarchy composed of the local area and the backbone. Each node has a detailed area topology and only know the shortest path to nets in other areas. The area border routers summarize the distances to nets in their own area, backbone routers run OSPF routing (limited to backbone), and boundary router's connect to other AS'es.

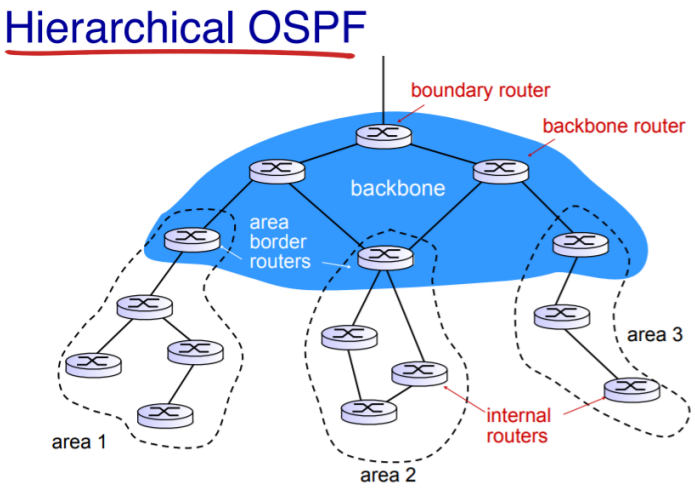


Figure 6: Hierarchical OSPF

1.5 Border Gateway Protocol

Internet inter-AS routing known as Border Gateway Protocol, BGP, is the *de facto* inter-domain routing protocol, the "glue that holds the Internet together".
BGP provides each AS a means to:

- **eBGP**: Use neighboring ASes to obtain subnet reach-ability information
- **iBGP**: Inform all AS-internal routers of reach-ability
- using these terms determine "good" routes to other networks based on reach-ability information and policy

Figure 7 visualizes these concepts stated above to give a better understanding of how the eBGP and iBGP work within the ASes and interaction between ASes.

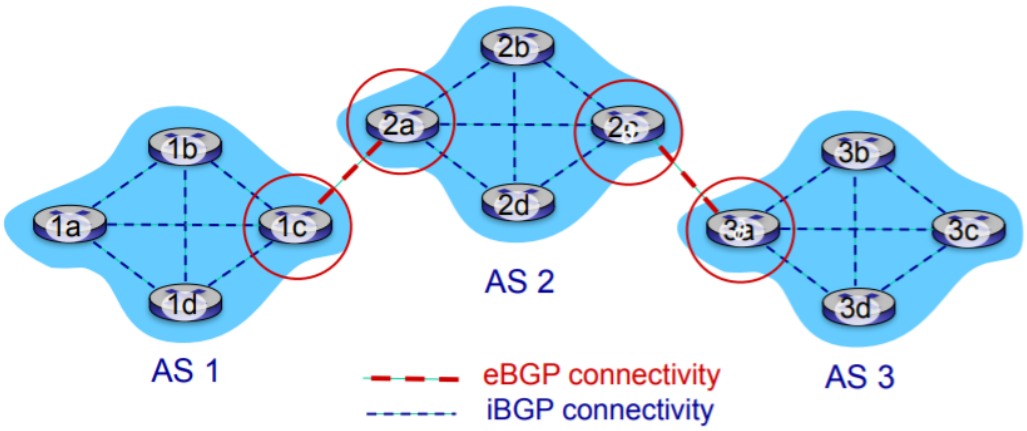


Figure 7: eBGP, iBGP connections

1.5.1 BGP Basics

A BGP session is when two BGP router exchange BGP messages over semi-permanent TCP connection, it advertises "paths" to different destination network prefixes since BGP is a "path vector" protocol.

Path Attributes

Advertised prefixes includes BGP attributes, prefix + attributes = "route", there are two important attributes. AS-PATH which is a list of ASes which the advertised prefix has passed. And NEXT-HOP which indicates specific internal-AS router to next-hop AS.

Policy-based routing:
A gateway that is receiving a route advertisement uses imported policies to accept or decline the

path, policies also determine whether to advertise path to other neighbors. In Figure 8 AS2 router 2c recieves path advertisement from AS3, X (via eBGP of AS3 router 3a). Now based on the AS2 policy router 2c accepts and propagates to all internal routers, then advertises (via eBGP to AS1 1c) path AS2, AS3, X.

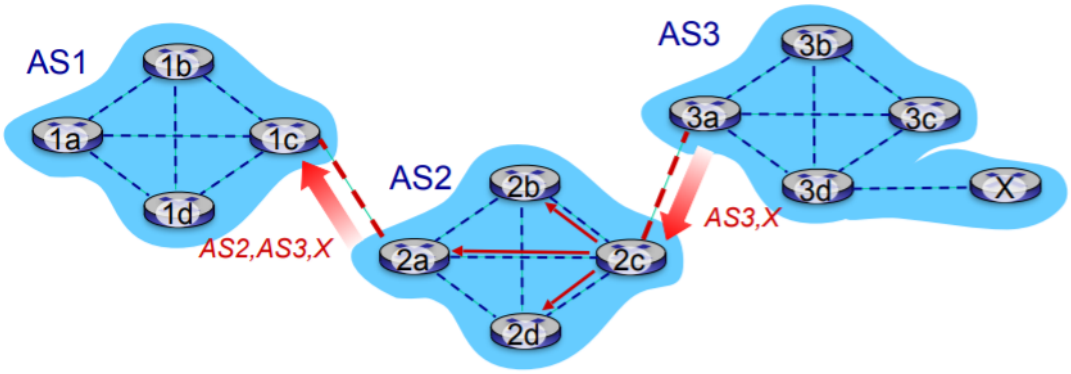


Figure 8: BGP Path Advertisement

BGP messages

BGP messages are exchanged between peers over TCP connections and can have different types:

- OPEN: opens TCP connection to BGP peer and verifies sending BGP peers validity
- UPDATE: advertises new path
- KEEPALIVE: keeps connection alive in absence of UPDATES
- NOTIFICATION: reports errors in previous messages

BGP route selection

The route selection isn't random and has different items to base which route to take. The router may learn about multiple way to reach the destination and it selects the route based on:

1. local preference value attribute: policy decision
2. shortest AS-PATH
3. closest NEXT-HOP router: hot potato routing (choosing a local gateway that has the least intra-domain cost)

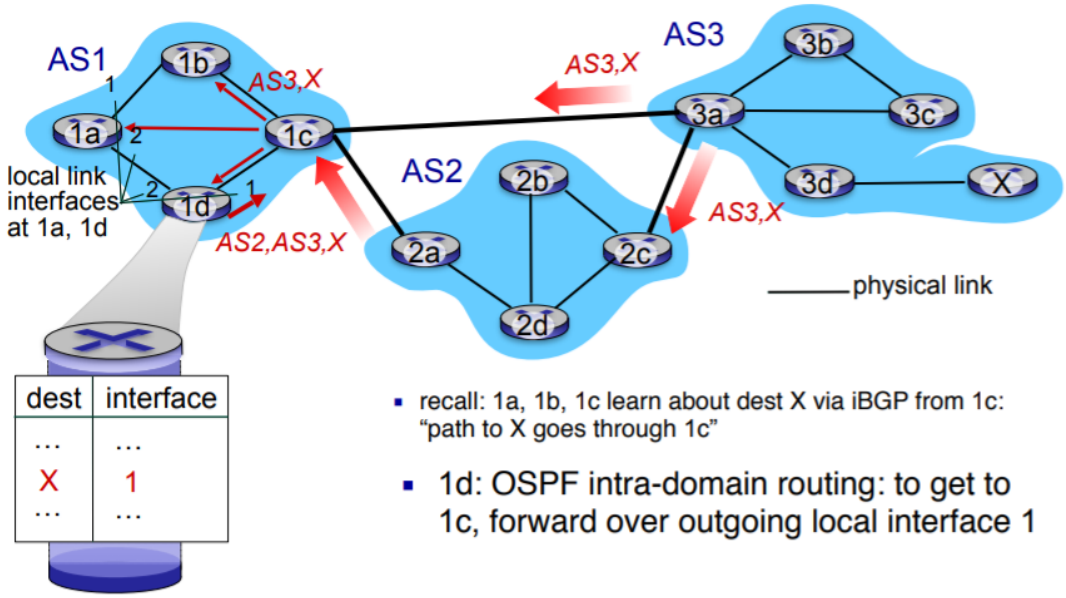
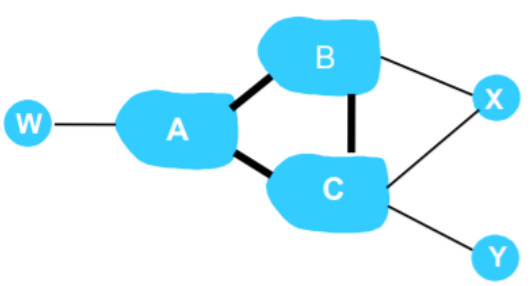
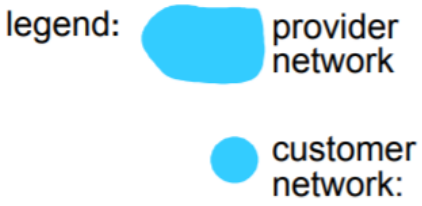


Figure 9: BGP, OSPF, forwarding table entries





Suppose an ISP only wants to route traffic to/from its customer networks (does not want to carry transit traffic between other ISPs)

- A advertises path Aw to B and to C
- B *chooses not to advertise* BAw to C:
  - B gets no "revenue" for routing CBAw, since none of C, A, w are B's customers
  - C does not learn about CBAw path
- C will route CAw (not using B) to get to w

Figure 10: BGP: achieving policy via advertisements