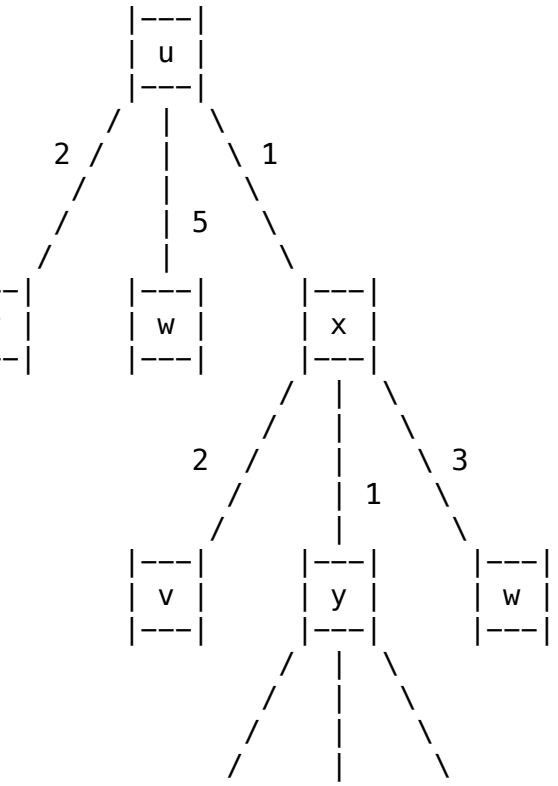
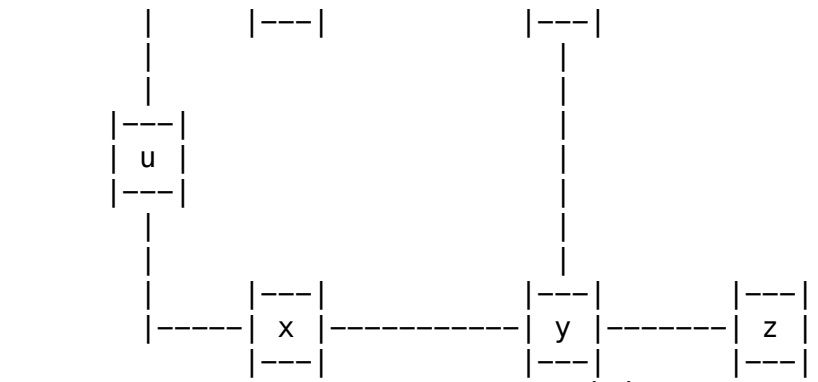
Week.9.txt

- March 8th, 2021
    - Recap From Last Class
        - Dijkstra's Algorithm
            - Is an iterative algorithm
            - In each step it will include a node among the remaining undecided nodes that has the least cost path from the source
            - Easier to understand from the point-of-view of constructing a tree from the source
                - i.e. Tree of Graph



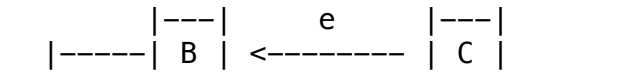
            - The source node is `u`
                - The direct neighbours for `u` are: `v`, `w`, and `x`
            - `x` is a direct neighbour of `u`, with the least cost link
                - The direct neighbours for `x` are: `v`, `y`, and `w`
            - To get to node `w`, from `u`, you can take:
                - The direct path with a cost of 5
                - The path: `u` -> `x` -> `w`, with a cost of 4
            - The end result is a spanning tree, rooted at `u`
    - Dijkstra's Algorithm: Example 2
        - Resulting shortest path from `u`:

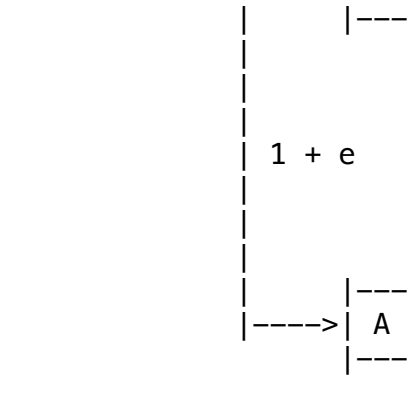


- Resulting forwarding table in `u`:

Destination	Link
v	(u,v)
x	(u,x)
y	(u,x)
w	(u,x)
z	(u,x)

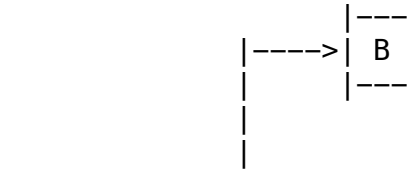
- Dijkstra's Algorithm: Discussion
  - Dijkstra's algorithm is done in an iterative manner
    - In every iteration, the algorithm reduces the number of undetermined nodes by one
      - The total number of iterations is `N`
        - `N` is the total number of nodes in the network
    - In each iteration, the algorithm compares the nodes in the undecided set, and finds the node with the least cost path
      - The cost of this is O(n)
  - The Big-O complexity for Dijkstra's algorithm is: O(n^2)
    - There are more efficient algorithms of Dijkstra's algorithm that can determine the least cost path from the root node to all other nodes in: O(n ∗ log n)
  - Dijkstra's algorithm works best on static links
    - If the cost of the links do not change, then everything works fine
  - Dynamic links may cause oscillations in Dijkstra's algorithm
    - Occurs if the link's cost can change
      - i.e. Costs are associated with changing traffic load
    - Figures Depicting Oscillation In Dijkstra's Algorithm:
      - i.e. Step 1: Initial Phase

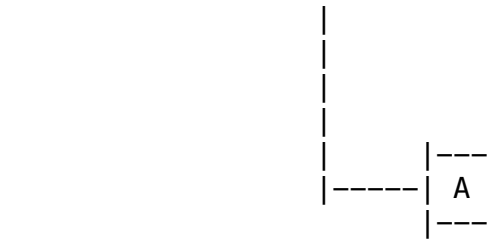




- In total, there are 4 routers; and 4 links to connect the routers
- Initially, there is no traffic that is sent among routers, and along the links
  - But, there is some traffic that is coming into:
    - Router `D` with a load of 1
    - Router `C` with a load of 'e'
      - Note: 'e' is a small number; less than 1
    - Router 'B' with a load of 1
- If the link costs are set equal to incoming traffic, then you can figure out a route
  - Hypothetically, the initial route is:
    - 'D' >>> 'A'
    - 'C' >>> 'B' >>> 'A'
- If the link costs are updated based on the traffic load, we can compute, based on the routing information, that on the link from 'D' to 'A' it carries a traffic load of 1, the link from 'C' to 'B' it carries a traffic load of 'e', the link from 'B' to 'A' carries a traffic load of (1 + e)
  - In this situation, since the link costs are updated, when Dijkstra's algorithm is executed, you want to find out the least cost path from 'D' to 'A', 'C' to 'A', and 'B' to 'A'
    - As a result, you end up with a different route, because there are routes that have a lower cost due to less traffic load on their corresponding links

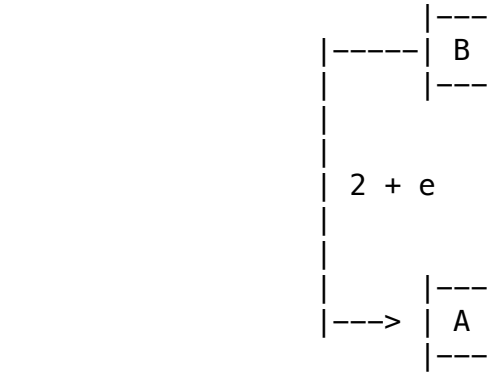
- i.e. Step 2: Update Least Cost Path





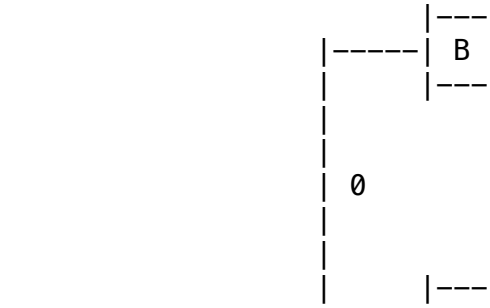
- After the update and the execution of Dijkstra's algorithm, the new route is: 'B' -> 'C' -> 'D' -> 'A', because on the previous graph there is zero traffic flow from 'B' to 'C', 'C' to 'D', and the cost from 'D' to 'A' is 1. On the other hand, the cost from 'C' to 'A' through 'B', on the previous graph, is e + (1 + e) = 1 + 2e
  - Thus, the new route calculated by Dijkstra's algorithm is 'B' -> 'C' -> 'D' -> 'A', instead of 'C' -> 'B' -> 'A'
    - New route cost = 1
    - Old route cost = 1 + 2e
- When traffic starts being routed through the new calculated route, the traffic load increases, requiring the link cost to be updated again

- i.e. Step 3: Update Least Cost Path



- Due to constantly changing traffic, the route needs to be frequently updated by running Dijkstra's algorithm
  - As a result, the traffic needs to be routed through a different direction

- i.e. Step 4: Update Least Cost Path



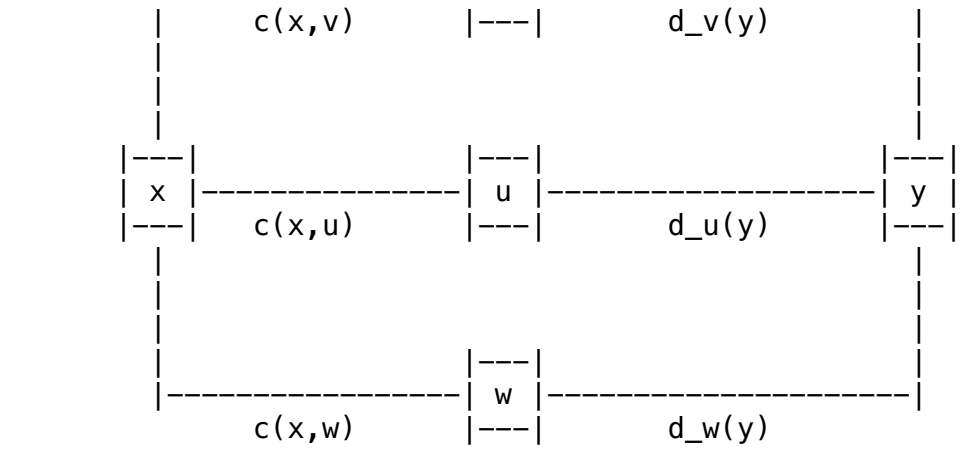
```
                |---> | A | <---------| D | <----|
                      |---|    2 + e  |---|
```

                - The least cost route oscillates, because the link costs are constantly changing
            - The key idea is that Dijkstra's algorithm works well if the link costs are fixed, but struggles if the link costs are dynamic
                - Dynamic link costs leads to oscillation of routes or instability in the network traffic
        - In Dijkstra's algorithm, every single node needs to know the full topology of the network
            - i.e.
                - How many nodes/vertices in the network?
                - How are the nodes connected?
                - What are the link costs in between neighbouring nodes/vertices?
            - In order to piece together the network's topology, nodes need to exchange their local information, such as connectivity and link cost
                - Thus, Dijkstra's algorithm is not a distributed algorithm, because its decision is made based on global information
        - To summarize:
            - Algorithm complexity: `n` nodes
                - Each iteration: Need to check all nodes, `w`, not in `N`
                - `n * (n + 1) / 2` comparisons: O(n^2)
                - More efficient implementations possible: O(n * log(n))
            - Oscillations possible if dynamic link costs are used
                - i.e. Support link cost equals amount of carried traffic
    - Bellman Ford Equation (1)
        - Dijkstra's algorithm is based on the Bellman-Ford equation
        - The Bellman-Ford equation implies that:
            - If you denote `d_x(y)` as the cost of the least path from `x` to `y`, then the following equation holds:

              d_x(y) = min {c(x,v) + d_v(y)}
                        (v in `N`) + x

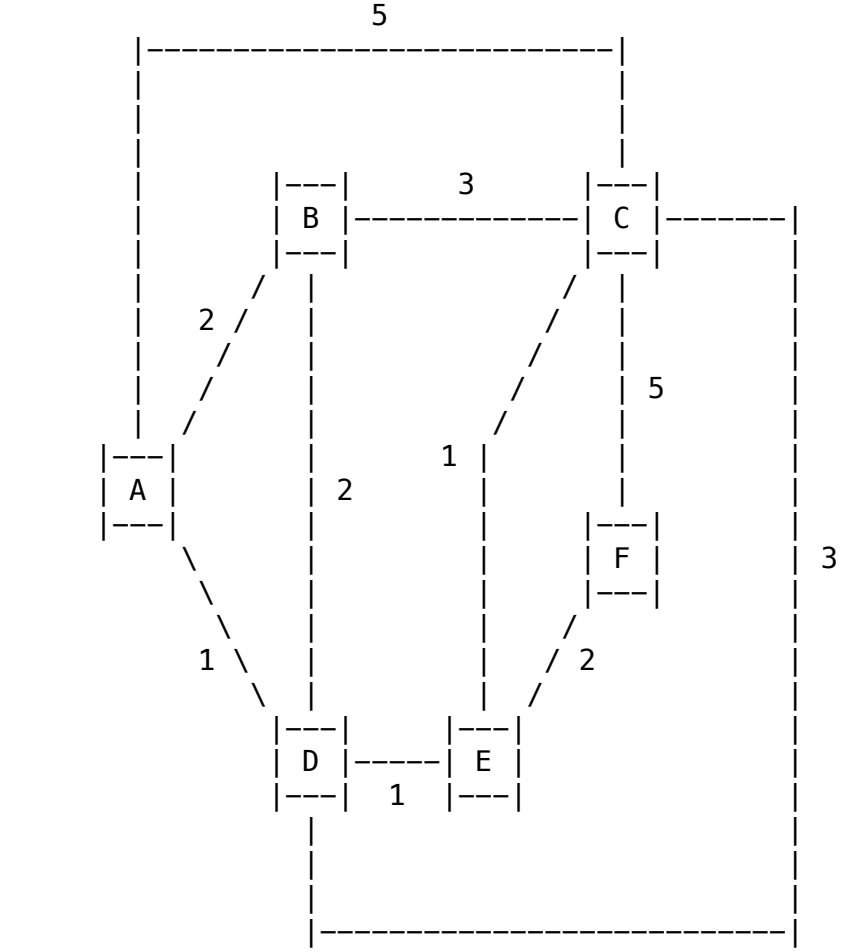
                - `c(x,v)` is the cost from `x` to its neighbour `v`
                - `d_v(y)` is the least cost path `v` to `y`
                - The 'min' is taken over all neighbours of `v` and `x`, including `x` itself
            - You can compute the least cost path from `x` to `y` by adding the cost from `x` to `v` with the least cost path from `v` to `y`, and then taking the minimum among the summation over all access neighbours, including itself
        - i.e. Graphical Example

```
                          |---|
        |-----------------| v |---------------------|
```



- Question: What is the least cost path from `x` to `y`?
  - Answer:
    - There might be multiple hops that a packet needs to go through from `x` to `y`. The Bellman-Ford equation tells us that we can look at the neighbours of `x` to determine the least cost path. Hypothetically, we already know the least cost path from `v` to `y`, `u` to `y`, and `w` to `y`
      - Least cost path of `v` to `y` is: d_v(y)
      - Least cost path of `u` to `y` is: d_w(y)
      - Least cost path of `w` to `y` is: d_u(y)
    - Bellman-Ford equation tells us that in order to find the least cost path from `x` to `y`, we need to go through one of the neighbours of `x`, because `x` is not directly connected to `y`. Additionally the least cost path will be determined from the neighbours of `x` to `y`. Thus, the least cost path is based on the minimum of `x` to a neighbour, and the neighbour to `y`, for all neighbours of `x`
      - If `x` has a neighbour with an inexpensive path to the destination, and the link cost to the neighbour is small, and as long as the summation of these two values is less than other neighbours of `x`, then this neighbour is the candidate to utilize to reach the destination
- The benefit of the Bellman-Ford equation is that nodes do not need to know the full topology of the network
  - If `x` wants to send a packet to `y`, it does not need to know how its neighbours, `u`, `v`, and `w`, are connected to `y`. All that matters is the least cost path from the neighbours of `x` to `y`
    - Instead of having to obtain the full topology, the source node just needs some information from its neighbour(s)

        - This facilitates a fully distributed algorithm, where neighbouring vertices can exchange local information
            - Based on this information, and applying Bellman-Ford equation, the least cost path can be figured out
- Bellman Ford Equation (2)
    - i.e. Graphical Example



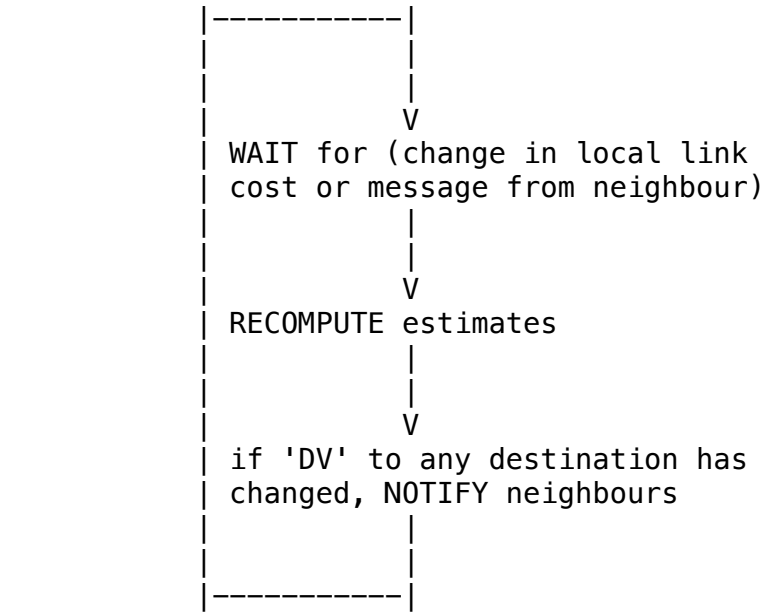
        - The following is known:
            - Least cost path from `B` to `F`: d_B(F) = 5
            - Least cost path from `D` to `F`: d_D(F) = 3
            - Least cost path from `C` to `F`: d_C(F) = 3
        - Question: What is the shortest path from `A` to `F`?
            - Answer:
                - We know the link cost from source-node `A` to its neighbours, and from those neighbours we know the least cost path to `F`
                    - Neighbours of `A`: `B`, `C`, and `D`
                - The least cost path from `A` to `F` can be represented by the following equation:
                  d_A(F) = min {
                    c(A,B) + d_B(F),
                    c(A,D) + d_D(F),
                    c(A,C) + d_C(F)

}
d_A(F) = min {
2 + 5,
1 + 3,
5 + 3
}
d_A(F) = min {
7,
4,
8
}
d_A(F) = 4
- Therefore, the least cost path from `A` to `F` is: 4
- The path is: `A` -> `D` -> `E` -> `F`
- If a packet needs to be sent to `F`, from node `A`, node `A` needs to forward the packet to node `D`, and node `D` will handle the rest

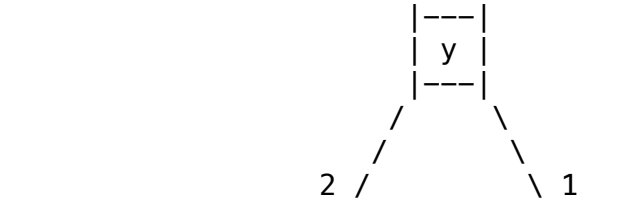
- Distance Vector Routing
  - The distance vector routing is a fully distributed algorithm to compute the least cost path
    - Similar to Dijkstra's algorithm, it goes through multiple iterations
      - Unlike Dijkstra's algorithm, there is no central/ global state that needs to be maintained
  - Distance vector: A node's least known cost to other nodes
  - On each iteration, the node will periodically send its distance vector (DV) estimation to neighbours, but ONLY when its distance vector (DV) changes
    - For each node, the 'DV' corresponds to the least cost path for a particular destination
      - i.e. `A` may report that the least cost path to `B` is 2, the least cost path to `D` is `, the least cost path to `C` is 5, and to all other nodes is 'infinite'
    - The 'DV' may not be the final result, but at a particular iteration, it is the node's best knowledge
  - Upon reception of a 'DV', from a neighbour, the node updates its own 'DV' utilizing the Bellman-Ford equation: D_x(y) <-- min_v{c(x,v) + D_v(y)} for each node `y` in `N`
    - Nodes obtain updated distance vectors from their neighbours, add the new distance vector with the link cost to the neighbour, and find the minimum. Finally, the node updates its own distance vector
  - Note: Initially, the node may not have the correct least cost path. It will take several iterations in order to figure out the precise least cost path to each destination in the network
- Distance Vector Algorithm
  - Note: 'DV' = Distance Vector
  - Can act upon local information

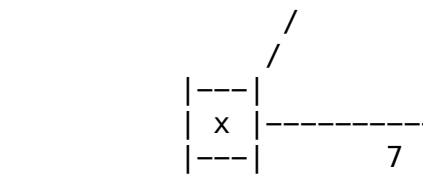
    - Even though the node only utilizes local information, it can come to an agreement of least cost path from source to destination
- Each router needs to maintain a table
    - The table contains the best known distance from itself and its neighbours to all routers
- Each local iteration caused by:
    - Local link cost change
      OR
    - Message, containing updated distance vector, from neighbours(s)
- Nodes notify neighbours ONLY when their own 'DV' changes
    - Neighbours then notify their neighbours if necessary
- The distance vector algorithm is suitable for distributed implementation because every single router carries out the same set of steps
    - As long as each node follows the steps, then after some iterations, the route should converge
    - The steps can be done in a synchronous or asynchronous manner
        - Performance may vary based on which is used, but the convergence is the same
- i.e. Figure of Distance Vector Algorithm

    Each node:



- i.e. Distance Vector Algorithm Example
    - Graphical Representation of Network





- In total, there are 3 routers and 3 links that interconnect the routers
- The numbers on the link correspond to the link cost
- The distance vector table for each node is below
  - Rows correspond to source vertex
  - Columns correspond to destination vertex
  - Note: '∗' corresponds to 'infinity'

- FIRST Iteration:
  - Distance vector table on `x`:

node `x` table	cost to: `x`	cost to: `y`	cost to: `z`
from `x`	0	2	7
from `y`	∗	∗	∗
from `z`	∗	∗	∗

  - The distance vector entry from source `x` to destination `x` is: 0
  - The distance vector entry from source `x` to destination `y` is: 2
  - The distance vector entry from source `x` to destination `z` is: 7
  - Since the other 2 vertices are neighbours of `x` their corresponding distance vector is instantiated based on link cost to the node
    - i.e. The distance vector from `x` to `y` is the same as the link cost from `x` to `y`. Same applies for `x` to `z`
  - The entries for `y` and `z` are set to infinity because `x` has not received any information from `y` or `z`
  - Distance vector table on `y`:

node `y` table	cost to: `x`	cost to: `y`	cost to: `z`
from `x`	∗	∗	∗
from `y`	2	0	1

from `z`	*	*	*
        - Initially, `y` is only able to populate its own distance vector to its neighbours
          - Cost from `y` to `x` is 2
          - Cost from `y` to `z` is 1
    - Distance vector table on `z`:
        |----------|-----------------|
        | node `z` | cost to: |
        | table |-----|-----|-----|
	`x`	`y`	`z`
from `x`	*	*	*
from `y`	*	*	*
from `z`	7	1	0
        - Initially, `z` is only able to populate its own distance vector to its neighbours
          - Cost from `z` to `x` is 7
          - Cost from `z` to `y` is 1
- NEXT Iteration:
    - Distance vector table on `x`:
        |----------|-----------------|
        | node `x` | cost to: |
        | table |-----|-----|-----|
	`x`	`y`	`z`
from `x`	0	2	3
from `y`	2	0	1
from `z`	7	1	0
        - On every iteration, node `x` exchanges information with its neighbours, `y` and `z`
        - The distance vector from `y` and `z` are copied into the distance vector table on `x`
        - Upon receiving new information from its neighbours, `x` changes its own distance vector table via the Bellman-Ford equation
        - The cost from source `x` to destination `x` is unchanged because it is 0
        - To get to `y`, from source `x`, you can go directly to `y`, or go to `z` and then to `y`
          - Option 1: Add the link cost of `x` to `y`, and the least cost path of `y` to `y`
            - i.e. 2 + 0 = 2
          - Option 2: Add the link cost of `x` to `z`,

        and the least cost path of `z` to `y`
          - i.e. 7 + 1 = 8
    - The cost from source `x` to destination `y` can be modelled via Bellman-Ford equation:
      D_x(y) = min {
        c(x,y) + D_y(y),
        c(x,z) + D_z(y)
      }
      D_x(y) = min {
        2 + 0,
        7 + 1
      }
      D_x(y) = 2
        - The distance vector from source `x` to destination `y` is updated to 2
    - To get to `z`, from source `x`, you can go directly to `z`, or go to `y` and then to `z`
        - Option 1: Add the link cost of `x` to `z`, and the least cost path of `z` to `z`
          - i.e. 7 + 0 = 7
        - Option 2: Add the link cost of `x` to `y`, and the least cost path of `y` to `z`
          - i.e. 2 + 1 = 3
    - The cost from source `x` to destination `z` can be modelled via Bellman-Ford equation:
      D_x(z) = min {
        c(x,z) + D_z(z)
        c(x,y) + D_y(z),
      }
      D_x(z) = min {
        7 + 0
        2 + 1,
      }
      D_x(z) = 3
        - The distance vector (DV) from source `x` to destination `z` is updated to 3
          - Since the 'DV' table is updated, `x` sends it to its neighbours, `y` and `z`, on the next iteration
- Distance vector table on `y`:

node `y` table	cost to:		
	`x`	`y`	`z`
from `x`	0	2	7
from `y`	2	0	1
from `z`	7	1	0

|----------|-----|-----|-----|
- There is no change for the distance vector (DV) table corresponding to `y`
  - Thus, `y` does not send its 'DV' table to its neighbours, `x` and `y`
- Distance vector table on `z`:

node `z` table	cost to:		
	`x`	`y`	`z`
from `x`	0	2	7
from `y`	2	0	1
from `z`	3	1	0

- The cost from `z` to `x` is updated from 7 to 3
  - Because, the cost from `x` to `z` is 7, while the cost from `x` to `y` to `z` is 3

- LAST Iteration:
  - Distance vector table on `x`:

node `x` table	cost to:		
	`x`	`y`	`z`
from `x`	0	2	3
from `y`	2	0	1
from `z`	3	1	0

  - Distance vector table on `y`:

node `y` table	cost to:		
	`x`	`y`	`z`
from `x`	0	2	3
from `y`	2	0	1
from `z`	3	1	0

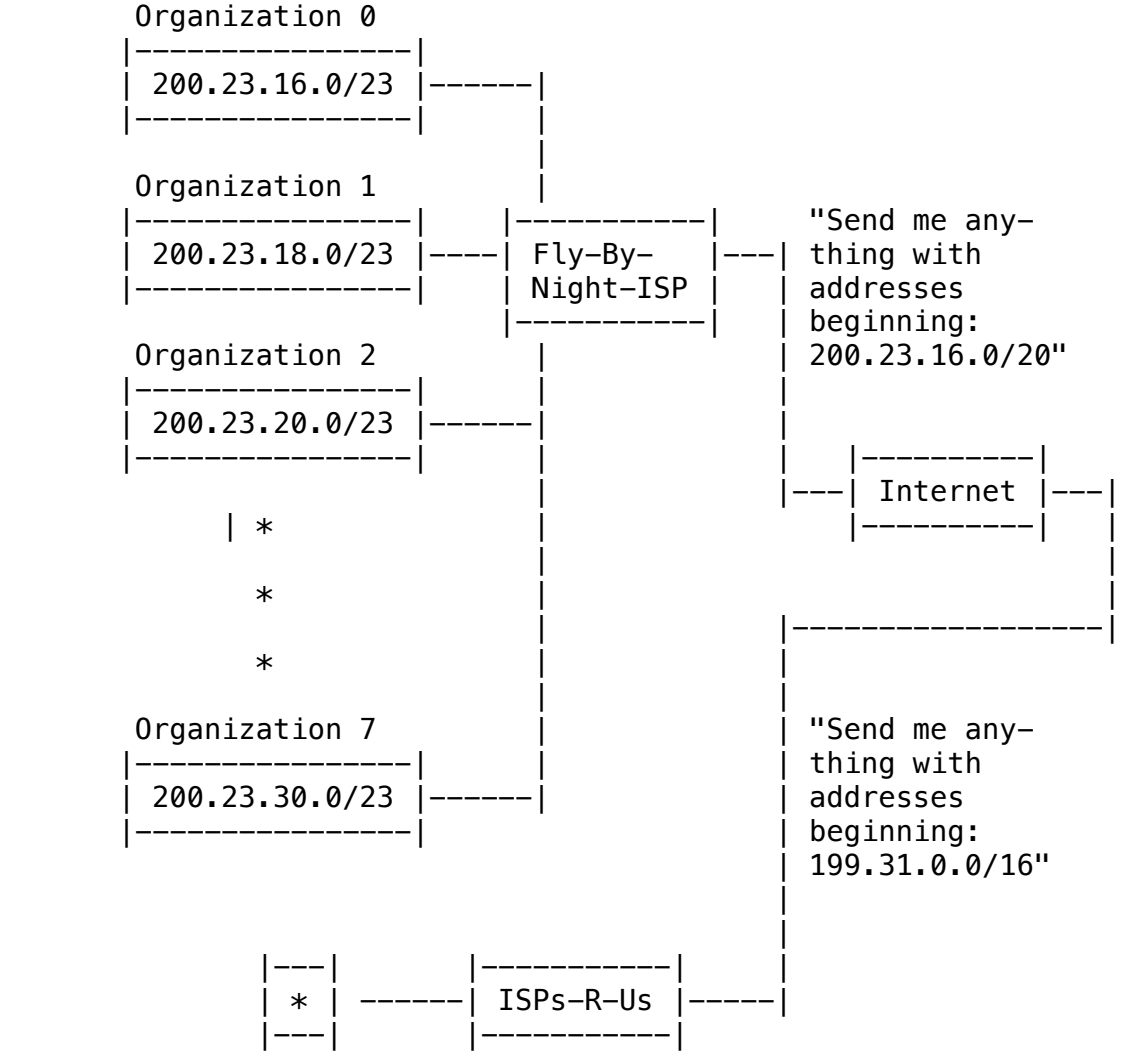
  - Distance vector table on `z`:

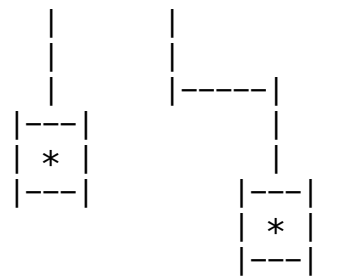
node `z` table	cost to:		
	`x`	`y`	`z`

|----------|-----|-----|-----|
from `x`	0	2	3
from `y`	2	0	1
from `z`	3	1	0

        - The distance vector table for all nodes are updated via the Bellman-Ford equation until all tables are the same, and no more change is required
            - Once this is achieved, the algorithm terminates
        - Future changes (i.e. Disconnectivity of certain links) requires the distance vector table to be updated until it converges, and the least cost path is achieved for all nodes
    - Tip
        - You need to be able to compute the distance vector table by hand
        - Know the difference between the distance vector algorithm, and Dijkstra's algorithm
- Routing Protocol In The Internet
    - Dijkstra's algorithm and the distance vector algorithm are two fundamental algorithms that are employed in the Internet
        - The actual implementation depends on the protocol itself
            - There are different routing protocols available in the Internet
    - Question: How exactly does routing work in the Internet?
        - Think about this going forward
    - Internet routing protocols
        - Responsible for constructing and updating the forwarding tables at router
    - Scalability
        - Is a huge issue for the Internet
        - With 32-bit IP addresses, there are up to 4 billion possible destination-hosts
            - Hence, we cannot store all possible destination addresses in forwarding tables
                - 1 entry per address is infeasible
            - In addition to the computation complexity, the storage complexity of storing all possible destination-hosts is huge
            - Forwarding table exchange would swamp links
    - Administrative Autonomy
        - Internet = Network of networks
            - The Internet is organized in an hierarchical manner
                - There are different tiers of networks
                - Each ISP is responsible for its own network
                    - They may or may not have information from other networks
            - We must minimize the complexity of storage tables

        - Different ISPs have their own view of their own internal topology, and may not share this information with the outside world
      - Each network admin may want to control routing in its own network
      - Recall that: in routing/forwarding tables, we only need to keep track of blocks of IP addresses, and have 1 entry for each block
        - This includes the longest prefix for each block, and the outgoing port for that particular block
- Hierarchical Addressing: Route Aggregation
  - Using a combination of address allocation, only 1 entry per block needs to be stored in the routing/forwarding table(s)
    - To achieve this, you should do your best to allocate a continuous block of address, and address aggregation within the network
  - Hierarchical addressing allows efficient advertisement of routing information
  - i.e. Diagram of Route Aggregation

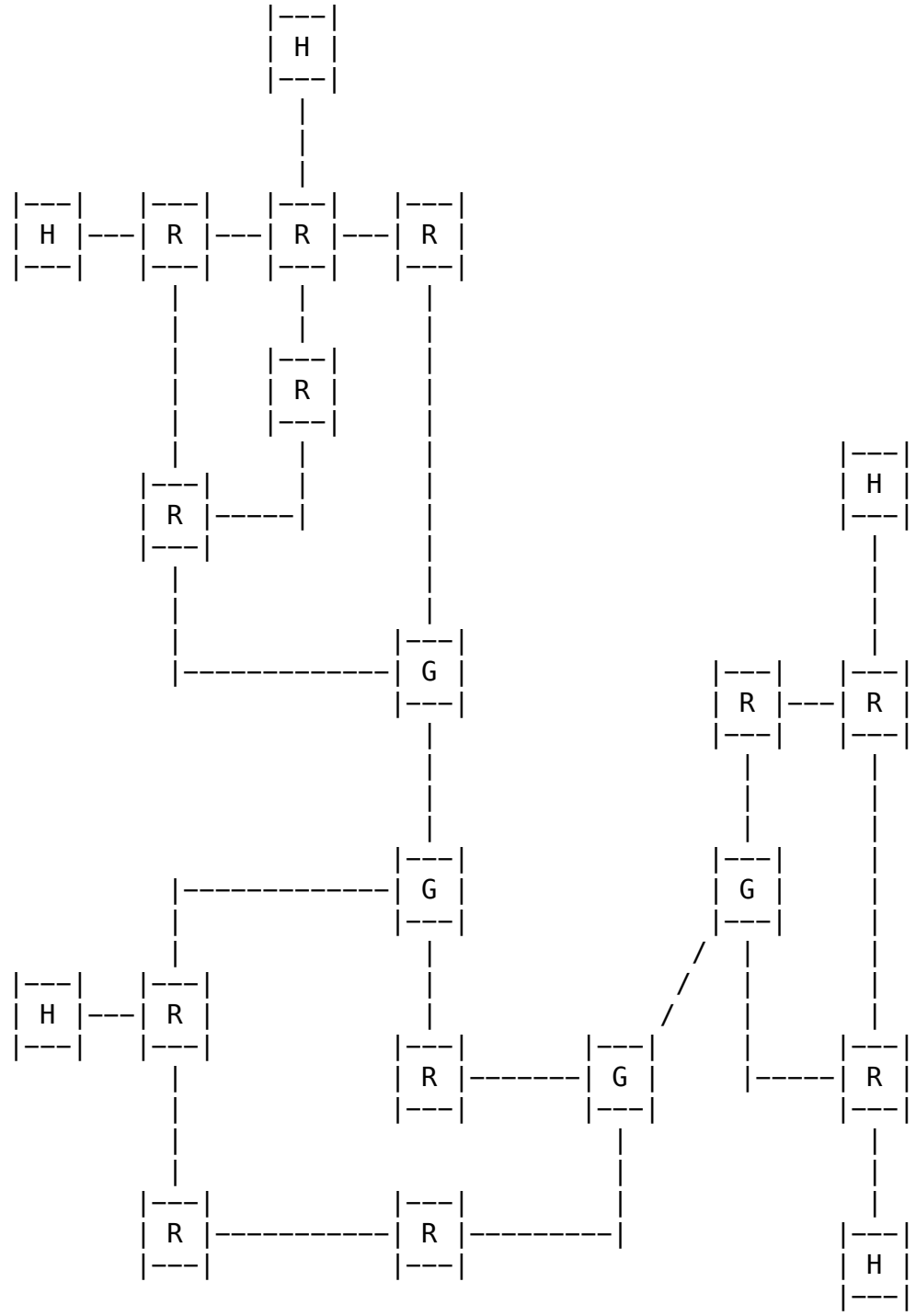




    - There are 8 organizations that are customers of 'Fly-By-Night-ISP'
    - The 'Fly-By-Night-ISP' is able to aggregate the block of addresses associated with each organization, using a larger block
      - 'Fly-By-Night-ISP' tells the rest of the Internet to send any address in the range `200.23.16.0/20`, instead of specifying the address range corresponding to each organization
    - Aggregation allows the reduction of entries in the forwarding table
      - i.e. Instead of having to store 8 entries, which all correspond to the output port, only 1 entry is required for all 8 organizations
- Hierarchical Routing (1)
  - Individual ISPs maintain their own topology, and do not necessarily share the information from the outside world
  - Organization allows for the notion of hierarchical routing
  - Autonomous system (AS): Region of a network under a single administrative entity
    - Administrative entity refers to ISP
    - An 'AS' can be thought of as region of networks that belong to the same ISP
    - The granular of information that is available within an autonomous system tends to be different from the outside autonomous system
  - The Internet is organized in a hierarchical manner, consisting of different multiple autonomous systems
    - Some autonomous systems are tier 3 ISPs, some maybe tier 2 ISPs, etc.
  - At a high level, Internet routing is a combination of routing within autonomous system, and routing across autonomous systems
    - Routing within an 'AS' is sending traffic within customer networks that are subscribed to the same ISP
  - Hierarchical structuring allows for better scalability, and reduces the number of routing table entries in the Internet
    - At the top level, you only need to make routing decisions from one 'AS' to another, without exposing yourself to the detailed information of the customer network
      - But, when you get to an 'AS', you should have more

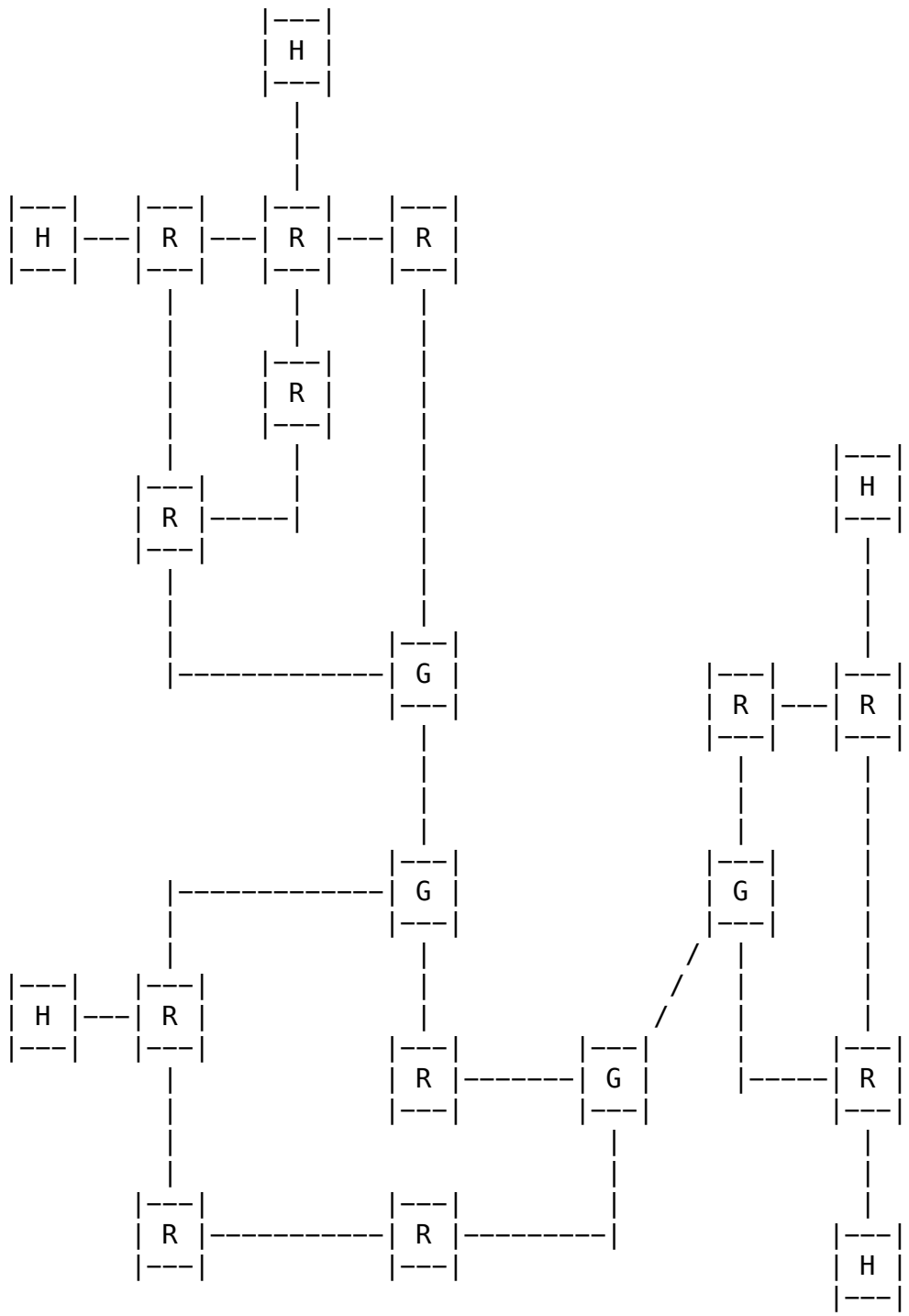
detailed routes to individual customer network

- i.e. Diagram of Autonomous Systems



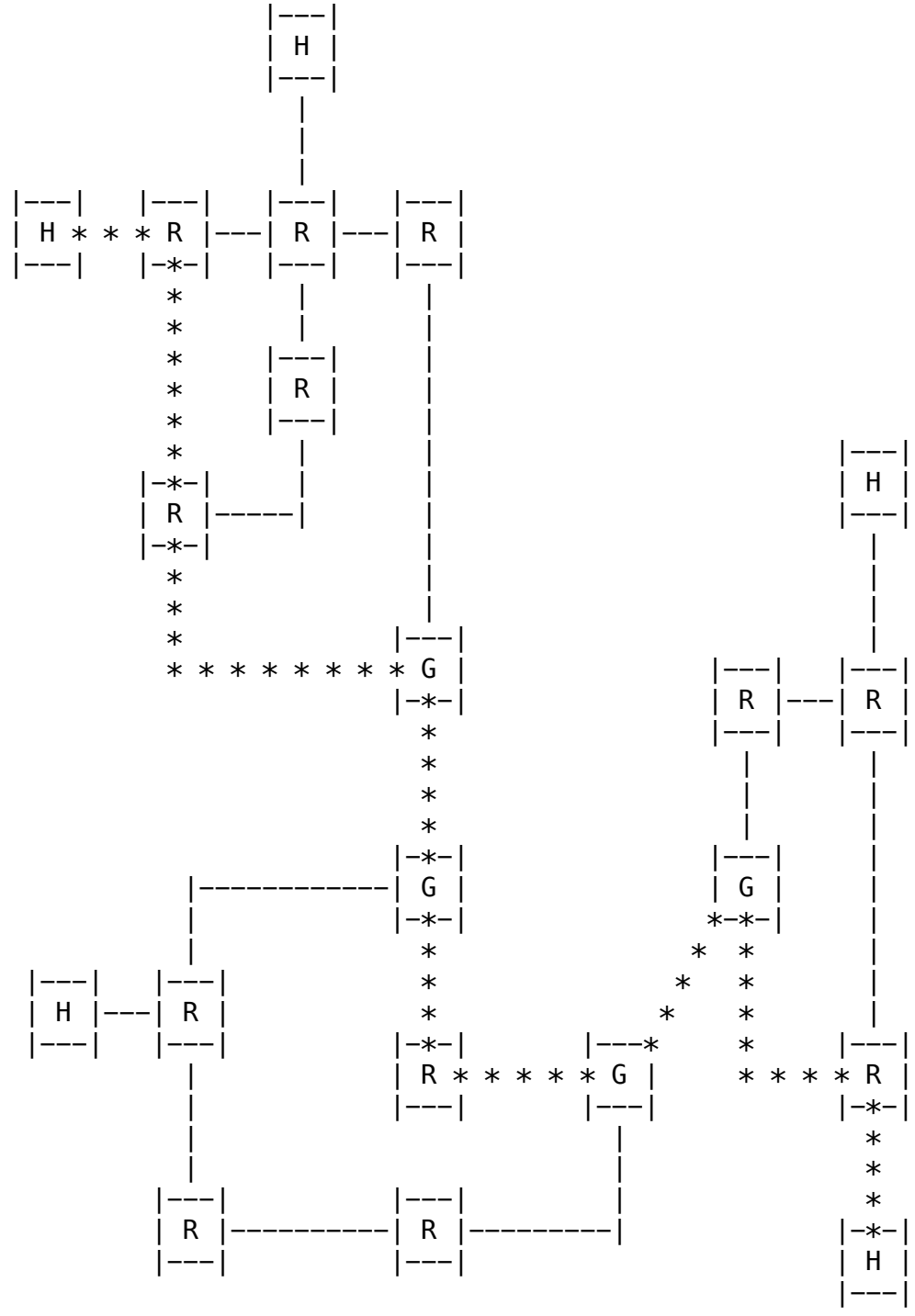
  - Legend:
    - 'R' = Routers
    - 'G' = Gateway Routers
    - 'H' = Hosts

- Hierarchical Routing (2)
  - i.e. Diagram of Autonomous Systems



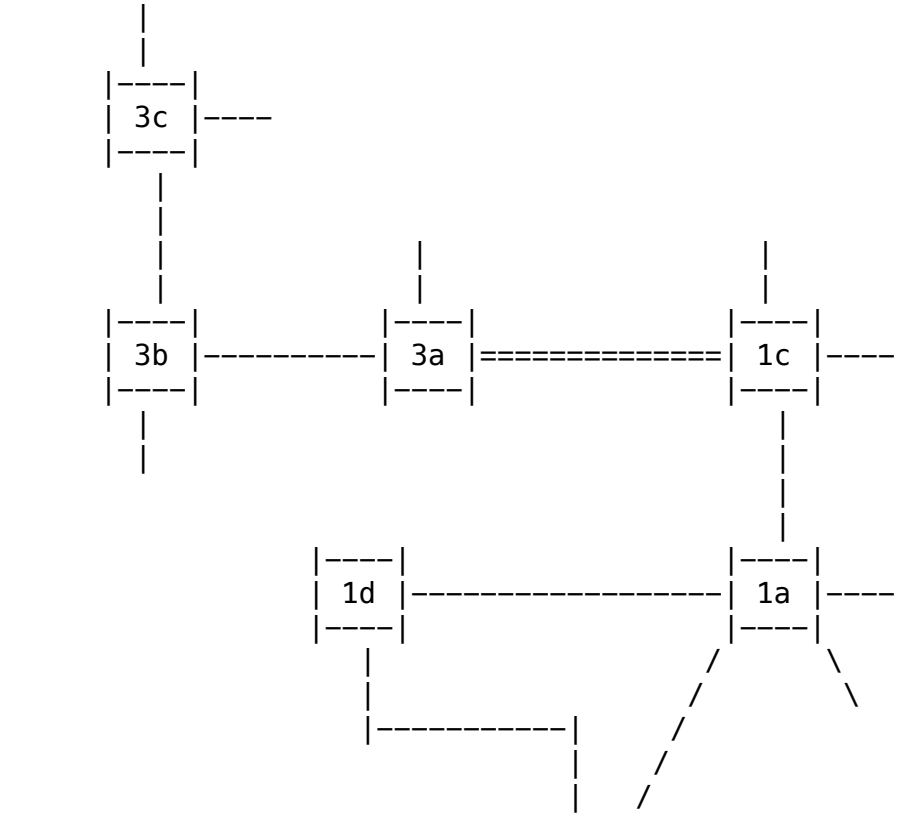
- Legend:
  - 'R' = Routers
  - 'G' = Gateway Routers
  - 'H' = Hosts
- In the network, routers that connect autonomous systems are known as gateway routers
- In an 'AS', non-gateway routers are called internal routers

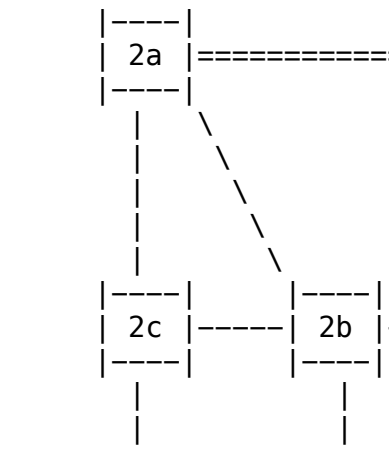
    - These routers, within an autonomous system, are not connected to other autonomous systems
  - Internal routers and gateway routers work together to determine the route from source-host to end-host
    - This path may span across multiple autonomous systems
- Hierarchical Routing (3)
  - i.e. Figure of Route/Path Across Autonomous Systems



        - Legend:
            - 'R' = Routers
            - 'G' = Gateway Routers
            - 'H' = Hosts
            - '*' = The path taken from one 'AS' to another 'AS'
- Hierarchical Routing (4)
    - Internet/hierarchical routing is done at 2 levels
    - 'AS': Region of network under a single administrative entity
    - Each 'AS' runs its own intra-domain routing protocol that establishes routes within its domain
        - This is possible because each 'AS' has:
            - Full knowledge about its topology
            - Full control over what kind of routing protocol is utilized
            - Full control over what kind of information needs to be exchanged between routers within the same domain
    - Different autonomous systems employ different (intra-domain) routing protocols
        - The commonly used routing protocols are:
            - Link state
                - i.e. Open Shortest Path First (OSPF)
                    - OSPF implements Dijkstra's algorithm
                    - In essence, it exchanges link state information to get the fully topology, and runs Dijkstra's algorithm to determine the least cost path
            - Distance vector
                - i.e. Routing Information Protocol (RIP)
    - Autonomous systems run inter-domain routing protocol(s) that establishes routes between domains
        - i.e.
            - From a customer network to another customer network
            - From a gateway router to another gateway router
            - To/From a gateway router to/from a customer network
        - Path Vector is a commonly used protocol
            - i.e. Border Gateway Protocol (BGP)
    - On the second level, you can extract autonomous systems as single vertices, and build a network topology that simply interconnects multiple autonomous systems
        - At this stage, the internal structure/topology within each autonomous system is hidden
            - You can think of an 'AS' as a single vertex, in a graph where the edges between the vertices correspond to the links between neighbouring autonomous systems; this is the link between the gateway routers
        - At this abstraction level, you can run inter-domain routing protocols that establish routes between the domains
    - By combining inter-domain, which routes from domain-to-

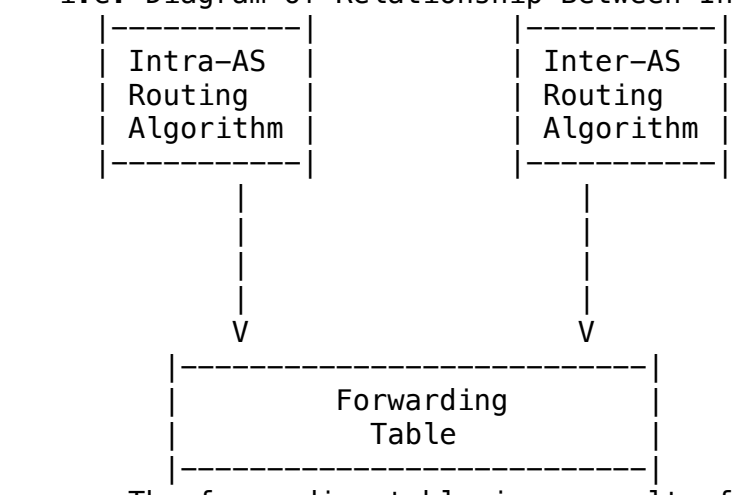
domain, and intra-domain, which routes within a domain, you can figure out the end-to-end path from any network host to another network host
    - The benefit of hierarchical routing is that the information for every single destination network does not need to be stored
        - Instead, the information (IP addresses) can be aggregated
    - Hierarchical routing is a very important concept in the Internet, because:
        - It helps realize scalability in routing table maintenance
        - Allows nodes/vertices to calculate routes in absence of the full knowledge of the topology of every single 'AS'
- Interconnected ASes
    - Forwarding table is configured by both intra-domain and inter-domain 'AS' routing algorithm
        - They work together to determine which outgoing port a particular packet needs to be forwarded to
        - Intra-AS sets entries for internal destinations
            - Internal destinations = Within the same 'AS'
        - Inter-AS & Intra-AS sets entries for external destinations
            - External destinations = Destination address outside the 'AS'
    - i.e. Diagram of Interconnected Autonomous Systems





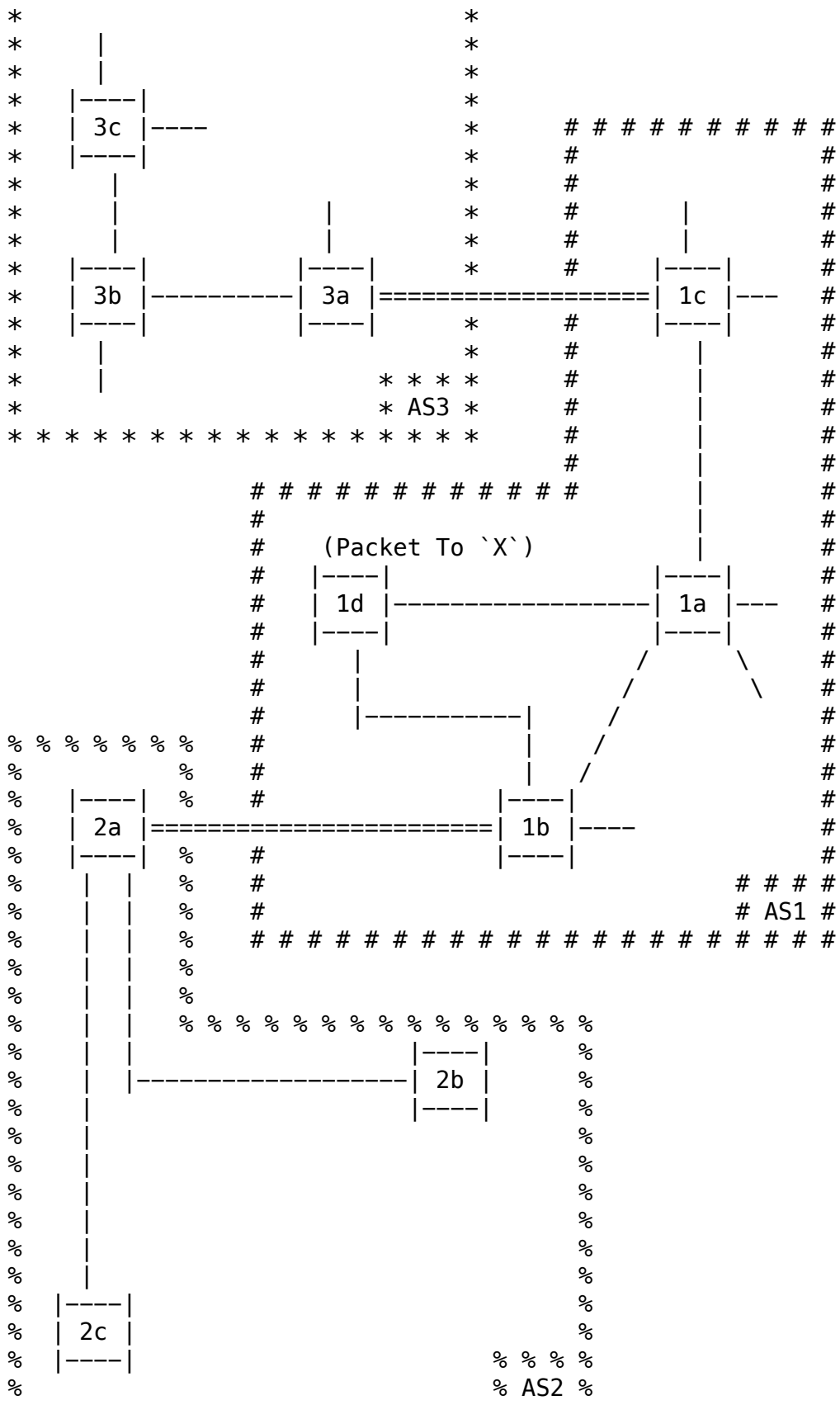
- There are 3 autonomous systems
- Every node corresponds to a router
  - `1b`, `2a`, `1c`, `3a` are gateway routers because they connect to a router that is not within the same autonomous system
    - Gateway routers are important because they interconnect different autonomous systems
  - `3c`, `3b`, `1a`, `1d`, `2c`, `2b` are internal routers
- '===' represents a connection between gateway routers
- '---' represents a connection between internal routers

- i.e. Diagram of Relationship Between Intra-AS & Inter-AS

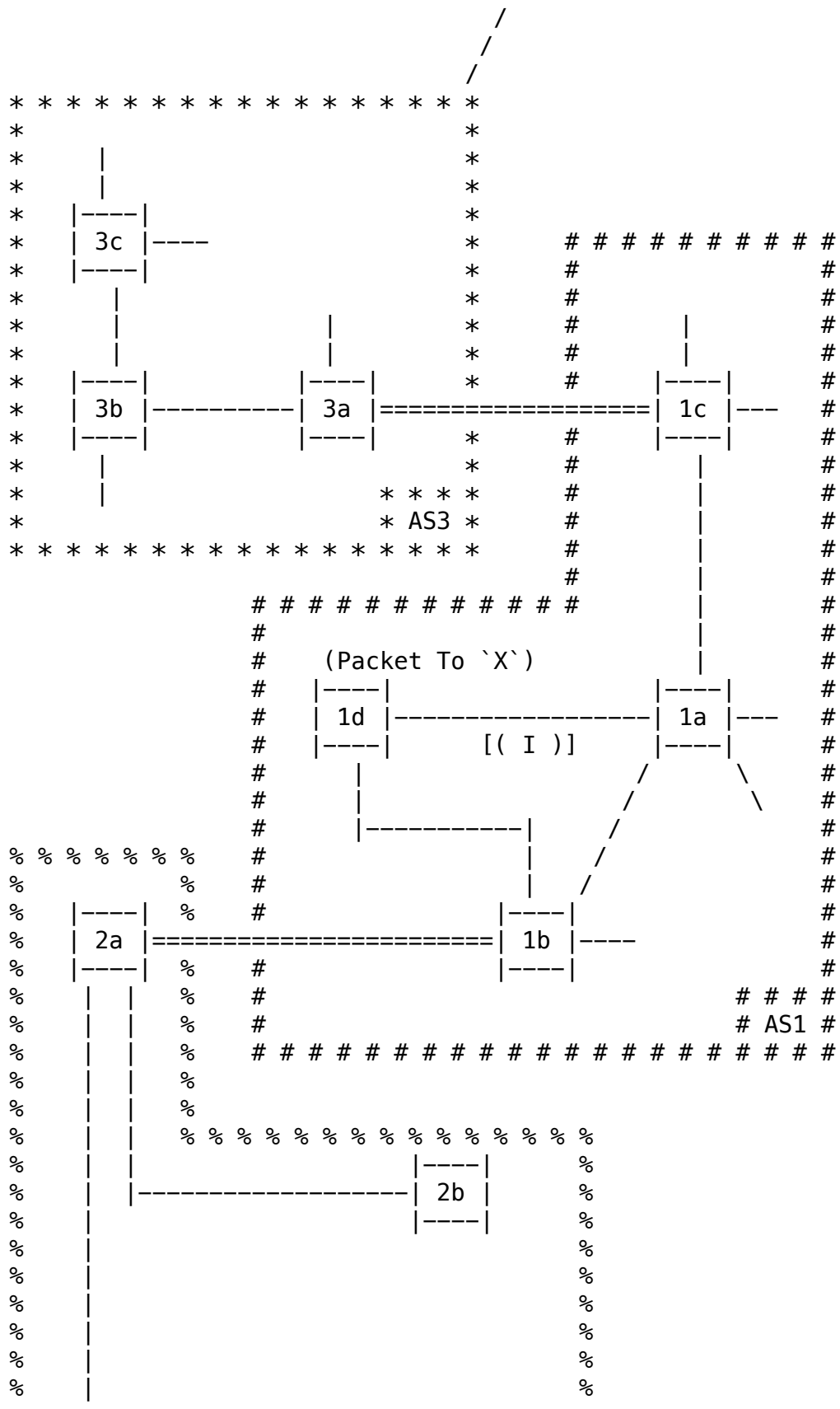


- The forwarding table is a result of inter-domain routing and intra-domain routing

- Inter AS Tasks
  - i.e. Diagram of Routing


3c
3b
3a
AS3
1c
(Packet To `X`)
1d
1a
2a
1b
AS1
2b
2c
AS2

% % % % % % % % % % % % % % % % % % % %
- 'AS3' is encased in '*'
    - The internal routers are: `3c`, `3b`
    - The gateway routers are: `3a`
- 'AS2' is encased in '%'
    - The internal routers are: `2b`, `2c`
    - The gateway routers are: `2a`
- 'AS1' is encased in '#'
    - The internal routers are: `1a`, `1d`
    - The gateway routers are: `1c`, `1b`
- Suppose router `1d` in `AS1` receives a datagram/packet whose destination is outside of `AS1` (i.e. X). Which gateway router should `1d` forward the packet to? How will `1d` figure this out? Assume that `X` is a customer network of `AS3`
    - In order to send the packet to `X`, `1d` needs to know:
        - Which gateway router needs to be utilized in order to reach an 'AS' that is outside of `AS1`
        - The internal route to follow in order to reach the gateway router, `1c`
    - `1d` needs to know that in order to reach `X`, it should go toward `AS3` via the gateway router `1c`, that connects `AS1` to `AS3`
        - `1d` should not forward the packet to `AS2`, because `AS2` is not connected to `X`
    - The requirements can be broken down into 3 parts. `AS1` needs:
        1. To learn which destinations are reachable through `AS2` and which through `AS3`
            - This is the job of the inter-domain routing protocol
        2. To propagate this reachability info to all routers in `AS1`
            - This is the job of the inter-domain routing protocol
        3. To figure out which gateway routers are connected to the respective 'AS'?
            - This information is static, because it entirely depends on the topology; generally, it does not change unless some kind of reconfiguration takes place
                - i.e. `1b` connects `AS1` with `AS2`
                - i.e. `1c` connects `AS1` with `AS3`
- Example: Setting Forwarding Table In Router 1D
    - i.e. Diagram of Routing

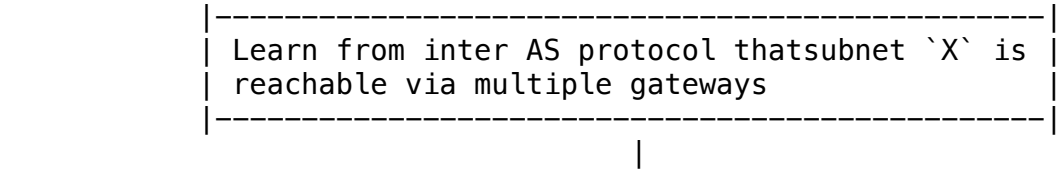

3c
3b
3a
1c
AS3
(Packet To `X`)
1d
[(I)]
1a
2a
1b
2b
AS1

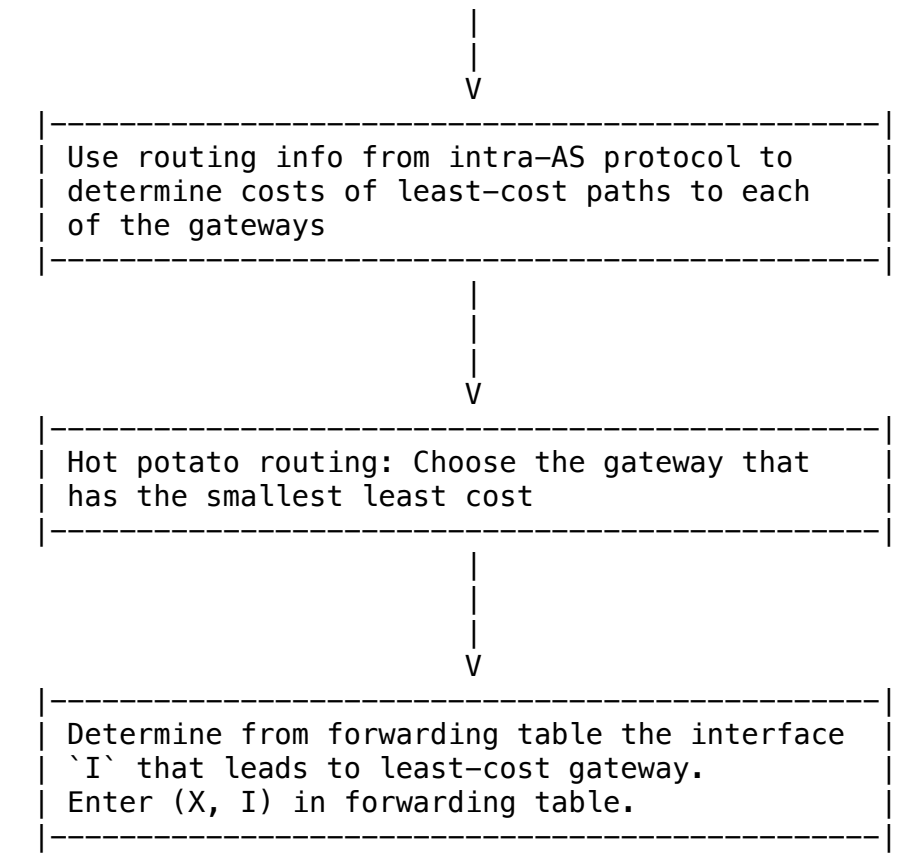
```
%  |----|                                  %
%  | 2c |                                  %
%  |----|                          % % % %
%                                  % AS2 %
% % % % % % % % % % % % % % % % % % % % % %
```

- Note: Interface `I` connects router `1d` and `1a`
- Steps involved in sending a packet to `X` from `1d`:
    1. Suppose `AS1` learns from the inter-AS protocol that subnet `X` is reachable from `AS3` (gateway `1c`) but not from `AS2`
        - This is done through the inter-domain routing protocol
            - It operates at the level of extracting each 'AS' as a vertex
    2. Inter-domain routing protocol is utilized to propagate reachability information obtained from the previous step to all internal routers
        - This information should not be limited to gateway routers; internal routers should know this information as well
    3. Upon reception of reachability information from the previous steps, router `1d` will know that in order to reach destination `X`, the packet/datagram has to go through `AS3`
        - This implies that the packet needs to go through gateway router `1c`
        - The interface `I` is on the least cost path to (gateway) router `1c`
        - This is done via intra-AS routing
    4. The next step is for `1d` to figure out how it can get to `1c`
        - This is accomplished by intra-AS routing; `1d` learns that interface `I` is on the least cost path from `1d` to `1c`
            - This allows the internal router to setup a routing table entry that corresponds to destination network `X`, and interface `I`
            - Puts in forwarding table entry (X, I)
- The key idea is that inter-domain and intra-domain routing protocols work together to populate the forwarding table on each router

- Example: Choosing Among Multiple ASes
    - Now, suppose `AS1` learns from the inter-AS protocol that subnet `X` is reachable from `AS3` AND from `AS2`
        - In other words, multiple autonomous systems can be used to reach the destination subnet
            - In this case, both `AS3` and `AS2` can reach the subnet. However, which one should be used?
    - To configure forwarding table, router `1d` must determine towards which gateway it should forward packets for

  destination `X`
- The job of the inter-domain routing protocol is to try to find the minimum number of autonomous systems a packet needs to traverse, but it makes no effort to find the least number of routers a packet needs to traverse
- When a router needs to choose between multiple autonomous systems, a heuristic called hot potato routing is utilized
    - Imagine you are sitting in a room full of people, and someone gives you a super hot potato, who do you pass the super hot potato to?
        - Answer: Pass the hot potato to whoever is nearest to you, and get rid of it as quickly as possible
- Hot potato routing: Sends the packet to the closest router
    - The source-host chooses the (gateway) router that has the least cost path within its own domain/'AS'
        - This information can be calculated using intra-domain routing protocols
            - Each router knows the least cost path within its own 'AS', because they have the full topology, or use distance vector to exchange information
    - The source-host does not care about the internal cost of a packet in another 'AS'
        - It wants to get rid of the packet from its own domain as quickly as possible
            - It is possible that the source-host will send the packet to an 'AS' that will have a higher forwarding cost than another 'AS'
        - In addition, the source-host does not know the internal route a particular packet will follow once the packet enters a different 'AS'
            - Routers in an 'AS' only know about the topology/ structure of their own 'AS'
- To summarize:
    - With inter-domain routing protocols the source-host learns that there are multiple gateways that can be used to forward a packet to an 'AS', and then intra-domain routing is used to figure out which gateway router is the closest or has the least cost path
        - However, the route taken by the packet is not necessarily the least cost path from an end-to-end point of view
- All routers, not just gateway routers, need to run both intra-domain AND inter-domain routing protocols
- i.e. Flowchart of Choosing Between Multiple Autonomous Systems

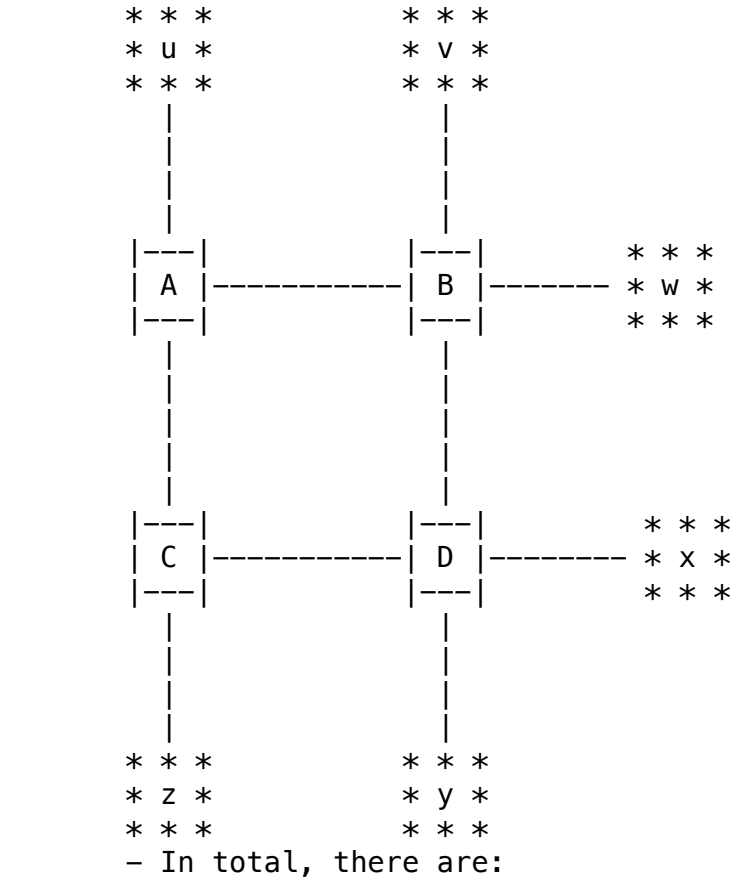




- March 10th, 2021
  - Recap From Last Class
    - Inter-domain and intra-domain routing are used to determine the forwarding table entries on routers
    - The key concept is that the inter-domain routing protocol allows the router to be able to learn reachability to the destination network outside its own 'AS'
      - Typically, this information is conveyed through a pass factor that consists of a collection of autonomous systems
    - From a high level, the connectivity between autonomous systems can be modelled as a graph
      - The vertices correspond to the autonomous systems, and links correspond to the neighbouring relationship between the autonomous systems
        - Then, the least cost path can be computed
        - The least-cost path corresponds to the number of autonomous systems a packet needs to traverse
    - Within an 'AS', the detailed topology of the connectivity of the routers determine the internal reachability
      - Intra-AS routing protocols are utilized to compute the topology/structure of the network
  - Intra AS Routing: IGP
    - The purpose of intra-domain routing is to determine the

internal reachability
  - i.e. How to reach a gateway router for your domain
  - i.e. How to reach a destination network within the same domain
  - Intra-domain routing operates on the assumption that the topology is known
- IGP: "Interior Gateway Protocol" = Intra-domain routing protocol
  - Provides internal reachability
- Most common Intra-AS routing protocols:
  - RIP: Routing Information Protocol
  - OSPF: Open Shortest Path First
  - IGRP: Interior Gateway Routing Protocol
    - This intra-domain routing protocol is proprietary, and was invented by Cisco

- Routing Information Protocol (RIP)
  - Distance vector (DV) algorithm is used in 'RIP'
    - 'DV' is a fully distributed algorithm
    - Neighbouring routers only need to exchange information of their 'DV' to the destination network
      - By exchanging this information, routers can update their internal distance vectors, and populate their forwarding table entry
  - i.e. Figure of RIP



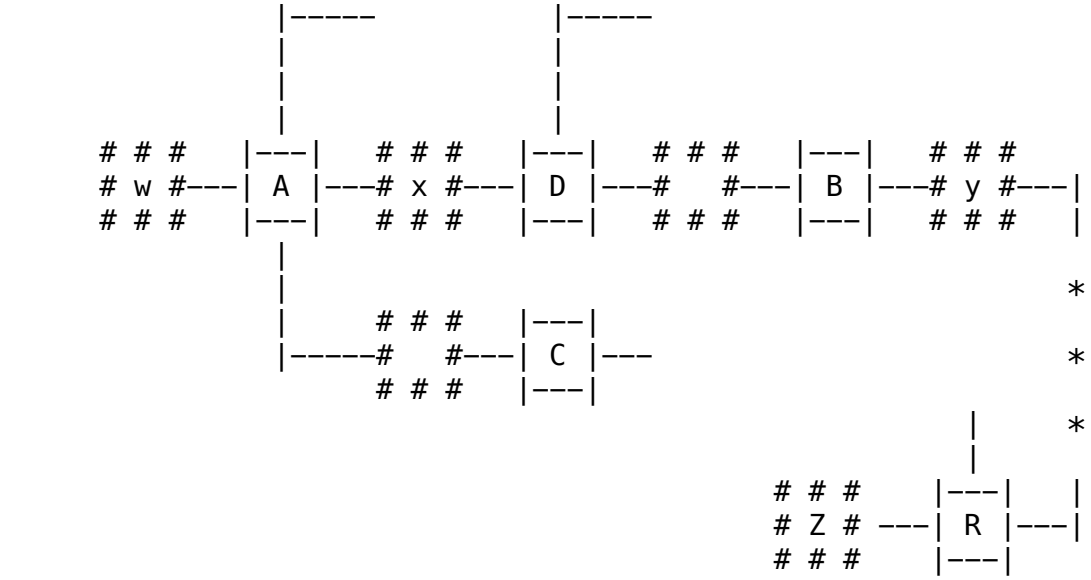
    - In total, there are:

        - 4 routers: `A`, `B`, `C`, and `D`
        - 6 destination networks: `u`, `v`, `w`, `x`, `y`, and `z`
    - Note: A hop is one link on a path between two adjacent routers
    - Each link in this network is associated with a cost of 1
        - i.e. The link cost from router `A` to `B` is 1
        - i.e. The link cost from router `A` to `D` is 2
    - If the distance vector algorithm is executed, then the forwarding table on router `A` will look like:

Destination	Hops
u	1
v	2
w	2
x	3
y	3
z	2

        - This table contains the cost from router `A` to surrounding subnets in the figure
    - Since router `A` is directly connected to `u`, the path cost (or number of hops) is 1
    - In order to reach destination network `x`, from router `A`, the path taken by a packet/datagram is:
      `A` -> `B` -> `D` -> `x`
        - This path takes 3 hops
- 'RIP' is a special implementation of the distance vector algorithm
    - In the 'DV' algorithm we can have a different type of link cost, but 'RIP' utilizes hop count as path cost
- 'RIP' is among the first intra-domain routing protocol that has ever been standardized and implemented
    - Included in BSD-UNIX Distribution in 1982
        - Note: In 1982, 'RIP' ran as a daemon on a computer, and not on specialized routing devices
            - A daemon is a background process that immediately starts with the operating system
    - Utilizes hop counts as the path cost
        - In other words, the distance metric is number of hops, and every link is associated with a unit cost
        - The least cost path is equivalent to the shortest/least hop count path
- In 'RIP' it is assumed that the maximum diameter of the

autonomous system is 15 hops
  - In other words,
- RIP: Advertisements
  - 'RIP' runs based off of information exchanged by routers
    - Neighbouring routers need to exchange `DV` information
      - This is done through 'RIP' advertisement messages
  - 'RIP' advertisement messages are broadcasted among the neighbours of a router every 30 seconds via a 'Response Message'
    - Also called advertisement
    - This message is disseminated locally, only to neighbours, and not sent throughout the entire 'AS'
    - Even if there is no update, this message is sent every 30 seconds
  - Within each advertisement (or Response Message) the router can list up to 25 destination networks within an 'AS'
    - If there are more than 25 destination networks, then you must send multiple advertisement messages
- RIP: Example
  - i.e. Diagram of Distance Vector Algorithm In RIP



    - This example illustrates how the distance vector algorithm will work
      - However, with 'RIP' we only consider the unit link cost, and use hop count as path cost
    - Initially, the forwarding table for router `D` is:

Forwarding/Routing Table In D		
Destination Network	Next Router	Number of Hops To Destination
w	A	2

y	B	2
z	B	7
x	-	1
...	...	...

  - To destination network `w`, the next hop router is `A`, and the number of hops is 2

- Assume that `D` receives the following response message, from `A`, via advertisement:

Advertisement From A To D		
Destination	Next	Hops
w	-	1
x	-	1
z	C	4
...	...	...

  - From `A`, it takes 1 hop count to get to `w`, and there is no next hop
  - It takes 1 hop count to get to `x`, and there is no next hop
  - To get to destination `z` the next hop is `C` and the hop count is 4

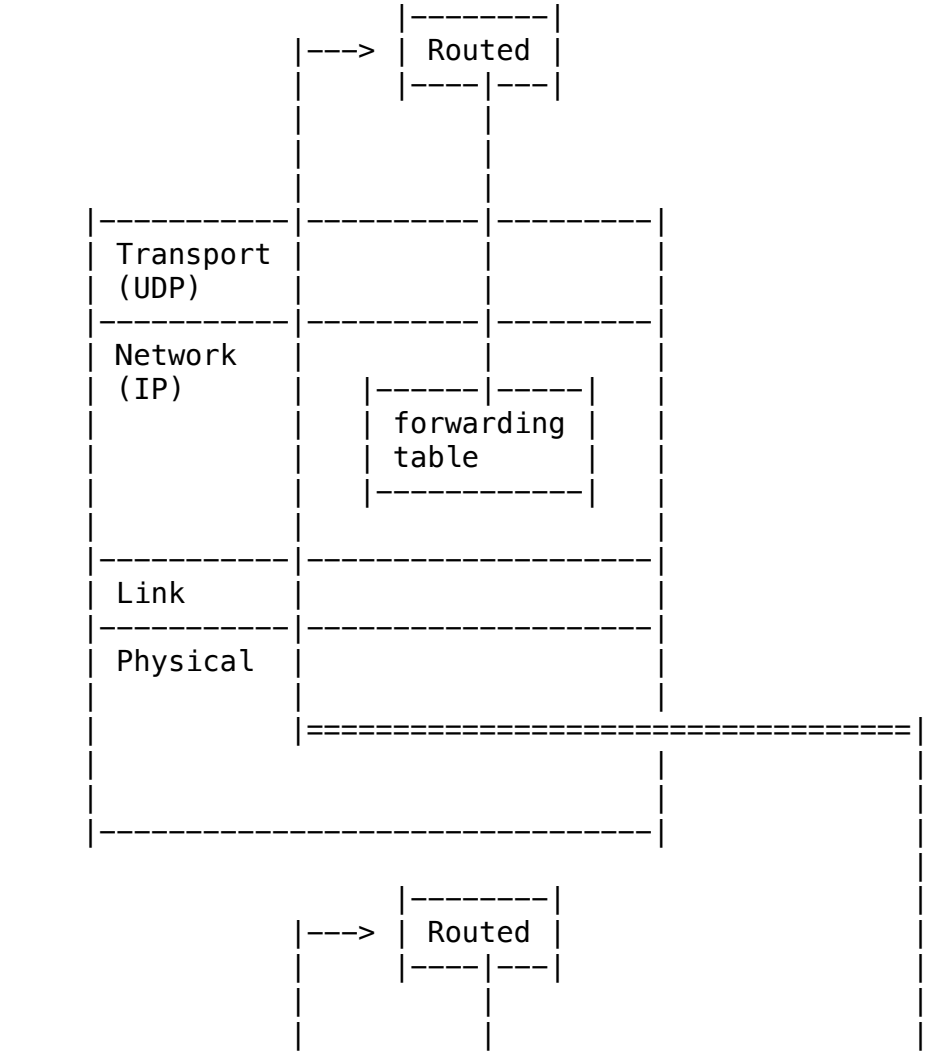
- Upon reception of the advertisement message from router `A`, router `D` will apply the distance vector algorithm and, if need be, update its forwarding table. The updated table is:

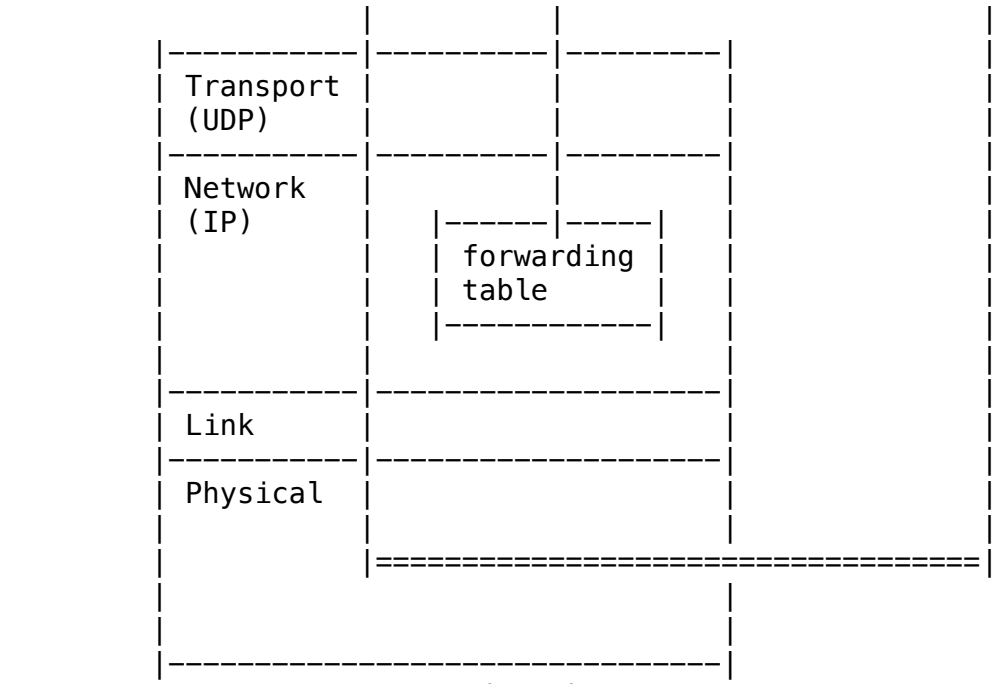
Forwarding/Routing Table In D		
Destination Network	Next Router	Number of Hops To Destination
w	A	2
y	B	2
z	A	5
x	-	1
...	...	...

|-------------|--------|----------------|
  - The distance vector to `w` and `x` does not change
    - According to the Bellman-Ford equation, `A` does not have a shorter path to `w` than in its table
      - i.e. min = {d_A(w) + c(A,D), d_D(w)}
                 = {1 + 1, 2}
                 = 2
    - The path-cost does not improve
  - The distance vector to `y` does not change, because `y` is not in the advertisement message from `A`
  - The ONLY change is the distance vector to `z`
    - i.e. min = {d_A(z) + c(A,D), d_D(z)}
               = {4 + 1, 7}
               = {5, 7}
               = 5
    - If `D` wants to send a message to `z`, then it should route the packet through `A`
- Once `D` updates its routing/forwarding table, it sends the newly updated information to other routers via advertisement messages
  - This allows other routers to update their tables
    - i.e. `B` can update its table for destination network corresponding to `z`

- RIP: Link Failure & Recovery
  - The inter-domain routing protocol needs to take changes in the network's topology into account
    - It needs to be able to recover from disconnectivity caused by link failures, routers going offline, etc.
    - As long as the network is connected, the routing table needs to be updated based on the most up-to-date topology
  - Topological updates are sent via 'heartbeat' messages
    - Recall that 'RIP' messages are sent every 30 seconds
  - If no message is received in 180 seconds, 6 x 30, then the corresponding neighbour can be declared as dead/unreachable
    - Upon detection, the router will first invalidate any routes that use the disconnected neighbour
      - Then, new advertisement messages are propagated to neighbours, and, in turn, the neighbours will update their tables and send out advertisements if their forwarding table has changed
      - This mechanism allows link failure information to be quickly propagated throughout the entire network
  - Note: The advertisement messages sent by routers every 30 seconds should be sent regardless; even if there is no update in the routing/forwarding table. If there is an update, then the information should be immediately propagated to neighbouring routers, so they can update their tables as quickly as possible.
- RIP: Table Processing

- 'RIP' routing tables are managed by an application-level process called 'route-d' (daemon)
  - Note: A daemon is a background process that runs immediately when the computer boots up
  - You can use your computer as a router by running the daemon, 'route-d'
    - Under the hood, 'route-d' executes the distance vector algorithm and exchange information among the neighbours
- Advertisements, aka response messages, are sent via 'UDP' packets, periodically repeated
  - Although 'RIP' is a routing protocol that (supposedly) operates on the network layer, it utilizes the transport layer protocol, via UDP, to send advertisement messages
- Since advertisements are sent every 30 seconds, even if some messages are lost, subsequent messages will be successfully received
  - Thus, TCP does not need to be used to ensure reliability
- i.e. Diagram of Forwarding Tables In Routers





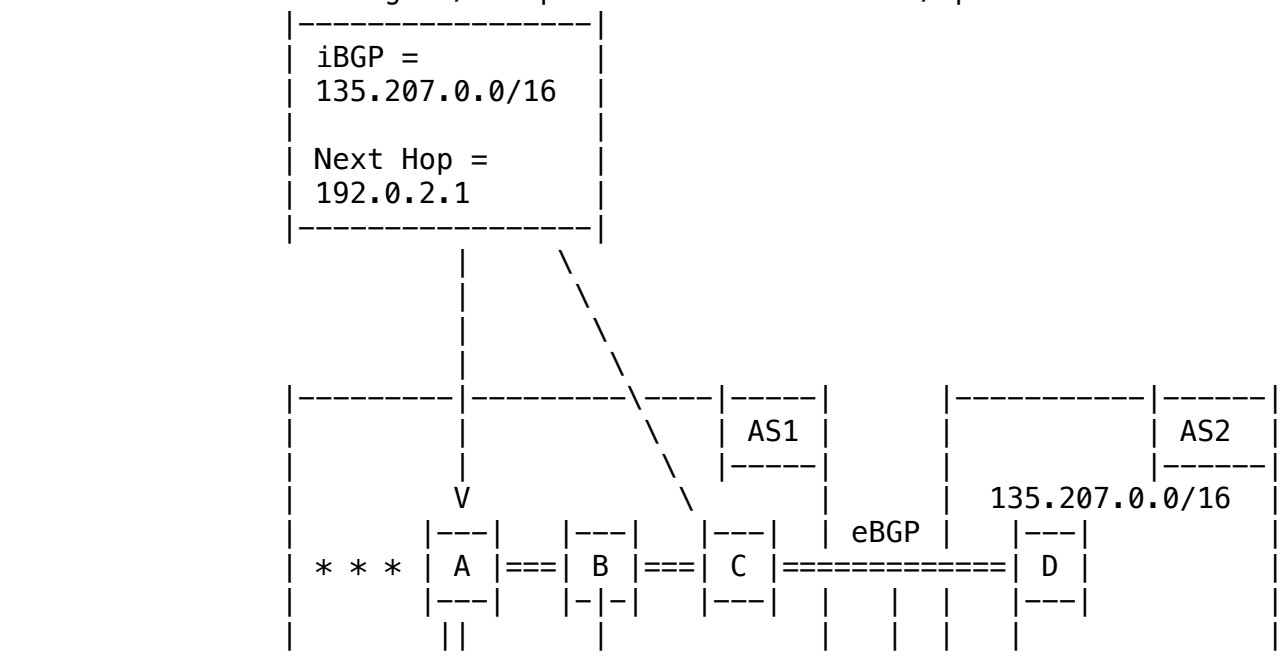
- Open Shortest Path First (OSPF)
  - 'OSPF' is a, relatively, new intra-domain routing protocol
    - Created in the late 80s and early 90s
  - The "Open" part of 'OSPF' means that the reference implementation is publicly available
    - Note: "Open" software means that the source is publicly available for viewing, but it does not imply that anyone can license/use it
  - The key difference between 'OSPF' and 'RIP' is that 'OSPF' is implemented based on link state algorithm
    - Essentially, it needs to disseminate reachability information among routers
      - Then, using this information, each router builds a topology of the entire 'AS', and computes routes via Dijkstra's algorithm
  - Between routers, the 'OSPF' advertisement carries one entry per neighbour router
    - Each advertisement can contain multiple entries
    - The distance vector information is only exchanged between neighbouring routers
      - This is different from 'RIP'
  - Advertisements are disseminated to the entire 'AS'
    - The information about local connectivity is flooded throughout the entire network
    - This is done because every router in the 'AS' needs to learn the topology of the network
      - i.e. Are 2 remote routers, that are not direct neighbours, indirectly connected to each other?
      - The message complexity tends to be higher, because every single message needs to be repeated at least

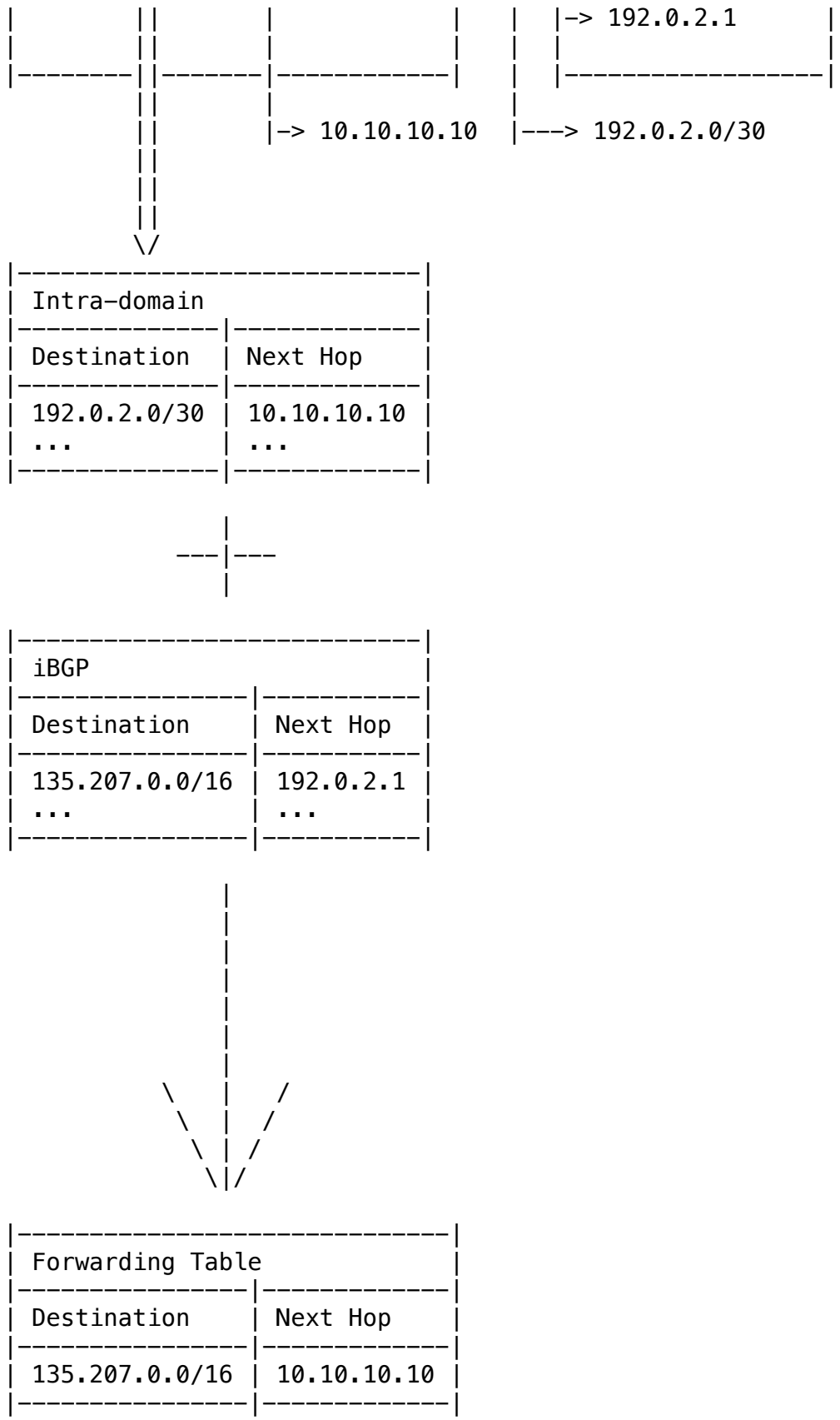
`N` times through the network
    - 'OSPF' messages are sent directly over IP, rather than TCP or UDP
        - This makes 'OSPF' a "clean" routing protocol operating at the network layer
- OSPF Advanced Features (Not In RIP)
    - 'OSPF' is newer, and has more features than 'RIP'
        - i.e. Security
    - Since routing is one of the most important/fundamental functions of the network layer, it needs to be tightly secured
        - If any information is spoofed or falsified, it can (potentially) cause disruptions or disconnectivities in the Internet
    - In 'OSPF' all messages are authenticated to prevent malicious intrusion
    - 'OSPF' supports the ability to accommodate multiple same-cost paths
        - In 'RIP', if there are multiple paths with the same cost, it only retains one of them
            - i.e. If there are 2 paths with the same cost of 3, 'RIP' will only keep one of the paths. On the other hand, 'OSPF' can maintain multiple same cost paths
        - The benefit of maintaining multiple same-cost paths is load balancing
    - In its design, 'OSPF' has a "quality of service"
        - In addition to using hop count as a cost metric, 'OSPF' can also incorporate different cost metrics for different type of services ('TOS'), for each link
        - In IP, there is a type of service field that can be used to indicate what type of service it is
        - Depending on the need, services can differ
            - i.e. Requirements can be:
                - Real time delivery
                - Provide (some kind of) guarantee in end-to-end latency
                - Use best effort services
        - In order to support different quality of service requirement for different flows, 'OSPF' can incorporate different cost metrics
            - i.e. Because of the bandwidth of satellite links, the link cost can be set to low; for best effort service. And, because of the high end-to-end delay the cost can be set to very high; from a real-time perspective
            - This gives more flexibility to determine different type of paths based on the requirement of the end-to-end flow
            - However, this (alone) cannot realize the end-to-end

quality of service for different flows; it needs to be combined with other mechanisms
- i.e.
- What kind of IP packets belong to the same flow?
- What kind of services the flow requires?
- Some kind of signaling mechanism to inform the router of such information
- Upon reception of packets, it needs to be able to figure out which flow the packet belongs to
- Even though 'OSPF' provides this capability, 'TOS', it is not widely utilized
- 'OSPF' has integrated unicast and multicast support
- In unicast, information/data is exchanged between a single source and a single destination
- In some situations, multicast may be desirable
- i.e. For TV channels that run on top of IP, commonly referred to as 'IPTV', it is desirable/ideal to deliver the content to multiple/many customers at the same time
- For this situation, it does not make sense to setup a unicast connection from the source (i.e. TV station) to every customer/subscriber. It is a lot more efficient to setup a multicast tree to disseminate information so that routers do not have to replicate the data for every single customer/subscriber
- Even though multicast is useful, it is not facilitated at the scale of the Internet because of the complexity of setting up the multicast tree, and dealing with the leaf/node dynamics
- However, within an ISP, like Rogers' network, or some kind of cable TV provider, they may implement multicast on their own private network
- Multicast 'OSPF' ('MOSPF') uses same topology database as 'OSPF'
- 'OSPF' can adopt a hierarchical organization for large domains
- This is done to achieve better scalability, because in 'OSPF' you need to flood advertisement messages throughout the entire 'AS'. If the 'AS' is (very) large, then the cost of the message complexity will be (very) high
- A large domain can be partitioned into smaller units, and messages can be flooded within each unit. Each unit can be treated as a vertex, and 'OSPF' can be executed across the unit
- Inter Domain Routing Protocol
- Unlike inter-domain, ISPs can run their own intra-domain

routing protocol, internally
  - i.e. ISP 'A' chooses to run 'OSPF', and ISP 'B' chooses to run 'RIP', and ISP 'C' can run something else
    - It does not matter which intra-domain routing protocol is used, because intra-domain is internal to autonomous systems. As long as there is some kind of routing, routers will be able to learn how to reach internal destination networks
- Inter-domain routing requires interoperability across different autonomous systems
  - Thus, there is only one protocol that is currently implemented in the Interface
    - It is called Border Gateway Protocol (BGP)
- The Border Gateway Protocol (BGP) is the standard for inter-domain routing
  - 'BGP' operates on the network topology that consists of autonomous systems
    - Each 'AS' is abstracted as a vertex, and is associated with different link costs for the links that connect gateway routers to different autonomous systems
    - The simplest way to compute the internal path is to find use something similar to the distance vector algorithm and compute the path with the least hop count to destinations network(s)
  - 'BGP' uses the 'AS' path vector to determine the cost
    - The distance vector within autonomous systems consists of routers internal to the 'AS'. However, the distance vector associated with intra-domain routing contains a collection of autonomous systems, which is the entire path a packet/datagram traverses to reach the destination network
  - 'BGP' includes the entire path; from the first hop to the second hop, to the third hop, and to the last hop
    - This is useful for detecting if there is any potential loop inside the network
      - In normal distance vector algorithms, loops will happen, but eventually they will be resolved
      - Ideally, 'BGP' should converge quickly
    - 'BGP' is an improvement upon the distance vector algorithm, because it includes the entire 'AS' path
    - In contrast, in the 'RIP' protocol, the distance vector only contains: the destination network, the next hop, and the hop count. This is the only information that is exchanged between routers
  - Due to its policy driven nature (or requirement), 'BGP' is a complex protocol
    - Computing least cost paths requires more than just reachability information (i.e. Hop counts). 'BGP' needs the type of agreement that different

autonomous systems, or ISPs, have among each other
  - i.e. Theoretically, it is possible that ISP 'B' does not allow data from ISP 'A' to travel to ISP 'C'
- From a very high level, 'BGP' can be considered to have 2 parts to it:
  - 'eBGP'
    - 'e' = external
    - Operates among the different autonomous systems, between their different gateway routers
  - 'iBGP'
    - 'i' = internal
    - Runs between routers within an 'AS'
    - Different from intra-domain routing, because 'iBGP' propagates reachability information to external networks
      - Note: 'iBGP' is part of the inter-domain routing protocol, NOT intra-domain
- Every single router in the network needs to run both inter-domain and intra-domain routing protocol
  - Inter-domain routing is used for external destinations that are outside the 'AS'
  - Intra-domain routing is used for internal destinations that are inside the 'AS'
- To summarize:
  - Forwarding tables are configured by both intra-domain and inter-domain routing algorithm/protocols
    - Intra-AS sets entries for internal destinations
    - Inter-AS and intra-AS sets entries for external destinations
- i.e. Diagram/Example of How 'BGP' Works/Operates




-> 192.0.2.1
-> 10.10.10.10
---> 192.0.2.0/30
Intra-domain
Destination
Next Hop
192.0.2.0/30
10.10.10.10
...
...
iBGP
Destination
Next Hop
135.207.0.0/16
192.0.2.1
...
...
Forwarding Table
Destination
Next Hop
135.207.0.0/16
10.10.10.10

192.0.2.0/30	10.10.10.10
...	...

- This example analyzes the 'iBGP' portion of 'BGP', and the inter-domain routing protocol that a router can learn how to reach a destination network
- There are 2 autonomous systems: `AS1` and `AS2`
  - `AS1` has 3 routers: `A`, `B`, & `C`
  - `AS2` has only 1 router: `D`
- Router `D` and router `C` are gateway routers
  - The subnet consists of these 2 interfaces
- The IP address of router `B` is: 10.10.10.10
- The IP address of router `D` is: 192.0.2.1
- `192.0.2.0/30` is a link, and corresponds to a very small subnet
- The destination network/subnet is: 135.207.0.0/16
  - This subnet may have 2^16 unique hosts, and they all share the same network address with the top 16-bits of the IP address
- We want to know:
  - How can information about the reachability of the destination network `135.207.0.0/16`, that is currently only known to `AS2`, reach `AS1`?
    - It is possible that the destination network is a customer of `AS2`, or it is a customer of some other 'AS' that has been identified from the 'eBGP', through the border gateway protocol
  - How will routers in `AS1` be able to populate the forwarding table (entry) to include reachability information from the routers in `AS2`
    - If a customer of `AS1` wants to send a packet to the destination network in `AS2`, it needs to know which particular interface the packet needs to be forwarded to
- Assuming that router `A` wants to communicate with a subnet/network in `AS2`:
  1. Through the intra-domain routing protocol, router `A` already knows that in order to reach the destination IP address of `192.0.2.0/30` it needs to forward the packet/datagram to the router with an IP address of `10.10.10.10`
     - This can be done through 'OSPF' or 'RIP'
     - Note: The IP address `192.0.2.0/30` corresponds to the subnet connecting `AS1` and `AS2`
       - It is an internal subnet
     - Note: The IP address `10.10.10.10` corresponds to the next hop router that the packet needs to be forwarded to
  2. The information of reachability of the destination

network is propagated via the 'BGP' protocol. In particular, information between `AS1` and `AS2` is exchanged via `eBGP`. Within `AS1`, reachability information is propagated via `iBGP` to all routers in `AS1`. The 'iBGP' message contains information about the destination network, and the next hop.
    - The next hop corresponds to the interface with an IP address of `192.0.2.1`
3. From the point-of-view of router `A`, the message received in 'step 2' does not indicate what is the next hop. Router `A` needs to know the next hop within its own `AS`, and not the next hop that corresponds to the interface that connects `AS1` and `AS2`
4. The intra-domain routing protocol provides information about how to get to the network `192.0.2.0/30`; the next hop is `10.10.10.10`, which is directly connected to router `A`. Plus, the 'iBGP' message indicates that the next hop corresponds to an IP address of a border gateway router that belongs to the subnet `192.0.2.0/30`. This border gateway router is along the path of the destination address `135.207.0.0/16`, which is outside `AS1`
5. Upon combining the information provided by the intra-domain and 'iBGP' routing protocols, the routers can figure out what is the next hop to forward the packet/datagram to, in order to reach the destination IP address of `135.207.0.0/16`

- The key idea is that 'iBGP' indicates that in order to reach the final destination, the packet needs to be forwarded to the hop/router with an IP address of `192.0.2.0/30`. And, in order to get to this destination network, the next hop is `10.10.10.10`
    - The diagram above shows what the forwarding table that corresponds to this route looks like
- Every router learns how to traverse their 'AS', and what is the next hop to reach a destination network outside of their own 'AS'
- The forwarding table is computed by combining the information given by given by the intra-domain and 'iBGP' routing protocols

- Summary of Difference Between Intra & Inter AS Routing

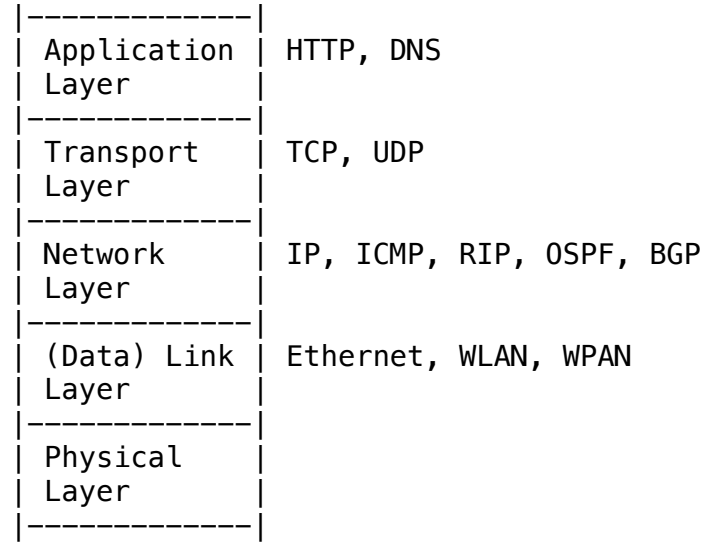
	Policy	Scale
Intra-Domain Routing	* Routing metric	* Internal to autonomous systems

Inter-Domain Routing	* Control over how its traffic is routed * Control over who can route through its 'AS'	* Prefix aggregation * Use 'AS' in path attributes * 'iBGP' to disseminate AS NEXT-HOP

- From a policy point of view, intra-domain routing protocol utilizes routing metrics such as link cost
  - i.e. Associate with certain type of service
  - In the case of 'RIP', it uses hop counts
- Intra-domain routing is simple
  - Run Dijkstra's algorithm or distance vector algorithm on top of the topology, as long as the link cost is associated with corresponding metrics
- Inter-domain routing controls whose traffic can be routed through its 'AS', and how the traffic can be routed
  - The final decision is a combination of network resources that are used to deliver the packet
    - i.e. Link cost, hop count, policy, etc.
      - Policy refers to agreements and payments between the customer and the provider networks along different tiers
- In terms of scalability, inter-domain routing is generally scaleable in the sense that it is only internal to autonomous systems
  - However, for large autonomous systems 'OSPF' can break the 'AS' down into a hierarchy to achieve better scalability
- Scalability on the internet scale for inter-domain routing protocol is difficult to achieve
  - It leverages a collection of mechanisms
- At the very top level, inter-domain routing does not convey information about the specifics of the routers
  - i.e. How are routers interconnected with one another within an 'AS'
    - Routing is done on the basis of autonomous systems
      - In an 'AS' path, there is a collection of autonomous system numbers, and routing is done based on it
- Inter-domain routing heavily utilizes prefix aggregation
  - Since routes consist of multiple autonomous systems, destination networks within the autonomous system are (effectively) aggregated
    - All routes toward an 'AS', regardless of the destination network, are likely to be associated with a single entry
      - This is because all packets/datagrams head to the same direction. As a result, all prefixes associated with different destination networks within the same 'AS' can be aggregated

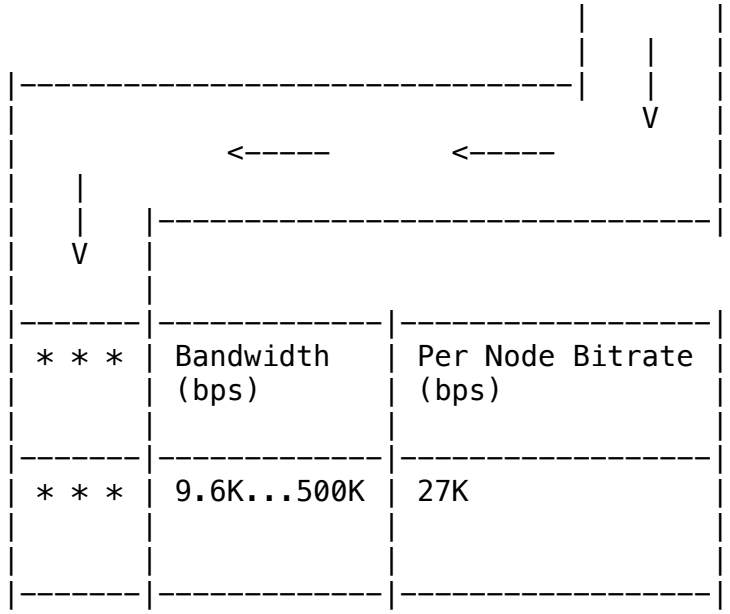
        - Inter-domain routing is divided into:
            - 'eBGP'
                - Responsible for exchanging information among the gateway routers at the border of autonomous systems
            - 'iBGP'
                - Used to disseminate the 'AS' and next hop information within other autonomous systems
                    - This is kind of like a hierarchy, and allows better scalability for the inter-domain routing protocol
        - 'BGP' is a complex protocol
            - There is active research of 'BGP' to study its stability
        - The main takeaway is to know the difference between the intra-domain and inter-domain routing protocol, and what are the key functions that they accomplish

- March 12th, 2021
    - Data Link Layer
        - This is the start of a brand new chapter
    - Data Link Layer & LANs
        - The top 3 layers of the TCP/IP protocol stack are:
            - Application Layer
                - i.e. HTTP, DNS, etc.
            - Transport Layer
                - i.e. TCP, UDP, etc.
            - Network Layer
                - i.e. IP, ICMP, RIP, OSPF, BGP, etc.
        - This lecture will focus on:
            - Data link layer services (at a high level)
            - Two, very commonly used, data link layer technologies:
                - Ethernet & Ethernet bridging
                - Wireless WLANs
                    - WLAN = Wireless Local Area Networks
        - i.e. Diagram of Network Layers



- High Diversity
  - Compared to other layers, except for the application layer, the data link layer has the most diversity
    - The application layer has the most number of unique protocols
  - There are all kinds of technologies to connect multiple devices together
    - Refer to table below
  - i.e. Table of Some Data Link Layer Protocols

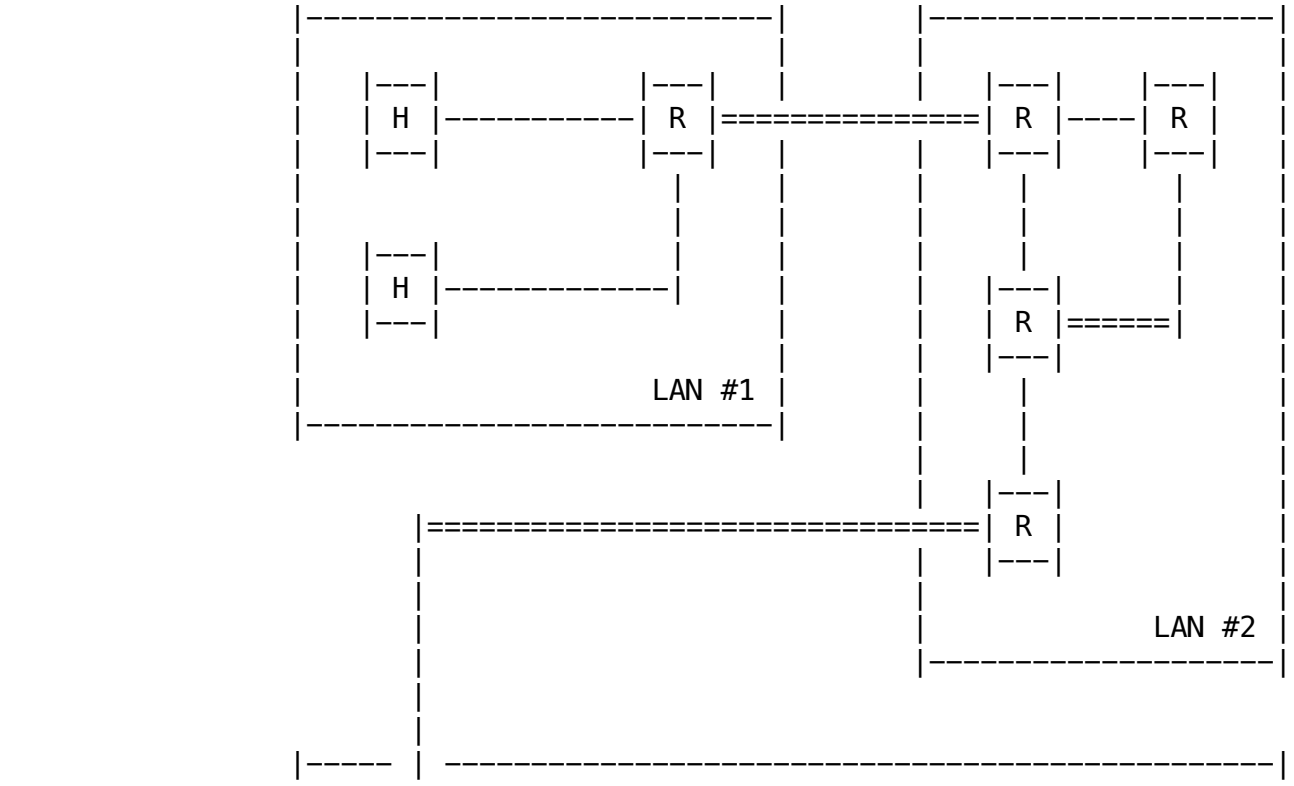
Protocol	Topology	# Of Nodes	Segment Length (m)	* * *
Modbus	BUS Master/ Slave	247	400...1200	* * *
PROFIBUS	BUS Master/ Slave	31 Per Segment	100...1200	* * *
CAN Bus	BUS CSMA	100	40...1000	* * *
Ethernet	STAR	N/A	100	* * *
USB Hub Network	BUS Master/ Slave	127	5 (Between Hubs)	* * *

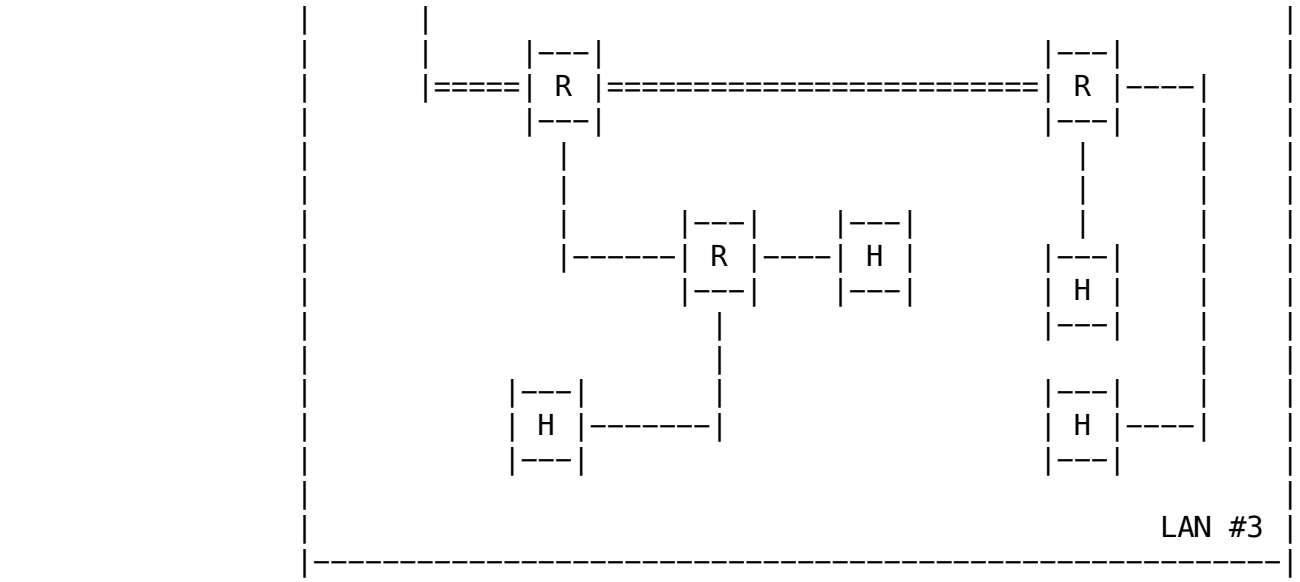


* * *	9.6K...12M	750K
* * *	50K...1M	60K
* * *	100M	6M
* * *	2M	100K

    - These data link layers are used in industrial settings
        - i.e. Automotive settings, local enterprise networks, etc.
    - The support provided by the data link layer is highly diverse
        - It can support a wide range of topologies, segment lengths, bandwidth, number of nodes, etc.
    - Not all of these protocols are part of TCP/IP
        - Many of them cannot interface with IP
- Fast Evolution
    - Another important aspect to the data link layer is its fast evolving nature/attribute
    - Typically, the data link layer is based on the underlying physical layer technology
        - As the physical layer technology advances, the data link layer will experience changes as well
    - The following link is a table that shows different Wi-Fi standards being introduced over a number of years: https://en.wikipedia.org/wiki/IEEE_802.11#Protocol
    - In 1997, 802.11b was ratified
        - During this time, Wi-Fi networks operated at the frequency of 2.4 Ghz, and could only carry a data rate of 1-2 Mbit/s (megabits per second)
    - In 2019, 20 years after the ratification of 802.11b, the advanced network PHY standard `HE-OFDM` was introduced
        - This is the Wi-Fi protocol 802.11ax
        - It has a max data rate of 10.53 Gbit/s (gigabits per second)
    - The Wi-Fi protocol 802.11ax can achieve a speed that is ten-thousand times greater than 802.11b
    - Physical link technologies are continuing to evolve
        - Recent versions of 802.11 are becoming commercialized, and available in the market
            - You can purchase and use them at home
- Link Layer: Introduction
    - The link layer is the communication channel that connects

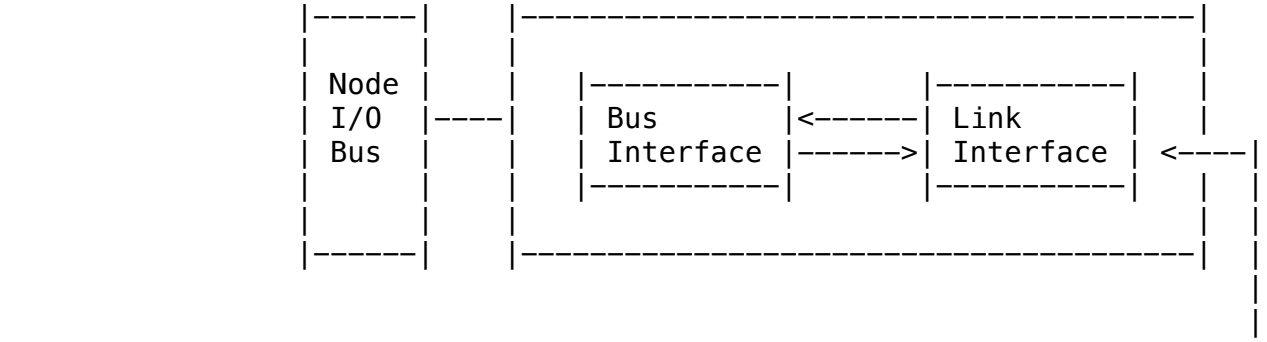
  neighbouring/adjacent nodes along a communication path
    - Connecting can be done via:
        - Wired links
        - Wireless links
- The link layer sits on top of the physical layer
    - The physical layer can be wired or wireless
    - It serves as the intermediary between the network layer and the physical layer
- The key responsiblity of the data link layer is to transfer frames from one node to another over a link
    - The (data) link layer is primarily concerned about the connectivity within the local area network between neighbouring nodes that have direct physical connections with one another
    - In contrast, the function of the network layer is to transfer datagrams from one end-host to another
        - The hosts do not (necessarily) need to be on the same local area network, nor do they need to have direct connectivity
        - Often times, the transmission of data from one host to another has to go through multiple routers
- Typically, a (link) layer-2-packet is called a frame
    - It encapsulates datagrams coming from the IP layer
- To summarize:
    - Data link layer has responsibility of transferring frames from one node to adjacent node over a link
- i.e. Diagram of a Networks Connected To Each Other

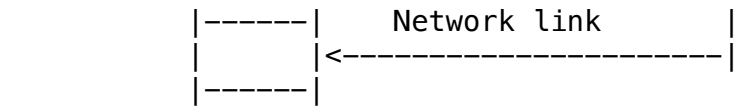




- The '===' represents a link between routers/devices

- Network Interface Card & Driver
  - From a hardware point-of-view, the network interface card (NIC) is a combination of link layer and physical layer
  - Modern network interface cards are integrated into the motherboards of desktop computers or laptops
  - Network interface cards are necessary pieces of hardware that facilitate communication
    - These cards can support wired or wireless connectivity
  - Typically, the functionality of network interface cards can be divided into what is implemented in the firmware, and the hardware
    - The firmware is a program stored in the network card's ROM (BIOS)
      - Configuration information is stored in E2PROM; allowing it to be modified
    - The hardware portion includes the chips and some connectors
      - This makes transmission and reception of bits possible through the network interface card
  - Drivers are software that provide an interface between the network card hardware/firmware and the operating system
  - i.e. Diagram of Physical Part of Network Interface Card





    - The adapter is a semi-autonomous unit
    - The physical part of a network interface card (NIC) will include some kind of interface
    - On the side of the operating system, there is some port that connects to the network interface card, and interface with the operating system through the device driver
- Link Layer Services (1)
  - Flow control
    - Paces the transmission of data from sender to receiver to ensure that the receiving node is not overwhelmed
  - Error detection
    - When packets are in transit, they may experience corruption due to:
      - Noise on the transmission medium
      - Signal attentuation
      - Interference from other devices/transmissions
      - Hardware failures
    - It is important for the receiver to be able to detect errors
      - Upon detection of an error, the receiver can signal the sender for transmission or drop the frame
    - Methods for detecting errors can be:
      - One dimensional parity checking
      - Two dimensional parity checking
      - Checksum
    - Typically, the link layer adopts cyclical redundancy code (CRC) for error detection
      - When compared to checksum, CRC is more efficient
      - Even though CRC can detect errors, it cannot be used for error correction
    - Error detection may be implemented by the link layer
  - Error correction
    - Some link layers may support the ability of error correction
      - Not all link layers can perform error correction
    - After receiving the frame, the receiver can utilize redundant bits included in the frame to correct bit error(s), instead of resorting to retransmission
  - Question: Which other layer provides similar services?
    - The transport layer, specifically TCP, implements the functionality of flow control and error detection. However, it does not perform error correction, but it does perform reliable data transfer through mechanisms such as package retransmission
      - The flow control mechanism in the link layer performs different services than the one in TCP

        - The job of flow control in the link layer is to avoid overflowing the receiver that is directly connected to the sending host. But, in TCP, flow control prevents overflowing the receiving host that is at the other end of the connection, multiple hops away
    - Error detection is implemented at the link layer because it gives the end-to-end TCP connection an illusion that the link being used is reliable. Packet losses and corruptions can be hidden from upper layer protocols. This is useful for links that are lossy and prone to error, such as Wiress links like Wi-Fi and cellular. Wireless links have a bit error rate of 10^-6; for every million bits you send, 1 bit is likely to be lost. Doing error detection/correction at the last hop or wireless link, makes it appear to be just as good as a wired link. From TCP's point-of-view, everything looks great, and retransmission is not required.
        - The overhead for detecting errors is cheap, and much better than retransmission; which is very expensive. In fact, Wi-Fi does retransmission if it does not get an acknowledgement from the receiver, after trying multiple times to deliver the frame. Timeouts (i.e. TCP timeout) are very costly. It is ideal to fix the problem locally.
            - Thus, some functionalities are duplicated on 2 network layers
- Congestion control is not a feature of the data link layer because the link layer is only concerned with the local area network (LAN), or directly connected hosts, while congestion control is (typically) concerned about not overflowing the network
    - In addition, as long as the layer performs flow control the LAN will not be congested
        - Thus, congestion control is not necessary
- Note: Some services provided by the link layer are offered by other layers/protocols, while other services are unique to the link layer

- Link Layer Services (2)
    - Half-duplex and full-duplex
        - These are tied to the particular physical layer technology that is utilized
        - With half-duplex, nodes at both ends of the link can transmit, but not at the same time
            - In order for one node to transmit, the other node has to be silent
            - The nodes have to take turns in transmitting data
            - Most wireless technologies implement half-duplex
                - Two ends cannot transmit at the same time

        - i.e. Example of half-duplex device: Walkie-Talkie
            - The speaker must say "over" when he stops talking
            - If both people talk at the same time, they will not be able to hear each other
    - Ethernet switches are full-duplex devices
    - The issue with half-duplex devices is figuring out who gets to talk first
        - Determining the order of transmission is unique to the link layer
- Framing
    - From the link layer's perspective, the data that arrives from the physical layer is a continuous bitstream
        - Over the transmission medium the data is analog, but after decoding, by the physical network interface, the analog waveform becomes a continuous bit stream
        - By looking at a bit stream, it is not possible to tell where a particular link layer frame/packet starts or ends
            - Imagine you have some data that communicates over the air or over a wire. How can you tell if something useful is being transmitted? It is possible that noise in the transmitted medium can be mistakenly interpreted as data
                - i.e. The sender is not transmitting anything, but noise is being picked up as real (sender's) data
        - Regardless of what kind of physical layer is used, the link layer needs to be able to partition bit streams into frames
            - To accomplish this, additional information called preamble is added to distinguish the beginning and end of incoming data
                - The preamble is added in addition to the regular header fields, and on top of the IP header
    - Framing is a way to breakdown a continuous bit stream into its corresponding frames
        - This is accomplished by adding some header or trailer bits before or after, respectively, to the actual data
    - Framing is essential to link layer services
- Link access
    - Is very important if the connected devices share a common medium
        - i.e. In the case of wireless transmission, all neighbouring devices within the same physical proximity share the air; the air around them is the transmission medium
        - i.e. Multiple devices connected to the same bus must

handle the issue of sharing the medium
        - If the medium is shared, how is channel access distributed among all devices?
            - You cannot have every single device talking through the transmission medium at the same time
    - The services above are not available in the transport layer
        - All of these services are unique to the link layer
- Channel/Medium Access
    - An important part of the link layer protocol is: "How do you share the medium access across different devices?"
    - There are 2 types of "links":
        1. Point-to-point
            - Also called dedicated pairwise communication
            - Consists of 2 communicating entities at both sides of a wire
                - The wire is dedicated to this pair
            - If the link is half-duplex, which one of the two hosts gets to transmit over the point-to-point link must be determined
                - Other nodes that are contending the same transmission medium, with the pair, do not pose an access issue
            - i.e.
                - Dial-up access
                    - With a phone line, you need to dial up through a modem, to get a connection
                        - i.e. AOL
                - Point-to-point link beween host and switch
                - Switched Ethernet
                    - Connecting a device to a port on an Ethernet switch creates a point-to-point connection between the port and the device
                        - Other devices connected to the same Ethernet switch will be (somewhat) isolated by different ports
            - Advantage: The medium is shared with others
            - Disadvantage: Not resource efficient
        2. Broadcast (shared wire or medium)
            - Multiple devices share a wire, like a bus, or share a wireless medium
                - Devices contend the medium, and must decide who gets to transmit
            - i.e.
                - Traditional Ethernet
                    - Includes devices that can be connected through an Ethernet bus, instead of switches
                    - Need to detect collision
                - 802.11 wireless LAN
                    - If all devices on a wireless local area

network (WLAN) transmit a lot of traffic at the exact same time, then congestion on the WLAN may occur
- Occurs because there is no dedicated link between the devices and Wi-Fi access point. Thus, devices must take turns transmitting data

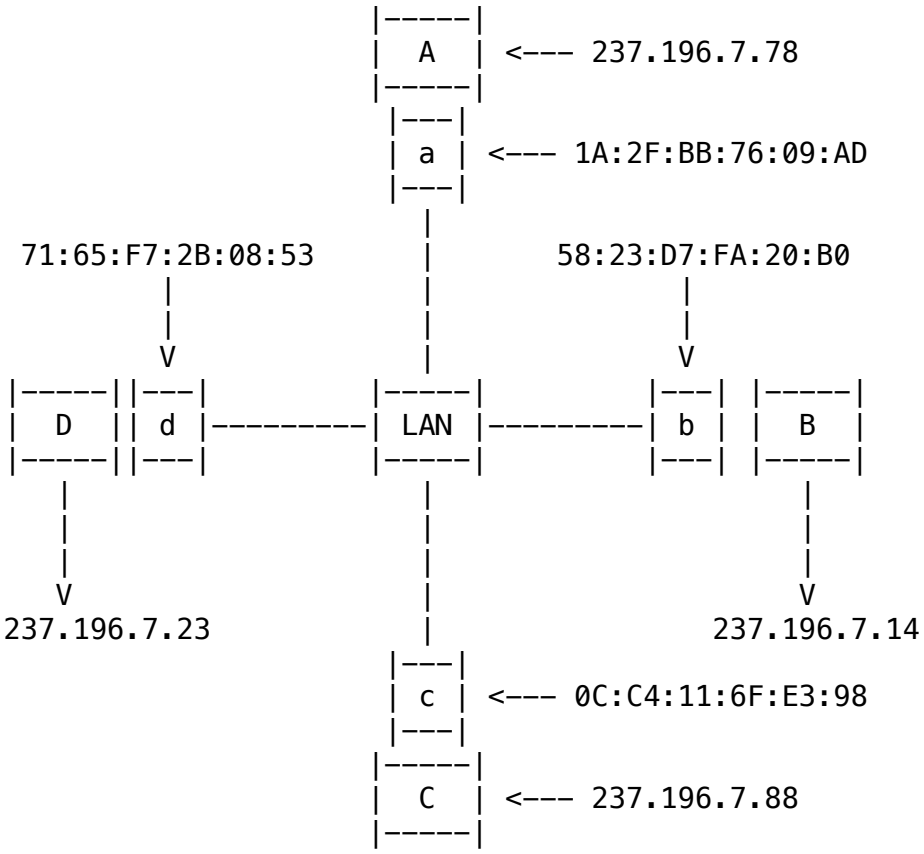
- MAC Addresses
  - Recall that IP addresses:
    - Are network layer addresses which are globally unique, or unique in the local area network (LAN)
    - Can be hard wired, manually configured, or dynamically learned through the DHCP protocol
    - Used to get datagrams/packets to the destination IP subnet
      - On routers, the next hop is determined by performing a lookup in the forwarding table using longest prefix match
    - IP addresses are, typically, 32 bits in size
      - However, IPV6 addresses are 128 bits in size
  - The equivalent of IP addresses in a LAN setting or at the link layer is the MAC address
  - MAC addresses are used to get datagrams from one interface to another physically connected interface, on the same network
    - Also, they are used to distinguish one interface from another
  - Typically, a MAC address is 48 bits in size
    - The 48 bits are broken into 6 bytes, and a dash line is used to separate each byte/segment
      - The format is hexadecimal (HEX)
        - i.e. 1A-2F-BB-76-09-AD
  - MAC addresses are unique and hard-coded in the ROM of the nework interface card when it is manufactured
  - The Broadcast address is a uniue MAC address that sends the frame to all adapters
    - It is composed of all 1's in HEX format
      - i.e. FF-FF-FF-FF-FF-FF
- MAC Addresses (More)
  - Are administered by IEEE, and allocated based on the vendor's criteria/specification
    - Typically, device vendors buy a portion of MAC addresses and utilize them for their own network interface cards
      - This is done to assure uniqueness
      - Often times, the first few bytes of a MAC address can be used to tell which vendor it is
      - Sniffing software can tell that a device is a Cisco Wi-Fi access point, or an Apple Wi-Fi card
        - Even though this information is not broadcasted in the frames transmitted by the interface card,

the sniffing software infers this information from the first few bytes of the MAC address associated with the network interface card (NIC)

- MAC addresses are entirely flat
  - Even though the first few bytes might correspond to the vendor, no other bits/bytes contains any significance
    - In other words, there is no hierarchical structure of MAC address, or class based MAC addresses
    - This enables devices to be portable
      - i.e. Devices can be moved from one LAN to another
  - In a local area network (LAN), there can be a very diverse collection of MAC addresses
    - This depends on the type, and manufacturer, of the device
  - On the other hand, IP addresses do have some kind of hierarchical allocation
    - An IP address has 2 portions:
      1. The top number of bits correspond to the network address portion
      2. The lower bits correspond to the host portion
    - IP addresses are not portable
      - The address depends on the subnet that a device is connected too
- Typically, MAC addresses are hard-coded on the (device's) local network interface card (NIC)
  - If the device is taken to a different network, it will utilize the same MAC address
    - On the contrary, if a device is moved from one network to another, the IP address will be different; due to DHCP
      - This is good practice because it provides the ability to aggregate network addresses for better scalability; for Internet browsing
- Note: MAC addresses are utilized between devices that are directly connected. They are not used between end-hosts that may not be directly reachable, and require packets to be routed through multiple routers.
- Note: MAC addresses are utilized to address devices that are directly connected through a LAN, like Ethernet or Wi-Fi.
  - In some cases, MAC addresses are used for point-to-point connections
    - It is used to address the device on the other end of the dedicated pairwise link
- Question: If we have IP addresses, then what is the point of MAC addresses?
  - Answer: There are 2 reasons why both are needed
    1. IP addresses are not permanent. They are not permanently associated with a network interface

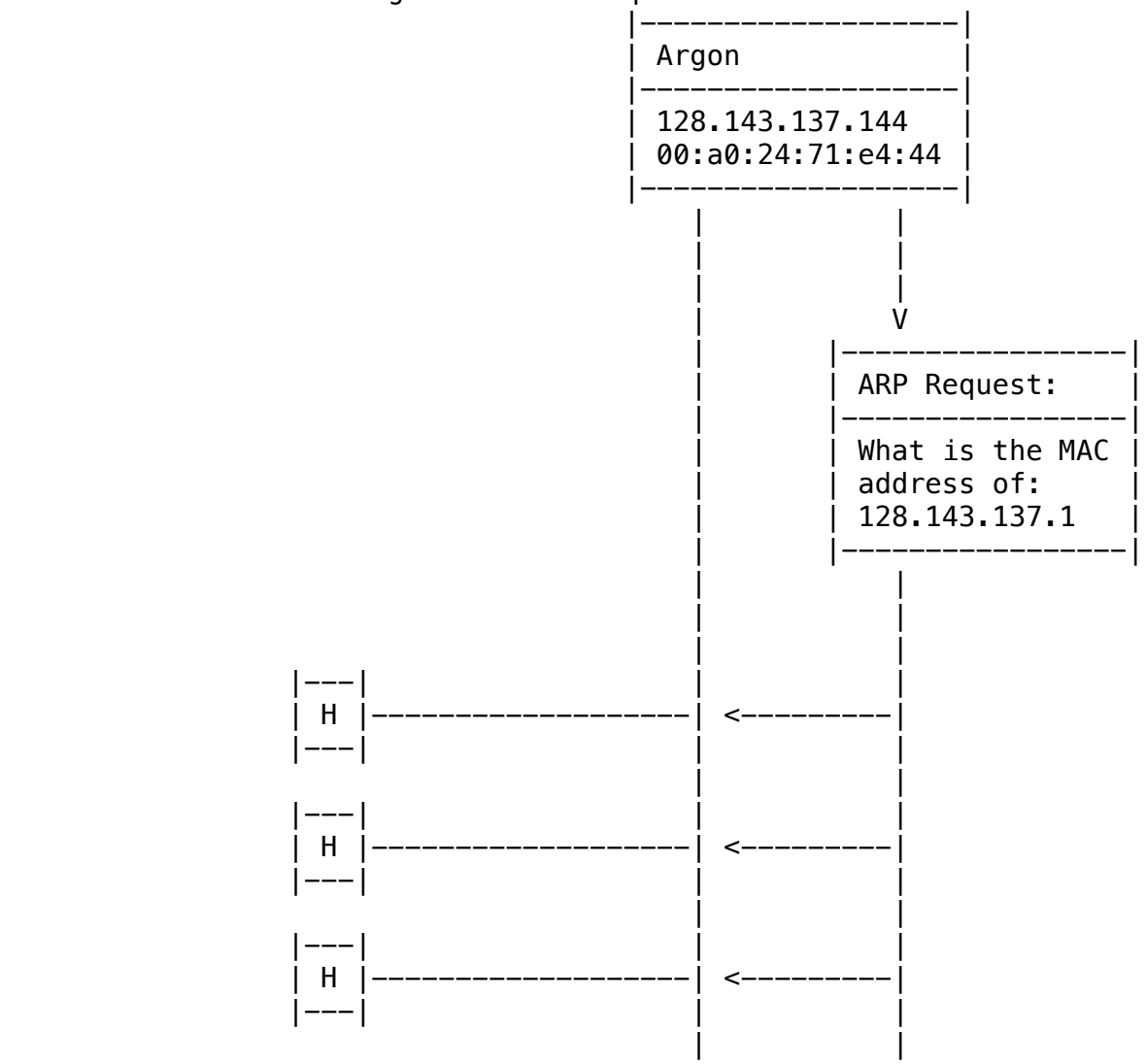
        card, like MAC addresses. Moving from one network to another will change the IP address, but the MAC address will not change because it is innately tied to the network interface card (NIC)
    2. From the view of the TCP/IP protocol stack, it is possible to have protocols that do not work under IP. Protocols that are not under the TCP/IP umbrella can support direct connectivity with the link layer that has its own MAC address
        - In essence, when compared to IP, MAC addresses are broader, and are not limited to the TCP/IP protocol suite
- To summarize:
    - The allocation of MAC address are administered by IEEE
    - Manufacturers buy a portion of MAC address space
        - This ensures uniqueness
    - MAC addresses are flat
        - This allows for portability
            - Can move NIC from one LAN to another
- IP hierarchical address is NOT portable
    - It depends on IP subnet to which node is attached
- ARP: Address Resolution Protocol
    - There are 2 (main) addresses:
        1. IP Address
            - Can change depending on which network a particular device is connected to
        2. MAC Address
            - Hard coded on the network interface
    - Question: How do we translate between IP address and MAC address?
        - Answer: Via the 'ARP' table
    - Address Resolution Protocol (ARP) works by storing and maintaining information for each IP node, on a local area network, in an 'ARP' table
        - An IP node can be a host, or a router
    - The 'ARP' table stores mappings between the IP Address, MAC Address, and Time To Live (TTL)
        - The 'TTL' field indicates how long the association/ mapping is valid for
            - Upon expiration of the 'TTL', the entry is deleted
            - The typical value of 'TTL' is 20 minutes
                - This is why 'ARP' entries are prevalent in Wireshark. Even if nothing has changed in the LAN, and the collection of devices is the same, along with their information
        - When an entry in the 'ARP' table expires, an additional message is generated and exchanged with the corresponding host to re-establish the mapping between IP address and MAC address
            - Upon renewal, the 'TTL' field is reset to its

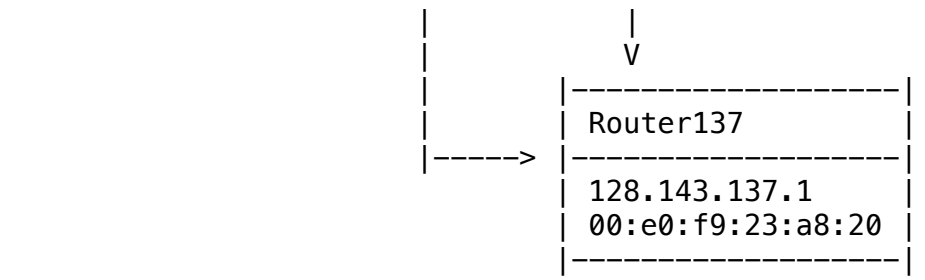
default value
- Summary of 'ARP':
  - Each IP node (Host/Router) on LAN has an ARP table
  - ARP Table: IP/MAC address mappings for some LAN nodes
    <IP address; MAC address; TTL>
  - TTL (time to live): Time after which address mapping
    will be forgotten (typically 20 min)
- i.e. Diagram of a LAN



  - 'A', 'B', 'C', and 'D' correspond to hosts on a LAN
    - 'a', 'b', 'c', and 'd' are the MAC addresses for
      their respective devices/hosts
  - Question: How to determine B's MAC address, only knowing
    its IP address?
    - Answer: The ARP table
      - The process of translating between IP address
        and MAC address functions in a similar manner to
        'DNS'. In DNS, from the user's point-of-view,
        only the domain names of web servers are known;
        the IP address is unknown. DNS services are used
        to assist the user in translating a domain name
        into its corresponding domain IP address.
        Conversely, the address resolution protocol
        (ARP) serves as a translator, translating
        between IP address and MAC address at the local
        area network (LAN)

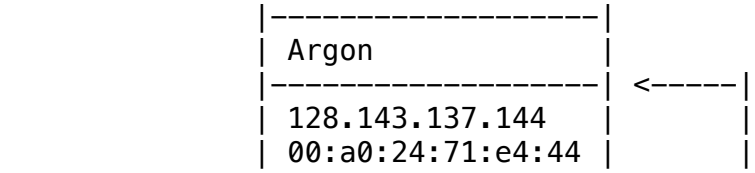
        - The address resolution protocol (ARP) is required to send link layer frames to another device in the same local area network. The end-host is addressed via its MAC address, and not its IP address
            - Association between MAC and IP address can be dynamic, because MAC addresses cannot be (easily) changed, while IP addresses can and do change over time
- ARP Request
    - The most basic way to learn the mapping between IP address and MAC address is through 'ARP' request and 'ARP' reply
    - In the example below:
        - Argon learns the IP address of 'Router137' through DHCP
        - Argon broadcasts an 'ARP' request to all stations on the network, such as: "What is the MAC address of Router137"
    - i.e. Diagram of ARP Request

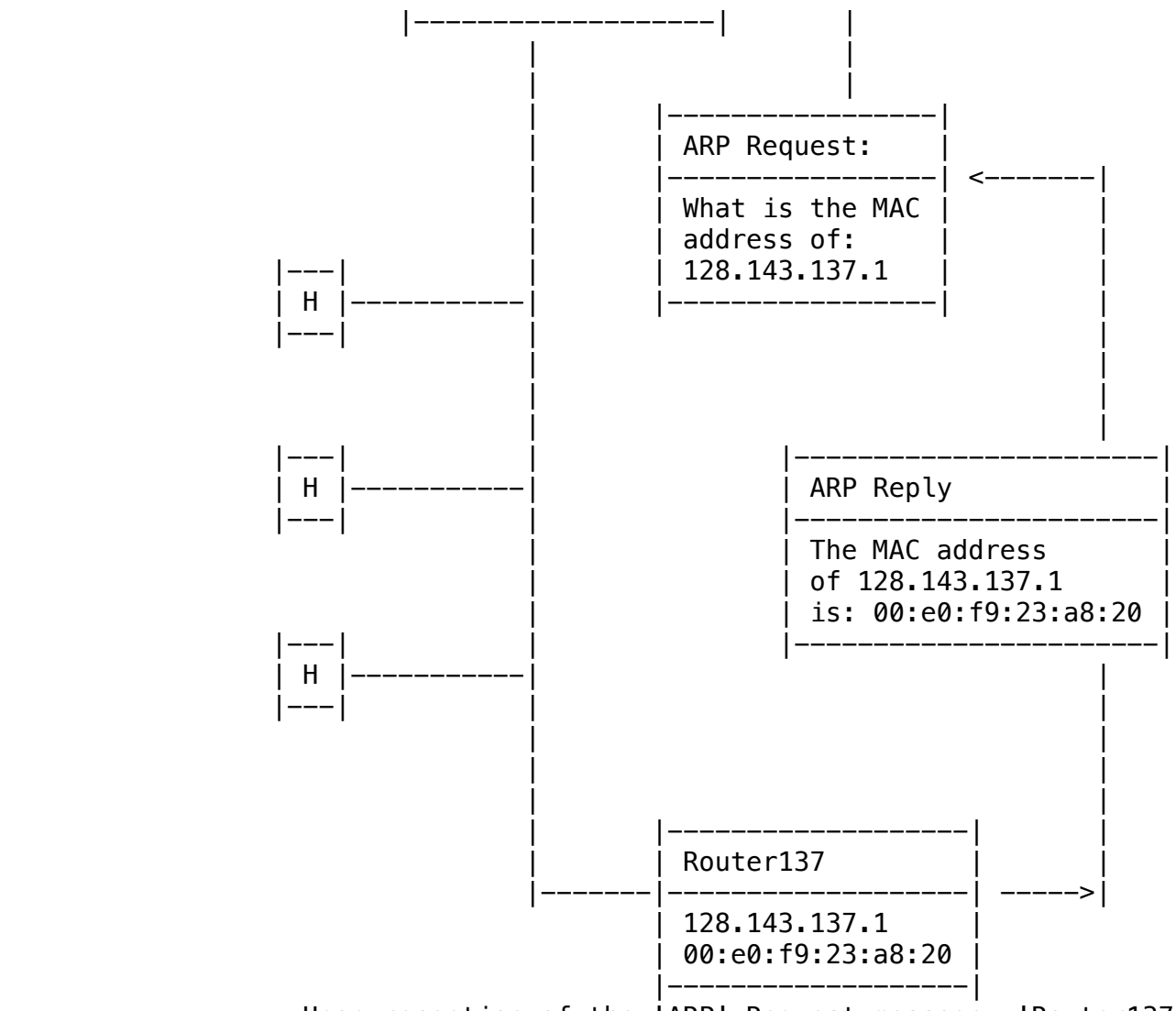




- 'Argon' is a network host with an IP address of `128.143.137.144`, and the MAC address for its ethernet/network interface is `00:a0:24:71:e4:44`
- 'Router137' is a router/device, on the network, with an IP address of `128.143.137.1`, and a MAC address of `00:e0:f9:23:a8:20`
- When the host, 'Argon', is first connected to the local area network, it relies on DHCP to get an IP address
  - Through DHCP, the host, 'Argon', will learn about the first hop router's IP address and other information such as DNS server
    - Put simply, 'Argon' acquires the IP address of 'Router137' through DHCP
- Assume that 'Argon' wants to send something, or connect, to a website (i.e. Google.com)
  - It needs more information than just the first hop router's IP address, which is `128.143.137.1`
    - Note: The first hop router is 'Router137'
  - The information that 'Argon' currently has is not sufficient to be able to put together a data link frame, and send an initial message
    - An initial message could be the initial SYN in the 3-Way TCP handshake
  - The first bit of information that 'Argon' needs is the MAC address of the router, 'Router137'
    - This is accomplished through 'ARP' request, where 'Argon' will broadcast the message: "What is the MAC Address of Router137"
      - 'Router137' is replaced by its own IP address; 'Argon' learned it through DHCP
      - The message is broadcasted throughout the entire local area network (LAN)

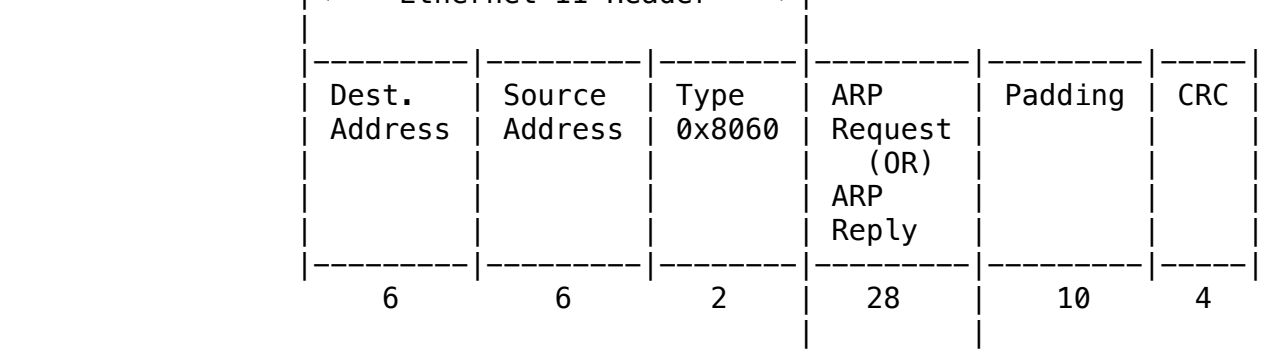
- ARP Reply
  - In the example below:
    - Router 137 responds with an ARP Reply which contains its MAC address
  - i.e. Diagram of ARP Reply

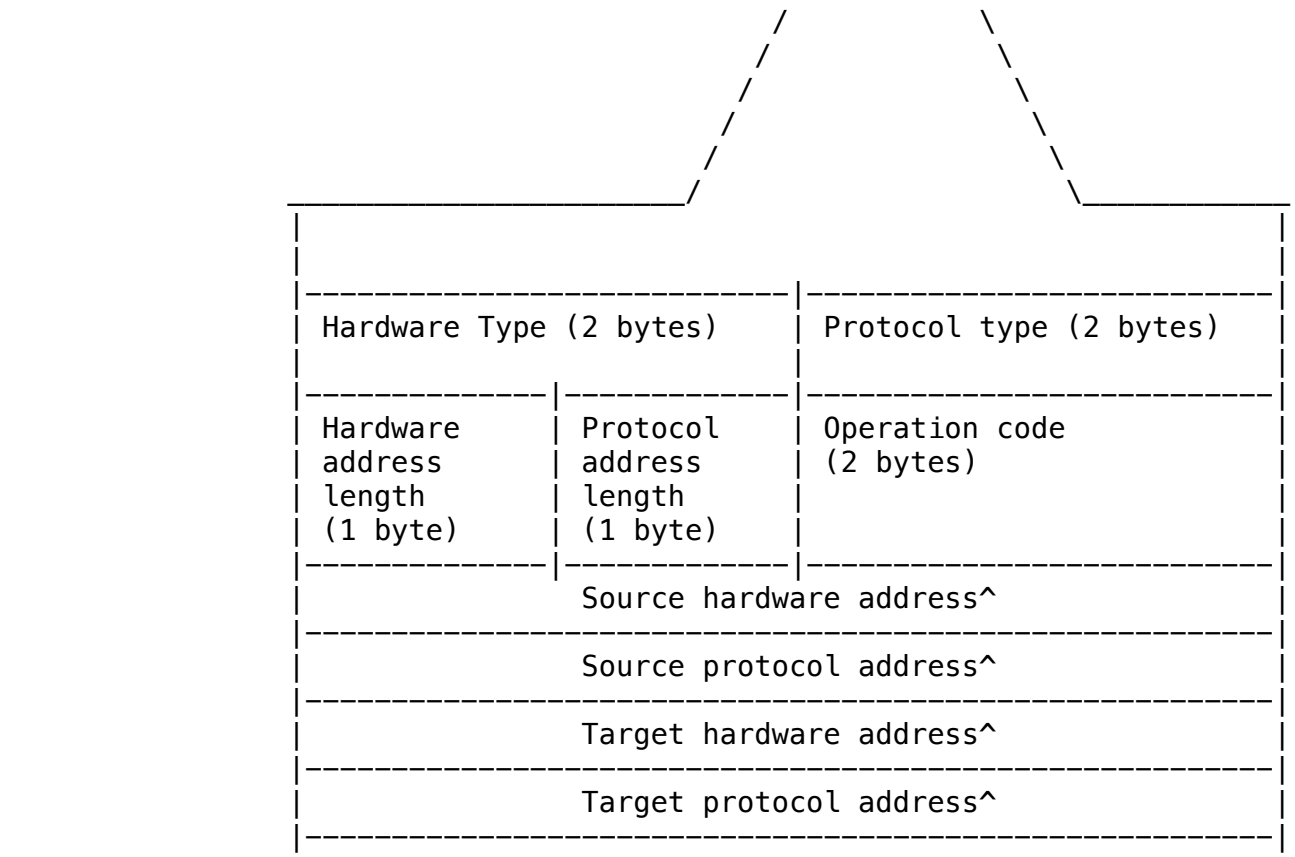




- Upon reception of the 'ARP' Request message, 'Router137' will generate an 'ARP' reply
  - The reply message will contain information about its own MAC address that is associated with the IP address in the previous 'ARP' request message

- ARP Packet Format
  - i.e. Format of an ARP Packet Format





- The 'ARP' Request/Reply is the most important field
  - This field includes (some kind of):
    - Hardware type
    - Protocol type
    - Hardware address length
    - Protocol address length
    - Operational code
- The most important parts in the 'ARP' Request/Reply message are the 4 addresses
  - i.e.
    - Source hardware address
    - Source protocol address
    - Target hardware address
    - Target protocol address
  - Each address is 48-bits long
  - Depending on what message is being referred to, request VS. reply, the 4 addresses will be different
- 'ARP' can work with other network layer protocols
  - This is why the addresses in the request/reply message use the world 'protocol' and not 'IP'
  - Note: For this class only IP protocol is discussed
- ^Note: The length of the address fields is determined by the corresponding address length fields

- Example
  - ARP Request from Argon:

        Source hardware address: 00:a0:24:71:e4:44
            - This is the MAC address of 'Argon'
        Source protocol address: 128.14.137.144
            - This is the IP address of 'Argon'
        Target hardware address: 00:00:00:00:00:00
            - Initially, 'Argon' does not know what the target MAC address is; hence the reason for the 'ARP' request message. Thus, 'Argon' sets the target hardware address field to be all 0's
        Target protocol address: 128.143.137.1
            - This is the IP address of the router; 'Argon' (previously) learned this address through DHCP
    - ARP Reply from Router137:
        - Source hardware address: 00:e0:f9:23:a8:20
            - MAC address of 'Router137'
        - Source protocol address: 128.143.137.1
            - IP address of 'Router137'
        - Target hardware address: 00:a0:24:71:e4:44
            - MAC address of the host, 'Argon'
        - Target protocol address: 128.143.137.144
            - MAC address of the host, 'Argon'
    - 'ARP' request and 'ARP' reply messages are sent throughout the entire local area network (LAN)
        - Other devices that are listening on the medium can (potentially) hear this transmission of 'ARP' messages
            - The transmission medium can be an ethernet bus or a wireless LAN
- Specialized ARP Messages
    - In addition to the request and reply messages, 'ARP' has specialized messages
        - i.e. ARP Probe, ARP Announcement
        - The request message explicitly asks for the MAC address corresponding to an IP address, and the reply message answers the request message
    - ARP Probe: Checks if anyone is utilizing an IP address
        - i.e. Sends a message like:
            - "Is anyone using this address?"
            - "This is the address I hope to use"
        - It is not used for the purpose of getting the MAC address of a particular IP address, rather it is used to check if there is another host that is utilizing the same IP address
        - Since DHCP ensures the uniqueness of IP addresses in a local area network (LAN), ARP Probe is not as useful as it was in the past
            - In the old days, when IP addresses were manually configured, ARP probing was quite useful
                - i.e. When you start your computer, and configure an IP address for it, you'll see that there is another host utilizing the same IP

address. In this case, ARP probing would be useful
        - ARP Probing works by setting:
            - Sender IP address to all 0's
            - Sender hardware is set to its own MAC address
            - Target MAC address to all 0's
            - Target IP address to the IP address that is being probed
        - If, on the local area network (LAN), there is a host with the same IP address, then it will respond with its own MAC address
            - Upon receiving this message, the host that sent the initial ARP Probe message will know that another host is utilizing the IP address
    - ARP Announcement: Informs the whole local area network (LAN) that a particular host is using a particular IP address
        - i.e. Sends a message like: "This is the IP address I am now using"
        - ARP Probe is different from ARP announcement
            - ARP Probing finds out if someone else is using an IP address, while ARP announcement makes a declaration that a particular host is using a particular IP address
        - Both the sender and target IP address fields are set to the IP address of the host making the announcement

        - Both the sender and target IP address fields contain the IP address being announced
- DHCP VS. ARP
    - DHCP: Dynamic Host Configuration Protocol
        - Used for acquiring an IP Address
        - A connecting host will send a broadcast message in the local area network (LAN) and ask for an IP address to be allocated to it
            - The DNS server will respond to the broadcast message, and after exchanging a few messages, like acknowledgements, the connecting host will be assigned a particular IP address
            - i.e.
                - Broadcast: "I need an IP address, please!"
                - Response: "You can have IP address 192.168.1.245"
    - ARP: Address Resolution Protocol
        - Operates on the link layer
        - Allows hosts to be able to discover the MAC address associated with a particular IP address
            - This is done via ARP Request/Reply
        - Allows hosts to declare that it is using a particular IP address
            - This is done via ARP Probe or ARP Announcement

        - i.e.
            - Broadcast: "Who has IP address 128.143.137.1?"
            - Response: "00-E0-23-6F-A8-20" has 128.143.137.1!"
- ARP Cache
    - The information learned through 'ARP' request and 'reply' messages are stored in an 'ARP' table
        - 'ARP' tables are also referred to as 'ARP' cache
    - Since sending an 'ARP' request/reply message for each IP datagram is inefficient, the 'ARP' information is stored over the shared medium
        - Hosts maintain a ('ARP') cache of current entries
            - In this case, hosts can learn the association between MAC addresses and IP addresses without explicitly requesting and receiving the corresponding reply
            - Entries consist of information such as IP address, and MAC address of the host's interface card, and the 'TTL' field
                - The default value of the 'TTL' field is 20 mins
            - Entires in the cache can be removed by the user, or they will be deleted upon timeout
    - i.e. Example of Contents of The ARP Cache on a Macbook:
        - (128.143.71.37) at 00:10:4B:C5:D1:15 [ether] on eth0
        - (128.143.71.36) at 00:B0:D0:E1:17:D5 [ether] on eth0
        - (128.143.71.35) at 00:B0:D0:DE:70:E6 [ether] on eth0
        - (128.143.136.90) at 00:05:3C:06:27:135 [ether] on eth0
        - (128.143.71.34) at 00:B0:D0:E1:17:DB [ether] on eth0
        - (128.143.71.33) at 00:B0:D0:E1:17:DB [ether] on eth0
- MAC Spoofing & ARP Poisoning
    - As the title suggests, there are some problems with ARP
    - Even though MAC addresses are hard coded in the network interface card (NIC), they can be (easily) modified in software, and a spoofed MAC address is utilized in place of the hard-coded MAC address on the NIC
        - i.e. On a Macbook running OSX, the following line, if executed in the terminal, will configure the network/ethernet interface with a particular MAC address: sudo ifconfig en0 ether 5e:c4:a4:99:b8:e3
            - This command requires root access, and it will alter the MAC address that is included in subsequent frames sent through the network/ethernet interface
        - MAC addresses can be (easily) changed/modified for Wi-Fi and Bluetooth devices
            - In fact, some Bluetooth devices use a dynamic address that constantly changes the address of the device to ensure that other people cannot trace one device by observing multiple transmissions from the same device
                - A practical application of this is COVID Contact Tracing (CTT). 'CTT' can determine if two

devices are coming in contact with one another. Since Bluetooth is a short range personal area network (PAN) protocol, where the transmission radius is tens of meters, devices can exchange messages corresponding to the user's COVID infection status.
        - In 'CTT', the communication needs to be as secure as possible, and it should not compromise people's privacy. Hence, the 'CTT' API/SDK built by Google and Apple utilizes the mechanism of dynamically generating a Bluetooth address. Thus, an attacker/sniffer cannot associate a device with a specific MAC address
    - The MAC address associated with packets/frames, that are transmitted over Ethernet/Air, can be easily changed
        - As a result, wireless access points can config some kind of user access control through address filtering
            - i.e. Limiting the set of MAC addresses that can communicate with the access point
                - This is bad practice
    - A MAC address can be easily changed/spoofed to someone else's MAC address
        - Thus, MAC address is not a good way to uniquely identify a device and use it for authentication purposes
    - MAC address filtering in WLAN access control isn't (really) "secure"
- ARP Poisoning
    - Since everything is sent in clear text, there is no way to tell that reply messages, to request messages, are accurate, originate from the device that has the corresponding IP address, and contain the correct MAC/ hardware address in the ARP reply
        - This is the root cause of the ARP poisoning problem
    - An attacker can do the following:
        - Generate a falsified ARP reply message, which will conflict with the ARP reply message generated by the legitimate host
            - This results in inconsistent information
        - Send an announcement declaring that the MAC address corresponding to an IP address is "this". Receiving hosts cannot determine if a (falsified) ARP announcement is accurate or not. The hosts will update their ARP table/cache, and utilize this information for subsequent communication with the newly specified host
    - If the cache is poisoned, then it may store a wrong MAC address that corresponds to the network interface of

        the attacker
          - As a result, instead of sending packets/frames to the router, they end up in the attacker's control
            - From here, an attacker may perform a 'Man In The Middle' attack, or another complicated attack
      - ARP poisoning is problematic, and is (typically) used as a way in for the attacker to launch other types of attacks
      - ARP poisoning is possible because there is no authentication
        - Authenticating all ARP messages creates a lot of overhead
        - There needs to be a better way to authenticate hosts, and make sure that the corresponding ARP messages are legitimate
- Q/A
  - Question: What are the implications of two devices having the exact same MAC address
    - First of all, from a hardware point-of-view this is impossible, because different blocks of MAC addresses are allocated to different vendors. However, since it is easy to spoof a MAC address, packets that are destined to the legitimate device are also received by the attacker. In addition, a spoofed MAC address allows an attacker to pretend to be the sender
    - A (typical) network interface card has the ability to filture packets based on information such as destination address. On a wireless network, a lot of communication takes place, where different devices talk to each other. So, the network interface and operating system will only allow packets that are destined to the MAC address that corresponds to the one on the network interface card, and broadcast messages/packets. Broadcast messages are allowed, because they are destined for everyone. Any packet that contains a MAC address different from the local interface will be filtered out. However, if the MAC address is spoofed, then incoming packets/frames will be passed on to the upper layer, and can get processed