# Network Layer

Study-Ready Notes

Compiled by Andrew Photinakis

November 17, 2025

## Contents

# Chapter 5: The Network Layer — Control Plane

# 1 5.1 Introduction

# 2 5.2 Routing Algorithms

## 2.1 Concept Overview

1. Logic of how packets forwarded is done in control plane, implemented by routing algorithms
2. A good path is synonymous with "least-cost", such path populates forwarding table in each router

## 2.2 Graph Abstraction: Modeling the Network

1. Assume graph G = (N , E)
2. N = nodes in graph that represent routers in network, are points where packet-forwarding decisions are made
3. E = edges, represent physical links that connect routers
4. Costs c(x, y) = each edge (x, y) is assigned a numerical cost
   (a) Cost is typically set by network admin, is a piece of policy
   (b) 3 types of costs
      i. Physical distance, a trans-oceanic link has a higher cost than a short link
      ii. Link speed, a 1 Gbps link might have a cost of 10, while a 100 Gbps link has a cost of 1. Often inversely proportional to bandwidth
      iii. Monetary cost, leasing a link from another provider may be more "costly"

## 2.3 Taxonomy of Routing Algorithms

1. Centralized
   (a) Algo computes least-cost path using complete, global knowledge of network
   (b) Has complete map of all nodes and all edge costs
   (c) Can be done at a single site (like an SDN controller) or replicated on every router, butkey is that the input is the global network state
   (d) Approach is known as a Link-State algorithm
2. Decentralized
   (a) Calculation distributed among routers

   (b) Each node begins with only local knowldge (cost of its own directly attached links)
   (c) Iterative process of communication only with neighbors, a node gradually learns least-cost paths to other nodes
   (d) Approach know as Distance-Vector algorithm

3. Static vs Dynamic
   (a) Static: routes are changed very slowly, often by human admin manually editing costs or forwarding tables
   (b) Dynamic: routes cahnge automatically as network topology or link costs change.

4. Load-Sensitive vs Load-Insentive
   (a) Load-Sensitive: link costs dynamically change to reflect current level of congestion on link. Idea to route around congestion
   (b) Load-Insensitive: link costs are fixed and do not reflect current load

## 2.4 5.2.1 Link State Routing Algorithm

1. Centralized approach that uses complete, global knowledge of network and lnk costs to compute least-cost paths
2. Each node in network responsible for a simple task
   (a) Discover neighbors and link costs
   (b) Broadcast this info to all other nodes in network
   (c) Receive link-state packets from all other nodes and build a complete, identical map of entire network
   (d) Compute least-cost paths from itself to all other nodes using map
3. Dijkstras Algorithm
   (a) Iterative algo, finds least-cost paths in increasing order of cost
   (b) After k-th iteration, algo has found least-cost path to kndoes that are closest to source
   (c) Sicne each router computes its own table independently based on received global info, a malfunctioning router can only broadcast an incorrect cost for its attached links, and cannot directly corrupt entire network's routing knowledge
   (d) Variables
      i. N: set of nodes to which least-cost path is definitively known
      ii. D(u): current cost of least-cost path from source (u) to node v. value updated as new, shorter paths found
      iii. p(u): predecessor node neighbor of v along current least-cost path from source. Used to build forwarding table

```
    // Initialization
1.  N' = {u}  // The source node is the only member
2.  for all nodes v
3.      if v is a neighbor of u
4.          then D(v) = c(u,v)
5.          else D(v) = infinity
6.
// Loop
7.  loop
8.      find w not in N' such that D(w) is a minimum
9.      add w to N'
10.     update D(v) for each neighbor v of w and not in N':
11.         // Does the path through w to v offer a shorter
12.         D(v) = min( D(v), D(w) + c(w,v) )
13.         // If D(v) was updated, set p(v) = w
14. until N' = N
```

(e) Pathological Oscillations

i. Problem arises if link costs are load-sensitive, that is if cost of a link depends on amount of traffic it is carrying

ii. Cost of link equals traffic it carries

iii. Solution is to not use load sensitive costs and ensure routers don't run algorithm at same time. Done by randomizing when link state updates are sent or when algorithm is run

## 2.5 5.2.2 Distance Vector Routing Algorithm

1. Distance vector algorithm is decentralized approach
2. Code is it "tell your neighbors what you know about the world"
3. Algorithm is:
   (a) Distributed, each node communicates only with direct neighbors
   (b) Iterative, process repreats until no more information is exchanged and calcualtions stabilized
   (c) Async, nodes don't need to operate in lock step
   (d) Self-Terminating, algorithm stops when calculations converge
4. In DV, a node knows cost to its neighbors, and receives it "distance vectors" from its neighbors. A distance vector is a list of cost estimates from that neighbor to all other nodes in network

### 2.5.1 Bellman-Ford Algorithm

1. **Definitions of variables:**
   - $d_x(y)$: Node $x$'s estimate of the least-cost path from $x$ to destination $y$.
   - $c(x, v)$: The cost of the direct link from node $x$ to neighbor $v$.
   - $N(x)$: The set of neighbors of node $x$.
   - $d_v(y)$: Node $v$'s estimate of the least-cost path from $v$ to destination $y$.
2. **Bellman–Ford update formula:**

$$d_x(y) = \min_{v \in N(x)} \{c(x, v) + d_v(y)\}$$

   - Node $x$ considers every neighbor $v \in N(x)$.
   - For each neighbor, it computes:

$$c(x, v) + d_v(y)$$

   - It chooses the neighbor $v$ that yields the smallest total cost.
3. **Intuition:** The cost from node $x$ to destination $y$ is the minimum, over all neighbors $v$, of the cost to reach $v$ plus the cost (as advertised by $v$) for $v$ to reach $y$.
4. **Distance Vector at each node:** Each node $x$ maintains a distance vector

$$D_x = \left[D_x(y) : y \in \mathcal{N}\right]$$

   representing its current estimates of $d_x(y)$ for all destinations $y$.
5. **Update behavior:** When node $x$ receives a distance vector $D_v$ from neighbor $v$, it recomputes its own distance vector using the Bellman–Ford formula. If any entry changes, node $x$ sends its updated distance vector $D_x$ to all neighbors.
6. **Distance Vector Algorithm (Pseudo-Code):**

```
At each node x:

1. Initialization:
2.     for all destinations y in N:
3.         D_x(y) = c(x,y)       // If y not a direct neighbor: c(x,y) = infinity
4.     for each neighbor w:
5.         send distance vector D_x to w

6. loop
7.     wait until a distance vector D_v is received from neighbor v
```

```
8.     for each destination y in N:
9.         // Bellman-Ford update
10.        D_x(y) = min over all neighbors u { c(x,u) + D_u(y) }

11.    if any D_x(y) changed:
12.        send updated distance vector D_x to all neighbors

13. forever
```

7. Pathological Dynamics
   (a) Complexity of DV is when link costs change
   (b) Cases to consider:
      i. Link cost decreases
      ii. Link cost increases at & count to infinity problem
   (c) Solution
      i. Poisoned Reverse
      ii. A simple fix for 2-node loops
         A. if z routes to x through y, z will "lie" to y and advertise its distance to x to inifinity
      iii.

### 2.5.2 Comparison of Link-State vs Distance-Vector Algorithms

### 2.5.3 5.2 Section-Wide Summary

**Key Terms and Definitions**

- **Routing Algorithm**: The control-plane process for finding "good" (least-cost) paths through a network of routers.
- **Graph** ($G = (N, E)$): Abstract model of a network where nodes $N$ are routers and edges $E$ are links with associated costs.
- **Least-Cost Path**: The path between two nodes with the minimum sum of edge costs.
- **Centralized Algorithm**: Requires complete, global network state (a full map) as input.
- **Decentralized Algorithm**: Nodes compute paths iteratively using only local information from neighbors.
- **Link-State (LS) Algorithm**: Canonical centralized approach (e.g., Dijkstra's). Each node floods its local link information to all others.

| **Feature** | **Link-State (LS) Algorithm** | **Distance-Vector (DV) Algorithm** |
|---|---|---|
| Information Shared | Each node broadcasts the cost of its directly attached links to all other nodes (flooding). | Each node sends its distance vector (its current cost estimates to all destinations) only to its immediate neighbors. |
| Knowledge | **Global**: Every node builds a complete map of the network topology. | **Local**: A node knows only its neighbors and the information received from them. |
| Message Complexity | High. Each link-state update is flooded across the entire network: typically $O(N\,E)$ messages. | Lower. Only neighbor-to-neighbor exchanges; typically $O(N^2)$ over time depending on convergence. |
| Speed of Convergence | Fast. Once link-state information propagates, each node runs Dijkstra in $O(N^2)$ or $O(N \log N)$. No loops during convergence. | Slower. Can suffer from routing loops and the *count-to-infinity* problem. |
| Robustness | High. A node computes its own routes independently; a faulty node can only report incorrect local link costs. | Lower. Incorrect distance vectors can propagate from one node to another and corrupt the entire network state. |
| Used By | OSPF, IS-IS | RIP, BGP (path vector variant) |

- **Dijkstra's Algorithm**: Greedy, iterative algorithm used in LS to compute shortest paths from a source to all other nodes.
- **Distance-Vector (DV) Algorithm**: Canonical decentralized approach (e.g., Bellman–Ford). Each node sends its distance vector to its neighbors.
- **Bellman–Ford Equation**:

$$d_x(y) = \min_{v \in N(x)} \{c(x, v) + d_v(y)\}$$

- **Count-to-Infinity**: A pathological DV behavior where a link cost increase causes slow propagation of "bad news," leading to routing loops and repeatedly increasing distance estimates.
- **Poisoned Reverse**: A mitigation for 2-node loops in DV; a node advertises an infinite distance to a destination through the neighbor it routes through.

**Core Relationships**

- Routing algorithms compute the paths that populate the forwarding tables in the data plane.

- Centralized algorithms correspond to LS (e.g., Dijkstra's).
- Decentralized algorithms correspond to DV (e.g., Bellman–Ford).
- The Bellman–Ford equation defines shortest paths recursively; DV implements this recursively and distributively.
- Count-to-Infinity is the major weakness of DV and results from slow propagation of link-cost increases.

**Key Insights and Takeaways**

- All routing algorithms model the network as a graph and attempt to solve the least-cost path problem.
- **Fundamental tradeoff:**
  - LS (centralized): robust, fast convergence, but high message overhead (flooding).
  - DV (decentralized): low message overhead (neighbor-only), but slow convergence and vulnerable to loops.
- Pathological behaviors matter:
  - LS can oscillate when link costs depend on load.
  - DV can count to infinity when link costs increase.
- The Internet uses both approaches:
  - OSPF $\rightarrow$ LS
  - BGP $\rightarrow$ DV-like (path vector)
- No single algorithm is universally superior—both excel in different contexts.

# 3 5.3 Intra-AS Routing in the Internet: OSPF

## 3.1 Need for Routing Hierarchy: Scale and Autonomy

A "flat" network where every router runs same algorithm doesn't work for global internet for 2 reasons

1. Scale
   (a) Storage and Computation
      i. Internet has hundreds of millions of routers
      ii. A link-state algorithm like Dijkstra's, which requires each router to store a map of the entire network and run an $O(N^2)$ or $O(N \log N)$ computation, is completely unfeasible at this scale.

   (b) COmmunication Overhead
      i. a link-state algo requires link-state 'advertisements' (LSAs) to be flooded to every other router
      ii. Communication overheard would be overwhelming.
      iii. Distance vector would never converge

2. Administrative Autonomy
   (a) Internet is not one network, it's composed of thousands of individual ISPs, universities, and corporate networks
   (b) An ISP is its own administrative domain. It wants and needs to run its network as it sees fit. SHould be able to choose its own internal routing algo and manage its network for its own performance and economic goals.
   (c) Furthermore, an ISP has no desire to "broadcast" details of its internal network topology and link costs to its competitors

## 3.2 Solution: Autonomous Systems (ASs)

1. Is a group of routers that are all under same admin control, typically a single ISP or a large organization
2. Each AS identified by a globally unique Autonomous System Number (ASN), which is assigned by ICANN regional registries
3. Two-level hierarchy
   (a) Intra-AS Routing (Interior Gateway Protocol - IGP)
      i. Routing algorithm that runs within a single AS
      ii. ALl routes in AS are teammates and run samw intra-AS protocol
      iii. Focus is on performance, finding best path within network
   (b) Inter-AS Routing (Exterior Gateway Protocol - EGP)
      i. routing algorithm that runs between ASs
      ii. Protocol is used to route traffic across different ISPs
      iii. Focus is on policy, enforcing economic and security rules (e.g. "I will not carry traffic from this competitor for free")

## 3.3 OSPF: Canonical Intra-AS Protocol

1. Open Shortest Path First
2. is a link-state protocol that uses Dijkstras algo to compute shortest paths
3. How it works

   (a) Construct map: each router in AS constructs a complete, identical topological map of entire AS

   (b) Flood Information: To build map, each router broadcasts its link-state information (i.e. its list of directly connected neighbors and costs of those links) to all other routers in AS.

   (c) Run Dijkstras: Once router has complete map, it runs Dijkstras alg using itself as root node. Finds least-cost path from itself to every other node (and subnet) in AS

   (d) Install Routes: router uses results of Dijkstras to build forwarding table

4. Practical View: Link Costs

   (a) Cost of a link is not standard, its a configured parameter set by network admin

   (b) OSPF provides mechanism to find least-cost path, but policy of what "cost" means is up to admin

## 3.4 OSPF Link-State Broadcasts

Detail of OSPF is how it manages its link-state advertisements (LSAs)

1. Transport: messages are not sent over TCP or UDP, sent directly inside IP datagrams, using IP protocol number 89

2. Implication: Doesn't use a transport protocol, must implement its own reliability. Checks that links are operational (using HELLO messages sent to neighbors), and allows routers to get a neighbors link-state database

3. When to broadcast

   (a) On Change: A router broadcasts its link-state information whenever one of its links changes state (e.g. goes down, comes up, or its cost is unchanged)

   (b) Periodically: for robustness, router also re-broadcasts its link state at a regular interval even if there has been no change

## 3.5 Advanced Features in OSPFv2

is not just a simple implementation of Dijkstras. [RFC 2328] specifies several advances features that are critical in real-world networks

1. Security

   (a) How do you stop a malicious router from joining AS and injecting false link-state advertisements?

   (b) OSPF allows for authenticaion of OSPF messages

      i. Simple Auth: clear-text password, which is weak

ii. MD5 Auth
   A. All routers configured with a shared secret key
   B. Router computes an MD5 hash of OSPF packet (including key) and sends hash with packet
   C. Receiver does same computation and verifies hash
   D. Provides message integrity and authentication, and sequence numbers are used to prevent replay attacks

2. Multiple Same-Cost Paths
   (a) What if Dijkstra's finds two paths to a destination with exact same cost?
   (b) Instead of just picking one, OSPF allows router to install both paths in its forwarding table
   (c) Traffic destined for that node can then be split across both paths
   (d) Simple and effective form of load balancing
3. Support for Hierarchy (Areas)
   (a) For a very large AS, flooding LSAs to every touer is too much overhead
   (b) Solution: AS can be partitioned into areas
   (c) Inra-Area routing: routers within an area run their own OSPF algorithm. THey flood LSAs only to other routers in same area. Each router in area has a complete map of only its area
   (d) Backbone area (Area 0): One special area is designated as backbone. Primary role of backbone is to route traffic between different areas
   (e) Area Border Routers (ABRs): Routers that sit at edge of an area. They're in backbone and in one or more other areas. Summarize routing information from their area and advertise it to the backbone, and vice-versa
4. Multicast support: OSPF extended by Multicast OSPF (MOSPF), which uses same link-state database and flooding mechanism to compute multicast paths

## 3.6 Section-Wide Summary

**Key Terms and Definitions**

- **Autonomous System (AS):** A group of routers under a single administrative control.
- **Intra-AS Routing (IGP):** Routing protocol used within an AS; focus is on performance.
- **Inter-AS Routing (EGP):** Routing protocol used between ASs; focus is on policy.
- **OSPF (Open Shortest Path First):** A widely used, open-standard, link-state, intra-AS routing protocol.

- **Link-State Advertisement (LSA):** A broadcast packet containing the state and cost of a router's links, used by OSPF.
- **Traffic Engineering:** Setting link costs to achieve network-wide performance goals (e.g., minimize congestion).
- **OSPF Area:** A sub-domain within a large AS used to create hierarchy and limit LSA flooding.
- **Backbone Area (Area 0):** The central area in a hierarchical OSPF network connecting all other areas.
- **Area Border Router (ABR):** A router that connects one or more areas to the backbone area.

**Core Relationships**

- The Internet is divided into ASs to address issues of scale and administrative autonomy.
- This leads to a two-tier routing system: Intra-AS (e.g., OSPF) and Inter-AS (e.g., BGP).
- OSPF is a practical implementation of the Link-State algorithm (using Dijkstra's).
- Routers flood LSAs to build an identical network map, then compute shortest paths independently.
- Large ASs use OSPF hierarchy (Areas + Backbone) to limit LSA flooding and reduce computation.

**Key Insights / Takeaways**

- Internet routing is fundamentally hierarchical: Intra-AS and Inter-AS levels enable global scalability.
- OSPF is the canonical Intra-AS protocol, relying on the Link-State approach.
- OSPF offers robustness: authentication, multi-path load balancing, and hierarchical scalability.
- OSPF link costs are policy tools—not purely physical measurements—and can be tuned for Traffic Engineering.

1. X
    (a) Y