Routing Basics

This section will focus on the necessary background knowledge needed to implement your own link-state routing protocol.

## What is Routing?

Routing is the process of learning all the paths through the [network](https://www.inetdaemon.com/tutorials/networking/lan/index.shtml) ([routes](https://www.inetdaemon.com/tutorials/internet/ip/routing/route.shtml)) and using routes to forward data from one [network](https://www.inetdaemon.com/tutorials/networking/lan/index.shtml) to another. Routing protocols are used to facilitate the exchange of routing information between routers.

## What is a Routing Protocol?

A routing protocol is a set of processes, algorithms, and messages that are used to exchange routing information and populate the **routing table** with the routing protocol’s choice of best paths. Routing protocols ride on top of the IP layer.

**Routing table** - a data table stored in a router or a network host that lists the routes to particular network destinations, and in some cases, metrics (distances) associated with those routes.

Routing protocols can be divided into two categories: *dynamic and static*. For this project, we will be focusing on dynamic routing.

## Dynamic Routing Protocols

#### What is a Dynamic Routing Protocol?

Dynamic routing protocols allow routers to dynamically share information about remote networks and automatically add this information to their own routing tables.

Compared to static routing, dynamic routing protocols require less administrative overhead. However, the expense of using dynamic routing protocols is dedicating part of a router’s resources for protocol operation, including CPU time and network link bandwidth. Despite the benefits of dynamic routing, static routing still has its place. There are times when static routing is more appropriate and other times when dynamic routing is the better choice. Networks with moderate levels of complexity may have both static and dynamic routing configured.

The purpose of dynamic routing protocols includes:

* Discovery of remote networks
* Maintaining up-to-date routing information
* Choosing the best path to destination networks
* Ability to find a new best path if the current path is no longer available

The main components of dynamic routing protocols include:

* ***Data structures***: Routing protocols typically use tables or databases for their operations. This information is kept in RAM.
* ***Routing protocol messages***: Routing protocols use various types of messages to discover neighboring routers, exchange routing information, and perform other tasks to learn and maintain accurate information about the network.
* ***Algorithm***: An algorithm is a finite list of steps used to accomplish a task. Routing protocols use algorithms for facilitating routing information and for best path determination.

([Routing Protocols Companion Guide](https://www.ciscopress.com/store/routing-protocols-companion-guide-9781587133237?w_ptgrevartcl=Cisco+Networking+Academy%27s+Introduction+to+Routing+Dynamically_2180210))

#### Dynamic Routing Operation

In general, the operations of a dynamic routing protocol can be described as follows:

1. The router sends and receives routing messages on its interfaces.
2. The router shares routing messages and routing information with other routers that are using the same routing protocol.
3. Routers exchange routing information to learn about remote networks.
4. When a router detects a topology change, the routing protocol can advertise this change to other routers.

## Link State Routing

#### What is Link-State Routing?

Link-State Routing is when a routing protocol shares information with other routers in order to determine its path. Rather than continuously broadcasting its routing tables like in distance vector protocols, a link-state protocol router only notifies its neighboring routers when it detects a change.

* With link-state routing protocols, a link is an interface on a router.
* Information about the state of those links is known as link-states.
* Link-state updates (LSUs) are the packets used for updates from the routing protocol.

#### Link-State Routing Operations

All routers in an the autonomous System network will complete the following generic link-state routing process to reach a state of convergence, where all routers that have the same topological information about the internetwork in which they operate:

1. Each router learns about its own links and its own directly connected networks. This is done by detecting that an interface is in the up state.
2. Each router is responsible for meeting its neighbors on directly connected networks. Link-state routers do this by exchanging Hello packets with other link-state routers on directly connected networks.
3. Each router builds a ***link-state packet (LSP)*** containing the state of each directly connected link. This is done by recording all the pertinent information about each neighbor, including neighbor ID, link type, and bandwidth.
4. Each router floods the LSP to all neighbors. Those neighbors store all LSPs received in a database. They then flood the LSPs to their neighbors until all routers in the area have received the LSPs. Each router stores a copy of each LSP received from its neighbors in a local database.
5. Each router uses the database to construct a complete map of the topology and computes the best path to each destination network. Like having a road map, the router now has a complete map of all destinations in the topology and the routes to reach them. The SPF algorithm is used to construct the map of the topology and to determine the best path to each network.

## Common Routing Protocol Operations

As in common across all routing protocols...

#### “Cold Start”:

When a router powers up, it knows nothing about the network topology. It does not even know that there are devices on the other end of its links. The only information that a router has is from its own saved configuration file stored in **NVRAM**.

**NVRAM** - Non-Volatile Random Access Memory (NVRAM) is a category of Random Access Memory (RAM) that retains stored data even if the power is switched off.

After a router boots successfully, it applies the saved configuration. If the IP addressing is configured correctly, then the router initially discovers its own directly connected networks.

Notice how the routers proceed through the boot process and then discover any directly connected networks and subnet masks. This information is added to their routing tables.

With this initial information, the routers then proceed to find additional route sources for their routing tables.

#### Network Discovery:

After initial boot up and discovery, the routing table is updated with all directly connected networks and the interfaces those networks reside on.

If a routing protocol is configured, the next step is for the router to begin exchanging routing updates to learn about any remote routes.

The router sends an update packet out to all interfaces that are enabled on the router. The update contains the information in the routing table, which currently comprises all directly connected networks.

At the same time, the router also receives and processes similar updates from other connected routers. Upon receiving an update, the router checks it for new network information. Any networks that are not currently listed in the routing table are added.

([Cisco Networking Academy's Introduction to Routing Dynamically](https://www.ciscopress.com/articles/article.asp?p=2180210))

## Open Shortest Path First (OSPF)

#### This project is modeled after the routing protocol Open Shortest Path First (OSPF), but with a different algorithm. It is a link-state routing protocol that utilizes Dijkstra's Algorithm to calculate the shortest path. Here are its properties…

* OSPF allows [routers](https://www.inetdaemon.com/tutorials/internet/ip/routing/router.shtml) to dynamically learn [routes](https://www.inetdaemon.com/tutorials/internet/ip/routing/route.shtml) from other [routers](https://www.inetdaemon.com/tutorials/internet/ip/routing/router.shtml) and to advertise [routes](https://www.inetdaemon.com/tutorials/internet/ip/routing/route.shtml) to other [routers](https://www.inetdaemon.com/tutorials/internet/ip/routing/router.shtml).
* Advertisements containing [routes](https://www.inetdaemon.com/tutorials/internet/ip/routing/route.shtml) are referred to as Link State Advertisements (LSAs) in OSPF.
* OSPF router keeps track of the state of all the various network connections (links) between itself and a [network](https://www.inetdaemon.com/tutorials/networking/lan/index.shtml) it is trying to send data to. This makes it a link-state [routing protocol](https://www.inetdaemon.com/tutorials/internet/ip/routing/routing_vs_routed.shtml).
* OSPF supports the use of [classless](https://www.inetdaemon.com/tutorials/internet/ip/addresses/classless.shtml) [IP address](https://www.inetdaemon.com/tutorials/internet/ip/addresses/index.shtml) ranges and is very efficient.
* OSPF uses areas to organize a [network](https://www.inetdaemon.com/tutorials/networking/lan/index.shtml) into a hierarchical structure; it summarizes [route](https://www.inetdaemon.com/tutorials/internet/ip/routing/route.shtml) information to reduce the number of advertised routes and thereby reduce network load and uses a designated router (elected via a process that is part of OSPF) to reduce the quantity and frequency of Link State Advertisements.
* OSPF does require the [router to have](https://www.inetdaemon.com/tutorials/internet/ip/routing/router.shtml) a more powerful [processor](https://www.inetdaemon.com/tutorials/computers/hardware/cpu/index.shtml) and more [memory](https://www.inetdaemon.com/tutorials/computers/hardware/memory/index.shtml) than other [routing protocols](https://www.inetdaemon.com/tutorials/internet/ip/routing/routing_vs_routed.shtml).
* OSPF selects the best routes by finding the lowest cost paths to a destination. All router interfaces (links) are given a cost. The cost of a route is equal to the sum of all the costs configured on all the outbound links between the router and the destination network, plus the cost configured on the interface that OSPF received the Link State Advertisement on.

#### Neighbor Discovery

An OSPF neighbor is any other adjacent router that is connected to the router (physically, or logically over a tunnel) and which runs OSPF and which the router is intended to exchange routing information. When OSPF is enabled, HELLO messages (type 89 in the IP header) are addressed to the IP address 225.0.0.5 and flooded out all interfaces participating in OSPF every 10 seconds. Any neighboring router that hears these HELLO messages will learn:

* The neighbor's Router ID (RID)
* The Area ID the neighbor participates in
* The HELLO interval (default is 10 seconds)
* The DEAD interval (usually 5 failed HELLOs)
* The Router ID (RID) of the Designated Router (if known)
* The Router ID (RID) of the Backup Designated Router (if known)
* The Router ID of all known routers

##### Forming Neighbor Adjacencies

Once a router sees its own ID in a neighbor's HELLO, it assumes two-way communication has been accomplished and the router begins sending link state advertisements (LSAs). This will only occur if the following items are configured identically on the two routers:

* The HELLO interval
* The DEAD interval
* Authentication checks
* The OSPF area ID
* The stub area flag value

What Makes Up a Routing Protocol?

This section talks about the parts of a routing protocol that will be implemented in code.

* node data structure (a node is a router or machine)
* neighbor data structure (keeping track of which routers are directly adjacent/connected to each node)
* network topology data structure (a graph of nodes)
* an interface for communicating with the router/getting info from the router (the link)
* the sending and receiving of routing protocol packets
* hello protocol (timer, sending and receiving hello packets)
* calculating the routing table
* flooding protocol for link state advertisements (LSAs) and updates (LSUs)
* set designated router and then backup designated router.

How to Run A Protocol on a Router?

You can run your own compiled programs on a router with some open source technologies like:

* OpenWRT -- **OpenWrt** (OPEN Wireless **RouTer**) is an open source project for embedded operating systems based on Linux, primarily used on embedded devices to route network traffic. ... **OpenWrt** can run on various types of devices, including CPE **routers**, residential gateways, smartphones, pocket computers (e.g. Ben NanoNote), and laptops. So, it can designate a machine as a router.
* <https://openwrt.org/docs/guide-user/network/routing>
* <https://www.gargoyle-router.com/old-openwrt-coding.html>
* pfSense -- a free and open source firewall and router that also features unified threat management, load balancing, multi WAN, and more. You would need to figure out how to build your own package to run on pfsense.
* <https://docs.netgate.com/pfsense/en/latest/recipes/dynamic-routing-basics.html#ospf>
* <https://forum.netgate.com/topic/19230/building-custom-packages-for-pfsense>
* <https://community.spiceworks.com/topic/531006-pfsense-creating-a-package>
* FRRouting -- includes protocol daemons for BGP, IS-IS, LDP, OSPF, PIM, and RIP. FRR’s seamless integration with the native Linux/Unix IP networking stacks makes it applicable to a wide variety of use cases including connecting hosts/VMs/containers to the network, advertising network services, LAN switching and routing, Internet access routers, and Internet peering. You would need to figure out how to run your protocol by adding it to the daemons collection in the file system.
* <https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/8/html/configuring_and_managing_networking/setting-your-routing-protocols_configuring-and-managing-networking>
* <https://docs.frrouting.org/_/downloads/en/latest/pdf/>

Useful Resources and C Libraries

<**sys**/**types**. **h**>

header contains a number of basic derived **types** that should be used whenever appropriate. In particular, the following are of special interest: clock\_t. The **type** clock\_t represents the system times in clock ticks.

<**stdint**. **h**>

is a header file in the **C** standard library introduced in the C99 standard library section 7.18 to allow programmers to write more portable code by providing a set of typedefs that specify exact-width integer types, together with the defined minimum and maximum allowable values for each type, using macros .

**netlink library**

[**https://www.infradead.org/~tgr/libnl/doc/route.html#\_routing**](https://www.infradead.org/~tgr/libnl/doc/route.html#_routing)

**IP Helper Functions**

[**https://www.winsocketdotnetworkprogramming.com/winsock2programming/winsock2advancediphelperfunction13b.html**](https://www.winsocketdotnetworkprogramming.com/winsock2programming/winsock2advancediphelperfunction13b.html)

## Similar Assignments From other Universities

<https://www.niche-associates.com/links/cd/tpd/instructor.pdf>

<https://www2.cs.siu.edu/~networksim/faq.html>

<https://www.cs.colostate.edu/~cs457dl/yr2016sp/more_assignments/proj3.html>

<http://www.cs.cornell.edu/skeshav/book/routing2/>

<https://github.com/chuan6/link-state-routing>

<https://github.com/pcyin/OSPF_Router>

https://www.pluralsight.com/blog/it-ops/dynamic-routing-protocol