Implementing a Dynamic Routing Protocol PWOSPF

Overview

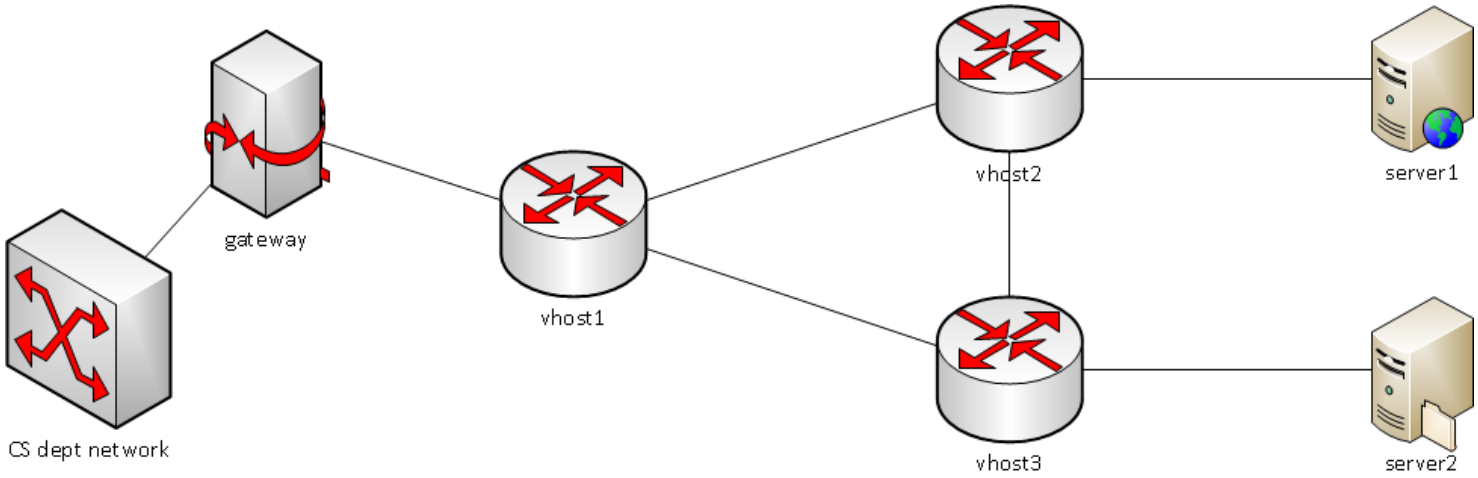
This assignment involves building advanced functionality on top of a basic virtual router. The goal is to implement a simple dynamic routing protocol, PWOSPF, so that your router can create its forwarding table automatically based on routes learned from other routers in the network. By the end of this project, your router is expected to be able to build its forwarding table from link-state advertisements sent from other routers, and route traffic through a topology containing multiple nodes.

PWOSPF

The routing protocol you will be implementing is a link state protocol that is loosely based on OSPFv2. You may find a full specification of PWOSPF here. Note that while PWOSPF is based on OSPFv2 it is sufficiently different that referring to the OSPFv2 as a reference will not be of much help and contrarily may confuse or mislead you.

Topology and Requirements Overview

Each student will be assigned the following three host topology for development:



When you are assigned your topology, please ensure that you can connect to the server and reserve each host. Your task is to implement PWOSPF within your existing router so that your router will be able to do the following:

- build the correct forwarding tables on the assignment topology
- detect when routers join/leave the topology, and/or when links fail/recover, and correct the forwarding tables accordingly

Running Multiple Routers

Because this assignment requires multiple instances of your router to run simultaneously you will want to use the **-r** and the **-v** command line options. **-r** allows you to specify the routing table file you want to use (e.g. -r rtable.vhost1) and **-v** allows you to specify the host you want to connect to on the topology (e.g. -v vhost3). Connecting to vhost3 on topology 300 should look something like:

./sr -t 300 -v vhost3 -r rtable.vhost3

Dealing With Routes

The basic router project does not really require sophisticated handling of routes. However, when implementing pwospf, special care must be taken to ensure that routes are treated correctly by your router. Before starting pwospf development, we suggest that you first ensure your router properly handles default routes, gateways and subnets.

next hop: The routing table consists of 4 fields: destination prefix, next-hop, network mask and the outgoing interface. The next-hop specifies the IP address of the next hop router towards the destination. What if the destination is directly connected to one of the router's interfaces? In this case, the next-hop value is 0 (0.0.0.0) serving as an indication to the router that the packets destination address is directly connected to outgoing interface specified by the route. If your router receives a packet to destination 1.2.3.4 and your router has a route:
1.2.3.0 0.0.0.0 255.255.255.0 eth1
Your router should use 1.2.3.4 as the next hop out of eth1.

subnets (and longest prefix mask): The subnet-mask field of the routing table specifies the size of the subnet being routed to. More specifically, given a particular destination IP address (ipdest), it matches with a route if (ipdest & mask) equals (dest & mask), where dest is the destination prefix field in the route. It is important to note that a given IP address can match with multiple routes (in fact this is

often the case). For example, all destinations match the default route ((ipdest & 0) = 0)! In these cases, the router must choose the **best** or **most specific** match. This is done by selecting the match with the longest prefix or the largest value subnet mask. Because you will be updating your forwarding table dynamically, you will need to find an (efficient) method for supporting longest prefix match rather than just selecting the first route that matches.

You will notice that each link in the 3 router topology above is assigned a 2 IP subnet (subnet mask of 255.255.255.254). Be sure to consider the subnet mask when handling routes.

Static vs. Dynamic Routes: During operation, your routing table should contain both static and dynamic routes. Static routes are read in from the routing table (rtable) and should never be removed or modified. Dynamic routes are added/removed by your PWOSPF implementation. You must therefore keep track of which routes are static. You handle this however you like, e.g. maintaining two separate tables, or marking routes as static/dynamic. For consistency your router should prefer static routes over dynamic routes when they both match the destination address and have the same prefix length, otherwise the longer prefix match should be used.

Control Path Versus Forwarding Path

High end hardware routers and software routing daemons abstract the control functionality of the router such as management, and routing protocols away from the forwarding functionality. The forwarding path should do just that, forwarding. The control path, on the other hand, tries to figure out what the particular configuration of the forwarding path should be and updates the forwarding path periodically. Typically, the configuration functionality will maintain copies of the forwarding path data structures (such as the routing table) and use those for updates. You have already developed a usable forwarding path in the basic router assignment. In this project you will build a control path on top of that to update the routing table based on the calculated shortest path to each reachable subnet.

What Routes to Advertise

Your router is responsible for advertising all the subnets connected to its interfaces. E.g., in the above topology graph, each vhost should advertise 3 subnets. Given any router on the network, aggregating all of this information properly should create a subnet graph from which you can calculate the correct routing table for that router to each of the advertised subnets. Exactly how to do this is part of the assignment challenge.

Some Pitfalls

Be careful not to add a route in your routing table for subnets directly connected to one of your interfaces even if these subnets are being advertised (in most cases they will be e.g. if there is another router on the subnet).

You're likely to need multiple threads in this assignment to support the periodic updates (e.g. HELLO packets). Be careful to avoid race conditions. Your routing table in particular will need to be locked off so that you don't fall of the end when looking up a route for a data packet during updates.

Testing and debugging in this assignment can be difficult. You may want to add code to your router so that you can disable an interface at run time. You can use this to test whether your implementation converges after a link goes down (e.g. if the link between vhost1 and vhost2 goes down, app1 should still be reachable on the path vhost1 -> vhost3 -> vhost2 -> app1).

Expected Functionality

We expect to start three instances of your router with **the only static route being the default route (to Internet) on vhost1,** and your router should be able to build the correct routing tables, route traffic to the application servers on the assignment topology, and correctly rebuild routing tables after a link failure or recovery.
