

Lab 6: Intergroup Routing

Sam Reynolds

March 27th, 2005

With lab 6 we will put the various group's OSPF networks together with what I am calling a BGP-lite. Those of you who have taken Unix Network Programming will recognize that this is similar, with a few key changes, to the RIP-link routing protocol that we used there. The changes make it more dynamic, and in my opinion, somewhat more elegant.

I have tried to weed out any problems I foresaw in this protocol, but that definitely does not mean there are no lurking caveats. If you find troubles, let me know and if changes need to be made I will update this document, and announce to the mailing list.

Headers

```
typedef struct{
    int networks;
    unsigned short length;
    unsigned short checksum;
}BGP_Header;

typedef struct{
    struct in_addr network;
    struct in_addr netmask;
    int num_hops;
}BGP_Data;
```

Note: Just use these, no need to prepend the Router_Header that we use for OSPF.

Protocol

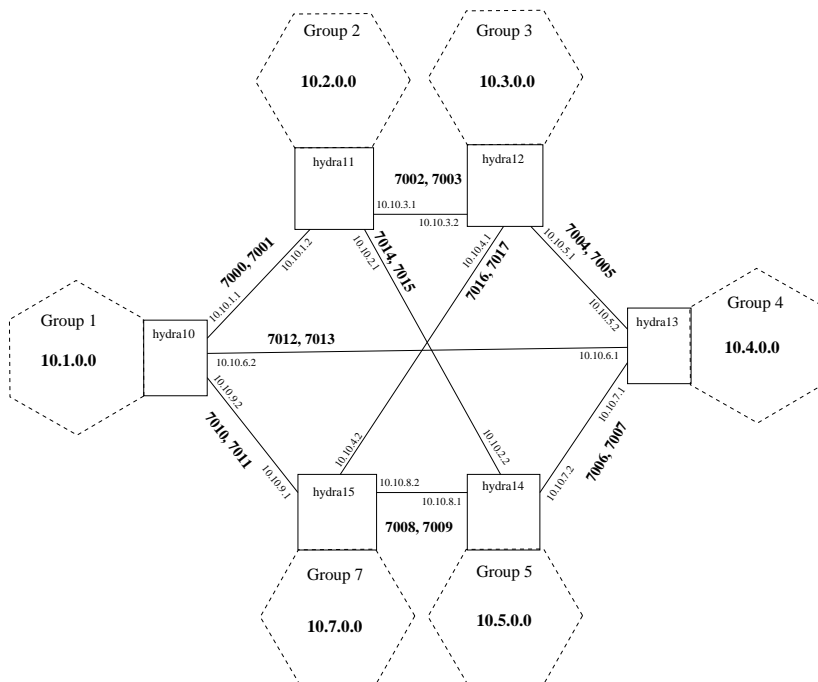
Here are the basic rules for our protocol:

- The IP protocol number to use is 6. This is TCP's protocol number, which seemed appropriate for a BGP-lite.

- Send out a full advertisement every 30 seconds only to your neighbors
- Time out routers after some number of missed advertisements. I recommend 3 or 4.
- Invalidate routes through a router that has gone down
- When you invalidate a route, you should alert your neighbors on your next update by having those routes in the advertisement with *num_hops* = -1. After this you can not include them in future advertisements, unless of course the router comes back to life.
- Implement the split-horizon hack to avoid count-to-infinity.
- Advertisements should be subnet broadcasted to each of your “BGP” interfaces.
- To make our lives easier, let’s continue to enforce that when you send out an advertisement on an interface, the source address should be the address of the interface it came from. This keeps us from having to grab neighbor addresses from a config file.
- For the checksum, just use the same checksum algorithm we have been using. It is linked on the website.
- Obviously, you should use the shortest number of hops to route packets. You do not need to keep alternate routes.

Layout

I have designed this layout for the groups. Note, you will need to have your neighbors correctly listed in your vnet config files, and you will have to have the ports match up as well. Dr. Straight may have a configuration he would like better, in which case I will update this document and announce to the mailing list.



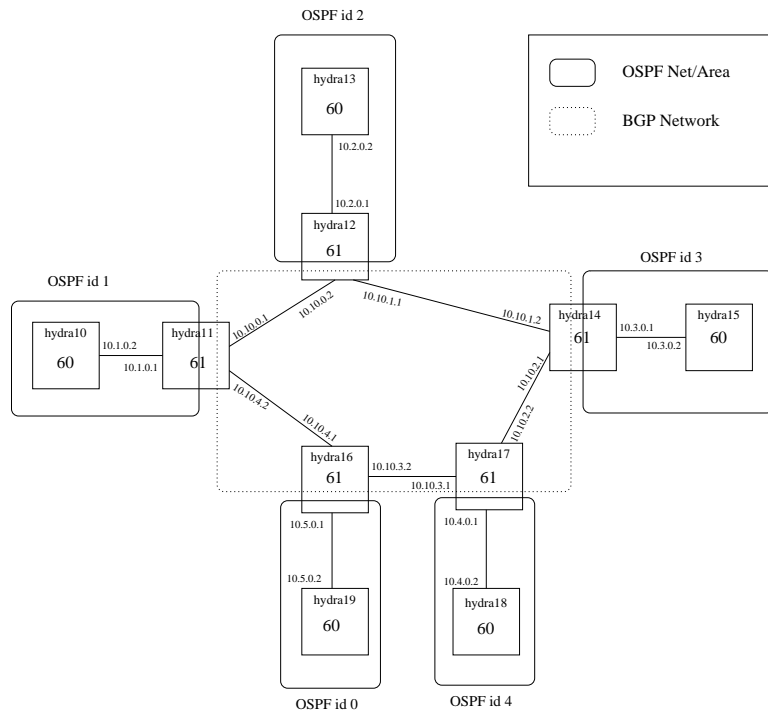
Tips

Here are some tips that may help make developing this a bit easier for you.

- Having linked groups together before, I can definitely recommend having an easy way to turn off your checksum checks for each level (IP, OSPF, BGP).
- Note that the header contains shorts and ints, you will need to do your byte-conversions accordingly.
- Having an elegant way to configure these networks in your config file will make your life much better, it can be quite annoying to test otherwise.

Testing

Here is a layout that you might find useful for testing your setup by yourself. You can check to make sure that routes update properly when something goes down, that the shortest available route is taken across the network. This is good to find the big bugs, but you will definitely need to test with other groups as well, to make sure you are using the protocol portably.



I am also planning on setting up a network with an interface for each group, which would let you test with my setup, for better or worse. Stay tuned to the mailing list.