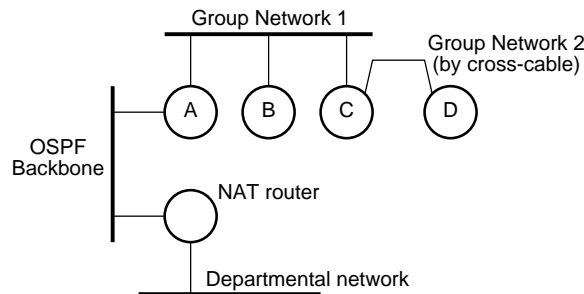


CSIS0234B Computer and Data Communication (Class B)

Tutorial 11

The multicast routing daemon

We will use the result of the last tutorial to examine how multicasting works over multicast routers. They are placed to the computer, under the home directory of the `root` user. The network is configured in the following form.



Each group will work with 4 computers. One of them (A) acts as OSPF router and exchange (unicast) routes with the routers of other groups (and with the NAT router). One of them (C) acts as an internal router with forward information between the two networks of the group. A has the information about C, so C needs not be an OSPF router. We use `mouted` on C so that B and D can exchange multicast routes. Note that at the end of the tutorial we will not recover the content of A, so please don't modify any files in it.

1. Configure the network interfaces for B, C and D. The IP addresses of each computer can be found in the white board. B and C should use A as default route (ignoring the problem that packets from B to D goes through an extra-hop), D should use C as default route. Setup C to forward packets (`echo 1 to /proc/sys/net/ipv4/ip_forward`), and stop the firewall (`ipchains`) at C.
2. Compile the programs of the last tutorial. Start `ethereal` to capture all packets of `eth0`. Try to send multicast packets from B, C and D to the group 239.255.0.x (x is the group number) when all of them are receiving packets. Observe that D never receive a multicast packet sent by B and C.
3. Look at the packets captured by Ethereal. Apart from ARP, DNS and the packet that you intentionally send to the network, what other packet you can find? These messages are the way the hosts tell any multicast router that it is joining a group.
4. Now run `mouted` at C, without any configuration. Confirm that it is running using `ps -x`. Try the above experiment again. Can host D sends multicasts to other hosts?
5. Capture the packet that you sent in `ethereal`, and find the TTL set for the IP packet. Why in the last step you cannot forward a packet?
6. Modify the sender program so that it sets the TTL to 2 before sending the packet. Retry the experiment, and confirm that the packet is indeed received.
7. Now send a `SIGUSR1` signal to the `mouted` process (with `kill -USR1 pid` where `pid` is the process id of `mouted`). Look at the file `/var/tmp/mouted.dump` to find the current routing information computed by the distance vector algorithm. Read the man page of `mouted`, in particular the `EXAMPLES` section at the end of the man page, to understand the information presented.

8. Repeat the last step, this time sending a SIGUSR2 signal, and look at the file `/var/tmp/mrouted.cache`. Also, try to send a multicast message with the other side **not** listening. Is there any pruning recorded in the cache file? Why?
9. Build a tunnel with the group opposite to you, and stop and restart `mrouted`. Remember that apart from the `mroute.conf` configuration, you must load the `ipip` module before restarting `mrouted`. It should have a threshold of 1 without setting any boundary. Can multicast of your network can reach to the other side and vice versa? Try to change the TTL value to an even larger value (say 10) and try again.
10. Repeat step 7 and 8. Note that with more than one `mrouted` communicating, pruning starts to be recorded in the cache file. In a larger network with many groups, `mrouted` must keep a lot of information about the groups.
11. Modify the tunnel so that it is a boundary for the range `239.255.0.0/8`. Restart `mrouted`, and try the experiment again. You should not be able to send packets to the other side, which should be unsurprising. What is interesting is the cache that is dumped when SIGUSR2 is sent to `mrouted`. Observe whether each side has pruning information about a group in the range, when a packet of such address is sent.