

CSIS0234B Computer and Communication Networks (Class B)

Reading for Tutorial 11

Inter-network Multicasting and mouted

In the last tutorial, we focus on multicasting within a single network, where the multicast packet is broadcasted to all hosts and filtered by the Ethernet cards and the OS kernels of the hosts. In the tutorial this week, we will study how multicast packets can work its way through multiple networks.

While it is possible to have multicast routing capabilities within regular routers¹, it is not typical. Indeed, none of the Internet routing protocol we in popular use (RIP, OSPF, BGP) is capable of multicast routing. Since most current routers do not support multicasting, a conventional method of routing (i.e., every router talks only to physical neighbours to exchange and forward routing information) has limited success.

There are multiple multicast routing protocols, and they disagree even on principles. In order to get some concrete experiences about multicast routing, we will look at an early but successful protocol, called DVMRP (Distance Vector Multicast Routing Protocol).

1. How far should multicast reach?

Due to obvious bandwidth constraints, it is impossible for every router in the world to listen to every multicast that could ever be generated. So multicast routers do not route all multicast traffic. The “scope” of multicasts are determined in two ways. The old solution is to “overload” the TTL field of the IP header: the **sender** determines how far should the packet go, by setting the TTL field of the IP header, using an IP option called `IP_MULTICAST_TTL`. So the sender program will run the following:

```
int ttl = ...;
setsockopt(sockfd, SOL_IP, IP_MULTICAST_TTL, &ttl, sizeof(ttl));
```

The TTL value is decremented by 1 everytime a router is visited. But it also carries the following additional meaning:

- $TTL < 32$: *Site local*. The packet is restricted to the same organization. The smaller number means less computers can receive the packet. In particular, if $TTL=0$, it will never be sent over a network interface. Such multicasts are “node local”. If $TTL=1$, the packet is restricted to the same subnet, and won’t be forwarded by an IP router. Such multicasts are “link local”.
- $32 \leq TTL < 64$, $64 \leq TTL < 128$: *Regional local, continental local*. The packet is restricted to the same region or continent. Seldom useful.
- $TTL \geq 64$: *Global* and unrestricted in scope.

It proves very hard to use such TTL reliably, and the resulting schemes are hard to understand. Such use is now discouraged. Instead, RFC 2365 breaks the class D address space to “sub-spaces”, each have their own scope. E.g.,:

- **Statically allocated addresses**: most addresses are centrally allocated to avoid multiple ap-

¹See, e.g., Multicast Extension to OSPF (MOSPF), RFC 1584.

plications or organization to use the address in incompatible ways. In particular, all class D addresses except 239.0.0.0/8 is centrally allocated, and the current allocation can be found in IANA web page. E.g., 224.0.0.1 always means “all hosts supporting multicasting”, while 224.0.1.1 is allocated for use with the NTP protocol to synchronize clocks of computers.

- Within statically allocated addresses, the addresses 224.0.0.0--224.0.0.255 have **link local** scope. They are for control within a physical network, so multicast routers shouldn't route packets for these groups.
- **Administratively scoped addresses:** the addresses within 239.0.0.0/16 are “administratively scoped”, i.e., for privately allocated addresses that depends on administrators of organizations to reduce the possibility of having incompatible uses of the same addresses. Within the range,
 - The addresses 239.255.0.0/16 have **site local scope**, i.e., packets in such groups. It mean a scope that is not further divisible (e.g., a site within an organization). The standard also suggests that if it becomes insufficient, the addresses 239.252.0.0/16 to 239.254.0.0/16 (currently “reserved”) would become site local scoped.
 - The addresses 239.192.0.0/16 to 239.251.0.0/16 have **organization local scope**. Organizations should allocate cross-site sub-ranges in this group of addresses. Administrators can implement different scopes for each of such addresses, by configuring multicast routers not to forward packets for some of such addresses. (So the router becomes a **boundary** for such group.)

2. DVMRP

Any multicasting protocol has two basic decisions: how to perform inter-network broadcasting, and how to limit it to interested parties and those routers which are needed to forward it to the interested parties. One of the earliest attempts to support inter-network multicasting is the Distance Vector Multicast Routing Protocol, DVMRP (RFC 1075). In DVMRP, inter-network broadcasting is done using reverse path broadcasting (RPB). To limit the broadcasting, pruning is used.

Pruning is a passive approach to limiting broadcasts. DVMRP routers listens to connected hosts about their joining and leaving of a group, so each of them keeps accurate information about which groups its hosts are interested in. Normally it does not forward such group membership information to neighbouring routers. As a result, when somebody wants to send a packet to a particular group, a full broadcast is performed using RPB. When a router receives such a packet, it checks whether any host is interested, and whether there is any router that it can forward the packet to. If neither is the case, a **prune message** is sent to the originator of the message. The originator will respond by stop sending packets of the same source and group addresses to it. The originator might find that all its links has pruned a group, and thus further prunes towards the ultimate sender. If a router which prunes a group later finds that it wants the group again, a **graft message** is sent to undo the prune message.

What makes DVMRP successful is perhaps the provision it provides for networks without multicasting routers. DVMRP allows the use of **tunnels** to connect parts (“islands”) of the Internet which have multicast capability. To the RPB algorithm, the tunnel can be treated as a link that connects the two end-points, although the end-points are not directly connected. Instead, when a multicast packet is received in one side of the tunnel, it is **encapsulated** in a **unicast** packet, and

sent to the other end using the unicast infrastructure. The other side will then extract the multicast packet and send it through its own network. In this way `mrouted` can work successfully across the global Internet, which probably explains the success it achieves.

3. Mrouted: a multicast daemon

The most popular implementation of DVMRP is a Unix program `mrouted` developed by Stanford. Unluckily, it is not completely free software, so most distributions do not install it, and it must be separately compiled and installed. The program is designed for BSD, and requires a patch to allow it to work on Linux. In Linux, the encapsulation required for a tunnel is done by a kernel module `ipip`. This module must be loaded before a tunnel can be used by `mrouted`.

To start `mrouted`, simply run the program. To stop it, just kill it. By default, `mrouted` listens to all broadcast network interfaces, and forwards messages from each interface to all others without using any tunnel. It will only succeed if there are at least 2 interfaces: otherwise `mrouted` has no interface to forward packets, and it refuses to start. The `/etc/mrouted.conf` file can be used to modify these defaults. A `phyint` (physical interface) section determines which interfaces to turn on and their operational parameters. The operational parameters include which ranges of multicast addresses not to forward to that interface (default none), a metric parameter which allows you to slightly affect the routing by increasing the cost of a link (default 1), and a threshold parameter which tells that if a packet has TTL less than the threshold it should not be forwarded (default 1). E.g.,

```
phyint eth0 boundary 239.255.0.0/16 threshold 16 metric 2
```

says that for a packet to be forwarded to the `eth0` interface, it must not be an address in the range from `239.255.0.0` to `239.255.255.255`; and it must have $TTL \geq 16$. It has a metric of 2, meaning that it should be considered to have double cost when the distance vector algorithm is executed.

Tunnels are similar, but apart from an interface, one has to specify the address of the other end of the tunnel. A tunnel can operate only if both sides configure the tunnel correctly. E.g., the following specifies that the other end is `192.168.6.3`.

```
tunnel eth0 192.168.6.3 boundary 239.255.0.0/16 threshold 16
```

The file can also contain other options, which are described in more details in the manpage. The sample config file is very readable, though, so you might not really need to consult the man page to understand what each option means.

On the other hand, the interface of `mrouted` for inspection of its internal state is quite crude. In particular, you can send `mrouted` a signal `SIGUSR1` or `SIGUSR2`, using `kill -USRn <pid>`. The former puts its current routing table to the file `/var/tmp/mrouted.dump`; the latter puts its current pruning information to the file `/var/tmp/mrouted.cache`. See the man page of `mrouted` if you want to see an example.