Network Working Group
Internet-Draft

Ross Finlayson
LIVE.COM
Radia Perlman
Sun Microsystems
Doron Rajwan
Bandwiz

Expire in six months

2001.02.19

Accelerating the Deployment of Multicast Using Automatic Tunneling

<draft-finlayson-mboned-autotunneling-00.txt>

Status of this Memo

Abstract

Many Internet users currently cannot participate in wide-area IP multicast sessions, because their first-hop routers (or beyond) do not support IP multicast routing. We describe an application level (UDP-based) tunneling mechanism that allows non-multicast-connected users - with no modification to their operating systems - to automatically receive a large class of multicast sessions, pending the deployment of multicast in their upstream routers.

1. Introduction

Many Internet users remain unable to participate in wide-area multicast sessions, because their first-hop router (e.g., a DSL router or a dialup 'portmaster') does not support IP multicast routing, and/or because their ISP is unwilling or unable to provide multicast connectivity.

A secondary issue is that most users' operating systems currently do not support IGMP version 3 [2] - the version of IGMP that is used, at the 'leaves' of the network, to implement a special form of IP multicast called "Source-Specific Multicast" (SSM) [3]. (SSM is expected to become widely used for a large class of multicast sessions in which all multicast data originates from a single source node.)

To accelerate the deployment and adoption of IP multicast, we wish to provide a mechanism that allows such users to participate in multicast sessions, even before their router(s) and/or operating systems become upgraded to support native multicast.

Fortunately, such a mechanism already exists: UDP multicast tunneling. In this mechanism, UDP multicast packets are tunneled within UDP unicast packets. Such a tunnel is implemented by two tunneling agents (the endpoints of the tunnel). One agent (the tunnel 'slave') resides on a node in the multicast-connected portion of the Internet. The other agent (the tunnel 'master') runs on the user's computer - either embedded within the end-user application (such as an audio/video tool), or else running as a separate application. Upon command from the master, the slave receives all multicast packets that are addressed to the desired multicast group and UDP port (and, in the case of SSM, originating from the desired IP source address). These multicast packets are encapsulated within UDP unicast packets and sent to the master (i.e., the user's computer), which then decapsulates them and delivers them (as multicast data) to the end application. (Also, for non-SSM sessions, multicast data could be delivered over the tunnel in the opposite direction as well.)

The benefit of this approach is that the user's tunneling agent can run at the application level, without requiring any modification to the host operating system. This means, however, that only UDP multicast packets can be tunneled - not 'raw IP' multicast packets. Fortunately, however, most (if not all) end user multicast applications use UDP.

For the purposes of this document, we also impose two additional restrictions: First, the proposal described here is only for SSM sessions. (Restricting multicast tunneling to SSM sessions makes it easier to prevent data loops, as well as making it easier for a receiver to detect when tunneling is no longer necessary.) Second, this proposal is only for IPv4. (In IPv6, native multicast routing is ubiquitous, so tunneling is not needed.)

Our proposal assumes the use of a UDP multicast tunneling protocol such as UMTP [4]. This particular protocol has been used for several years now to tunnel UDP multicast sessions (including SSM sessions)

over non-multicast-capable routers (and across firewalls). The protocol includes both control packets (e.g., "JOIN_GROUP", "LEAVE_GROUP") and data packets, all using a single UDP port. For each multicast session that the tunnel master wishes to have tunneled, the master sends - to the tunnel slave - a JOIN_GROUP command. This is done periodically, as a 'keep-alive' (and thus the tunnel slave maintains 'soft state' for each such session). The reader is referred to [4] for more information.

2. An Improvement: Automatic Tunneling, using Router Interception

The biggest problem with UDP multicast tunneling, as described above, is that before the end user's computer can request that a multicast session be tunneled, it needs to know which node to use as the remote (slave) tunnel endpoint. Either the end user has to know and enter this manually, or else some separate (unspecified) lookup or discovery mechanism must be used to determine the tunnel endpoint. In any case, there's no inherent guarantee that whatever tunnel endpoint gets used will be in an optimal place in the networking topology.

This problem can be overcome by if *multicast routers* can also act as tunnel endpoints. The user (master) end of the tunnel can send its JOIN_GROUP requests not to an explicit tunnel endpoint, but instead addressed to the SSM multicast source. The first multicast-capable router in the path from the user's computer to the SSM source can intercept this request, and - from then on - act as a UMTP tunnel slave for this master.

Thus, this mechanism automatically locates a tunnel endpoint, and one that is in an optimal location: the first multicast-capable router on the path between the user's computer and the the SSM source node. Furthermore, as additional routers below this become multicast-enabled, they, too, will automatically take over the tunneling duty. Should *all* routers on the path between the user's computer and the SSM source become multicast enabled, the user's computer will start seeing incoming native multicast packets from the SSM source node. When it sees such packets, it can shut down the tunnel, knowing that native multicast routing is now in place.

For this mechanism to work, SSM-multicast-capable (i.e., PIM-SSM) routers should also be capable of acting as UMTP tunnel slaves, and be able to intercept UMTP requests coming from below, as well as receiving incoming data packets from above (i.e., from the SSM source) and retransmitting them to the appropriate master(s) as tunneled UMTP "DATA" commands. A single, well-known UDP port number (assigned by IANA) would be used for UMTP; the routers would detect UDP packets addressed to this port as UMTP commands that need to be handled especially.

Ideally, *all* PIM-SSM routers would also be able to act as UMTP tunnel slaves. However, this mechanism would still work - albeit

with a less-than-optimal tunneling topology - even if only *some* of them have this capability.

The worst-case scenario is that *no* routers in the path between the user's computer and the SSM source have this capability. To handle this case, the SSM source node may also contain implement a UMTP slave implementation of its own. This allows the server to automatically stream to its multicast-connected customers via native multicast, and to its non-multicast-connected customers via tunneled multicast.

3. Behavior of multicast receivers

To join a SSM session (S,G) (on UDP port P), the receiving node would:

1/ do an IGMPv3 (S,G) join (if it can!), *and*

2/ (perhaps after a short delay) act as UMTP tunnel master, by periodically sending - to the desired SSM source - a UMTP command: JOIN_GROUP(S,G,P). This command is sent as a UDP packet addressed to the desired SSM source S.

(By doing a IGMPv3 join as well as sending a UMTP JOIN_GROUP command, we allow for the possibility that the receiver's upstream routers are multicast-enabled, but do *not* support UMTP.)

Similarly, to leave a SSM session, the receiving node would do both an IGMPv3 leave (if it can!), *and* send a UMTP LEAVE_GROUP command (again, addressed to the SSM source S).

If the receiver ever receives a native (i.e., non-tunneled) multicast packet from the SSM source S, it knows that tunneling is no longer necessary (for this source, at least). It then removes the tunnel, by sending a UMTP "TEAR_DOWN" command (addressed to S), and stops sending periodic JOIN_GROUP commands. (Thus, even if the TEAR_DOWN command gets lost in transit, the tunneling would later time out anyway. In the meantime, there would be some duplicate packets - tunneled and native.)

As the receiver receives encapsulated multicast packets across the tunnel (as UMTP DATA commands), it decapsulates them and delivers them to their intended local recipients - e.g., by re-multicasting them locally (to the appropriate UDP port). Note, however, that this means that the IP source address that the the ultimate receiving application(s) see will *not* be that of the original SSM source. Instead, the source address will be that of the local machine, and so the receiving applications need to made be aware of this.

Probably the best place to deal with this is in whatever 'wrapper' software is used to launch the application. This wrapper software can do the following:

1/ act as a UMTP tunnel master and send a JOIN_GROUP command for the desired SSM source, and then

2/ launch the application, but telling it that the *local node*

is to be the SSM source.

Another possible approach is to integrate the UMTP tunneling master implementation within the application itself (rather than running UMTP a separate application).

4. Behavior of routers

Non multicast-capable routers will, of course, simply forward UMTP packets (whether control or data) just like any other UDP packets.

Multicast-capable routers, however, should intercept all UDP packets that are addressed to the special port number for UMTP, and act as a UMTP slave server to process such packets. (See [4] for details.) The router performs the role of a UMTP slave in semantically exactly the same way as if UMTP were running on top of the router as a user-level application.

In particular, an incoming JOIN_GROUP command would be handled by (effectively) joining the specified SSM group (S,G). The UMTP implementation would then receive any subsequent (native) multicast packets for this group, and deliver these packets down the tunnel (i.e., encapsulated in UMTP DATA commands) to each UMTP recipient.

Note that - apart from this - the underlying router would handle these incoming native multicast packets in exactly the same way as usual (including continuing to forward them downstream if there are any native receivers). While the router's UMTP implementation receives and processes all native multicast packets that have a (S,G) that it's interested in tunneling, the router *intercepts* only UMTP commands (which are identified by UDP port number). Thus, only the handling of incoming UDP commands needs to be in the router's 'fast path'.

5. Behavior of multicast senders

As noted earlier, the sending node may also include its own UMTP slave implementation - allowing it to send data to non-multicast-connected recipients via tunneling. This is assuming, of course, that the sender has sufficient bandwidth to support these tunnels. (If necessary, the sender's UMTP implementation can limit the number of tunnels that get created.)

Alternatively, a sender without any multicast connectivity could set up just a single UMTP tunnel - to a second, multicast-connected node that would then do the actual multicasting. (This tunnel would be set up explicitly rather than automatically.) The drawback of this approach is that the actual SSM source would then be the second node rather than the first; prospective receivers would need to be made aware of this.

6. References

[1] Bradner, S.
"Key words for use in RFCs to Indicate Requirement Levels"
RFC 2119, March 1997.

[2] Cain, B., Deering, S., Fenner, B., Kouvelas, I., Thyagarajan, A.
"Internet Group Management Protocol, Version 3"
Work-in-Progress, Internet-Draft "draft-ietf-idmr-igmp-v3-06.txt,.ps"
January, 2001.

[3] Holbrook, H., Cain, B.
"Source-Specific Multicast for IP"
Work-in-Progress, Internet-Draft "draft-holbrook-ssm-arch-01.txt"
November, 2000.

[3] Finlayson, R.
"The UDP Multicast Tunneling Protocol"
Work-in-Progress, Internet-Draft "draft-finlayson-umtp-05.txt"
February, 2001.