Multicast

1-many or many-many efficient? Many applications eg, video streaming
May be replicate data along required paths

![Diagram of multicast routing]

- Bottleneck links (could avoid redundant transmissions)
- This is one way of doing it (at the routers)

Multicast

Fundamental problem \(\Rightarrow\) building trees \(\Rightarrow\) rooted at
regular unicast routing
builds trees rooted at the
destination

1-many: TV, software distribution \(\Rightarrow\) need reliable (patches) \(\Rightarrow\) multicast

When you join you get live feed diff. people start at different times
(harder problem)

\(\leftrightarrow\) Carousel (encode bits in a circle, keep transmitting)
It is possible to make software distribution look like a TV feed, but still need reliability.

Many-to-many: Conferencing may make sense.

Resource Discovery

224.0.0.0 to 239.255.255.255

Not topologically dependent

Lot of local reuse (scoped TTLs)

Administrative scoping -> 32 -> will not escape border router

IP multicast -> 1 million people

Best bet is to just use cable!

1000 -> 100,000 users] multicast makes sense

In reality, IP multicast is used within organizations, not well across ISPs.

Many concepts used in wireless sensor networks/software updates in sensor n/w

1) Membership - who gets to join/send?
   How do you name a multicast group?

2) Routing

3) Reliable Transport
For IP multicast, open vs closed membership is: does sender decide? or is receiver join allowed, followed by authorization?

If multicast uses receiver join, prevent snooping at higher layer.

Receiver based or sender based?

Receivers decide: senders decide

If multicast uses this, simply send to sender. No multicast name, need not know.

Who can send? In IP multicast, anybody can send to a multicast group (bad idea; needs to be filtered out)

Any vs 1 sender?

Can make other choices when using application-level multicast.

2) Routing: build trees rooted at sender. But in IP multicast, need one tree for all senders. "Pain!"
A Bootstrap at receiver

Receiver joins multicast group \( G \).

Join \((G)\) needs to be sent somewhere.


\[ \text{Internet} \]

\[ \text{node} \quad \text{local router} \]

How do you know whom to send to? Simply sends to local router.

What does \( R \) do? \( R \) builds up state.

Nodes could leave arbitrarily \( \Rightarrow R \) maintains only soft state.

State expires periodically.

\( R \) sends message asking if anyone is interested in \( G \). Doesn't scale.

\[ \text{slotted + Damping} \]

Enough if one is interested in local LAN.

Every node picks random \# \( I \sim u(0, d) \)

\( \text{Trickle} \)

Wait \( (r) \), then suppresses if you hear transmission. The respond.

\( \Rightarrow \text{minimize} \# \text{of duplicate responses} \)

Need to make \( d \) small. Fundamental tradeoff in protocol.
This is called IGMP (Internet Group Management Protocol).

Expected latency depends on $d$ from $[0, d]$.

- If there are $n$ receivers, then $d = \frac{n+1}{n}$.

If # of duplicates depends on time for broadcast (ratio of $d$ to that) on a LAN.

What happens when a node fails? Use querying protocol.

What does $R$ do? Hard question.

Ideally, you want to send it to the source.

Solutions

1. Let $R$ flood $\text{join}(G)$ to everybody on the internet. The moment someone sees a source interested in sending to $G$, flood back to large # of receivers.

2. Senders periodically flood diffusion (Good soln. in sensor networks).

Less routers "prunes" back information.

- Soft state times out.
Directory system → Rendezvous point

Known set of nodes that act as Rendezvous points (RP)

Send joins, send pks to RP

Problem: too much load on RP? Single bottleneck

You can distribute RP

IP multicast → use RPs

Then receivers decide

Start sending join messages to sender directly

For bootstrapping

Routers create sender rooted tree

Scales better

Routers implement this at high costs

Not clear if there are real benefits over unicast

Reasons why multicast is not deployed widely:

1) Pricing model → how do you compensate ISPs in the middle? Works if you go through at most 2-3 ASs

End node can

2) Creating state in N-W → security hole

3) Inefficient at very high speeds

Much worse than routing state

State overflows

Unpredictable behavior

So people turn it off (DoS attacks)
Making copies is expensive for routers. Is matching valid anymore? Horribly complicated. One possibility: Crossbar doesn't let you replicate here.

Concentrators can do it but expensive, complicated.

Application Level Multicast

Don't ask routers to do anything, simply use an overlay network.

Make multicast tree among nodes? Akamai, CDN.

\[ \text{URL} \Rightarrow \text{multicast stream} \]

Local receivers connect to multicast servers

Source sends to servers

Build a tree. Rendezvous problem is easier. Helps to solve most of the problem though there is some duplication.
once you're inside ISP, you can actually use IP multicast to do this] don't even bother
pricing model is also clearer CDN offered by companies
Eliminates problems 1, 2, 3
2: create state only after you're allowed to join CDN

[very good for ISPs]
(can be open or closed)

most of them are only 1-sender, not any-sender
Any-to-Any DHTs

APP level most + IP multicast to leaves [good solution]

one thing that's important - build decent performance tree (lots of proposals)

connect (root) measure -> 1 Mbps
connect (children) -> 2 Mbps

[keep probing + current mechanism]