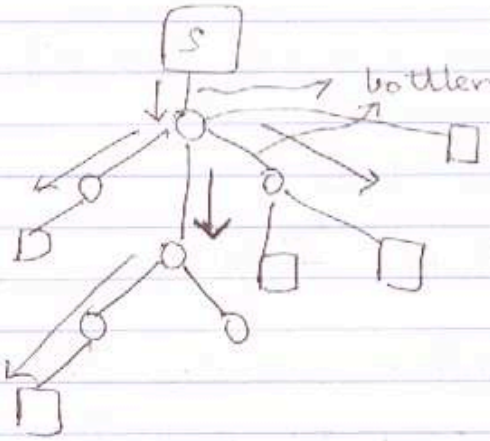
6.829 - Computer Networks - Contd.

Multicast

1-many or many-many] efficient } Many applications eg. video streaming

→ Maybe replicate data along required paths



bottleneck links (could avoid redundant trx missions)

↳ this is one way of doing it (at the routers)

⇓

Multicast

Fundamental problem ⇒ building trees } rooted at the sources

regular unicast routing builds trees rooted at the destination

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

↓ when you join you get live feed

↓ diff. people start at different times (harder problem)

↳ Carousel (encode bits in a circle, keep transmitting)

⇒ possible to make s/w distribution look like TV feed
(but still need reliability)

many - many : Conferencing,] may make sense

Resource Discovery

224.0.0.0 – 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!

1,000 → 100,000 users] multicast makes sense

In reality, IP multicast is used within organizations
→ doesn't work across ISPs too well

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 gp?

2) Routing

3) Reliable Transport

For IP multicast, open vs closed membership is
↳ does sender decide? or ~~does~~ a receiver join allowed, followed by authorization?

IP multicast ⇒ uses receiver join
prevent snooping at higher layer

→ Receiver based or sender based?

Receivers decide
↓
IP multicast uses this

senders decide
↓
simply send to multicast name
} sender 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?

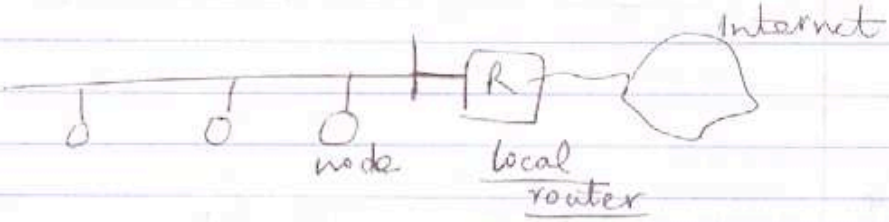
can make other choices when using application-level multicast

2) Routing → build trees rooted at sender

but in IP multicast, need one tree for all senders
↳ pain!

* Bootstrap at receiver } Receiver joins multicast group G1

join (G1) needs to be sent somewhere



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 ⇒ R maintains only soft state !

State expires periodically

R sends message asking if anyone is interested in G1 ⇒ doesn't scale

↓

slotting + Damping] Enough if one is interested in local LAN.

Like Trickle!

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

wait (r), then suppresses if you hear transmissio

else respond

↳ minimize # of duplicate responses

↓

Need to make d small] Fundamental tradeoff } similar to MAC protocol

This is called IGMP (Internet Group Management Protocol)

Expected latency depends on d } Minimum from {0, d}

$\sim \boxed{\frac{d}{d n + 1}}$ if there are n receivers

# of duplicates depends on time for broadcast (ratio of d to that)

↓ on a LAN

What happens when a node fails?] use querying protocol

(2) What does R do? Hard question.

Ideally, you want to send it to the source.

Solutions

1) Let R flood join (G) to everybody on the internet
The Moment someone sees a source interested in sending to G, flow back
↳ large # of receivers

2) Senders periodically flood } diffusion { Good soln. in sensor networks
↳ routers "prunes" back information
→ soft state times out.

3) Directory system → Rendezvous point

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

send joins, sender pkts to RP

Problem too much load on RP? } single bottleneck

You can distribute RP

IP multicast → uses 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 → doesn't work — how do you compensate ISPs in the middle? Works if you go through at most 2-3 ASs

2) End hosts can create state in n/w → security hole → Much worse than routing state

state overflows

↳ unpredictable behaviour

→ so people turn it off (DoS attacks)

3) Inefficient at very high speeds

Making copies is expensive for routers → Is matching valid any more?

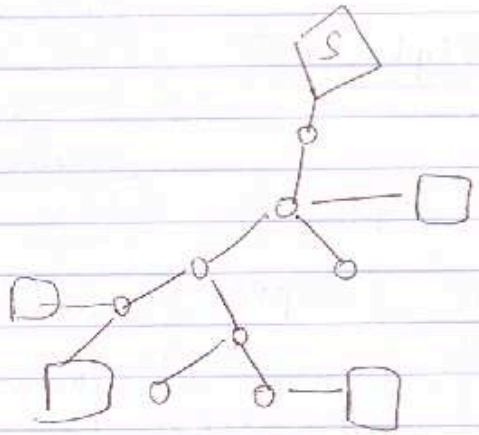
horribly complicated → one possibility: daisy chain → Cross bar doesn't let you replicate here

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



URL ⇒ multicast stream

Local receivers connects to multicast servers

Source sends to servers

Build a tree → Rendezvous problem is easier

Interesting solution → 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] CDNs offered by companies

Eliminates problems ①, ②, ③

② → create state only after you're [allowed to join CDN

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

cricket matches

CNN video feeds (sometimes)] most of them are only 1-sender, not any-sender

Any-to-Any] DHTs
Splitstream → read this!

App level mcast + IP multicast to leaves] good hybrid solution

one thing that's important → [build decent performance trees (lots of proposals)

can use greedy algorithm { depends on fan/in fan/out

connect (root) measure → 1 Mbps
connect (children) → 2 Mbps] Local satisfaction

Overcast

syncing

may be hard, or may not be { keep doing periodically

NP Complete] Steiner tree problems

↳ probing + current tree] mechanism