IP ANYCAST AND MULTICAST

READING: SECTION 4.4

COS 461: Computer Networks
Spring 2009 (MW 1:30-2:50 in COS 105)

Mike Freedman
Teaching Assistants: Wyatt Lloyd and Jeff Terrace
http://www.cs.princeton.edu/courses/archive/spring09/cos461/
Outline today

• IP Anycast

• Multicast protocols
  – IP Multicast and IGMP
  – SRM (Scalable Reliable Multicast)
  – PGM (Pragmatic General Multicast)
  – Bimodal multicast
  – Gossiping
Limitations of DNS-based failover

• Failover/load balancing via multiple A records

```plaintext
;; ANSWER SECTION:
www.cnn.com. 300 IN A 157.166.255.19
www.cnn.com. 300 IN A 157.166.224.25
www.cnn.com. 300 IN A 157.166.226.26
www.cnn.com. 300 IN A 157.166.255.18
```

• If server fails, service unavailable for TTL
  – Very low TTL: Extra load on DNS
  – Anyway, browsers cache DNS mappings 😞

• What if root NS fails? All DNS queries take > 3s?
Motivation for IP anycast

• Failure problem: client has resolved IP address
  – What if IP address can represent many servers?

• Load-balancing/failover via IP addr, rather than DNS

• IP anycast is simple reuse of existing protocols
  – Multiple instances of a service share same IP address
  – Each instance announces IP address / prefix in BGP / IGP
  – Routing infrastructure directs packets to nearest instance of the service
    • Can use same selection criteria as installing routes in the FIB
  – No special capabilities in servers, clients, or network
IP anycast in action

- Client
- Router 1
- Router 2
- Router 3
- Router 4
- Server Instance A
- Server Instance B

Announce 10.0.0.1/32

192.168.0.1 -> Router 2
192.168.0.2 -> Router 3
10.0.0.1 -> Router 2
10.0.0.1 -> Router 4

10.0.0.1 -> Server Instance A
10.0.0.1 -> Server Instance B
IP anycast in action

Routing Table from Router 1:

<table>
<thead>
<tr>
<th>Destination</th>
<th>Mask</th>
<th>Next-Hop</th>
<th>Distance</th>
</tr>
</thead>
<tbody>
<tr>
<td>192.168.0.0</td>
<td>/29</td>
<td>127.0.0.1</td>
<td>0</td>
</tr>
<tr>
<td>10.0.0.1</td>
<td>/32</td>
<td>192.168.0.1</td>
<td>1</td>
</tr>
<tr>
<td>10.0.0.1</td>
<td>/32</td>
<td>192.168.0.2</td>
<td>2</td>
</tr>
</tbody>
</table>
IP anycast in action

DNS lookup for http://www.server.com/ produces a single answer:

www.server.com. IN A 10.0.0.1
IP anycast in action

Routing Table from Router 1:

<table>
<thead>
<tr>
<th>Destination</th>
<th>Mask</th>
<th>Next-Hop</th>
<th>Distance</th>
</tr>
</thead>
<tbody>
<tr>
<td>192.168.0.0</td>
<td>/29</td>
<td>127.0.0.1</td>
<td>0</td>
</tr>
<tr>
<td>10.0.0.1</td>
<td>/32</td>
<td>192.168.0.1</td>
<td>1</td>
</tr>
<tr>
<td>10.0.0.1</td>
<td>/32</td>
<td>192.168.0.2</td>
<td>2</td>
</tr>
</tbody>
</table>
IP anycast in action

Routing Table from Router 1:

<table>
<thead>
<tr>
<th>Destination</th>
<th>Mask</th>
<th>Next-Hop</th>
<th>Distance</th>
</tr>
</thead>
<tbody>
<tr>
<td>192.168.0.0</td>
<td>/29</td>
<td>127.0.0.1</td>
<td>0</td>
</tr>
<tr>
<td>10.0.0.1</td>
<td>/32</td>
<td>192.168.0.1</td>
<td>1</td>
</tr>
<tr>
<td>10.0.0.1</td>
<td>/32</td>
<td>192.168.0.2</td>
<td>2</td>
</tr>
</tbody>
</table>
IP anycast in action

Routing Table from Router 1:

<table>
<thead>
<tr>
<th>Destination</th>
<th>Mask</th>
<th>Next-Hop</th>
<th>Distance</th>
</tr>
</thead>
<tbody>
<tr>
<td>192.168.0.0</td>
<td>/29</td>
<td>127.0.0.1</td>
<td>0</td>
</tr>
<tr>
<td>10.0.0.1</td>
<td>/32</td>
<td>192.168.0.1</td>
<td>1</td>
</tr>
<tr>
<td>10.0.0.1</td>
<td>/32</td>
<td>192.168.0.2</td>
<td>2</td>
</tr>
</tbody>
</table>
IP anycast in action

From client/router perspective, topology could as well be:

Routing Table from Router 1:

<table>
<thead>
<tr>
<th>Destination</th>
<th>Mask</th>
<th>Next-Hop</th>
<th>Distance</th>
</tr>
</thead>
<tbody>
<tr>
<td>192.168.0.0</td>
<td>/29</td>
<td>127.0.0.1</td>
<td>0</td>
</tr>
<tr>
<td>10.0.0.1</td>
<td>/32</td>
<td>192.168.0.1</td>
<td>1</td>
</tr>
<tr>
<td>10.0.0.1</td>
<td>/32</td>
<td>192.168.0.2</td>
<td>2</td>
</tr>
</tbody>
</table>
Downsides of IP anycast

• Many Tier-1 ISPs ingress filter prefixes > /24
  – Publish a /24 to get a “single” anycasted address: Poor utilization

• Scales poorly with the # anycast groups
  – Each group needs entry in global routing table

• Not trivial to deploy
  – Obtain an IP prefix and AS number; speak BGP

• Subject to the limitations of IP routing
  – No notion of load or other application-layer metrics
  – Convergence time can be slow (as BGP or IGP convergence)

• Failover doesn’t really work with TCP
  – TCP is stateful; other server instances will just respond with RSTs
  – Anycast may react to network changes, even though server online

• Root name servers (UDP) are anycasted, little else
Multicast protocols
Multicasting messages

• Simple application multicast: Iterated unicast
  – Client simply unicasts message to every recipient
  – **Pros:** simple to implement, no network modifications
  – **Cons:** \(O(n)\) work on sender, network

• Advanced overlay multicast
  – Build receiver-driven tree
  – **Pros:** Scalable, no network modifications
  – **Cons:** \(O(\log n)\) work on sender, network; complex to implement

• IP multicast
  – Embed receiver-driven tree in network layer
  – **Pros:** \(O(1)\) work on client, \(O(\# \text{ receivers})\) on network
  – **Cons:** requires network modifications; scalability concerns?
## Another way to slice it

<table>
<thead>
<tr>
<th></th>
<th>Best effort</th>
<th>Reliable</th>
</tr>
</thead>
<tbody>
<tr>
<td>Iterated Unicast</td>
<td>UDP-based communication</td>
<td>TCP-based communication; Atomic broadcast</td>
</tr>
<tr>
<td>Application “Trees”</td>
<td>UDP-based trees (P2P)</td>
<td>TCP-based trees; Gossiping; Bimodal multicast *</td>
</tr>
<tr>
<td>IP-layer multicast</td>
<td>IP multicast</td>
<td>SRM; PGM; NORM; Bimodal multicast *</td>
</tr>
</tbody>
</table>
Another way to slice it

<table>
<thead>
<tr>
<th></th>
<th>Best effort</th>
<th>Reliable</th>
</tr>
</thead>
<tbody>
<tr>
<td>Iterated Unicast</td>
<td>UDP-based communication</td>
<td>TCP-based communication; Atomic broadcast</td>
</tr>
<tr>
<td>Application “Trees”</td>
<td>UDP-based trees (P2P)</td>
<td>TCP-based trees; Gossiping; Bimodal multicast *</td>
</tr>
<tr>
<td>IP-layer multicast</td>
<td>IP multicast</td>
<td>SRM; PGM; NORM; Bimodal multicast *</td>
</tr>
</tbody>
</table>
IP Multicast

• Simple to use in applications
  – Multicast “group” defined by IP multicast address
    • IP multicast addresses look similar to IP unicast addr
    • 224.0.0.0 to 239.255.255.255 (RPC 3171)
      – 265 M multicast groups at most
  – Best effort delivery only
    • Sender issues single datagram to IP multicast address
    • Routers delivery packets to all subnetworks that have a receiver “belonging” to the group

• Receiver-driven membership
  – Receivers join groups by informing upstream routers
  – Internet Group Management Protocol (v3: RFC 3376)
IGMP v1

- Two types of IGMP msgs (both have IP TTL of 1)
  - Host membership query: Routers query local networks to discover which groups have members
  - Host membership report: Hosts report each group (e.g., multicast addr) to which belong, by broadcast on net interface from which query was received

- Routers maintain group membership
  - Host sends an IGMP “report” to join a group
  - Multicast routers periodically issue host membership query to determine liveness of group members
  - Note: No explicit “leave” message from clients
IGMP

• IGMP v2 added:
  – If multiple routers, one with lowest IP elected querier
  – Explicit leave messages for faster pruning
  – Group-specific query messages

• IGMP v3 added:
  – Source filtering: Join specifies multicast “only from” or “all but from” specific source addresses
IGMP

• Parameters
  – Maximum report delay: 10 sec
  – Query internal default: 125 sec
  – Time-out interval: 270 sec
    • 2 * (query interval + max delay)

• Questions
  – Is a router tracking each attached peer?
  – Should clients respond immediately to membership queries?
  – What if local networks are layer-two switched?
So far, we’ve been best-effort IP multicast...
Challenges for reliable multicast

• Ack-implosion if all destinations ack at once
• Source does not know # of destinations
• How to retransmit?
  – To all? One bad link effects entire group
  – Only where losses? Loss near sender makes retransmission as inefficient as replicated unicast
• Once size fits all?
  – Heterogeneity: receivers, links, group sizes
  – Not all multicast applications need reliability of the type provided by TCP. Some can tolerate reordering, delay, etc.
Another way to slice it

<table>
<thead>
<tr>
<th></th>
<th>Best effort</th>
<th>Reliable</th>
</tr>
</thead>
<tbody>
<tr>
<td>Iterated Unicast</td>
<td>UDP-based communication</td>
<td>TCP-based communication; Atomic broadcast</td>
</tr>
<tr>
<td>Application “Trees”</td>
<td>UDP-based trees (P2P)</td>
<td>TCP-based trees; Gossiping; Bimodal multicast *</td>
</tr>
<tr>
<td>IP-layer multicast</td>
<td>IP multicast</td>
<td>SRM; PGM; NORM; Bimodal multicast *</td>
</tr>
</tbody>
</table>
Scalable Reliable Multicast

• Receives all packets or unrecoverable data loss
• Data packets sent via IP multicast
  – ODATA includes sequence numbers
• Upon packet failure:
  – Receiver multicasts a NAK
    • ... or sends NAK to sender, who multicasts a NAK confirmation (NCF)
  – Scale through NAK suppression
    • ... if received a NAK or NCF, don’t NAK yourself
    • What do we need to do to get adequate suppression?
      – Add random delays before NAK’ing
      – But what if the multicast group grows big?
  – Repair through packet retransmission (RDATA)
    • From initial sender
    • From designated local repairer (DLR – IETF loves acronyms!)
Another way to slice it

<table>
<thead>
<tr>
<th></th>
<th>Best effort</th>
<th>Reliable</th>
</tr>
</thead>
<tbody>
<tr>
<td>Iterated Unicast</td>
<td>UDP-based communication</td>
<td>TCP-based communication; Atomic broadcast</td>
</tr>
<tr>
<td>Application “Trees”</td>
<td>UDP-based trees (P2P)</td>
<td>TCP-based trees; Gossiping; Bimodal multicast *</td>
</tr>
<tr>
<td>IP-layer multicast</td>
<td>IP multicast</td>
<td>SRM; PGM; NORM; Bimodal multicast *</td>
</tr>
</tbody>
</table>
Pragmatic General Multicast (RFC 3208)

• Similar approach as SRM: IP multicast + NAKs
  – ... but more techniques for scalability

• Hierarchy of PGM-aware network elements
  – NAK suppression: Similar to SRM
  – NAK elimination: Send at most one NAK upstream
    • Or completely handle with local repair!
  – Constrained forwarding: Repair data can be suppressed downstream if no NAK seen on that port
  – Forward-error correction: Reduce need to NAK

• Works when only sender is multicast-able
A stronger “reliability”? 

• Atomic broadcast
  – “Everybody or nobody” receives a packet
  – Clearly not guaranteed with SRM/PGM:
    • Requires consensus between receivers
    • Performance problem: One slow node hurts everybody

• Performance problems with SRM/PGM?
  – Sender spends lots of time on retransmissions as heterogeneous group increases in size
    • Local repair makes this better
“Virtual synchrony” multicast performance
Another way to slice it

<table>
<thead>
<tr>
<th></th>
<th>Best effort</th>
<th>Reliable</th>
</tr>
</thead>
<tbody>
<tr>
<td>Iterated Unicast</td>
<td>UDP-based communication</td>
<td>TCP-based communication; Atomic broadcast</td>
</tr>
<tr>
<td>Application “Trees”</td>
<td>UDP-based trees (P2P)</td>
<td>TCP-based trees; Gossiping; Bimodal multicast *</td>
</tr>
<tr>
<td>IP-layer multicast</td>
<td>IP multicast</td>
<td>SRM; PGM; NORM; Bimodal multicast *</td>
</tr>
</tbody>
</table>
Bimodal multicast

- Initially use UDP / IP multicast
Bimodal multicast

- Periodically (e.g. 100ms) each node sends *digest* describing its state to randomly-selected peer.
- The digest identifies messages; it doesn’t include them.
Recipient checks gossip digest against own history

Solicits any missing message from node that sent gossip
Recipient checks gossip digest against own history
Solicits any missing message from node that sent gossip
Processes respond to solicitations received during a round of gossip by retransmitting the requested message.
• Respond to solicitations by retransmitted requested msg
Delivery? Garbage Collection?

• Deliver a message when it is in FIFO order
  – Report an unrecoverable loss if a gap persists for so long that recovery is deemed “impractical”

• Garbage collect a message when no “healthy” process could still need a copy

• Match parameters to intended environment
Optimizations

• Retransmission for most recent multicast first
  – “Catch up quickly” to leave at most one gap in sequence

• Participants bound the amount of data they will retransmit during any given round of gossip.
  – If too much is solicited they ignore the excess requests

• Label gossip msgs with sender’s gossip round #
  – Ignore if expired round #; node probably no longer correct

• Don’t retransmit same msg twice in row to same dest
  – Retransmission may still be in transit
Optimizations

• Use UDP multicast when retransmitting a message if several processes lack a copy
  – For example, if solicited twice
  – Also, if a retransmission is received from “far away”
  – Tradeoff: excess messages versus low latency

• Use regional TTL to restrict multicast scope
Why “bimodal”? 

- There are two phases? 
- Nope; description of duals “modes” of result 

![Pbcast bimodal delivery distribution](chart)

Either sender fails... 
... or data gets through w.h.p.
Idea behind analysis

• Can use the mathematics of epidemic theory to predict reliability of the protocol
  – Assume an initial state
  – Now look at result of running B rounds of gossip: Converges exponentially quickly to atomic delivery
Another way to slice it

<table>
<thead>
<tr>
<th></th>
<th>Best effort</th>
<th>Reliable</th>
</tr>
</thead>
<tbody>
<tr>
<td>Iterated Unicast</td>
<td>UDP-based communication</td>
<td>TCP-based communication; Atomic broadcast</td>
</tr>
<tr>
<td>Application “Trees”</td>
<td>UDP-based trees (P2P)</td>
<td>TCP-based trees; Gossiping; Bimodal multicast *</td>
</tr>
<tr>
<td>IP-layer multicast</td>
<td>IP multicast</td>
<td>SRM; PGM; NORM; Bimodal multicast *</td>
</tr>
</tbody>
</table>
Epidemic algorithms via gossiping

• Assume a fixed population of size $n$

• For simplicity, assume epidemic spreads homogenously through popularly
  – Simple randomized epidemic: any one can infect any one with equal probability

• Assume that $k$ members are already infected
• Infection occurs in rounds
Probability of Infection

- Probability $P_{\text{infect}}(k,n)$ that a uninfected member is infected in a round if $k$ are already infected?

$$P_{\text{infect}}(k,n) = 1 - P (\text{nobody infects})$$
$$= 1 - (1 - 1/n)^k$$

$$E (\#\text{newly infected}) = (n-k) \cdot P_{\text{infect}}(k,n)$$

- Basically it’s a Binomial Distribution
- # rounds to infect entire population is $O(\log n)$
Two prevailing styles

• **Gossip push ("rumor mongering"):**
  – A tells B something B doesn’t know
  – Gossip for multicasting
    • Keep sending for bounded period of time: $O(\log n)$
  – Also used to compute aggregates
    • Max, min, avg easy. Sum and count more difficult.

• **Gossip pull ("anti-entropy")**
  – A asks B for something it is trying to “find”
  – Commonly used for management replicated data
    • Resolve differences between DBs by comparing digests
    • Amazon S3
Still several research questions

- Gossip with bandwidth control
  - Constant rate?
  - Tunable with flow control?
  - Prefer to send oldest data? Newest data?

- Gossip with heterogeneous bandwidth
  - Topology / bandwidth-aware gossip

- ...

...
Summary

• IP Anycast
  – Failover and load balancing between IP addresses
  – Uses existing routing protocols, no mods anywhere
  – But problems: scalability, coarse control, TCP stickiness
  – Primarily used for DNS, now being introduced inside ISPs

• Multicast protocols
  – Unrealiable: IP Multicast and IGMP
  – Realiable: SRM, PGM, Bimodal multicast
  – Gossiping