Effects of anycast on K-root

Some early results
Current deployment

• 5 global nodes (BGP transit)
  – LINX
  – AMS-IX (since 7/2004)
  – Tokyo (since 5/2005)
  – Miami (since 7/2005)
  – Delhi (since 8/2005)

• 11 local nodes (announced with no-export)
  – Frankfurt, Athens, Doha, Milan, Reykjavik, Helsinki, Geneva, Poznan, Budapest, Abu Dhabi, Brisbane
Node structure

• 2 machines running nsd, switches, routers

• Production IP: OSPF load balancing
  – K-root IP address: 193.0.14.129

• Service interfaces
  – Normally firewalled, don’t reply to queries
    • LINX: 193.0.16.1, 193.0.16.2
    • AMS-IX: 193.0.17.1, 193.0.17.2
    • …

• Management interfaces, …
Why anycast?

• Reasons for anycasting:
  – Provide resiliency and stability
  – Reduce latency
  – Spread server and network load, contain DOS attacks
  – ...

• Is it effective?

• Measurements taken April-July 2005
Latency
Latency comparison

• Ideally, BGP should choose the node with the lowest RTT.
• Does it?

• Measure RTTs from the Internet to:
  – Anycasted IP address (193.0.14.129)
  – Service interfaces of global nodes (not anycasted): LINX, AMS-IX
    • At the time, there were only two global nodes

• Compare results

• Just to make sure this is apples to apples:
  – Are AS-paths to service interfaces the same as to production IP?
  – According to the RIS, “mostly yes”
Probe locations: TTM (bias?)
Method

• Send DNS queries from all test-boxes
  – For each K-root IP:
    • Do a “dig hostname.bind”
    • Extract RTT
    • Take minimum value of 5 queries
  – Compare results of anycast IP with those of service interfaces

• $\alpha = RTT_K / \min(\text{RTT}_i)$
  – $\alpha \approx 1$: BGP picks the right node
  – $\alpha > 1$: BGP picks the wrong node
  – $\alpha < 1$: local node?
Latency comparison

Local nodes: = better RTT = worse RTT
Local worse than global?

$ cat tt89
193.0.14.129 k2.denic 29 k2.denic 30 k2.denic 29 k2.denic 30 k2.denic 29
193.0.16.1 k1.linx 4 k1.linx 3 k1.linx 3 k1.linx 3 k1.linx 3
193.0.16.2 k2.linx 3 k2.linx 3 k2.linx 3 k2.linx 3 k2.linx 4
193.0.17.1 k1.ams-ix 12 k1.ams-ix 11 k1.ams-ix 12 k1.ams-ix 13 k1.ams-ix 13
193.0.17.2 k2.ams-ix 12 k2.ams-ix 13 k2.ams-ix 11 k2.ams-ix 12 k2.ams-ix 13

(This example has since been fixed)

• What’s going on here? Perhaps:
  – Local node announcements don’t necessarily leak
  – But they do get announced to customers
    …and customers of customers
    …where they compete with announcements from global nodes
    …which lose out due to prepending
Latency comparison (global)
Latency: conclusions

• Local nodes “confuse” the situation due to transit and prepending
• But all in all, BGP does a surprisingly good job

• This contrasts with other work (Ballani & Francis)
  – Perhaps it is because we only saw two global nodes
  • Will it get worse when more nodes are deployed?
  – Perhaps it is because both nodes are in Europe and we are measuring from Europe

• When this was done there were only two global nodes
Load balancing
Usefulness of local nodes

• How much traffic does a local node get?

• Do local nodes take load off the global nodes?

• Where do local queries come from?
  – From the global K nodes?
  – From the other root servers?
Local queries
Local queries (cumulative)
Local vs global

Local vs global queries

Global nodes
Local nodes
Total

~23%
Load balancing: conclusions

- The traffic a local node gets depends on where it is
- Wide variation
  - 2 orders of magnitude!

- We need a way to choose where to put a new node

- Local nodes do take load off the global nodes
  - but not much
- Increase in local traffic does not correspond to decrease in global traffic
  - Traffic mostly seems to come from the other roots
Stability
Node switches

• Didn’t measure resiliency
  – Pretty much a given: the more servers there are,
    • the more they can withstand
    • the more localised the impact of an attack
• What about stability?
  – The more routes competing in BGP, the more churn
  – Doesn’t matter for single-packet exchanges (UDP)
  – Does matter for TCP queries
• How frequent are node switches?
Detecting node switches

• Measure at the server
• Look at node switches that actually occur

• Procedure:
  – Look at packet dumps
    • At the time, there were only 2 global nodes
  – Extract all port 53/UDP traffic
  – For each IP address, remember where it was last seen
  – If the same IP is seen elsewhere, log a switch

• Caveats:
  – K nodes are only NTP synchronized
Node switches: results

- 24 hours of data:
  - 527,376,619 queries
  - 30,993 node switches (~0.006%)
  - 884,010 IP’s seen
  - 10,557 switching IPs (~1.1%)

- What do the switches look like?
Time since last switch
Time since last switch, log-log
Top switching IPs
Stability: conclusions

- Node switches are rare
- But some IPs switch a lot
  - Load balancing?
  - Need to look into this

- What do the switch profiles mean?
  - We don’t know yet
  - Further analysis needed
To sum it up...

• Anycast works very well for clients
  – Latency is very good
    • But local nodes can make things worse instead of better
  – Affinity does not seem to be a problem
    • 99.994% of queries hit same server as last query
    • 98.9% of IPs never switched in one a day

• Anycast works well for operators
  – Location for new nodes must be carefully considered
  – Local nodes don’t take much load off global nodes
  – When a new node is deployed, traffic mostly comes from the other roots
Next steps

• Detailed writeup of results in progress

• Short term:
  – Look at effects of new global nodes
    • Does anycast still work so well?
  – Look at traffic distribution between global nodes

• Longer term:
  – Look at “pathological” node switchers
  – Develop methodology to choose location for new nodes

• Suggestions?
Questions?