Effects of anycast on K-root performance

Status update

Work presented @ RIPE 51

- Evaluated anycast goals:
  - Latency
    - Measured by querying from TTM
  - Load balancing
    - Looked at activity logs
  - Stability
    - Looked at instance switches seen by servers

RIPE 51 results

- Anycast is good for latency
  - TTM saw very good performance
  - BGP almost always picked the right node
    - Although local nodes seem to confuse things
- Not so good for load balancing
  - Wide variation in node load
- Instance switches are infrequent
  - But there are “pathological” switchers

Unanswered Questions

- Only 2 global nodes measured, and only on 2 occasions
  - Do the same results hold for the current 5 nodes?
  - Are the results consistent over time?
- Did measurement point bias affect the results?
  - TTM boxes are mostly based in Europe
- “Pathological” instance switchers
  - What causes this?

Latency measurements using 5 nodes

Latency with TTM

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

- Measure RTTs from the TTM boxes to:
  - Anycasted IP address (193.0.14.129)
  - Service interfaces of global nodes (not anycasted)
- Compare results

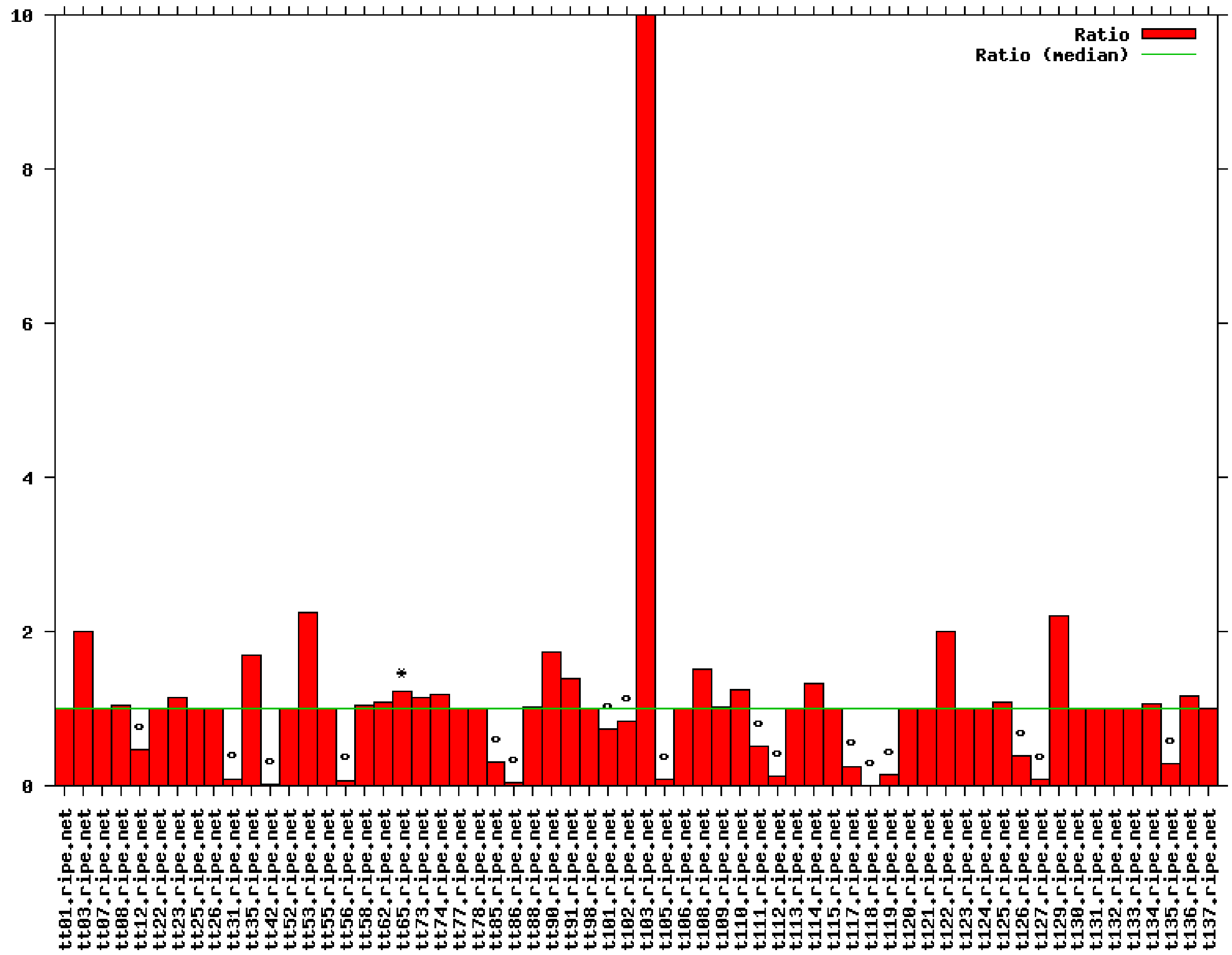
- To make sure this is apples to apples:
  - Are paths to service interfaces the same as to production IP, if picked?
  - According to the RIS, “mostly yes”

Latency with TTM: methodology

- 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

- $\alpha = RTT_K / \min(RTT_i)$
  - $\alpha \approx 1$: BGP picks the right node
  - $\alpha > 1$: BGP picks the wrong node
  - $\alpha < 1$: local node?

Latency with TTM: results (5 nodes)



What's up with tt103?

```
results/200604120000 $ cat tt103.ripe.net
193.0.14.129 k1.delhi 422 k1.delhi 416 k1.delhi 423 k1.delhi 428 k1.delhi 419
[...]
203.119.22.1 k1.tokyo 2 k1.tokyo 2 k1.tokyo 2 k1.tokyo 2 k1.tokyo 2
```

- tt103 is in Yokohama
  - Tokyo is 2ms away
    - But it goes to Delhi
    - ... through Tokyo, Los Angeles and Hong Kong

- RTT = 416 ms, α = 208

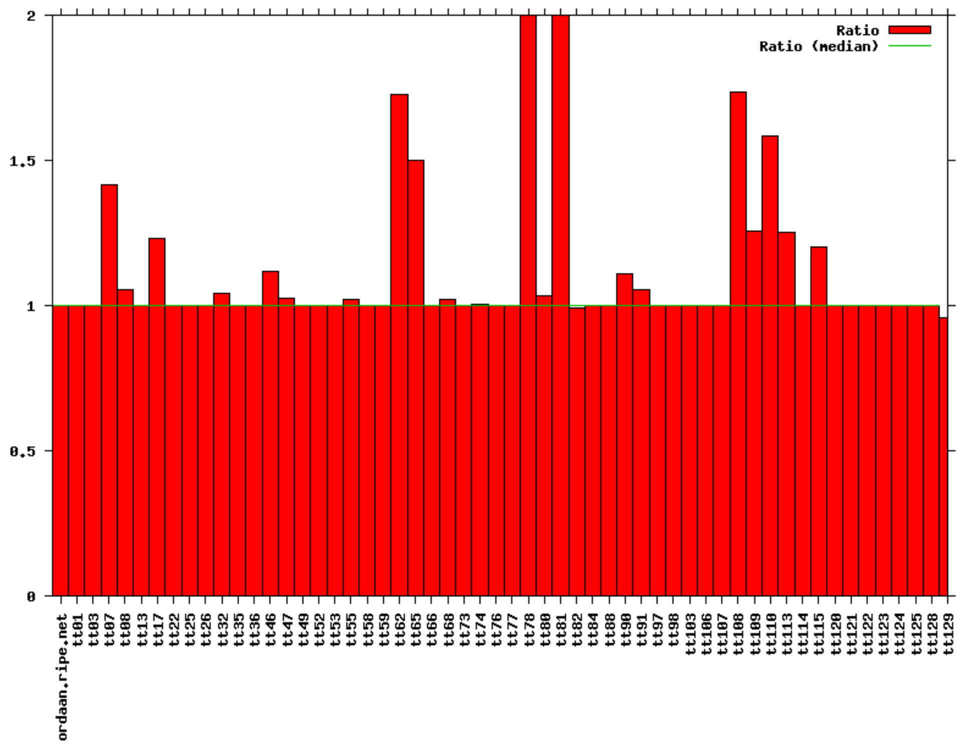
Problem: different prepending lengths

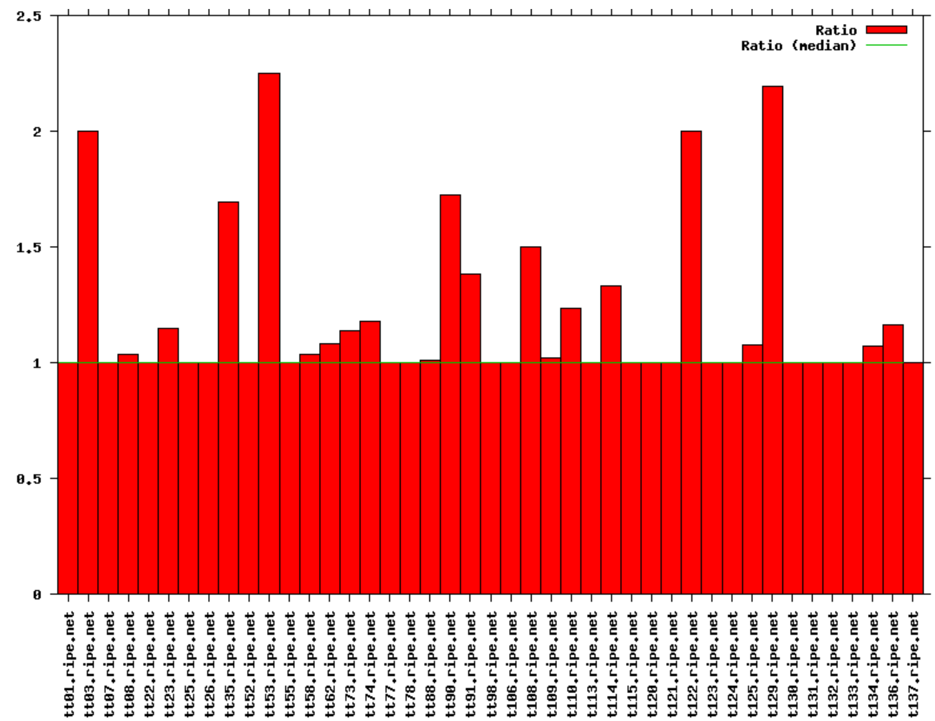
- Got BGP paths from AS2497
  - Thanks to Matsuzaki and Randy Bush

- Problem: bad interaction of different prepending lengths
  - Tokyo:
    - 2914 25152 25152 25152 25152
    - 4713 25152 25152 25152 25152
    - 6461 25152 25152 25152 25152
  - Delhi:
    - 2200 9430 25152 25152

- We need to fix prepending on Tokyo node

5-node vs 2-node results





2 nodes

5 nodes

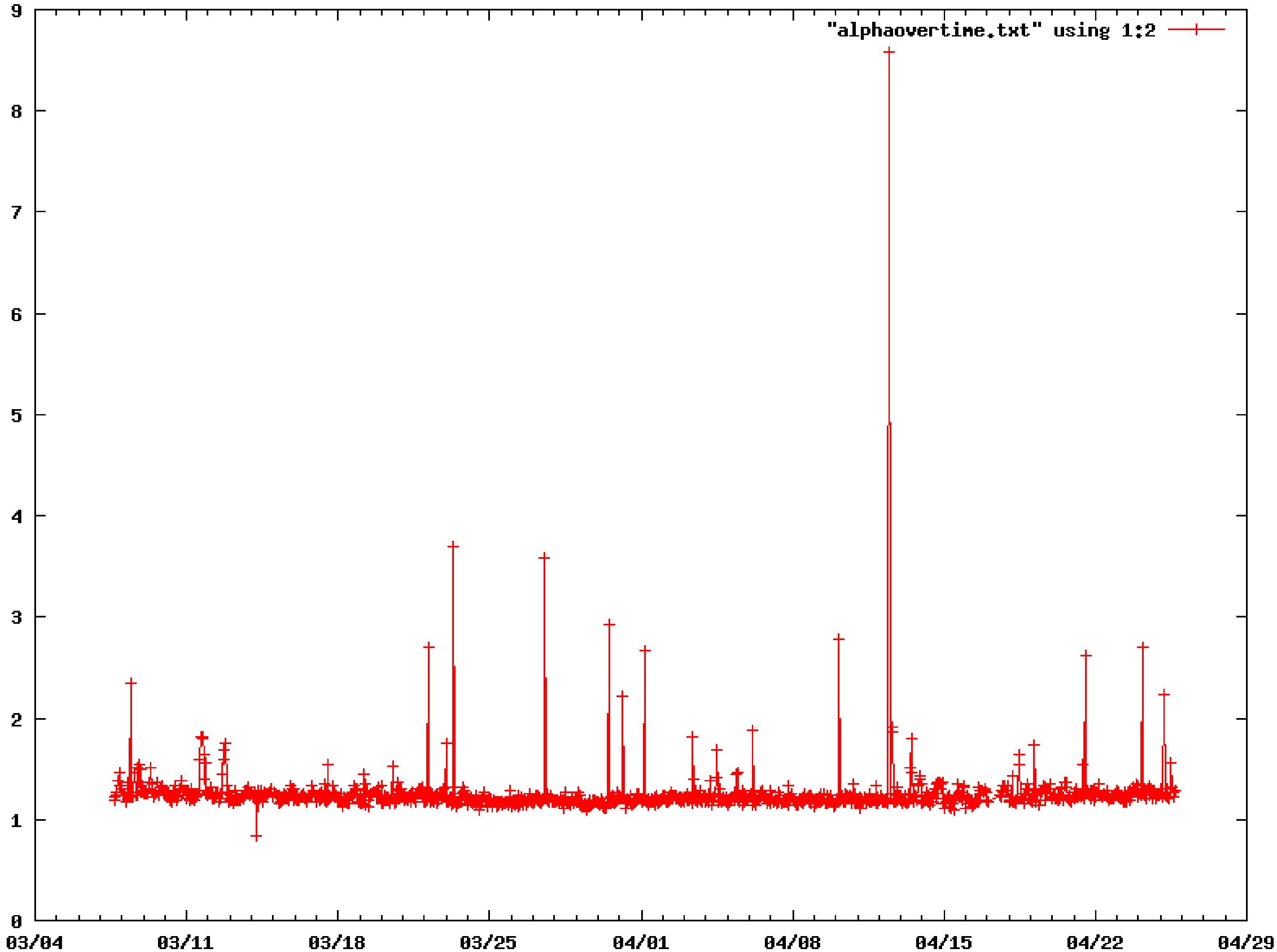
Essentially no different

Consistency of results over time

Consistency of α over time

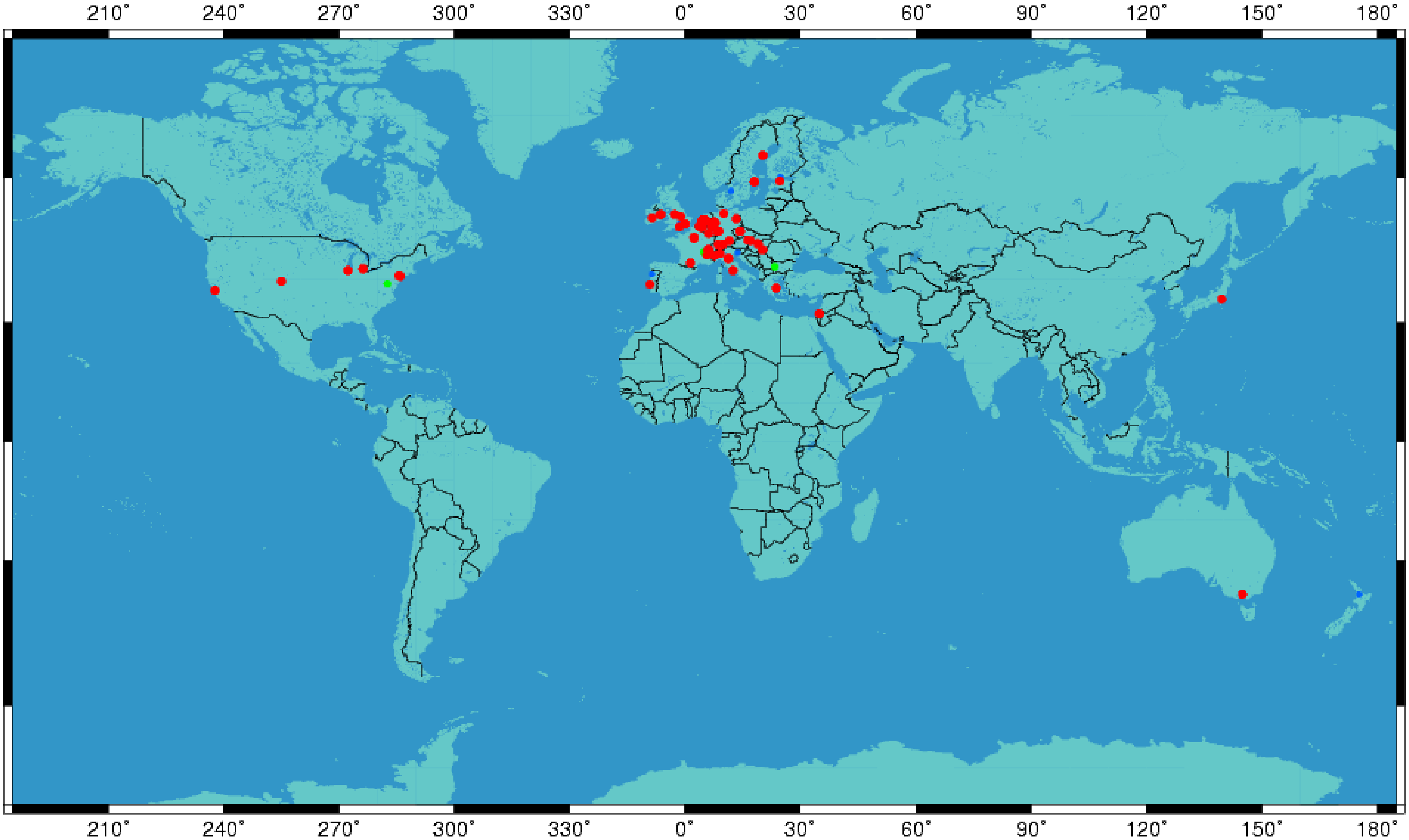
- Is this a chance event or is this behaviour consistent?
- Plot average α over time
  - Collect α for all test-boxes every hour
  - Take average (excluding tt103)
  - Plot over time
- Results:
  - Average: 1.25, median: 1.22
  - BGP is fairly consistent

Average value of α over time



Is TTM data meaningful?

TTM: probe locations



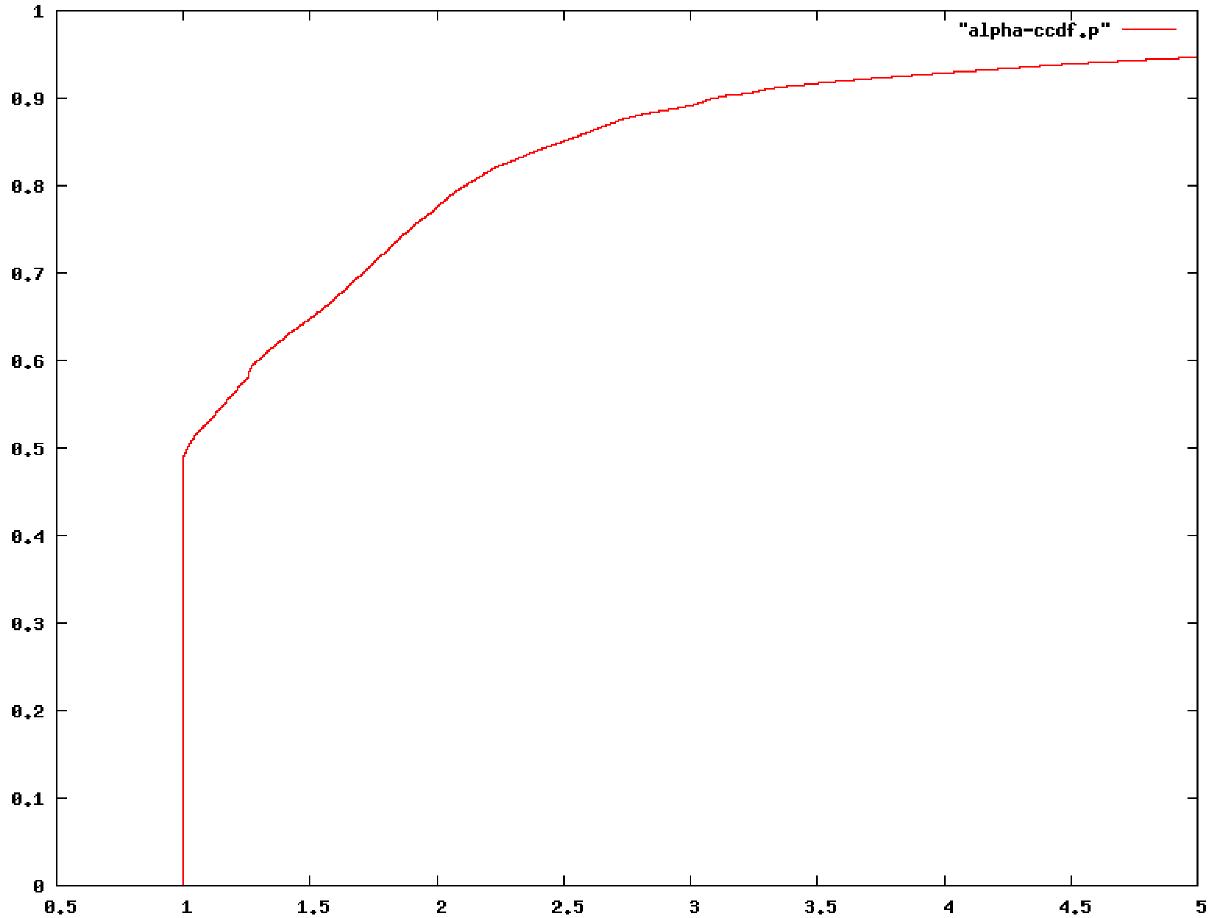
Measuring from TTM and from servers

- TTM latency measurements not optimal
  - Locations biased towards Europe
  - Only limited number of probes (~100)
  - Do not necessarily reflect K client distribution
- How do we fix this?
- Ping servers from clients
  - Much larger data set (~100 -> ~ 1M)
  - Measures the effect K's actual clients

Methodology

- Methodology:
  - Analyse packet traces on K global nodes
  - Extract list of IP addresses, merge lists
  - Ping all addresses from all servers
  - Plot distribution of α
- Results:
  - 6 hours of data
  - 246,769,005 queries
  - 845,328 IP addresses

CDF of α seen from servers



- Results not as good as seen by TTM
  - Only 50% of clients have $\alpha = 1$

Latency: conclusions

- 5-node results comparable to 2-node results
- TTM clients (= Europe) very well served by K
- If we look at total K client population, things not so rosy

Incremental benefit of nodes

How many nodes are enough?

- Does it make sense to deploy more instances?
  - Have we reached the point of diminishing returns?
- Evaluate benefit of existing instances
  - Hope this will tell us at what point in the curve we're on
- How do we measure the benefit of an instance?
  - We can quantify how much performance would worsen if that instance did not exist

Methodology

- Assume optimal instance selection
  - That is, every client sees closest instance
  - This is an upper bound to benefit
    - Consistent with our aim of seeing whether we have reached the point of diminishing returns

- For every client, see how much its performance would suffer if a given instance did not exist
  - We can do this because we ping all clients from all instances

Loss factor

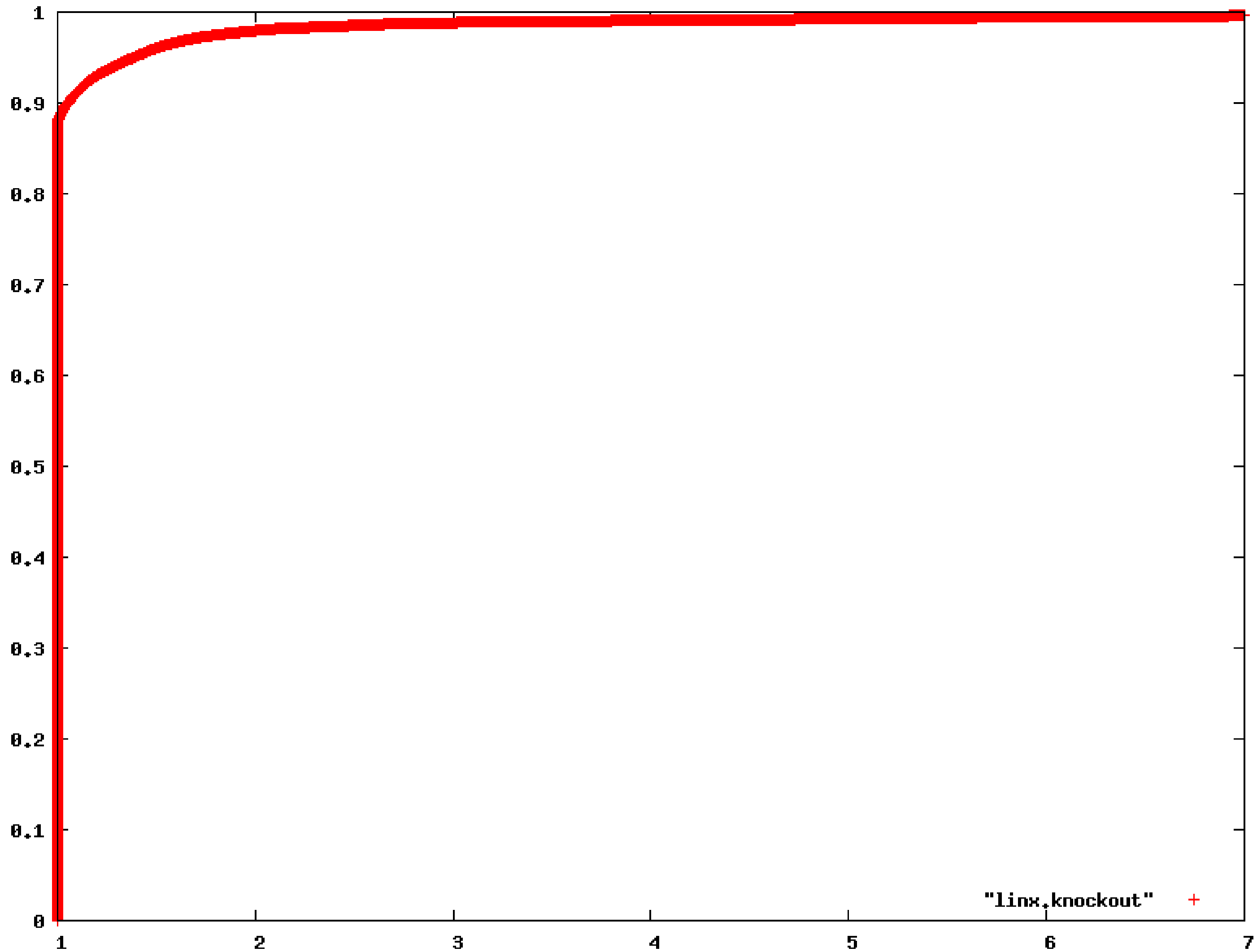
- "Loss factor" β determines how much a client would suffer if an instance were knocked out

$$\beta = \frac{RTT_{knockout}}{RTT_{best}}$$

- If $\beta = 1$, the client would see no loss in performance
- If $\beta = 2$, the client sees double RTT

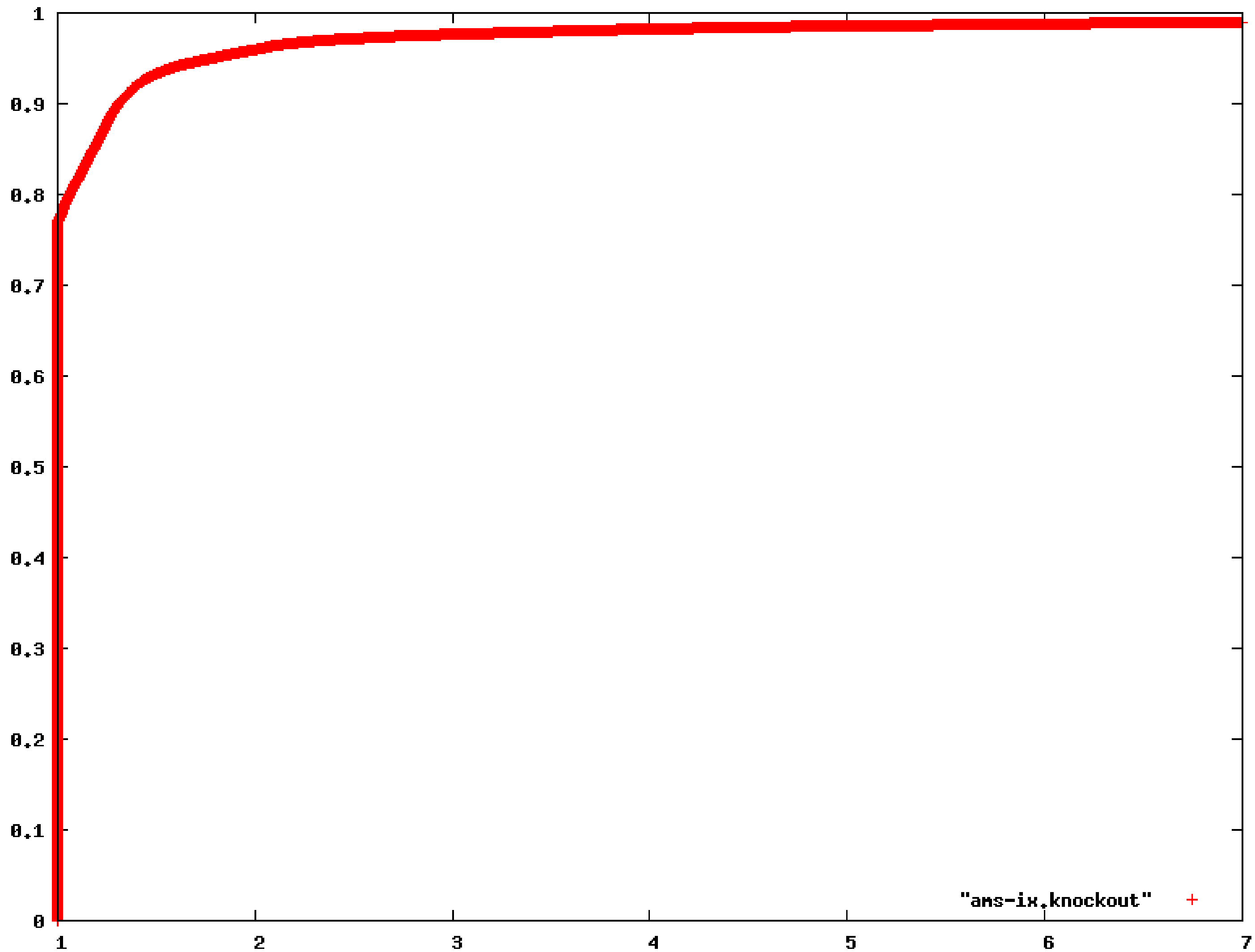
- Plot CCDF of β for every node
- This gives us an idea of how "important" a node is

Results: LINX



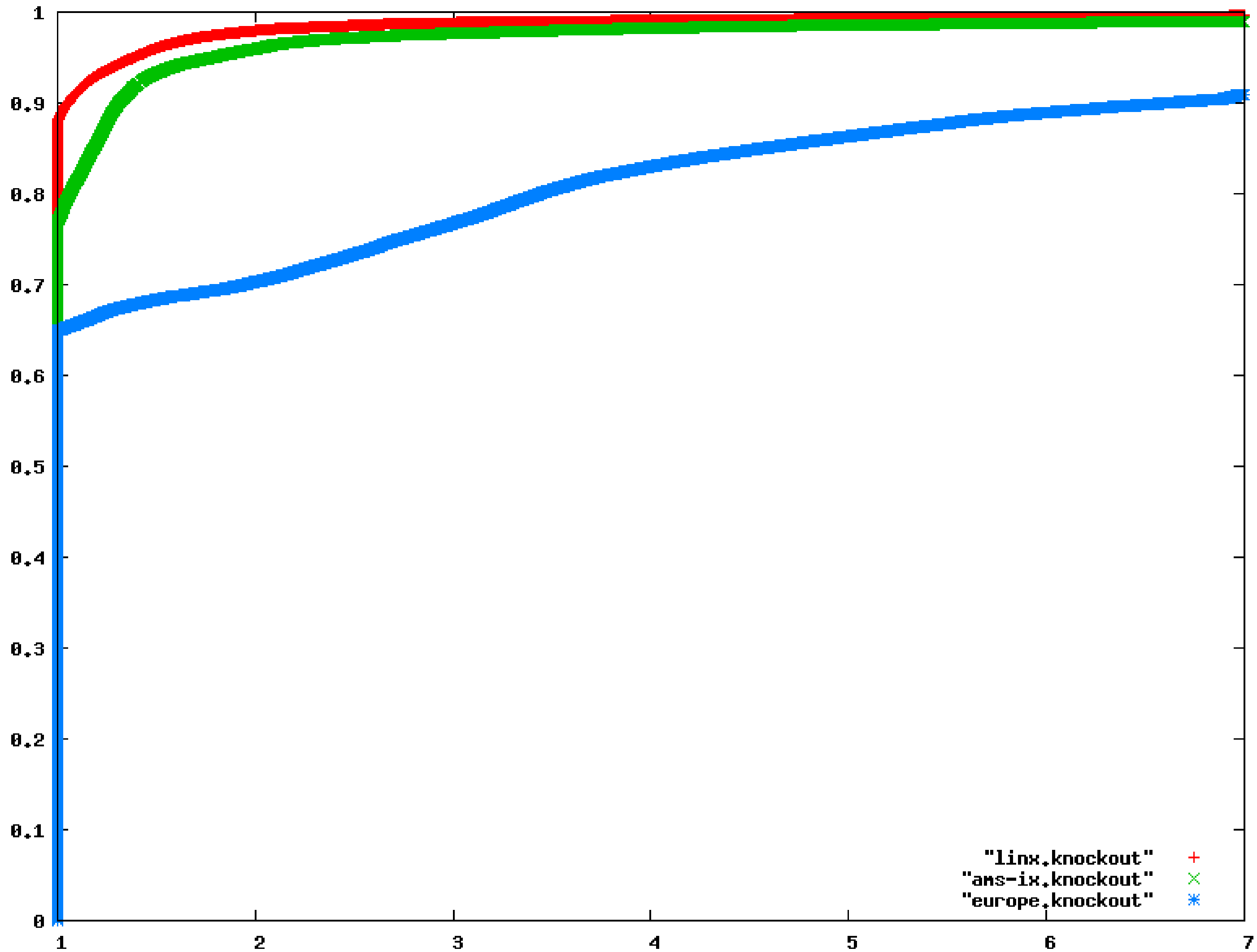
Not much benefit on its own

Results: AMS-IX



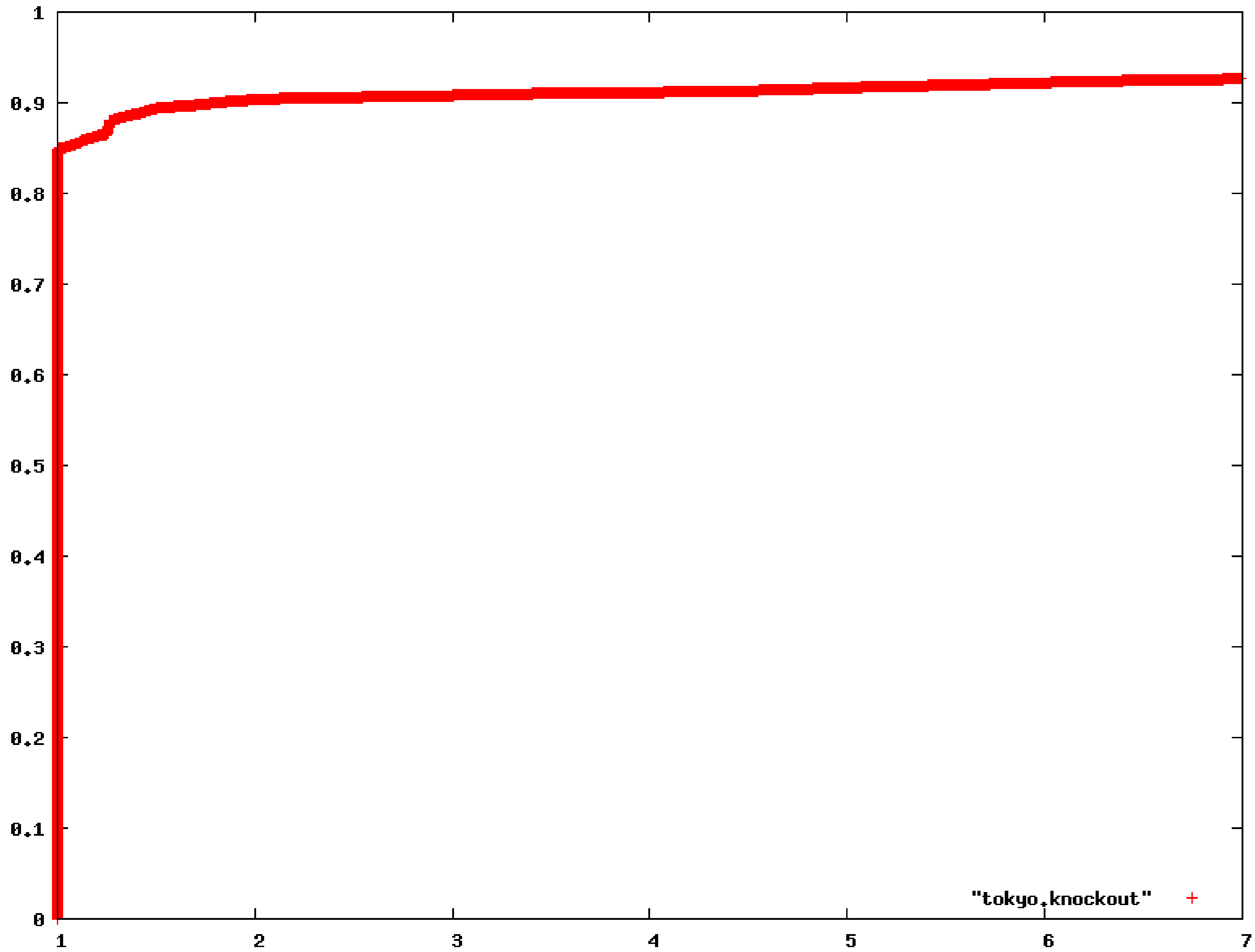
Not much benefit on its own

Results: LINX and AMS-IX



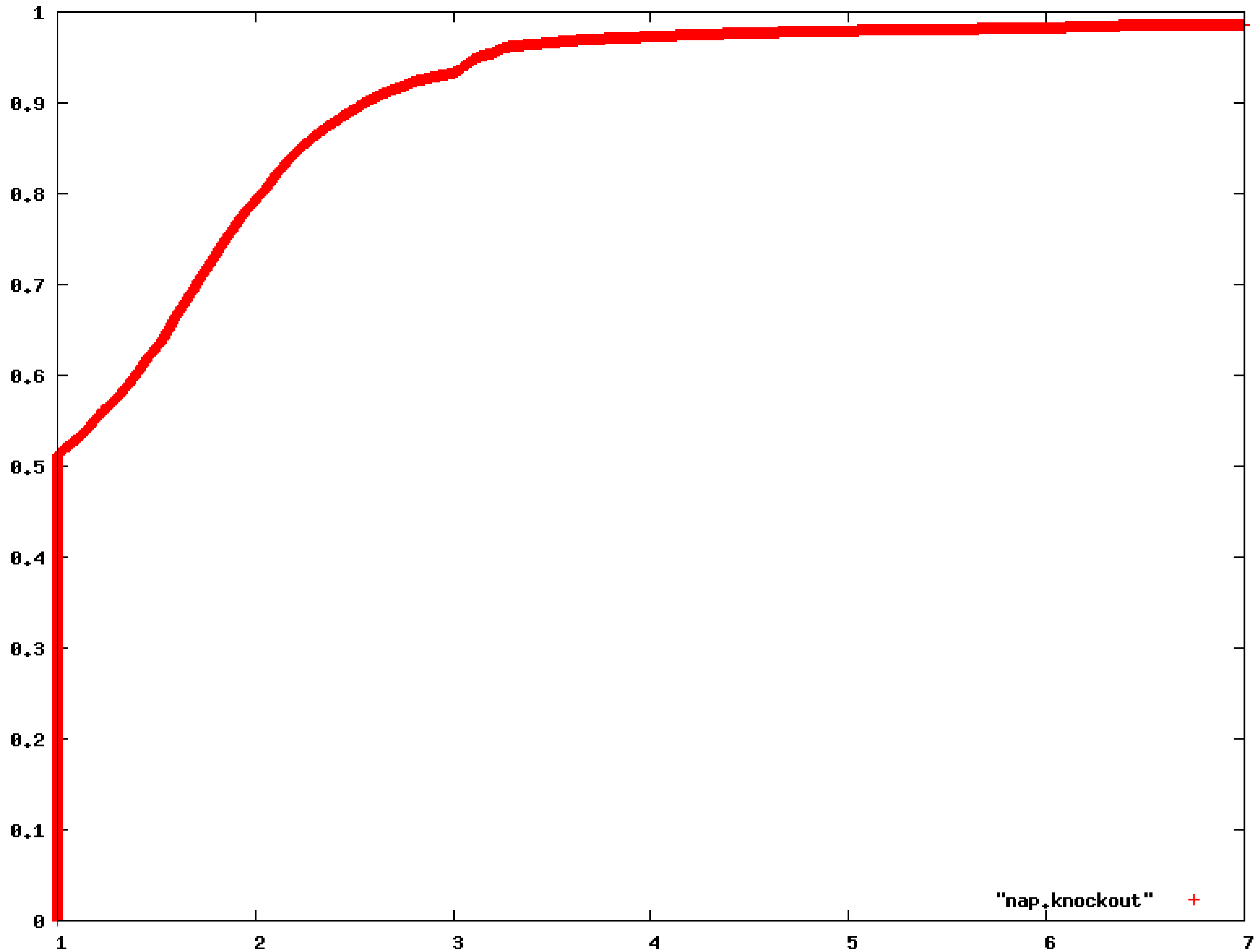
But wait, LINX and AMS-IX are important taken together…

Results: Tokyo



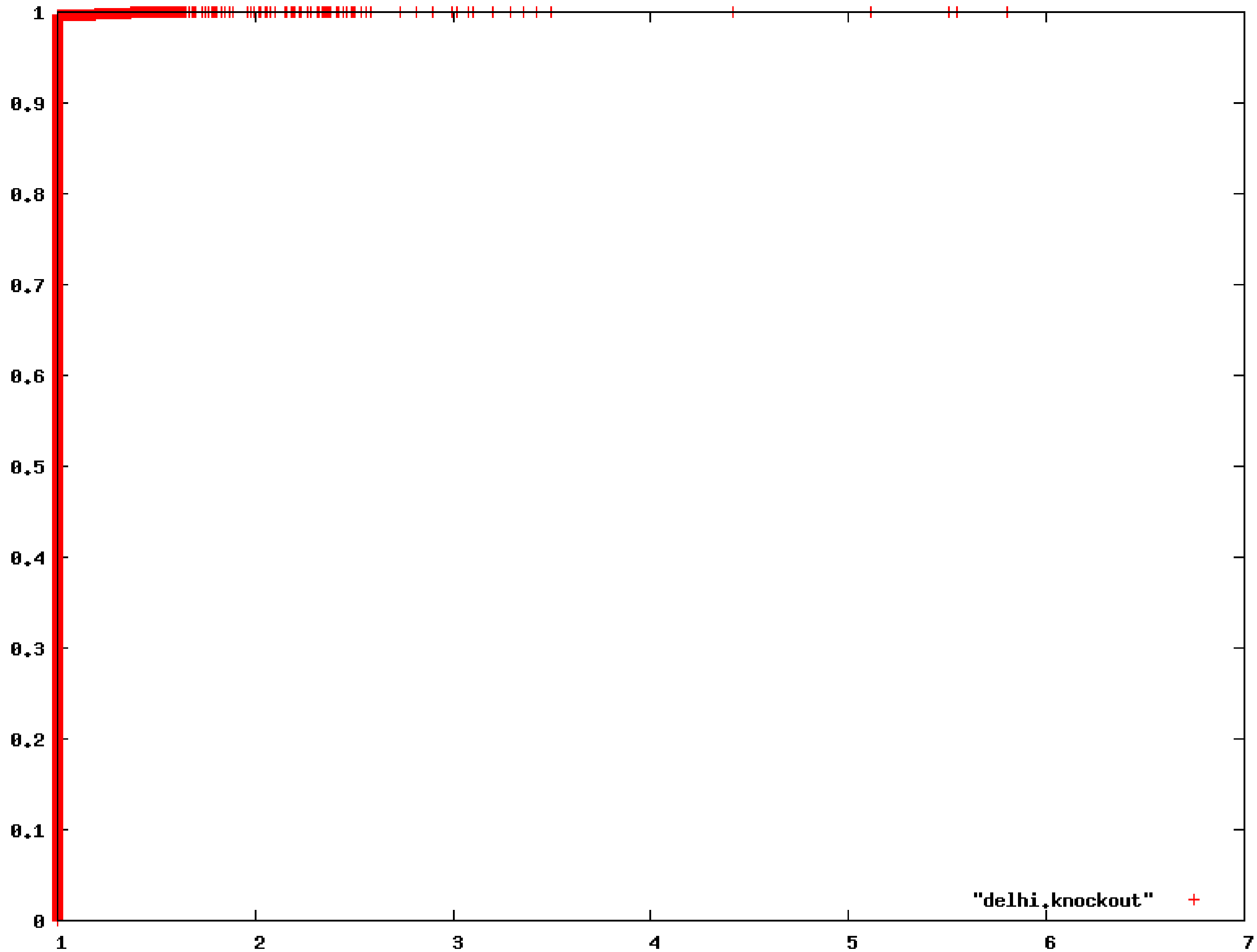
Few clients, but very badly served by other nodes

Results: NAP



Moderately better for some clients

Results: Delhi



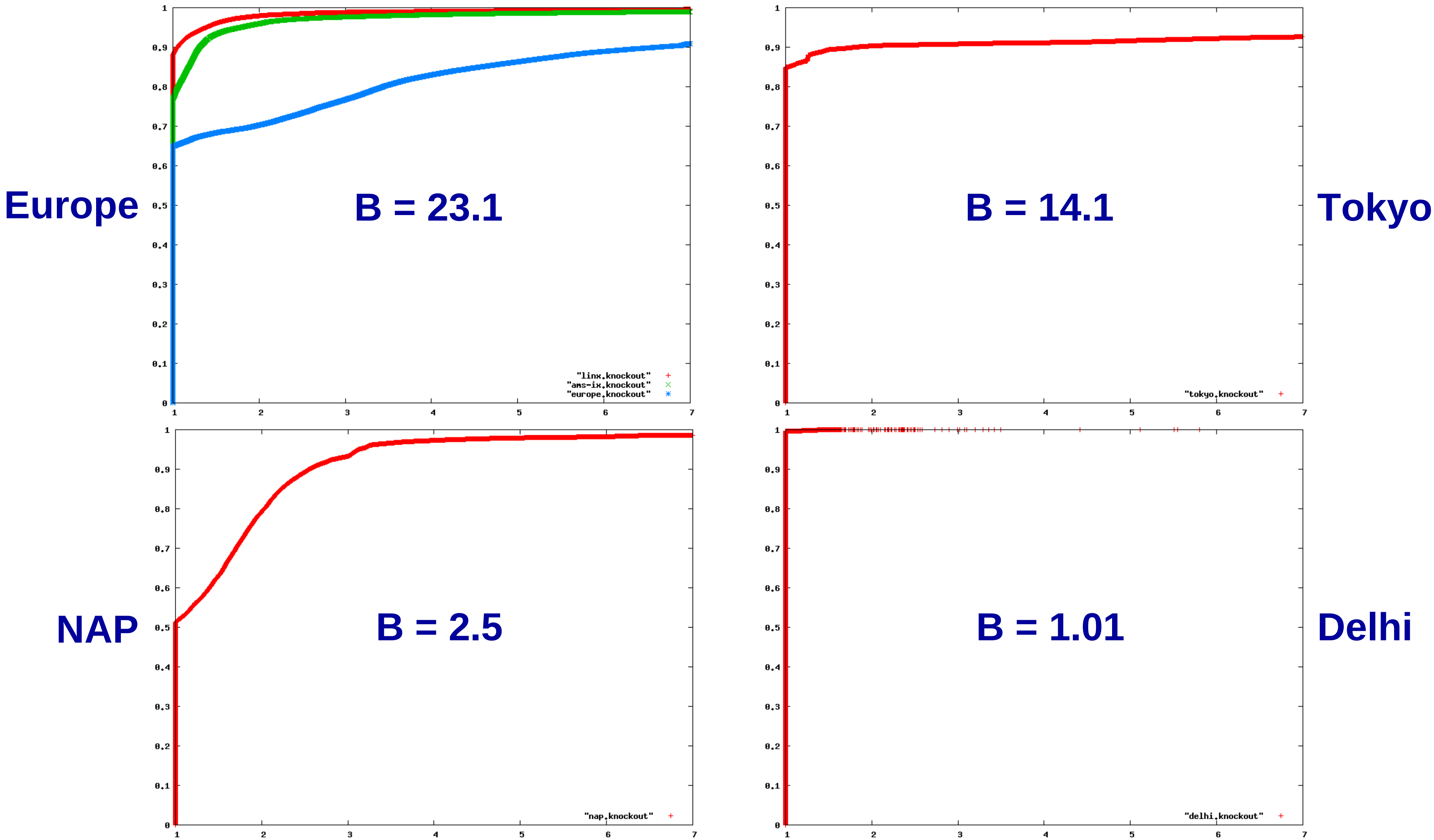
Not very effective

Incremental benefit of a node

- Take β values for all clients
- Take the weighted average, where the weights are the number of queries seen by each client

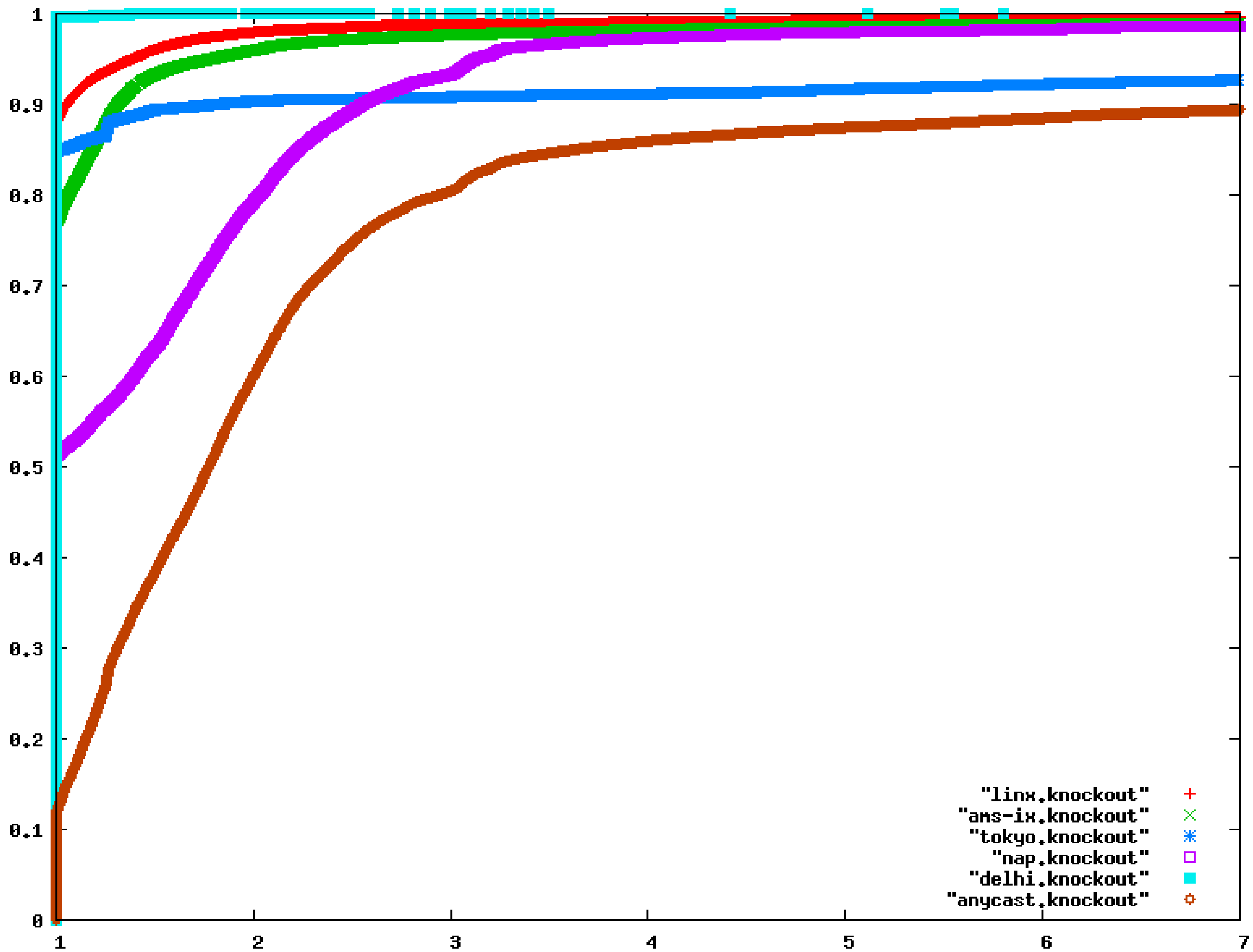
$$B = \frac{\Sigma_i \beta_i Q_i}{\Sigma_i Q_i}$$

Values of B


Europe
B = 23.1
"linx.knockout"
"ams-ix.knockout"
"europe.knockout"
Tokyo
B = 14.1
"tokyo.knockout"
NAP
B = 2.5
"nap.knockout"
Delhi
B = 1.01
"delhi.knockout"

Does anycast provide any benefit?

- What if we didn't do anycast at all?
- Knock out all except LINX: dark red curve
- B = 18.8
- For K, anycast **works well**



Stability

Stability

- RIPE 51 presentation concluded that instance switches are not a problem
- Is this still the case with 5 nodes?
  - The more nodes, the more routes in BGP and the more churn

Stability results

2 nodes (RIPE 51)

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

5 nodes

- ~5 hours of data:
  - 246,769,005 queries
  - 150,938 switches (0.06%)
  - 845,328 IPs seen
  - 2,830 switchers (0.33%)

Still does not seem a serious problem

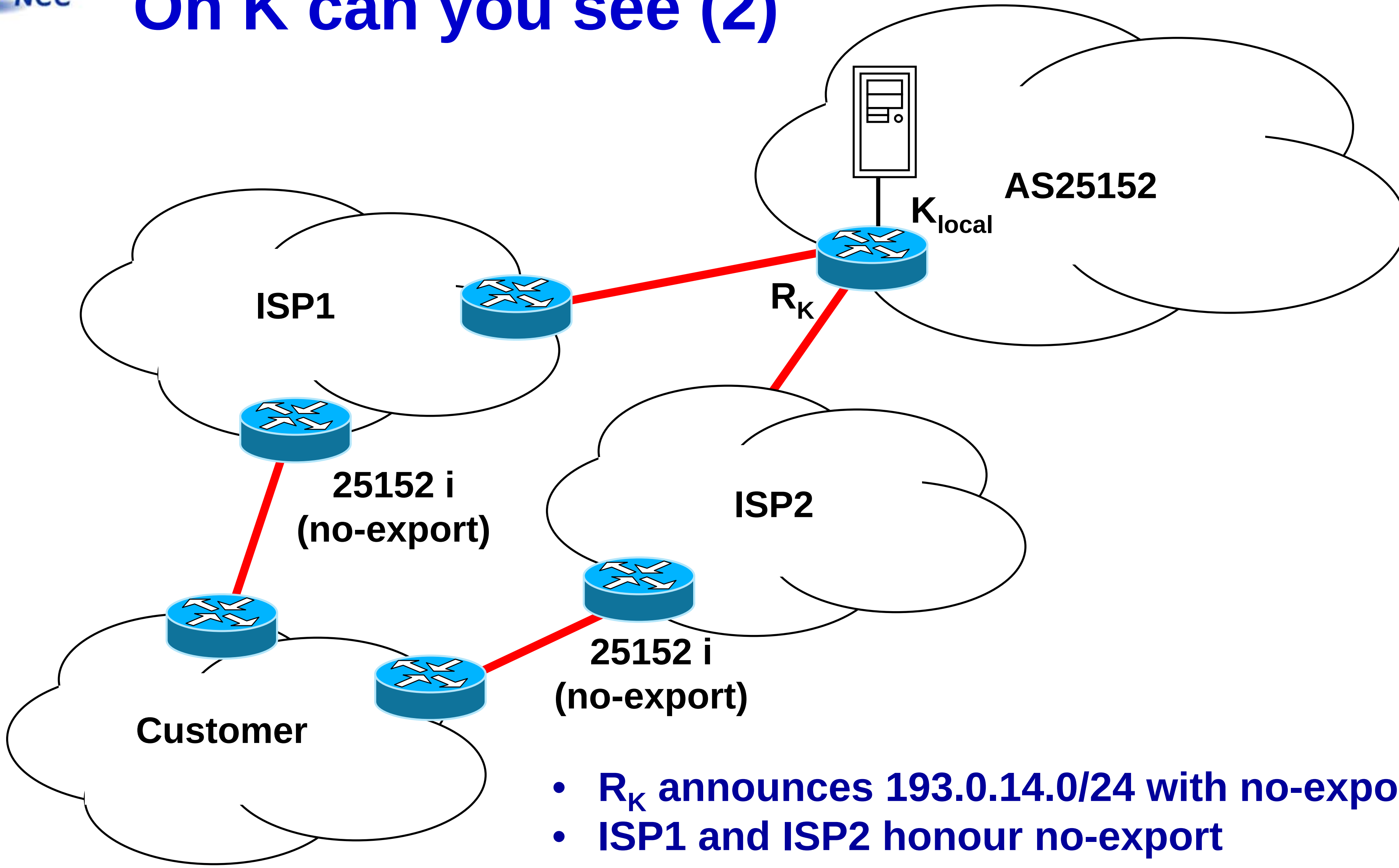
Questions?

Oh K can you see

- Problem pointed out by Randy Bush
- http://www.merit.edu/mail.archives/nanog/2005-10/msg01226.html

- Nasty interaction of no-export with anycast
  - We use no-export to prevent local nodes from leaking
  - If we have a customer AS
    - Whose providers all peer with a local node
      - And honour no-export
    - They might see no route at all!

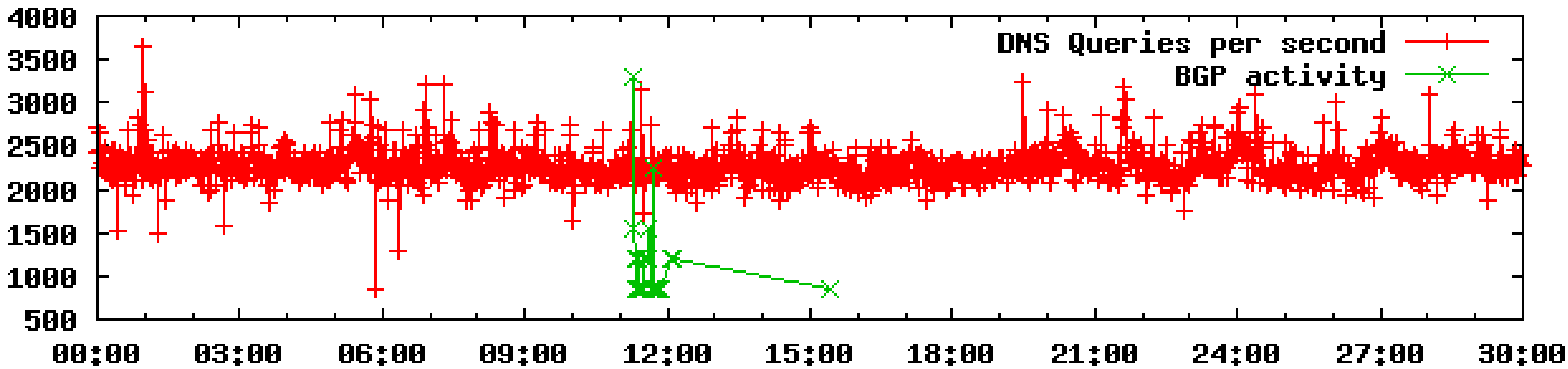
Oh K can you see (2)



- R_K announces 193.0.14.0/24 with no-export
- ISP1 and ISP2 honour no-export
- Customer has no route to 193.0.14.0/24

Extent of the problem

- Solution: announce 193.0.14.0/23 without no-export @ams-ix
- Was this a problem?
- See what happened when prefix was announced



- Red: AMS-IX queries per second
- Green: BGP activity
- “Nothing here”

RIPE 51 results (2 nodes)

- 24 hours of data:
  - 527,376,619 queries
  - 30,993 node switches (~0.006%)

  - 884,010 IPs seen
  - 10,557 switching IPs (~1.1%)

- Is this still the situation with 5 nodes?
  - The more routes competing in BGP, the more churn

Covering prefix announcement