RPKI Rsync Performance Test Update

David Mandelberg <dmandelb@bbn.com>

Thanks to Steve Kent, Andrew Chi (BBN), and Kotikalapudi Sriram (NIST) for valuable input.

Background

- Resource Public Key Infrastructure (RPKI) is designed to use rsync to distribute files to all ISPs running BGP
- Various concerns over the performance of rsync in the RPKI environment, e.g.:
  - Can it handle all the files in a repository as it grows?
  - How quickly can clients synchronize everything?
  - How many clients can a server handle?
- Oleg Muravskiy and Tim Bruijnzeels (RIPE NCC) ran a scaling experiment with 5 Mac Minis on a LAN[1]
  - We corroborate scaling with a realistic server and network

[1] T. Bruijnzeels, O. Muravskiy, and B. Weber, "RPKI Repository Analysis and Delta Protocol," March 2013. http://www.ietf.org/proceedings/86/slides/slides-86-sidr-2.pdf

Recap of Presentation[1] at IETF 86

- Estimated ~290k total objects in the global RPKI with full deployment, or ~445k with BGPSEC (there were 8,448 objects on 2013-07-22)
- With only one client and one server:
  - Scaling by total number of files in a repository is roughly linear between 100k and 700k files.
  - It takes less than 10 minutes to download 700k 1458-byte files.
- So, what happens when many simultaneous clients download from a server?

[1] S. Kent and K. Sriram, "RPKI rsync Download Delay Modeling," March 2013. http://www.ietf.org/proceedings/86/slides/slides-86-sidr-1.pdf

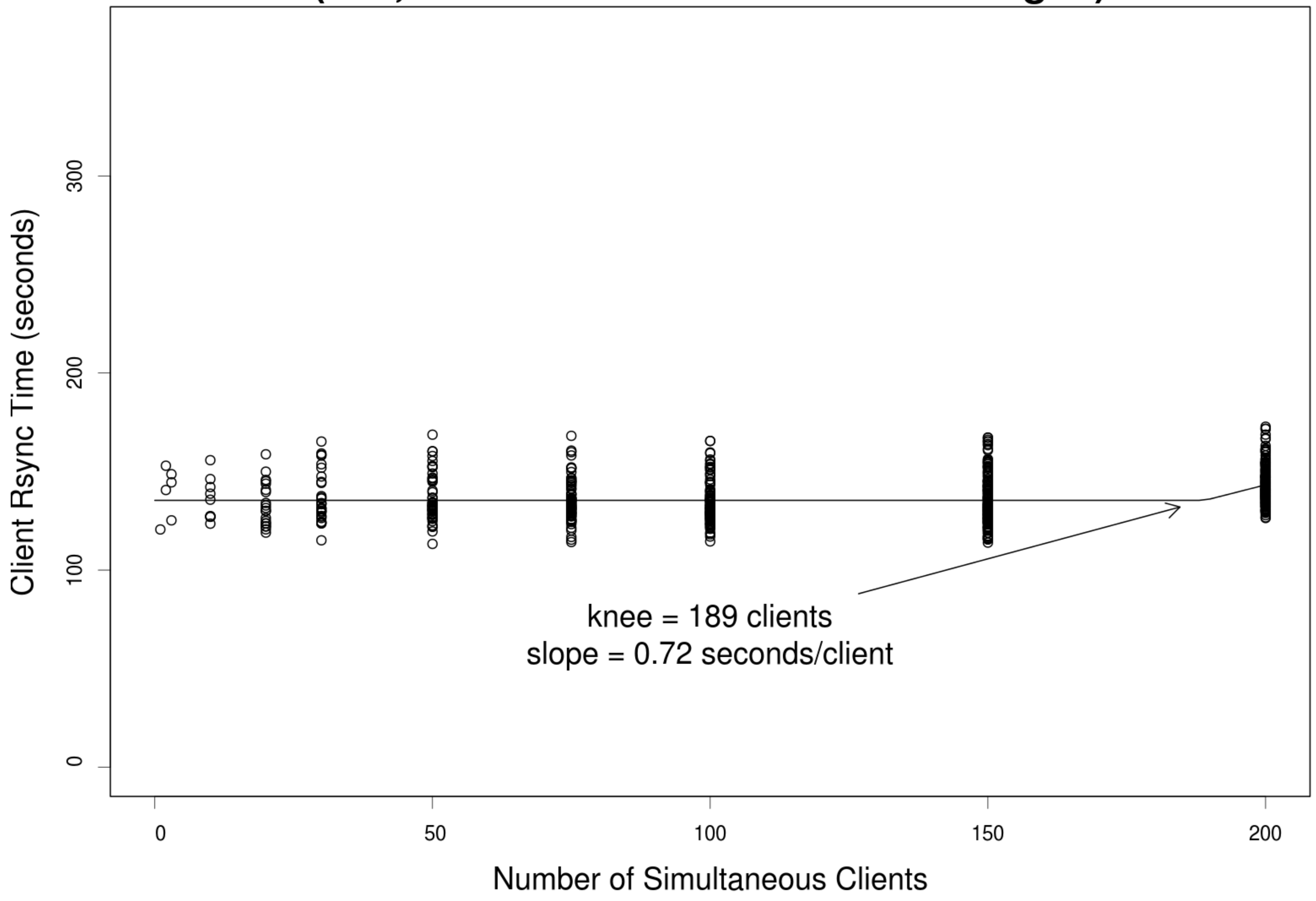
Experiment

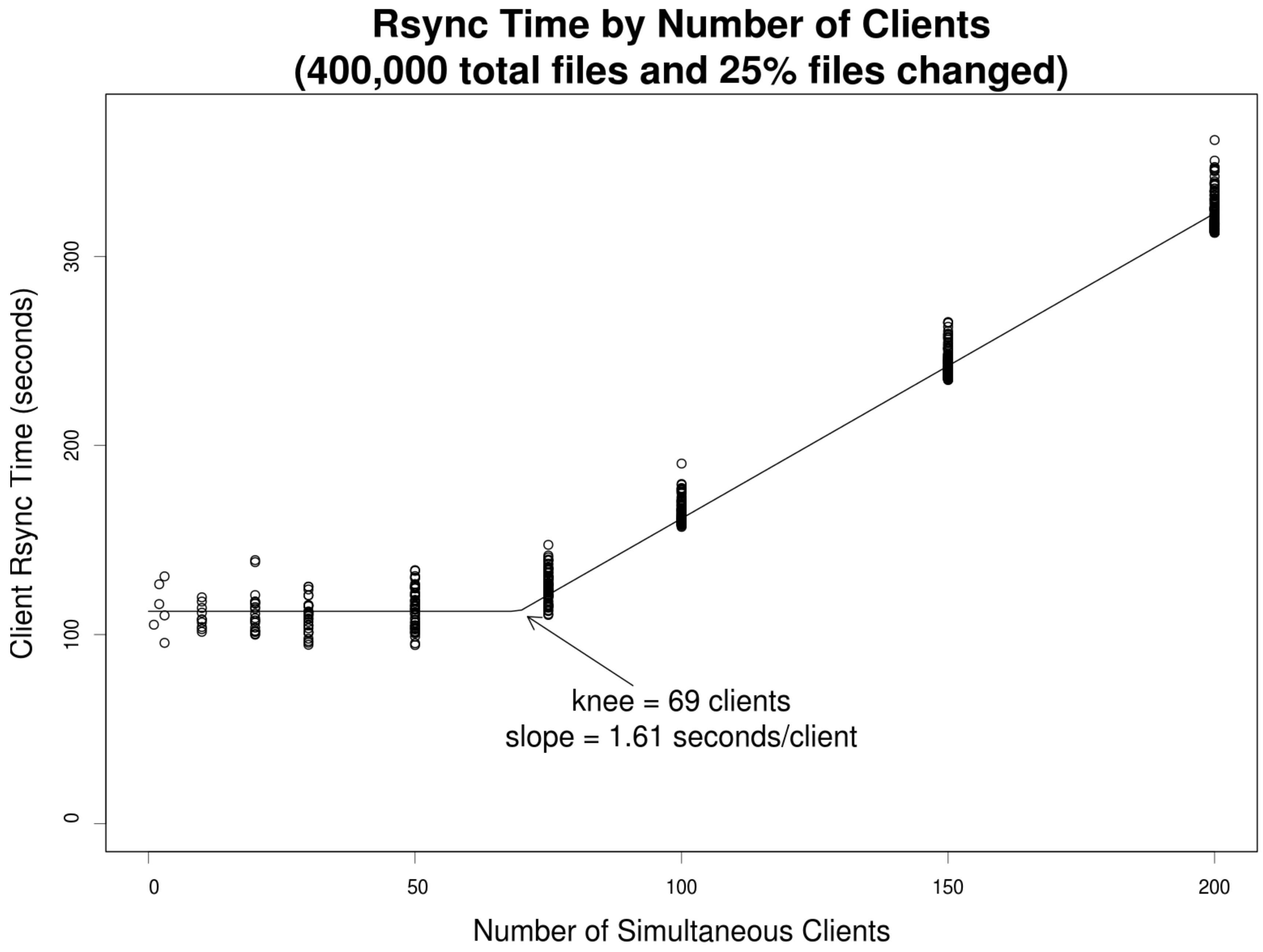
- Rsync only, no validation (this experiment's emphasis is on server performance)
- Simulated repositories:
  - 400,000 files representing RPKI objects
    - The previous experiment showed linear scaling between 100k and 700k (400k is in the middle and we like round numbers)
  - 1458 bytes of pseudo-random data per file
    - When we downloaded from the five RIRs on 2013-01-28 at 17:15 UTC, the median size of RPKI objects was 1458 bytes (and the mean was 1405).
- Varying number of simultaneous clients. For each measurement, all clients are started as close to the same time as possible.
- Varying percent of files changed between the server and the clients' caches
  - 25% corresponds to relying parties polling every 24 hours.
  - 5% corresponds to relying parties polling every ~5 hours, assuming updates are evenly distributed in time.

Experiment

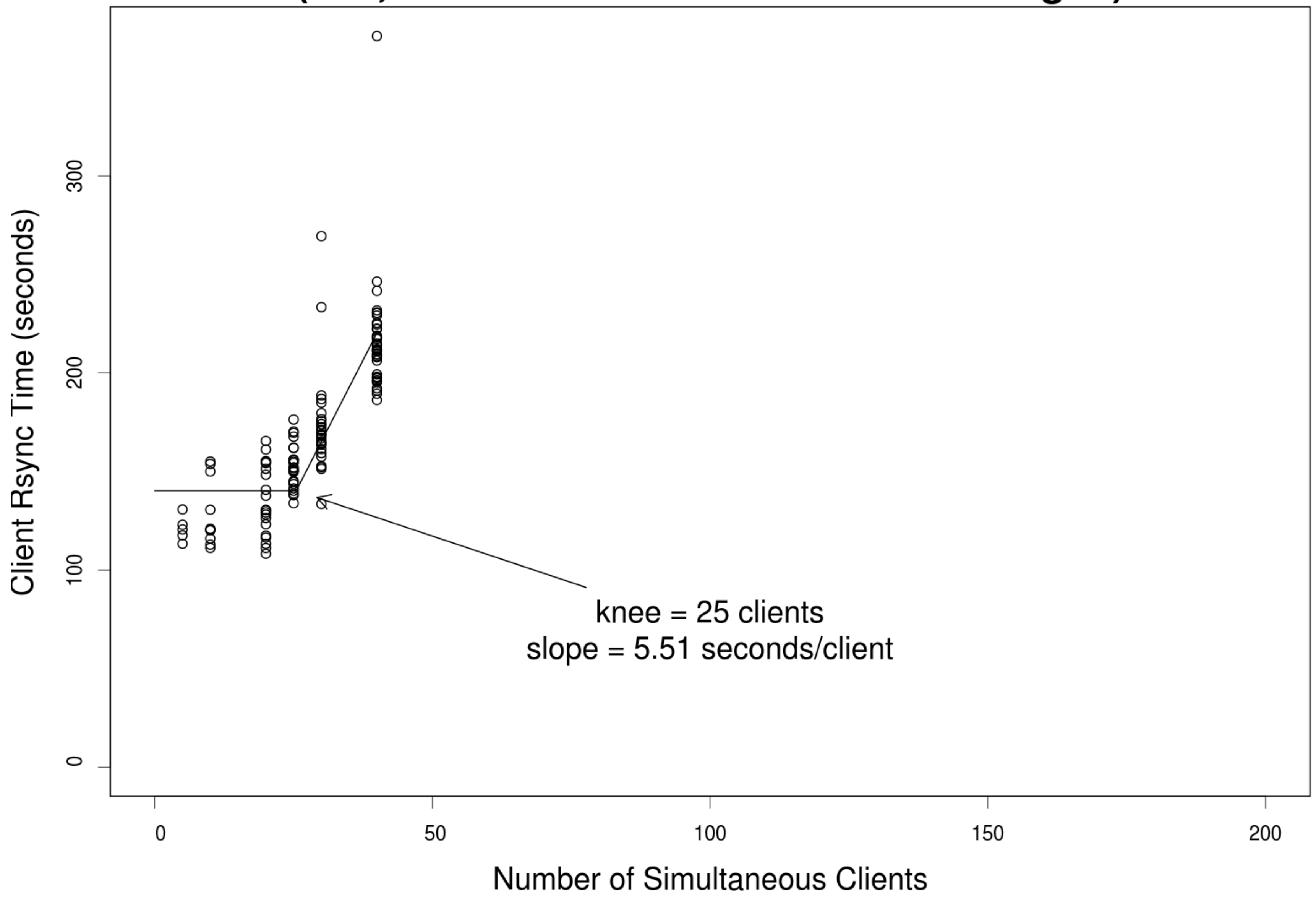
- Clients were geographically distant from the server to represent realistic network delays.
- Server
  - Amazon EC2 m3.2xlarge in Tokyo (US$ 1.52/hour = US$ 13,315.20/year)
  - 8 core Xeon E5-2670 at 2.60GHz
  - 30 GB RAM (but we could have used less)
  - Based on discussion with Chris Morrow (Google), we believe this to be a realistic model of a medium-capacity server.
  - Repository stored in RAM using Linux's ramfs (~600MB)
- Clients
  - Amazon EC2 m1.medium in North Virginia
  - Xeon E5-2650 CPUs at 2.00GHz, 1 core per client virtual machine
  - 3.75 GB RAM
  - Repository cache on instance store
- Ubuntu 12.04 64-bit, with no additional performance tuning.

Rsync Time by Number of Clients (400,000 total files and 5% files changed)




Rsync Time by Number of Clients
(400,000 total files and 25% files changed)
Client Rsync Time (seconds)
0
100
200
300
Number of Simultaneous Clients
0
50
100
150
200
knee = 69 clients
slope = 1.61 seconds/client

Rsync Time by Number of Clients (400,000 total files and 100% files changed)



Note: The points with 50 simultaneous clients were removed due to issues on the clients' side of the experiment.

Server Resource Usage

- CPU was limiting factor for 5% and 25% of files changed
  - **With 5% files changed and 200 clients: 100%**
  - **With 25% files changed and >= 75 clients: 100%**
  - With 100% files changed: ~80% or lower
- Server transmit capacity was limiting factor for 100% of files changed
  - With 5% files changed and 200 clients: ~550Mbit/s
  - With 25% files changed and >= 75 clients: ~800Mbit/s
  - **With 100% files changed and >= 30 clients: ~900Mbit/s**
- Other Factors
  - RAM: 400,000 files * 1458 bytes per file < 600MB
  - Server receive capacity: minimal usage
  - Disk: minimal usage (repository stored in RAM)

Conclusions

- For lower percentages of change:
  - CPU was the limiting factor in our setup.
  - If CPU were increased to 12 or 16 cores, a 1 Gb/s network interface probably would become the limiting factor.
- For higher percentages of change:
  - Network was the limiting factor.
- Flat scaling before limiting factor reached, linear scaling after, with slopes that are operationally tractable
- RIRs or other operators of large repositories can use multiple servers (preferably geographically distributed) with DNS-based load balancing to reduce the number of simultaneous clients using each server.
  - Cloud computing might provide this with minimal up-front costs.
- Servers with dual network interfaces may be useful for mixed workloads when most clients have small changes to download and a few clients need to download everything.
- **A reasonable server should be able to saturate a 1Gbit link.**

Example Context For All These Numbers (i.e., more numbers)

- 43k ASes[1] polling the RPKI
- 5 hour polling interval, uniformly distributed
- 140 seconds to synchronize everything (per-client, as seen by the server, for the flat portions of the curves)
- DNS round-robin of 5 servers per repository
- 43k * (140s / 5hr) = 334 simultaneous clients synchronizing with a repository
- 334 clients / 5 servers = 67 simultaneous clients per server
- 5 hour polling interval corresponds to 5% of files changed[2], for which a single server can easily handle 189 simultaneous clients.
- 67 < 189

[1] S. Kent and K. Sriram, "RPKI rsync Download Delay Modeling," March 2013. http://www.ietf.org/proceedings/86/slides/slides-86-sidr-1.pdf

[2] Calculated from numbers in [1].

Questions?