Deployment Considerations in RPKI

Alternative Communication Designs

Deltas! Why?

- rsync integrates both a delta, and a transport, protocol
  - deltas help relying parties
  - but are an easy DoS vector for large repositories
  - to the extent where they may be forced to disallow deltas

- introducing a separate standard to do deltas has advantages
  - allow other transport protocols (http, rsync, carrier pigeons)
  - reduce resource usage when fetching updates
  - reduce propagation times of new objects
  - tell relying parties what has changed

Update notification file

current version	6296
delta X - Y pointer	http://.../delta-X-Y

Server publishes deltas for e.g. every 1, 5, 10, 50, 100, 500, 1000, 5000 etc. increment.

So a RP that has no prior info fetches:

(1) 0-5000, (2) 5000-6000, (3) 6000-6100, (4) 6100-6200, (5) 6200 - 6250, (6) 6250 - 6260, (7) 6270 - 6280, (8) 6280 - 6290, (9) 6290 - 6295, (10) 6295 - 6296, (11) 6296 - 6297, (12) 6297 - 6298

Delta file format

Operation	Key	Data
publish	rsync://..../cert.cer	base64 object
publish	rsync://..../cert2.cer	base64 object
withdraw	rsync://..../cert3.cer	n/a

- Very similar to publication doc PDUs
    - pub server can replay messages to create deltas

Replaying deltas

- Combine all new deltas by scanning them in order
- Withdraw cancels publish for object with key
- New publish overwrites previous for object with key

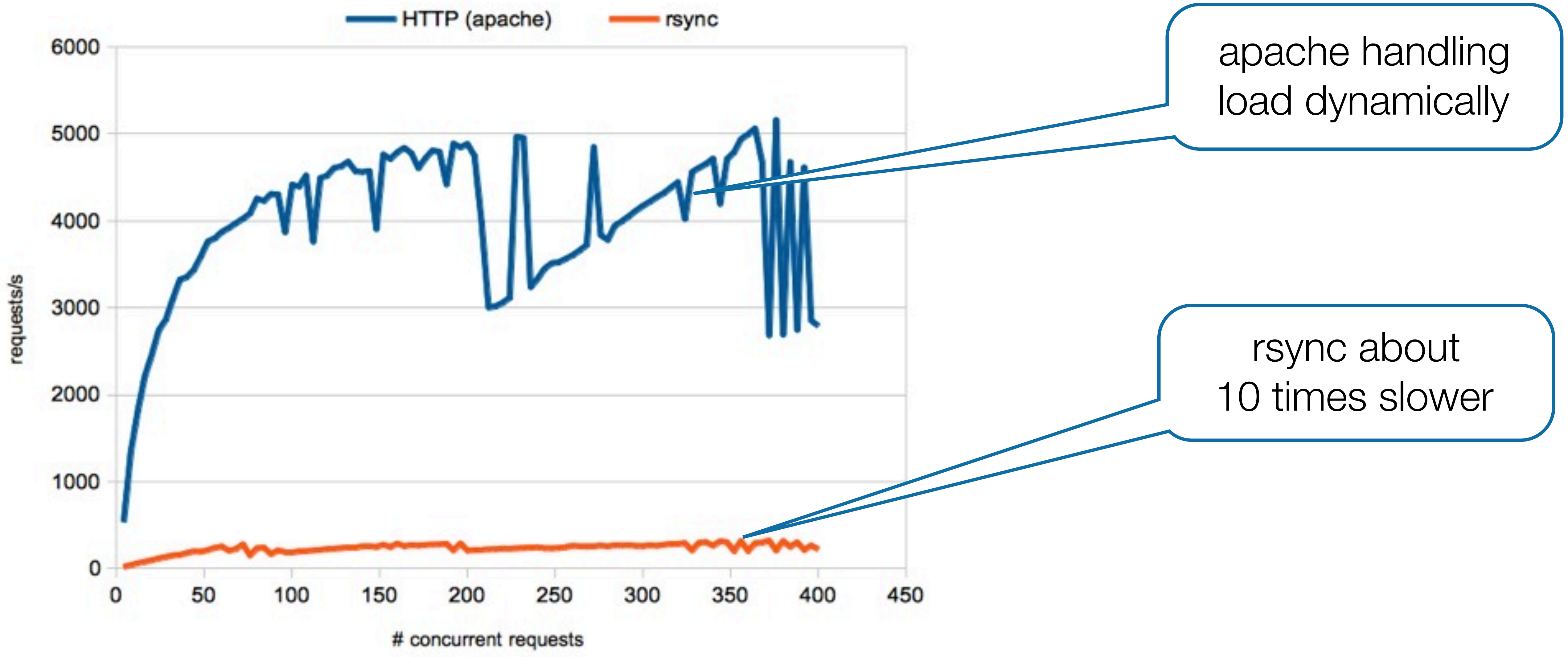
The lab - http performance

```
for n in {1..100}; do
 for i in {1..4}; do

   run apache-benchmark from host i
         with concurrency n

 done
 wait
 # log time
done
```

fetching small files: http vs rsync



- File size 10kB
- Intended for comparison to rsync
  (many other http server benchmarks exist)

Communicating the update notification uri?

	.cer	.mft	.repo	http header for objects
+	Already contains mft pointer	CA knows pub servers	Extra info in mft may confuse RPs	Pub server knows about deltas
-	?	At least 1 extra fetch without knowing about alternative	At least 1 extra fetch without knowing about alternative	untrusted & tied to http protocol

One more thing.. fetch by hash

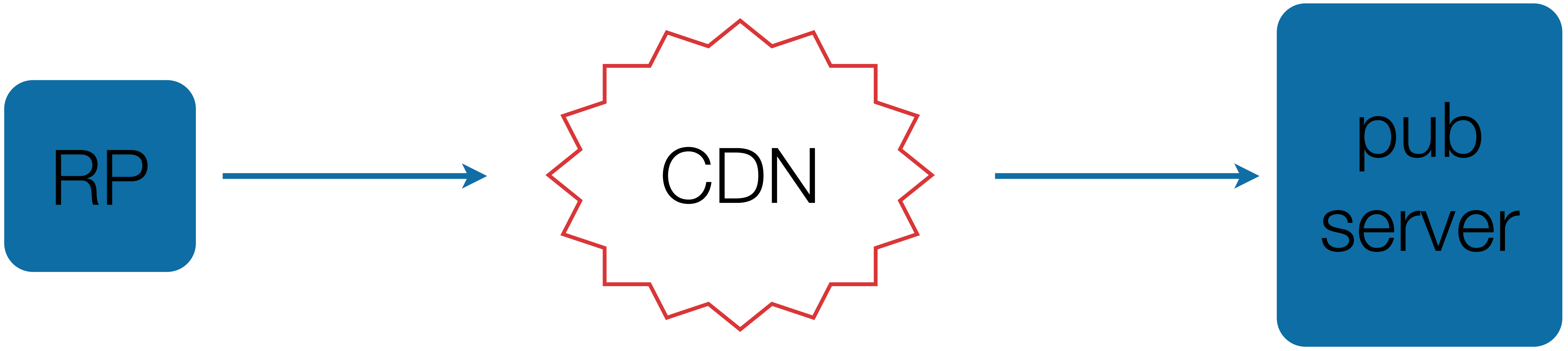
1) Update notification message could contain a pointer like:

    - http://..../byhash/<object-hash>

2) RP can use the fetch by hash base url to fetch objects by the hash mentioned on the manifest

    - will get the exact versions mentioned on mft
    - resources can be cached ‘forever’ by http proxies / cdn

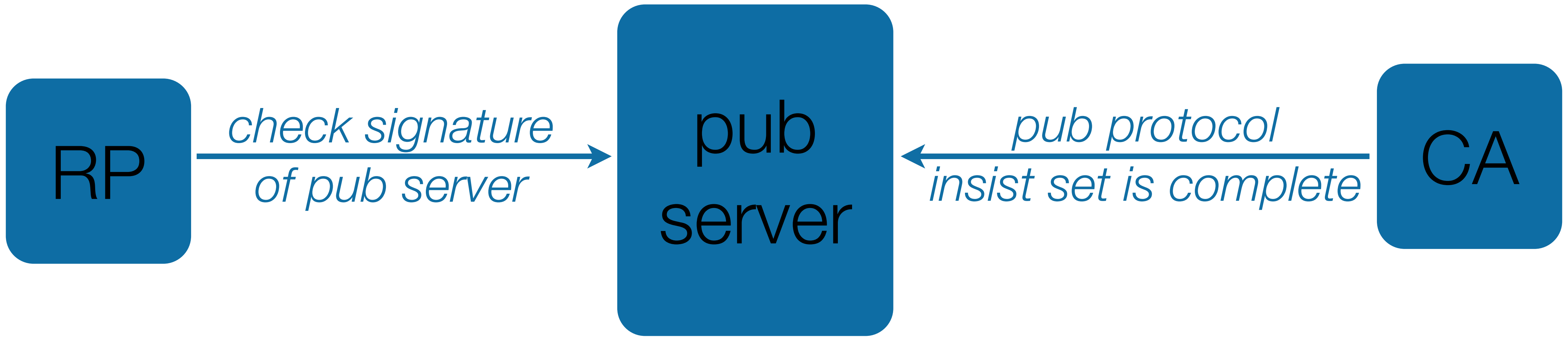
Caching / Content Delivery Networks



resource	cache
update notification	no cache (5 mins?)
repo-object-by-name	no cache (5 mins?)
delta-X-Y	for ever
repo-object-by-hash	for ever

immutable data
write once

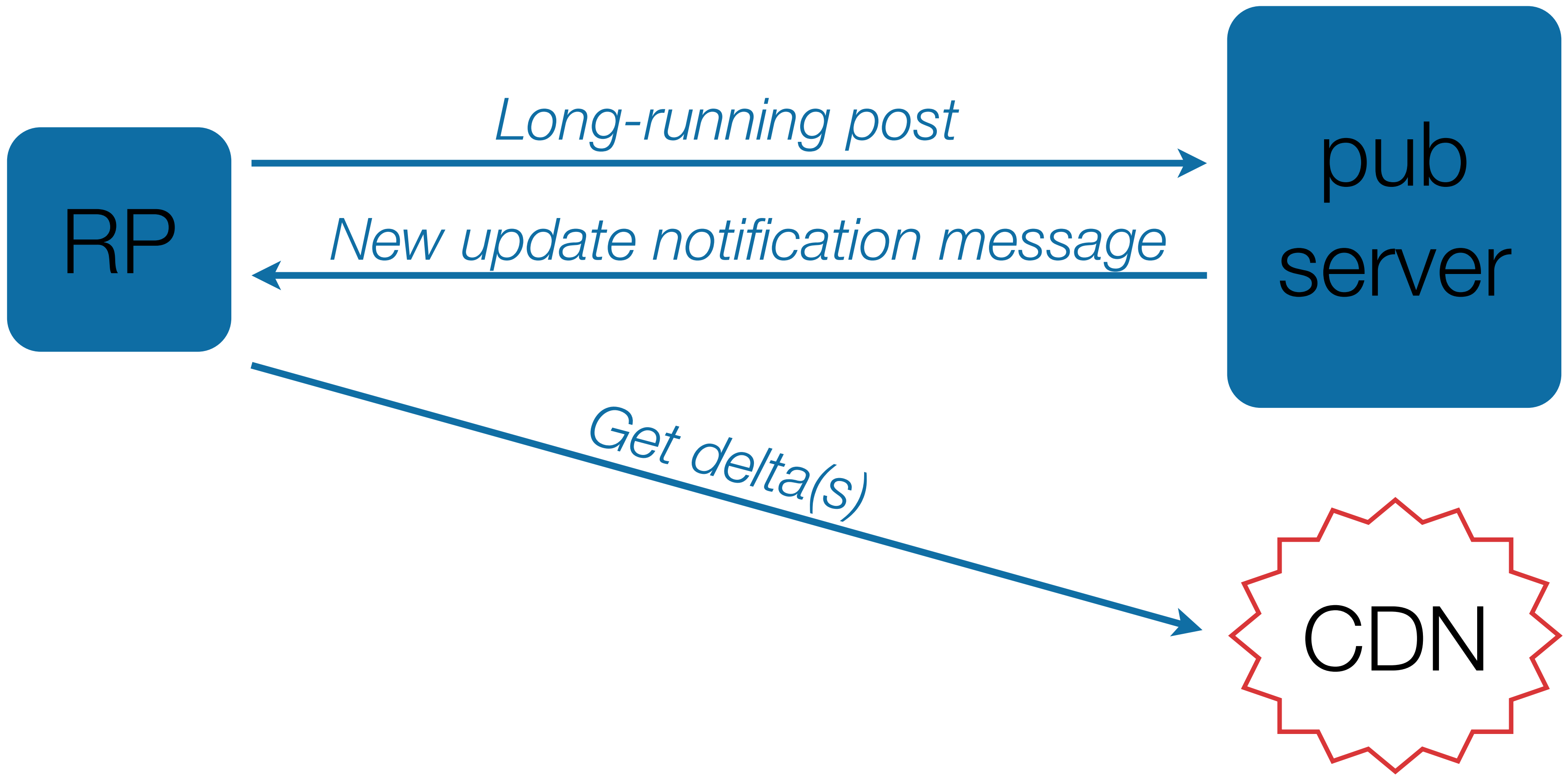
Signing?


RP
check signature of pub server
pub server
pub protocol insist set is complete
CA

- ➡ Makes man in the middle detectable
- ➡ Communicating publication server key, and key rolls is problematic
  - ➡ encode in pointer in .cer?

- ➡ Objects are already signed.. so is this really worth the pain?

Pub-sub



- ➡ Used by high load dynamic websites (like twitter)
- ➡ Makes sub-minute notifications possible
- ➡ Optional, RP can still fall back to polling

Deltas & http selling points 1/2

- Reduce the load / dependency on the server
  - Letting clients work out deltas scales better
  - Immutable deltas allow for caching and using CDNs
  - Write-once simplifies implementation: write and forget
- Deltas are ‘transactional’ so should be consistent

Deltas & http selling points 2/2

- Possible further optimisations
  - Detect man-in-the-middle
  - Fast updates (router keys? bgp freshness)
- Quality of relying party code
  - Can use native libraries
  - Get clear error messages
  - Know exactly what was changed (avoid unnecessary crypto)