Experiments in DNSSEC

George Michaelson

ggm@apnic.net

APNIC

Experiments in DNSSEC

Experiments in DNSSEC

- We've got a system which can perform thousands of measurements per day, on random Internet users
  - It measures DNS (UDP and TCP), and it measures HTTP (TCP)

Experiments in DNSSEC

- We've got a system which can perform thousands of measurements per day, on random Internet users
  - It measures DNS (UDP and TCP), and it measures HTTP (TCP)
- We've used it to do IPv6 measurements
- We've used it to do routing visibility on IPv4 blocks
- What else can we do with this experimental rig?

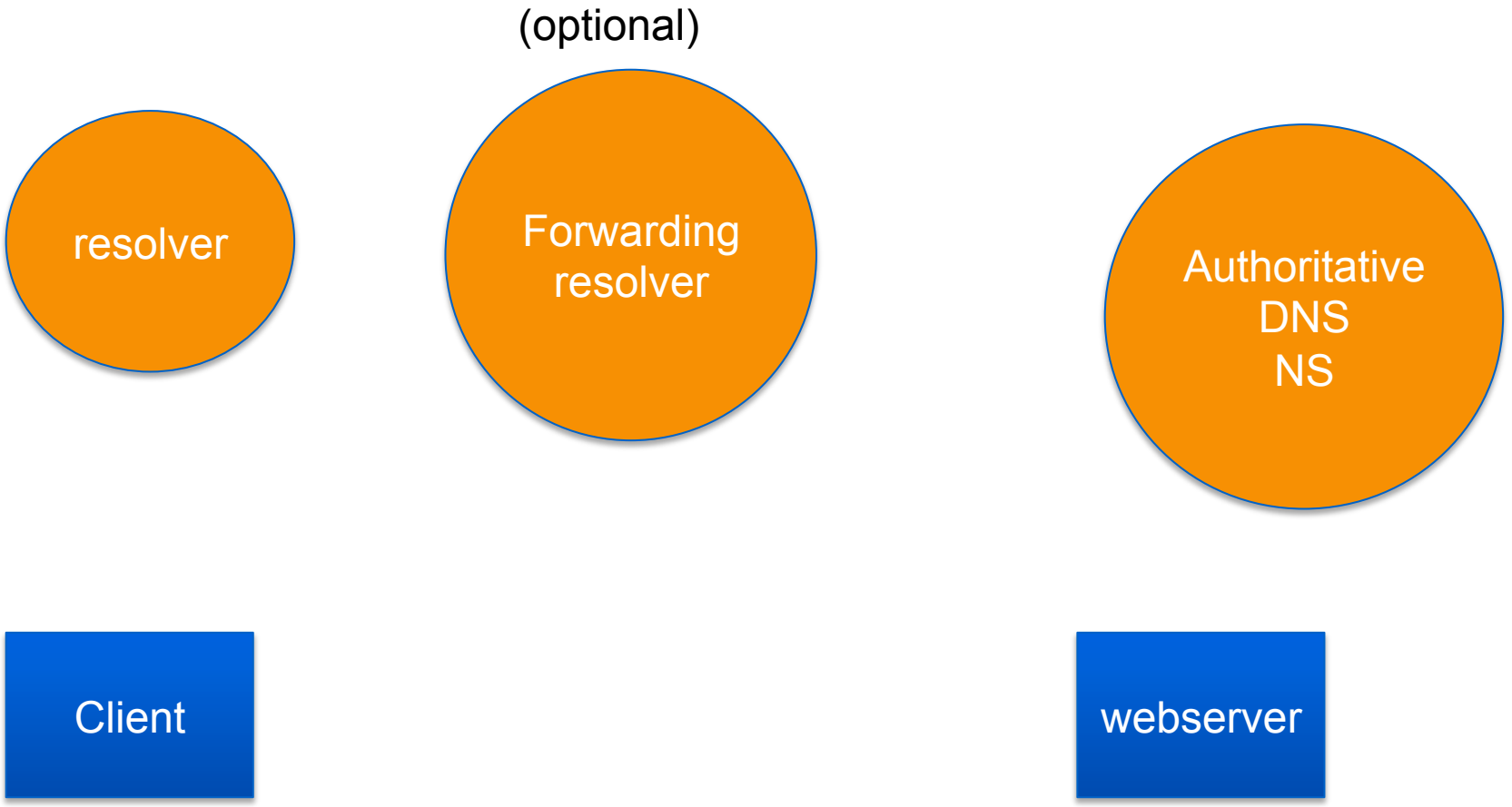
Experiments in DNSSEC

- We've got a system which can perform thousands of measurements per day, on random Internet users
  - It measures DNS (UDP and TCP), and it measures HTTP (TCP)
- We've used it to do IPv6 measurements
- We've used it to do routing visibility on IPv4 blocks
- What else can we do with this experimental rig?

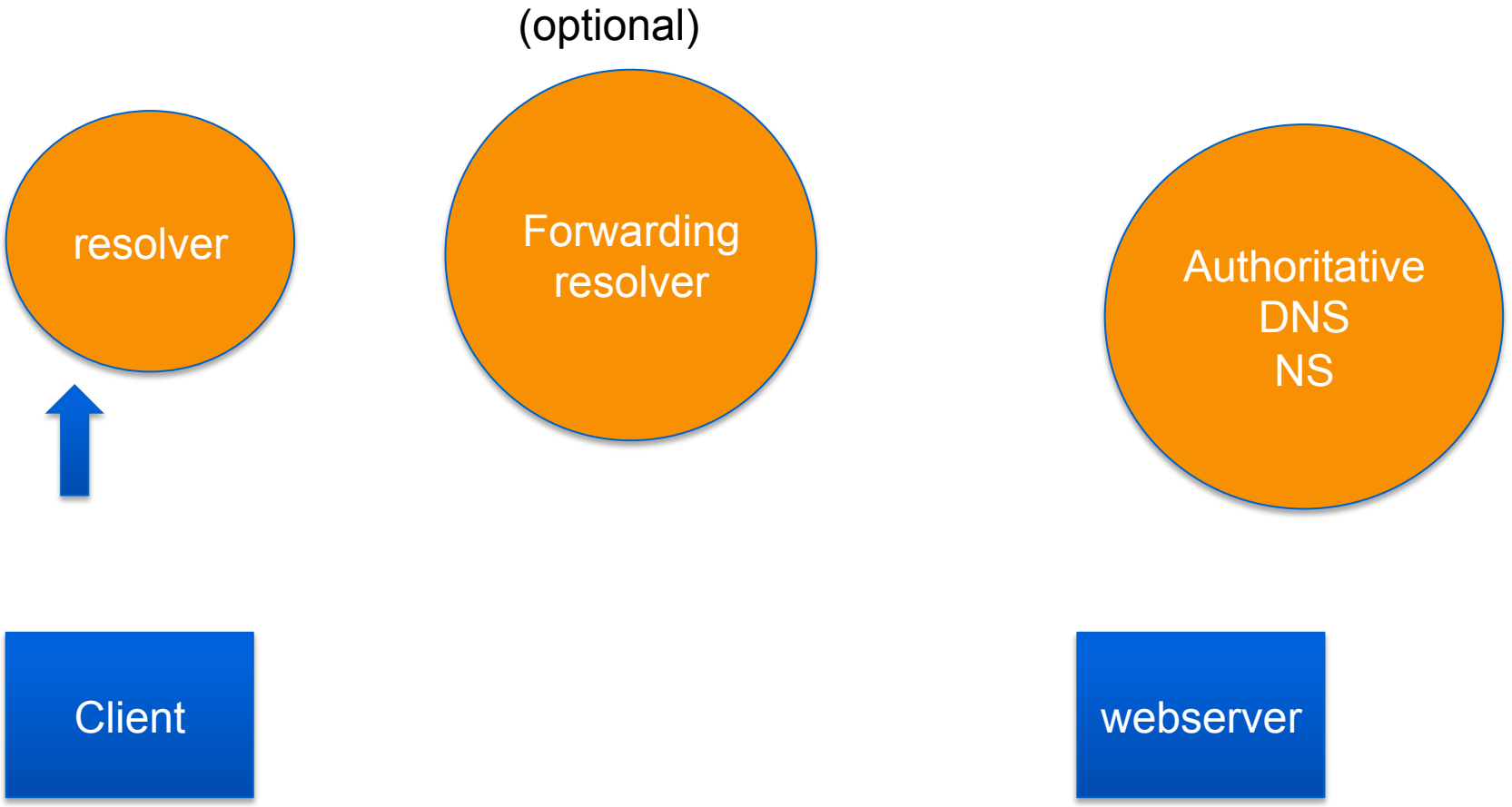
- Can we use it to tell us something about DNSSEC?

Clean State dns/web fetch

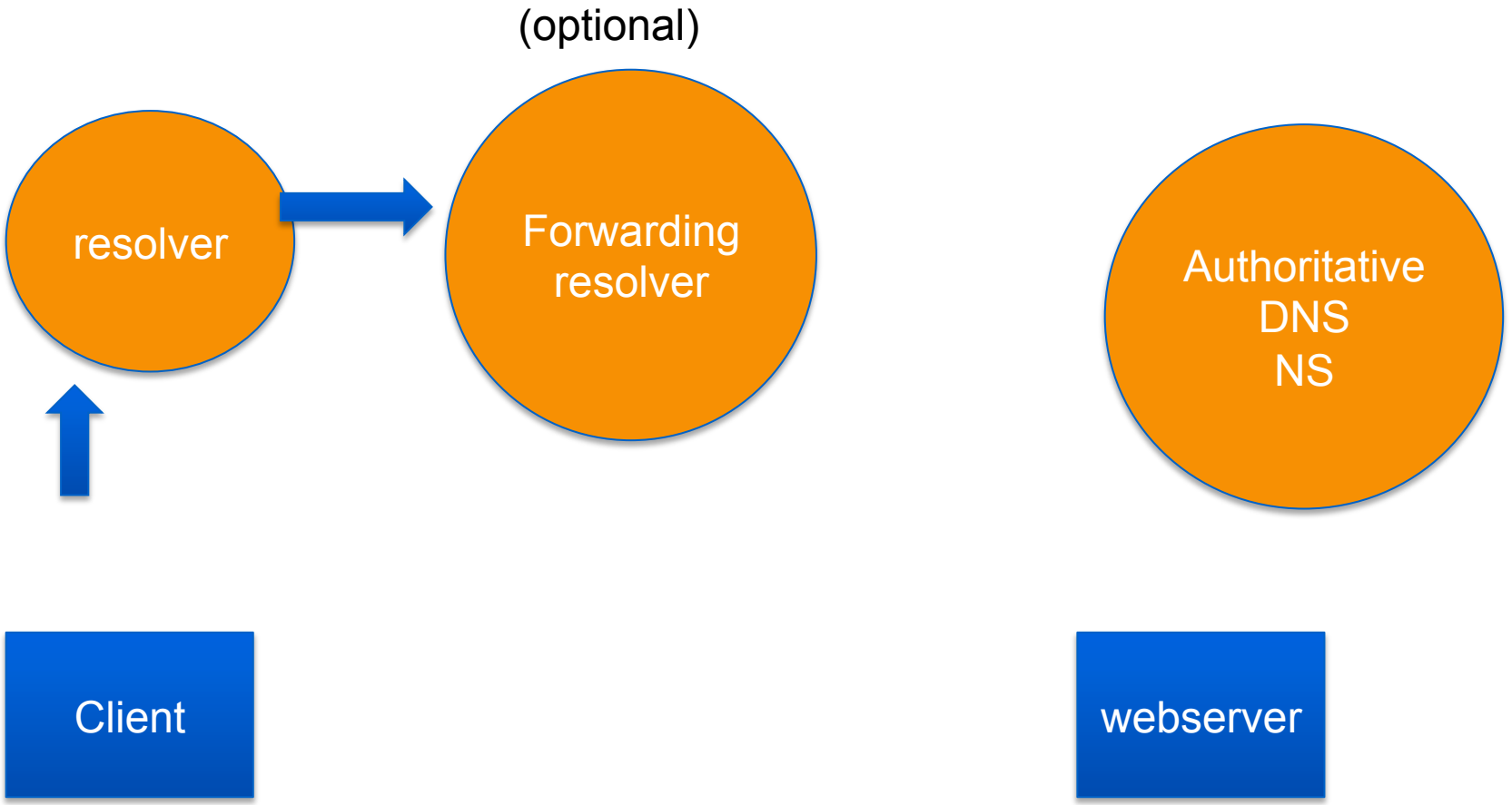
Clean State dns/web fetch



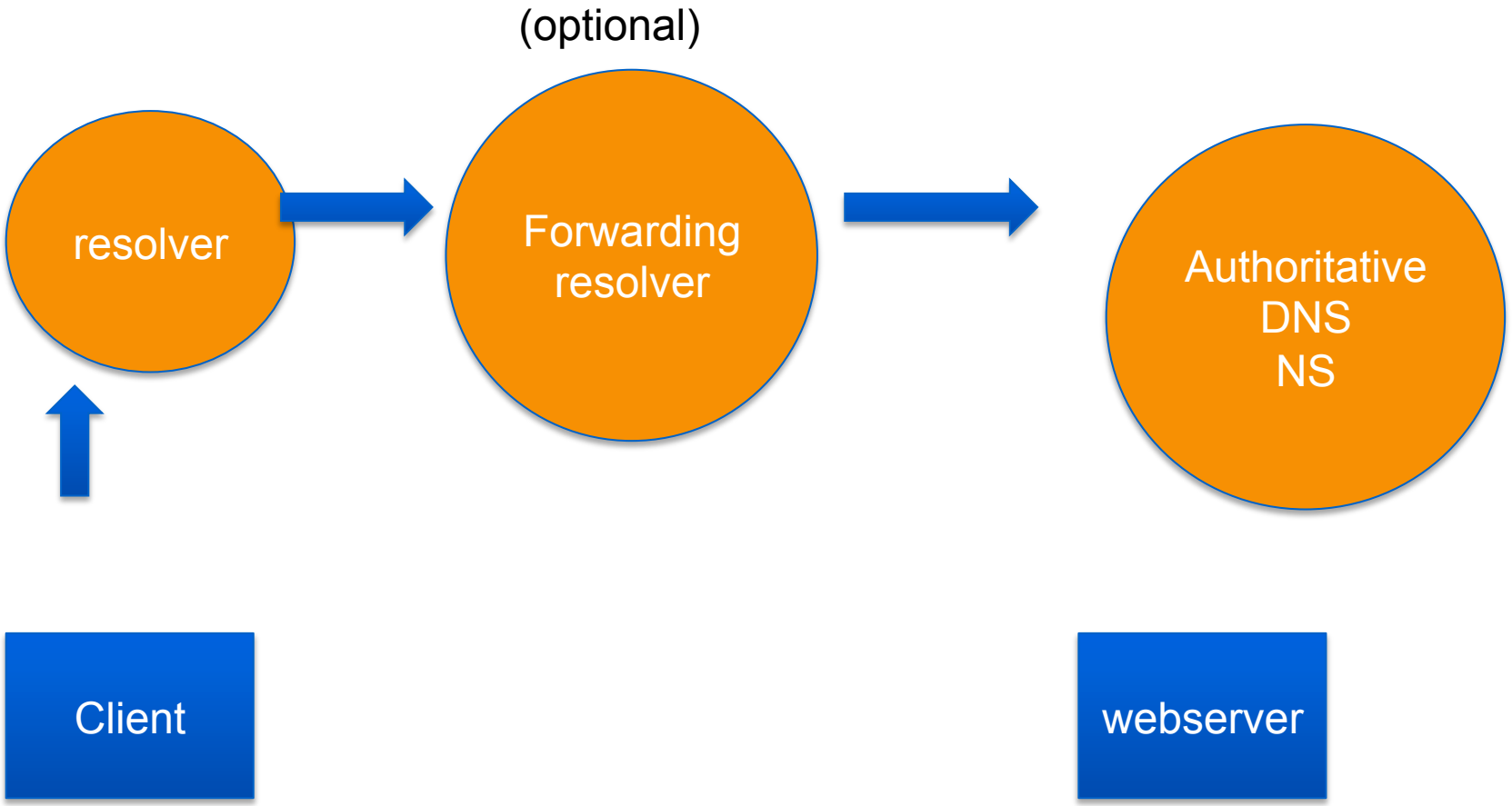
Clean State dns/web fetch



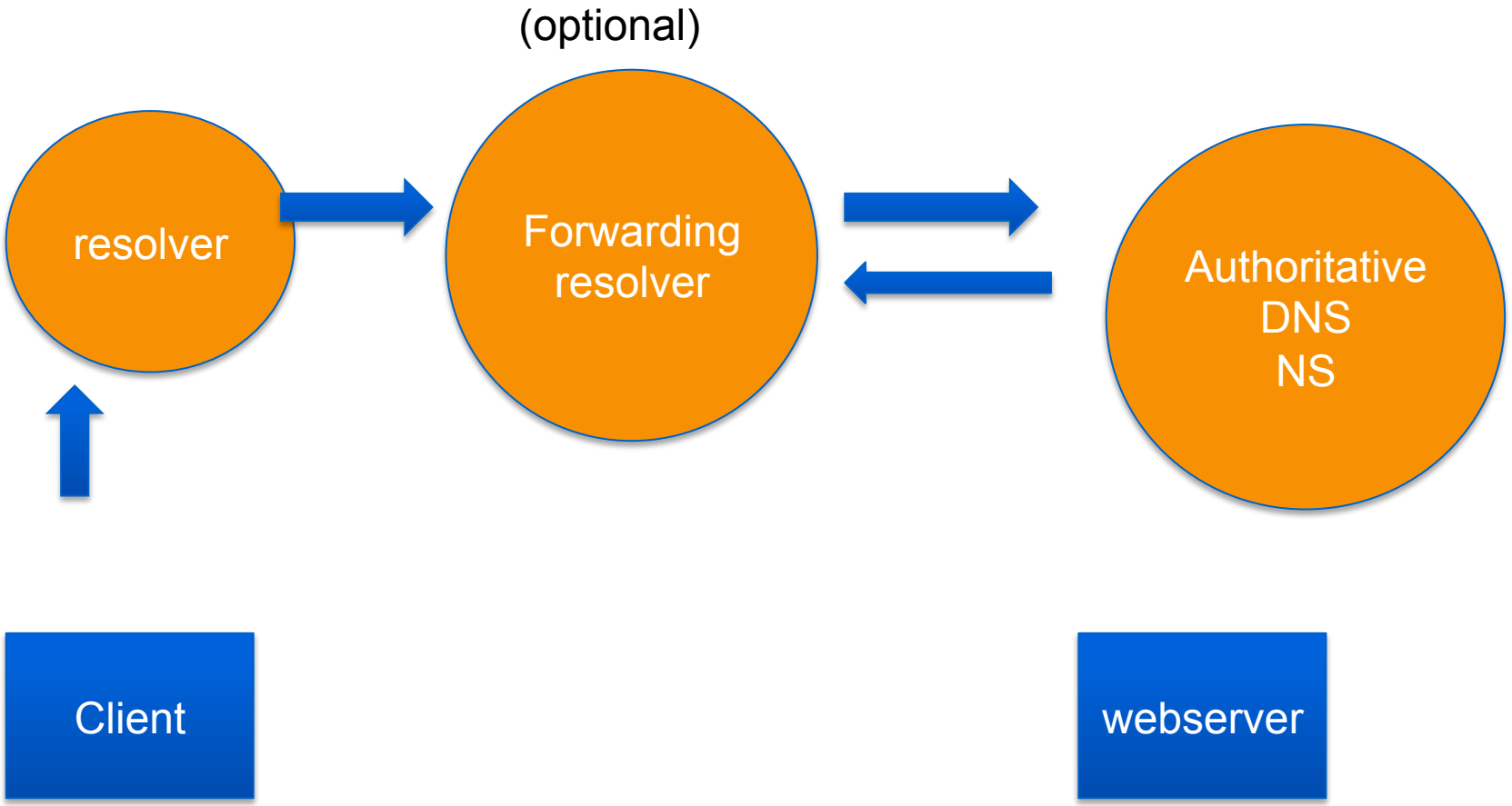
Clean State dns/web fetch



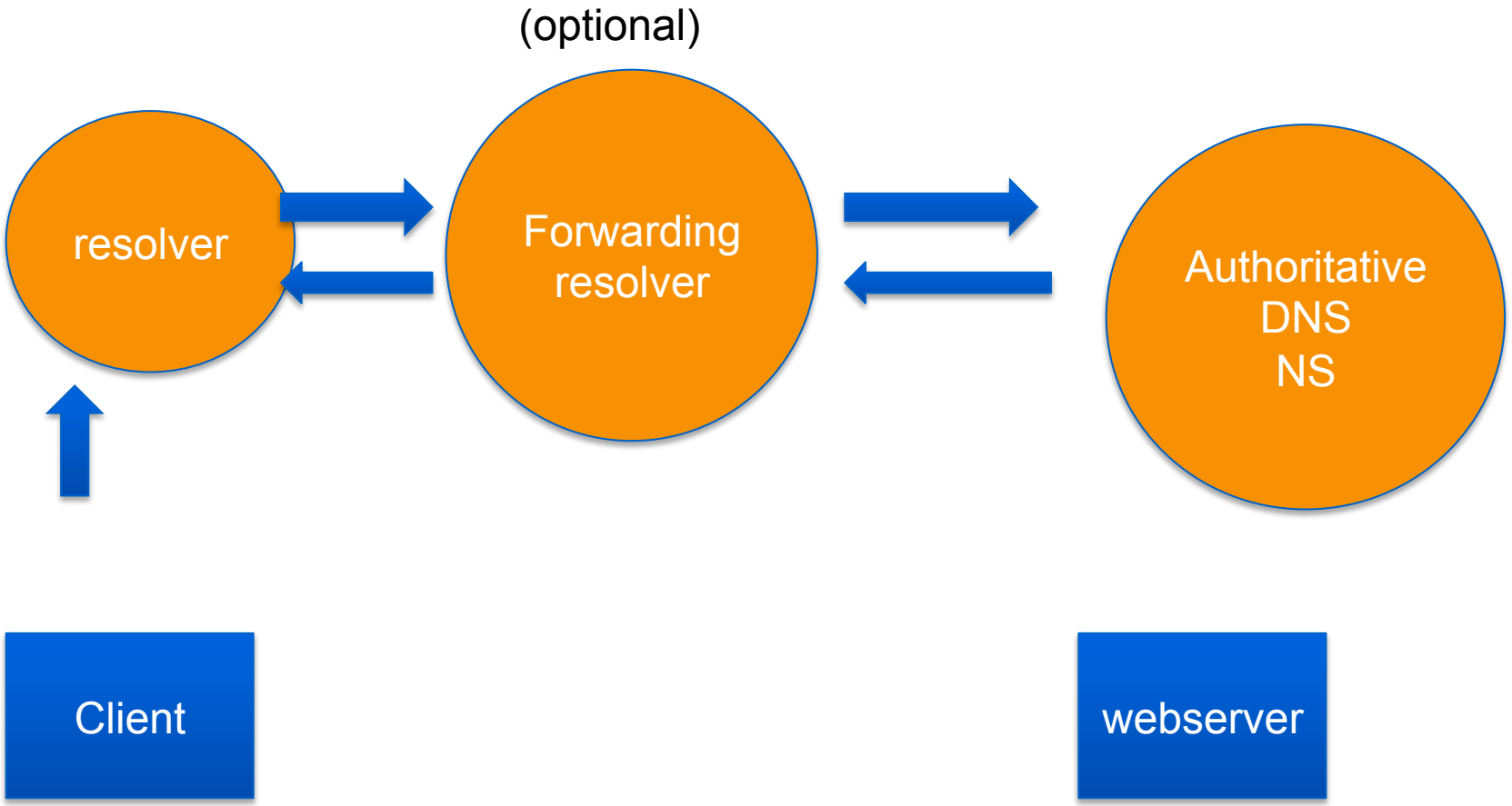
Clean State dns/web fetch



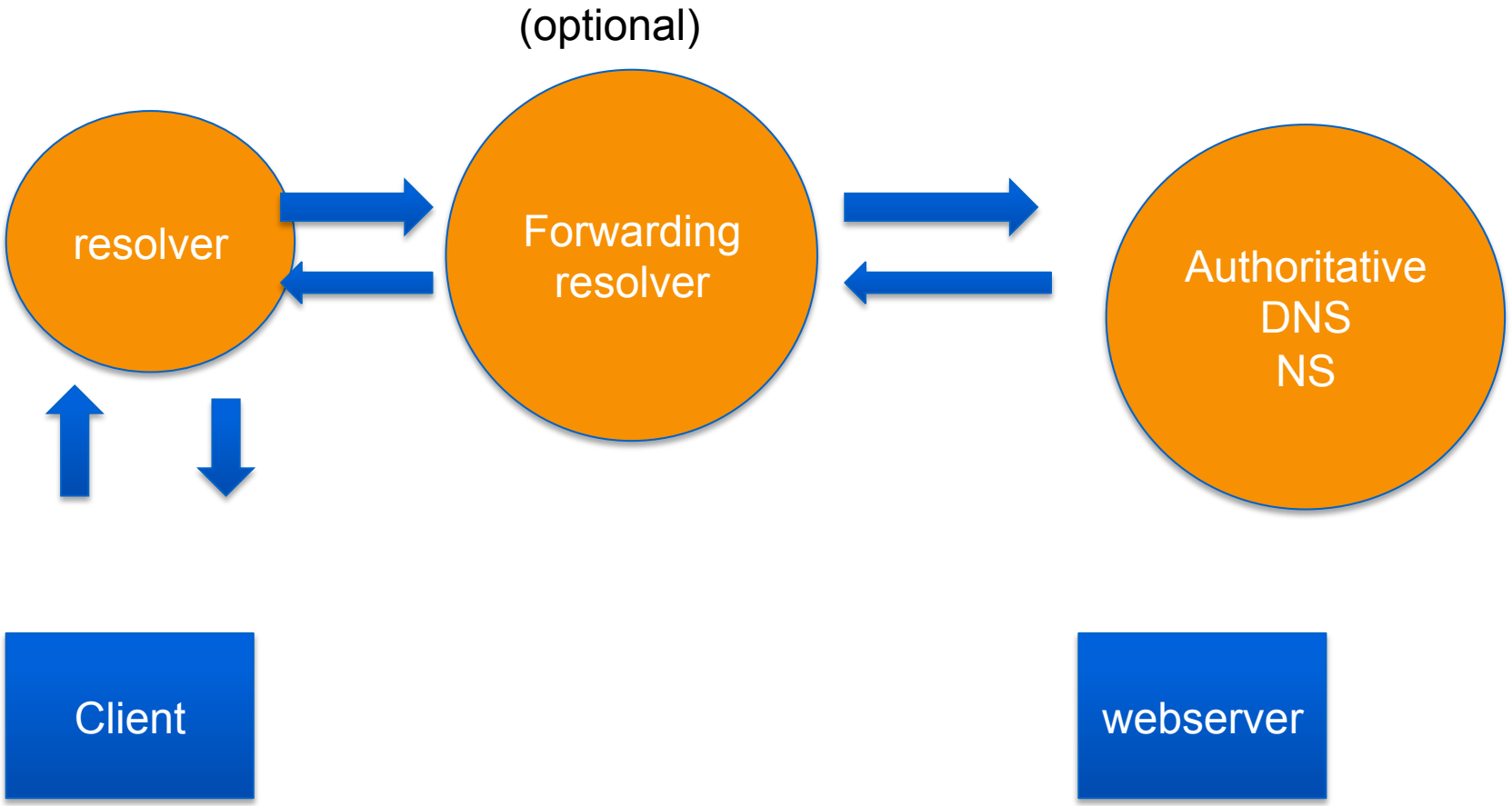
Clean State dns/web fetch



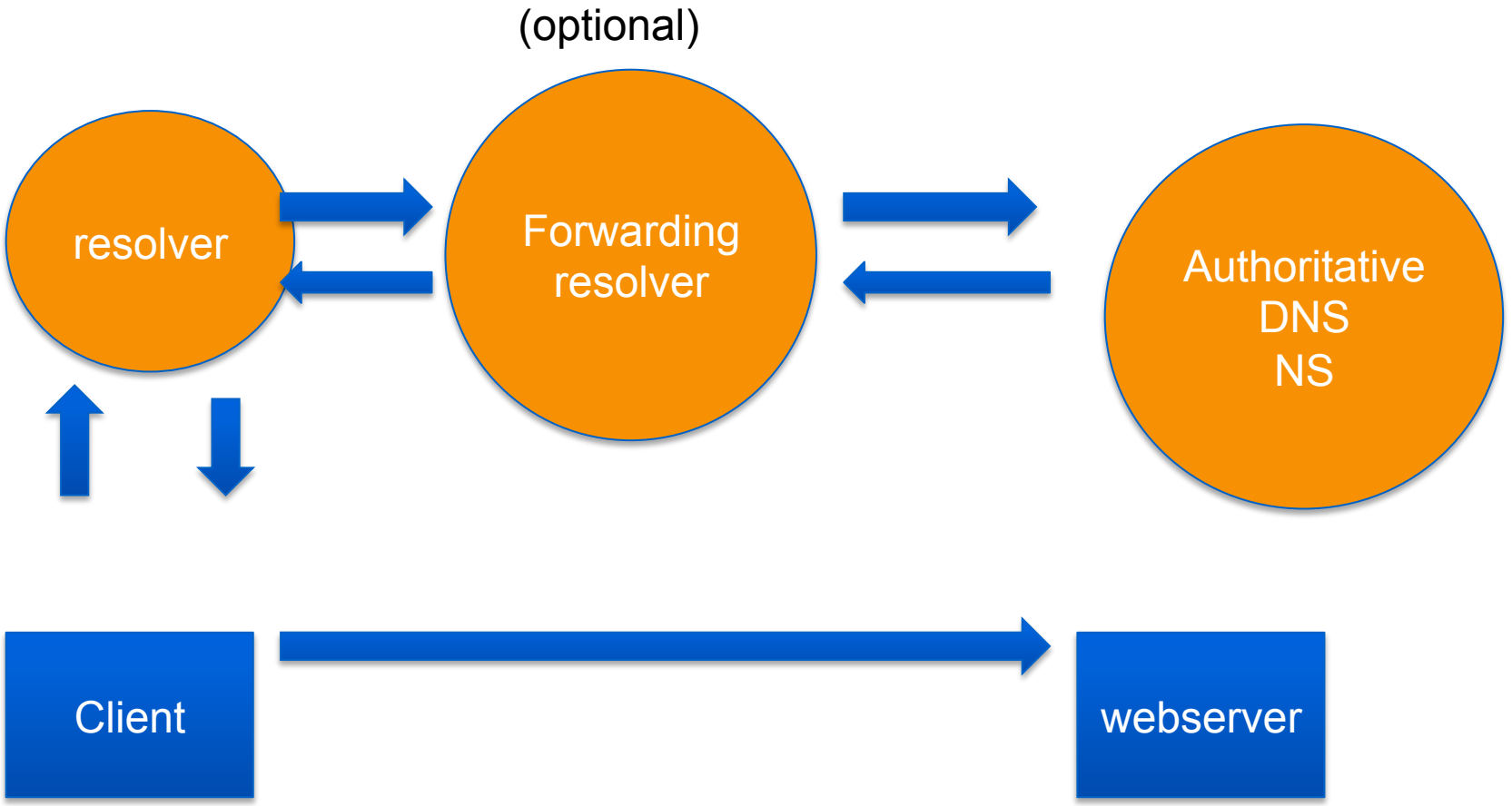
Clean State dns/web fetch



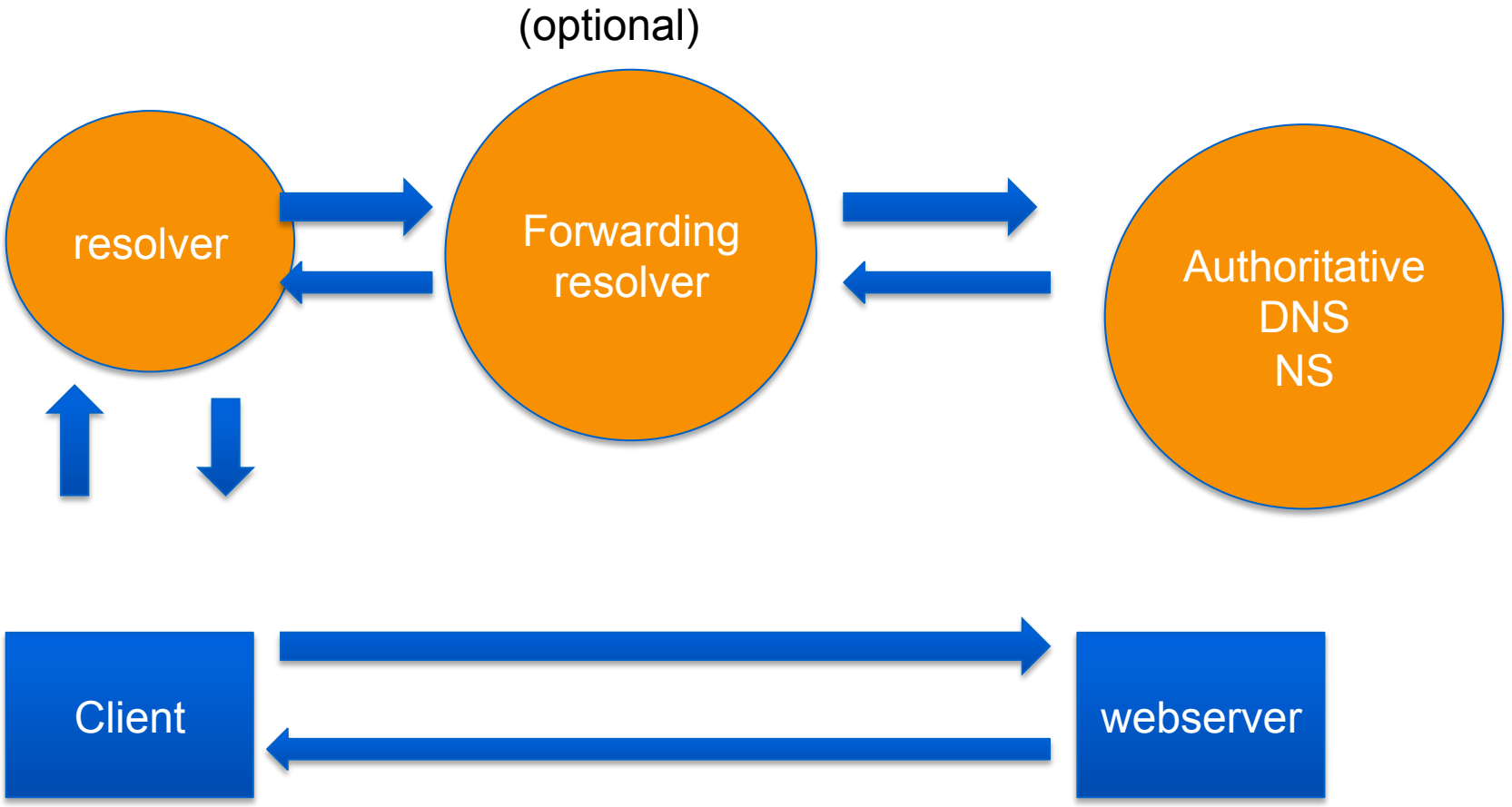
Clean State dns/web fetch



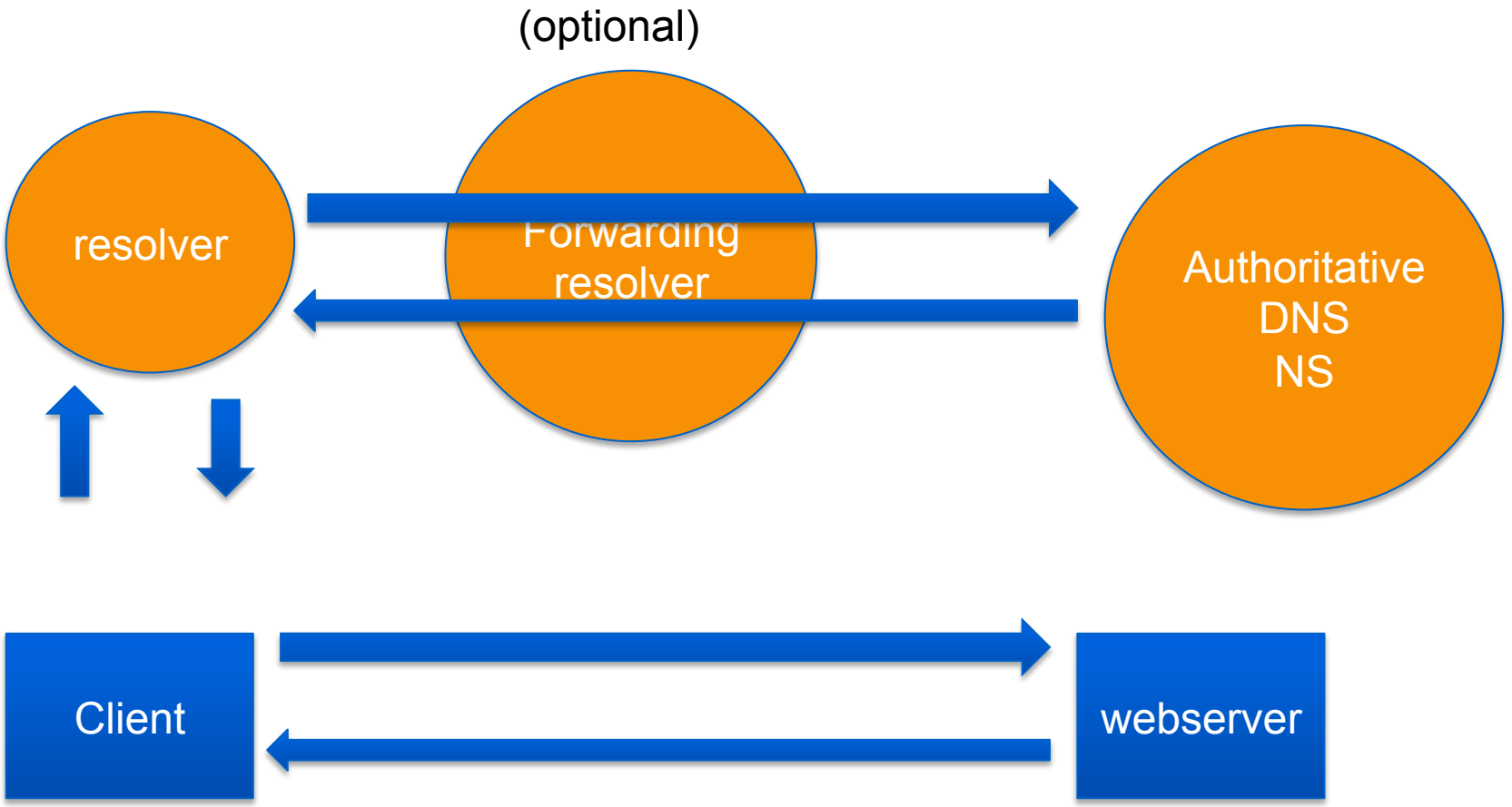
Clean State dns/web fetch



Clean State dns/web fetch



Clean State dns/web fetch

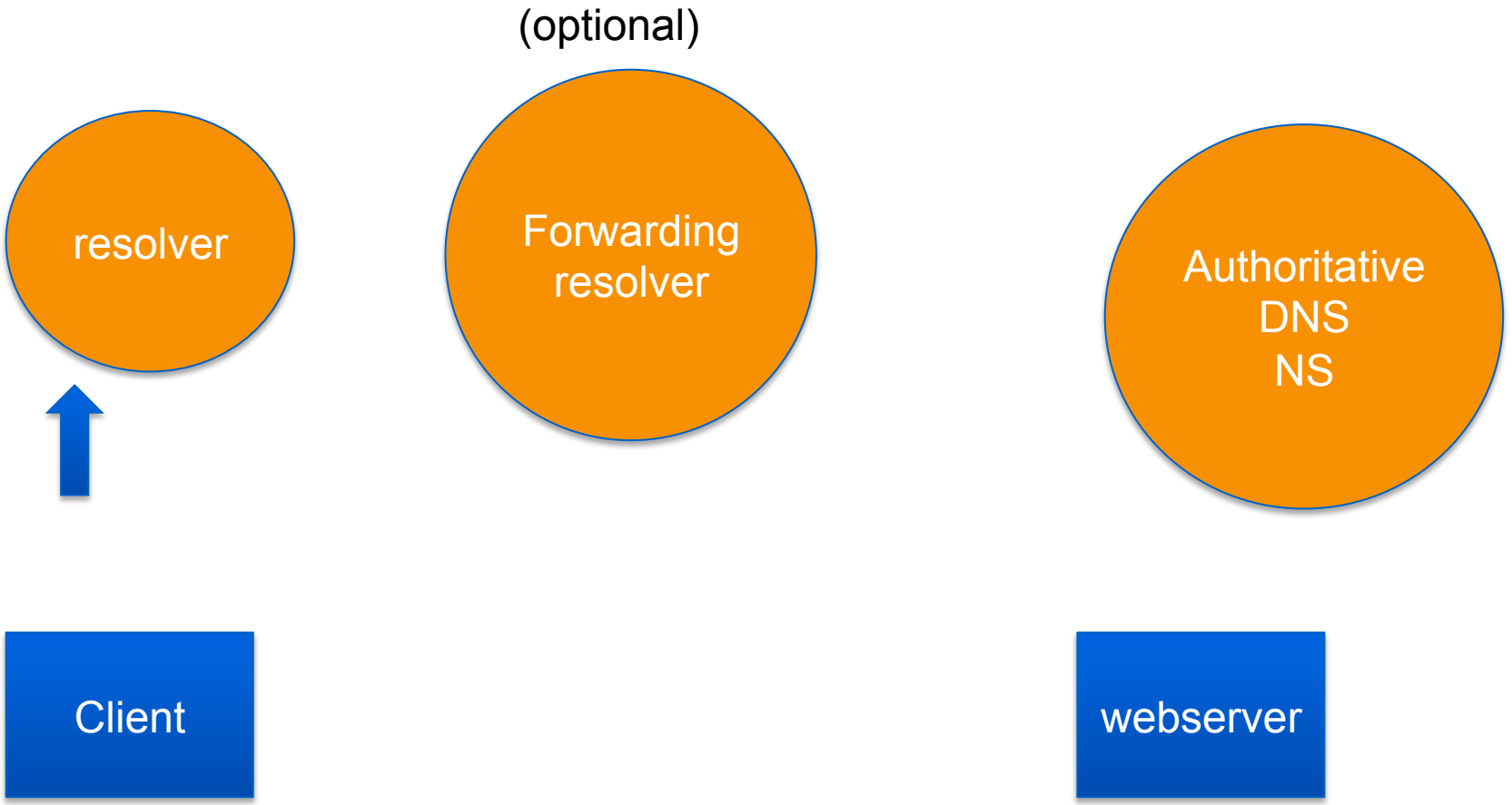


Clean State dns/web fetch

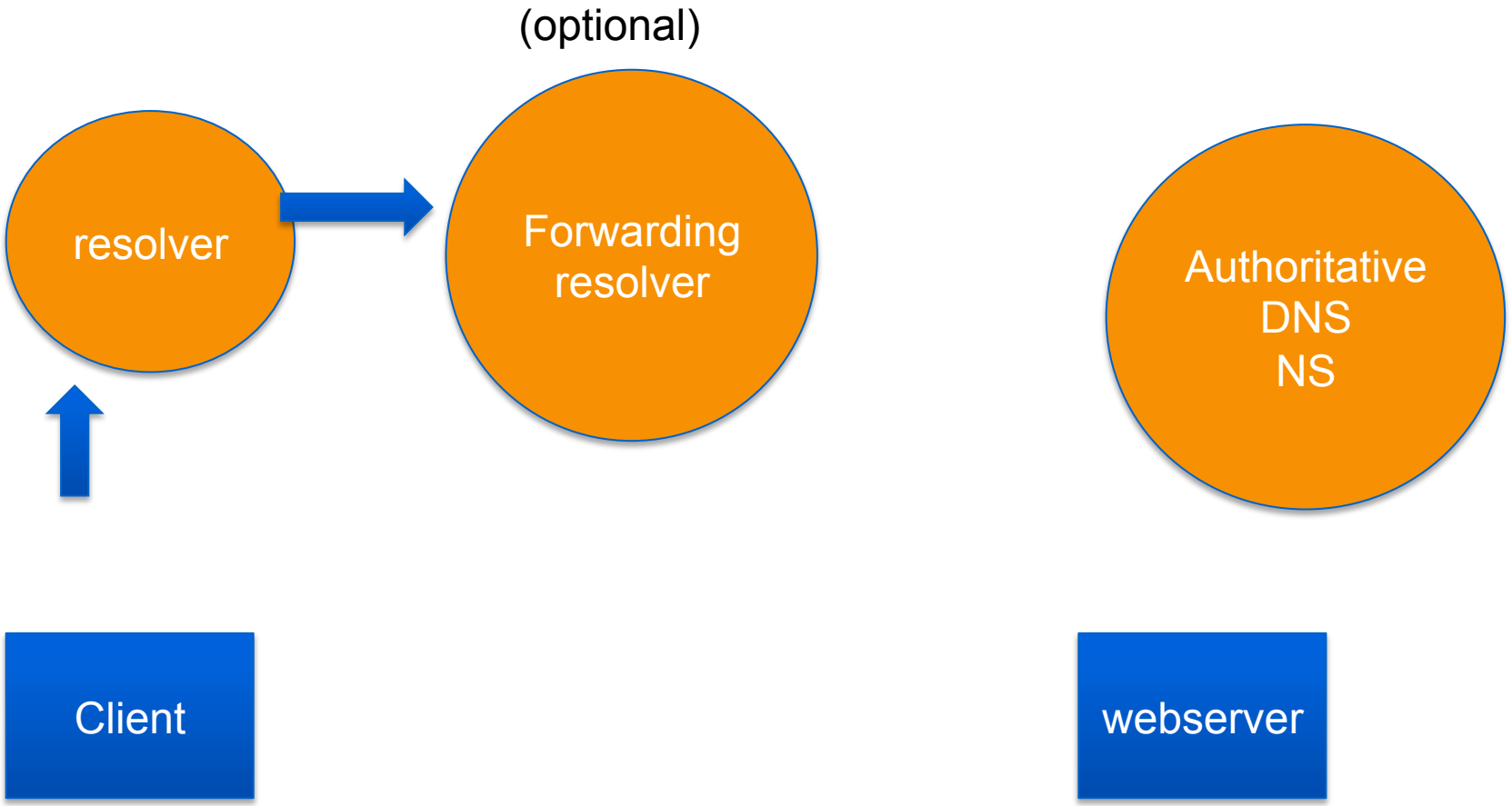
- Before web, DNS has to find the A/AAAA bindings
- Which means the DNS service has to traverse the namespace (from the root) to find the NSERVER of the domain
  - And then ask the NSERVER for the A/AAAA information
- The traversal is cached, so not all web fetches demand a hunt from the root
  - Cached state can be set by other peoples queries
- And it works!

Cached State dns/web fetch

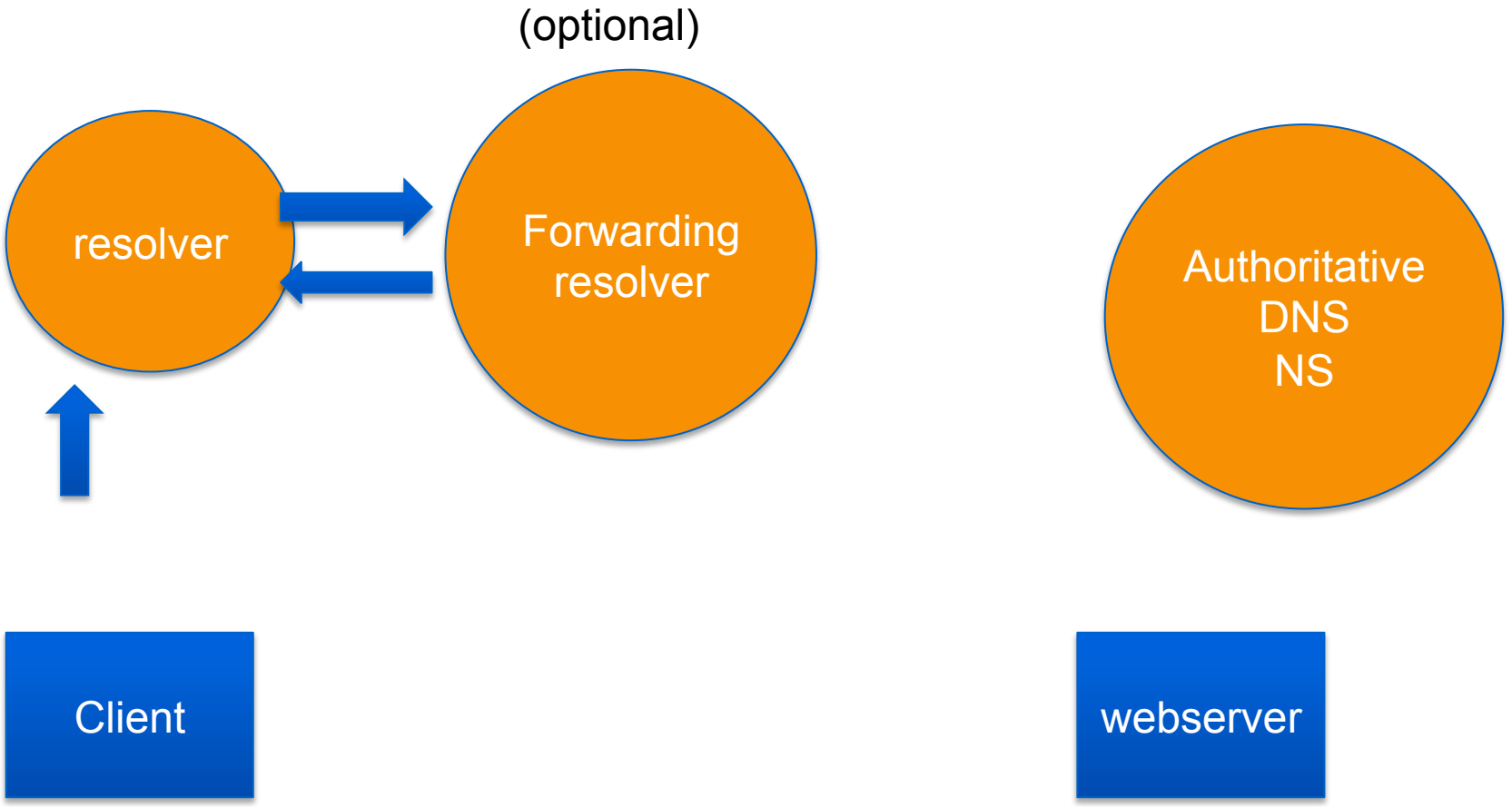
Cached State dns/web fetch



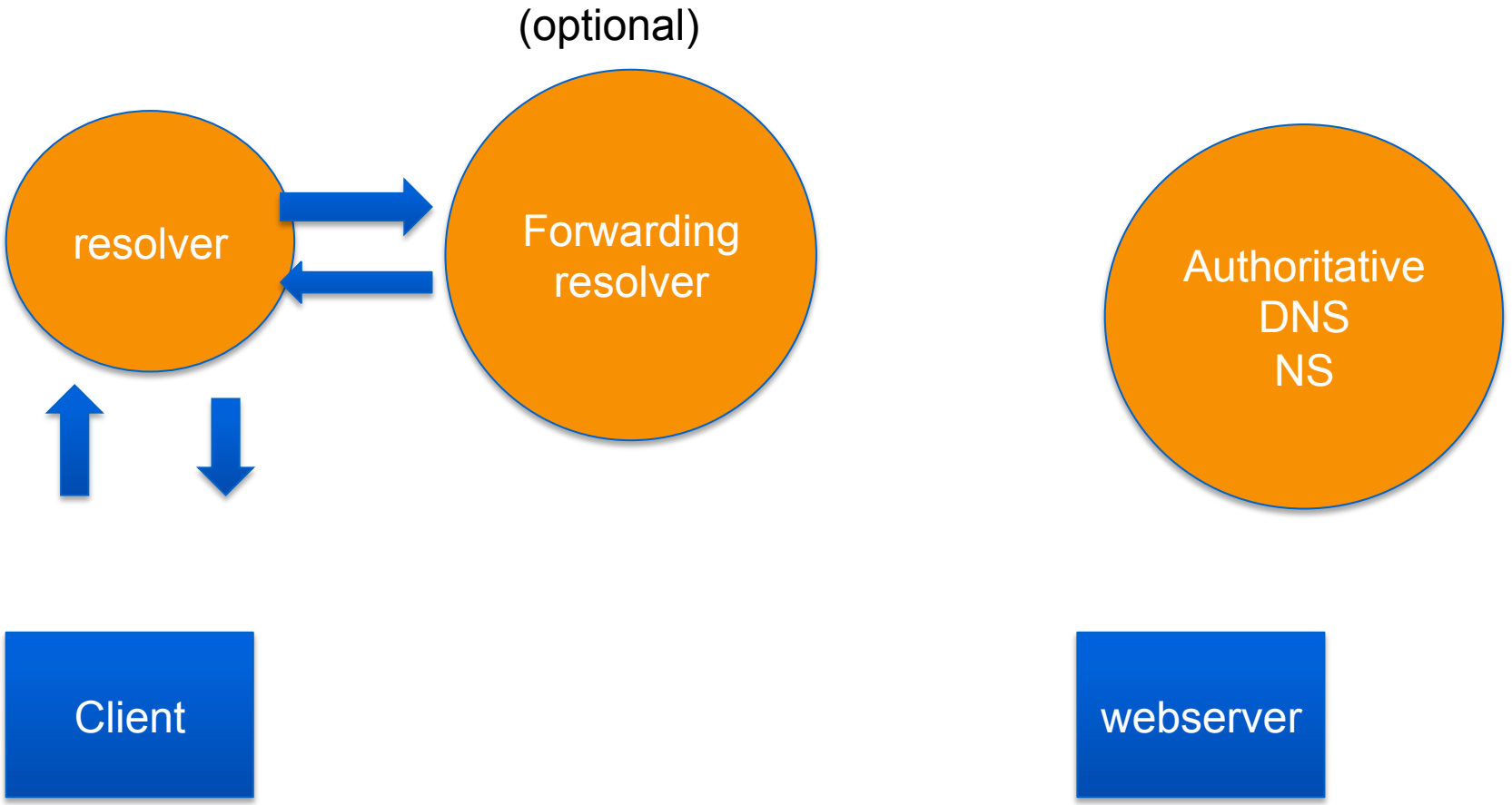
Cached State dns/web fetch



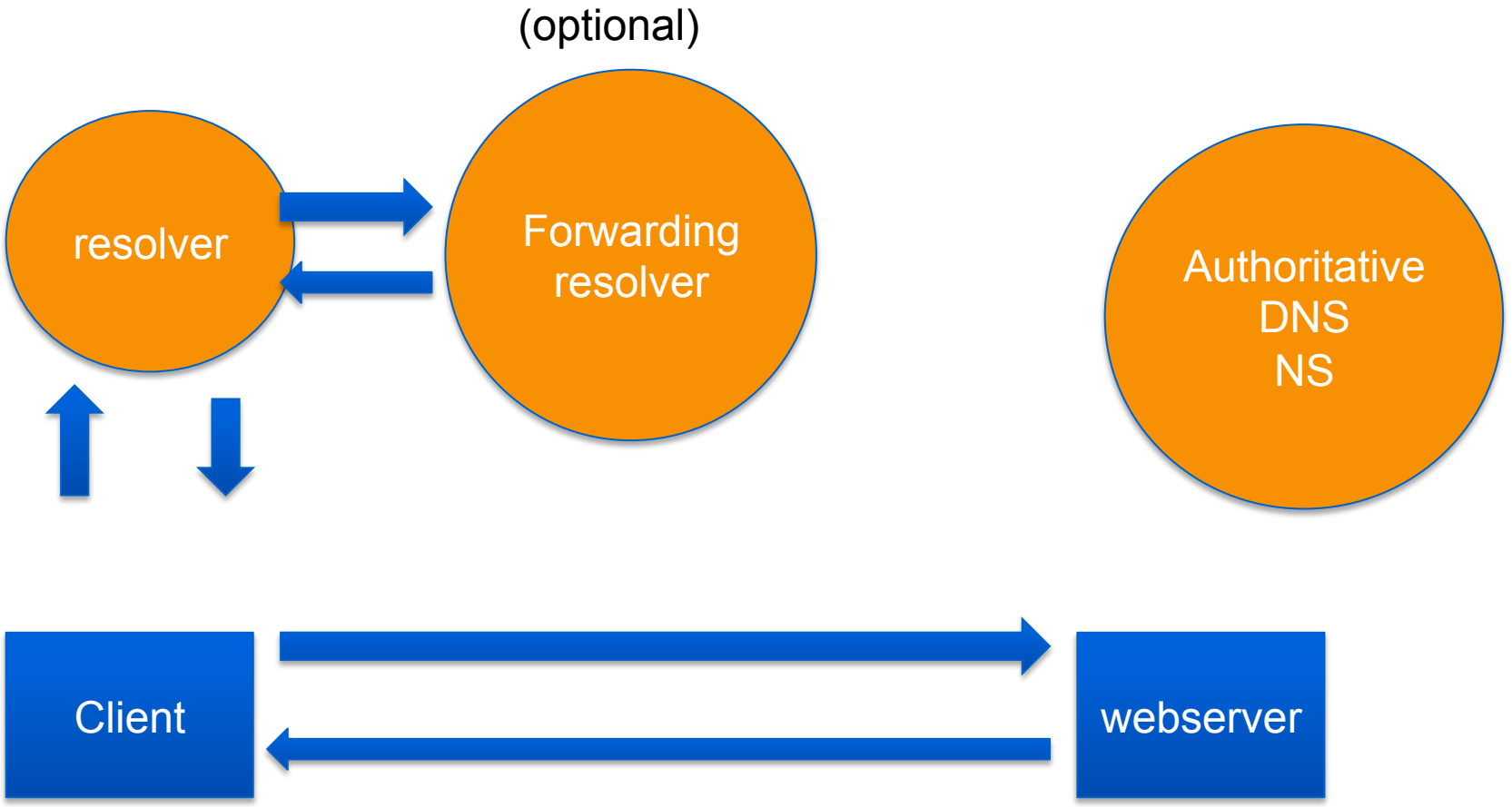
Cached State dns/web fetch



Cached State dns/web fetch



Cached State dns/web fetch



DNS is indirect, cached

DNS is indirect, cached

- First client, establishes state in the DNS resolver chain relating to the domain
- Subsequent clients may or may not be exposed to information along the DNS namepath
  - Cached NS
  - Cached A/AAAA
  - Cached RRSIG/DS/NSEC
- Since resolvers can pass through forwarders, this state can indirectly cache along a DNS resolver chain from the authority point

DNS is indirect, cached

- Can we detect when a DNS resolver is ‘in the path’?
- Can we detect when a DNS resolver has cached state?

- Can we detect when a DNS resolver has DNSSEC enabled?

- Can we detect when a client has validation enabled?

DNSSEC adds data, queries

- Public, private digital signatures over DNS data
- Public keys used to verify, fetched from parent in DNSKEY requests which fetch DS records
- Resources are signed with RRSIG signatures via the private key, RRSIG passed in ‘additional’ data fields of request
- NSEC records provide chain over records in the zone so you can tell if you have had something hidden from you, or if it really doesn’t exist

DNSSEC is indirect, but only partly cached

DNSSEC is indirect, but only partly cached

- DNSSEC has a characteristic signature in the DNS query sequence

1. You signal you can ‘DO’ DNSSEC in the query flags
2. You receive RRSIG in the additional section of a reply
3. If you want to validate, you have to explicitly ask for DNSKEY for the DS records.

- If you see a sequence of {A,DNSKEY/DS} in a short space of time from the same IP, you have high confidence the IP is doing DNSSEC fetches.
- Once fetched, can be re-used for some time

DNS complications

- People configure more than one resolver
  - You can see more than one source IP asking questions for one client
- Sometimes, the client software does ‘scattergun’ fetches to all the listed NS
  - You can see more than one authoritative NS being queried by a system you’ve never seen before (ie, no cached state)
- DNS re-queries if you don’t answer in time
  - ‘udp is unreliable’ history?

DNSSEC complications

- People don't always configure ALL their resolvers to do DNSSEC
  - You can see the same query, signalling DO and not signalling DO
  - You can see the same query, asking associated DNSKEY/DS
- Forwarder chains can create cached state
  - If the forwarder is itself DNSSEC enabled, it learns at least the RRSIG
  - If the forwarder is not DNSSEC enabled directly, it can forward on both DNSSEC and non-DNSSEC enabled clients

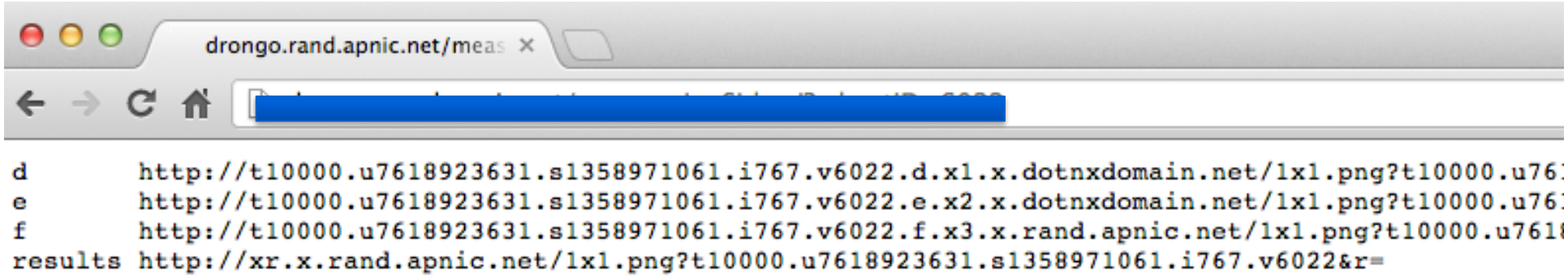
Serverside complications

- DNSSEC configuration is fiddly
- New commands to be run to bootstrap keys
  - Keys associate tightly with domains
  - Keys have lifetimes
  - Keys are themselves complex multipart data
- New data to be lodged in parent domain
  - DS record has to be placed in parent zone, signed over by parent
  - DS records are complex multipart data
  - Registries don't necessarily honour the 'file' format you have in-hand
- New data to be included in your zonefile
  - You have to re-sign over your zone on each edit.

Serverside complications

- Seriously:
- we spent a horrendous amount of time fixing up stupid mistakes in the key application to the zone
  - ZSK and KSK aren't easily distinguished by filename in BIND 9
  - Lots of keys in one flat /etc/named/keys directory didn't help
- Every zone edit demands a re-sign
  - Forget to sign, now have zonefile and zonefile.signed out of sync
- This process is highly ameanable to fat finger trouble.

Anatomy of a DNSSEC Experiment



Anatomy of a DNSSEC Experiment

- Configure five domains

1. DNSSEC enabled, validly signed:
2. DNSSEC enabled, invalidly signed:
3. DNSSEC enabled, IPv6 only NS:
4. DNSSEC enabled IPv6 only NS, extra large response required (> 1500 bytes):
5. No DNSSEC

- Get clients to fetch from first 4, return results on 5

Anatomy of a DNSSEC Experiment

- Configure five domains

1. DNSSEC enabled, validly signed: *checks if clients resolver is enabled to do DNSSEC*
2. DNSSEC enabled, invalidly signed: *checks if the clients resolver is enabled for validation*
3. DNSSEC enabled, IPv6 only NS: *checks if the clients resolver can do DNS over IPv6*
4. DNSSEC enabled IPv6 only NS, extra large response required (> 1500 bytes): *checks if the clients resolver can handle pMTU*
5. No DNSSEC *(takes results)*

- Get clients to fetch from first 4, return results on 5

Invalid DNSSEC?

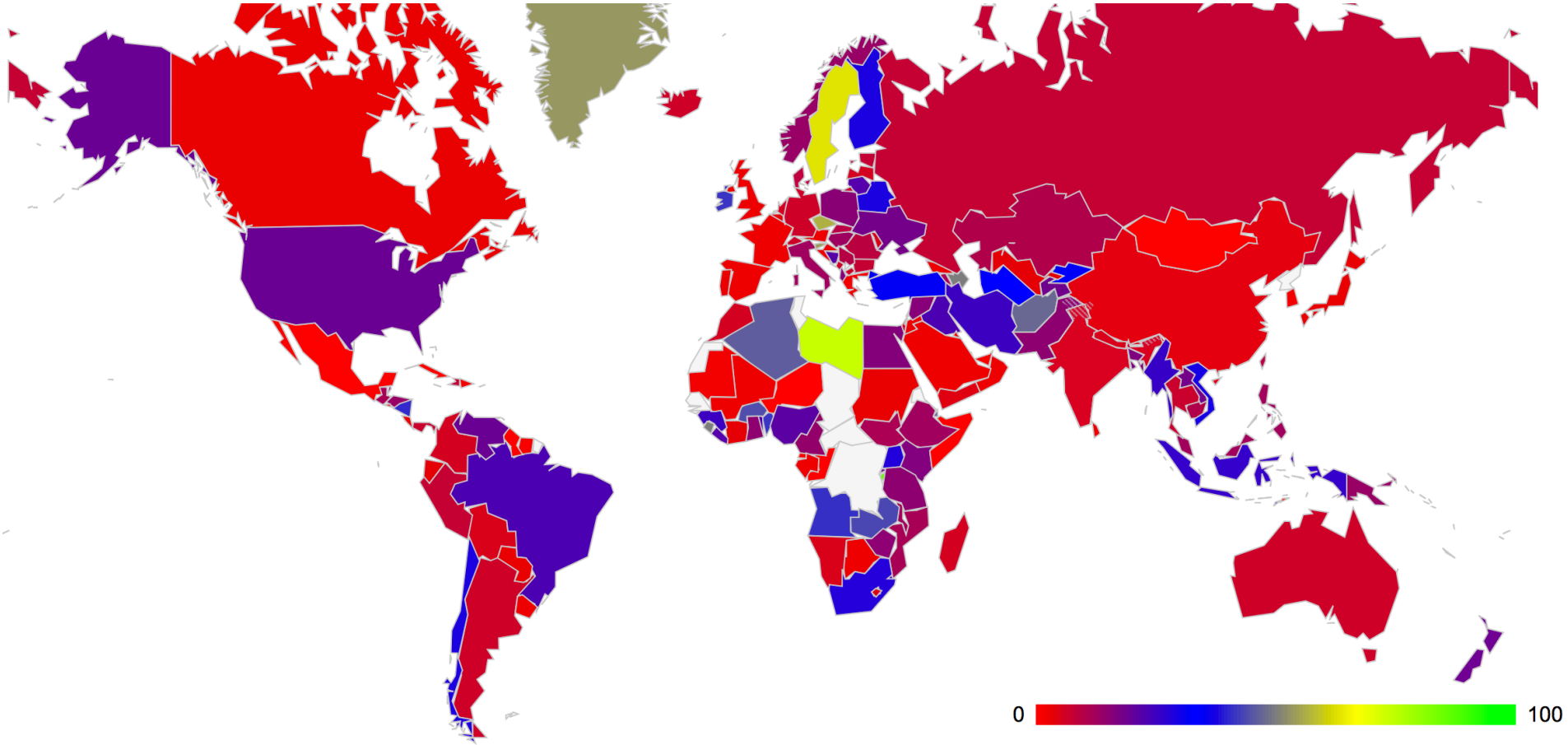
- Use the tools to validly sign a zone
- Lodge the DS records with the parent zone
- Use the keys to sign the zone
- Then, corrupt the RRSIG over labels in the zone
- Result: valid DNSSEC chain TO the zone, but the RRSIG return for the label is incorrectly signed.
- Can also lodge corrupted DS with parent and have an entire sub-domain invalidly signed
- RRSIG works with wildcards. Corrupted RRSIG ‘fails’ properly with wildcards (NSEC is still valid)

Results

- 1 weeks advert placement, $140/day bought ~ 250,000 placements per 24h billing period
- Total of 1,838,084 experiments run worldwide
- 67,766 resolvers seen
- 2,984 appear to be DNSSEC enabled 4.4%

- 1,329,084 clients seen
- 188,112 appear to be DNSSEC enabled 14.15%

Hows the world doing on DNSSEC?



NZ Results

- 7500 experiments ran against NZ end-users in the last week
  - 79 ASNs had resolvers active
  - 111 ASNs had clients tested
- In aggregate, 15% of clients appear to have DNSSEC enabled resolvers
  - Since the average number of nservers per host is 1.9, this means nearly everyone has a non-DNSSEC enabled alternate and so has far lower validation coverage.
  - But.. This is from a small sample, with a couple of standout ASN
- Almost 300 distinct resolvers were seen in the DNS and around 2.7% were DNSSEC enabled

NZ Results

ASN	NetName	Sample size	%clients
4771	NZTELECOM	3274	0.95%
4768	CLIX-NZ	1241	1.37%
7657	VODAFONE-NZ-NGN-AS	1041	93.37%
9790	CALLPLUS-NZ-AP	587	2.73%
17746	ORCONINTERNET-NZ-AP	334	2.69%
17412	WOOSHWIRELESSNZ	116	2.59%
17705	INSPIRENET-AS-AP	22	95.45%
23655	SNAP-NZ-AS	85	3.53%
4648	NZIX-2	84	2.38%
17435	WXC-AS-NZ	72	2.77%

NZ Results

ASN	NetName	Sample size	%clients
4771	NZTELECOM	3274	0.95%
4768	CLIX-NZ	1241	1.37%
7657	VODAFONE-NZ-NGN-AS	1041	93.37%
9790	CALLPLUS-NZ-AP	587	2.73%
17746	ORCONINTERNET-NZ-AP	334	2.69%
17412	WOOSHWIRELESSNZ	116	2.59%
17705	INSPIRENET-AS-AP	22	95.45%
23655	SNAP-NZ-AS	85	3.53%
4648	NZIX-2	84	2.38%
17435	WXC-AS-NZ	72	2.77%

Future work

- Improve the ‘is DNSSEC active’ tests
  - Unique intermediate domains, with DNSSEC DS per experiment
  - Less cached data, more instances of the {A,DNSKEY} fetch signature
- Mapping the relationships of clients, resolvers
  - We already have initial data on the use of google 8.8.8.8
- Oddities
  - Evidence resolver does DNSSEC, but fetches malsigned URL
  - Evidence of >10 nservers covering a client
    - Probably use of services with anycast/replicated infrastructure

DNSSEC observations

DNSSEC observations

- DNSSEC is complicated to configure on the server side

DNSSEC observations

- DNSSEC is complicated to configure on the server side
- Its also complicated to configure on the client side
  - DNSSEC enabled resolver, but not validating
  - Validating resolver, but client has more than one resolver
  - Clients bypass OS installed resolver logic, invent their own
  - (we measure on average 1.8 resolvers per client)
- Trust comes from the root
  - If you don't have the root out of band, you depend on in-band trust
  - Bad…
- Hop-Over DNS (8.8.8.8) is very good QoS, popular
  - But isn't DNSSEC enabled in itself (it will forward)
- Chained DNS behind forwarders may not be adding value