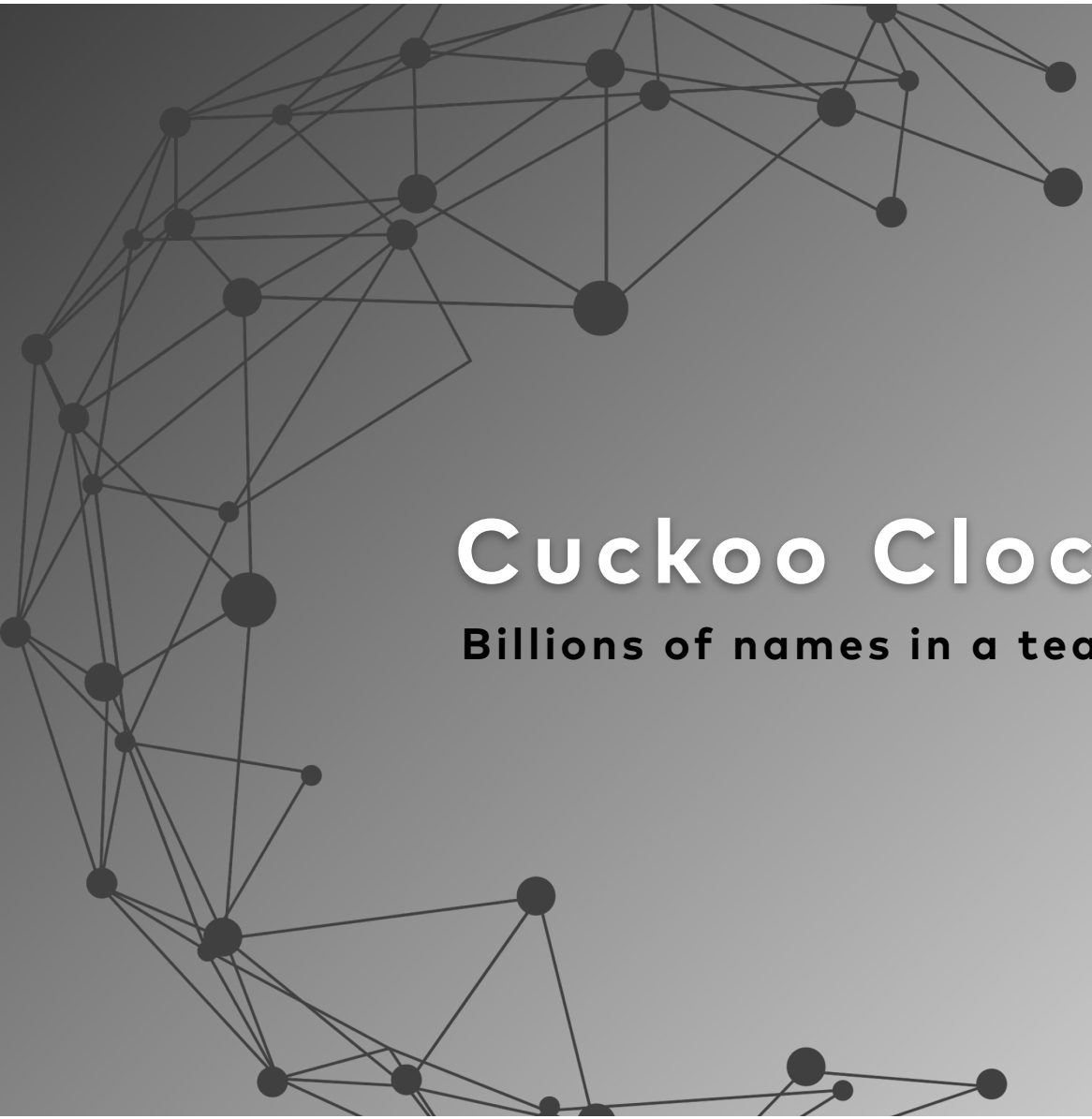


quad9

Cuckoo Clock

Billions of names in a teacup

John Todd

DNS-OARC 47 - November 2026

the problem

quad9

the problem

quad9

the DNS is like... big, man. like... really big.

what's even in it?

who cares?

quad9

threat intelligence providers, mostly

researchers

internal trending and behavioral analysis

do not stare into the abyss...

the problem

quad9

What domains have we seen?

What domains have we not seen?

Examples:
yoyodyne. com

example. com

the bigger problem

quad9

what FQDNs have we seen?

what FQDNs have we not seen?

examples:
whorfin. yoyodyne. com

foo. bar. example. com

the EVEN BIGGER problem

quad9

what FQDNs + QTYPE + RRs have we seen?

what FQDNs + QTYPE + RRs have we not seen?

examples:
whorfin. yoyodyne. com A (CNAME) lizardo. princeton. edu

foo. bar. example. com A (A) 192.168.55.4

the needs:

quad9

we need to keep data away from the core that already exists

need a data cache as "gatekeepers" to slow floods and deduplicate events

can we know with some reasonable accuracy: "have we seen this already or not?"

collect data useful to threat hunters

don't kill the network!

don't kill the back-end store!

other requirements:

quad9

TTL - no item should live forever (but a long time!)

counters are needed for future-proofing

LRU - when full, the least frequently seen items get booted first

need to selectively delete items arbitrarily

need predictable maximium memory size - no surprises!

each machine is independent from all others (islanding)

60 day uniqueness window

no need to store "safe" zone data

options for memory stores

quad9

exact store methods:
- actual object data
- packed arrays
- compressed objects
- tries/radix trees
- sorted hashes
- minimal perfect hashes

problems (various mixed bag):
large memory space
cache misses on each lookup
static (no rebuilds)
decompression costs
lookups are slow

options for tracking (cont'd)

quad9

probabilistic filters:
- Bloom
- counting Bloom
- XOR/binary fuse filters
- ribbon filter
- quotient filter

Problems (various mixed bag):
static (no rebuilds)
lookups are slow
random deletions
not as memory efficient
slows significantly when loaded

quad9

just... pick... something.

cuckoo filter

quad9

2014 paper by Bin Fan, David G. Andersen, Michael Kaminsky, Michael D. Mitzenmacher

like Bloom but can support targeted deletions

better performance when nearing capacity (which will be "always")

supports extensions

uses less space (in theory)

cuckoo filter

quad9

answers question with 100% certainty that "this thing is not in the set"

however, an answer of "this thing is in the set" may sometimes be false,
and the frequently of false positive answers is a function of the
selectable bit size of the fingerprint of the object (among other things)

examples with common settings:

8 bit fingerprint: 1 in 4.5 false positive
16 bit fingerprint: 1 in 1.024 false positive
24 bit fingerprint: 1 in 262.000 false positive
32 bit fingerprint: 1 in 527.000.000 false positive

this looks great... so how?

quad9

looks like the start of a solution, and we see it should go in vector but...

quad9 is not a "code shop" - we are a service provider

we need to add a bunch of features on cuckoo, and then add to vector and pdns

.
.
.

nominet DNS fund

quad9



nominet DNS fund

quad9

nominet



+ Ensar Sarajčić

cuckoo clock

quad9

rust library - generically usable in all rust implementations

first implementation in vector (vector.dev)

second implementation in powerdns dnsdist (dnsdist. org) (pending PR)

third-ish implementation in go-dnscollector (not our library, but liked the idea so much it got implemented)

cuckoo clock

quad9

like cuckoo, but we added more features:

- TTL - how long until we reap this name, freeing space?

- counters - how many times have we matched this name?

- LRU - when we are full and need to make space, delete least-frequently-used items first

downside:

- uses memory in powers of 2

- configuration is too flexible - all knobs change other knobs

cuckoo filter - efficient!

quad9

32 bits per object (1:70,000,000 false result)

~127 million objects at 95% capacity

30 day absolute TTL (14 bits)

3.5 days of LRU (10 bits)

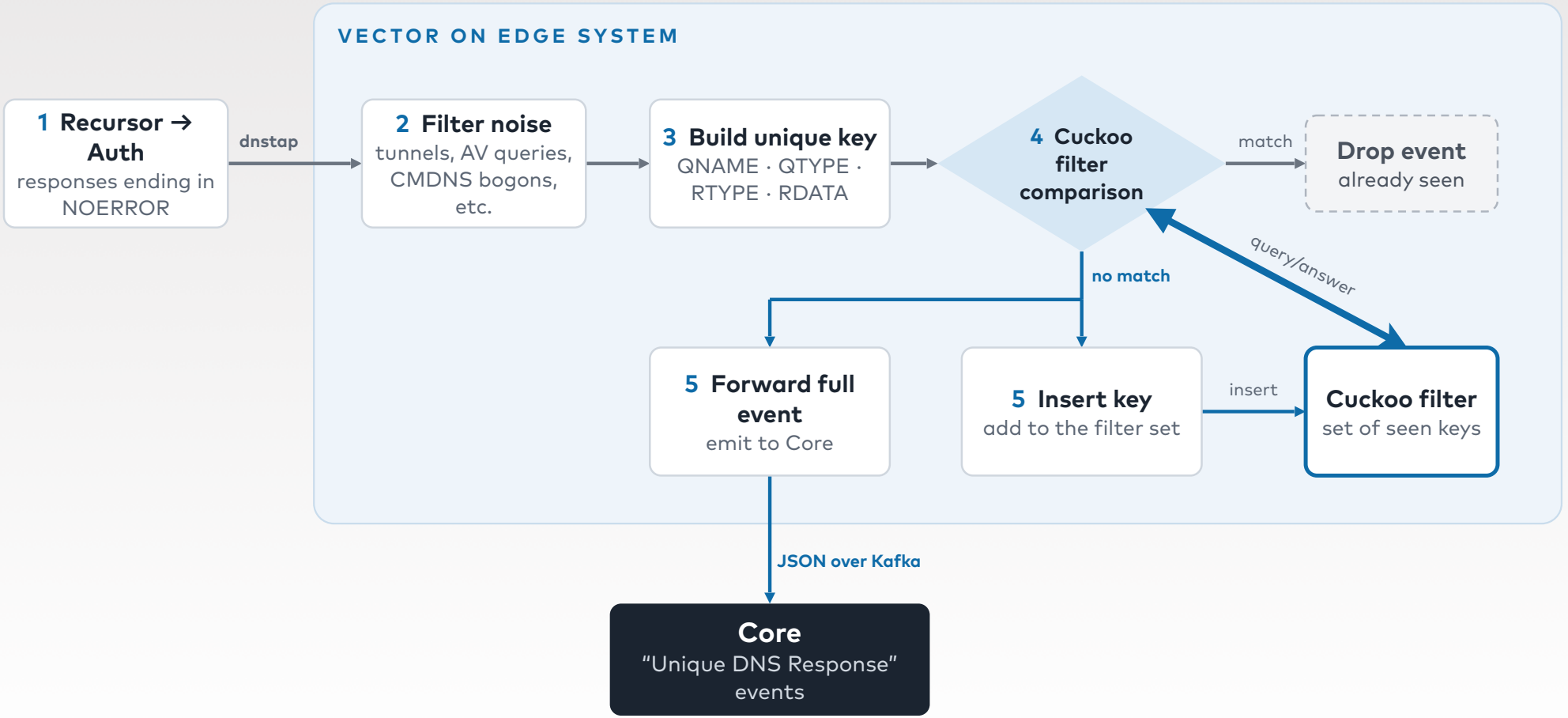
memory: 1,073,741,824 bytes (1G)

= ~67 bits per object

(could drop the LRU and get 10 bits back in exchange for random deletions)

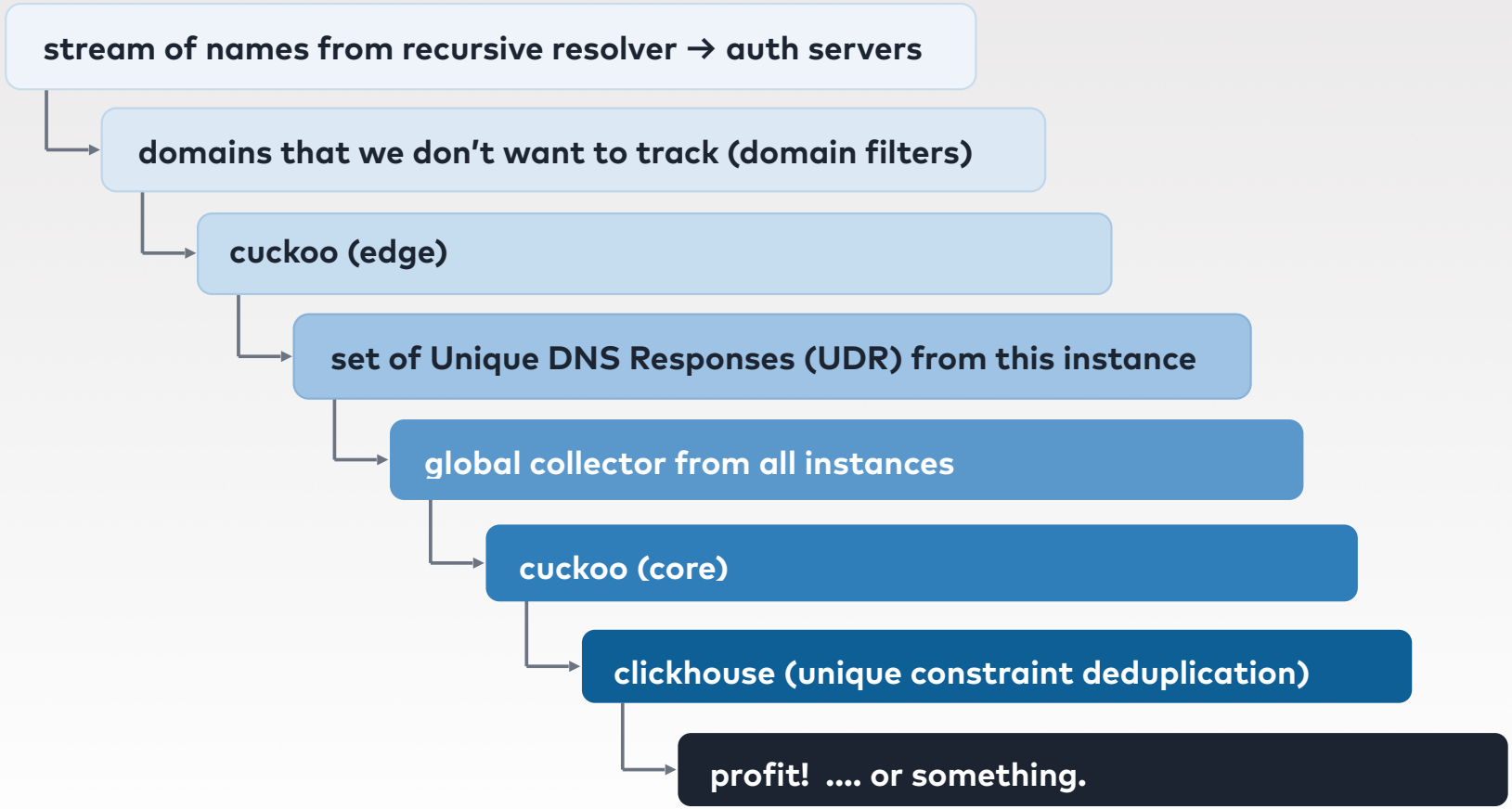
pipeline

quad9



what this looks like

quad9



configs - edge

quad9

127.000.000 keys @95% capacity
32 bit fingerprint size (1 in 67 million false positive)
30 day TTL (14 bit)
3.5 day LRU countdown (10 bits)
32 buckets (often 4, but there are tradeoffs)
6 hour "tick" of the clock

memory size with above configs:
1 GB

when cache is full, it "runs hot" without falling over - TTL clears things every once in a while, and LRU allows

configs - edge (vector)

quad9

```
udr_cuckoo_table:
  type: memory
  ttl: 2592000
  flush_interval: 30
  scan_interval: 300
  reload_behavior: preserve-state
  inputs: ["udr_cuckoo_population_parser"]
  filter:
    type: cuckoo
    persistence_path: /var/lib/vector/cache/udr_cuckoo.state
    export_interval: 21600
    max_entries: 127000000
    bucket_size: 32
    ttl_enabled: true
    ttl_bits: 14
    lru_enabled: true
    lru_bits: 10
    lru_starting_value: 1022
    lru_increment: 1
    lru_aging_strategy:
      strategy: decrement
      value: 1
    lru_deletion_enabled: false
    fingerprint_bits: 32
```

configs - core

79.500.000.000 keys @100% capacity
32 bit fingerprint size (1 in 29 million false positive)
60 day TTL (8 bit)
74 buckets
no LRU (random overwites)
12 hour "tick" of the clock

memory size with above configs:
480 GB (fits into a 512GB machine)

persistence here is a big challenge - loading/saving to disk lags

configs - core

```
udr_cuckoo_table:
  type: memory
  ttl: 5184000                # 60 days
  flush_interval: 30
  scan_interval: 43200        # 12-hour clock tick
  reload_behavior: preserve-state
  inputs: ["udr_cuckoo_population_parser"]
  filter:
    type: cuckoo
    persistence_path: /var/lib/vector/cache/udr_cuckoo_80b.state
    export_interval: 21600
    max_entries: 79456894976  # 2^30 buckets x 74 = 100% of slots
    bucket_size: 74
    ttl_enabled: true
    ttl_bits: 8
    lru_enabled: false
    fingerprint_bits: 32
```

the basic numbers

quad9

~3% of queries are unique from that instance's perspective (30 day window)

1% of queries are unique globally after application of "noise" domain filters

16 cores @ 2.5ghz can keep up with >100,000 events (haven't tried more)

daily summary of around 3.5 billion new UDR items after filtering noise

sharding

quad9

79 billion keys still only gets a week or so per 512G for us

we're just "eating" some of the duplication on clickhouse

large-scale answer: sharding based on inverted qname or hash of UDR

quad9

outputs



quad9

outputs

newly observed domains (NOD)

Domain Change Monitor (NOD) stringmatching (stringsimile)

NOD + "first 7 days of changes"

stringsimile pattern matching (github.com/Quad9DNS/stringsimile)

MX research

tunnel domain identification

novel fast flux targets

etc.

code details

quad9

Rust library:
https://crates.io/crates/cuckoo-clock

Vector implementation (complete):
https://github.com/vectordotdev/vector/pull/25143

dnsdist implementation (awaiting review)
https://github.com/PowerDNS/pdns/pull/17889

quad9

questions?

jtodd@quad9.net