Privacy-friendly ECS

Implementing EDNS Client Subnet in a privacy preserving way

Andrey Meshkov
CTO and Co-Founder of AdGuard
am@adguard.com
@ay_meshkov

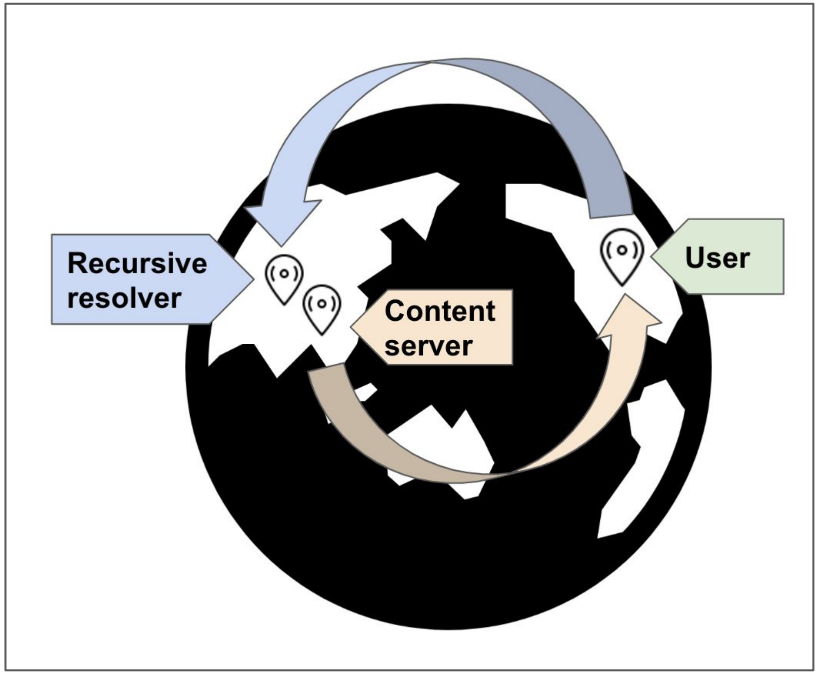
Introduction

- Recursive resolver
- Avg. 1.5-2M RPS
- Over 70M users all over the world
- 95% of traffic is encrypted
- 16 points of presence

ADGUARD DNS

Challenges With Traditional DNS

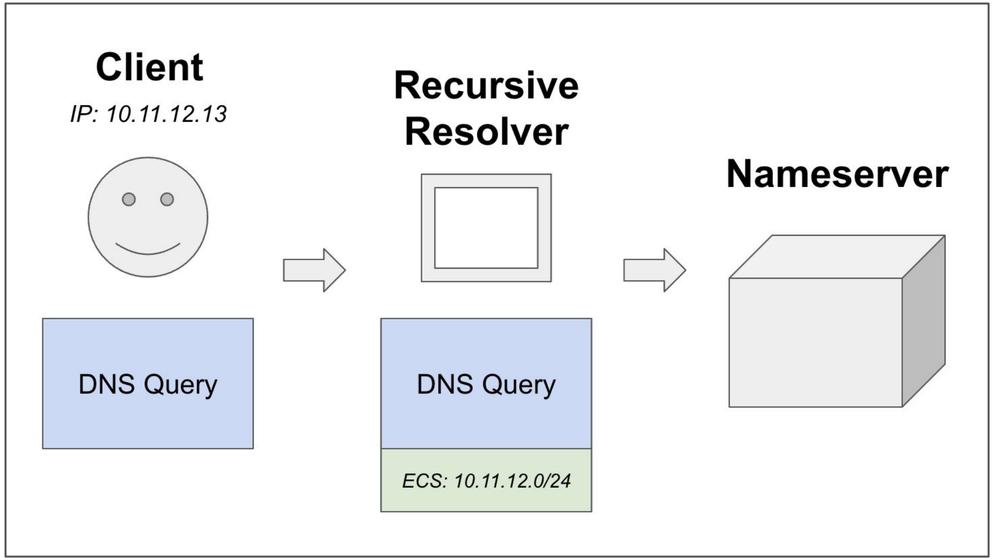
- Traditional DNS resolution is typically location-agnostic
- Growth in popularity of public resolvers thwarted GeoDNS



Simplified diagram showing how ECS works

What Is EDNS Client Subnet

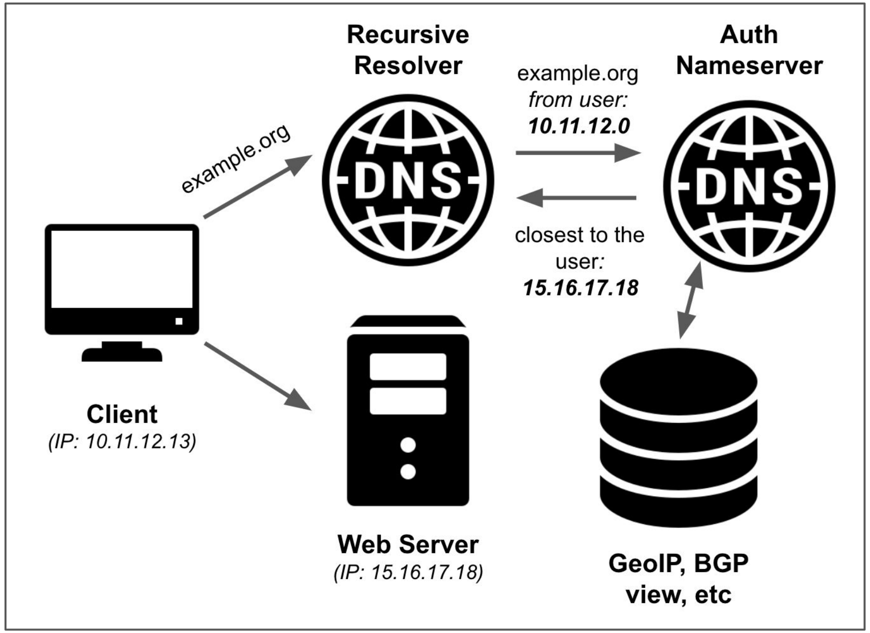
The client's subnet information (usually a truncated part of the client's IP address) is included in the DNS query to authoritative nameservers.



Simplified diagram showing how ECS works

Use Cases For ECS

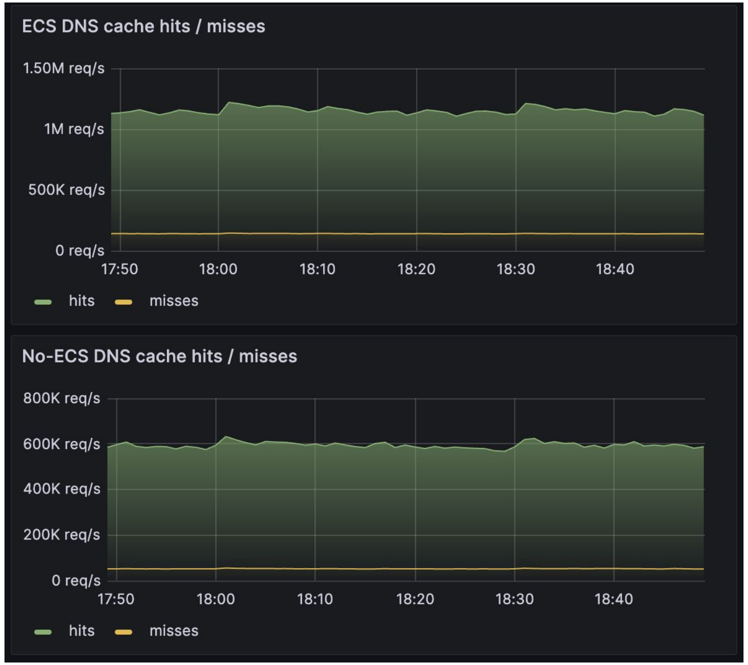
- Content Delivery Networks (CDNs)
- Online advertising
- Regional restrictions and compliance



The most popular case - using ECS for steering

ECS Is Actually Popular

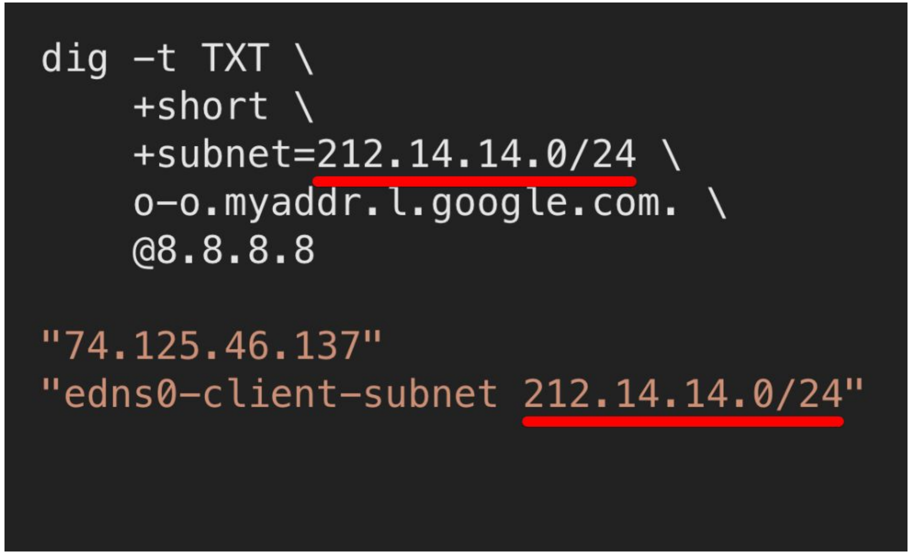
- About **67%** of all queries we receive are for domains that **support ECS**



Comparing queries to ECS-enabled domains with other queries.

Privacy Issues

- ECS leaks users' location information
- Ethical concerns
- Legal concerns
- RFC even recommends keeping it off by default

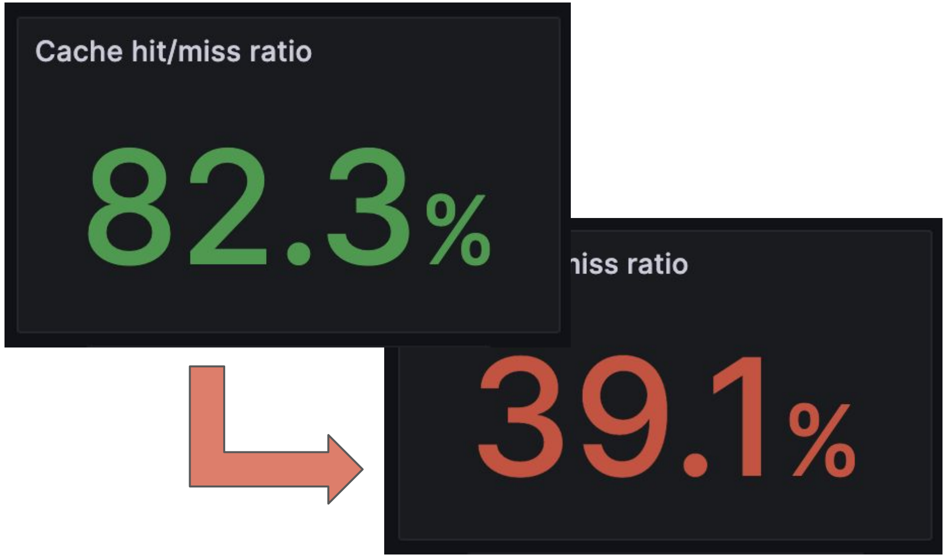


```
dig -t TXT \
    +short \
    +subnet=212.14.14.0/24 \
    o-o.myaddr.l.google.com. \
    @8.8.8.8

"74.125.46.137"
"edns0-client-subnet 212.14.14.0/24"
```

Just a demonstration what part of your IP will be received by nameservers.

Negative Impact On Caching

- Reduces effectiveness of caching
- May lead to cache pollution
- Often used incorrectly by nameservers

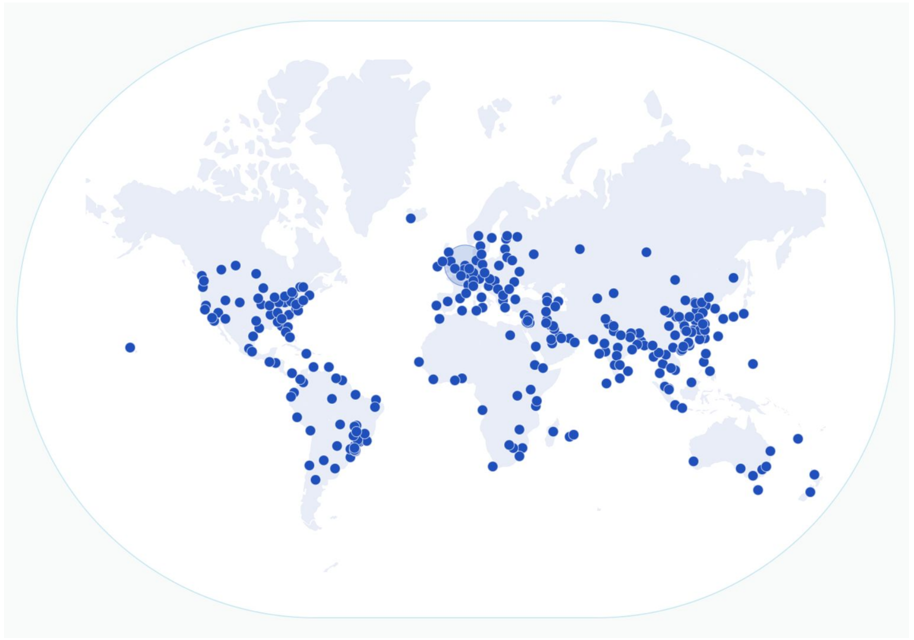


If you're still not bothered by privacy issues, maybe these numbers will convince you.

Do We Actually Need ECS?

Cloudflare argues that their cache is sufficiently localized as they have hundreds of PoPs.

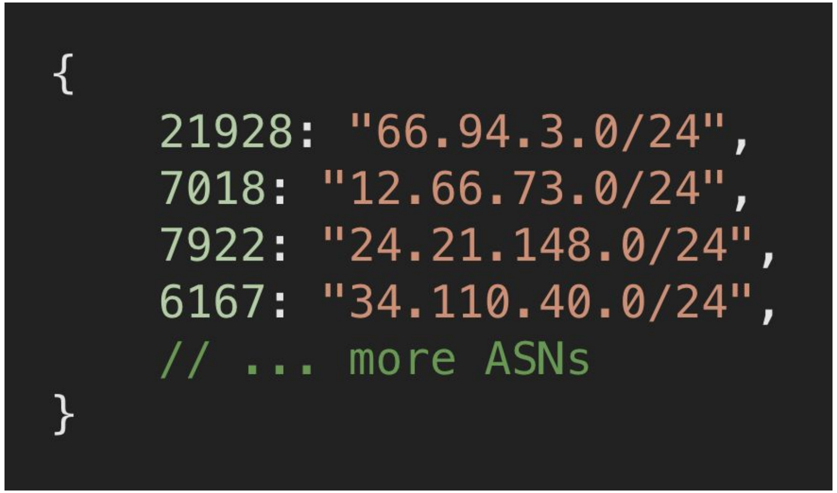
- But what if you have fewer PoPs?
- Large content providers have servers inside ISP networks.



Source: https://www.cloudflare.com/network/

Replace Subnets With AS Numbers

- Build a map: ASN to IP subnet
- Use one random IP subnet per ASN as a ECS
- First introduced by NextDNS in 2019 [1]

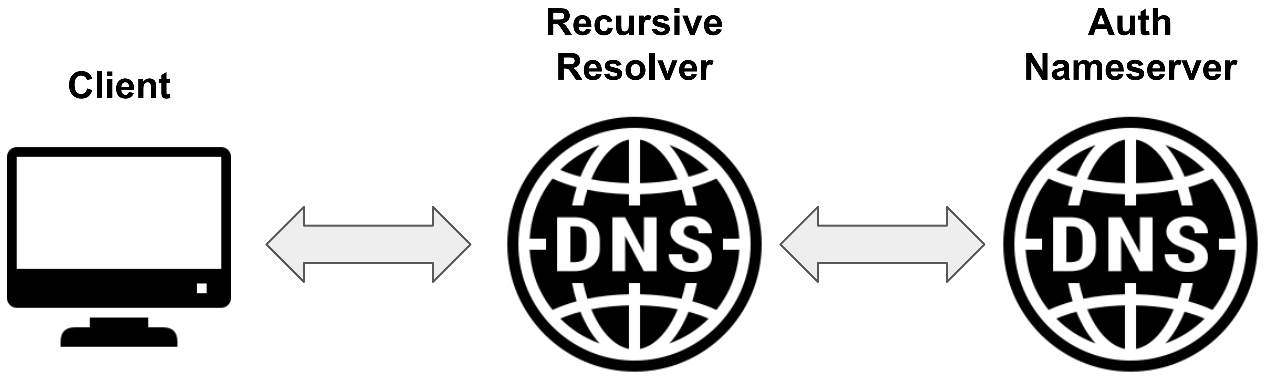


```
{
    21928: "66.94.3.0/24",
    7018: "12.66.73.0/24",
    7922: "24.21.148.0/24",
    6167: "34.110.40.0/24",
    // ... more ASNs
}
```

A map where key is AS number and value is a random /24 subnet that belongs to that AS.

[1]: https://medium.com/nextdns/how-we-made-dns-both-fast-and-private-with-ecs-4970d70401e5

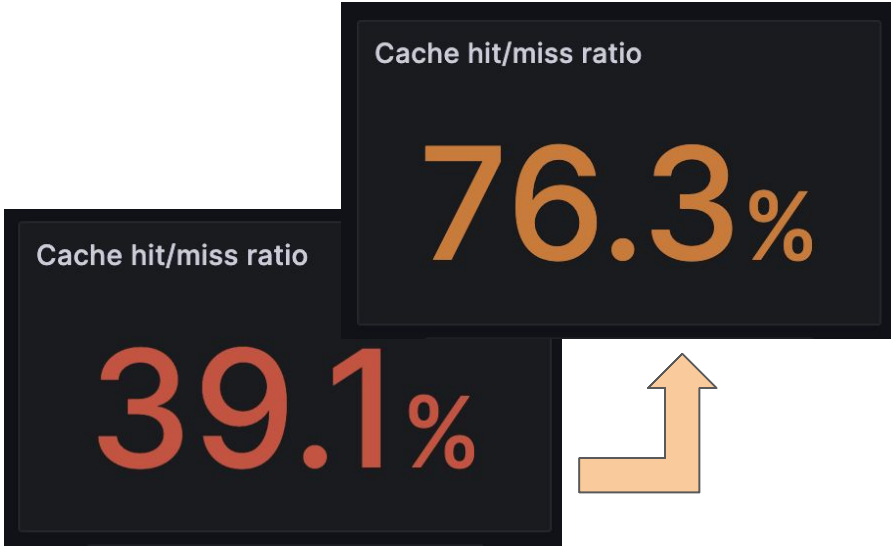
Replace Subnets With AS Numbers



```
asn := lookupASN(ip)
subnet := subnetByASN(asn)
addECS(msg, subnet)
```

Cache Efficiency

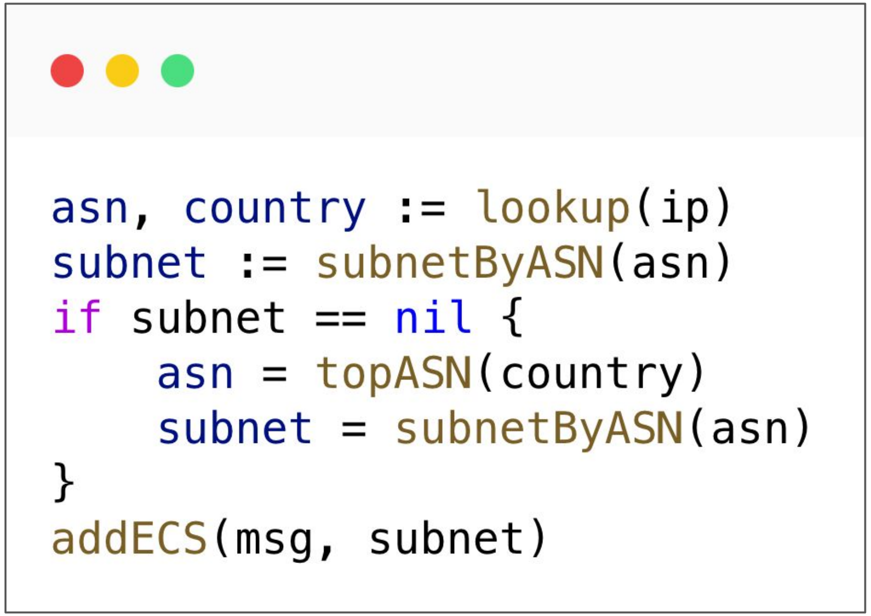
- Hit/miss ratio went up to 75-80% depending on the location
- ECS cache is 5 times bigger than the regular cache
- Hit/miss ratio is ~15% worse than the regular cache



Note, that the numbers depend on the server location and load.

Improving Cache Efficiency

- Choose the top ASNs per country
- If the query is from a different ASN, use a subnet from the most popular ASN in that country instead

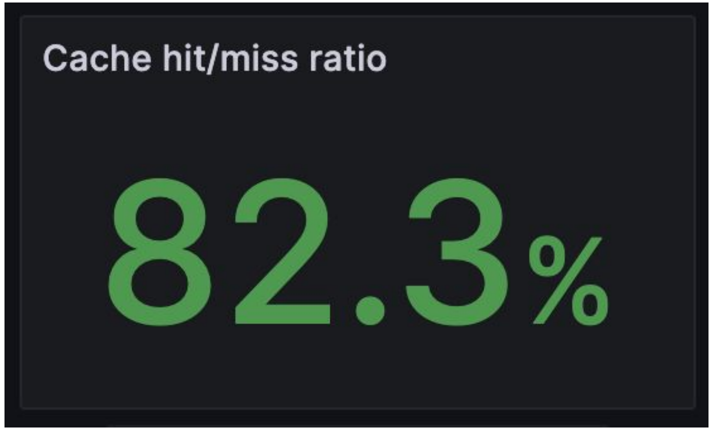


```
asn, country := lookup(ip)
subnet := subnetByASN(asn)
if subnet == nil {
    asn = topASN(country)
    subnet = subnetByASN(asn)
}
addECS(msg, subnet)
```

Using only top ASNs for each country

Improving Cache Efficiency

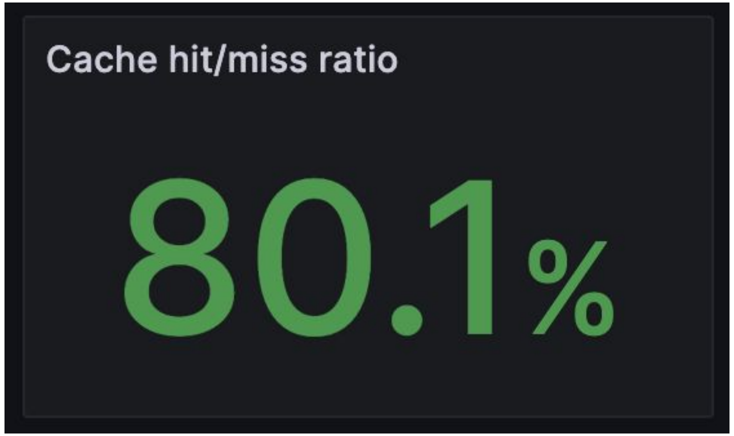
- Up to 10 ASN per country
- Discard ASNs from which we receive fewer than 3% queries
- **Result:** ~5% worse than the regular cache



Cache hit/miss ratio

82.3%

- Up to 50 ASN per country
- Discard ASNs from which we receive fewer than 0.1% queries
- **Result:** ~8% worse than the regular cache

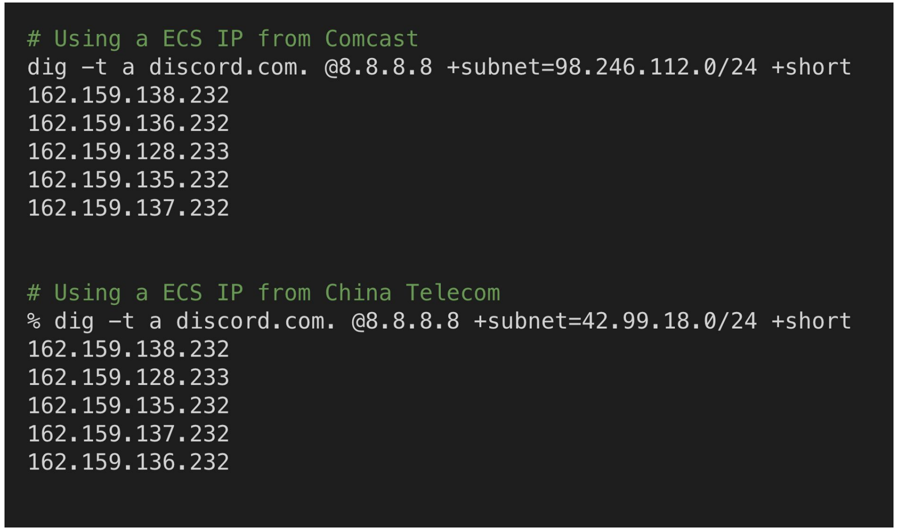


Cache hit/miss ratio

80.1%

Improving Cache Efficiency

- Some nameservers indicate ECS support, but in fact they return the same IP every time [1] and pollute cache.
- Analyzing ~1000 popular domains [2]:
  - ~50% of all indicate ECS support
  - ~15% of all despite that return the same IP for different ECS supplied

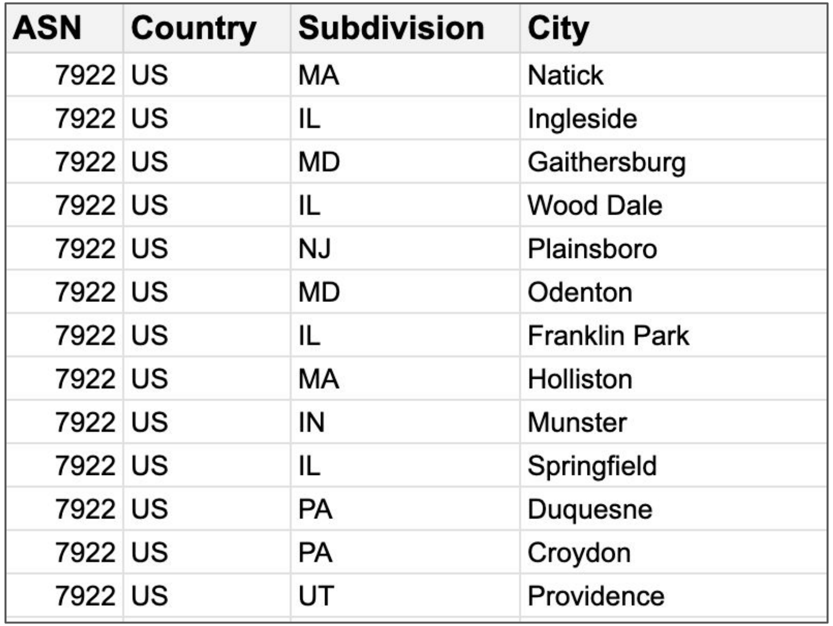


```
# Using a ECS IP from Comcast
dig -t a discord.com. @8.8.8.8 +subnet=98.246.112.0/24 +short
162.159.138.232
162.159.136.232
162.159.128.233
162.159.135.232
162.159.137.232


# Using a ECS IP from China Telecom
% dig -t a discord.com. @8.8.8.8 +subnet=42.99.18.0/24 +short
162.159.138.232
162.159.128.233
162.159.135.232
162.159.137.232
162.159.136.232
```

[1]: https://medium.com/nextdns/how-we-made-dns-both-fast-and-private-with-ecs-4970d70401e5
[2]: https://github.com/ameshkov/ecssupportchecker

Not Ideal Solution Yet

Large ISPs can announce prefixes from lots of different places and using just the ASN is not enough to achieve the necessary quality.



ASN	Country	Subdivision	City
7922	US	MA	Natick
7922	US	IL	Ingleside
7922	US	MD	Gaithersburg
7922	US	IL	Wood Dale
7922	US	NJ	Plainsboro
7922	US	MD	Odenton
7922	US	IL	Franklin Park
7922	US	MA	Holliston
7922	US	IN	Munster
7922	US	IL	Springfield
7922	US	PA	Duquesne
7922	US	PA	Croydon
7922	US	UT	Providence

According to MaxMind GeoIP2 ISP database, Comcast prefixes are attributed to ~1600 different cities / 45 subdivisions

Large ISP Example

```
% dig -t a \
    www.google.com \
    @8.8.8.8 \
    +subnet=98.246.112.0/24 \
    +short

142.251.215.228
```

ECS from AS7922, response IP location is US West Coast.

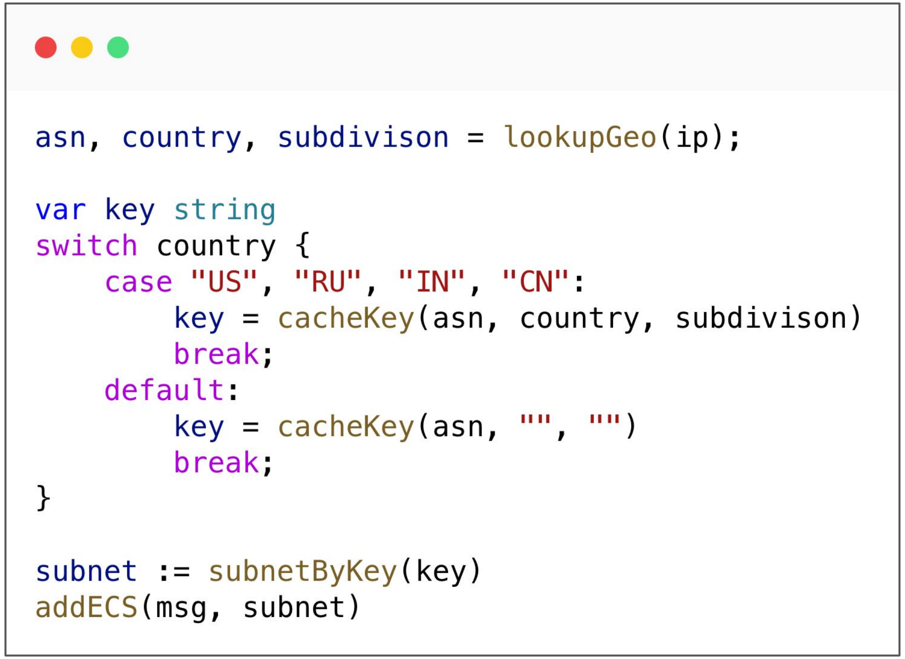
```
% dig -t a \
    www.google.com \
    @8.8.8.8 \
    +subnet=23.24.0.0/24 \
    +short

142.251.40.132
```

ECS from AS7922, response IP location is US East Coast.

Dealing With Large ASNs

What we tried:

- Add country and subdivision to the subnet selection algorithm
- Limit it to large countries
- Resulting responses are more precise
- Cache efficiency stays the same



```
asn, country, subdivison = lookupGeo(ip);

var key string
switch country {
    case "US", "RU", "IN", "CN":
        key = cacheKey(asn, country, subdivison)
        break;
    default:
        key = cacheKey(asn, "", "")
        break;
}

subnet := subnetByKey(key)
addECS(msg, subnet)
```

Using country and subdivision in addition to ASN

Thank you!

Questions?

Andrey Meshkov
am@adguard.com
@ay_meshkov