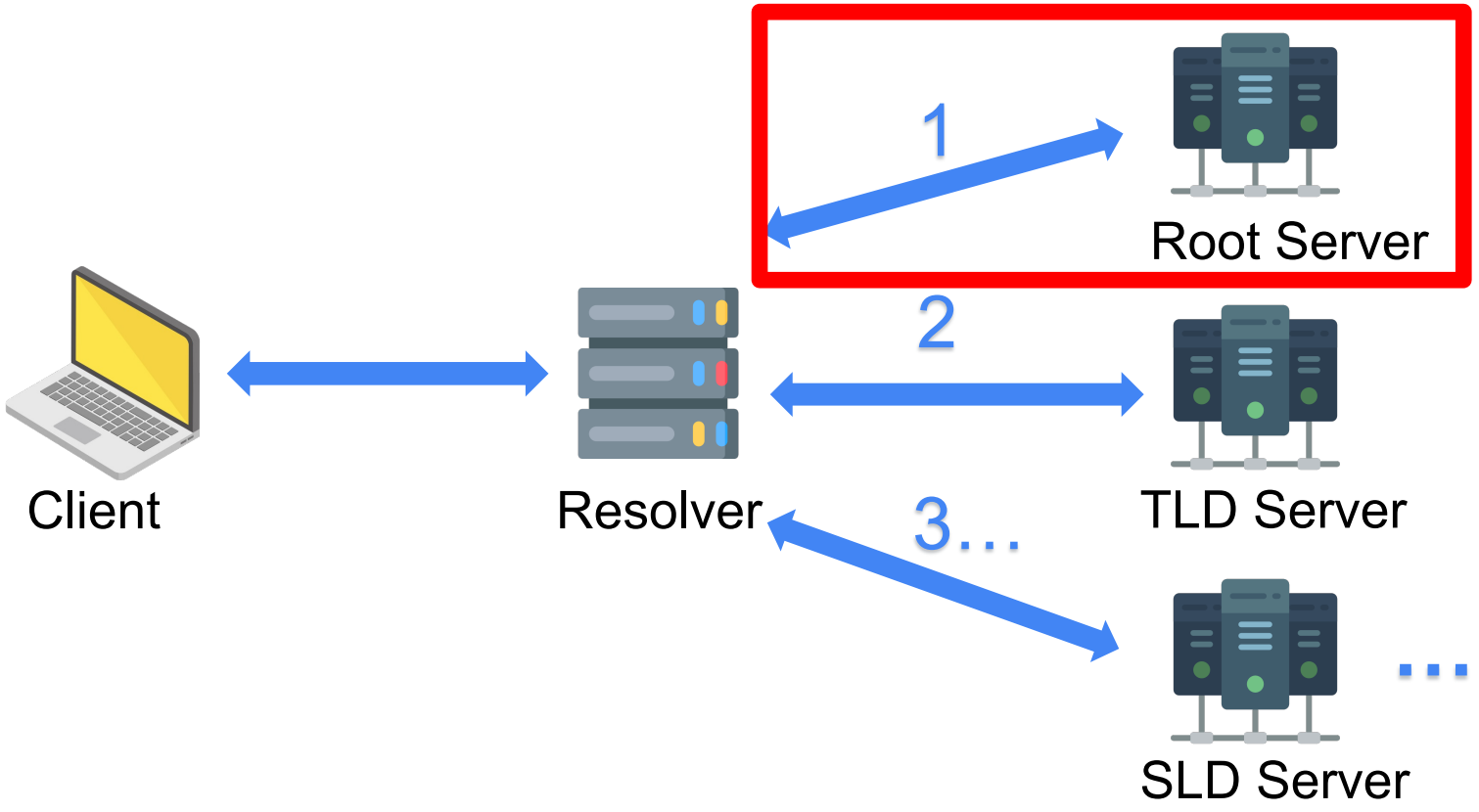
# Measuring the Practical Effect of DNS Root Server Instances:
## A China-Wide Case Study

Fenglu Zhang, Chaoyi Lu, Baojun Liu,
Haixin Duan and Ying Liu


# DNS resolution process



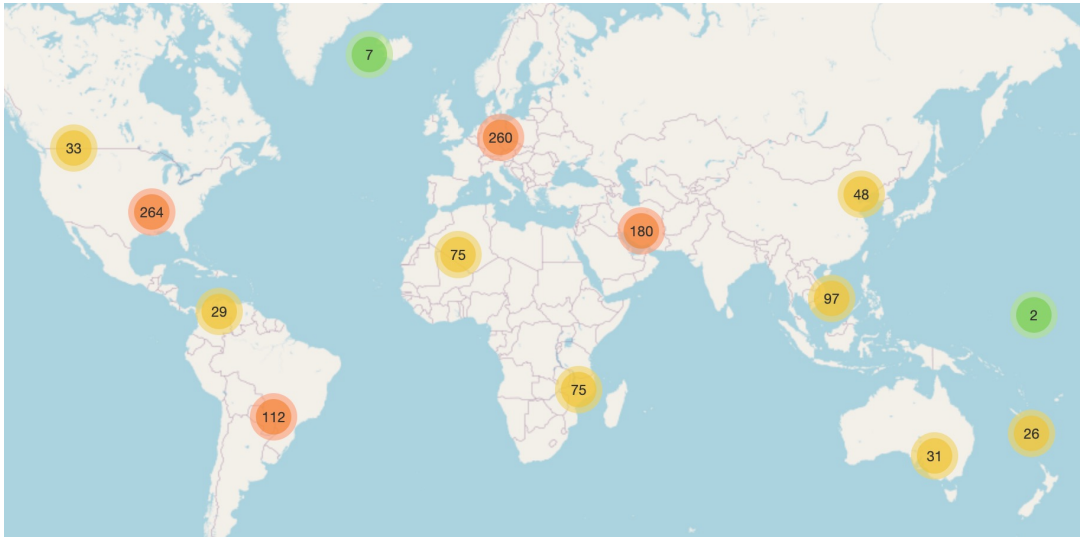
# Anycast and root Instance

- To improve stability, root servers are deployed using **anycast**.

13 root servers

```
a.root-servers.net.
b.root-servers.net.
…
m.root-servers.net.
```

1518 root instances



# Motivation

- Root instances deployed rapidly
  - Dec 2019: 1033 

Dec 2021: 1478
- How about their **practical effects?**
- A China-wide case study.

Motivation

# Methodology

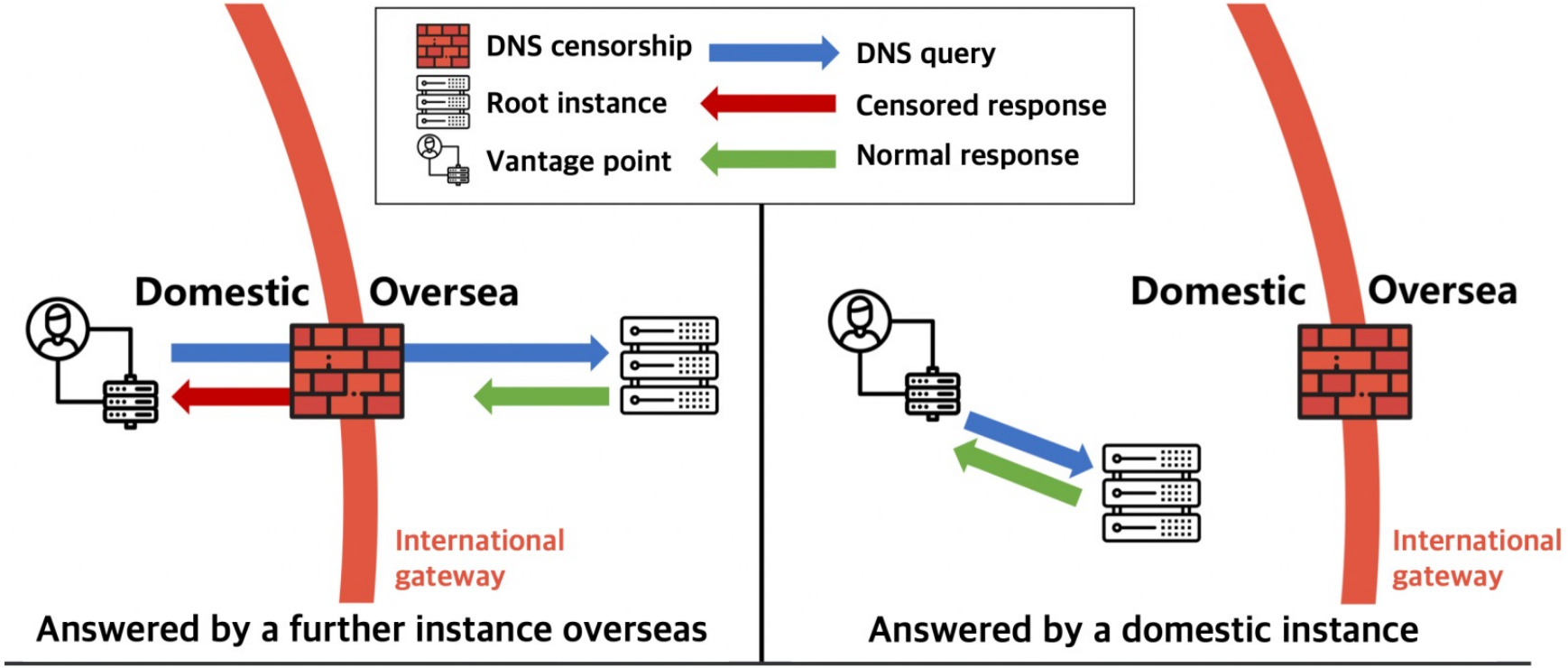
Measurement and Result

Recommendation

# Challenge and solution

- **Challenge**
  - Under anycast, how to find the **exact instance** that responds?
- **Our solution**
  - Answer another question:
    - Are queries resolved domestically or overseas?
  - Use **DNS censorship** to determine it

# Determine if domestic instances serve queries



Motivation

Methodology

# Measurement and Result

Recommendation

**Q1: Which networks are served by domestic root instances?**

**Q2: Why are some networks not served by domestic instances?**

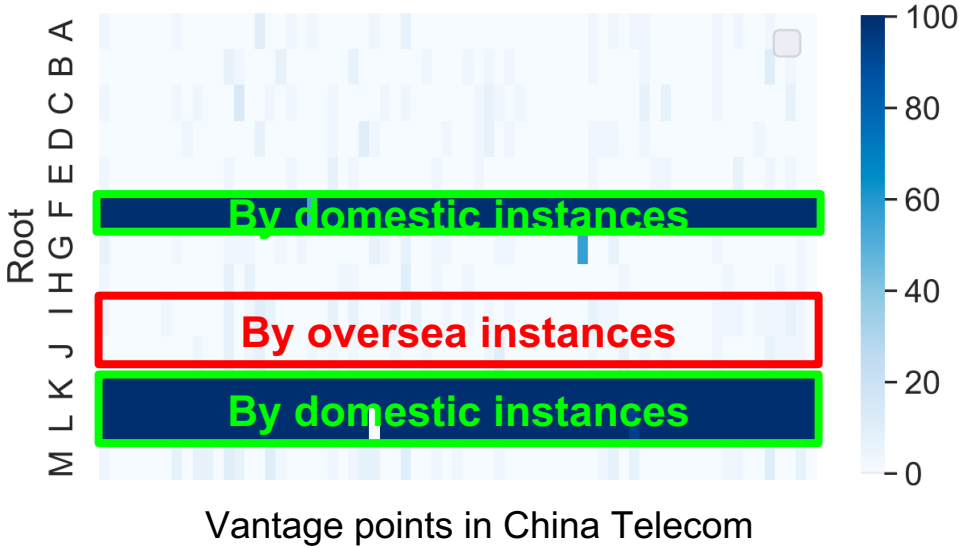
**Q3: How do domestic instances affect root server selection?**

**Q1: Which networks are served by domestic root instances?**

**Q2: Why are some networks not served by domestic instances?**

**Q3: How do domestic instances affect root server selection?**

# Example: China Telecom



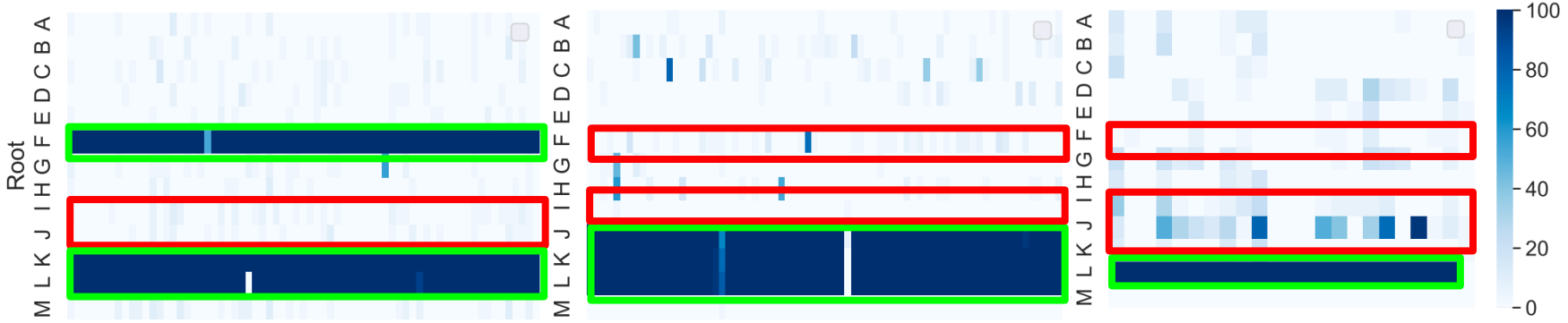

Vantage points in China Telecom

- Except for F, K, and L, almost all queries to other roots are responded by oversea instances.
- Domestic instances in Chinese mainland

| Root | F | I | J | K | L |
|---|---|---|---|---|---|
| # Instance | 4 | 1 | 2 | 3 | 6 |

**Darker** cells -> **more** queries from the VPs are resolved **by domestic instances**

# Summary




(a) Telecom (b) Unicom (c) Mobile

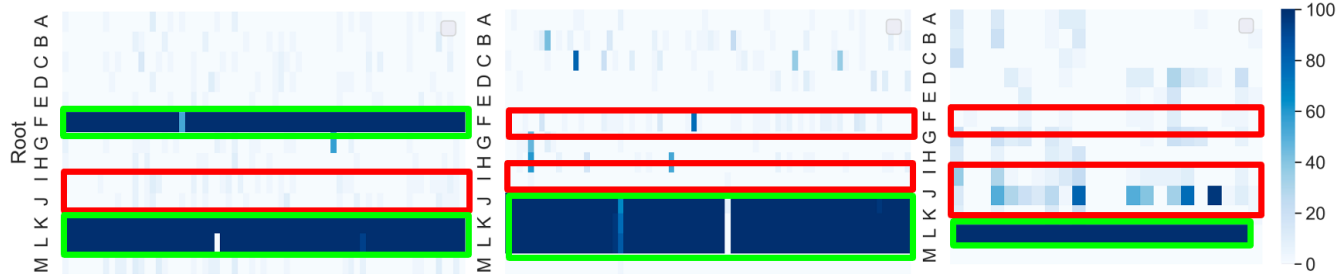
- Domestic instances help queries in domestic
- **Some networks not served by domestic instances**

Q1: Which networks are served by domestic root instances?

**Q2: Why are some networks not served by domestic instances?**

Q3: How do domestic instances affect root server selection?

# Unshared domestic instances




(a) Telecom (b) Unicom (c) Mobile

Reason:

- **Location** of root instance
- **Peering** between major ISPs
- Limitation of **BGP routing policies**

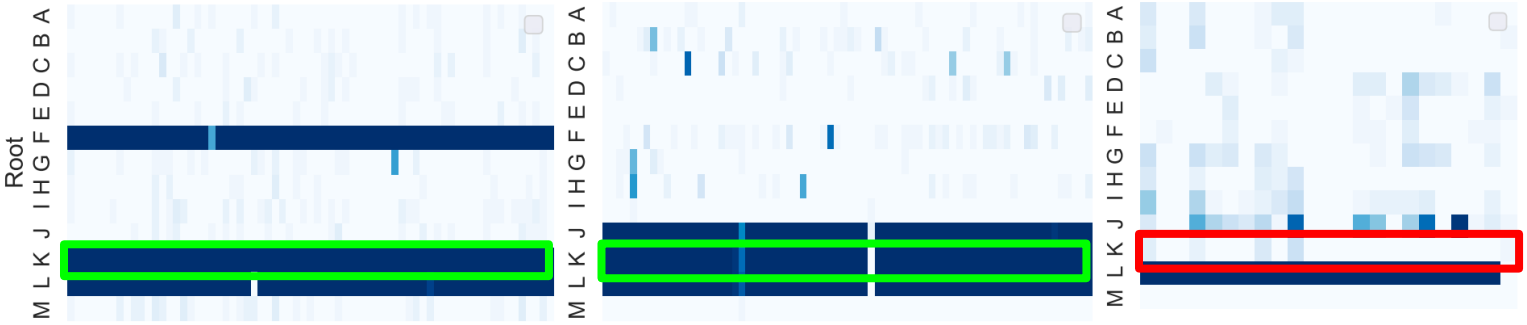
Q1: Which networks are served by domestic root instances?

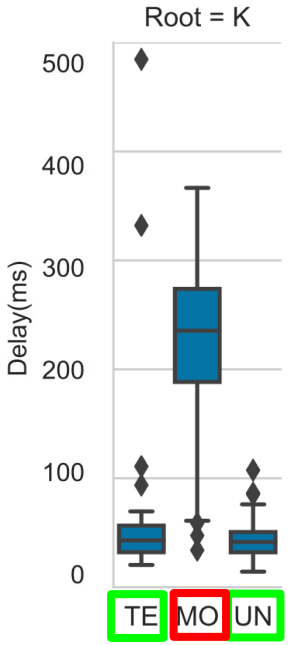
Q2: Why are some networks not served by domestic instances?

**Q3: How do domestic instances affect root server selection?**

# F1: Instances contribute lower delay

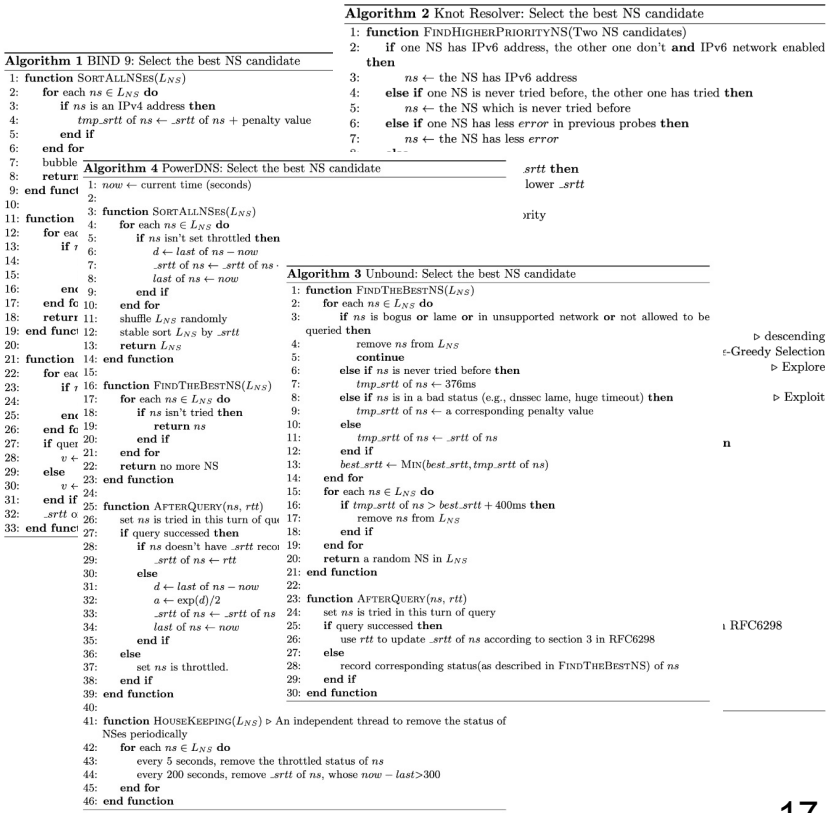
Example: K root



# F2: Software prefers lower delay

- Targets:
  - BIND9, Unbound, Knot Resolver, PowerDNS Recursor
- Methods:
  - **Source code reviewing**
  - **Dynamic debugging**



**Algorithm 1** BIND 9: Select the best NS candidate

```
function SortAllNSes(L_NS)
    for each ns ∈ L_NS do
        if ns is an IPv4 address then
            tmp_srtt of ns ← _srtt of ns + penalty value
        end if
    end for
    bubble
    return
end funct
function
    for ea
        if
            enc
    end fo
    return
end func
function
    for ea
        if
            enc
    end fo
    if quer
        v ←
    else
        v ←
    end if
    _srtt o
end func
```

**Algorithm 2** Knot Resolver: Select the best NS candidate

```
function FindHigherPriorityNS(Two NS candidates)
    if one NS has IPv6 address, the other one don't and IPv6 network enabled then
        ns ← the NS has IPv6 address
    else if one NS is never tried before, the other one has tried then
        ns ← the NS which is never tried before
    else if one NS has less error in previous probes then
        ns ← the NS has less error
```

**Algorithm 4** PowerDNS: Select the best NS candidate

```
now ← current time (seconds)

function SortAllNSes(L_NS)
    for each ns ∈ L_NS do
        if ns isn't set throttled then
            d ← last of ns − now
            _srtt of ns ← _srtt of ns ·
            last of ns ← now
        end if
    end for
    shuffle L_NS randomly
    stable sort L_NS by _srtt
    return L_NS
end function

function FindTheBestNS(L_NS)
    for each ns ∈ L_NS do
        if ns isn't tried then
            return ns
        end if
    end for
    return no more NS
end function

function AfterQuery(ns, rtt)
    set ns is tried in this turn of qu
    if query successed then
        if ns doesn't have _srtt reco
            _srtt of ns ← rtt
        else
            d ← last of ns − now
            a ← exp(d)/2
            _srtt of ns ← _srtt of ns
            last of ns ← now
        end if
    else
        set ns is throttled.
    end if
end function

function HouseKeeping(L_NS) ▷ An independent thread to remove the status of NSes periodically
    for each ns ∈ L_NS do
        every 5 seconds, remove the throttled status of ns
        every 200 seconds, remove _srtt of ns, whose now − last>300
    end for
end function
```

**Algorithm 3** Unbound: Select the best NS candidate

```
function FindTheBestNS(L_NS)
    for each ns ∈ L_NS do
        if ns is bogus or lame or in unsupported network or not allowed to be queried then
            remove ns from L_NS
            continue
        else if ns is never tried before then
            tmp_srtt of ns ← 376ms
        else if ns is in a bad status (e.g., dnssec lame, huge timeout) then
            tmp_srtt of ns ← a corresponding penalty value
        else
            tmp_srtt of ns ← _srtt of ns
        end if
        best_srtt ← Min(best_srtt, tmp_srtt of ns)
    end for
    for each ns ∈ L_NS do
        if tmp_srtt of ns > best_srtt + 400ms then
            remove ns from L_NS
        end if
    end for
    return a random NS in L_NS
end function

function AfterQuery(ns, rtt)
    set ns is tried in this turn of query
    if query successed then
        use rtt to update _srtt of ns according to section 3 in RFC6298
    else
        record corresponding status(as described in FindTheBestNS) of ns
    end if
end function
```

# F2: Software prefers lower delay

- **BIND9** and **Knot** prefer the **smallest RTT**
- **Unbound** and **PowerDNS** tend to select root servers **randomly**, despite that they are designed to consider RTT
- Considering market share[1, 2], domestic instances **effectively absorb queries** within their catchment

Reference:
[1] Going Wild: Largescale Classification of Open DNS Resolvers, IMC 15
[2] BIND DNS Holds Lead, https://www.serverwatch.com/server-news/bind-dns-holds-lead/

Motivation

Methodology

Measurement and Result

# Recommendation

# Recommendation

- **BGP peering with** root server networks or **alternative methods** that improve access to the root server system
- More **transparent** peering information between ISPs and root servers
- **Reviewing** if the DNS implementation is consistent with **original goals**

# Conclusion

A study for the practical effect of root instances in the Chinese mainland

- A method to measure the catchment area of domestic instances
- Findings:
  - Unshared root instances
  - Impact of deploying domestic root instances

# Measuring the Practical Effect of DNS Root Server Instances:
## A China-Wide Case Study

Fenglu Zhang, Chaoyi Lu, Baojun Liu,
Haixin Duan and Ying Liu

zfl20@mails.tsinghua.edu.cn