APNet 2024
Co-located with ACM SIGCOMM 2024

Rethinking DNS Configuration Verification with a Distributed Architecture

Yao Wang[1], Kexin Yu[1], Ziyi Wang[1], Kaiqiang Hu[2], Haizhou Du[2], Qiao Xiang[1],
Xing Fang[1], Geng Li[3], Ruiting Zhou[4], Linghe Kong[5], Jiwu Shu[6,1]
[1]Xiamen University, [2]Shanghai University of Electric Power, [3]HUAWEI,
[4]Southeast University, [5]Shanghai Jiao Tong University, [6]Minjiang University

August 2024, Sydney, Australia,
ACM APNET'24

DNS Misconfiguration Leads to Disastrous Impact

CLOUDFLARE The Cloudflare Blog

Subscribe to receive notifications of new posts: Email Address Subscribe

Product News | Speed & Reliability | Security | Zero Trust | Developers | AI | Policy | Partners | Life at Cloudflare

1.1.1.1 lookup failures on October 4, 2023

2023-10-04

Ólafur Guðmundsson

9 min read

This post is also available in 简体中文, 繁體中文, 日本語, Deutsch, Français, 한국어 and Español.

On 4 October 2023, Cloudflare experienced DNS resolution problems starting at 07:00 UTC and ending at 11:00 UTC. Some users of 1.1.1.1 or products like WARP, Zero Trust, or third party DNS resolvers which use 1.1.1.1 may have received SERVFAIL DNS responses to valid queries. We're very sorry for this outage. This outage was an internal software error and not the result of an attack. In this blog, we're going to talk about what the failure was, why it occurred, and what we're doing to make sure this doesn't happen again.

Background

Engineering

DNS Outage Post Mortem

Last week on Wednesday, January 8th, GitHub experienced an outage of our DNS infrastructure. As a result of this outage, our customers experienced 42 minutes of downtime of services along...

DNS Outage at iFastNet: Softaculous down

System Issues

Admin Owner of InfinityFree 1 Aug 2019

Since August 11 22:00 UTC, there appears to be an outage at iFastNet's main DNS servers. This means that iFastNet's own domains (primarily ifastnet.com 16 and it's subdomains) are currently unreachable. The DNS servers for hosted websites are still working.

SIGN IN / UP The Register

PATCHES

15

Just one bad packet can bring down a vulnerable DNS server thanks to DNSSEC

'You don't have to do more than that to disconnect an entire network' *El Reg* told as patches emerge

Thomas Claburn Tue 13 Feb 2024 // 23:27 UTC

Source:
https://github.blog/2014-01-18-dns-outage-post-mortem/
https://forum.infinityfree.com/t/dns-outage-at-ifastnet-softaculous-down/19374
https://blog.cloudflare.com/1-1-1-1-lookup-failures-on-october-4th-2023
https://therecord.media/russia-top-level-domain-internet-outage-dnssec

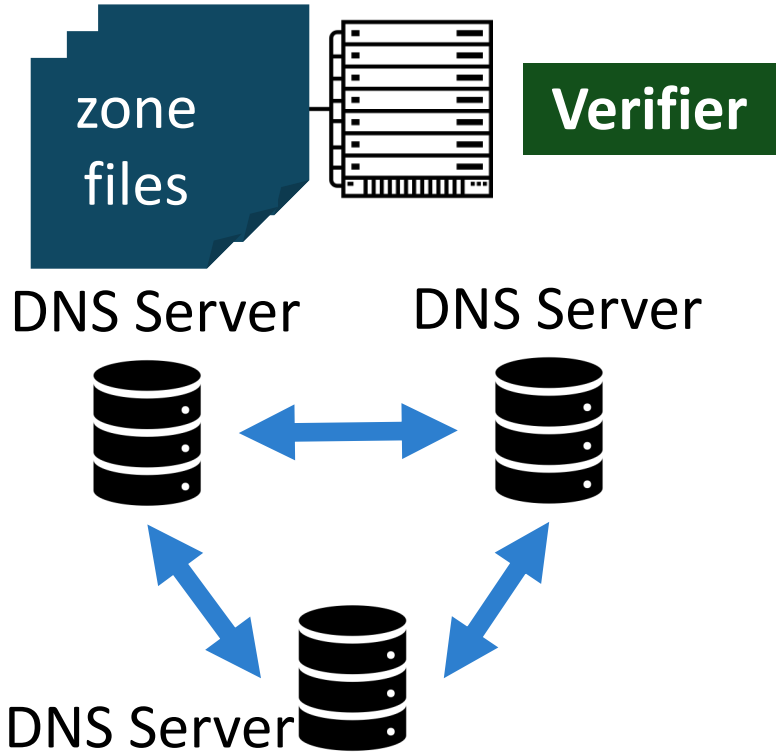
Related Work

- **Static analysis on single zone files**
  - Bind9, ThounsandEyes, …
  - **Limitaion**: serious completeness deficiency

- **DNS configuration verification**
  - GROOT[SIGCOMM'20], Formal Framework[SIGCOMM'23], …
  - **Limitation**:
    - **Scalability** – takes GRoot ~3 hours to complete verification for over 10 million RRs
    - **Privacy** – requires trusted third party to collect the zone files

Our Proposal: Offload Centralized Verification to Distributed Computations

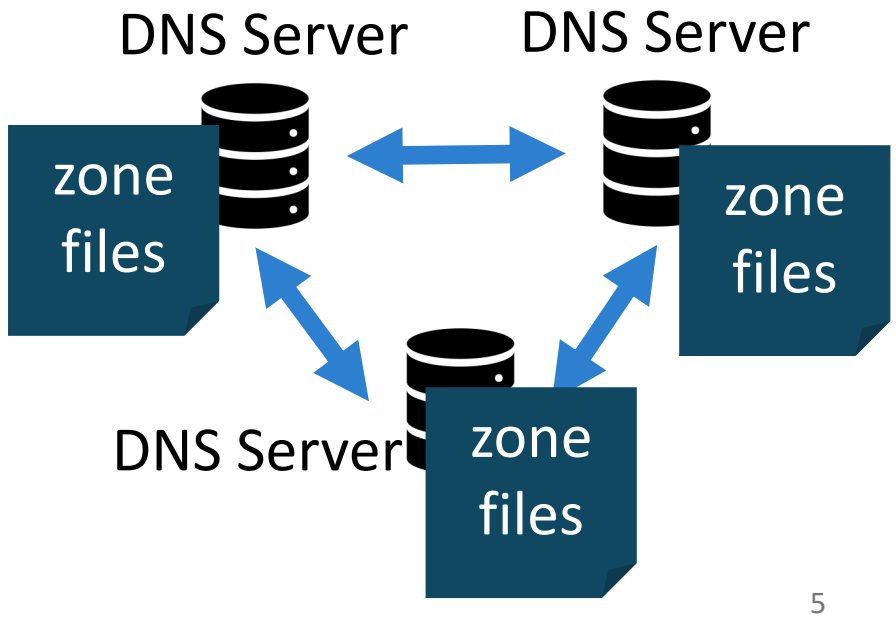
- **Basic idea**: each DNS server acts as a local verifier and exchange local verification results



Our Proposal: Offload Centralized Verification to Distributed Computations

- **Basic idea**: each DNS server acts as a local verifier and exchange local verification results

- **Benefits**:
  - No performance bottleneck
    - Verifier analysis the behavior of the zone file on its own
  - No need for a trusted third party
    - Each server holds its own zone files

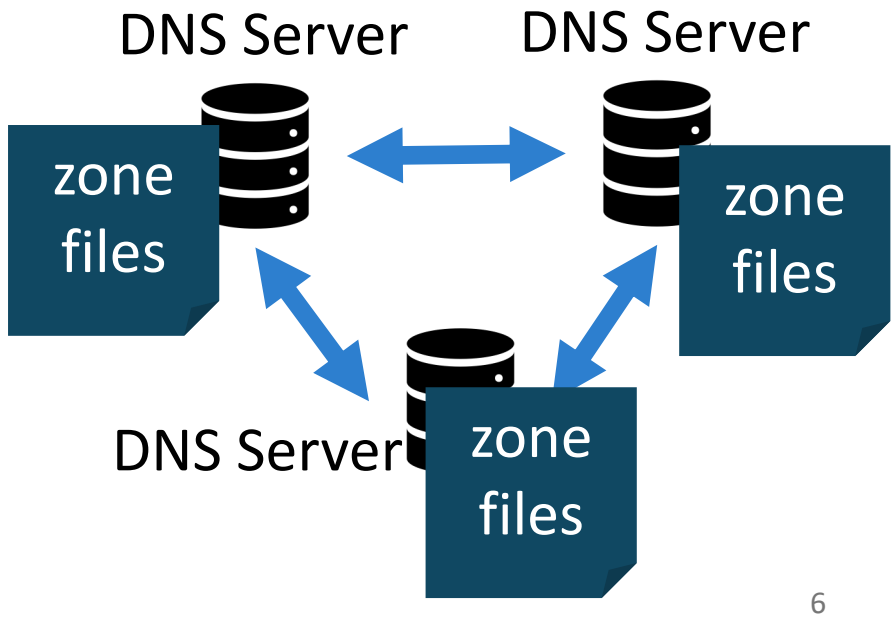


Our Proposal: Offload Centralized Verification to Distributed Computations

- **Basic idea**: each DNS server acts as a local verifier and exchange local verification results

How to realize it?

- **Benefits**:
  - No performance bottleneck
    - Verifier analysis the behavior of the zone file on its own
  - No need for a trusted third party
    - Each server holds its own zone files



An Illustrating Example

- **Setting**: the IP addresses of domain cs.com. is 10.26.43.0/24

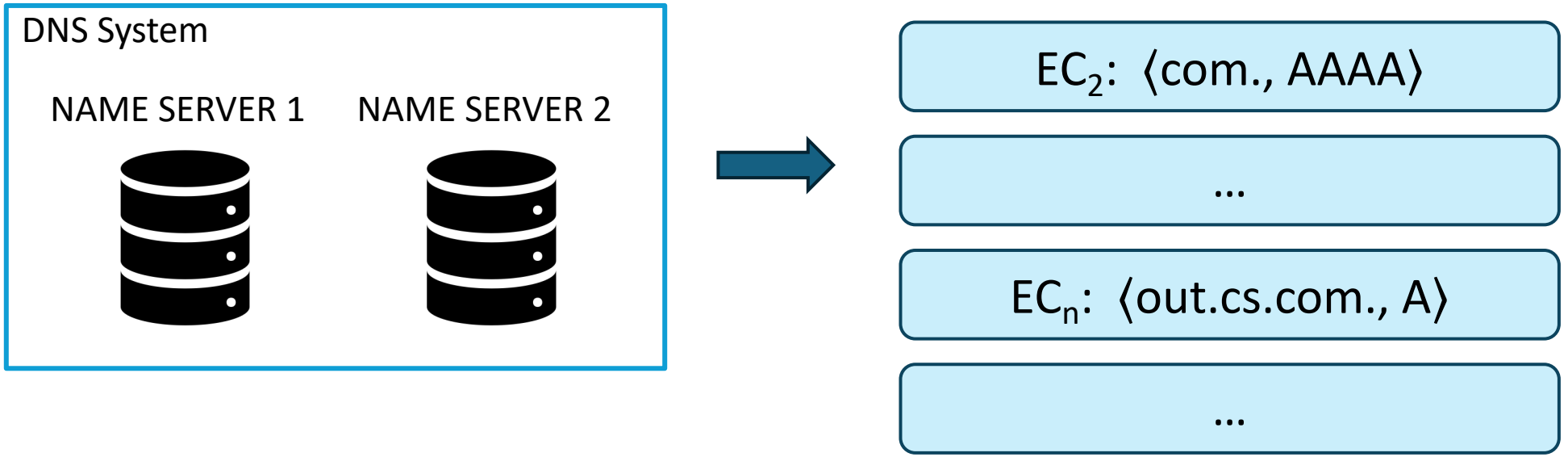
- **Two DNS servers**:

NAME SERVER 1		
$ORIGIN com.		
cs.com.	NS	ns1.cs.com
ns1.cs.com	A	10.26.43.177

NAME SERVER 2		
$ORIGIN cs.com.		
ignore.new.cs.com.	A	10.26.43.164
new.cs.com.	DNAME	new.com.
out.cs.com.	A	10.58.44.134
*.cs.com.	A	10.26.43.166

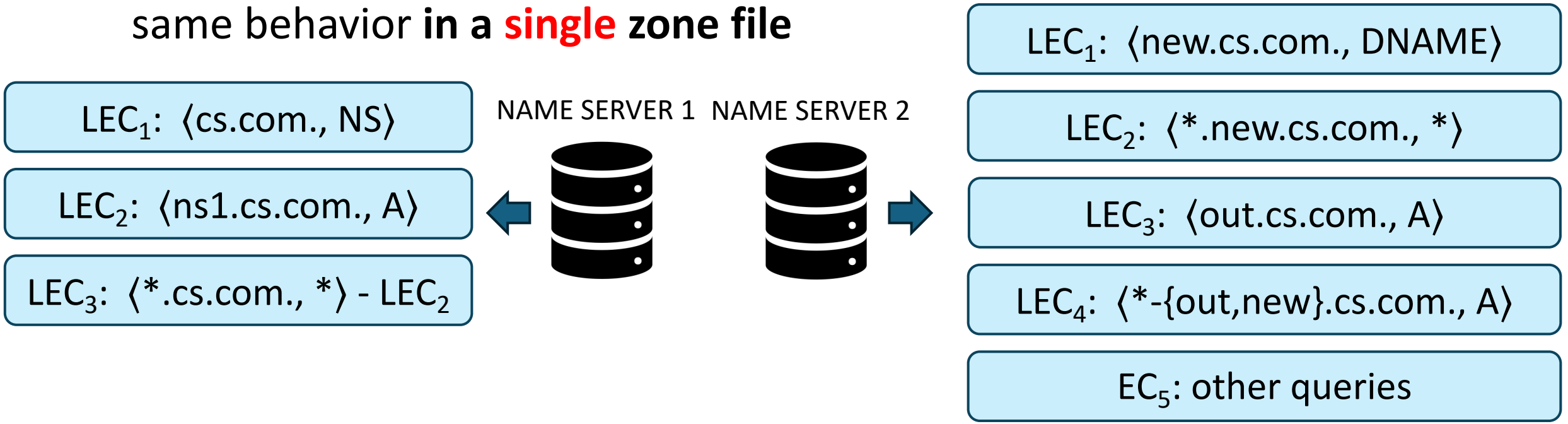
Design Point 1:
Local Equivalence Class (LEC) for Zone File

- **EC**: a equivalence class (EC) is a set of DNS queries with the same behavior **in the whole DNS system**
- **LEC**: a local equivalence class (LEC) is a set of DNS queries with the same behavior **in a single zone file**



Design Point 1:
Local Equivalence Class (LEC) for Zone File

- **EC**: a equivalence class (EC) is a set of DNS queries with the same behavior **in the whole DNS system**
- **LEC**: a local equivalence class (LEC) is a set of DNS queries with the same behavior **in a single zone file**

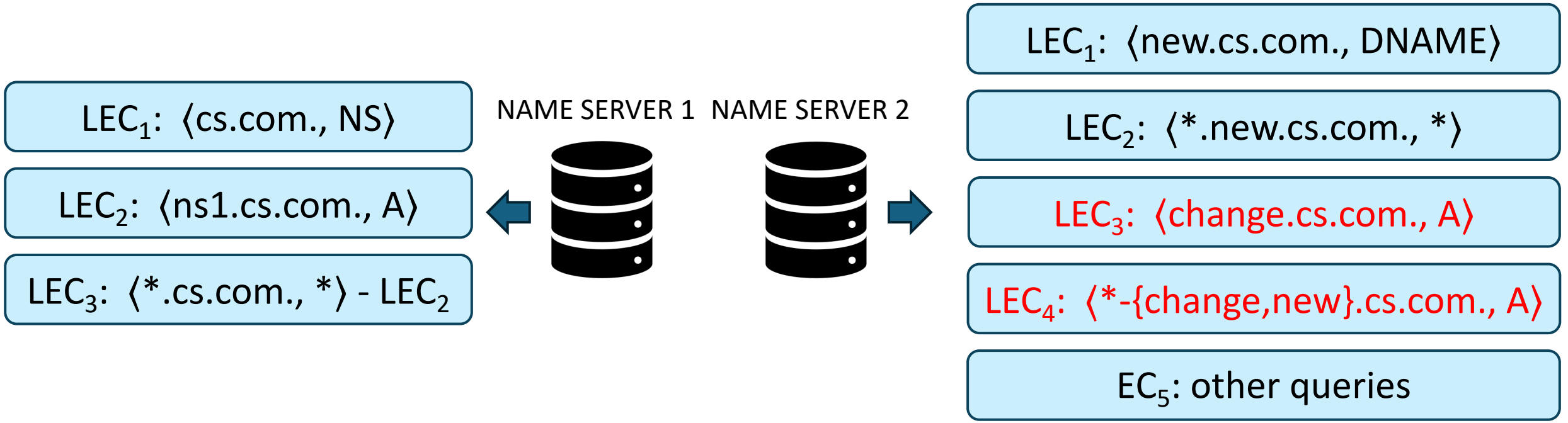


LEC is updated only for updated zone files

NAME SERVER 2		
$ORIGIN cs.com.		
ignore.new.cs.com.	A	10.26.43.164
new.cs.com.	DNAME	new.com.
out.cs.com.	A	10.58.44.134
*.cs.com.	A	10.26.43.166

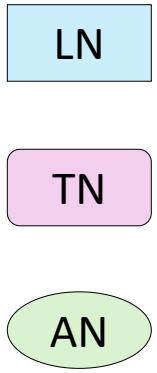
NAME SERVER 2		
$ORIGIN cs.com.		
ignore.new.cs.com.	A	10.26.43.164
new.cs.com.	DNAME	new.com.
change.cs.com.	A	10.58.44.134
*.cs.com.	A	10.26.43.166

LEC is updated only for updated zone files



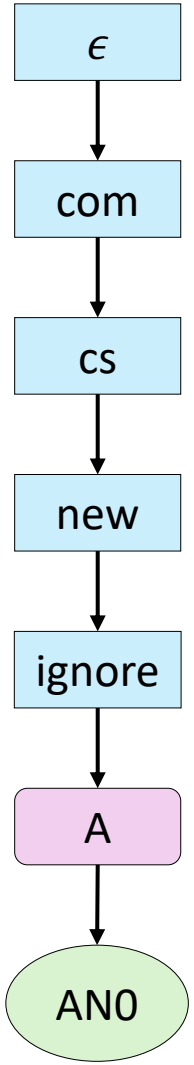
Computing LEC: Action Trie (AT)

Three types of nodes:

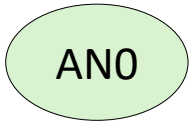


Label node	Used to match query domains
Type node	Used to match query types
Action node	Used to perform the operation after matching

Computing LEC: Action Trie (AT)

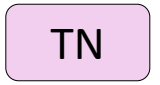


NAME SERVER 2		
$ORIGIN cs.com.		
ignore.new.cs.com.	A	10.26.43.164
new.cs.com.	DNAME	new.com.
out.cs.com.	A	10.58.44.134
*.cs.com.	A	10.26.43.166

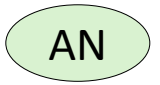


Return value 10.26.43.164

Label node

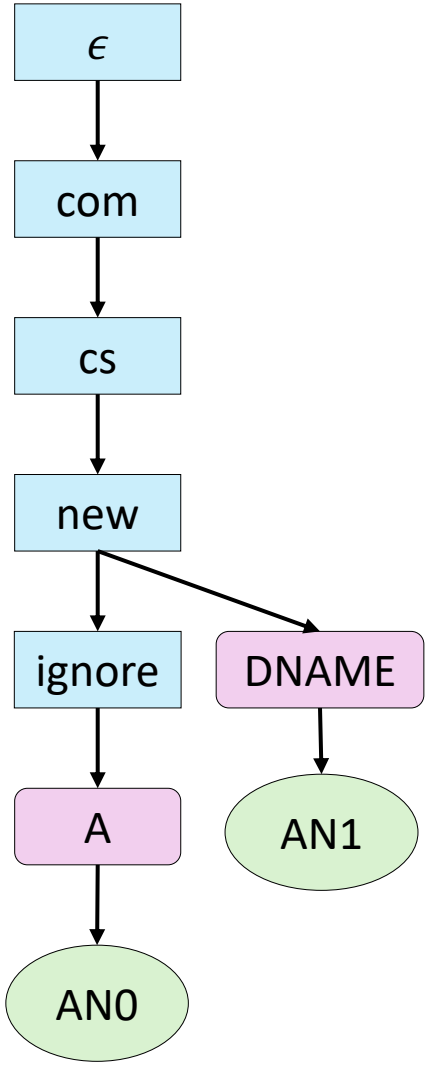


Type node

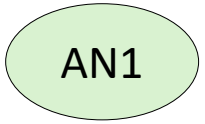


Action node

Computing LEC: Action Trie (AT)



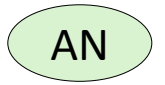
NAME SERVER 2		
$ORIGIN cs.com.		
ignore.new.cs.com.	A	10.26.43.164
new.cs.com.	DNAME	new.com.
out.cs.com.	A	10.58.44.134
*.cs.com.	A	10.26.43.166



Return value new.com.

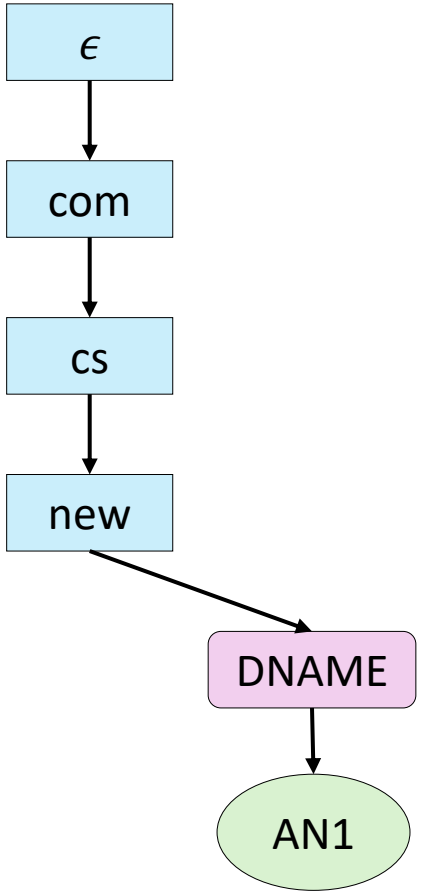
Label node

Type node

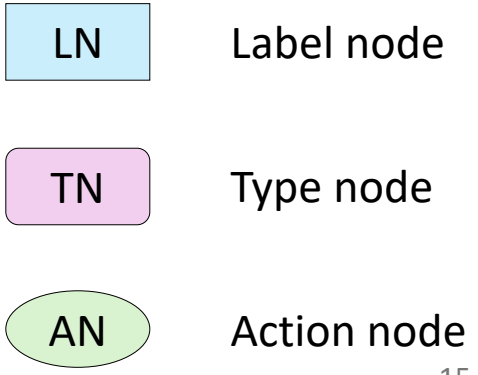


Action node

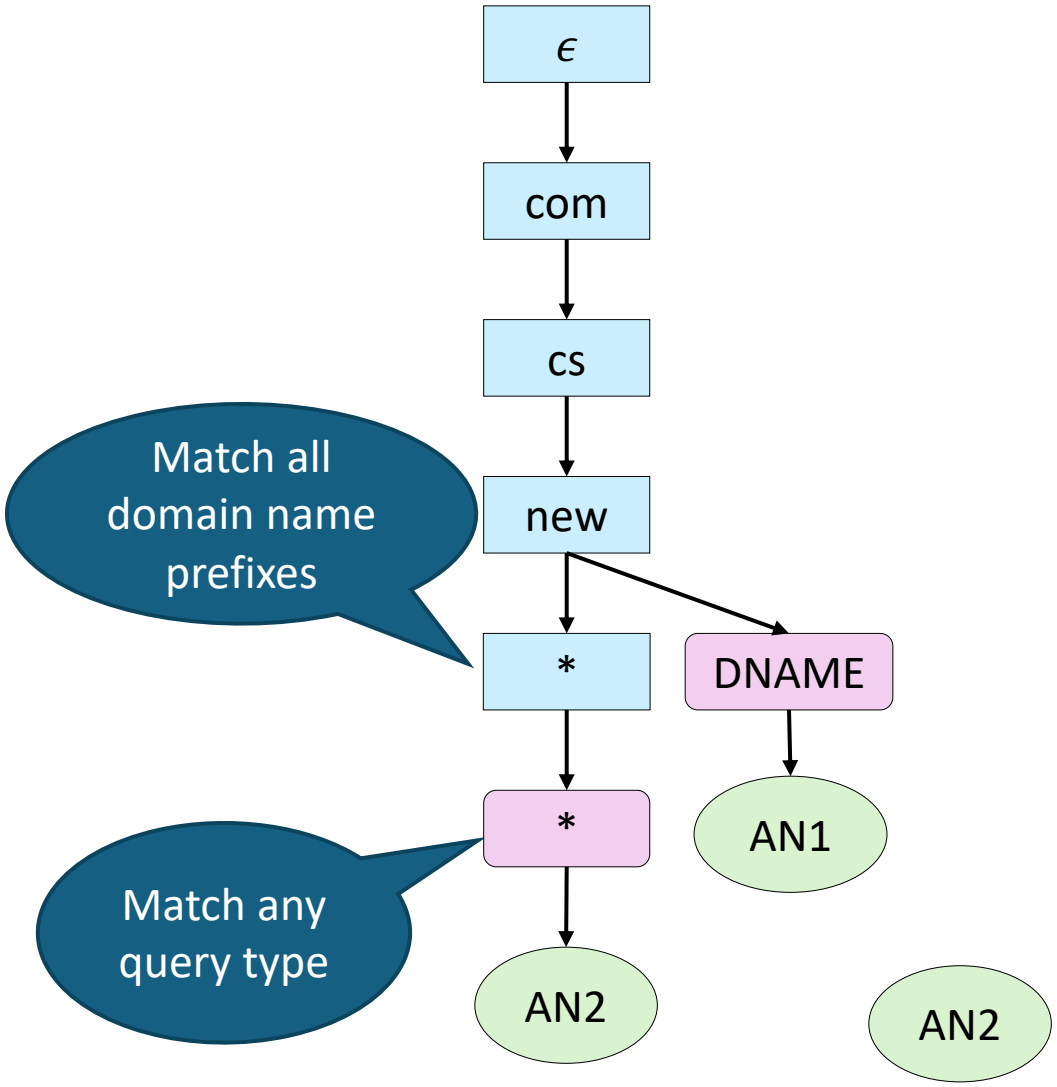
Computing LEC: Action Trie (AT)



NAME SERVER 2		
$ORIGIN cs.com.		
ignore.new.cs.com.	A	10.26.43.164
new.cs.com.	DNAME	new.com.
out.cs.com.	A	10.58.44.134
*.cs.com.	A	10.26.43.166



Computing LEC: Action Trie (AT)



NAME SERVER 2		
$ORIGIN cs.com.		
ignore.new.cs.com.	A	10.26.43.164
new.cs.com.	DNAME	new.com.
out.cs.com.	A	10.58.44.134
*.cs.com.	A	10.26.43.166

AN2

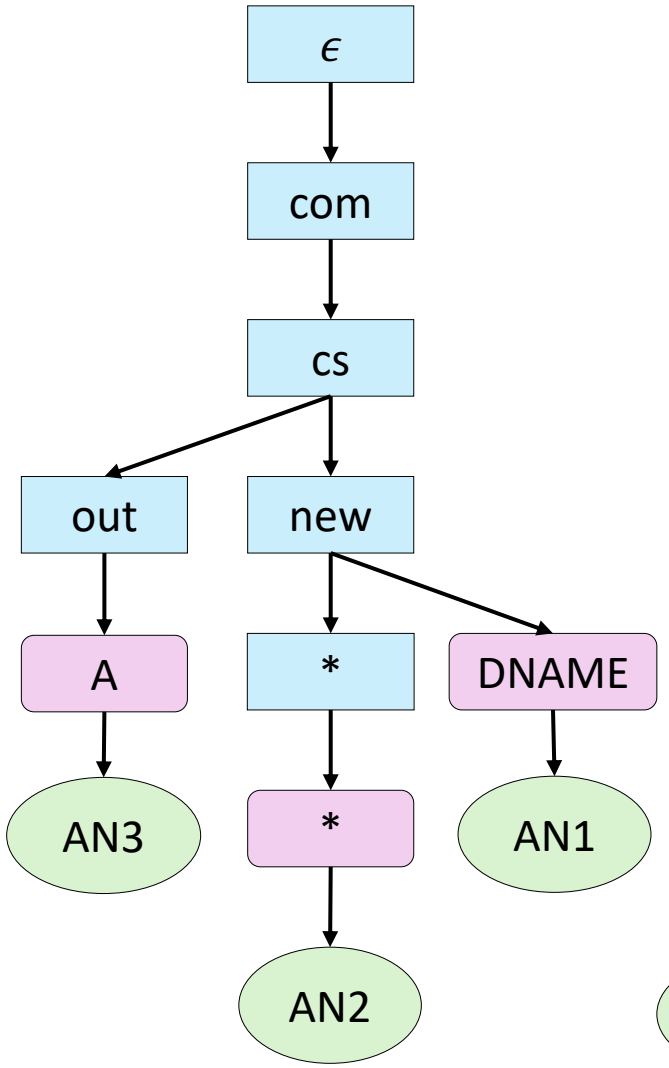
Replace new.cs.com. with new.com. in the query dmain and forward it

LN — Label node

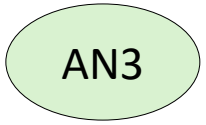
TN — Type node

AN — Action node

Computing LEC: Action Trie (AT)

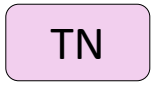


NAME SERVER 2		
$ORIGIN cs.com.		
ignore.new.cs.com.	A	10.26.43.164
new.cs.com.	DNAME	new.com.
out.cs.com.	A	10.58.44.134
*.cs.com.	A	10.26.43.166

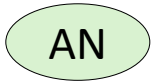


Return value 10.58.44.134

Label node

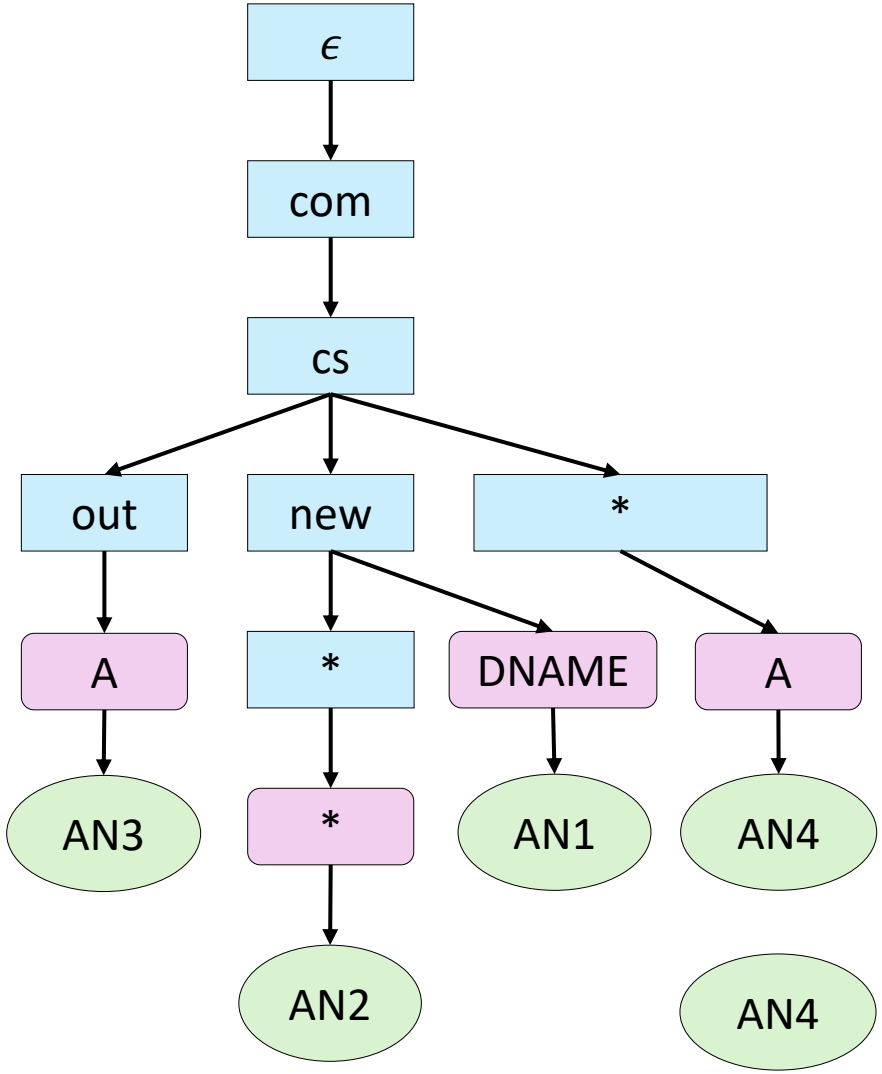


Type node



Action node

Computing LEC: Action Trie (AT)

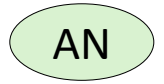


NAME SERVER 2		
$ORIGIN cs.com.		
ignore.new.cs.com.	A	10.26.43.164
new.cs.com.	DNAME	new.com.
out.cs.com.	A	10.58.44.134
*.cs.com.	A	10.26.43.166

Return value 10.26.43.166

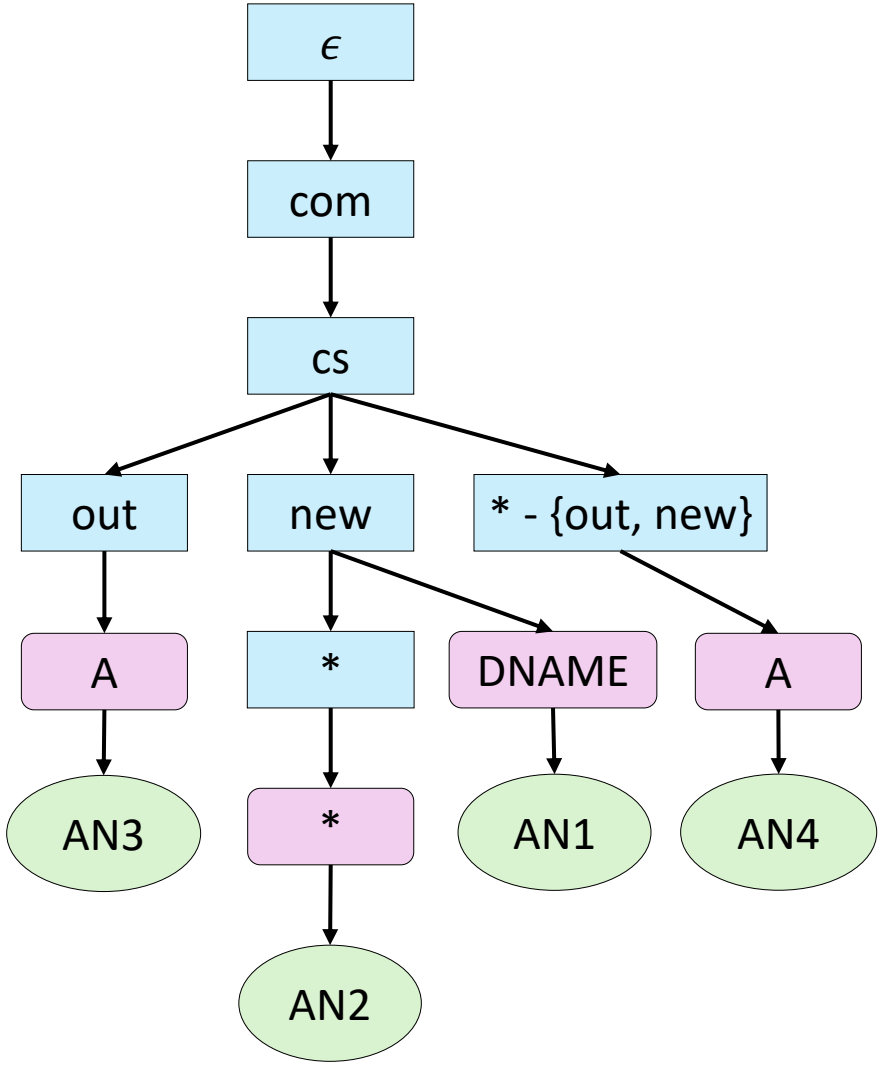
Label node

Type node

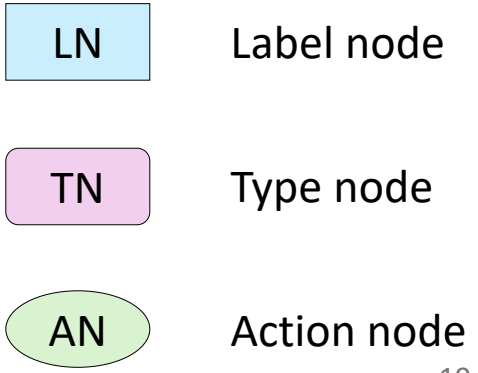


Action node

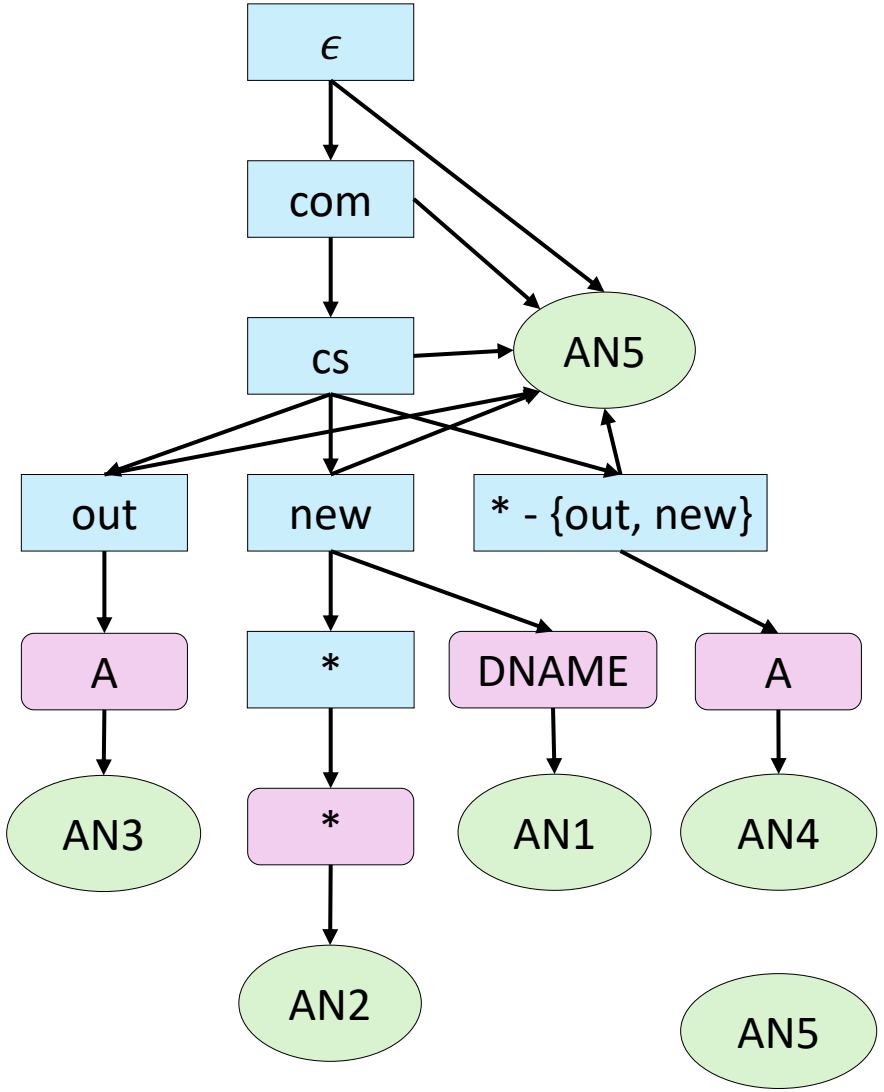
Computing LEC: Action Trie (AT)



NAME SERVER 2		
$ORIGIN cs.com.		
ignore.new.cs.com.	A	10.26.43.164
new.cs.com.	DNAME	new.com.
out.cs.com.	A	10.58.44.134
*.cs.com.	A	10.26.43.166

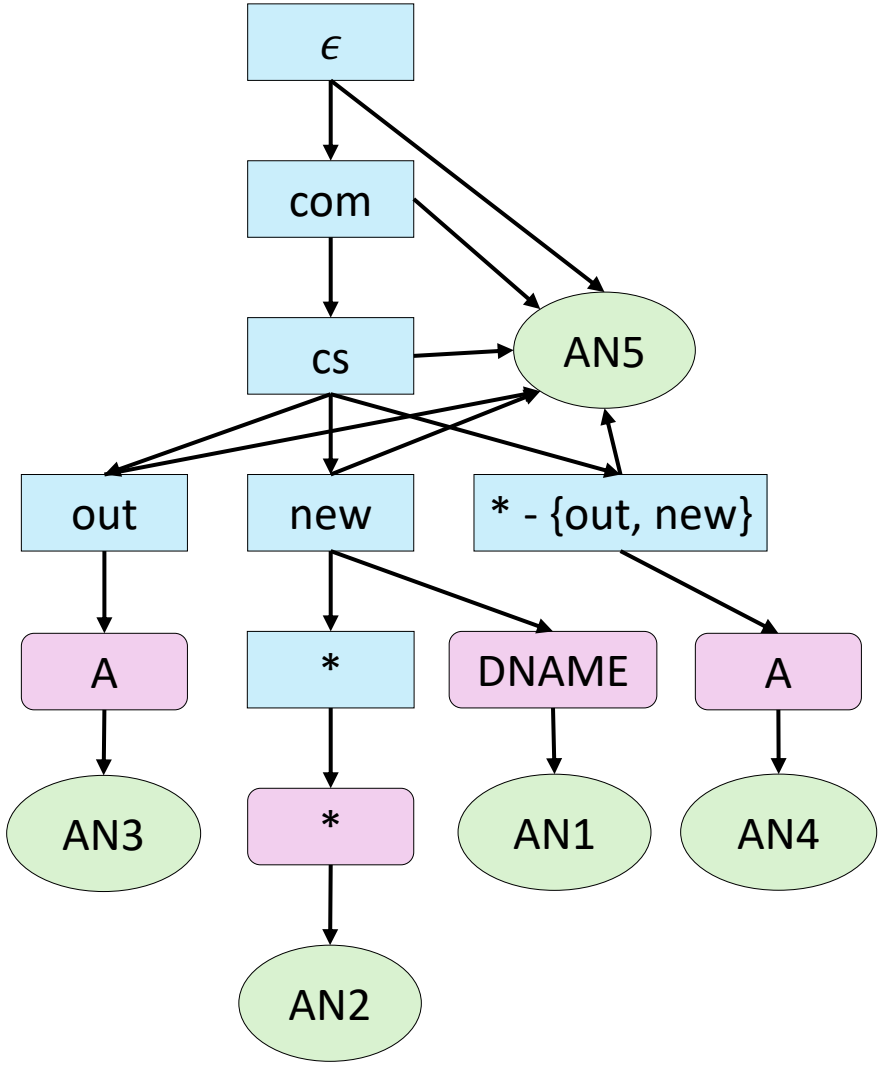


Computing LEC: Action Trie (AT)



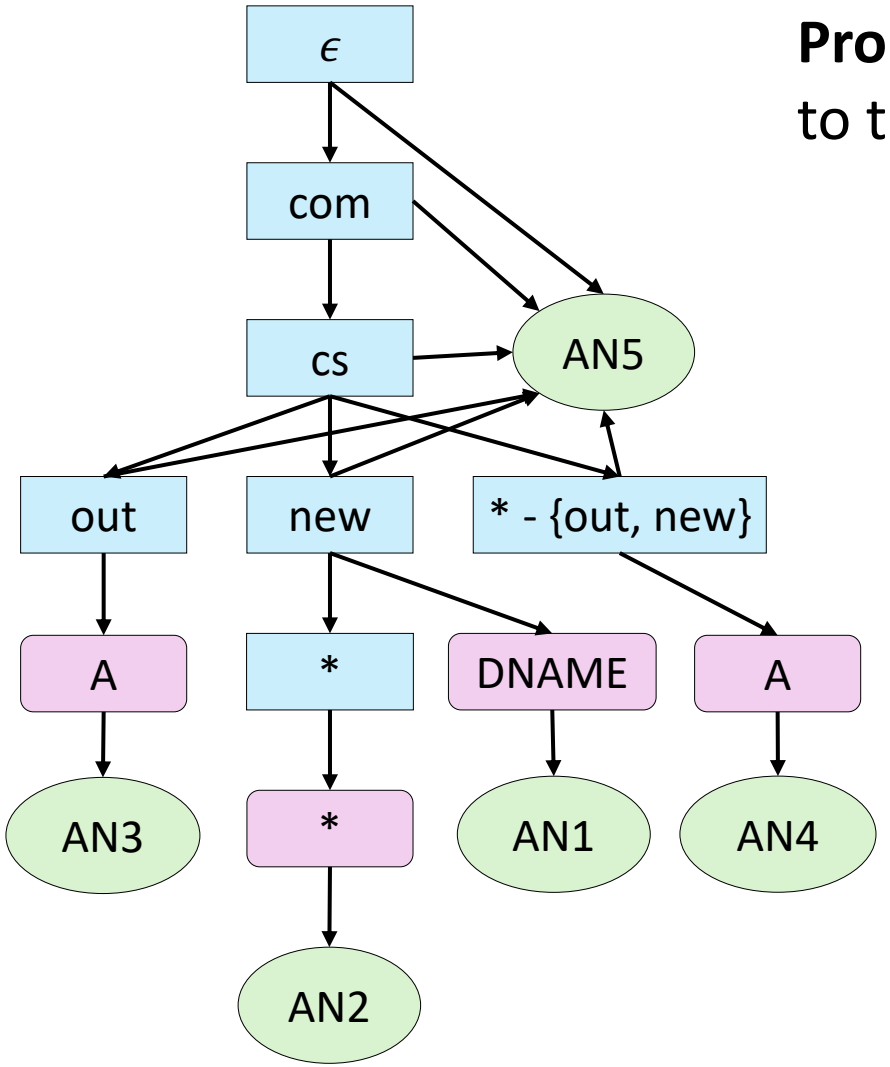
NAME SERVER 2		
$ORIGIN cs.com.		
ignore.new.cs.com.	A	10.26.43.164
new.cs.com.	DNAME	new.com.
out.cs.com.	A	10.58.44.134
*.cs.com.	A	10.26.43.166

Analysis



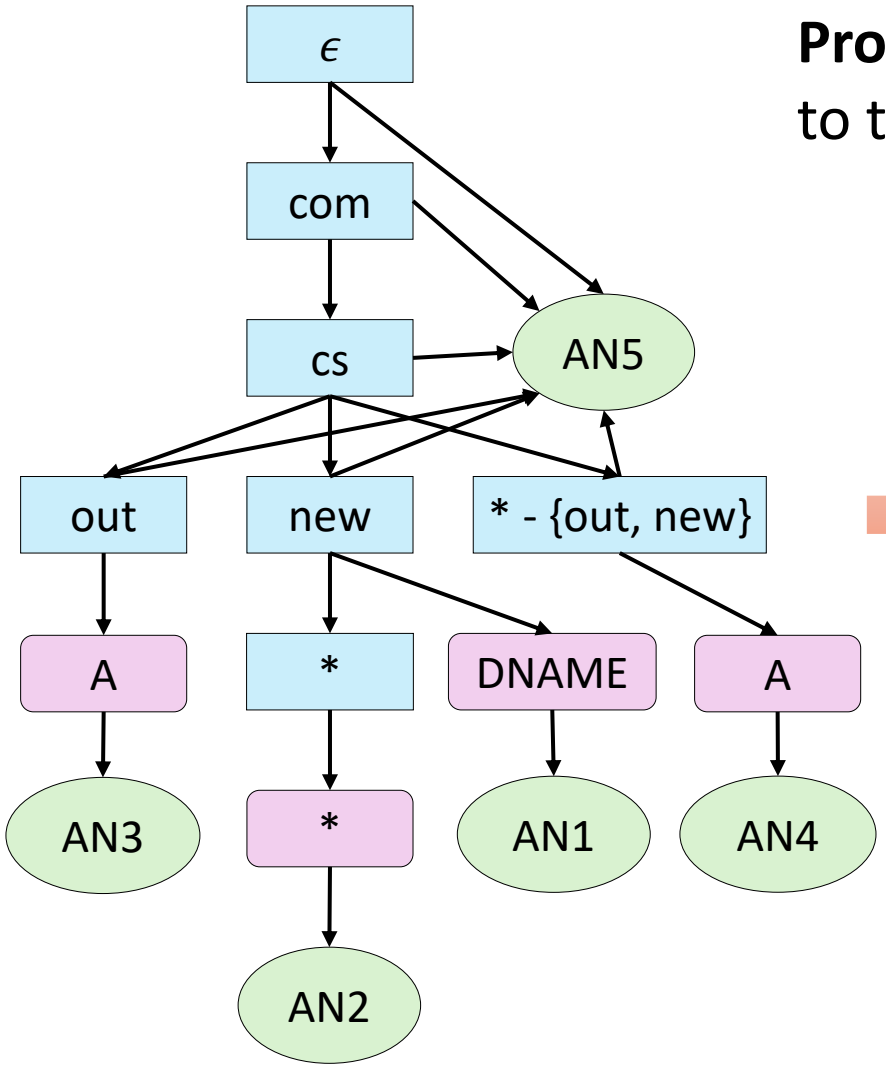
Why AT?

Analysis



Proposition 1: AT is able to compute fewer LECs/ECs compared to the Label Graph in GROOT with the same zone file(s).

Analysis



Proposition 1: AT is able to compute fewer LECs/ECs compared to the Label Graph in GROOT with the same zone file(s).

LEC_1: ⟨new.cs.com., DNAME⟩

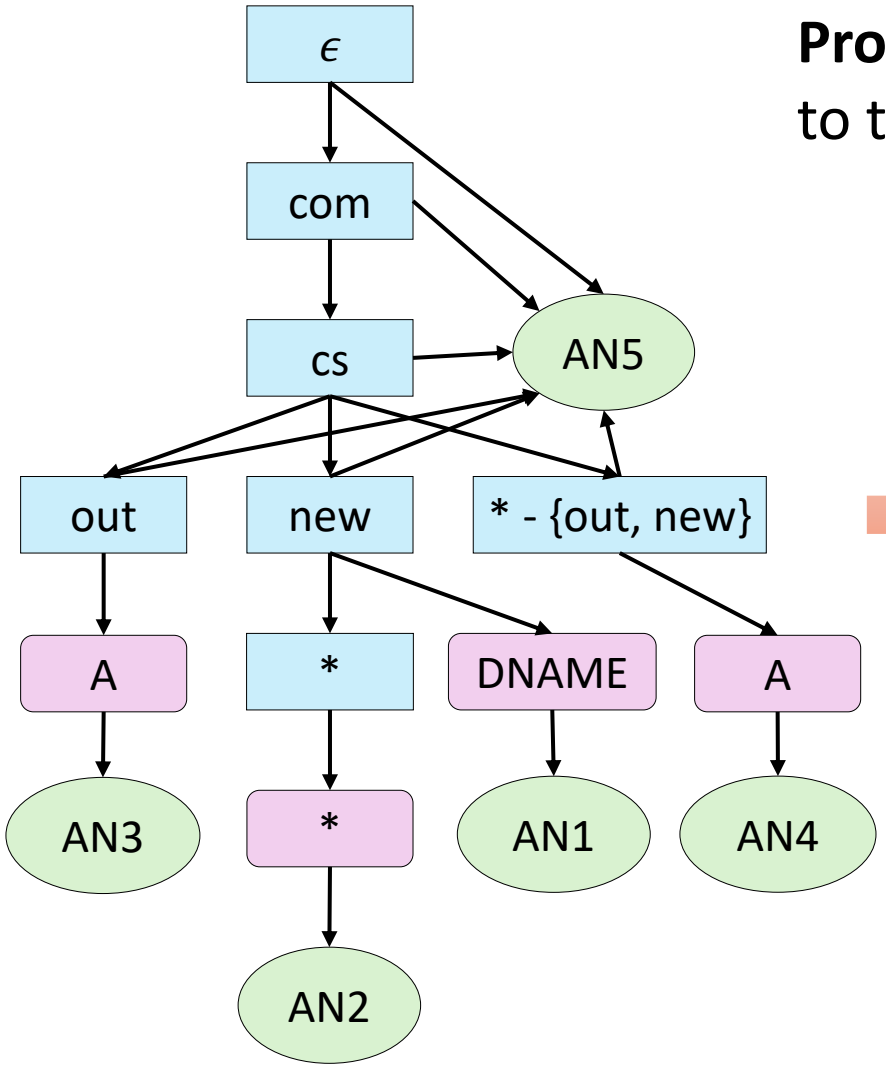
LEC_2: ⟨*.new.cs.com., *⟩

LEC_3: ⟨out.cs.com., A⟩

LEC_4: ⟨*-{out,new}.cs.com., A⟩

EC_5: other queries

Analysis



Proposition 1: AT is able to compute fewer LECs/ECs compared to the Label Graph in GROOT with the same zone file(s).

LEC_1: ⟨new.cs.com., DNAME⟩

LEC_2: ⟨*.new.cs.com., *⟩

LEC_3: ⟨out.cs.com., A⟩

LEC_4: ⟨*-{out,new}.cs.com., A⟩

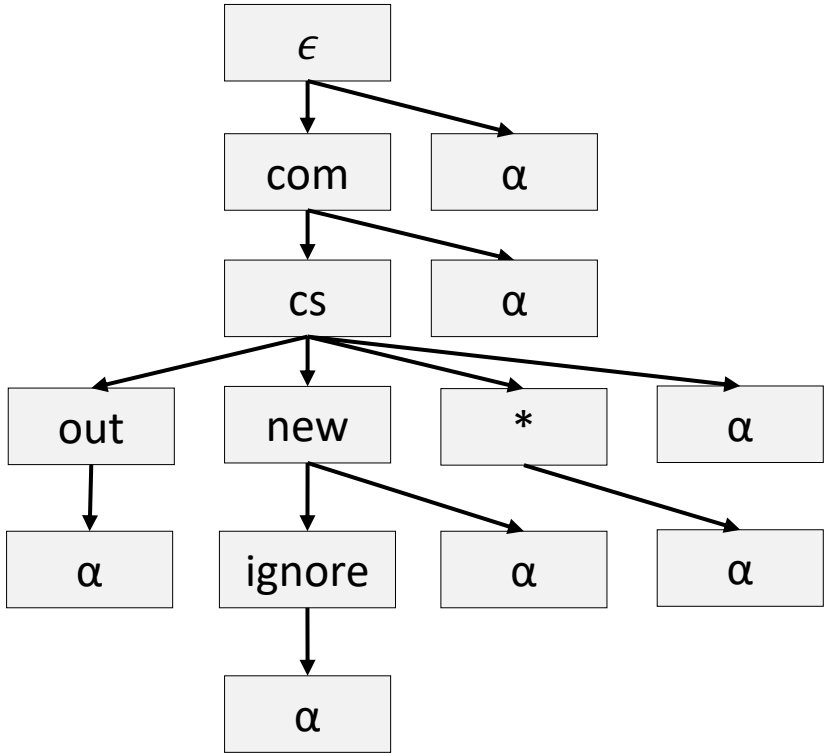
EC_5: other queries

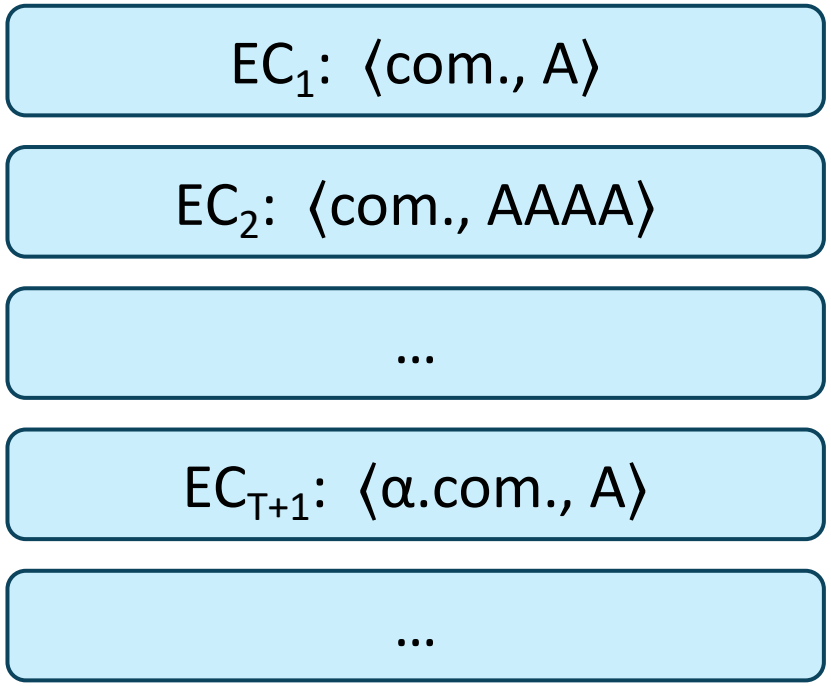
AT computes 5 LECs

Analysis

Proposition 1: AT is able to compute fewer LECs/ECs compared to the Label Graph in GROOT with the same zone file(s).

AT computes 5 LECs

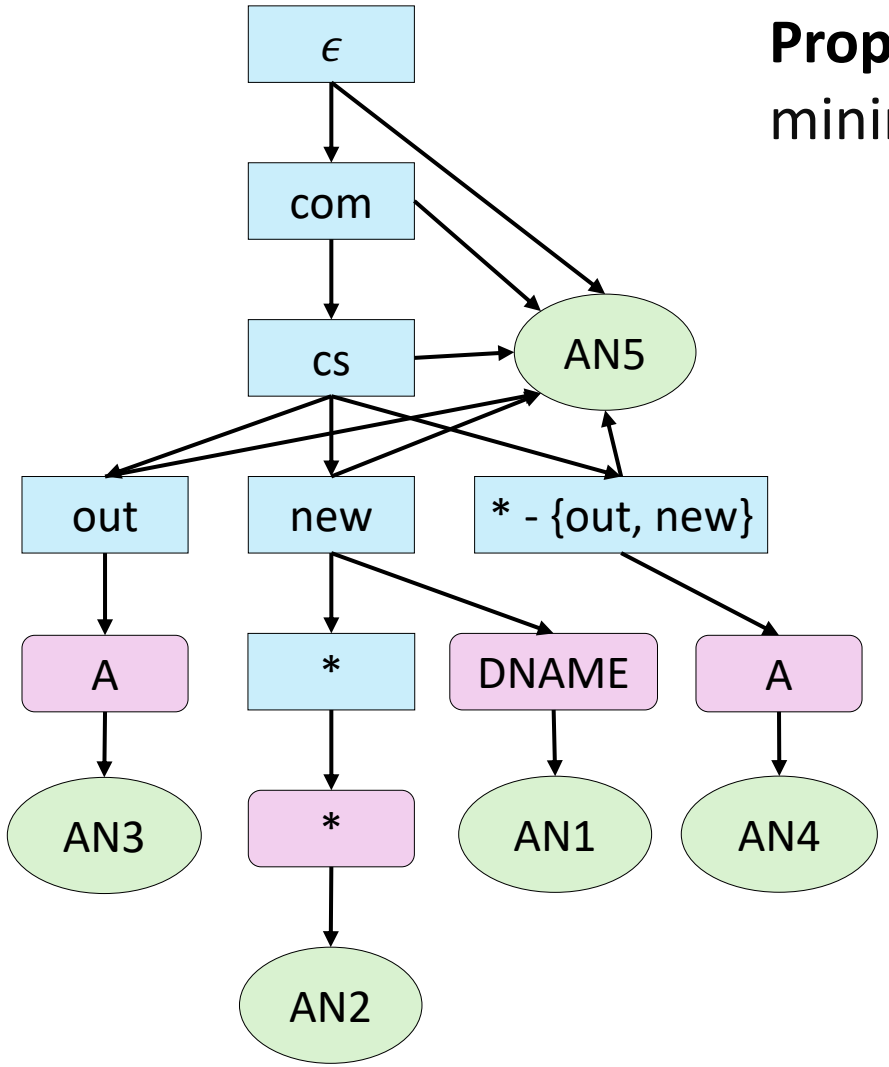




LT computes 13*T ECs

(T is # of query types)

Analysis



Proposition 2: Under our definition of LEC, AT can compute the minimum number of LECs

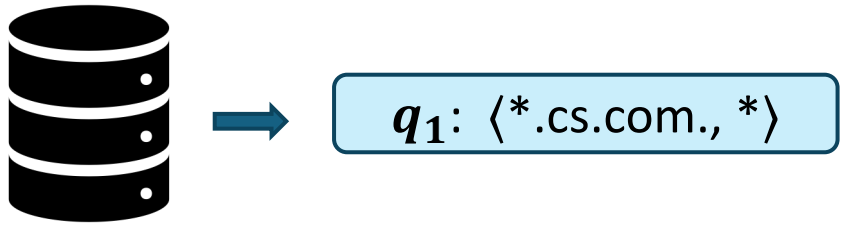
# of LECs = # of ANs

No intersection between ANs

Design Point 2: Symbolic DNS Query

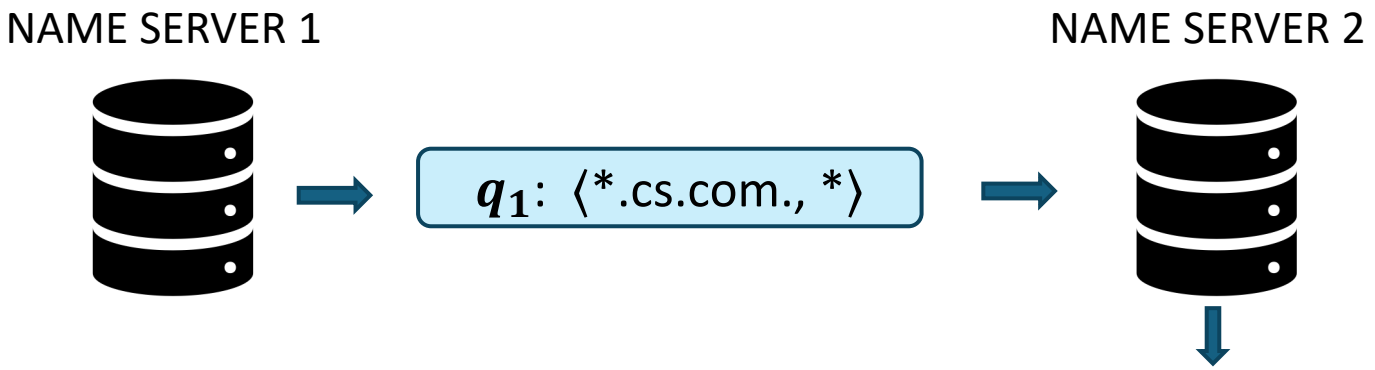
- Suppose we want to verify two properties
  - The IP addresses for cs.com must be 10.26.43.0/24
  - cs.com must have MX records

NAME SERVER 1



Design Point 2: Symbolic DNS Query

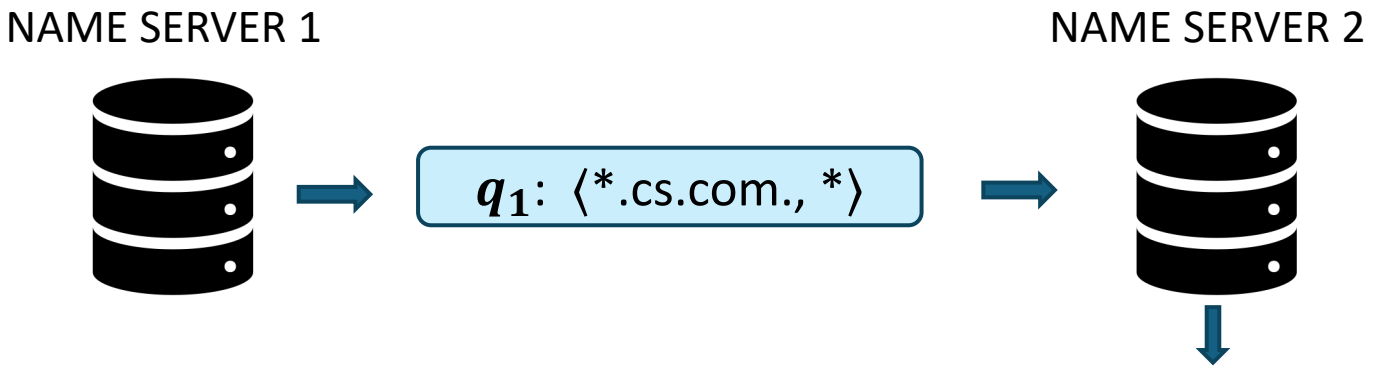
- Suppose we want to verify two properties
  - The IP addresses for cs.com must be 10.26.43.0/24
  - cs.com must have MX records



LECs leading to fault	Action description
⟨out.cs.com., A⟩	Return value 10.58.44.134
⟨*.cs.com., MX⟩	Return NXDOMAIN.

Design Point 2: Symbolic DNS Query

- Suppose we want to verify two properties
  - The IP addresses for cs.com must be 10.26.43.0/24
  - cs.com must have MX records



LECs leading to fault	Action description
⟨out.cs.com., A⟩	Return value 10.58.44.134
⟨*.cs.com., MX⟩	Return NXDOMAIN.

Not in 10.26.43.0/24

No MX records

Supported Properties: the Same as GROOT

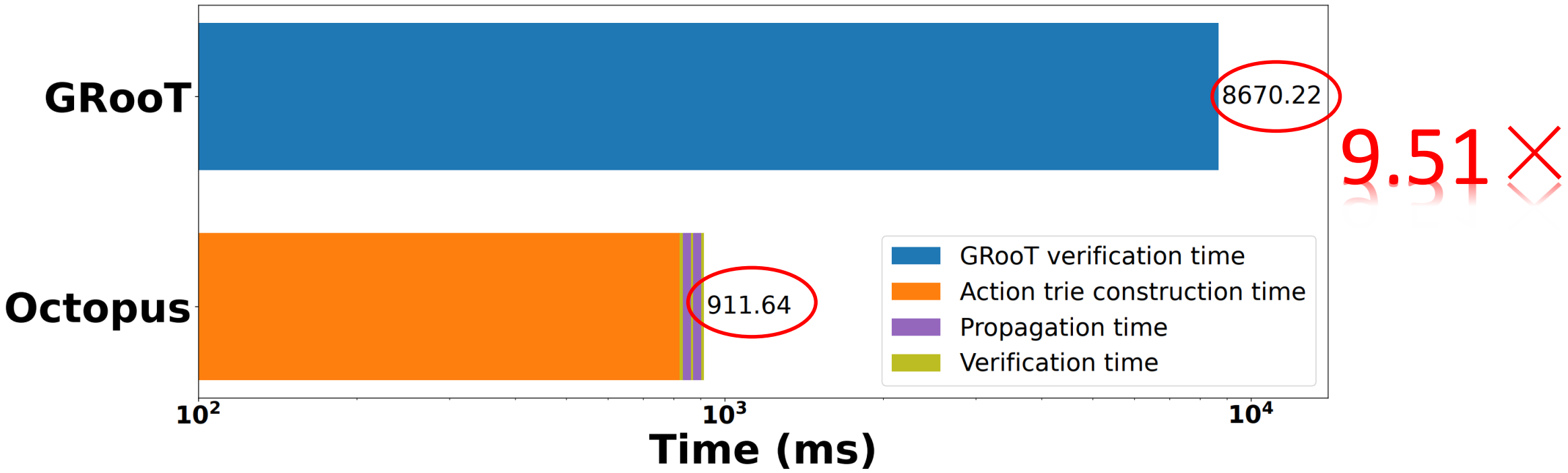
Support Property	
Delegation Inconsistency	Domain Overflow
Lame Delegation	Answer Inconsistency
Non-Existent Domain	Rewrite Blackholing
Cyclic Zone Dependency	Repeated Query
Rewrite Loop	Domain Overflow at Nameserver
……	

Experiment 1: Testbed Experiments

- **Environment:**
  - Distributed testbed (server 1-6) vs. centralized verifier (server 7)
- **Properties:**
  - Loop: same query received ≥ 2 times
  - Query exceeds the maximum length (after the DNAME rewrites the query size>253 characters)
- **Metric:**
  - End-to-end verification time = AT construction time + verification time
  - Propagation latencies = $distance$[China,Japan] / c (lightspeed)

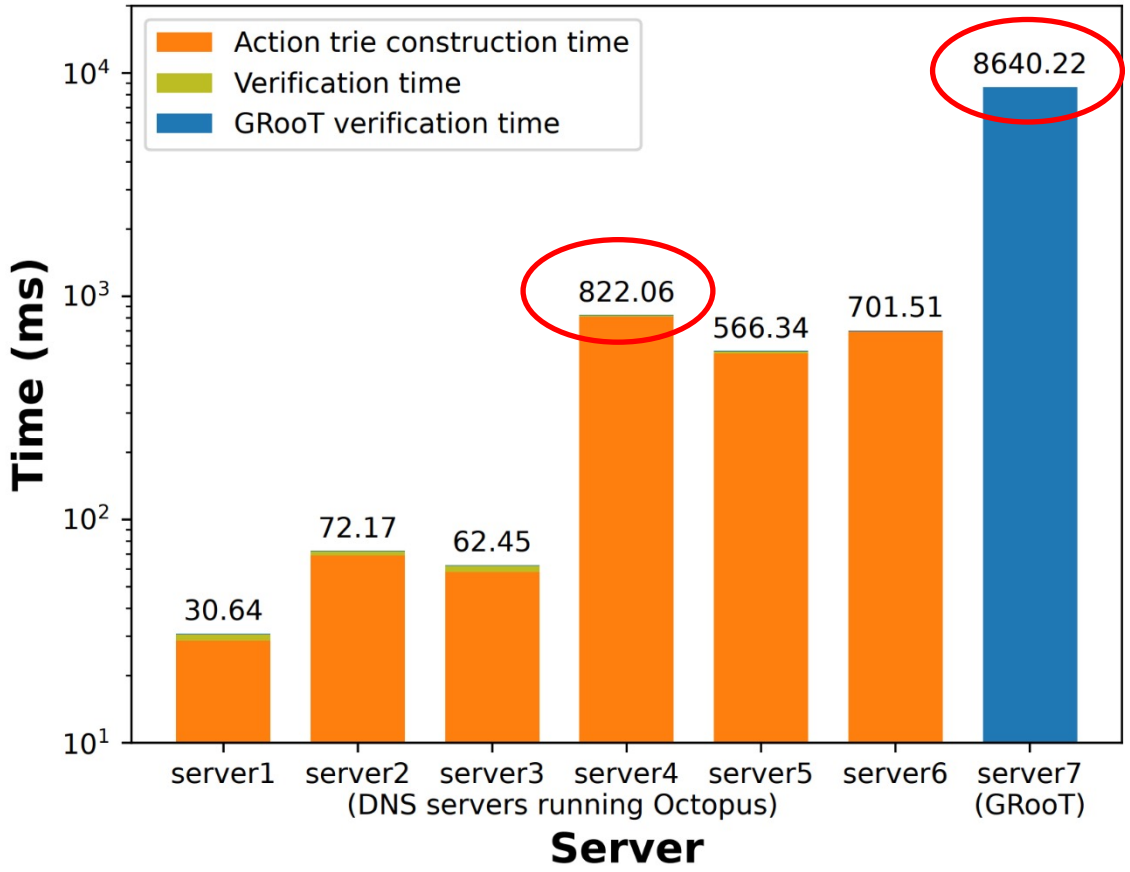
Experiment 1: Testbed Experiments

- **End-to-end verification time (including propagation latency):**
  - 911.64 ms to verify 410,000 resource records, 9.51× faster than GROOT



Experiment 1: Testbed Experiments

- **End-to-end verification time**
  - For single server, 10.5× faster than GROOT



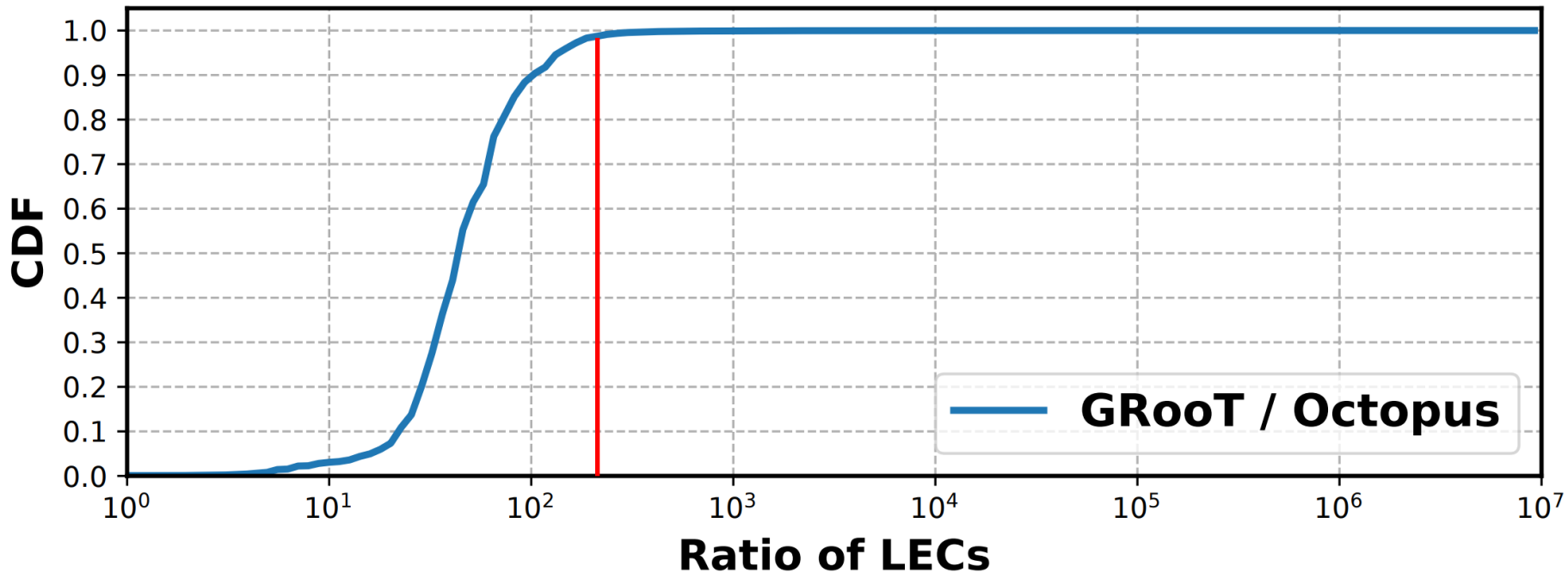
10.5×

Experiment 2: Benefit of LEC

- **Environment:**
  - Our prototype (~1500 lines of Java code), GROOT Label Graph (reproduced by Java)
  - Intel(R) Xeon(R) Gold 6126 CPU 2.60GHz
- **Dataset:**
  - Census: anonymized public datasets, 1,368,523 domains, over 65 million resource records in total
- **Metric:**
  - LECs generation time
  - Number of LECs

Experiment 2: Benefit of LEC

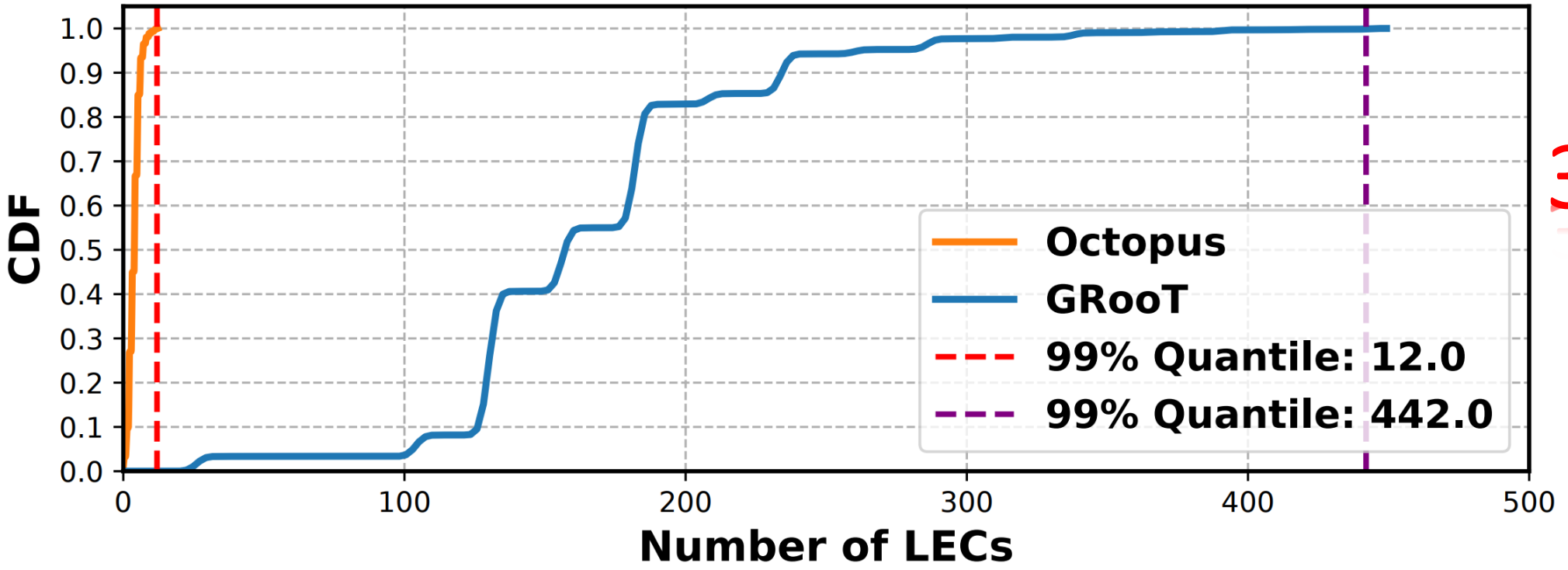
- **Ratio of the number of AT computed LECs/ lable tree computed ECs**
  - GROOT computation is up to over **300X** higher than our prototype



316×

Experiment 2: Benefit of LEC

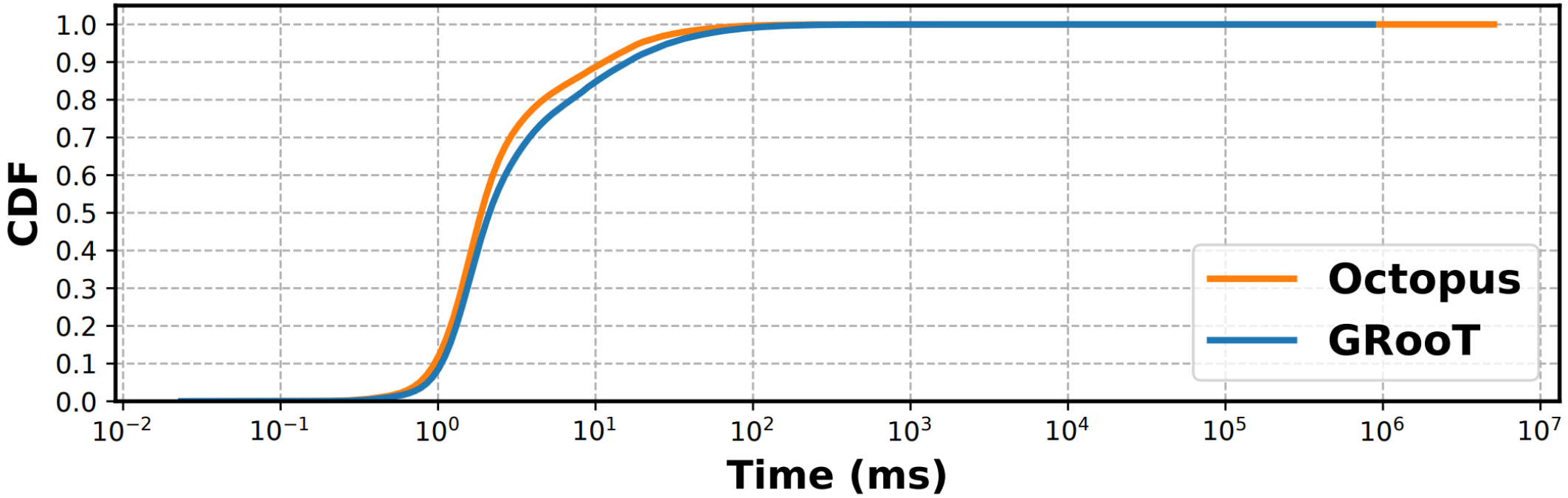
- **Number of LECs/ECs**
  - In the 99% quantile, our prototype consistently outperforms GRooT by **36x**



36.8×

Experiment 2: Benefit of LEC

- **LECs/ECs computing time**
  - Our prototype and GROOT has the same level of LEC/EC computing time (i.e., ≤ 100 ms)



Summary

- **Basic idea**: offload centralized DNS verification to distributed computing
  - **Local equivalence classes** computed for each zone with a novel Action Trie data structure
  - **Symbolic execution** to verify the complete name space with a small number of symbolic queries

- **Future work:**
  - Supporting incremental verification
  - A simple and expressive specification language for DNS invariants
  - More extensive experiments