A Formal Framework for End-to-End DNS Resolution

Si Liu*, Huayi Duan*, Lukas Heimes, Marco Bearzi, Jodok Vieli, David Basin, and Adrian Perrig
*(co-first author)

SIGCOMM 2023, New York
DNS is more complex than you might think

Textbook example for name resolution
DNS is more complex than you might think

Textbook example for name resolution

Client → Recursive resolver
2.0.2.3

Recursive resolver → Name server com.
Refer to NS a.com.

Name server com. → Name server root
Refer to NS com.

www.a.com.? → Name server root
Refer to NS com.

www.a.com.? → Recursive resolver
DNS is more complex than you might think

Textbook example for name resolution
DNS is more complex than you might think

Reality: subqueries to resolve referrals

*Ignore root NS, assume TLD NS addresses are known
DNS is more complex than you might think

Reality: subqueries to resolve referrals

*Ignore root NS, assume TLD NS addresses are known
DNS is more complex than you might think

Reality: subqueries to resolve referrals

*Ignore root NS, assume TLD NS addresses are known
DNS is more complex than you might think

Reality: subqueries to resolve referrals

*Ignore root NS, assume TLD NS addresses are known
DNS is more complex than you might think

Reality: subqueries to resolve referrals

```
Resolver

k.gtld-servers.net.
ZONE net.

ns1.b.net.


l.gtld-servers.net.
ZONE com.

ns2.c.com.

```

Recursion
DNS is more complex than you might think

Reality: subqueries to resolve referrals

Non-determinism
DNS is more complex than you might think

Reality: caching (TTL, positive/negative, concurrency, data credibility, …)
DNS is more complex than you might think

Reality: caching (TTL, positive/negative, concurrency, data credibility, …)
DNS is more complex than you might think

Reality: subqueries, query rewrite, caching, …

“DNS Camel” and beyond:

- Over 300 RFCs
- Over 5000 pages
- Growing at ~2 pages / week

DNS features are entangled!
Unsurprisingly, DNS bugs and vulnerabilities prevail

Frequent outages due to **misconfigurations**

Sources:
- https://www.zdnet.com/
- https://www.datacenterdynamics.com/
- https://www.theregister.com/
Unsurprisingly, DNS bugs and vulnerabilities prevail

Frequent outages due to **misconfigurations**

Frequent discovery of **security vulnerabilities**

- Infinite delegation [DNS-OARC’15]
- Unchained [RAID’15]
- NXNS [SEC’20]
- Zaw [CCS’20]
- SADDNS [CCS’20, CCS’21]
- TsuNAME [IMC’21]
- MaginotDNS [SEC’23]
- NRDelegation [SEC’23]
- PHOENIX DOMAIN [NDSS’23]
- …

Sources:
- https://www.zdnet.com/
- https://www.datacenterdynamics.com/
- https://www.theregister.com/
Reasoning about DNS requires a principled approach

**Break-and-Fix** is insufficient
Reasoning about DNS requires a principled approach

**Break-and-Fix** is insufficient

Need **proactive, systematic & automated** analyses
Reasoning about DNS requires a principled approach

**Break-and-Fix** is insufficient

Need **proactive, systematic & automated** analyses

on a **mathematically precise** DNS model

RFCs are written in **natural language** with **ambiguities** and **underspecifications** … lead to problems!
Our framework — modeling language and scope

Maude: a formal language supporting
✓ Expressive formalism based on \textit{rewriting logic}
✓ \textit{Concurrent} computation with \textit{state}
✓ \textit{Extensive} tools for formal specification & verification
Our framework — modeling language and scope

Maude: a formal language supporting
✓ Expressive formalism based on rewriting logic
✓ **Concurrent** computation with state
✓ **Extensive** tools for formal specification & verification

Our scope: **end-to-end** name resolution up to the **latest** algorithmic refinements in RFC9156

---

**Our model’s RFC coverage**

<table>
<thead>
<tr>
<th>RFC</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1034</td>
<td>Core specification</td>
</tr>
<tr>
<td>1035</td>
<td>Core specification</td>
</tr>
<tr>
<td>2181</td>
<td>Clarifications</td>
</tr>
<tr>
<td>2308</td>
<td>Negative caching</td>
</tr>
<tr>
<td>4592</td>
<td>Wildcards</td>
</tr>
<tr>
<td>6604</td>
<td>RCODE clarifications</td>
</tr>
<tr>
<td>6672</td>
<td>DNAME redirection</td>
</tr>
<tr>
<td>8020</td>
<td>NXDOMAIN clarification</td>
</tr>
<tr>
<td>9156</td>
<td>QNAME minimization</td>
</tr>
</tbody>
</table>

---

**Diagram:**

- **Client**
- Recursive resolver
- Name servers
- Cache
Our framework — modeling language and scope

Maude: a formal language supporting
✓ Expressive formalism based on **rewriting logic**
✓ **Concurrent** computation with **state**
✓ **Extensive** tools for formal specification & verification

Our scope: *end-to-end* name resolution up to the *latest* algorithmic refinements in RFC9156

---

**Our model’s RFC coverage**

<table>
<thead>
<tr>
<th>RFC</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1034</td>
<td>Core specification</td>
</tr>
<tr>
<td>1035</td>
<td>Core specification</td>
</tr>
<tr>
<td>2181</td>
<td>Clarifications</td>
</tr>
<tr>
<td>2308</td>
<td>Negative caching</td>
</tr>
<tr>
<td>4592</td>
<td>Wildcards</td>
</tr>
<tr>
<td>6604</td>
<td>RCODE clarifications</td>
</tr>
<tr>
<td>6672</td>
<td>DNAME redirection</td>
</tr>
<tr>
<td>8020</td>
<td>NXDOMAIN clarification</td>
</tr>
<tr>
<td>9156</td>
<td>QNAME minimization</td>
</tr>
</tbody>
</table>

*Abstracted away in GRoot* [Sigcomm’20]
Our framework — executable DNS semantics

Modeled as labelled transition system in *actor paradigm*

- System dynamics specified by rewriting rules
- Non-deterministic and probabilistic variants
Our framework — executable DNS semantics

Modeled as labelled transition system in *actor paradigm*

- System dynamics specified by rewriting rules
- Non-deterministic and probabilistic variants

Resolve ambiguities whenever possible, e.g., resolver case distinction; otherwise, make them *configurable*, e.g., data credibility rule

---

**Option** | **Definition** | **Default**
--- | --- | ---
rsvMinCredClient | The minimum credibility requirement [21] for data served to a client | 2
rsvMinCredResolver | The equivalent credibility requirement for resolver subqueries | 2
maxMinimiseCount | The MAX_MINIMIZE_COUNT parameter to limit extra work for QMIN [11] | 10
minimiseOneLab | The MINIMIZE_ONE_LAB parameter from the same mechanism above | 4
rsvTimeout | Whether and how long a resolver applies a timeout for each query it sends | false, 20.0
rsvOverallTimeout | Whether and how long a resolver applies an overall timeout for a client request | false, 100.0
Our framework — formal analysis

**Simulation** for semantics sanity checks, serving as reference implementation
Our framework — formal analysis

*Model checking* on *qualitative* properties
Our framework — formal analysis

*Model checking* on *qualitative* properties, e.g., RFC compliance of zone config

---

Does P hold?  
P := absence of  
lame delegation  
circular dependency  
answer inconsistency  
rewrite blackhole  
...  

*Include all properties in GRoot [Sigcomm'20]*
Our framework — formal analysis

*Model checking* on *qualitative* properties, e.g., RFC compliance of zone config

Does P hold for a given set of zone files & a given client request?
Our framework — formal analysis

*Model checking* on *qualitative* properties, e.g., RFC compliance of zone config

Non-deterministic *initial state exploration* with automation

---

**Diagram:**

- **A** → **IS 1**
- **B** → **IS 2**
- **C** → **IS 3**

***query space is huge!***
Our framework — formal analysis

*Model checking* on *qualitative* properties, e.g., RFC compliance of zone config

Non-deterministic *initial state exploration* with automation

Query equivalence class (EC)

Sample query from each EC

Caveat: definition of EC is critical
  - Use GRoot’s EC as a *heuristic*
Our framework — formal analysis

**Model checking** on **qualitative** properties, e.g., RFC compliance of zone config

Does P hold for a given set of zone files & all queries up to EC?
Our framework — formal analysis

**Statistical verification** on **quantitative** properties

- Initial State Generator
- Executable Semantics

**E.g., PVeStA SMC**
Our framework — formal analysis

**Statistical verification** on **quantitative** properties, e.g., query success ratio

What is the *probability* that P holds with a *given statistical confidence*?

- **Initial State Generator**
- **Executable Semantics**

Statistical Verifier

0.83
Our framework — formal analysis

*Statistical verification* on *quantitative* properties, e.g., query success ratio

Example: Under NXNS attack [SEC’20], with 0.05 error margin and 95% statistical confidence, the query success ratio of a legitimate client is

0.71
Our framework — formal analysis

*Statistical verification* on *quantitative* properties, e.g., query success ratio

Example: Under NXNS attack [SEC’20], with 0.05 error margin and 95% statistical confidence, the query success ratio of a legitimate client is

![Diagram showing double attack intensity, MaxFetch(1) mitigation, and MaxFetch(5) mitigation with corresponding query success ratios of 0.71, 0.52, 0.91, and 0.86.](image-url)
Application: automated analysis of DoS vulnerabilities

Excessive queries triggered by a single client request: high *amplification* factor (AF)

Randomised and Tunable Zone Configuration Generation

Initial State Generator  Executable Semantics

Simulator

AF < threshold

AF >= threshold

Manual investigation
Application: automated analysis of DoS vulnerabilities

Excessive queries triggered by a single client request: high \textit{amplification} factor (AF)

Re-discovered major known vulnerabilities [DNS-OARC’15, RAID’18, SEC’20, IMC’21]

New vulnerabilities

\begin{itemize}
  \item Exploit interaction btwn features
  \item 100s of MAF
  \item Validated in DNS software
  \item Reported, investigation WIP
\end{itemize}

See paper for detail!
Summary and outlook

Our framework establishes a formal foundation for DNS

- **Comprehensive** semantics
- **Versatile** in verification (quantitative property 1st time)
- **Automated** toolset
Summary and outlook

Our framework establishes a formal foundation for DNS

- **Comprehensive** semantics
- **Versatile** in verification (quantitative property 1st time)
- **Automated** toolset

Future work

- More DNS features, e.g., DNSSEC, DoT/DoH
- Richer property library, better automation
- Sound and complete definitions of EC