DNS ✓

Automated Verification of an In-Production DNS Authoritative Engine

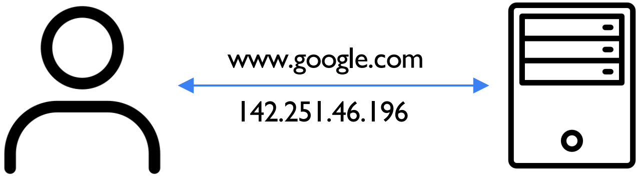
Naiqian Zheng[1*], Mengqi Liu[2*], Yuxing Xiang[1], Linjian Song[2], Dong Li[2], Feng Han[2], Nan Wang[2], Yong Ma[2], Zhuo Liang[2], Dennis Cai[2], Ennan Zhai[2], Xuanzhe Liu[1], Xin Jin[1]

1 北京大學 PEKING UNIVERSITY

2 Alibaba Cloud

Domain Name System is essential

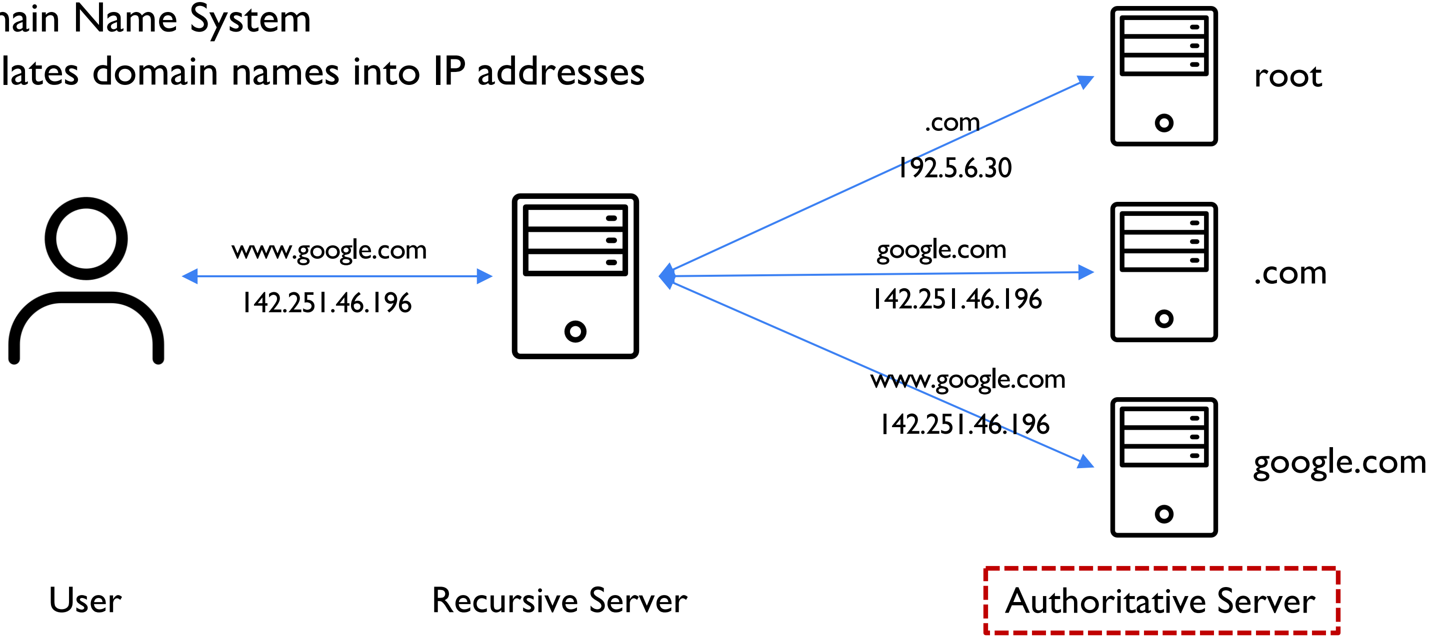
DNS: Domain Name System
DNS translates domain names into IP addresses



Domain Name System is essential

DNS✓

DNS: Domain Name System
DNS translates domain names into IP addresses



DNS software is complex

DNS✓

```
; <<>> DiG 9.10.6 <<>> google.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 8085
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 4, ADDITIONAL: 9

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
;; QUESTION SECTION:
;google.com.                    IN      A

;; ANSWER SECTION:
google.com.             254     IN      A       142.251.43.14

;; AUTHORITY SECTION:
google.com.             28492   IN      NS      ns1.google.com.
google.com.             28492   IN      NS      ns2.google.com.
google.com.             28492   IN      NS      ns3.google.com.
google.com.             28492   IN      NS      ns4.google.com.

;; ADDITIONAL SECTION:
ns1.google.com.         110729  IN      A       216.239.32.10
ns2.google.com.         110729  IN      A       216.239.34.10
ns4.google.com.         110729  IN      A       216.239.38.10
ns3.google.com.         110729  IN      A       216.239.36.10
ns1.google.com.         110729  IN      AAAA    2001:4860:4802:32::a
ns2.google.com.         110729  IN      AAAA    2001:4860:4802:34::a
ns4.google.com.         110729  IN      AAAA    2001:4860:4802:38::a
ns3.google.com.         110729  IN      AAAA    2001:4860:4802:36::a
```

Specification details:

- RFC 1034, 1035, 2136, 2181, 4592, etc.
- DNS Answer:

  status + flags + answer + authority + additional section + …

DNS software is complex

DNS✓

```
; <<>> DiG 9.10.6 <<>> google.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 8085
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 4, ADDITIONAL: 9

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
;; QUESTION SECTION:
;google.com.                    IN      A

;; ANSWER SECTION:
google.com.             254     IN      A       142.251.43.14

;; AUTHORITY SECTION:
google.com.             28492   IN      NS      ns1.google.com.
google.com.             28492   IN      NS      ns2.google.com.
google.com.             28492   IN      NS      ns3.google.com.
google.com.             28492   IN      NS      ns4.google.com.

;; ADDITIONAL SECTION:
ns1.google.com.         110729  IN      A       216.239.32.10
ns2.google.com.         110729  IN      A       216.239.34.10
ns4.google.com.         110729  IN      A       216.239.38.10
ns3.google.com.         110729  IN      A       216.239.36.10
ns1.google.com.         110729  IN      AAAA    2001:4860:4802:32::a
ns2.google.com.         110729  IN      AAAA    2001:4860:4802:34::a
ns4.google.com.         110729  IN      AAAA    2001:4860:4802:38::a
ns3.google.com.         110729  IN      AAAA    2001:4860:4802:36::a
```

Specification details:

- RFC 1034, 1035, 2136, 2181, 4592, etc.
- DNS Answer:

  status + flags + answer + authority + additional section + …

Implementation complexity:

- Bind9 (>50k LOC),

  Alibaba Cloud DNS (>100k LOC)
- Frontend server, authentication, cache, …

PEKING UNIVERSITY 北京大學

Alibaba Cloud

SOSP '23

DNS failures lead to network outages

DNS ✓

Security

A DNS outage just took down a large chunk of the internet

Zack Whittaker @zackwhittaker / 12:5

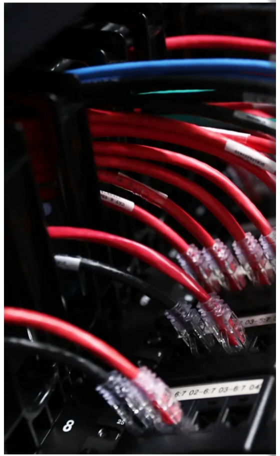


Image Credits: Joe Raedle / Gett

2021 Facebook outage

Article Talk

From Wikipedia, the free encyclopedia

On October 4, 2021, at 15:39 UTC, the social network Facebook and its subsidiaries, Messenger, Instagram, WhatsApp, Mapillary, and Oculus, became globally unavailable for a period of six to seven hours.[1][2][3] The outage also prevented anyone trying to use "Log in with Facebook" from accessing third-party sites.[4] It lasted for 7 hours and 11 minutes.

During the outage, many users flocked to Twitter, Discord, Signal, and Telegram, resulting in disruptions on these sites' servers.[9] The outage was caused by the loss of IP routes to the Facebook Domain Name System (DNS) servers, which were all self-hosted at the time.[10][5] Border Gateway Protocol (BGP) routing was restored for the affected prefixes at about 21:50, and DNS services began to be available again at 22:05 UTC, with application-layer services gradually restored to Facebook, Instagram, and WhatsApp over the following hour, with service generally restored for users by 22:50.[11]

.CLUB

On Oct. 7th, 2021, three days after Facebook's outage, the .club and .hsbc TLDs also experienced a three-hour outage. In this case, the relevant authoritative servers remained reachable, but responded with SERVFAIL messages. The effect on recursive resolvers was essentially the same: Since they did not receive useful data, they repeatedly retried their queries to the parent zone. During the incident, the Verisign-operated A-root and J-root servers observed an increase in queries for .club domain names of 45x, from 80 queries per second before, to 3,700 queries per second during the outage.

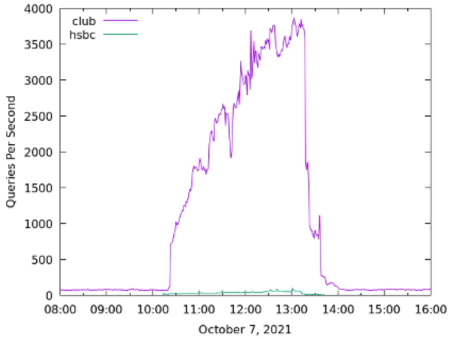


Figure 3: Rate of DNS queries to A and J root servers during the 10/7/2021 .club outage.

How to keep it reliable?

DNS ✓

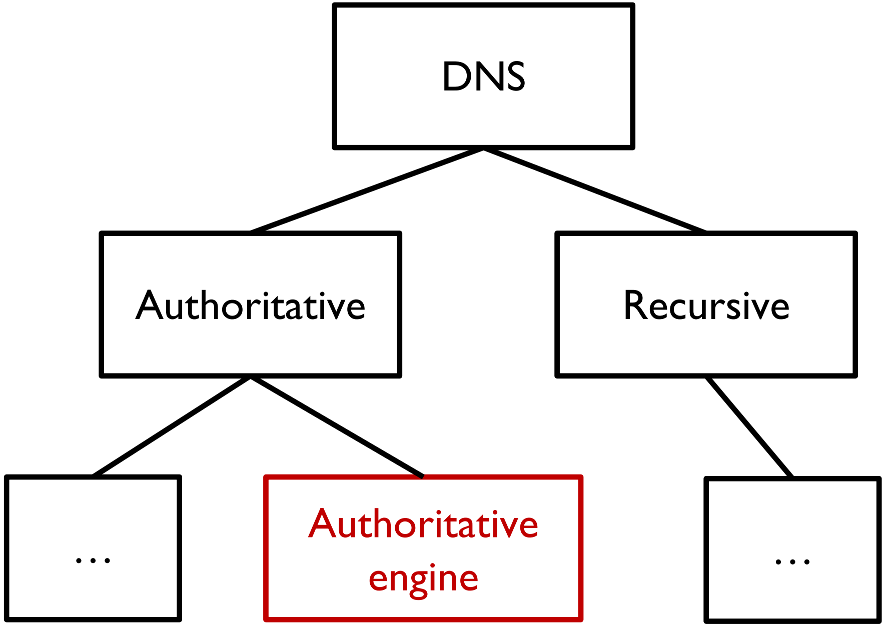
Testing	Interactive Verification	Push-button Verification
KLEE, sieve	seL4 (Security. Performance. Proof.), CertiKOS	Serval, Jitterbug
Weak correctness guarantees	Require manual proofs	Too large

Verifying the core: authoritative engine

DNS ✓

Verify Configuration:
Groot, …

Verify Implementation:
DNS-V

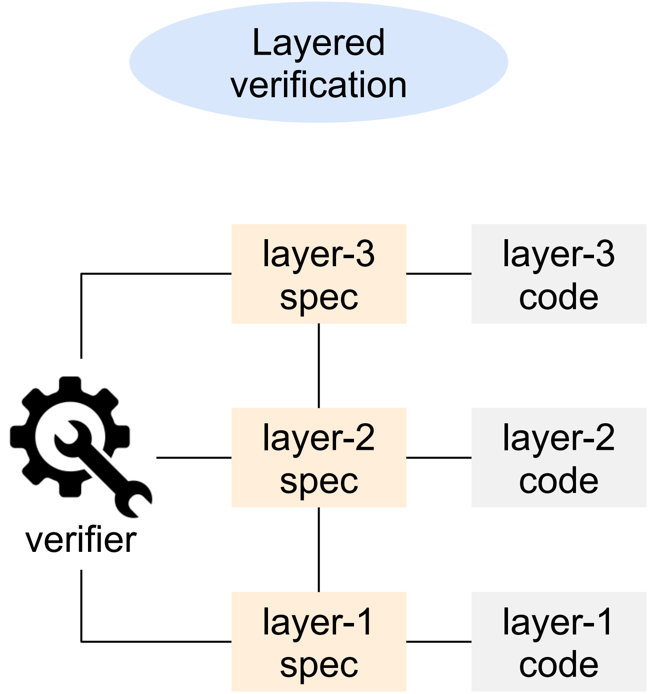


How?

Challenges of applying push-button verification

DNS

- **Large scale**
  2,000+ LOC of Go code, 50+ functions
  Path explosion, complicated encoding strategies



PEKING UNIVERSITY 北京大學

Alibaba Cloud

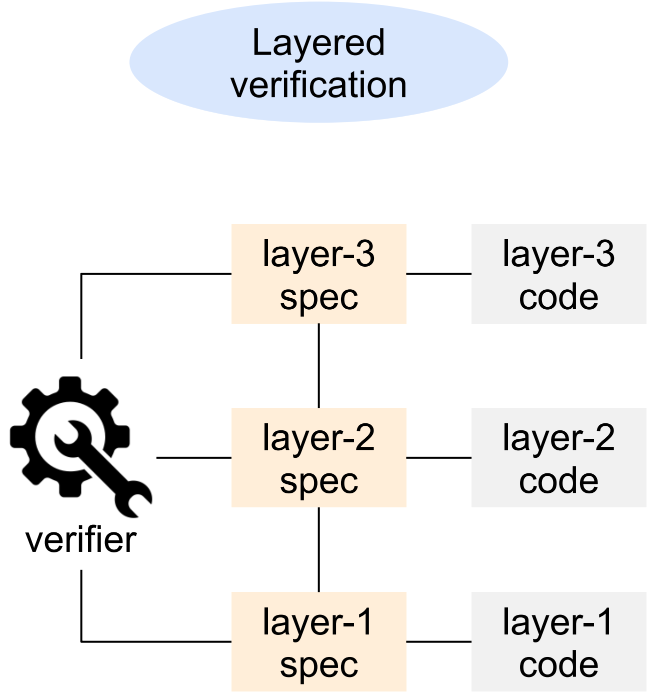
SOSP '23

Challenges of applying push-button verification

DNS ✓

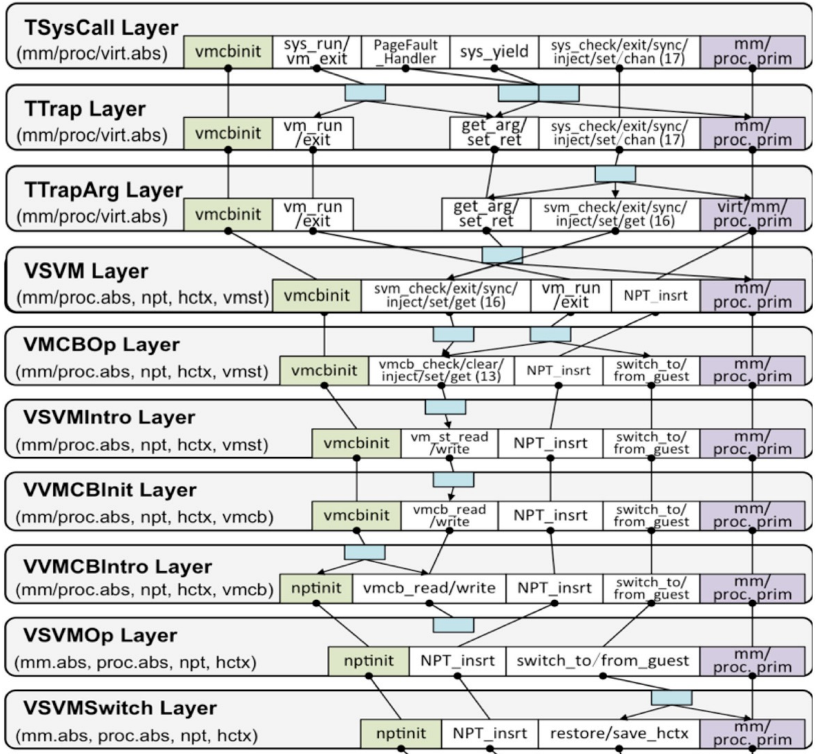
- **Large scale**
  2,000+ LOC of Go code, 50+ functions
  Path explosion, complicated encoding strategies

- **Non-verification-friendly implementation**
  difficult to develop and maintain
  correct specifications for layers



Challenges of applying push-button verification

DNS✓



- Non-verification-friendly implementation
  difficult to develop and maintain correct specifications for layers

1. Unclean interface & function division

CertiKOS:
clean interface

https://www.cs.columbia.edu/~rgu/RonghuiGu_files/certikos_layer.jpg

北京大學 PEKING UNIVERSITY   Alibaba Cloud   SOSP '23

Challenges of applying push-button verification

DNS✓

```
func TreeSearch(domain Name, flag int)
                   (TreeNode, RetFlag){
    if is_relevant(domain) {
      // domain in zone file
    } else {
      // not relevant
    }

    // dispatch flags
    switch flag {
      // find wildcard? FQDN? NS? A?
    }

    // ...
}
```

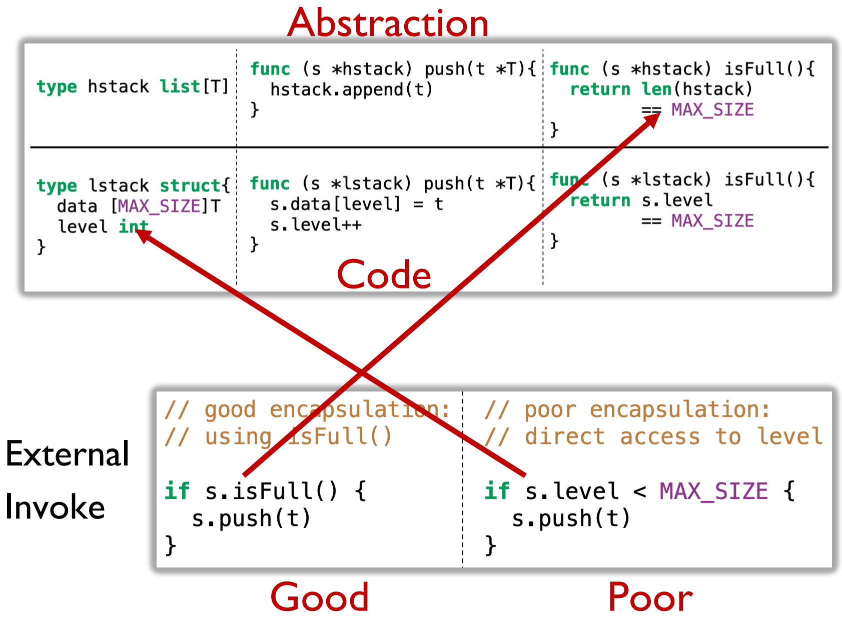
- Non-verification-friendly implementation
  difficult to develop and maintain correct specifications for layers

1. Unclean interface & function division

In-production:
unclean interface

Challenges of applying push-button verification

DNS✓



- Non-verification-friendly implementation
  difficult to develop and maintain correct specifications for layers

1. Unclean interface & function division

2. Poor data structure encapsulation

PEKING UNIVERSITY 北京大學 Alibaba Cloud SOSP '23

Challenges of applying push-button verification

DNS ✓

```go
func compareRaw(n1 *RawName, n2 *RawName) int {
  l1 := len(n1.offsets) - 1
  l2 := len(n2.offsets) - 1
  lcount := 0
  for l1 >= 0 && l2 >= 0 {
    p1 := n1.offsets[l1]
    p2 := n2.offsets[l2]
    for n1.data[p1] != '.' && n2.data[p2] != '.' {
      if n1.data[p1] == n2.data[p2] {
        p1++
        p2++
      } else {
        break
      }
    }
    if n1.data[p1] != '.' || n2.data[p2] != '.' {
      if lcount > 0 {
        return PARTIALMATCH
      } else {
        return NOMATCH
      }
    }
    l1
    l2
  }
  if l
    re
  } el
    re
  }
}
```

Intentionally choosing raw bytes (instead of high-level language constructs) makes it more complex.

```go
type RawName struct {
  // e.g. byte array for "www.example.com."
  data        []byte
  // starting offset for each label.
  // e.g., [0, 4, 12]
  offsets     []int
}
```

- Non-verification-friendly implementation
  difficult to develop and maintain correct specifications for layers

1. Unclean interface & function division
2. Poor data structure encapsulation
3. Complex low-level implementation

PEKING UNIVERSITY 北京大學

Alibaba Cloud

SOSP '23

Challenges of applying push-button verification

DNS ✓

- Verification can follow the rapid pace of software iteration.

DNS ✓

- Non-verification-friendly implementation

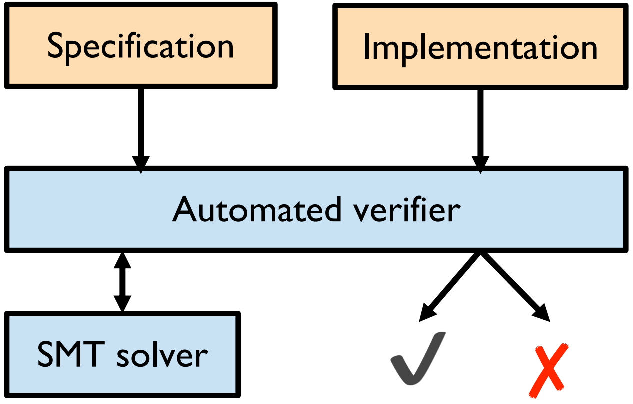
difficult to develop and maintain correct specifications for layers

1. Unclean interface & function division
2. Poor data structure encapsulation
3. Complex low-level implementation

Automated refinement with code summary

DNS✓

Challenge 1: Unclean interface & function division
Hard to maintain correct specification



Basic refinement

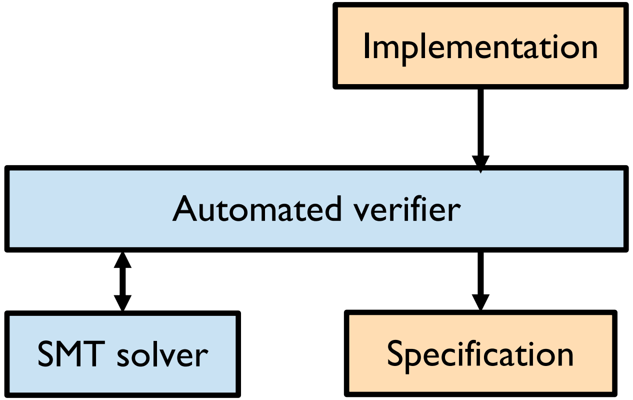
Automated refinement with code summary

DNS✓

Challenge 1: Unclean interface & function division

Hard to maintain correct specification

- Symbolic execution, accumulate path conditions and effects
- Represent behavior in abstract summary specification



Specification summarization

Hard to maintain correct specifications?

Let the verifier help you!

PEKING UNIVERSITY

Alibaba Cloud

SOSP '23

Automated refinement with code summary

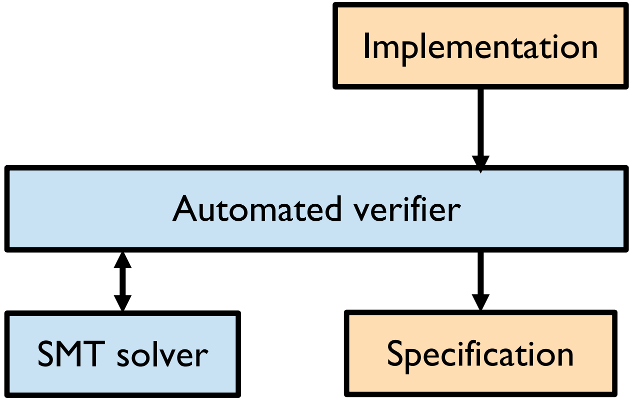
DNS ✓

Challenge 1: Unclean interface & function division

Hard to maintain correct specification

```
Func match(NodePtr, nameLen, n0, n1, …) {     Input
    if nameLen == 0 {
        NodePtr = NODE(".");
        return WILDCARD;                      Effect
    } else {
        if n0 == int("com") {
            NodePtr = NODE("com.");  Effect
            return EXACT;
        } else {
            NodePtr = NULL_NODE;
            return NOMATCH;   Effect
        }
    }
}
```

Go Code



Specification summarization

Automated refinement with code summary

DNS✓

Challenge 1: Unclean interface & function division

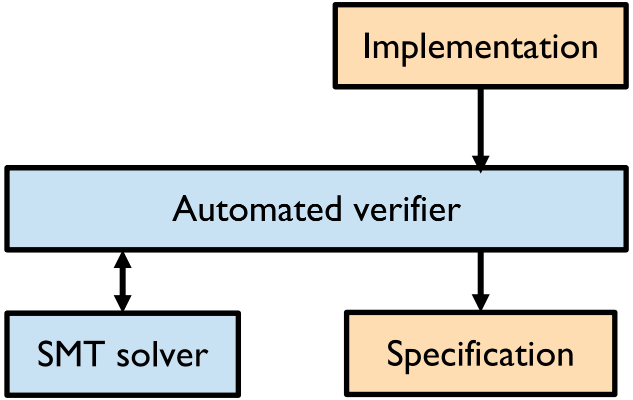
Hard to maintain correct specification

```
if (nameLen != 0 && n0 == int("com")){
    match_result := EXACT;
    match_NodePtr := NODE("com.");
}
else if (nameLen != 0 && n0 != int("com")){
    match_result := NOMATCH;
    match_NodePtr := NULL_NODE;
}
else {
    match_result := WILDCARD;
    match_NodePtr := NODE(".");
}
```

Path Condition

Effect

Specification



Specification summarization

PEKING UNIVERSITY 北京大學

Alibaba Cloud

SOSP '23

Automated refinement with code summary

Challenge 1: Unclean interface & function division

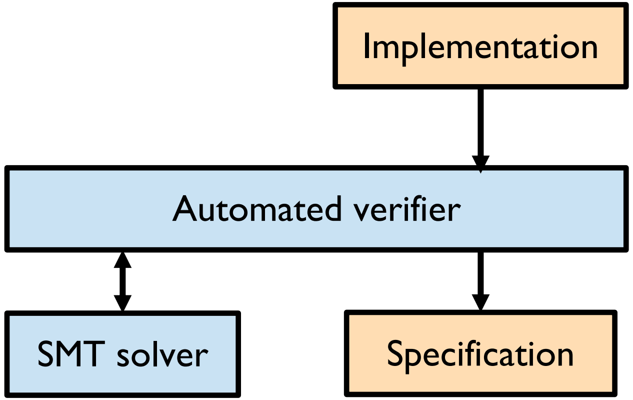
Hard to maintain correct specification

How to get input-effect pairs?

- Stateless ->

  Infer inputs from function arguments.
- Limited effect patterns ->

  Infer effects with patterns of returning values, allocating new structures, appending to an array.



Specification summarization

Flexible memory model for partial abstraction

DNS✓

Challenge 2: Poor data structure encapsulation

- Do not have to abstract memory when direct access occurs.
- A flexible memory model for specifications and code.
- Memory model: non-overlapping nested blocks.

  Concrete code: *p

  Abstract spec: rrset[1][idx]
- Each block contains an abstract array or struct, either concrete or abstract.

Partial abstraction is better than no abstraction!

PEKING UNIVERSITY

Alibaba Cloud

SOSP '23

Integration with manual abstractions

DNS ✓

Challenge 3: Complex low-level implementation

- Manually designed abstractions for low-level library modules.
- One-time effort (the underlying library rarely changes).
- Based on assumptions on code implementation.
- Domain specific primitives.

```
func compareRaw(n1 *RawName, n2 *RawName) int {
  l1 := len(n1.offsets) - 1
  l2 := len(n2.offsets) - 1
  lcount := 0
  for l1 >= 0 && l2 >= 0 {
    p1 := n1.offsets[l1]
    p2 := n2.offsets[l2]
    for n1.data[p1] != '.' && n2.data[p2] != '.' {
      if n1.data[p1] == n2.data[p2] {
        p1++
        p2++
      } else {
        break
      }
    }
    if n1.data[p1] != '.' || n2.data[p2] != '.' {
      if lcount > 0 {
        return PARTIALMATCH
      } else {
        return NOMATCH
      }
    }
    l1--
    l2--
  }
  if l1 == l2 {
    return EXACTMATCH
  } else {
    return PARTIALMATCH
  }
}
```

```
type RawName struct {
  // e.g. byte array for "www.example.com."
  data        []byte
  // starting offset for each label.
  // e.g., [0, 4, 12]
  offsets     []int
}
```

Manual abstraction based on memory layout of Name

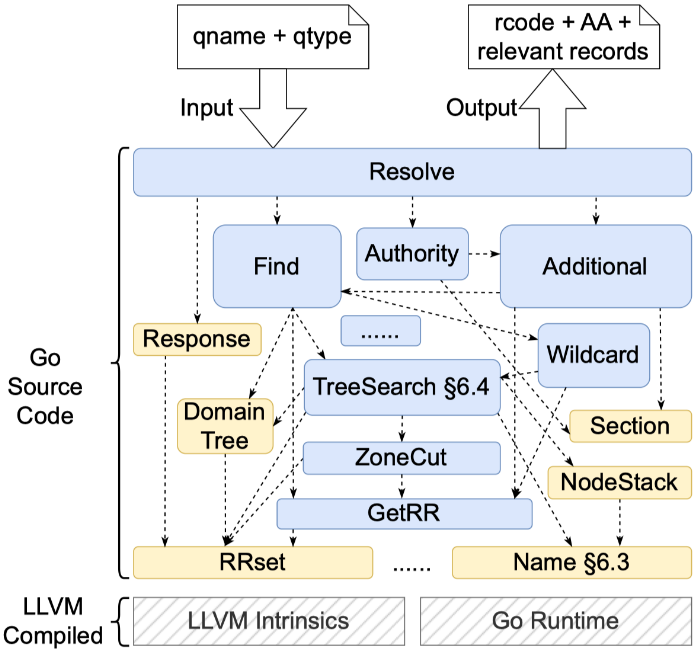
```
// e.g., [int("com"), int("example"), int("www")]
type Name List[Int]

int compareAbs(Name n1, Name n2){
  if (n1[0] != n2[0]){
    return NOMATCH
  }else{
    if (listEq(n1, n2)){
      return EXACTMATCH
    }else{
      return PARTIALMATCH
    }
  }
}
```

Too complex for the machine?
Let humans help!

Summarized specification vs. manual specification

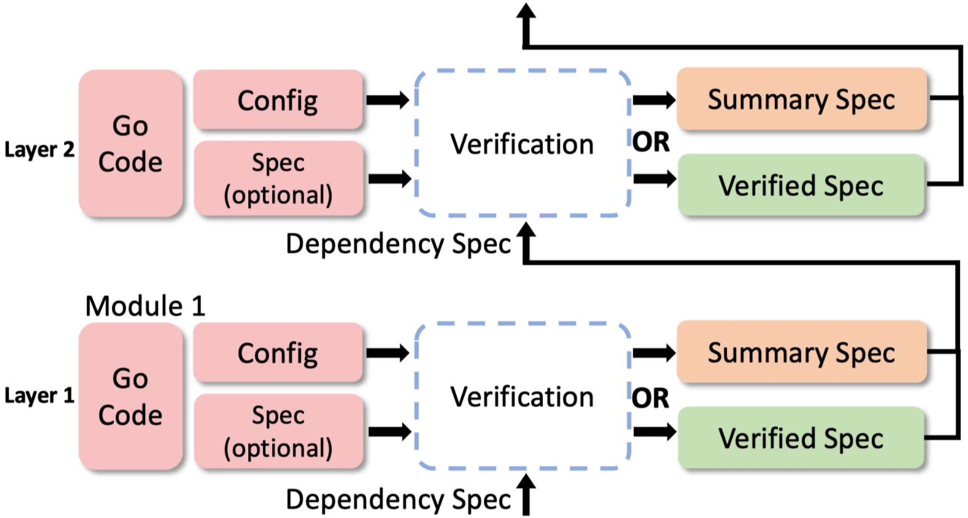
DNS ✓



- Automated refinement with code summary
  - simple formulas and relatively large-size encodings
  - complex input arguments and unclear functionality
  - e.g., DNS matching operations
- Manual specification abstraction
  - concise and highly abstracted
  - complex internal logic but clear functionality
  - e.g., domain name comparison

An overview of DNS-V

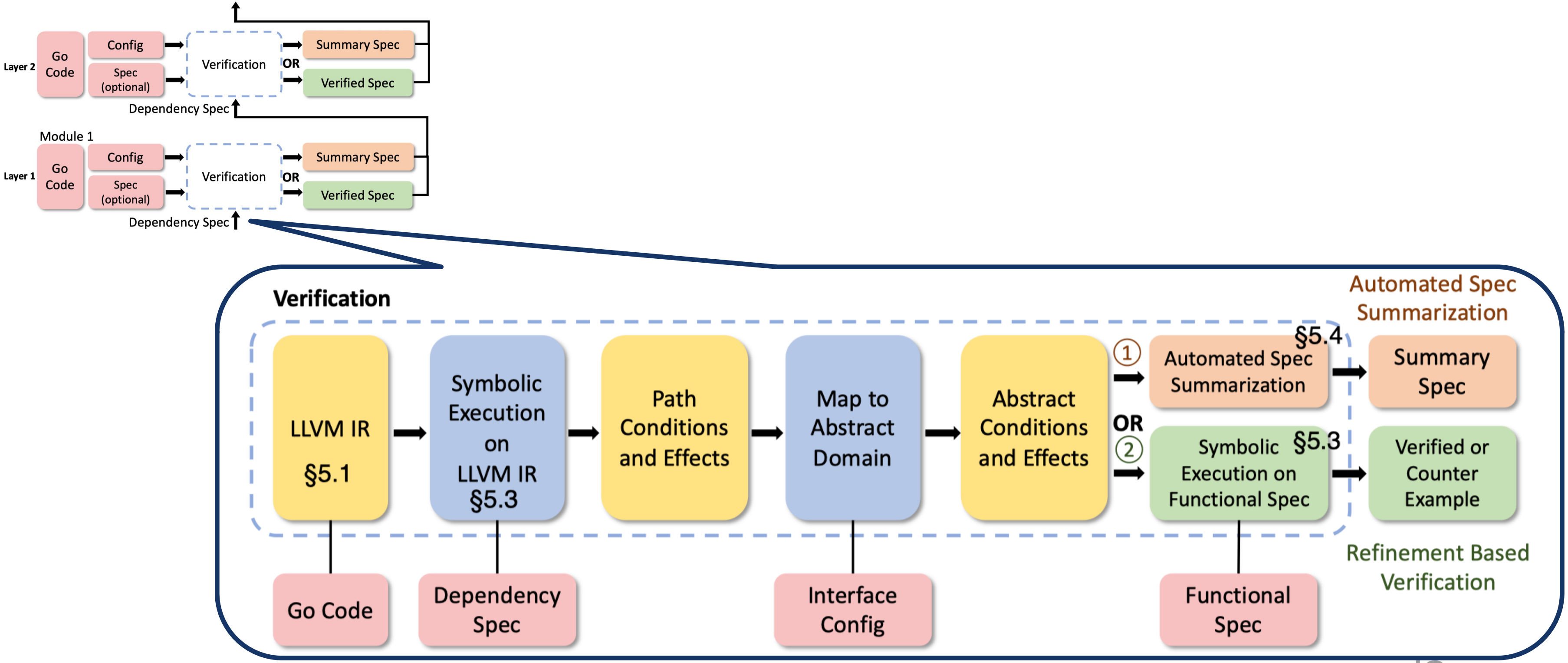
DNS✓



- Divided into layers manually.
- Input:
  code, verification config, specification
- Get a summarized specification
  or verify a manual specification

An overview of DNS-V

DNS✓



DNS-V implementation

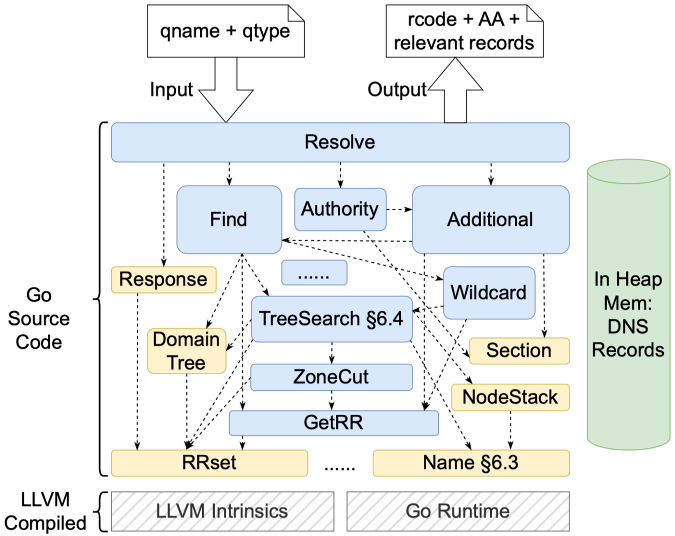
DNS✓

- Implemented in 10,000 lines of Java
- LLVM IR as frontend input (generated by GoLLVM)
- Z3 SMT Solver as backend

- Support LLVM types and syntax
- Distinguish stack memory and heap memory in memory model
- Encode `List` with variable length
- …

Refer to our paper for details

Verify an in-production DNS authoritative engine DNS✓



- Code base:
  2,000 lines of Go, stateless, no unbounded loops

- Modules:
  Matching operations: summarized spec, evolving
  Low-layer lib functions: manual spec, stable
  LLVM Intrinsics, Go Runtime: trusted computing base
  In-heap memory: from control plane, concrete

- Manual annotations:
  - assign types for Go interfaces
  - separate the code to be verified from the code base

Verify an in-production DNS authoritative engine

DNS ✓

Functional correctness:

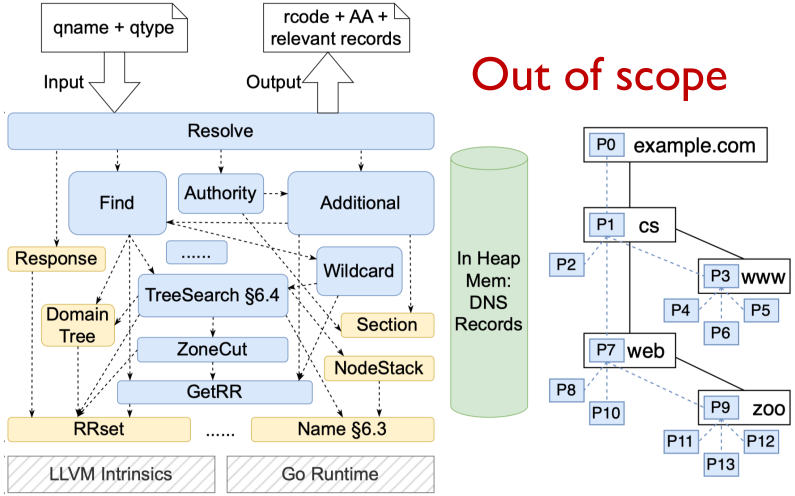
$\forall\, req,$
$spec(req) = code(req)$

Safety:

$\forall\, req,$
$\neg(code(req) \rightarrow crash)$

- The top-level specification
  A complete top-level specification that decides the authoritative response for any query
- Functional correctness
  Same as RFC standards
- Safety guarantee
  No runtime error on any input

Verify an in-production DNS authoritative engine DNS✓



We rely on the control plane to supply concrete in-heap domain trees as the runtime environment.

- Removing unbounded loops,
  making the program's behavior finite.
- Avoiding reasoning on symbolic tree data,
  simplifying the verification,
  especially for specification summarization.
- Thousands of zone config by heuristics.

PEKING UNIVERSITY

Alibaba Cloud

SOSP '23

Verify an in-production DNS authoritative engine

DNS✓

Index	Codebase Version	Classification	Description
1	1.0	Wrong Flag	AA flag missing for certain authoritative answers
2	1.0	Wrong Authority	Extraneous NS/SOA authority
3	1.0	Wrong Answer	Incorrect resource record matching on MX
4	2.0	Wrong Additional	Incomplete glue for certain queries
5	2.0	Wrong Additional	Incomplete glue when handling wildcard
6	2.0	Wrong Answer/rcode	Incorrect domain tree search for certain wildcard domains
7	2.0	Wrong Additional	Extraneous records in the additional section
8	3.0/dev	Wrong Answer/rcode	Incorrect judgments on certain wildcard domains
9	dev	Runtime Error	Incomplete bug fix may cause invalid memory access

- Prevented 10+ bugs in multiple versions and participated in bug fixing and software evolving.
- Some bugs can not be fixed properly in one go.
- Fixing bugs produces new bugs.

Verification should follow the pace of software evolving.

Verify an in-production DNS authoritative engine

DNS ✓

Index	Codebase Version	Classification	Description
1	1.0	Wrong Flag	AA flag missing for certain authoritative answers
2	1.0	Wrong Authority	Extraneous NS/SOA authority
3	1.0	Wrong Answer	Incorrect resource record matching on MX
4	2.0	Wrong Additional	Incomplete glue for certain queries
5	2.0	Wrong Additional	Incomplete glue when handling wildcard
6	2.0	Wrong Answer/rcode	Incorrect domain tree search for certain wildcard domains
7	2.0	Wrong Additional	Extraneous records in the additional section
8	3.0/dev	Wrong Answer/rcode	Incorrect judgments on certain wildcard domains
9	dev	Runtime Error	Incomplete bug fix may cause invalid memory access

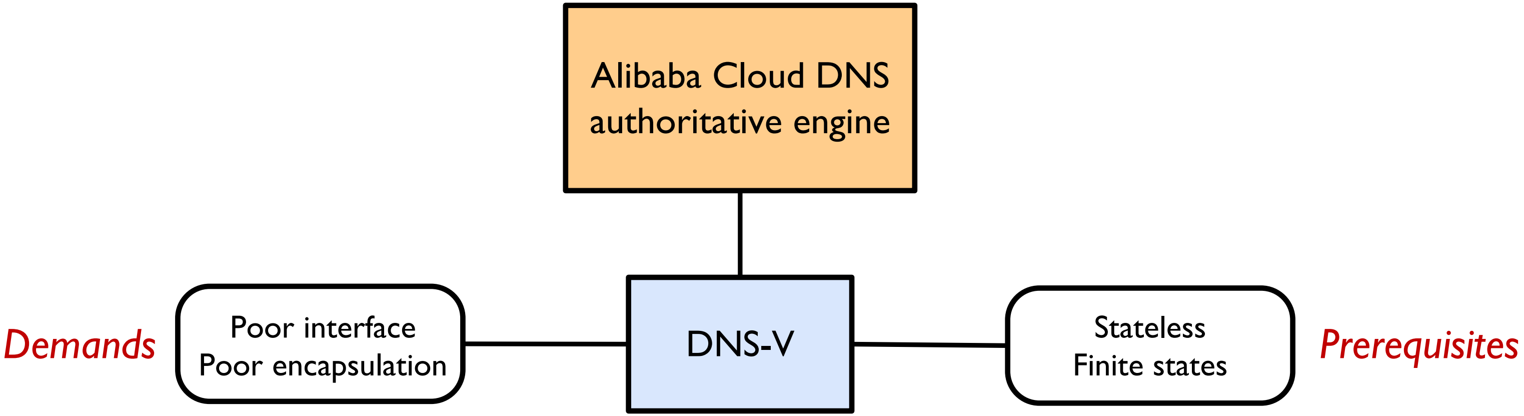
- Deployed in Alibaba Cloud DNS system for half a year.
- Verification effort: One person-week in avg.

  three person-days minimum

> Verification should follow the pace of software evolving.

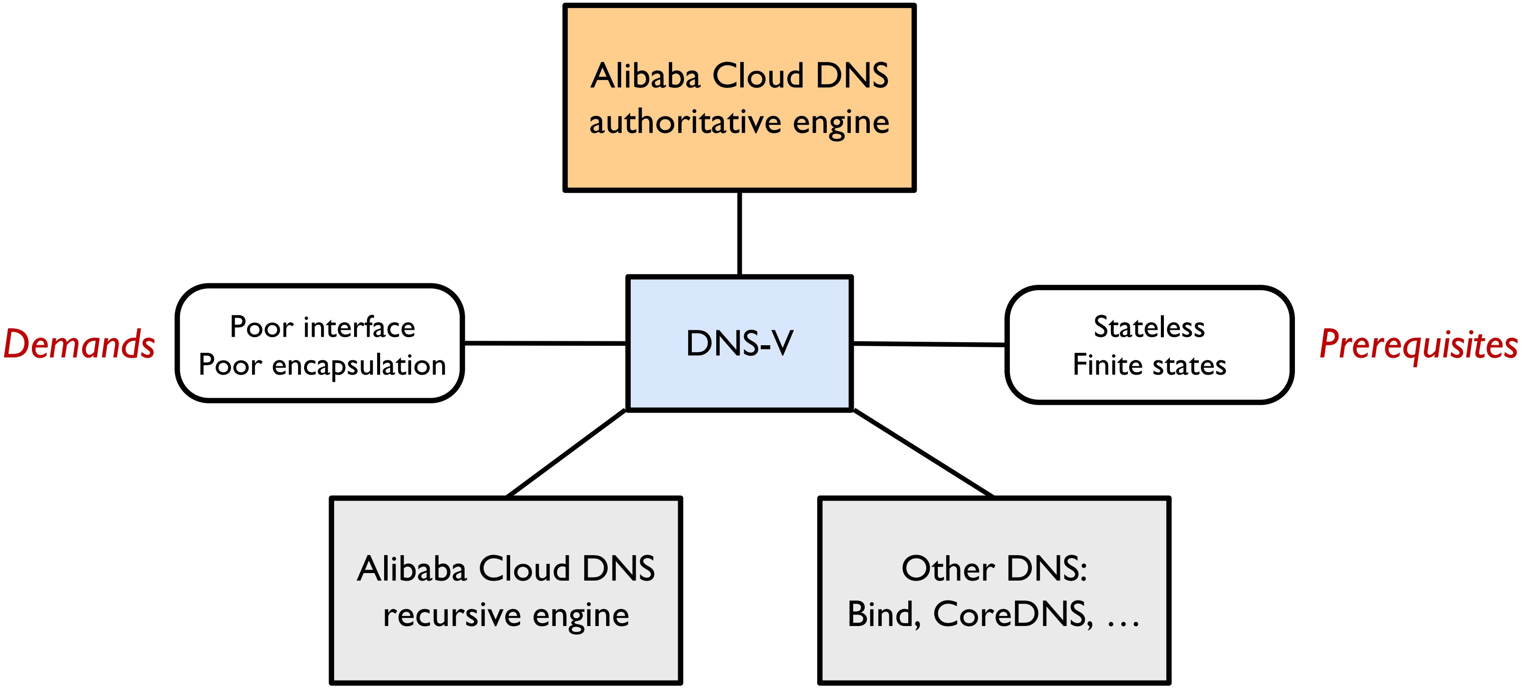
From authoritative engine to more

DNS✓



From authoritative engine to more

DNS✓



PEKING UNIVERSITY 北京大學 Alibaba Cloud SOSP '23

Take-away

DNS✓

- DNS-V is an automated verification tool for in-production DNS authoritative engines.
- DNS-V techniques

  Unclean function division --- Specification summarization

  Poor data encapsulation --- Partial abstraction memory model

  Complex lib function --- Abstract manual specification
- We verified an in-production DNS authoritative engine with DNS-V.

Thanks!

nq.zheng@pku.edu.cn
www.zhengnq.com