SH Dr. Soufiane Hourri

drsoufiane.com

DUT · SEMESTER 3 · COMPUTER SCIENCE

Network Administration

Session 03 · DNS: from names to addresses

Dr. Soufiane Hourri, PhD, HDR

Associate Professor & AI Consultant

École Supérieure de Technologie de Safi · Cadi Ayyad University

2026-2027

SESSION 03 · OVERVIEW

Today's session

01 Why names

02 The DNS tree

03 How a name is resolved

04 DNS records and zone files

05 A DNS server on Fedora with BIND

TODAY'S FOCUS

Since Session 02, client1 is told to use 192.168.50.2 as its DNS server. That server does not exist yet.

Today you learn how DNS works, then we build that server.

Learning objectives

1. Explain why DNS exists and read a domain name from right to left
2. Describe the roles of the stub resolver, the recursive resolver and the authoritative server
3. Explain caching and TTL, and list the common record types
4. Write a zone file, run a BIND server on Fedora and test it with dig

PREREQUISITES

- Session 02: DHCP gives clients a DNS server address
- Session 01: nano, systemctl, firewall-cmd
- CCNA 1: UDP and TCP ports, client and server

MODULE OVERVIEW

Where we are in the module

01
Linux server
Done: srv1 with a fixed address

02
DHCP
Done: client1 receives its settings

03
DNS
Today: names resolve to addresses

04
Web server
A site published on the lab network

05
Full network
DHCP, DNS and web working together

Where we stopped in Session 02

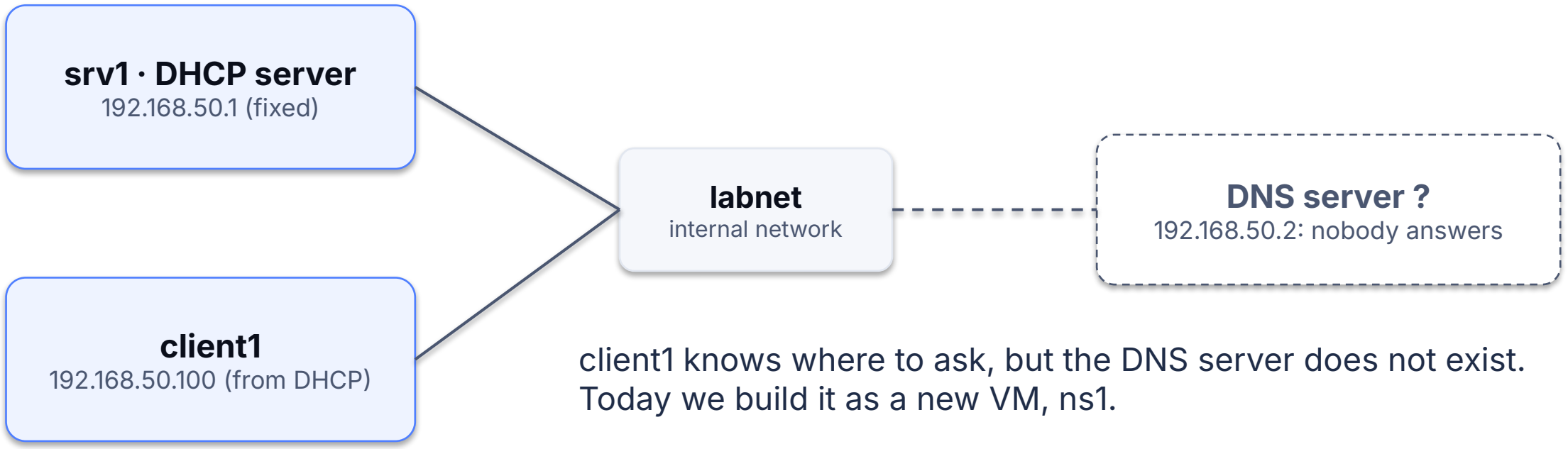


Figure 1. The lab network labnet, 192.168.50.0/24. Legend: blue = working, grey dashed = announced by DHCP but not built yet.

01

CHAPTER

Why names

Computers use addresses, people use names: how the two are connected

Names for people, addresses for machines

DEFINITION

Name resolution translates a name that people can remember, such as srv1.lab.internal, into the IP address that packets need, such as 192.168.50.1.

EASIER TO REMEMBER

You remember the name of a website, not the address of its server.

STABLE WHEN THINGS CHANGE

If a server moves to a new address, only the name record changes. Users keep typing the same name.

Before DNS: one file for the whole Internet

- Until the 1980s, every host kept a copy of one file, HOSTS.TXT
  - One central organisation maintained it; every machine downloaded it regularly
- It stopped working as the network grew
  - The file became huge, copies were out of date, and two people could choose the same name
- The idea survives as /etc/hosts on Linux
  - Windows has the same file: C:\Windows\System32\drivers\etc\hosts

KEY TAKEAWAY

A single file cannot scale. DNS replaced it with a database split across many servers.

/etc/hosts: name resolution by hand

BASH · CLIENT1

```
$ cat /etc/hosts
127.0.0.1   localhost localhost.localdomain
::1         localhost localhost.localdomain

# add one line by hand
$ sudo nano /etc/hosts
192.168.50.1  srv1.lab.internal  srv1

$ ping -c 2 srv1
```

LINES 1-3

Each line: an address, then one or more names

LINE 7

Now client1 knows srv1, and only client1

LINE 9

Works, but must be repeated on every machine

What DNS is

DEFINITION

DNS (Domain Name System) is a database of names, split across thousands of servers. Each organisation manages its own part.

IN THE PROTOCOL STACK

DNS servers listen on port 53, mostly over UDP. TCP is used for large answers and zone transfers.

HISTORY

Designed by Paul Mockapetris in 1983. The current standard is RFC 1034 and RFC 1035 (1987).

Knowledge check

QUESTION

You add "192.168.50.1 srv1" to /etc/hosts on client1. Which machines can now reach srv1 by name?

A

Every machine on labnet

B

Only client1

C

client1 and srv1

D

Every machine that uses DHCP

02

CHAPTER

The DNS tree

How names are organised, who manages each level, and why we read them from right to left

Reading a name from right to left

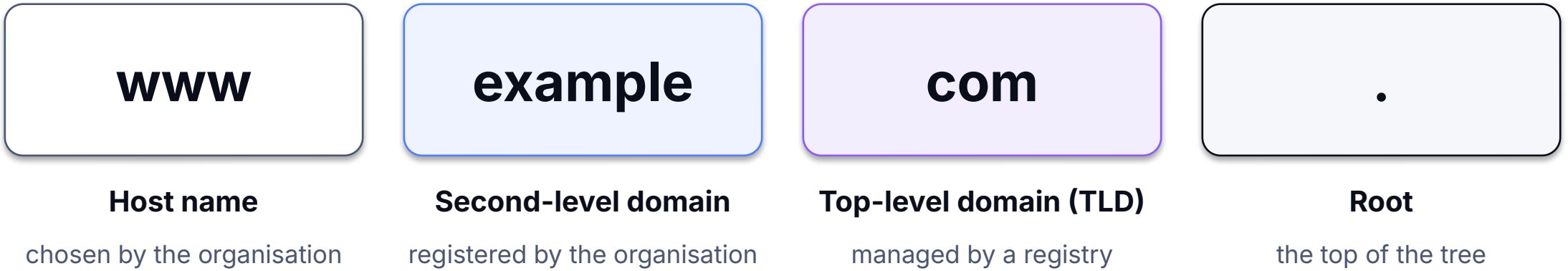


Figure 2. The parts of a fully qualified domain name (FQDN). The final dot stands for the root and is usually not typed.

The DNS tree

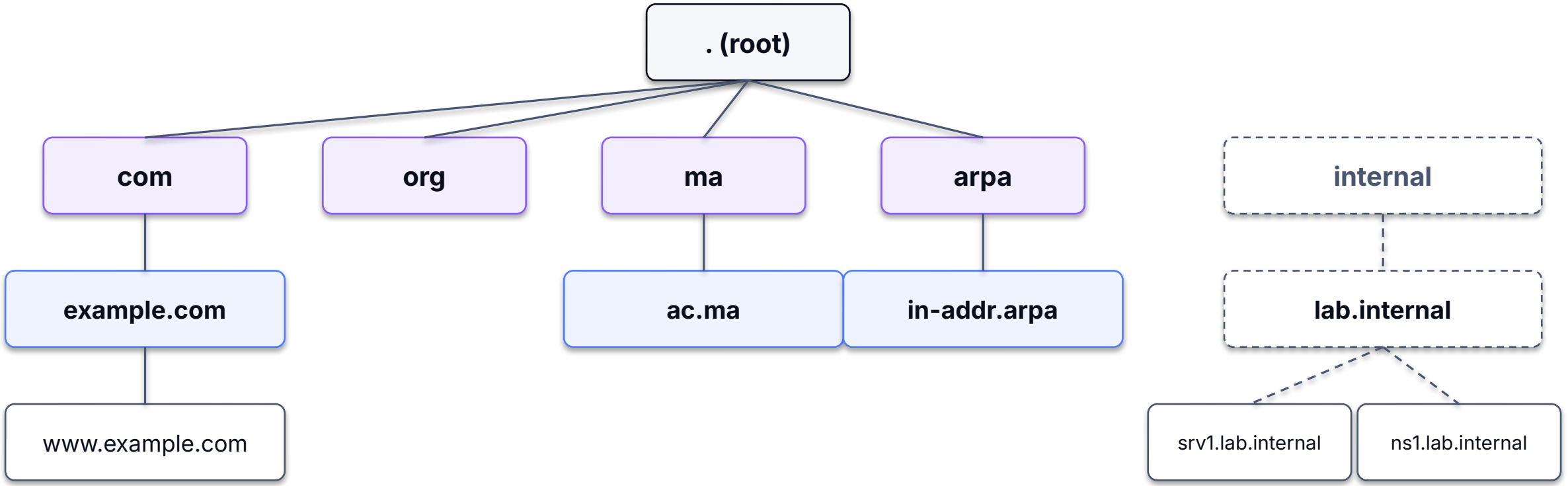


Figure 3. A small part of the DNS tree. Legend: violet = TLDs, blue = domains, dashed = .internal, reserved for private networks and never added to the root.

Kinds of top-level domains

Kind	Examples	Managed by
Generic TLD (gTLD)	.com .org .net .info	Registries under contract with ICANN
Country-code TLD (ccTLD)	.ma .fr .jp .de	One registry per country; .ma: ANRT
Special use	.arpa (reverse lookups)	Technical infrastructure (IANA)
Reserved for private use	.internal	Never delegated in the root (ICANN, 2024)

Table 1. Sources: IANA Root Zone Database (.ma: Agence Nationale de Réglementation des Télécommunications); ICANN Board resolution 2024.07.29.06 (.internal).

The root servers

DEFINITION

The root servers answer questions about the top of the tree: they know which servers are responsible for each TLD.

13 NAMES, 2,047 MACHINES

There are 13 root server identities, A to M, but each is copied in many places: 2,047 instances worldwide in October 2026, including one in Casablanca.

WHO DECIDES THE CONTENT

ICANN, through its IANA function, decides what goes into the root zone: which TLDs exist and who runs them.

Zones and delegation

DEFINITION

A zone is the part of the tree that one organisation manages. It hands lower parts to others by delegation.

EXAMPLE

The .com servers do not store www.example.com. They only say which servers manage example.com.

IN OUR LAB

ns1 will be the authoritative server for the zone lab.internal: the official source of its names.

Knowledge check

QUESTION

In the name mail.example.ma, which part is the top-level domain, and what kind is it?

A

mail, a host name

B

example, a gTLD

C

ma, a ccTLD

D

example.ma, a gTLD

03

CHAPTER

How a name is resolved

Who asks whom, in which order, and how caching makes it fast

Three roles in every DNS lookup

Stub resolver

Where: on your computer

Job: asks one question and waits for the final answer

On Fedora: systemd-resolved

Recursive resolver

Where: your network or your provider

Job: finds the answer by asking other servers, then caches it

Example: ns1 for labnet

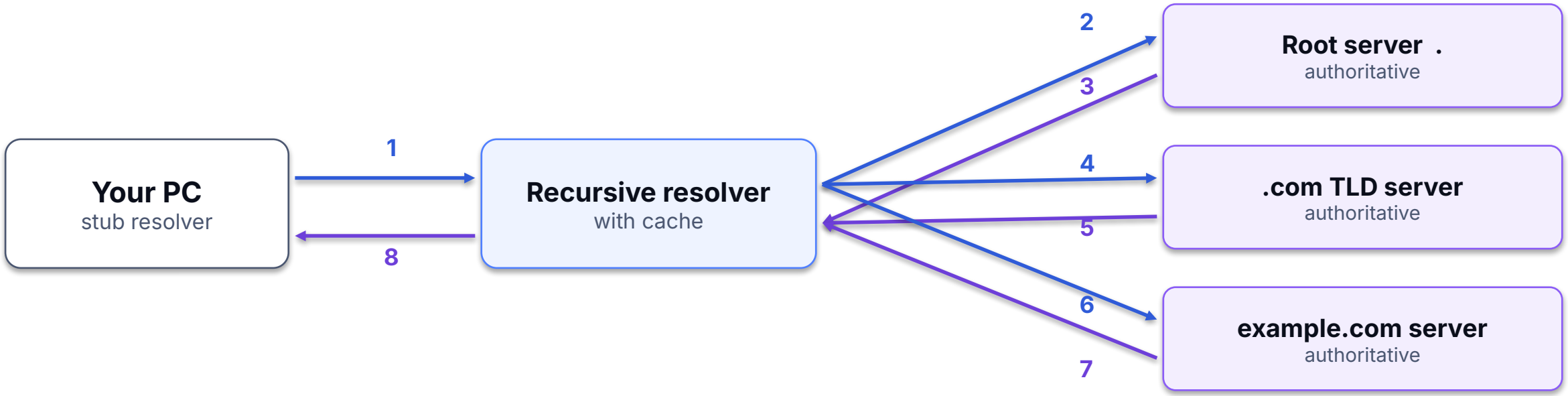
Authoritative server

Where: run by the owner of a zone

Job: gives the official answer for its zone

Examples: root, .com, ns1 for lab.internal

Resolving www.example.com with an empty cache



1 PC asks the resolver. 2-3 Root: "ask the .com servers". 4-5 .com: "ask the example.com servers".

6-7 example.com server: "www.example.com is 192.0.2.10". 8 The resolver answers the PC and keeps the answer in its cache.

Figure 4. The eight steps of a lookup when nothing is cached. 192.0.2.10 is a documentation address used as an example. Legend: blue = questions, violet = answers.

Recursive and iterative queries

Recursive query: PC to resolver

- "Give me the final answer, or an error"
- The resolver does all the work
- Used in step 1 of the lookup

Iterative query: resolver to other servers

- "Give me the best answer you have"
- The server may answer with a referral: "ask those servers"
- Used in steps 2 to 7

Caching and TTL

DEFINITION

Every DNS answer comes with a TTL (time to live) in seconds. Until it reaches zero, the answer can be reused from a cache without asking again.

EXAMPLE

srv1.lab.internal has a TTL of 3600 s. A resolver that has seen it answers directly for the next hour.

WHY IT MATTERS

Most lookups never reach the root: the answer, or the .com referral, is already cached.

Where answers are cached

Cache	Where it lives	How to see or clear it
Browser	Inside the browser process	Browser settings or restart the browser
Operating system	systemd-resolved on Fedora; DNS Client service on Windows	resolvectl statistics · resolvectl flush-caches · ipconfig /flushdns
Recursive resolver	The resolver server: your provider, or ns1 in our lab	rndc flush on a BIND server

Table 2. A query stops at the first cache that has a valid answer. The authoritative servers are asked only when every cache has missed.

CH. 03 · RESOLUTION

"I changed the record, but users still see the old address"

× INCORRECT APPROACH

Changing a record with a TTL of one day, then expecting everyone to see the new address immediately

✓ CORRECTION

A day before the change, lower the TTL to a few minutes. Make the change, then raise the TTL again.

WHY IT FAILS

Caches keep the old answer until its TTL expires. They do not know the record has changed.

Knowledge check

QUESTION

You run the same lookup twice. The second answer arrives much faster. Why?

A

The root server remembered your PC

B

The answer was still in a cache, within its TTL

C

The second query used TCP

D

The authoritative server compressed the answer

04

CHAPTER

DNS records and zone files

The kinds of information a DNS server stores, and how to write them

Common record types

Type	Meaning	Example in our zone
A	Name to IPv4 address	`srv1  A  192.168.50.1`
AAAA	Name to IPv6 address	`srv1  AAAA  2001:db8::1 (not used today)`
CNAME	Alias: this name is another name	`www  CNAME  web`
MX	Mail server of the domain, with a priority	`@  MX  10 mail (not used today)`
NS	Authoritative server of the zone	`@  NS  ns1.lab.internal.`
SOA	Start of authority: the zone's identity card	`One per zone, first record`
PTR	Address to name (reverse)	`1  PTR  srv1.lab.internal.`

Table 3. TXT records hold free text, often used to prove domain ownership or for e-mail security. 2001:db8::/32 is reserved for documentation.

Reverse lookups: from an address to a name

DEFINITION

A reverse lookup finds the name of an address. The address is written backwards under in-addr.arpa, and a PTR record gives the name.

EXAMPLE

192.168.50.1 becomes 1.50.168.192.in-addr.arpa, whose PTR record is srv1.lab.internal.

WHY BACKWARDS?

DNS names go from specific (left) to general (right). Addresses go from general (192) to specific (.1), so they are reversed.

The zone file of lab.internal

LAB.INTERNAL.ZONE

```
$TTL 3600
@  IN SOA ns1.lab.internal. admin.lab.internal. (
           2026102601 ; serial
           3600       ; refresh
           900        ; retry
           604800     ; expire
           300 )      ; negative TTL
@     IN  NS     ns1.lab.internal.
ns1   IN  A      192.168.50.2
srv1  IN  A      192.168.50.1
web   IN  A      192.168.50.3
gw    IN  A      192.168.50.254
www   IN  CNAME  web
```

LINE 1

Default TTL for every record: one hour

LINES 2-7

The SOA record (next slide)

LINE 8

@ means the zone itself: lab.internal

LINES 9-13

Short names: "srv1" means srv1.lab.internal.

The SOA record: the zone's identity card

Field	Our value	Meaning
Primary server	`ns1.lab.internal.`	The main authoritative server of the zone
Administrator e-mail	`admin.lab.internal.`	admin@lab.internal, with the @ written as a dot
Serial	`2026102601`	Version of the zone: increase it at every change (date + number)
Refresh / Retry	`3600 / 900`	How often a secondary server checks, and retries after a failure
Expire	`604800`	A secondary stops answering after one week without the primary
Negative TTL	`300`	How long caches remember that a name does not exist

Table 4. Times are in seconds. Refresh, retry and expire are used by secondary (backup) servers; we run only a primary today.

Forgetting the final dot

× INCORRECT APPROACH

```
@  IN  NS  ns1.lab.internal
```

✓ CORRECTION

```
@  IN  NS  ns1.lab.internal.
```

WHY IT FAILS

Without the dot, BIND adds the zone name. named-checkzone then reports: NS 'ns1.lab.internal.lab.internal' has no address records (A or AAAA). The zone is not loaded.

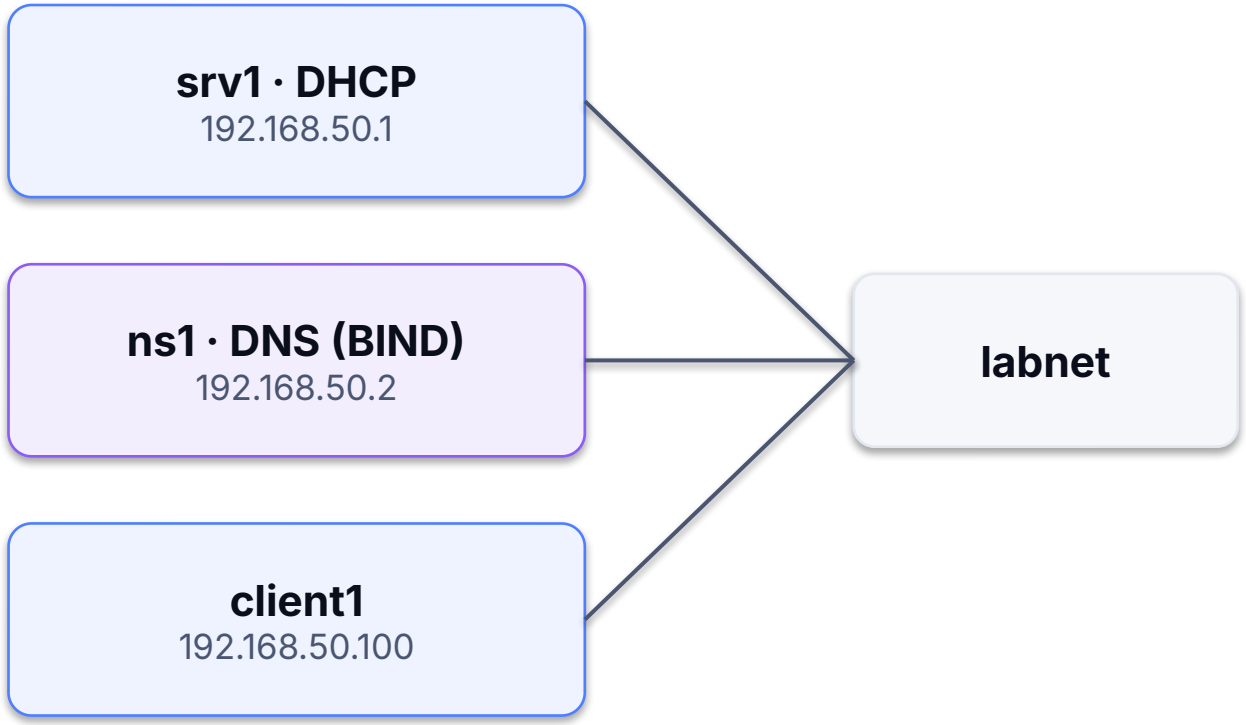
05

CHAPTER

A DNS server on Fedora with BIND

Configuring, checking, starting and testing ns1, then what to do when it fails

The lab with ns1



STEP 1

client1 asks ns1: "srv1.lab.internal?" ns1 is authoritative for lab.internal and answers 192.168.50.1.

STEP 2

client1 asks ns1 about an Internet name. ns1 is not authoritative for it, so it acts as recursive resolver, through its NAT card.

Figure 5. ns1 joins labnet at 192.168.50.2, the address that srv1's DHCP server already announces. Legend: blue = existing, violet = built today.

Five steps to a working DNS server

01

Install

sudo dnf install bind bind-utils

02

Configure

Edit /etc/named.conf: listen, allow, zones

03

Write zones

/var/named/lab.internal.zone and the reverse zone

04

Check

named-checkconf and named-checkzone

05

Open and start

Firewall dns, then systemctl enable --now named

named.conf, part 1: who may ask

NAMED.CONF · OPTIONS

```
options {
    listen-on port 53 { 127.0.0.1;
                        192.168.50.2; };
    directory  "/var/named";
    allow-query { localhost;
                  192.168.50.0/24; };
    recursion yes;
    ...
};
```

LINES 2-3

Add ns1's labnet address: by default BIND listens only on itself

LINE 4

Zone files live in /var/named

LINES 5-6

Add labnet: by default only ns1 itself may ask

LINE 7

Act as resolver for other names

named.conf, part 2: our two zones

NAMED.CONF · ZONES

```
zone "lab.internal" IN {
    type primary;
    file "lab.internal.zone";
};

zone "50.168.192.in-addr.arpa" IN {
    type primary;
    file "192.168.50.rev";
};
```

LINES 1-4

The forward zone: names to addresses

LINE 2

primary = ns1 holds the original copy (older tutorials write master)

LINES 6-9

The reverse zone for 192.168.50.0/24

The reverse zone file

192.168.50.REV

```
$TTL 3600
@  IN SOA ns1.lab.internal. admin.lab.internal. (
          2026102601 3600 900 604800 300 )
@    IN  NS   ns1.lab.internal.
1    IN  PTR  srv1.lab.internal.
2    IN  PTR  ns1.lab.internal.
3    IN  PTR  web.lab.internal.
254  IN  PTR  gw.lab.internal.
```

LINES 1-4

Same TTL, SOA and NS as the forward zone, on one line

LINES 5-8

"1" means 1.50.168.192.in-addr.arpa

RIGHT SIDE

Full names, so every one ends with a dot

Check, protect, open, start

BASH · NS1

```
sudo named-checkconf
sudo named-checkzone lab.internal \
    /var/named/lab.internal.zone
sudo chown root:named /var/named/lab.internal.zone \
    /var/named/192.168.50.rev
sudo chmod 640 /var/named/lab.internal.zone \
    /var/named/192.168.50.rev

sudo firewall-cmd --permanent --add-service=dns
sudo firewall-cmd --reload
sudo systemctl enable --now named
```

LINE 1

Nothing printed: named.conf is correct

LINES 2-3

Prints "loaded serial 2026102601" and OK

LINES 4-7

named can read the files; others cannot

LINES 9-11

Open UDP and TCP 53, then start

Testing with dig: reading the answer

BASH · CLIENT1

```
$ dig @192.168.50.2 srv1.lab.internal
;; ->>HEADER<<- opcode: QUERY,
;;     status: NOERROR, id: 4014
;; flags: qr aa rd ra; QUERY: 1,
;;     ANSWER: 1, AUTHORITY: 0

;; ANSWER SECTION:
srv1.lab.internal. 3600 IN A 192.168.50.1

;; Query time: 4 msec
;; SERVER: 192.168.50.2#53(192.168.50.2) (UDP)
```

LINE 1

@ chooses which server to ask

LINES 2-3

NOERROR: the query succeeded

LINE 4

aa: authoritative answer, from the zone itself

LINE 8

Name, TTL, class, type, value

Three more real tests

BASH · CLIENT1

```
$ dig @192.168.50.2 +short www.lab.internal
web.lab.internal.
192.168.50.3

$ dig @192.168.50.2 -x 192.168.50.1 +noall +answer
1.50.168.192.in-addr.arpa. 3600 IN PTR
                           srv1.lab.internal.

$ dig @192.168.50.2 pc9.lab.internal
;; ... status: NXDOMAIN ...
```

LINES 1-3

+short: just the values. The CNAME, then the address

LINES 5-7

-x: reverse lookup. dig builds the in-addr.arpa name for you

LINES 9-10

NXDOMAIN: the name does not exist

The status in dig's answer

Status	What it means	Seen in our tests when…
`NOERROR`	The query succeeded	Asking for srv1.lab.internal
`NXDOMAIN`	The name does not exist	Asking for pc9.lab.internal
`SERVFAIL`	The server could not answer	The zone file had an error, so the zone was not loaded
`REFUSED`	The server will not answer you	allow-query listed only localhost
`(no answer)`	dig: communications error, then "no servers could be reached"	named stopped, firewall closed, or wrong listen-on

Table 5. Each case was reproduced with BIND 9.18 and these zone files. The status line is near the top of dig's output.

Troubleshooting: where to look

Symptom	First command	Typical fix
dig: no servers could be reached	systemctl status named; firewall-cmd --list-all	Start named; add service dns; fix listen-on
SERVFAIL for the zone	journalctl -u named; named-checkzone	Fix the zone file; file name; chown root:named
REFUSED	grep allow-query /etc/named.conf	Add 192.168.50.0/24 to allow-query
Old answer after an edit	dig: compare the SOA serial	Increase the serial, then systemctl reload named
dig @ns1 works, ping by name fails	resolvectl status (on the client)	Client uses another DNS server: check DHCP option 6

Table 6. When BIND refuses to load a zone, the reason is in its log: journalctl -u named shows the file name and line number.

Knowledge check

QUESTION

dig @192.168.50.2 srv1.lab.internal returns SERVFAIL. Which command gives you the reason fastest?

A

ping 192.168.50.2

B

sudo journalctl -u named

C

sudo firewall-cmd --list-all

D

ipconfig /flushdns

DNS and security

Threats

- Cache poisoning: an attacker sends fake answers to a resolver, which caches them and gives them to every user
- Open resolver: a server that answers anyone can be used to flood a victim with large answers
- Anyone on the path can read unencrypted DNS queries

Protections

- DNSSEC: zones are signed, and resolvers check the signatures
- Answer only your own network: allow-query and allow-recursion
- DNS over TLS or HTTPS encrypts queries between the client and the resolver

Recap

KEY TAKEAWAYS

- DNS is a distributed database: each zone is managed by its owner
- Read names right to left: root, TLD, domain, host
- Stub asks the recursive resolver; it asks root, TLD, authoritative
- Answers are cached for their TTL
- BIND: named.conf, zone files with final dots, check, dig

NEXT

- Practical: build ns1, write lab.internal and its reverse zone, test from client1
- Session 04 (masterclass): the web server

References

(1) Mockapetris, P. RFC 1034: Domain Names, Concepts and Facilities. IETF, 1987

(2) Mockapetris, P. RFC 1035: Domain Names, Implementation and Specification. IETF, 1987

(3) Root Server Technical Operations. root-servers.org (2,047 instances, 18 Oct 2026)

(4) IANA. Root Zone Database: .ma (ANRT). iana.org/domains/root/db

(5) ICANN Board. Resolution 2024.07.29.06: reserving .INTERNAL. icann.org

(6) ISC. BIND 9 Administrator Reference Manual, 9.18. bind9.readthedocs.io

(7) Fedora Project. Package bind (9.18.50 in Fedora 44). packages.fedoraproject.org

(8) Fedora Project. Changes/systemd-resolved (Fedora 33). fedoraproject.org/wiki

(9) Red Hat. RHEL 10: Setting up and configuring a BIND DNS server. docs.redhat.com

SH

Thank you · Questions?

Next: build ns1 and the lab.internal zone in the practical session

Dr. Soufiane Hourri, PhD, HDR

drsoufiane.com