



# DNS Basics

---

BUPT/QMUL

2019-04-08



北京邮电大学

BEIJING UNIVERSITY OF POSTS AND TELECOMMUNICATIONS

Electronic Engineering 



# Related Information

---

- Basic function of DNS
- Host entry structure in Unix
- Two system calls for DNS database retrieving
  - `gethostbyname ()`
  - `gethostbyaddr ()`



# Agenda

---

- Brief introduction to DNS
- Elements of the DNS
- DNS services
- DNS Protocols
- DNS tools

*Refer to Chapter 23 of textbook*



---

# Brief Introduction to DNS

- Basic functions of DNS
- A short history of DNS



# Basic Functions of DNS (1)

---

- Generally, applications refer to hosts/mailboxes and network resources by **ASCII strings** - such as
  - www.bupt.edu.cn
  - www.qmul.ac.uk
  - webmaster@company-a.com
- Nevertheless, the network itself only understands **binary addresses**, so some mechanism is required to convert the ASCII string to network addresses and vice versa.
- Low-level name: IP address
- High-level name: hostname



# Basic Functions of DNS (2)

---

- Translating between addresses

- **Hostname** (www.company-a.com)

- Convert between Domain Name and IP

- **IP address** (128.9.32.254)

- Convert between IP and MAC

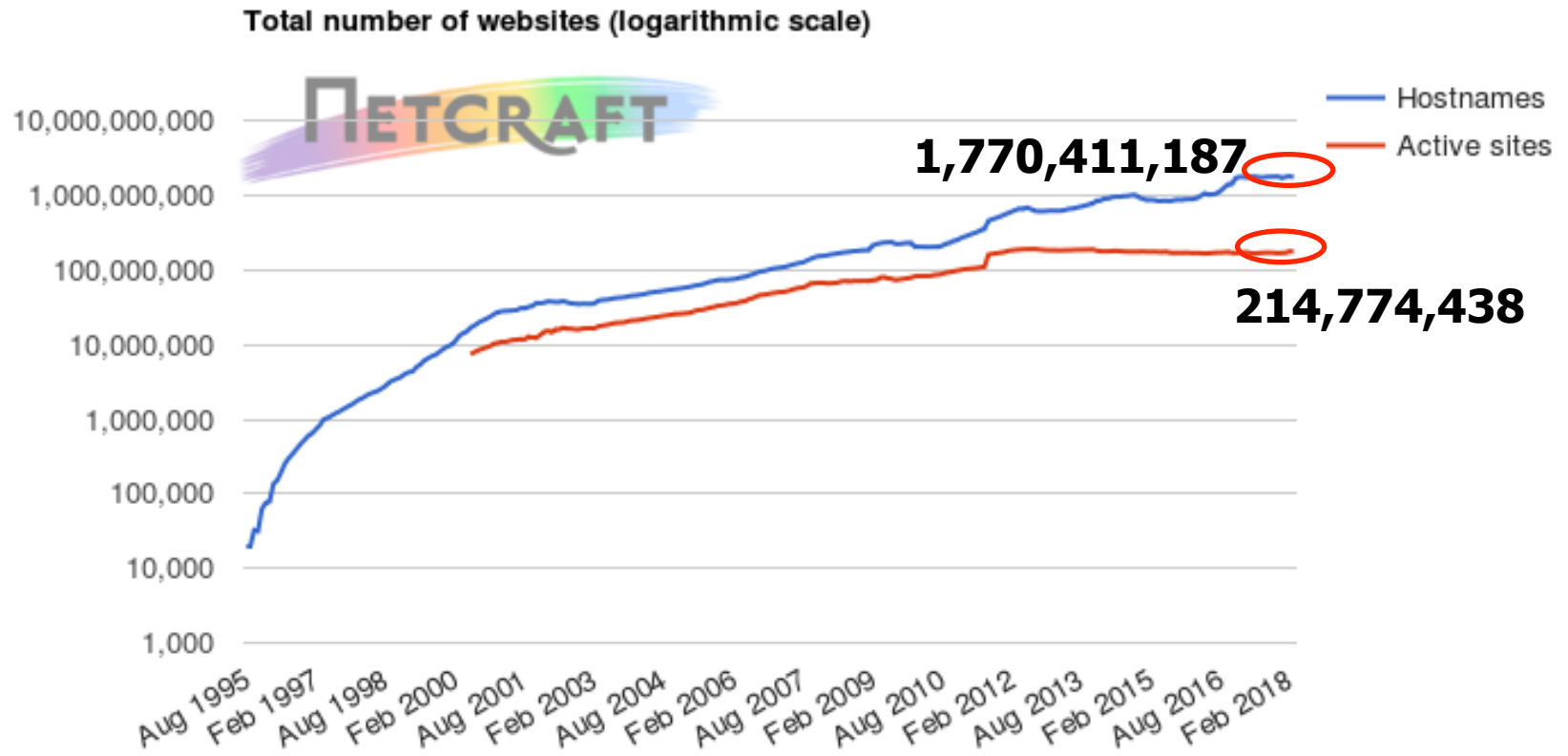
- **MAC address** (C2-67-3E-95-C5-F1)

**DNS**

**ARP**

# A Short History (1)

- Hostnames growth trend, by March 2018



(From: <http://news.netcraft.com/>)



# A Short History (2)

– from flat namespace to hierarchical namespace

---

- Original status

- There was a simple file, *hosts.txt*, that listed all the hosts and their IP addresses

***flat structure***

- Every night all the hosts would collect this file from the host that maintained it

***Centralized control***

- Incurred problems

- Each name had to be unique because the namespace was flat
- Excessive access to the machine maintaining the list
- Difficult to maintain when the network grew

***Not scalable***





## A Short History (3)

– from flat namespace to hierarchical namespace

---

- Nowaday status

***Hierarchical structure***

***Distributed database***

***Efficient, reliable, general purpose***

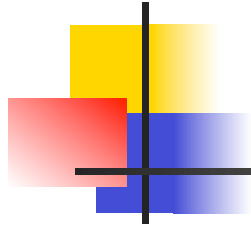
- It is primarily used for **mapping** host names and email destinations to IP addresses - but can be used for **other purposes**
- It is a **query / response** protocol running on top of **UDP/TCP**, with default port number **53**



# Summary: What is DNS?

---

- Domain Name System
- A **distributed database** providing mapping between Domain name and IP address
  - Implemented in hierarchy of many **name servers**
- An **application protocol**
  - Used by hosts and name servers to communicate to **resolve names**



# Elements Of The DNS



# Elements Of DNS

---

- Domain namespace and resource records
- Name servers
- Name resolvers
- Protocol



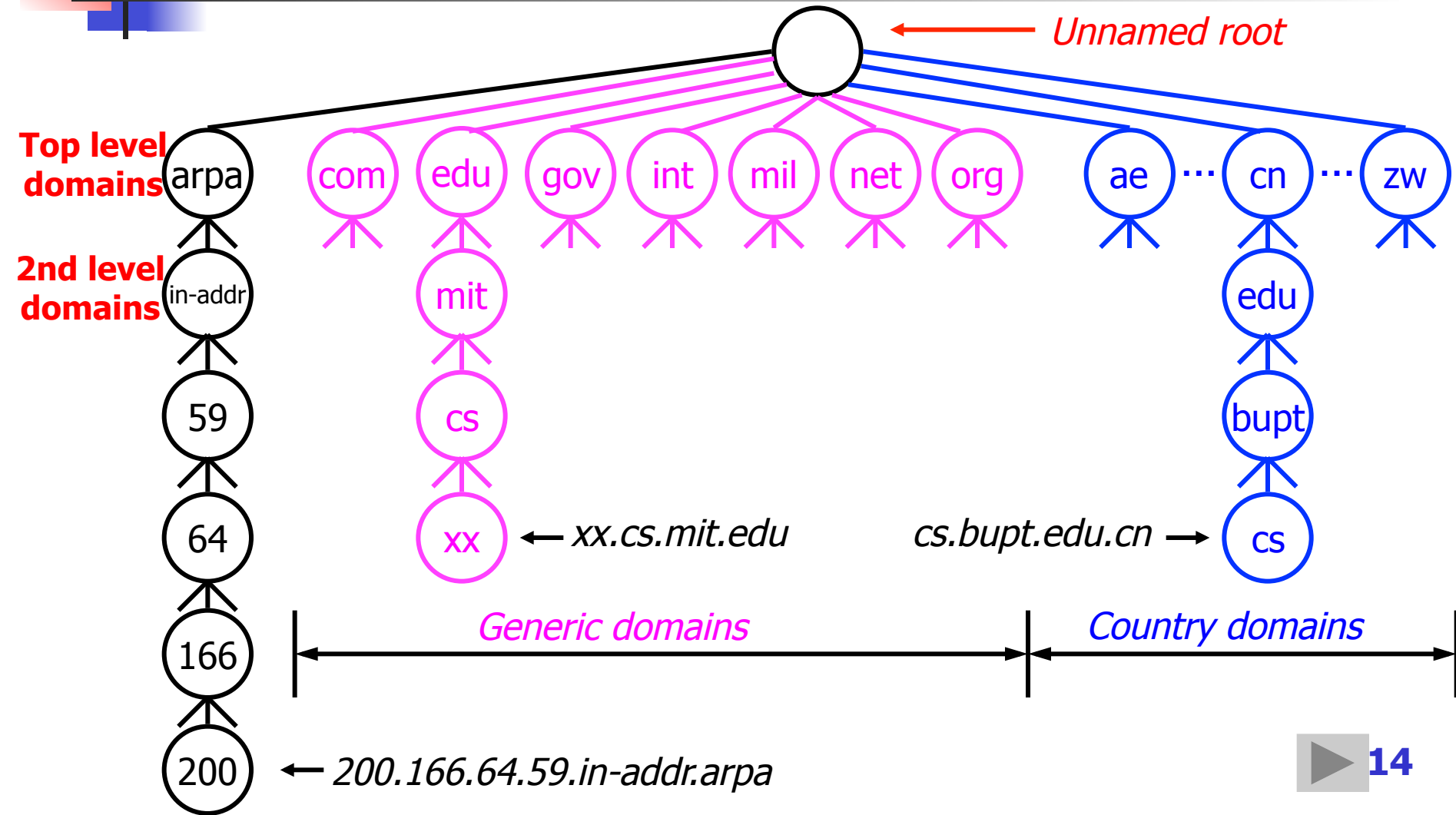
## Domain namespace and resource records

---

- **Domain namespace** is of a hierarchical structure like an upended tree
- **RR** (Resource Record) is the data associated with a particular name

# Domain Namespace (1)

– the hierarchical structure





# Domain Namespace (2)

## – organization of domains

---

- Each element of the hierarchy is referred to as a **domain**
- At the top of the hierarchy is the **root domain**, known as simply “.”
- Subdomains directly underneath the root domain are called **top-level domains**
  - gTLD: .edu/.gov/.com/.org/.net/.mil/.int
  - other gTLD: .tv/.cc/.asia/.info/.store/.web/.biz/.tel/.mobi/.  
中国
  - country level domain (ccTLD): .cn/.us/.eu/.kr/...
  - Infrastructure TLD: .arpa
  - New gTLD: under discussion
- Domains directly underneath top-level domains are called **second-level domains**, and so on



# Domain Namespace (3)

– the domain names

---

- The domain name of a node is **the list of the labels** on the path from the node to the root of the tree separated by dots(".")
- Conventionally printed or read **left to right**, from the most specific (lowest, farthest from the root) to the least specific (highest, closest to the root)
  - eg. www.bupt.edu.cn
- **Case insensitive**
- Components can be up to **63** characters long, the full pathname must not be more than **255** characters
- The full name of a domain is also called its **Fully Qualified Domain Name (FQDN)**

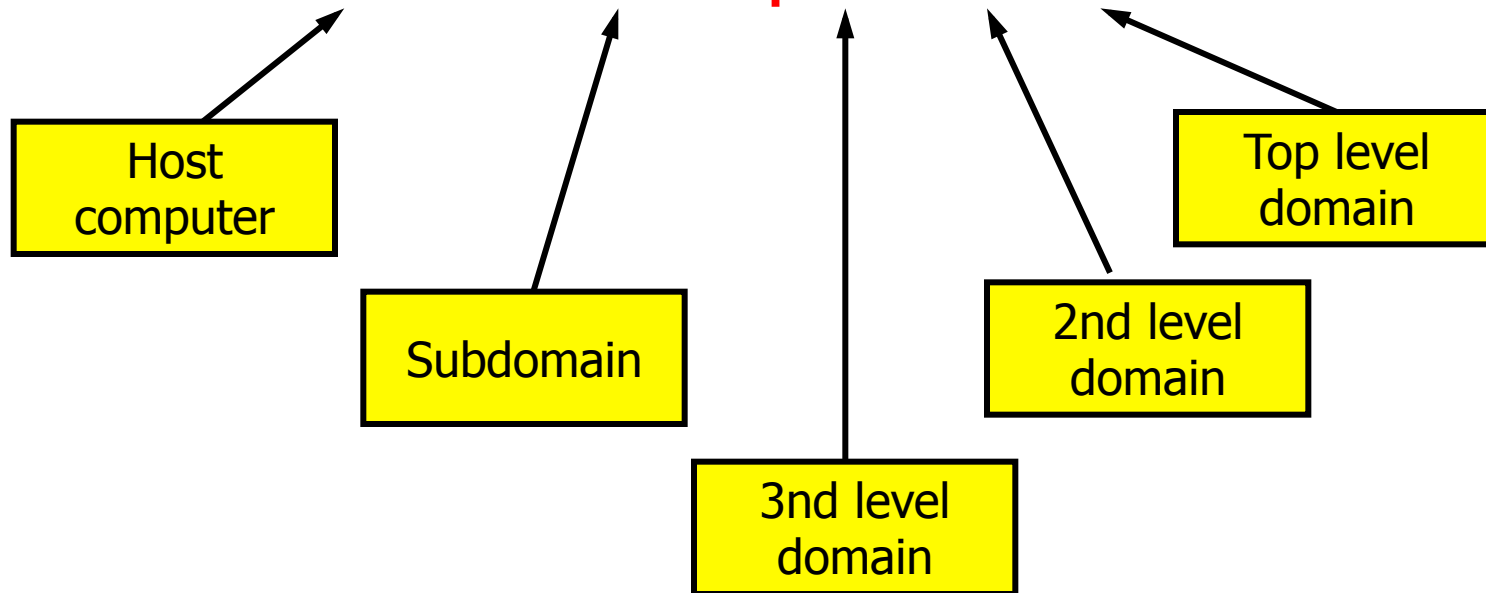


# Domain Namespace (4)

– the domain names

- Example of domain name

chimera.eecs.qmul.ac.uk





# Resource Record

---

- Each domain in the DNS has one or more **Resource Records** (RRs), which are fields that contain information about that domain
- Each RR has the following information
  - **Owner**: the domain name where the RR is found
  - **Type**: specifies the type of the resource in this RR
    - A – Host Address
    - MX – Mail Exchanger
    - NS – Name Server
    - CNAME – Canonical Name
    - ...
  - **Class**: specifies the protocol family to use
    - IN – the Internet system
  - **TTL**: specifies the Time To Live (in unit of second) of the cached RRs
  - **RDATA**: the resource data

(name, type, class, TTL, RDATA )

## Example of Resource Records(1)

- Type=A

- Name= Domain name , Value= IP Address

*ns.bupt.edu.cn* **A** *IN* 86400 202.112.10.37

- Type=NS

- Name= Domain, eg. bupt.edu.cn
- value= Domain name of Authoritative Name Server

*bupt.edu.cn* **NS** *IN* 86400 *ns.bupt.edu.cn*

(name, type, class, TTL, RDATA )

## Example of Resource Records (2)

- Type=CNAME

- Name= domain name
- Value= canonical name

*dns.bupt.edu.cn CNAME IN 86400 ns.bupt.edu.cn*

- Type=MX

- Name= Domain
- Value= canonical name of mail server

*bupt.edu.cn MX IN 86400 mail.bupt.edu.cn*



# DNS command: nslookup

---

- Function: query Internet name servers interactively
- Examples:

```
[shiyang@localhost]$ nslookup mail.263.net
```

```
Server:          202.106.0.20
```

```
Address:         202.106.0.20#53
```

```
Non-authoritative answer:
```

```
Name:   mail.263.net
```

```
Address: 211.150.96.52
```

```
Name:   mail.263.net
```

```
Address: 211.150.96.51
```

```
[shiyang@localhost]$ nslookup 211.150.96.52
```

```
Server:          202.106.0.20
```

```
Address:         202.106.0.20#53
```

```
Non-authoritative answer:
```

```
52.96.150.211.in-addr.arpa
```

```
name = mail.263.net.
```

`nslookup -query = <type> <target-domain>`

```
student@BUPTIA:~$ nslookup 166.111.4.100
Server:          10.3.9.5
Address:         10.3.9.5#53

Non-authoritative answer:
100.4.111.166.in-addr.arpa      name = www.tsinghua.edu.cn.
```

```
student@BUPTIA:~$ nslookup -query=PTR 166.111.4.100
Server:          10.3.9.5
Address:         10.3.9.5#53

Non-authoritative answer:
100.4.111.166.in-addr.arpa      name = www.tsinghua.edu.cn.
```

```
student@BUPTIA:~$ nslookup -query=MX bupt.edu.cn
Server:          10.3.9.5
Address:         10.3.9.5#53

Non-authoritative answer:
BUPT.edu.cn      mail exchanger = 5 mx1.bupt.edu.cn.
BUPT.edu.cn      mail exchanger = 5 mx3.bupt.edu.cn.
BUPT.edu.cn      mail exchanger = 5 mx2.bupt.edu.cn.
```

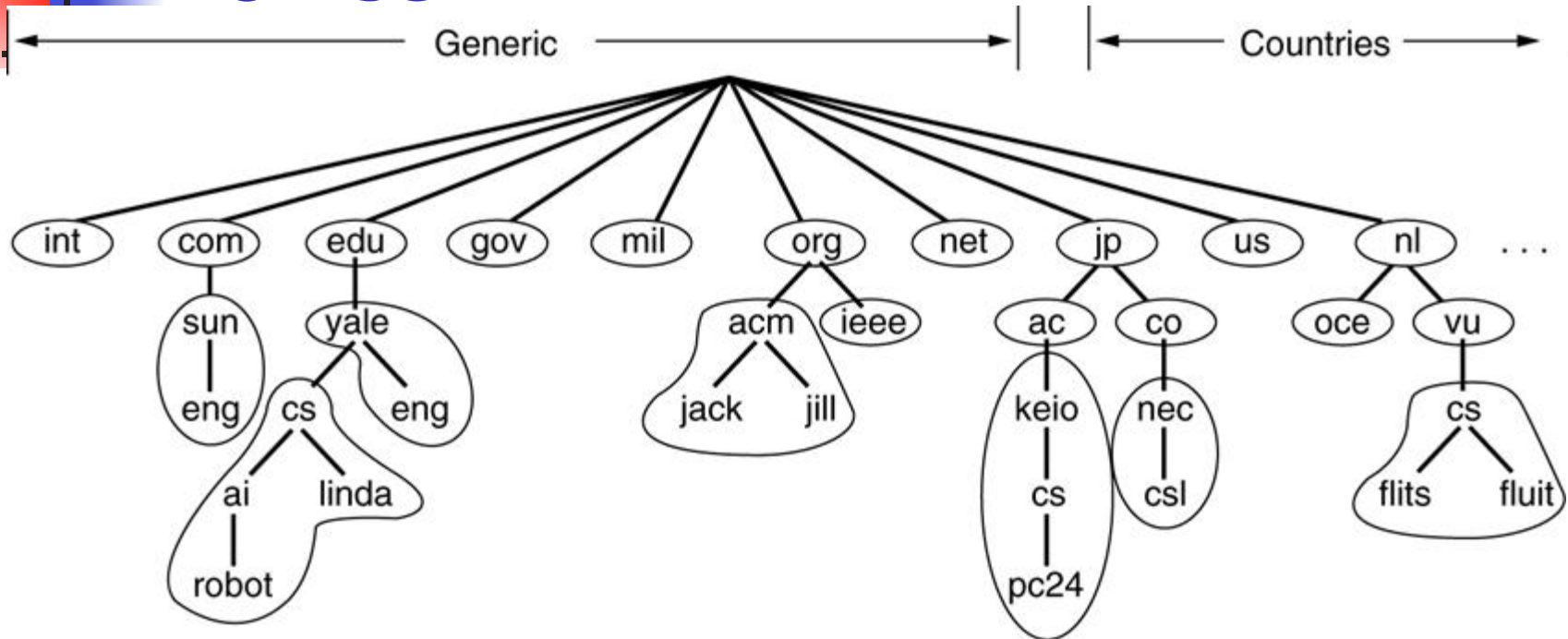


# Name Servers (1)

---

- Name servers are the **repositories** of information that make up the domain database.
- The database is divided up into sections called **zones**, which are distributed among the name servers. A zone may be one or more domains or even a sub-domain
- Each name server handles **one or more** zones. And the essential task of a name server is to **answer queries** using data in its zones.
- Name servers can answer queries in a simple manner. The response can always be generated **using only local data**, and either contains **the answer to the question** or **a referral to other name servers** "closer" to the desired information.
- A given zone will be **available from several name servers** to ensure its availability.

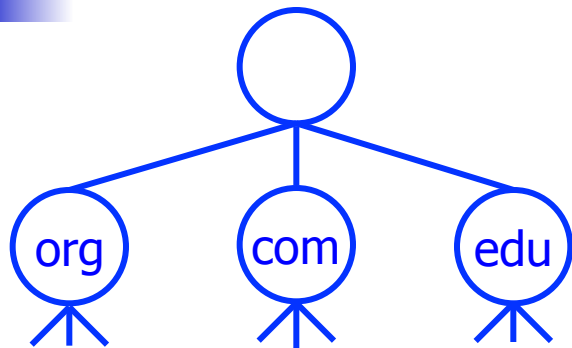
# Zones



- A **zone** corresponds to an **administrative** authority that is responsible for that portion of the hierarchy
- Eg. **BUPT** controls *x.bupt.edu.cn*

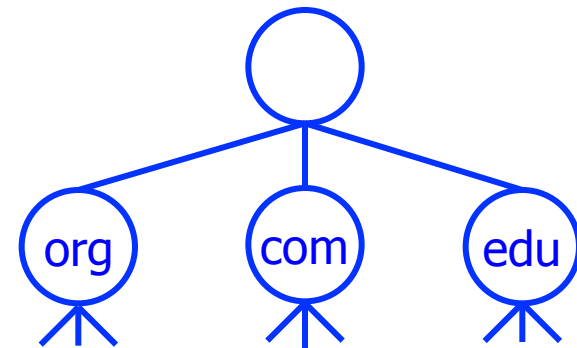
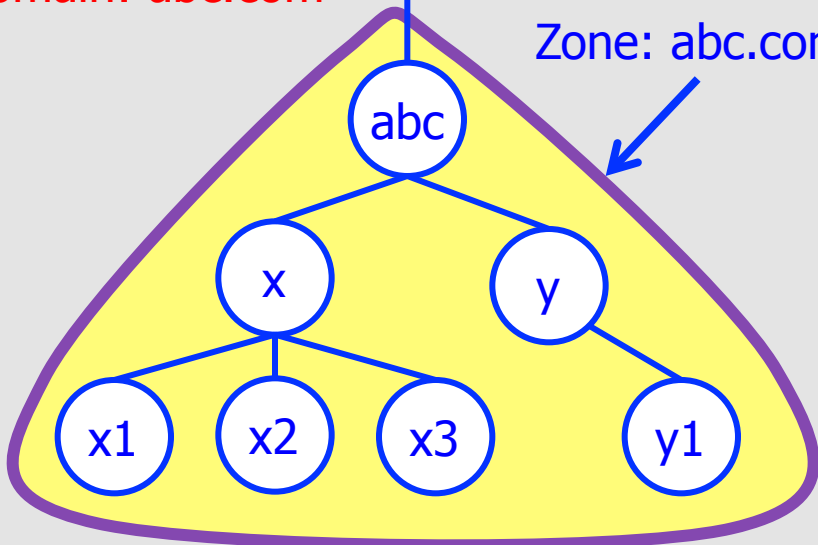


# Example of Zone and domain



Domain: abc.com

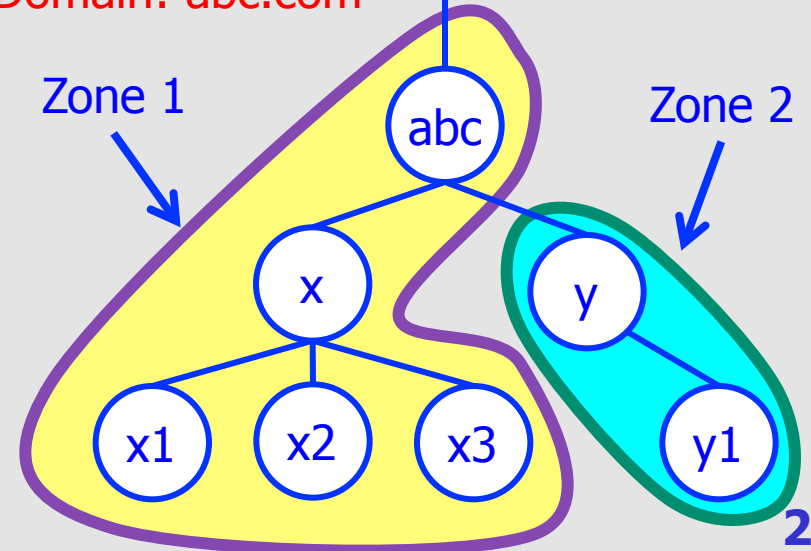
Zone: abc.com



Domain: abc.com

Zone 1

Zone 2





# Name Servers (2)

---

- **Primary server / Authoritative server**
  - holds in its database the name-to-address mappings for the group of hosts it administers
  - knows the **official** answer
- **Secondary server**
  - maintains a copy of the Primary Server's database
- **Caching server**
  - asks DNS queries to other servers but maintains a cache of the responses together with a “time to live” value
  - **non-authoritative** data about other parts of the tree



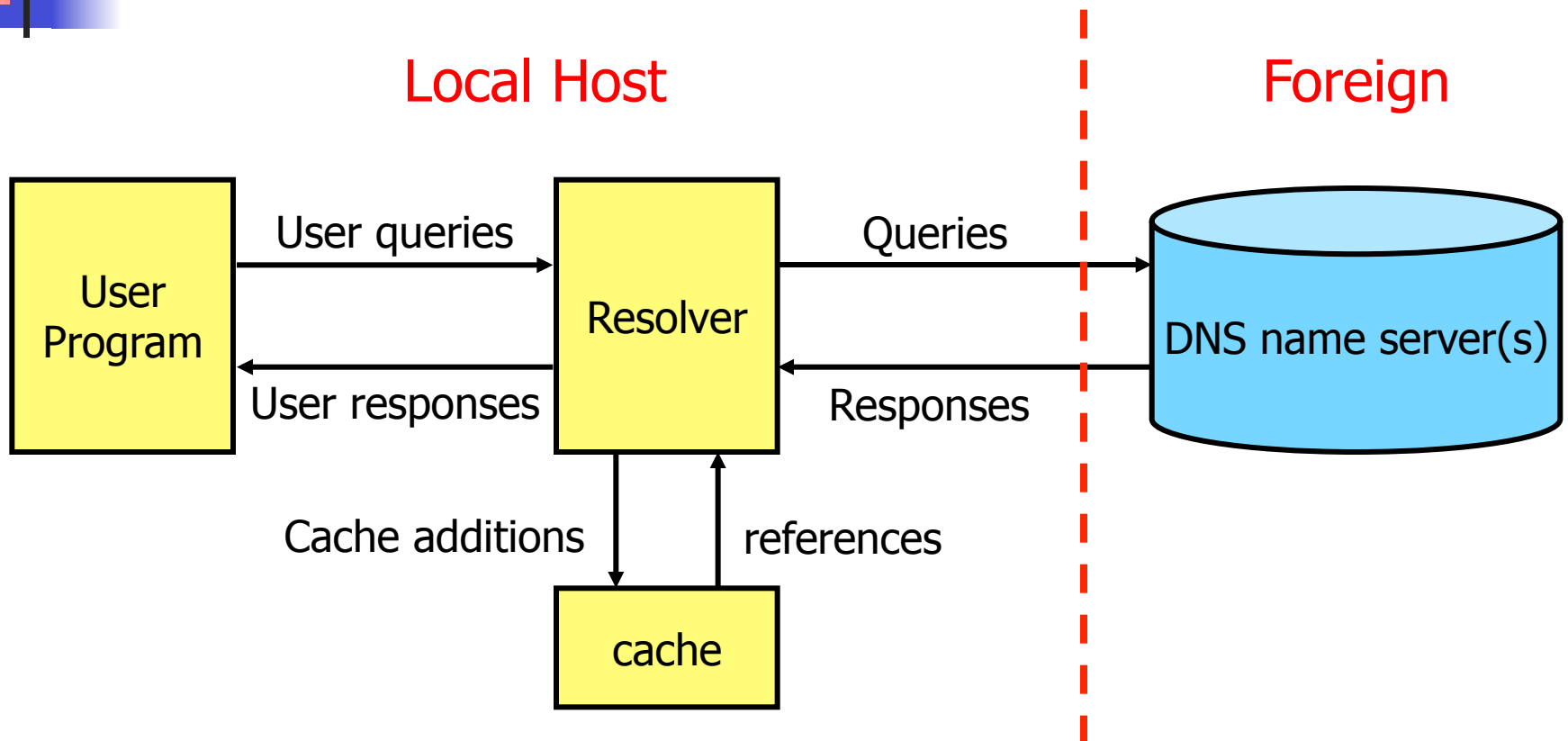
# Name Resolvers (1)

---

- The **client** side of DNS
- A resolver is the **interface** between the user program and the domain name servers
  - A resolver receives a request from a user program (e.g., mail programs, TELNET, FTP)
  - asks questions to the DNS system on behalf of the application
  - returns the desired information

# Name Resolvers (2)

– communication model



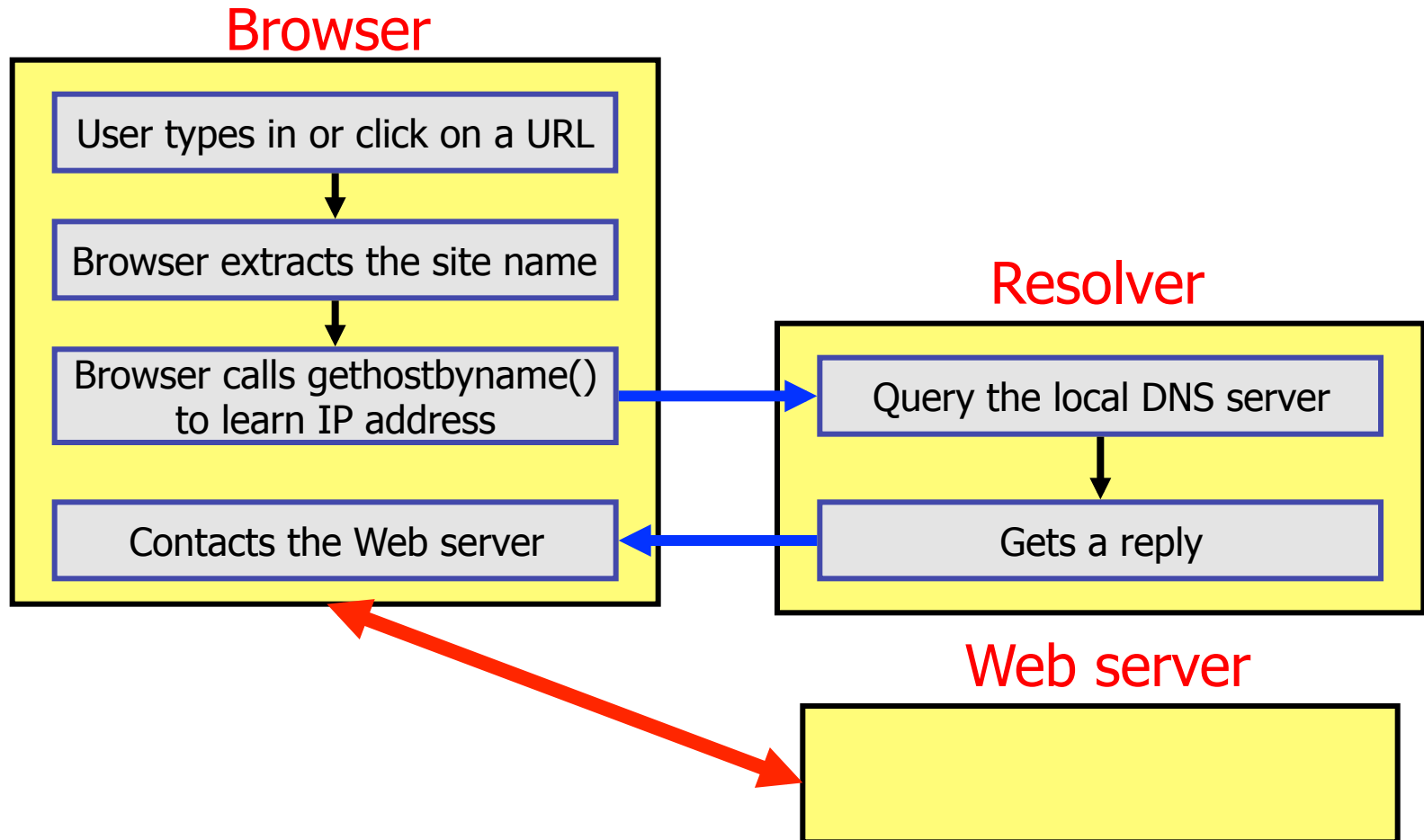
Resolvers are normally implemented in system calls.

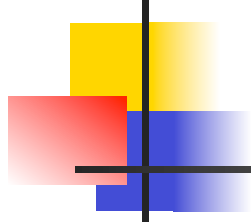
E.g. `gethostbyname()`, `gethostbyaddr()`

# Name Resolvers (3)

## – an example

- DNS working together with HTTP application





---

# DNS Services



# DNS Services

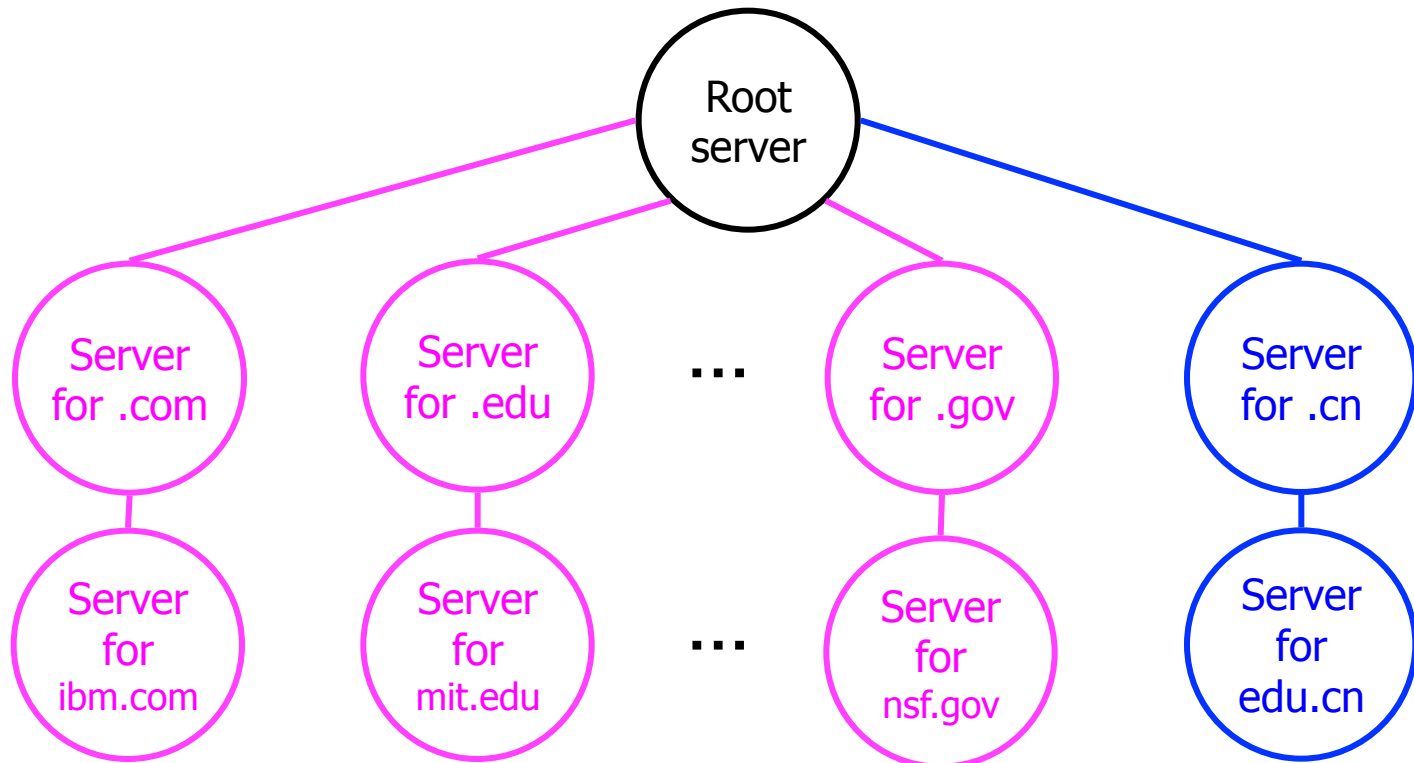
---

- Mapping domain name to addresses
- Inverse queries (optional)
- Pointer queries

# Mapping Domain Names to Addresses (1)

## – arrangement of name servers

- Conceptual arrangement of name servers in a **tree** that corresponding to the name hierarchy
- Each server knows the addresses of all **lower-level servers** for all **subdomains** within the domain it handles

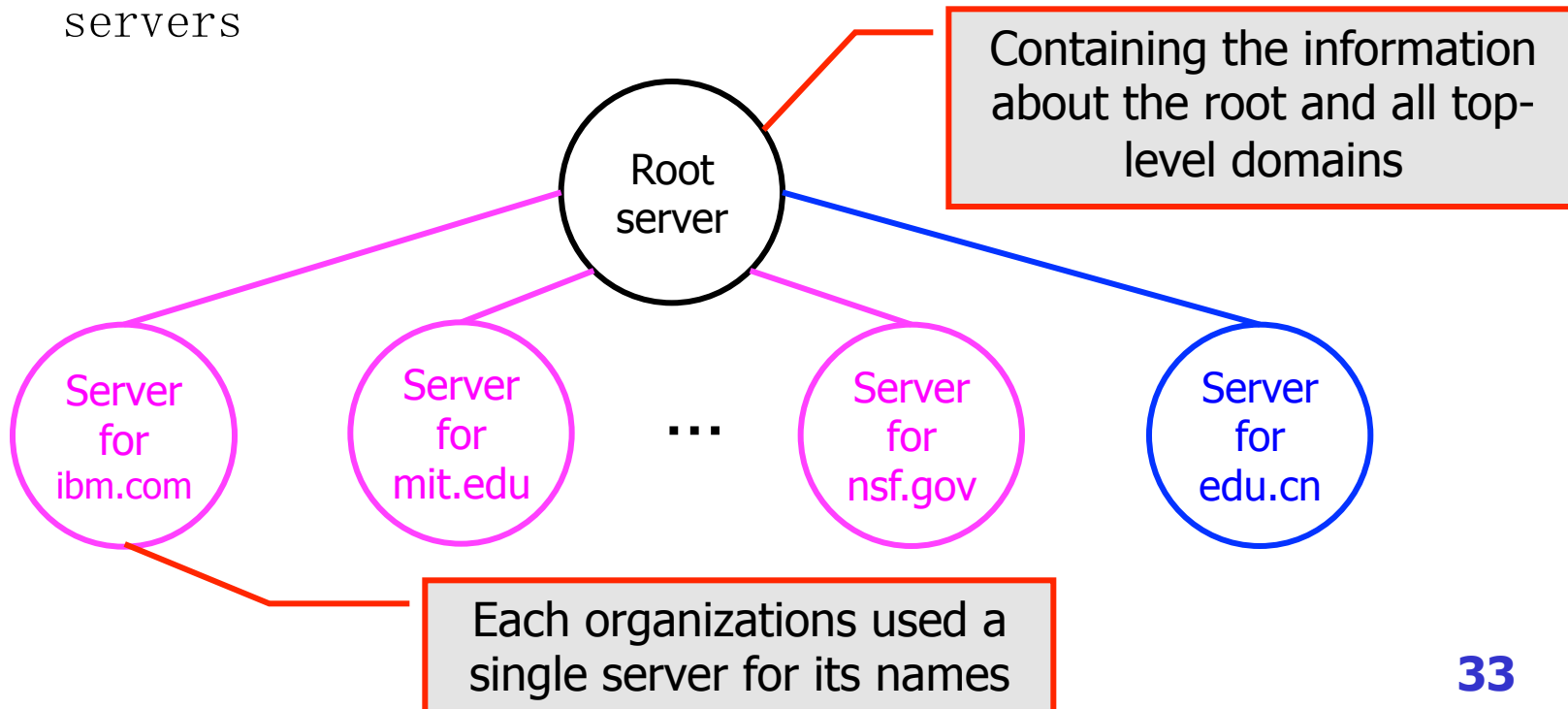




# Mapping Domain Names to Addresses (2)

## – arrangement of name servers

- A more realistic organization of name servers
- The tree is broad and flat and fewer servers need to be contacted when resolving a name
- Globally, there are 13 root servers and multiple mirror servers





# Mapping Domain Names to Addresses (3)

## – name resolution process

---

- **Two-step** name resolution process
  - Beginning with the **local name server** (default name server)
  - If the local server can not resolve a name, the query must be sent to another server in the domain system
- It can improve the query efficiency because most queries to name servers refer to local name.



# Mapping Domain Names to Addresses (4)

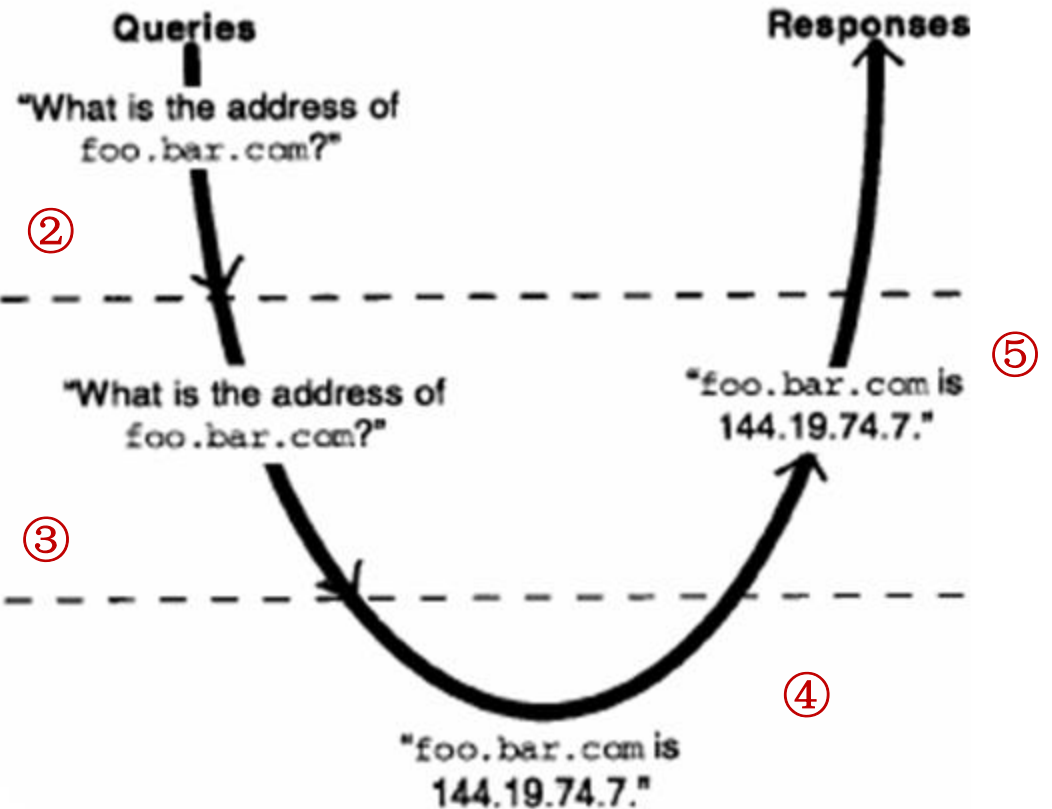
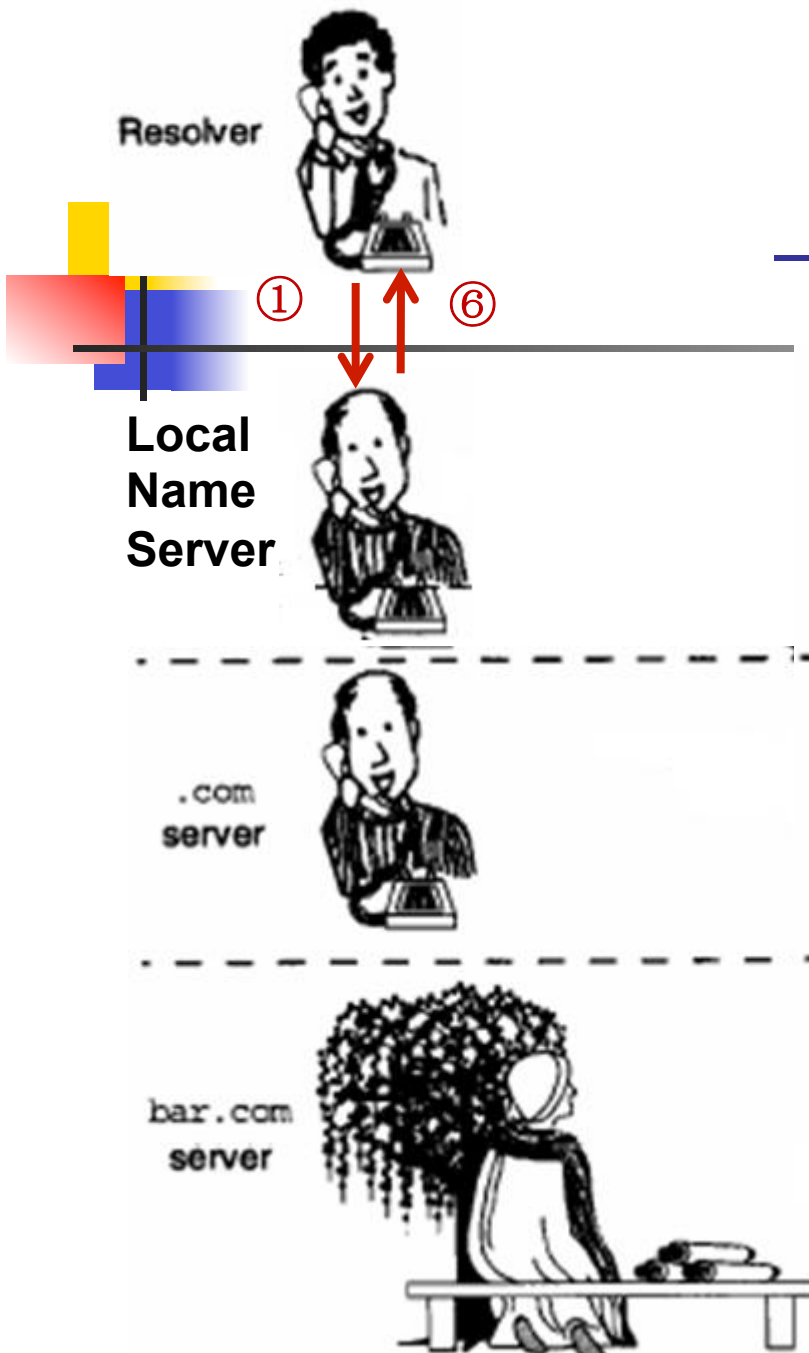
## – two name resolution methods

---

- A query is made to a **local name server**. If local name server can not resolve the name, then it will query another server.
- **Recursive resolution:**
  - If the **queried server** does not have the information, it must **make** the appropriate **query** or queries to get the information
  - Generally, a server fulfills a recursive query either with data in its own memory or by making another recursive query
- **Iterative resolution:**
  - If the queried server does not have the information, it may then respond with the address of another server; the local name server (on behalf of resolver )then queries that server (which might respond with the address of another server, and so on)
  - Commonly used by name servers on the **Internet**

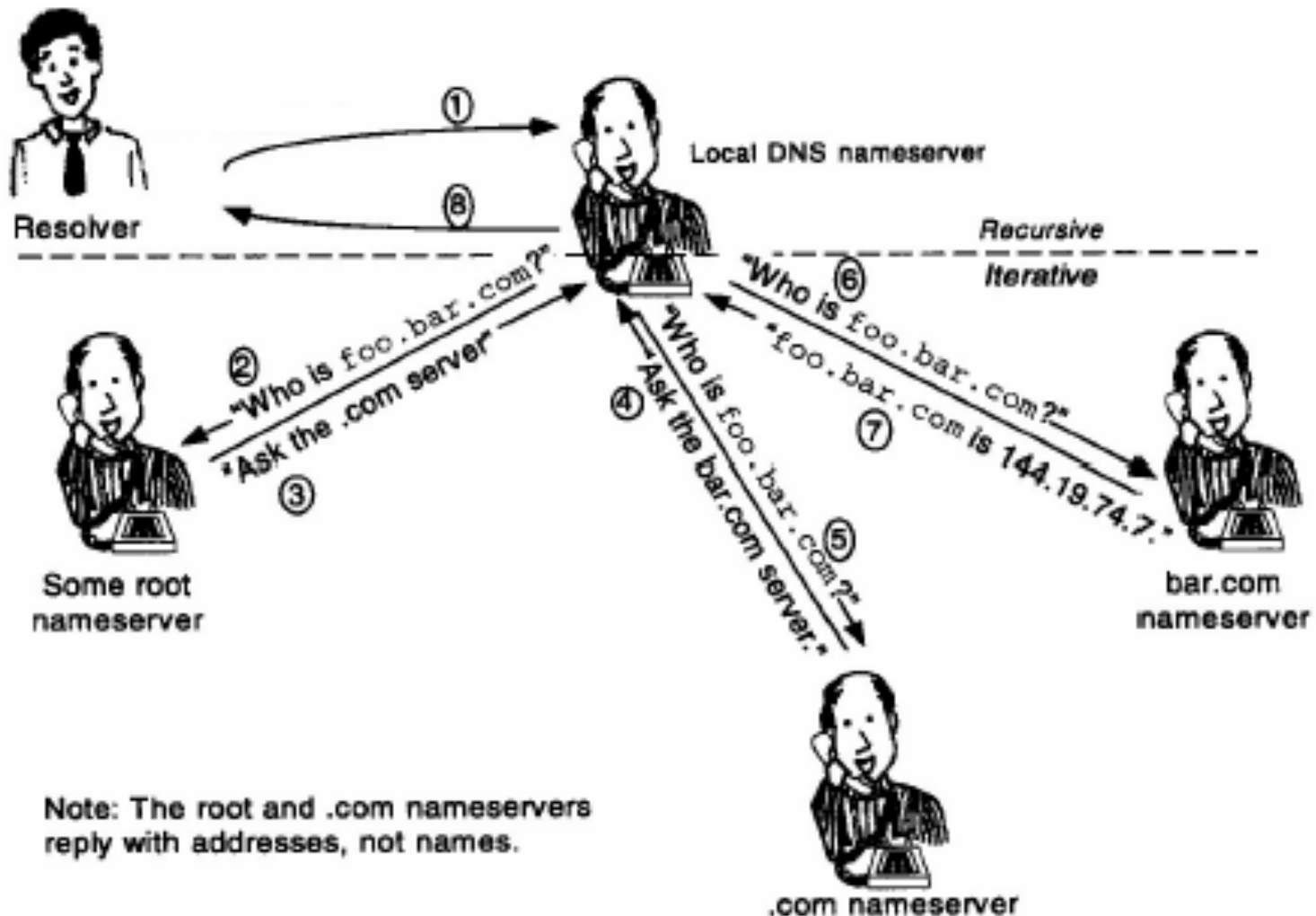
# Mapping Domain Names to Addresses (5)

– example of recursive resolution

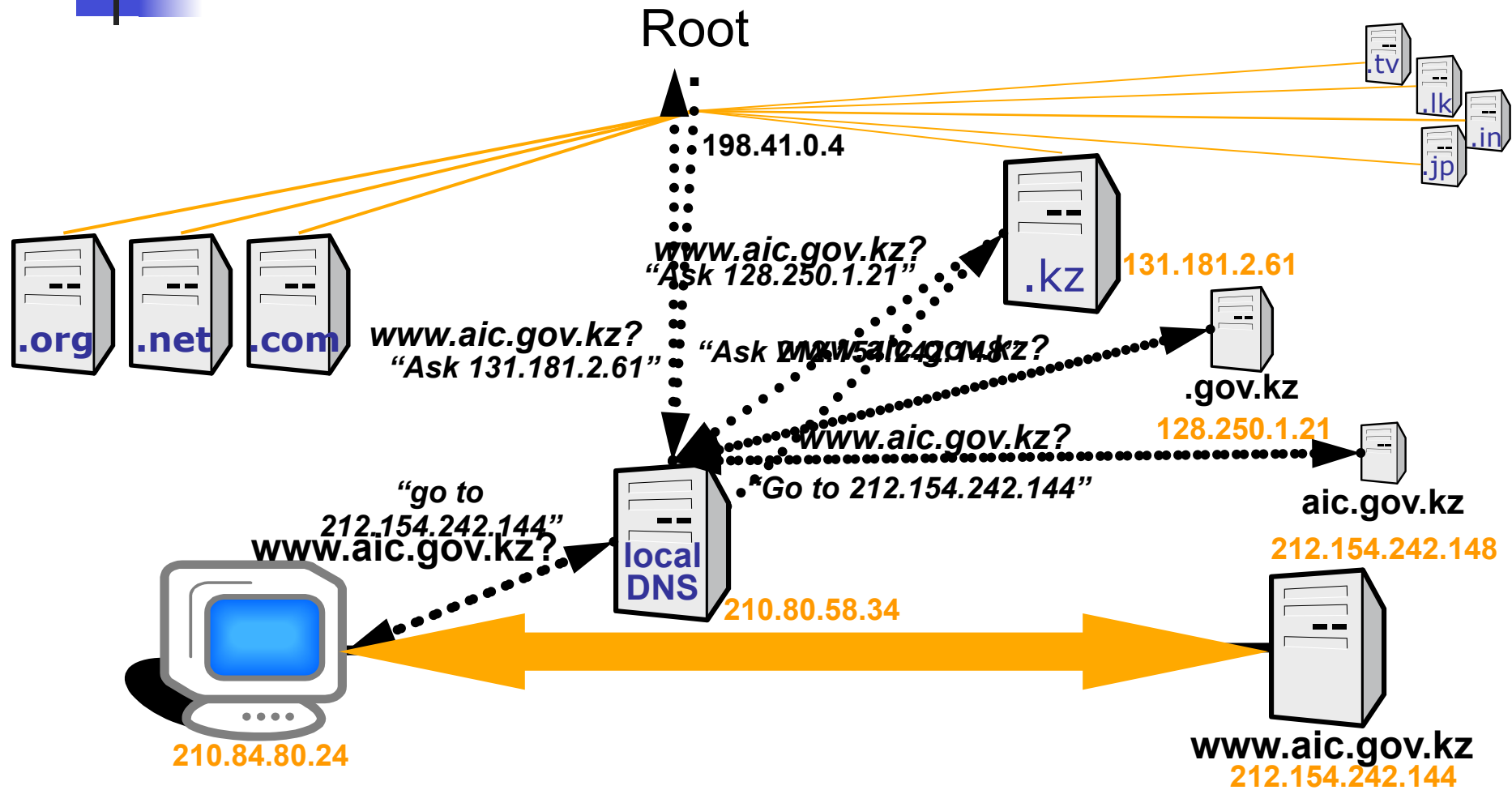


# Mapping Domain Names to Addresses (6)

– example of iterative resolution



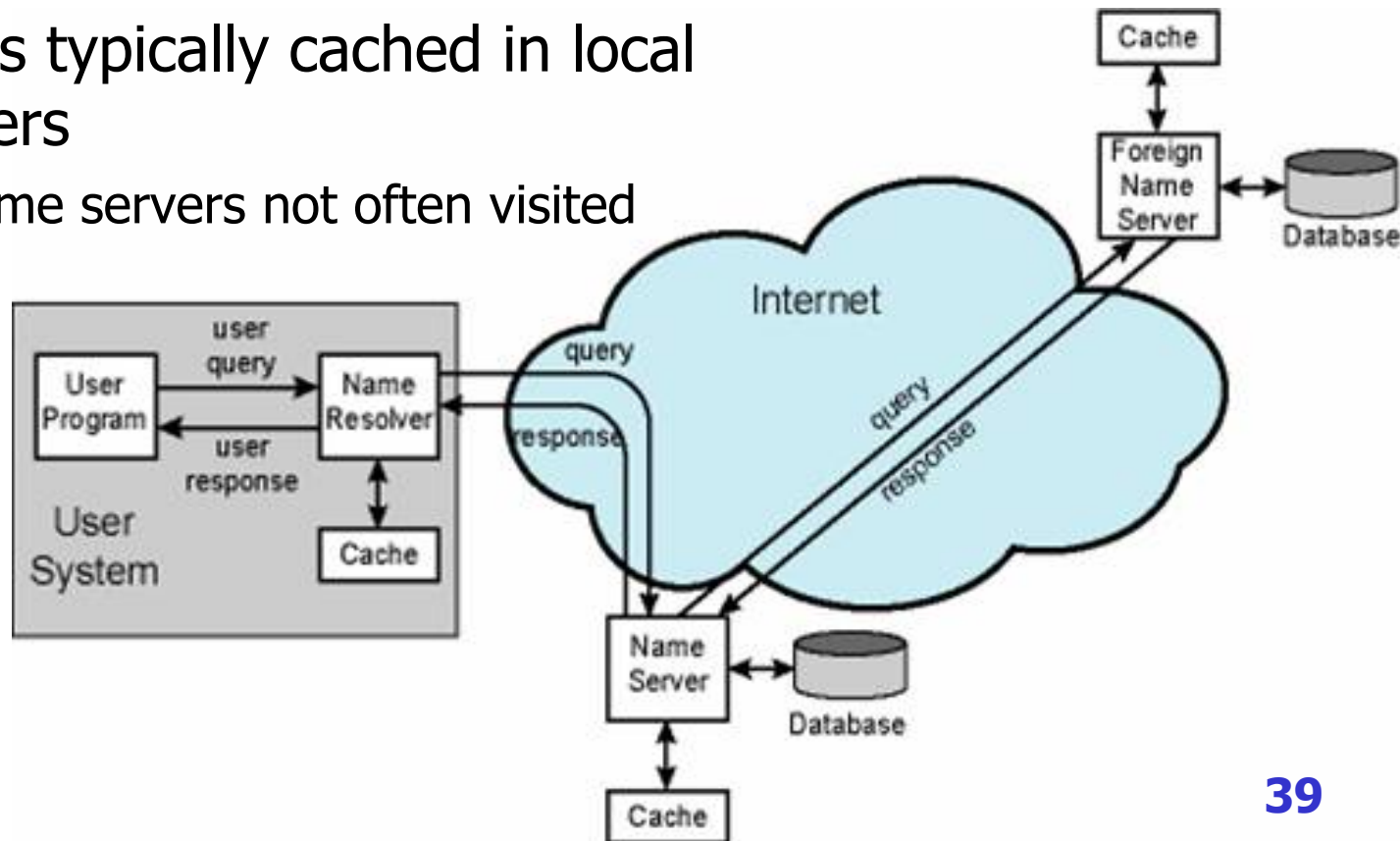
# Internet addressing: an example of DNS



# Mapping Domain Names to Addresses (7)

– caching mechanism to improve efficiency

- Caching at the name servers
- Caching at the hosts
- TLD servers typically cached in local name servers
- Thus root name servers not often visited





# Inverse Queries

---

- Mapping a particular resource to a domain name or domain names that have that resource
  - **Standard query**: mapping a domain name to a resource
  - **Inverse query**: mapping a **resource** to a **domain name**
- **Optional** part of DNS
- Generally **NOT** used because there is often no way to find the server that can resolve the query without searching the entire set of servers
- **NOT** an acceptable method of mapping host addresses to host names, use the **IN-ADDR.ARPA domain** instead





# Pointer Queries

---

- Using IN-ADDR.ARPA domain for address to host mapping
  - An IP address in dotted-decimal format is included in the query
  - The correct domain name for the machine with the specific IP address
- **Note:** Since the IN-ADDR.ARPA special domain and the normal domain for a particular host or gateway will be in different zones, the possibility exists that that the data may be inconsistent

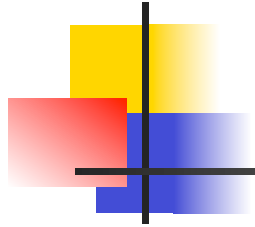




# Inverse query vs. Pointer query

---

- Similarity: IP address → domain name
- Differences:
  - Inverse query uses the same domains as standard query, may need to search the entire set of servers
  - Pointer query uses IN-ADDR.ARPA domain



# DNS Protocols



# Related RFCs

---

- **RFC1034**, DOMAIN NAMES - CONCEPTS AND FACILITIES
- **RFC1035**, DOMAIN NAMES - IMPLEMENTATION AND SPECIFICATION
- Over 137 DNS related RFCs
  - See webpage at <http://www.dns.net/dnsrd/rfc/>

- Query* and *Response* messages, both with *same message format*

**15 16**

31

ID	QR	OPCODE	AA	TC	RD	RA	Z	Rcode
Question count	Answer count							
Authority count	Additional count							
<p><b>Question Section</b> (variable number of questions)</p>								
<p><b>Answer Section</b> (variable number of RRs)</p>								
<p><b>Authority Section</b> (variable number of RRs)</p>								
<p><b>Additional Section</b> (variable number of RRs)</p>								



# DNS Message Header Structure(1)

---

- **ID**: 16-bit field used to correlate queries and responses.
- **QR**: 1-bit field that identifies the message as a query (0) or response (1).
- **OPCODE**: 4-bit field that describes the type of query:
  - 0: Standard query (name to address). 1: Inverse query (address to name). 2: Server status request.
- **AA**: Authoritative Answer. 1-bit field. When set to 1, identifies this response is made by an authoritative name server.
- **TC**: Truncation. 1-bit field. When set to 1, indicates the message has been truncated due to length greater than that permitted.
- **RD**: Recursion Desired. 1-bit field. Set to 1 by the resolver to request recursive service by the name server.
- **RA**: Recursion Available. 1-bit field. Set to 1 by name server to indicate recursive query support is available.
- **Z**: 3-bit field. Reserved for future use. Must be set to 0.



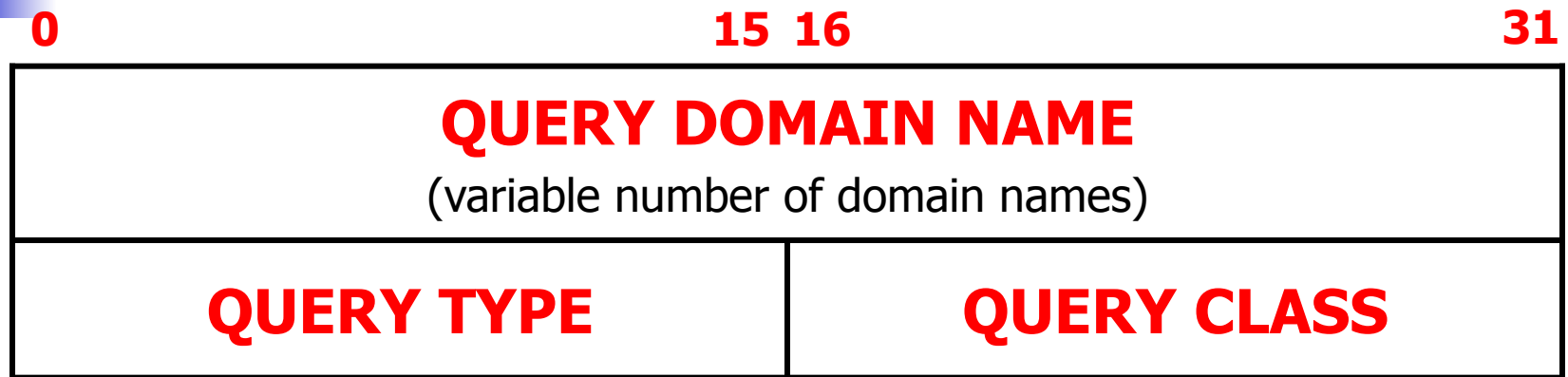
# DNS Message Header Structure(2)

---

- **RCODE**: Response Code. 4-bit field that is set by the name server to identify the status of the query:
  - 0: No error condition. 1: Unable to interpret query due to format error.
  - 2: Unable to process due to server failure. 3: Name in query does not exist.
  - 4: Type of query not supported. 5: Query refused for policy reasons.
- **QDCOUNT**(Question count): 16-bit field that defines the number of entries in the question section.
- **ANCOUNT**(Answer count): 16-bit field that defines the number of resource records in the answer section.
- **NSCOUNT**(Authority count): 16-bit field that defines the number of name server resource records in the authority section.
- **ARCOUNT**(Additional count): 16-bit field that defines the number of resource records in the additional records section.



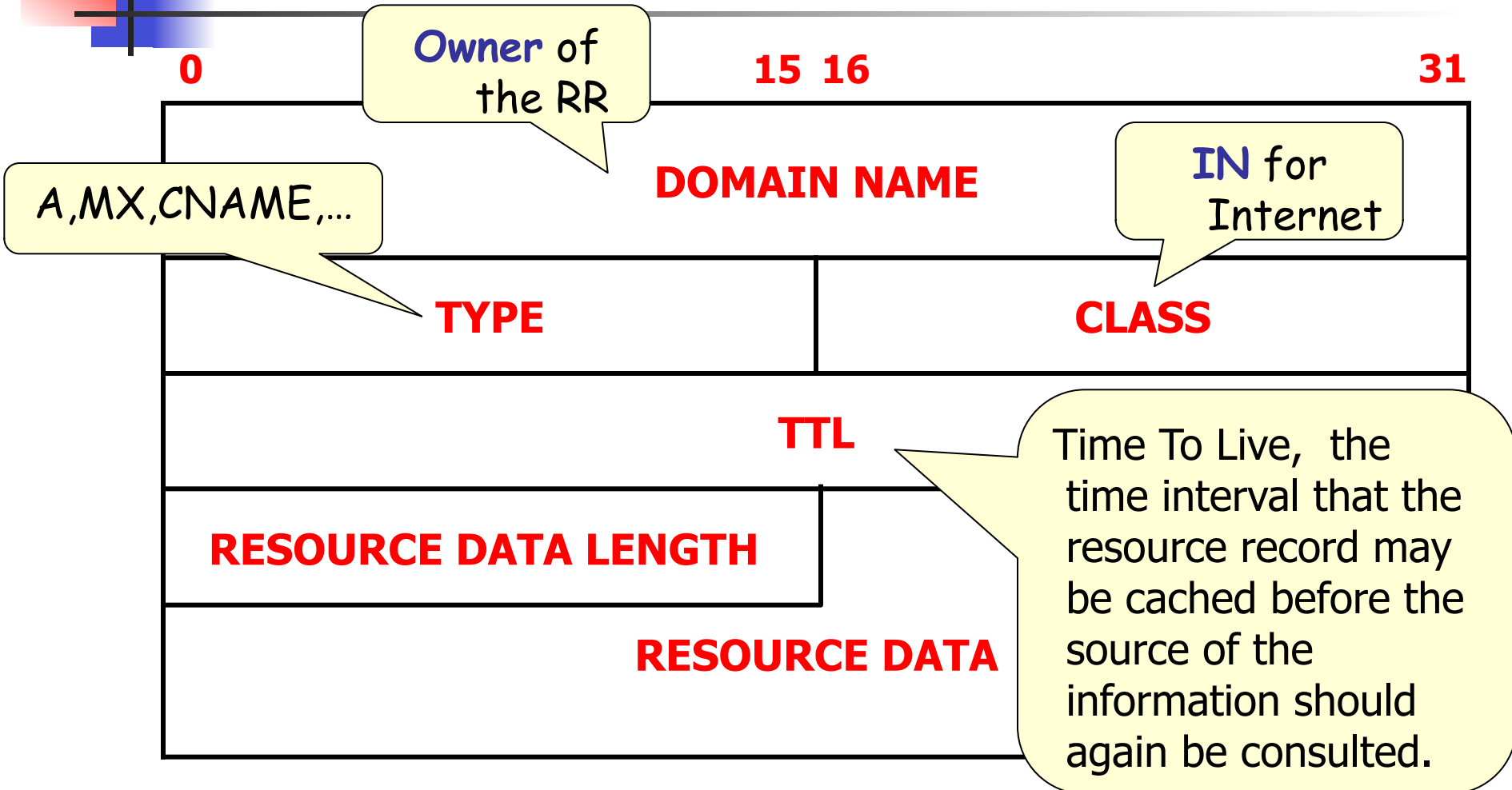
# Question Section Format



- **QUERY TYPE**: 16-bit field used to specify the type of the query
  - A – Host address
  - MX – Mail exchanger for the domain
  - ...
- **QUERY CLASS**: 16-bit field used to specify the class of the query
  - IN – Internet system
  - ...



# Response Section: Resource Record Format





# Resource Record Format (2)

## – type field

---

- SOA
  - Start Of Authority--identifies the domain or zone and sets a number of parameters
- NS
  - Maps a domain name to the name of a computer that is authoritative for the domain
- A
  - Maps the name of a system to its address. If a system (e.g., a router) has several addresses, then there will be a separate record for each.
- AAAA
  - Maps the name of a system to its IPv6 address. If a system (e.g., a router) has several addresses, then there will be a separate record for each.
- CNAME
  - Maps an alias name to the true, canonical name
- MX
  - Mail Exchanger. Identifies the systems that relay mail into the organization



# Resource Record Format (3)

## – type field

---

- TXT
  - Provides a way to add text comments to the database. For example, a txt record could map abc.com to the company's name, address, and telephone number
- WKS
  - Well Known Services. It can list the application services available at the host. Used sparingly, if at all
- HINFO
  - Host Information, such as computer type and model. Rarely used
- PTR
  - Maps an IP address to a system name. Used in address-to-name files.



# Sample of DNS Database

---

Domain name	TTL	Class	Type	Value
; Authoritative data for cs.vu.nl				
cs.vu.nl.	86400	IN	SOA	star boss (952771,7200,7200,2419200,86400)
cs.vu.nl.	86400	IN	TXT	"Divisie Wiskunde en Informatica."
cs.vu.nl.	86400	IN	TXT	"Vrije Universiteit Amsterdam."
cs.vu.nl.	86400	IN	MX	1 zephyr.cs.vu.nl.
cs.vu.nl.	86400	IN	MX	2 top.cs.vu.nl.
flits.cs.vu.nl.	86400	IN	HINFO	Sun Unix
flits.cs.vu.nl.	86400	IN	A	130.37.16.112
flits.cs.vu.nl.	86400	IN	A	192.31.231.165
flits.cs.vu.nl.	86400	IN	MX	1 flits.cs.vu.nl.
flits.cs.vu.nl.	86400	IN	MX	2 zephyr.cs.vu.nl.
flits.cs.vu.nl.	86400	IN	MX	3 top.cs.vu.nl.
www.cs.vu.nl.	86400	IN	CNAME	star.cs.vu.nl
ftp.cs.vu.nl.	86400	IN	CNAME	zephyr.cs.vu.nl

# Example of DNS Queries and Responses(1) (RFC1034)

The query would look like:

Header	OPCODE=QUERY	
Question	QNAME=SRI-NIC.ARPA., QCLASS=IN, QTYPE=A	
Answer	<empty>	
Authority	<empty>	
Additional	<empty>	

Type=A

The response from C.ISI.EDU would be:

Header	OPCODE=QUERY, RESPONSE, <u>AA</u>	
Question	QNAME=SRI-NIC.ARPA., QCLASS=IN, QTYPE=A	
Answer	SRI-NIC.ARPA. 86400 IN A 26.0.0.73 86400 IN A 10.0.0.51	
Authority	<empty>	
Additional	<empty>	

Authoritative  
answer (AA)  
is returned.

# Example of DNS Response(2)

## 6.2.3. QNAME=SRI-NIC.ARPA, QTYPE=MX

This type of query might be result from a mailer trying to look up routing information for the mail destination HOSTMASTER@SRI-NIC.ARPA. The response from C.ISI.EDU would be:

Header	OPCODE=QUERY, RESPONSE, AA	
Question	QNAME=SRI-NIC.ARPA., QCLASS=IN, QTYPE=MX	Type=MX
Answer	SRI-NIC.ARPA. 86400 IN MX 0 SRI-NIC.ARPA.	MX record
Authority	<empty>	
Additional	SRI-NIC.ARPA. 86400 IN A 26.0.0.73   A 10.0.0.51	A record of mail exchanger

# Example of DNS Response(3)

6.2.6. QNAME=BRL.MIL, QTYPE=A

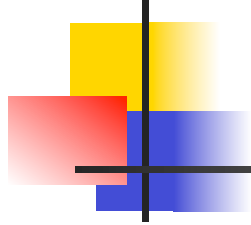
If this query is sent to C.ISI.EDU, the reply would be:

(C.ISI.EDU is not authoritative for the MIL domain)

Header	OPCODE=QUERY, RESPONSE			
Question	QNAME=BRL.MIL, QCLASS=IN, QTYPE=A			
Answer	<empty>			
Authority	MIL.	86400	IN NS	SRI-NIC.ARPA.
		86400	NS	A.ISI.EDU.
Additional	A.ISI.EDU.		A	26.3.0.103
	SRI-NIC.ARPA.		A	26.0.0.73
			A	10.0.0.51

NS record of MIL domain

A record of the  
authoritative  
servers shown in  
Authority Section



# DNS Tools





# DNS Tools: nslookup

---

- Function: query Internet name servers interactively
- Examples:

```
[shiyang@localhost]$ nslookup mail.263.net
```

```
Server:                202.106.0.20
```

```
Address:               202.106.0.20#53
```

```
Non-authoritative answer:
```

```
Name:   mail.263.net
```

```
Address: 211.150.96.52
```

```
Name:   mail.263.net
```

```
Address: 211.150.96.51
```

```
[shiyang@localhost]$ nslookup 211.150.96.52
```

```
Server:                202.106.0.20
```

```
Address:               202.106.0.20#53
```

```
Non-authoritative answer:
```

```
52.96.150.211.in-addr.arpa
```

```
name = mail.263.net.
```



# DNS Tools: dig

---

- Function: a flexible tool for interrogating DNS name servers
- Performs DNS lookups and displays the answers that are returned from the name server(s)

```
[shiyang@localhost]$ dig bupt.edu.cn
```

```
[shiyang@localhost]$ dig +norecurse bupt.edu.cn
```

```
[shiyang@localhost]$ dig +trace bupt.edu.cn
```

```
[shiyang@localhost]$ dig -x 211.68.71.130
```

**dig [domain] [q-type] [q-class] [q-opt] [d-  
opt]**

```
student@BUPTIA:~$ dig bupt.edu.cn MX
```

```
; <<>> DiG 9.9.5-3ubuntu0.8-Ubuntu <<>> bupt.edu.cn MX
```

```
;; global options: +cmd
```

```
;; Got answer:
```

```
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 32713
```

```
;; flags: qr rd ra; QUERY: 1, ANSWER: 3, AUTHORITY: 0, ADDITIONAL: 1
```

```
;; OPT PSEUDOSECTION:
```

```
; EDNS: version: 0, flags:; udp: 4096
```

```
;; QUESTION SECTION:
```

```
;bupt.edu.cn.                IN      MX
```

```
;; ANSWER SECTION:
```

```
BUPT.edu.cn.                5675    IN      MX      5 mx3.bupt.edu.cn.
```

```
BUPT.edu.cn.                5675    IN      MX      5 mx1.bupt.edu.cn.
```

```
BUPT.edu.cn.                5675    IN      MX      5 mx2.bupt.edu.cn.
```

```
;; Query time: 1555 msec
```

```
;; SERVER: 10.3.9.5#53(10.3.9.5)
```

```
;; WHEN: Tue Apr 18 09:53:00 CST 2017
```

```
;; MSG SIZE rcvd: 105
```



# Other Issues


---

- Dynamic DNS (DDNS)
  - RFC2136
- IDNS
  - Internationalized Domain Names (IDN)
- DNSSEC and other security issues
  - Security concern on DNS information exchange
- DNSv6
  - New frontier for next generation Internet
- ENUM
  - Convergence with telephony?
- More advanced topics
  - Digital Object Identifier, <http://www.doi.org/faq.html>
  - Handle System, <http://www.handle.net/>



# Summary of Important Terms in DNS

---

- Domain / domain name
- Domain namespace
- Resource Record
- Name server
- Resolver
- Zone
- Query / response 
- Standard query / inverse query / pointer query
- recursive resolution / iterative resolution
- Primary server/ secondary server / caching server



## Some procedures should be understood

---

- How does DNS work together with the user programs (e.g. TELNET, FTP, HTTP) ?
- How is the recursive resolution and iterative resolution?
- What are the mechanisms in DNS that are possible to improve the querying efficiency?
- What is similarity of inverse query and pointer query? What are their differences?