
Network Applications: DNS; Network Programming: UDP, TCP

Qiao Xiang, Congming Gao, Qiang Su

<https://sngroup.org.cn/courses/cnns-xmuf25/index.shtml>

09/18/2025

Outline

- ❑ Admin. and recap
- ❑ Basic network applications
 - Email
 - DNS
- ❑ Java in a Nutshell
- ❑ Network application programming

DNS Message Flow: Two Types of Queries

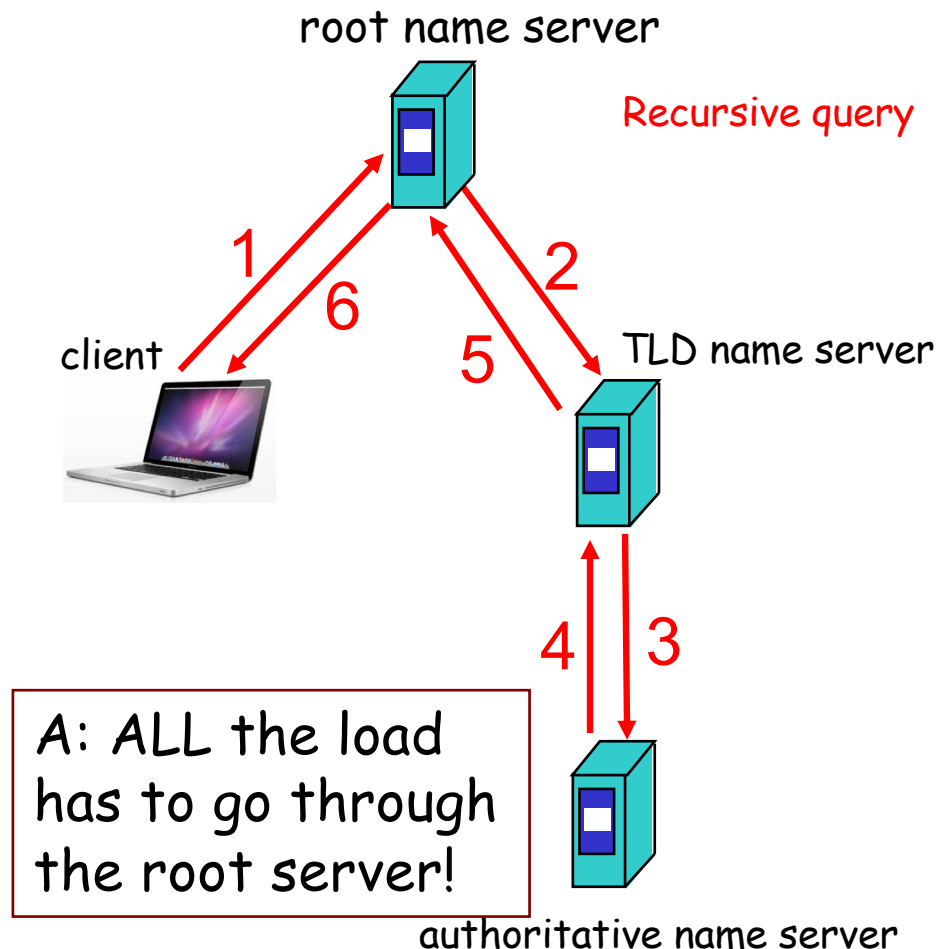
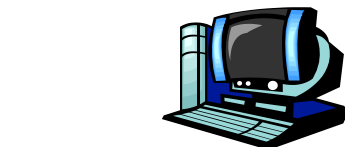
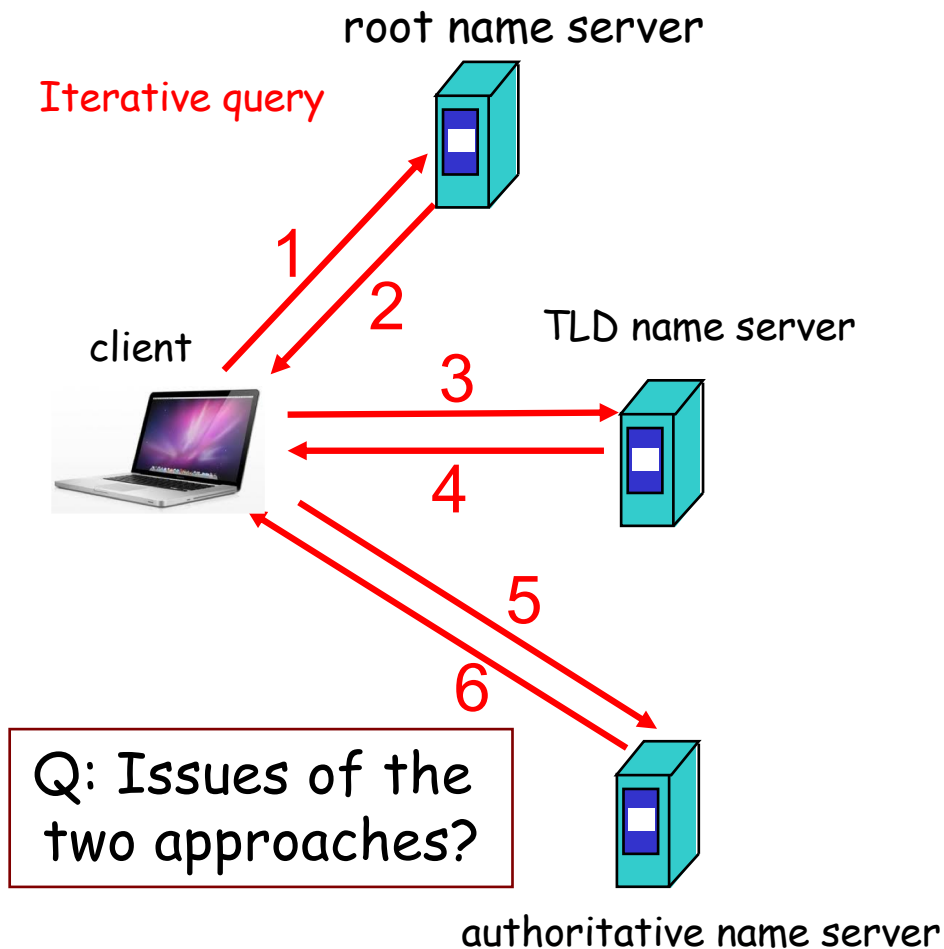
Recursive query:

- ❑ The contacted name server resolves the name completely

Iterated query:

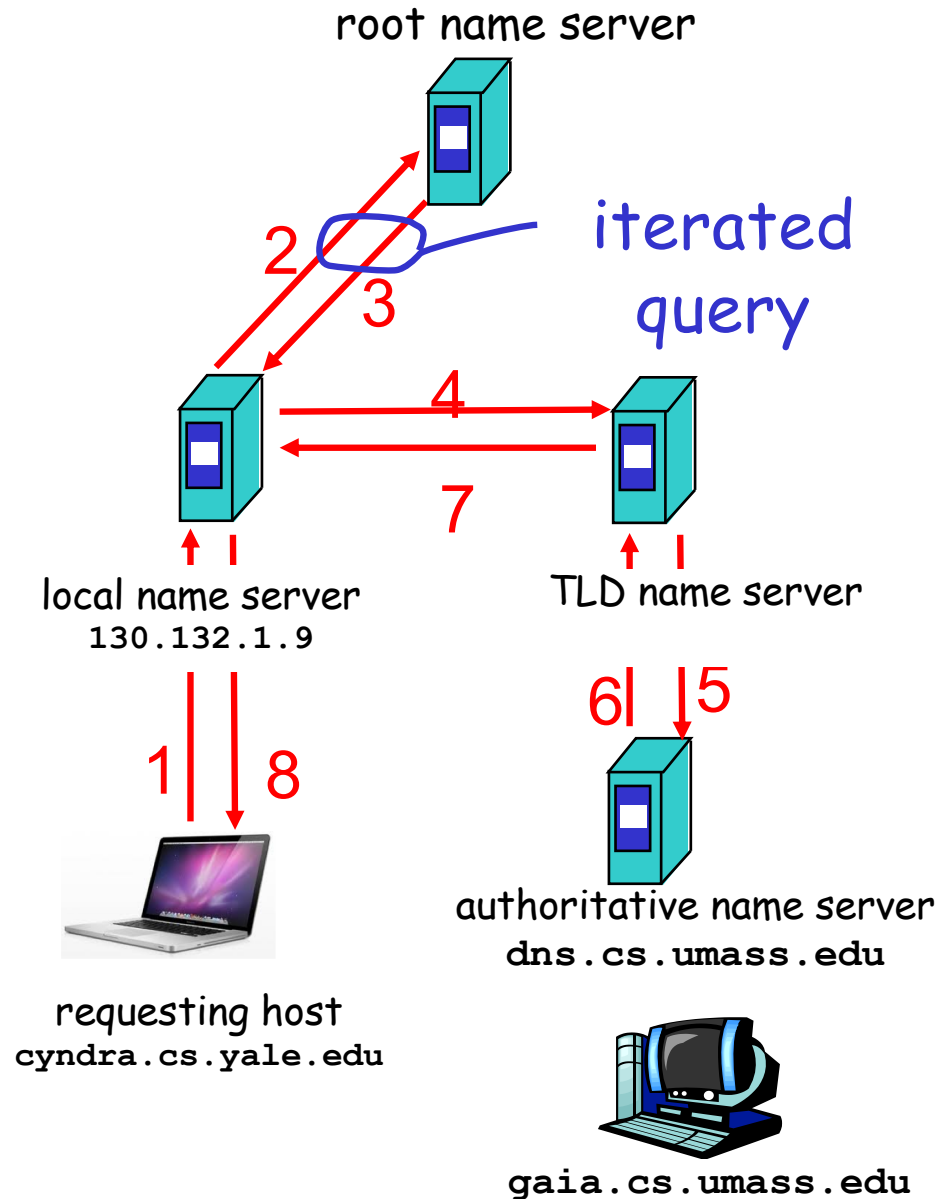
- ❑ Contacted server replies with name of server to contact
 - “I don’t know this name, but ask this server”

Two Extreme DNS Message Flows



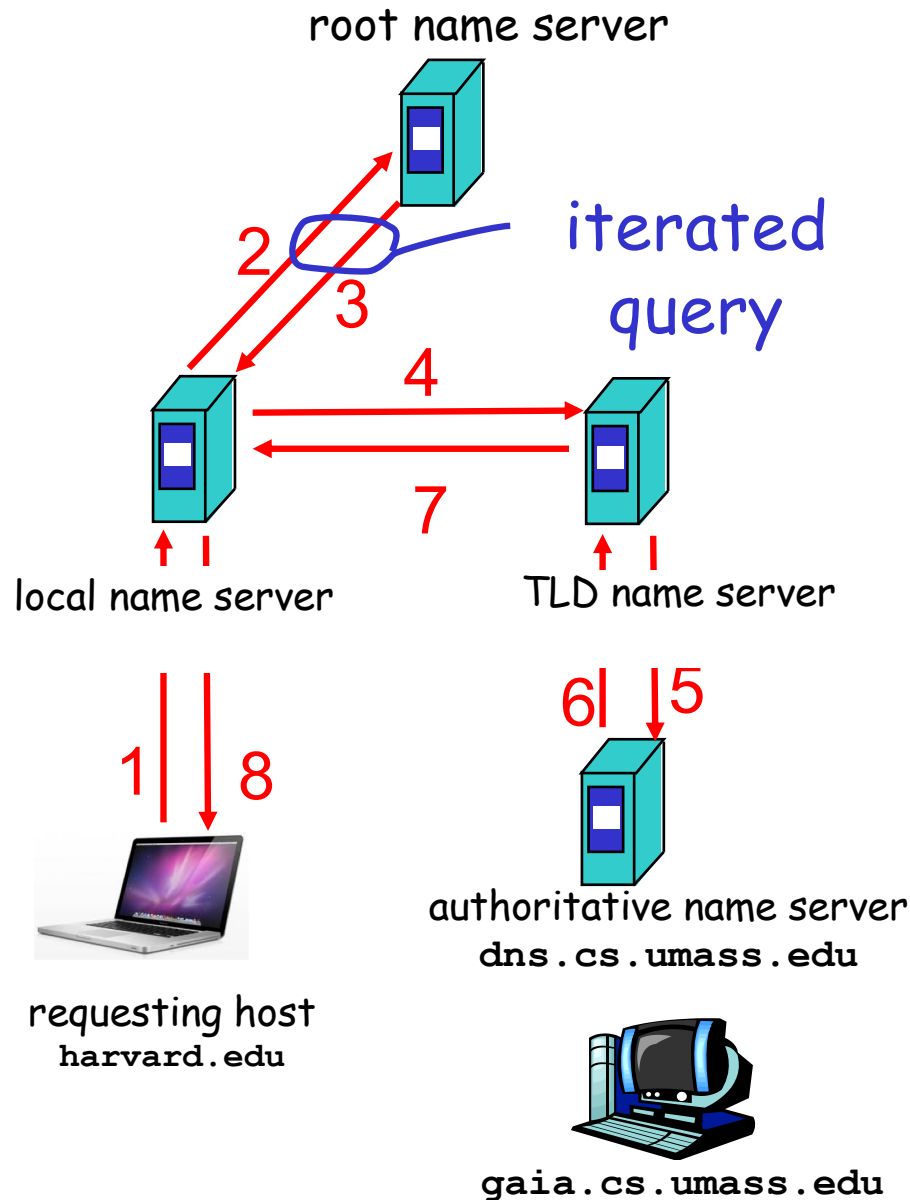
Typical DNS Message Flow: The Hybrid Case

- Host knows only local name server
- Local name server is learned from DHCP, or configured, e.g. /etc/resolv.conf
- Local DNS server helps clients resolve DNS names



Typical DNS Message Flow: The Hybrid Case

- Host knows only local name server
- Local name server is learned from DHCP, or configured, e.g. /etc/resolv.conf
- Local DNS server helps clients resolve DNS names
- Benefits of local name servers (often called **resolvers**)
 - simplifies client
 - caches/reuses results

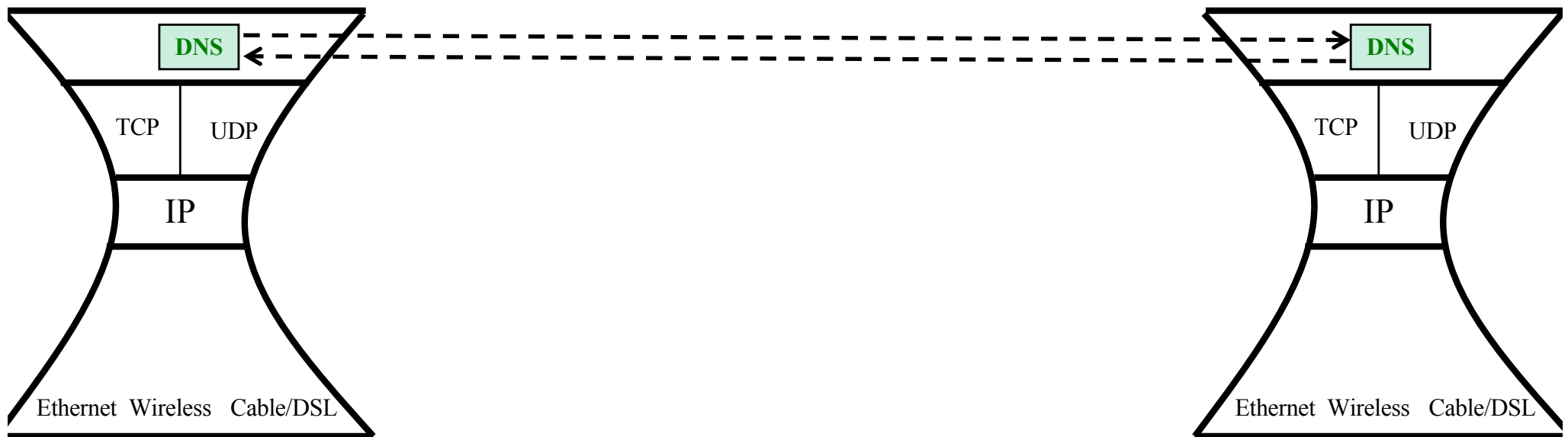


Outline

- ❑ Admin. and recap
- ❑ *DNS*
 - High-level design
 - *Details*

DNS Message Format?

Basic encoding decisions: UDP/TCP,
how to encode domain name, how to
encode answers...



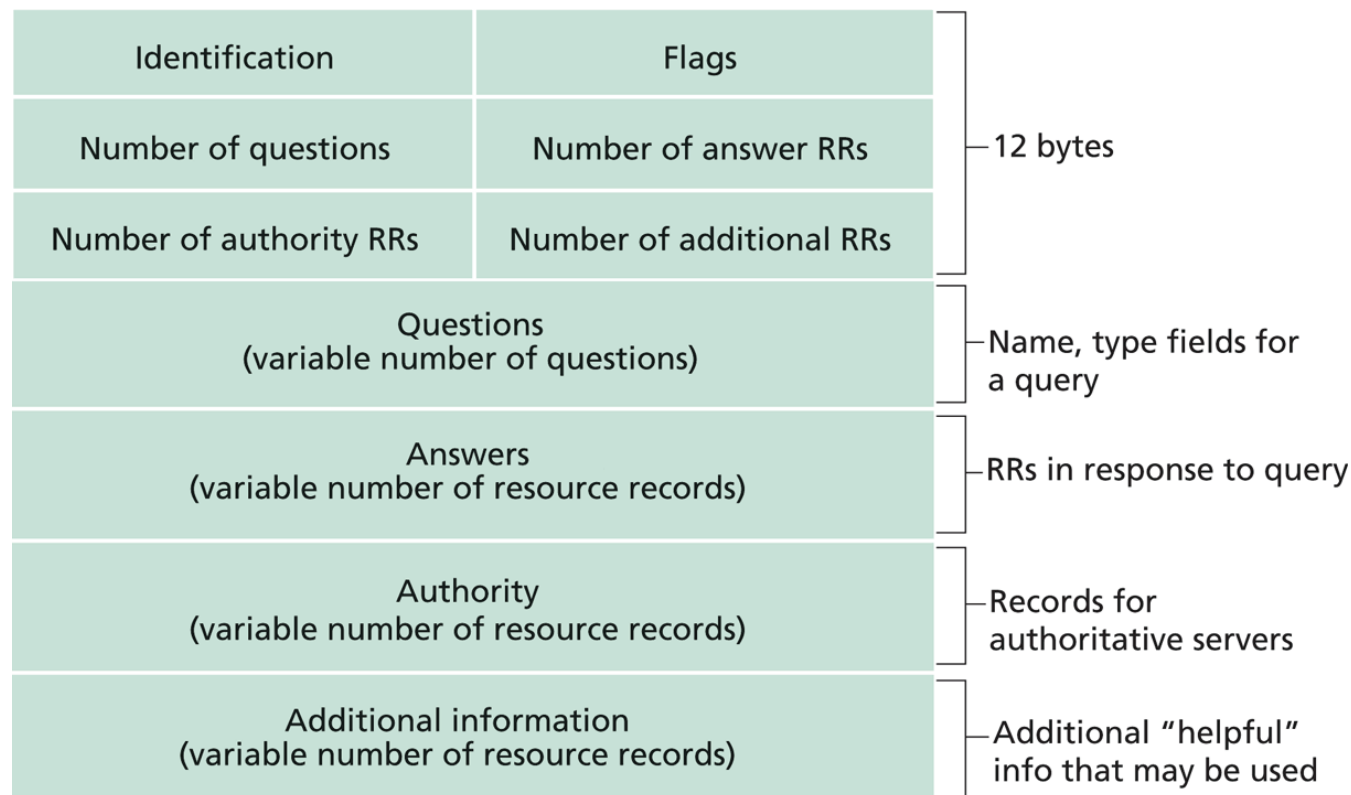
Observing DNS Messages

❑ Capture the messages

- DNS server is at port 53
 - Display and clear DNS cache
 - MacOS: <https://support.apple.com/en-us/HT202516>
`sudo killall -HUP mDNSResponder`
 - Ubuntu:
`sudo systemd-resolve --flush-caches`
`sudo systemd-resolve --statistics`
- Try to load the dns-capture file from class Schedule page, if you do not want live capture

DNS Protocol, Messages

DNS protocol : typically over UDP (can use TCP);
query and *reply* messages, both with the **same**
message format



DNS Details

- ❑ Header (Sec. 4.1.1 of <https://www.ietf.org/rfc/rfc1035.txt>)
- ❑ Encoding of questions (Sec. 4.1.2):
 - [Label-length label-chars]
- ❑ Encoding of answers (Sec. 4.1.3)
 - Pointer format
(<http://www.iana.org/assignments/dns-parameters/dns-parameters.xhtml>)
- ❑ See example DNS packets

Name Encoding

▼ Queries

▼ xmu.edu.cn: type A, class IN

Name: xmu.edu.cn

[Name Length: 10]

[Label Count: 3]

Type: A (Host Address) (1)

Class: IN (0x0001)

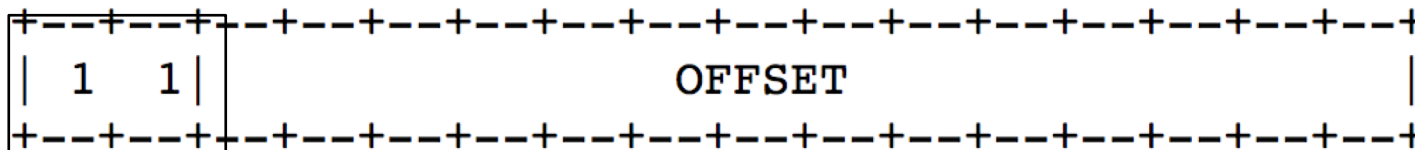
[\[Response In: 3\]](#)

0000	34	71	46	af	81	aa	08	00	27	ba	b3	67	08	00	45	00	4qF..... '...g...E..
0010	00	38	02	d4	40	00	40	11	b0	5b	c0	a8	03	34	c0	a8	.8...@...@... [...4...
0020	03	01	9b	8f	00	35	00	24	87	bb	ed	00	01	00	00	015...\$.....
0030	00	00	00	00	00	00	03	78	6d	75	03	65	64	75	02	63x mu.edu.c
0040	6e	00	00	01	00	01											n.....

Diagram illustrating the Name Encoding for xmu.edu.cn. The table shows the hexadecimal representation of the name, with labels and their lengths indicated by red arrows and text:

- cn**: Points to the last two bytes (01 00) of the first label.
- xmu**: Points to the last three bytes (75 03 65) of the second label.
- edu**: Points to the last three bytes (64 75 02) of the third label.
- length**: Points to the first byte (03) of the second label, indicating its length.

Message Compression (Label Pointer)



▼ Domain Name System (response)

Transaction ID: 0xed00

► Flags: 0x8180 Standard query response, No error

Questions: 1

Answer RRs: 1

Authority RRs: 0

Additional RRs: 0

► Queries

▼ Answers

▼ xmu.edu.cn: type A, class IN, addr 210.34.0.35

Name: xmu.edu.cn

Type: A (Host Address) (1)

Class: IN (0x0001)

DNS
start

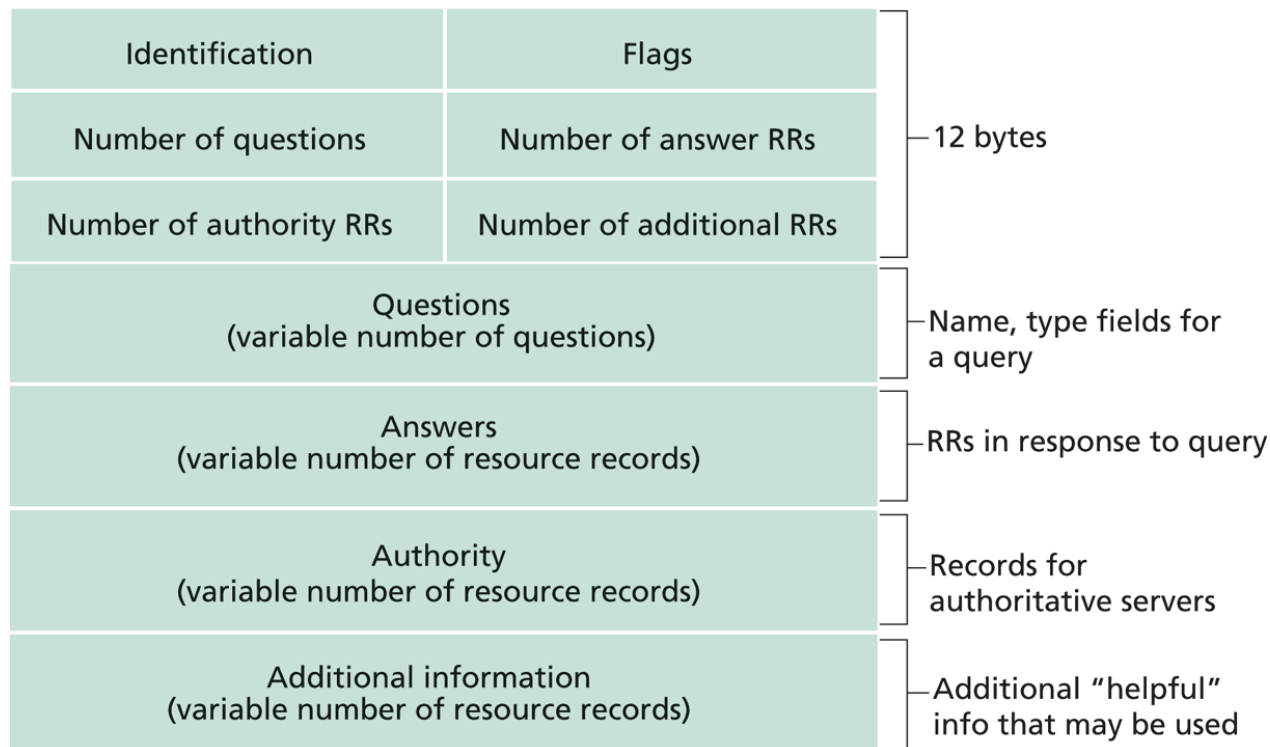
question

Answer:
offset 12

0000	08 00 27 ba b3 67 34 71 46 af 81 aa 08 00 45 00	..'.g4q F.....E.
0010	00 48 7f 0e 40 00 40 11 34 11 c0 a8 03 01 c0 a8	.H..@.@.4.....
0020	03 34 00 35 9b 8f 00 34 91 ee ed 00 81 80 00 01	.4.5...4.....
0030	00 01 00 00 00 00 03 78 6d 75 03 65 64 75 02 63x mu.edu.c
0040	6e 00 00 01 00 01 c0 0c 00 01 00 01 00 00 00 44	n.....D
0050	00 04 d2 22 00 23	...".#

Recap: DNS Protocol, Messages

Many features: typically over **UDP** (can use TCP); *query* and *reply* messages with the **same message format**; *length/content encoding of names*; simple **compression**; *additional info as server push*



What DNS did Right?

- ❑ Hierarchical delegation avoids central control, improving manageability and scalability
- ❑ Redundant servers improve robustness
 - see <http://www.internetnews.com/dev-news/article.php/1486981> for DDoS attack on root servers in Oct. 2002 (9 of the 13 root servers were crippled, but only slowed the network)
- ❑ Caching reduces workload and improves robustness
- ❑ Proactive answers reduce # queries on server and latency on client

Problems of DNS

- ❑ Simple query model, relatively static resource values and types make it harder to implement generic service discovery
 - e.g., service discovery of all printers
 - Although theoretically you can update the values of the records, it is rarely enabled
- ❑ Early binding (separation of DNS query from application query) does not work well in mobile, dynamic environments
 - e.g., load balancing, locate the nearest printer
- ❑ Each local domain needs servers, but an ad hoc domain may not have a DNS server

Outline

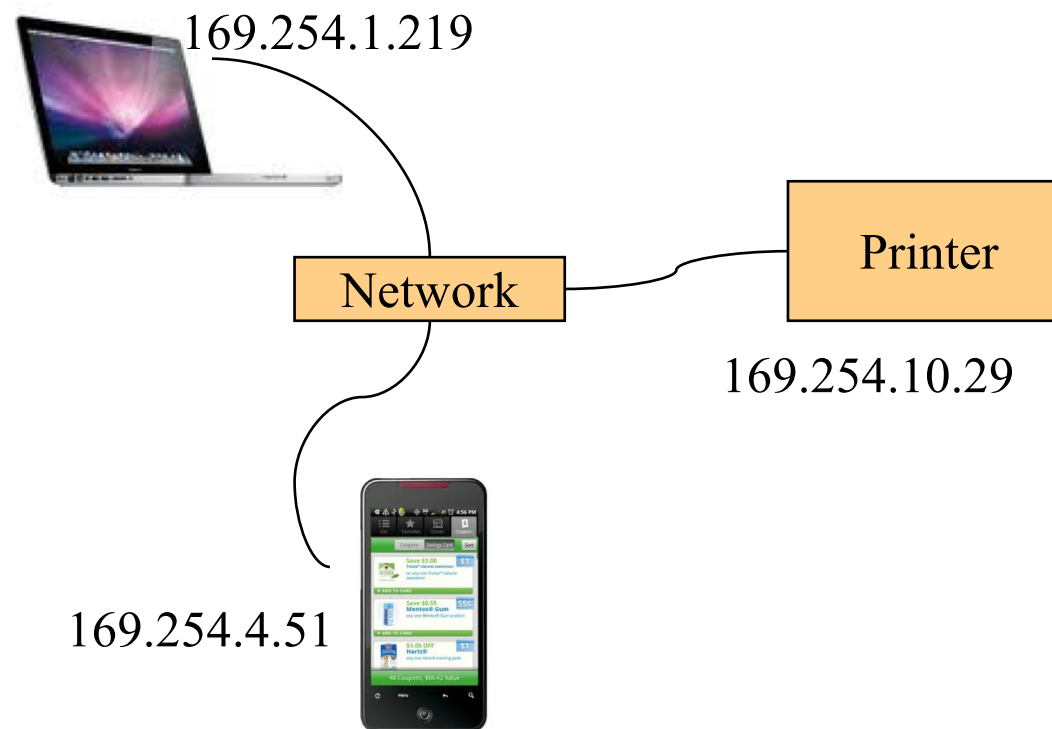
- ❑ Admin. and recap
- ❑ DNS
 - High-level design
 - Details
 - *Extensions/alternatives*

Discussions

- ❑ What extension(s) to standard DNS operations do we need to allow service discovery, say to implement Bonjour (discover all local printers)?
 - each printer needs to provide the following info: host, port, printer info (e.g., support postscript)

DNS-Service Discovery

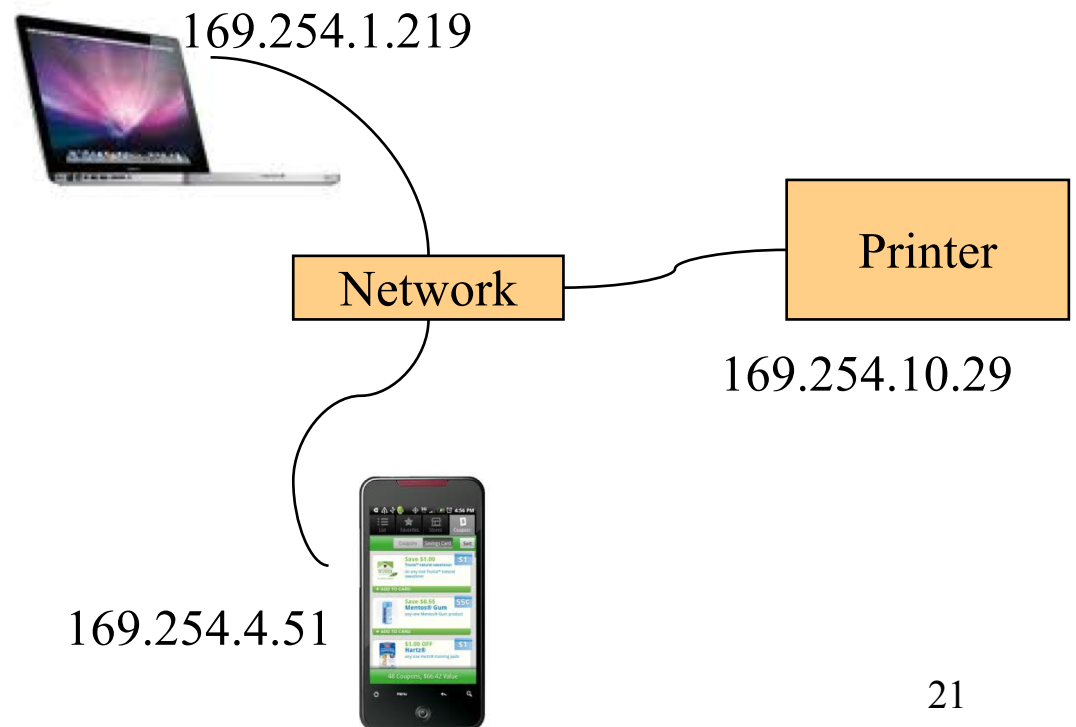
- ❑ Leverage DNS message format, but each node can announce its own services



Realizing DNS-SD without Central DNS Server: mDNS

❑ Multicast in a small world

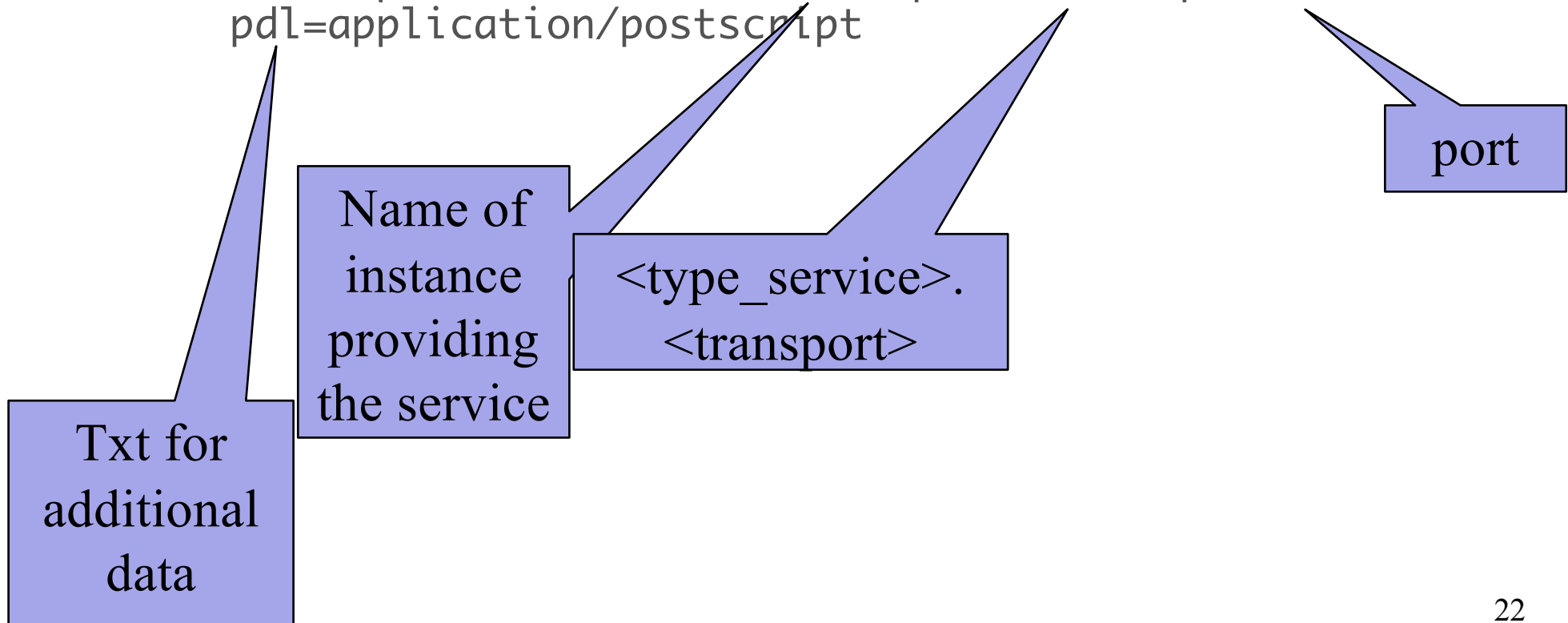
- no central address server
 - each node is a responder
- link-local addressing
 - send to **multicast**
address: **224.0.0.251**



Example

- ❑ Use the `avahi-publish-service` command on Ubuntu as example
 - Advertise (register) an LPR printer on port 515

```
avahi-publish-service test _printer._tcp . 515  
pdl=application/postscript
```



Example

- ❑ Use the dns-sd command on Mac as example
 - Advertise (register) an LPR printer on port 515

```
dns-sd -R "test" _printer._tcp . 515  
pdl=application/postscript
```

Name of
instance
providing
the service

<type_service>.
<transport>

domain (. means
default, which is
local

port

Txt for
additional
data

1	0.000...	fe80::5052:5af:92fa...	ff02::fb	MDNS	179	Standard	query	0x0000	ANY	test._printer._tcp.local, "QM" question SRV 0 0 515 qiao-VirtualBox.l...
2	0.000...	192.168.3.52	224.0.0.251	MDNS	159	Standard	query	0x0000	ANY	test._printer._tcp.local, "QM" question SRV 0 0 515 qiao-VirtualBox.l...
3	0.255...	fe80::5052:5af:92fa...	ff02::fb	MDNS	179	Standard	query	0x0000	ANY	test._printer._tcp.local, "QM" question SRV 0 0 515 qiao-VirtualBox.l...
4	0.255...	192.168.3.52	224.0.0.251	MDNS	159	Standard	query	0x0000	ANY	test._printer._tcp.local, "QM" question SRV 0 0 515 qiao-VirtualBox.l...
5	0.501...	fe80::5052:5af:92fa...	ff02::fb	MDNS	179	Standard	query	0x0000	ANY	test._printer._tcp.local, "QM" question SRV 0 0 515 qiao-VirtualBox.l...
6	0.501...	192.168.3.52	224.0.0.251	MDNS	159	Standard	query	0x0000	ANY	test._printer._tcp.local, "QM" question SRV 0 0 515 qiao-VirtualBox.l...
7	0.701...	192.168.3.52	224.0.0.251	MDNS	248	Standard	query	response	0x0000	TXT, cache flush PTR test._printer._tcp.local SRV, cache flush 0...
8	0.701...	fe80::5052:5af:92fa...	ff02::fb	MDNS	252	Standard	query	response	0x0000	TXT, cache flush PTR test._printer._tcp.local SRV, cache flush 0...
9	1.919...	192.168.3.52	224.0.0.251	MDNS	248	Standard	query	response	0x0000	TXT, cache flush PTR test._printer._tcp.local SRV, cache flush 0...
10	1.919...	fe80::5052:5af:92fa...	ff02::fb	MDNS	252	Standard	query	response	0x0000	TXT, cache flush PTR test._printer._tcp.local SRV, cache flush 0...
11	4.127...	192.168.3.52	224.0.0.251	MDNS	248	Standard	query	response	0x0000	TXT, cache flush PTR test._printer._tcp.local SRV, cache flush 0...
12	4.127...	fe80::5052:5af:92fa...	ff02::fb	MDNS	252	Standard	query	response	0x0000	TXT, cache flush PTR test._printer._tcp.local SRV, cache flush 0...
13	11.73...	fe80::5052:5af:92fa...	ff02::fb	MDNS	183	Standard	query	0x0000	PTR	_services._dns-sd._udp.local, "QM" question PTR _companion-link._tcp...
14	11.73...	192.168.3.52	224.0.0.251	MDNS	305	Standard	query	0x0000	PTR	_services._dns-sd._udp.local, "QM" question PTR _companion-link._tcp...
15	11.83...	192.168.3.38	224.0.0.251	MDNS	230	Standard	query	response	0x0000	PTR _companion-link._tcp.local SRV, cache flush 0 0 54580 Qiaos...
16	12.71...	fe80::5052:5af:92fa...	ff02::fb	MDNS	183	Standard	query	0x0000	PTR	_services._dns-sd._udp.local, "QM" question PTR _companion-link._tcp...
17	12.71...	192.168.3.52	224.0.0.251	MDNS	212	Standard	query	0x0000	PTR	_services._dns-sd._udp.local, "QM" question PTR _companion-link._tcp...
18	14.72...	fe80::5052:5af:92fa...	ff02::fb	MDNS	183	Standard	query	0x0000	PTR	_services._dns-sd._udp.local, "QM" question PTR _companion-link._tcp...
19	14.72...	192.168.3.52	224.0.0.251	MDNS	212	Standard	query	0x0000	PTR	_services._dns-sd._udp.local, "QM" question PTR _companion-link._tcp...

Offline Exercise

- ❑ Use the `dns-sd /avahi-publish-service` command as example
 - Advertise (register) a web page on local machine

```
dns-sd -R "My Test" _http._tcp . 80  
path=/path-to-page.html
```


Issue: How to Query

- ❑ Query needs a back pointer, PTR records
- ❑ Exercise: Use the `dns-sd` / `avahi-service-publish` command as example

- Browse web pages on local machines

```
dns-sd -B _http._tcp
```

```
avahi-browse -rt _http._tcp
```

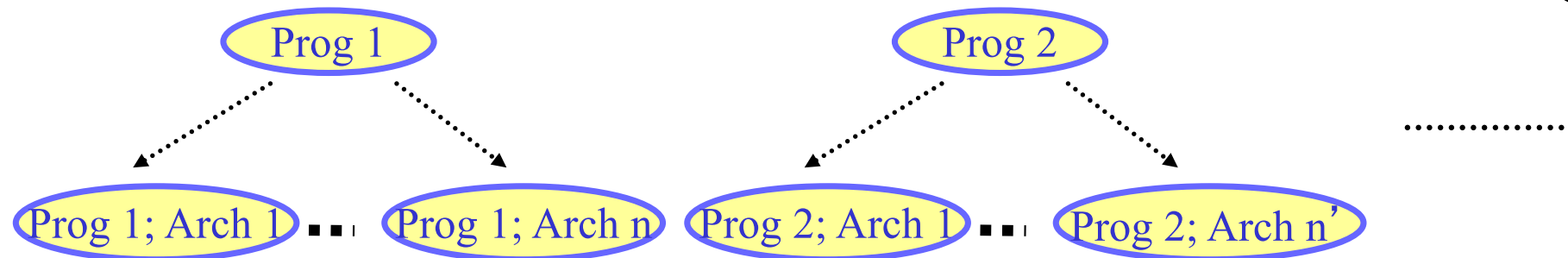
Network Service Discovery in Android

- ❑ Based on DNS-SD/mDNS
- ❑ Foundation for peer-to-peer/Wi-Fi Direct in Android
- ❑ See <https://developer.android.com/training/connect-devices-wirelessly/nsd.html> for programming using nsd

Outline

- ❑ Admin. and recap
- ❑ Basic network applications
 - Email
 - DNS
- *Java in a Nutshell*

High-level Picture



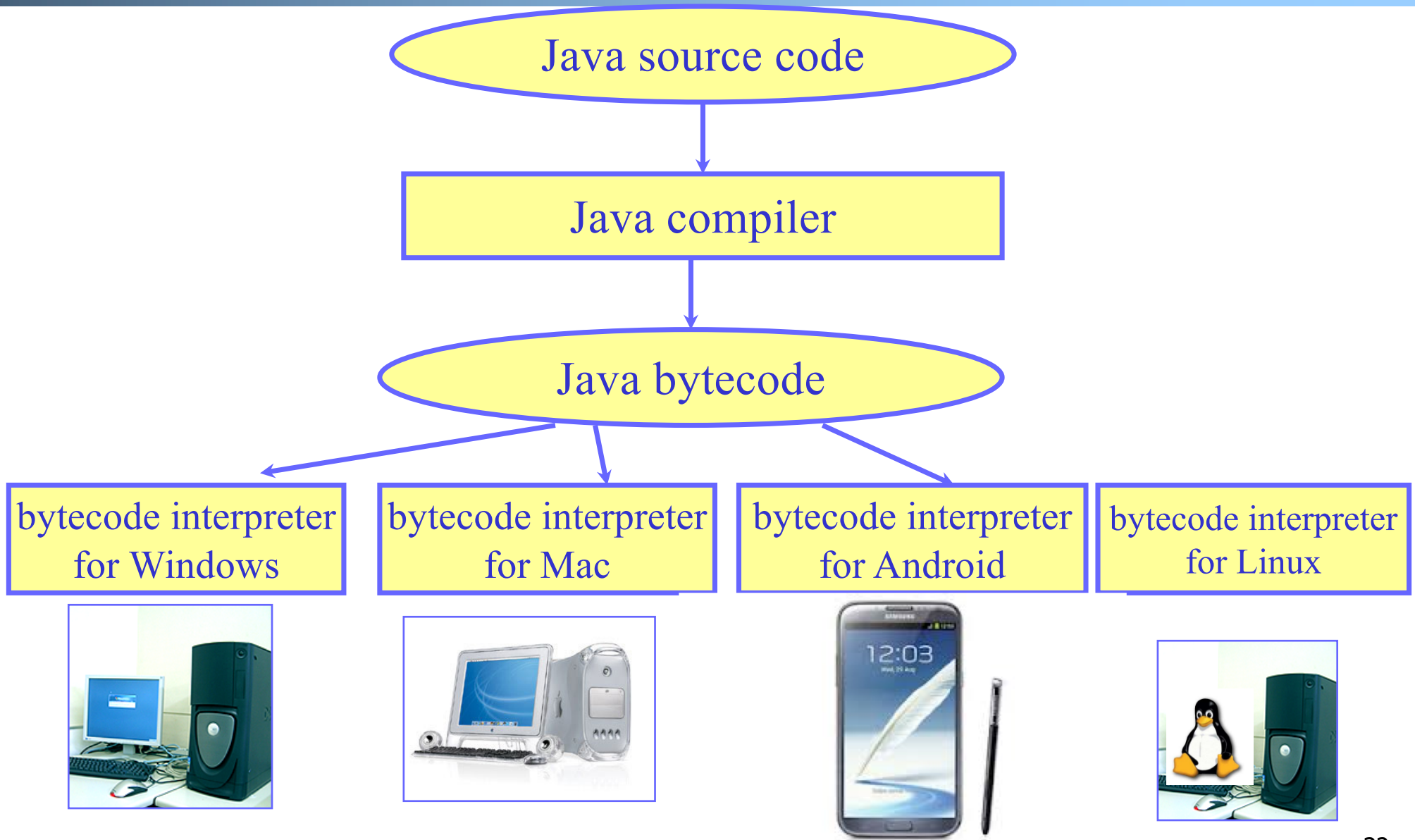
Java Virtual Machine

- ❑ To be platform independent, Java designers introduced Java Virtual Machine (JVM), a machine different from any physical platform, but a *virtual machine*
 - The language of the virtual machine is referred to as *bytecode*
 - Thus Java actually has two programming languages
- ❑ A Java compiler translates Java source code (.java files) into *bytecode* (in .class files)
 - Each Java software program needs to be compiled only once: from the Java source code to bytecode
- ❑ Other languages (e.g., Jruby, Jython, Scala) may also compile to bytecode

Java Execution

- ❑ To execute a Java program, another piece of software called an *interpreter*, translates between bytecode and the actual machine
 - an interpreter is specific to a specific platform
 - the interpreter understands java bytecode, and then issues instructions in the specific platform for which it is written
 - we also say that an interpreter provides a java virtual machine (JVM)

Java Translation and Execution



Comparing Traditional (e.g., C/C++) and Java Software Development

Traditional, e.g., C/C++

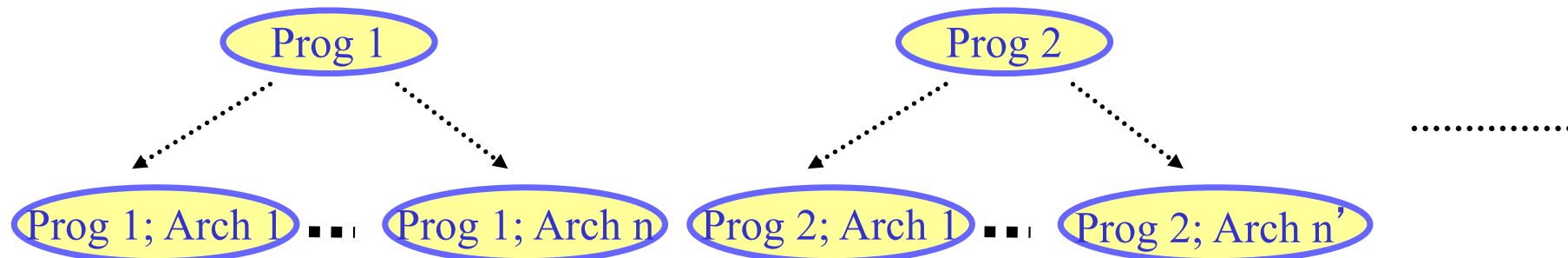
- ❑ A developer writes a program in C/C++
- ❑ The C/C++ source code is generally considered proprietary, and not released
- ❑ The developer compiles the C/C++ program for each platform it intends to support, and distributes one version for each platform
 - thus each program has **multiple compiled** versions
 - each compiled version can run by itself
- ❑ **Platform dependency handled by each software developer**

Java

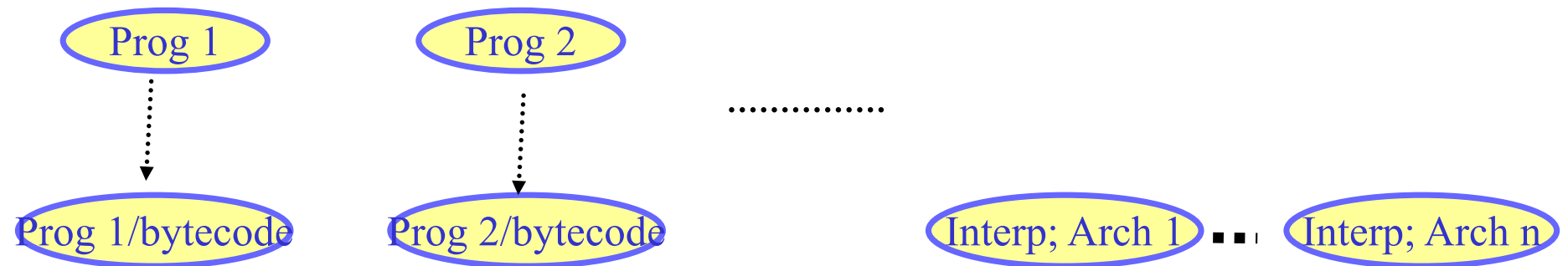
- ❑ A developer writes a program in Java
- ❑ The Java source code is generally considered proprietary, and not released
- ❑ The developer compiles the Java program to bytecode, and distributes the bytecode version
 - thus each program has only **one compiled version**
 - the compiled bytecode needs an interpreter for each platform
- ❑ **Platform dependency handled by platform vendor**

High-level Picture

C/C++

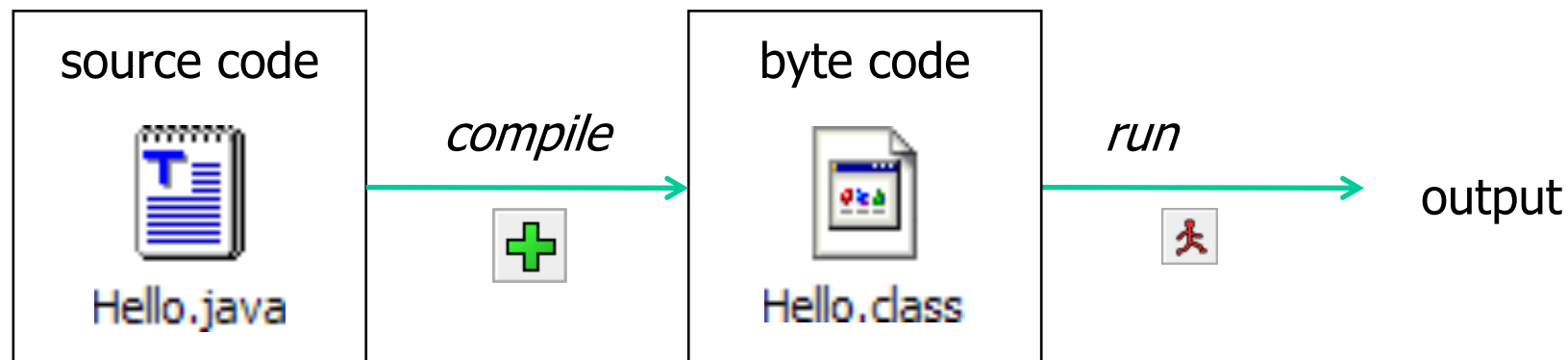


Java



Recall: Java Programming Steps

- ❑ Programming in Java consists of 3 simple steps
 - Create and edit "Java source code" (.java files)
 - Compile into "Java bytecode" (.class files)
 - Execute bytecode with a "Java interpreter"



Programming in Java (Step 1): Create/Edit

- ❑ The basic way is to use a text editor
 - Example editors: vim, sublime, Notepad, TextEdit (Format/Make Plain Text) etc.
 - **Note: MS Word is NOT a text editor**
 - The key is that your .java file *cannot* include any markup or stylistic formatting; just text.
 - You enter your Java code following Java Language syntax (more soon).

Programming in Java (Step 2): Compile

- ❑ Compile a Java program

\$ javac HelloWorld.java

- ❑ Take a look to see that HelloWorld.class is generated

\$ ls

HelloWorld.java HelloWorld.class

Programming in Java (Step 3): Execute

- ❑ Run Java interpreter

\$ java HelloWorld

First Java Program

```
/* *****  
 * Prints "Hello World"  
 * Everyone's first Java program.  
 ***** */  
public class Hello {  
    public static void main(String[] args) {  
        System.out.println("Hello, world!");  
    }  
}
```

Another Java Program

```
public class Hello2 {  
    public static void main(String[] args) {  
        System.out.println("Hello, world!");  
        System.out.println();  
        System.out.println("This program produces");  
        System.out.println("four lines of output");  
    }  
}
```

Programming in Java: Method 2

- ❑ Another way is to use an Integrated Development Environment (IDE)
 - Example IDEs: Eclipse, IDEA, DrJava, etc.
 - An IDE usually presents the user with a space for text (like an editor) but layers *additional features* on top of the text for the user's benefit.
 - **Note: The underlying file contains pure text, just like a text editor.**
 - These features can be very useful and save time.
 - Example features are GUI compile, GUI execution, code completion, and syntax highlighting.
 - IDEs take more time to get started than a simple text editor, e.g.,
 - set up where to find the "java" and "javac" programs
 - find out where does the IDE save my file

Java Syntax Structure: A Top-Down View

A class:

- has a name, defined in a **file with same name**
Convention we follow: capitalize each English word
- starts with {, and ends with }
- includes a group of methods

```
public class <class name> {  
    public static void main(String[] args) {
```

```
        <statement>;
```

```
        <statement>;
```

```
        ...
```

```
        <statement>;
```

```
    }
```

```
}
```

A method:

- has a name
Convention we follow: lowercase first word, capital following
- starts with {, and ends with }
- includes a group of statements

statement:

- a command to be executed
- end with ;

The `System.out.println` statement

- ❑ A statement that prints a line of output on the console.
 - pronounced "print-linn"
- ❑ Two ways to use `System.out.println` :
 - `System.out.println(<string>);`
Prints the given message `<string>` as output.
 - `System.out.println();`
Prints a blank line of output.

Java program structure

- ❑ A top-down view
- ❑ A bottom-up view

Java Syntax: A Bottom-Up View

```
// Comment 1: A Java program
/* Comment 2: a long comment
*****/
public class Hello {
    public static void main(String[] args) {
        System.out.println("Hello, world!");
        System.out.println();
        System.out.println("This program produces");
        System.out.println("four lines of output");
    }
}
```

Java Syntax: A Bottom-Up View

```
// Comment 1: A Java program
/* Comment 2: a long comment
*****/
public class Hello {
    public static void main(String[] args) {
        System.out.println("Hello, world!");
        System.out.println();
        System.out.println("This program produces");
        System.out.println("four lines of output");
    }
}
```

- ❑ Basic Java syntax units
 - white space and comments
 - identifiers (words)
 - symbols: { } " () < > [] ; = ...
 - strings
 - numbers

Syntax: White Space

- ❑ White space
 - includes spaces, new line characters, tabs
 - white space is used to separate other entities
 - extra white space is ignored

- ❑ White space allows a Java program to be formatted in many ways, and should be formatted to enhance readability
 - the usage of white space forms part of programming style

Syntax: Comments

- ❑ **comment:** A note written in source code by the programmer to describe or clarify the code.
 - Comments are ignored by the compiler
 - Useful for other people (and yourself!) to understand your code

- ❑ **Two types of comments in Java**
 - single-line comments use `//...`
`// this comment runs to the end of the line`
 - multi-lines comments use `/* ... */`
`/* this is a very long
multi-line comment */`

Syntax: Identifier

- ❑ **Identifier:** A name given to an item in a program.

- ❑ **Syntax requirement on identifier:**
 - must start with a letter or `_` or `$`
 - subsequent characters can be any of those or a number
 - Important: Java is case sensitive:
 - Hello and hello are different identifiers

Three Types of Identifiers

1. Identifiers chosen by ourselves when writing a program (such as `HelloWorld`)
2. Identifiers chosen by another programmer, so we use the identifiers that they chose (e.g., `System`, `out`, `println`, `main`)

```
public class HelloWorld
{
    public static void main(String[] args)
    {
        System.out.println("Hello World!");
    }
}
```

Three Types of Identifiers

3. Special identifiers called *keywords* or *reserved words*:
A keyword has a special meaning in Java.

abstract	default	if	private	this
boolean	do	implements	protected	throw
break	double	import	public	throws
byte	else	instanceof	return	transient
case	extends	int	short	try
catch	final	interface	static	void
char	finally	long	strictfp	volatile
class	float	native	super	while
const	for	new	switch	
continue	goto	package	synchronized	

Java reserved words: they are all **lowercase!**

Syntax: Strings

- ❑ **string:** A sequence of characters that starts and ends with a " (quotation mark character).

- The quotes do not appear in the output.

- Examples:

- `"hello"`

- `"This is a string. It is very long!"`

- ❑ **Restrictions:**

- May not span multiple lines

- `"This is not
a legal String."`

Examples

❑ Which of the following are legal strings in Java?

- `"This is a string. It's very long!"`
- `"This cool string spans
two lines. "`
- `"It is a great thing when children cry, "I
want my mommy"! "`

Escape Sequences

- **escape sequence:** A special sequence of characters used to represent certain special characters in a string.

<code>\b</code>	backspace
<code>\t</code>	tab character
<code>\n</code>	new line character
<code>\"</code>	quotation mark character
<code>\\</code>	backslash character

- **Example:**

```
System.out.println("\\hello\nhow\tare \"you\"?\\\\\");
```

- **Output:**

```
\\hello
how      are "you"?\\
```

Comment on syntax errors

- ❑ **A syntax/compile error:** A problem in the structure of a program that causes the compiler to fail, e.g.,
 - Missing semicolon
 - Too many or too few { } braces
 - Class and file names do not match
 - ...
- ❑ Compilers can't (DO not) read minds.
- ❑ Compilers don't make mistakes.
- ❑ If the program is not doing what you want, do NOT blame the computer---it's **YOU** who made a mistake.

Outline

- ❑ Admin. and recap
- ❑ Basic network applications
 - Email
 - DNS
- ❑ Java in a Nutshell
- ❑ *Network application programming*

Socket Programming

Socket API

- ❑ introduced in BSD4.1 UNIX, 1981
- ❑ Two types of sockets
 - connectionless (UDP)
 - connection-oriented (TCP)

socket

an interface (a “door”) into which one application process can **both send and receive** messages to/from another (remote or local) application process

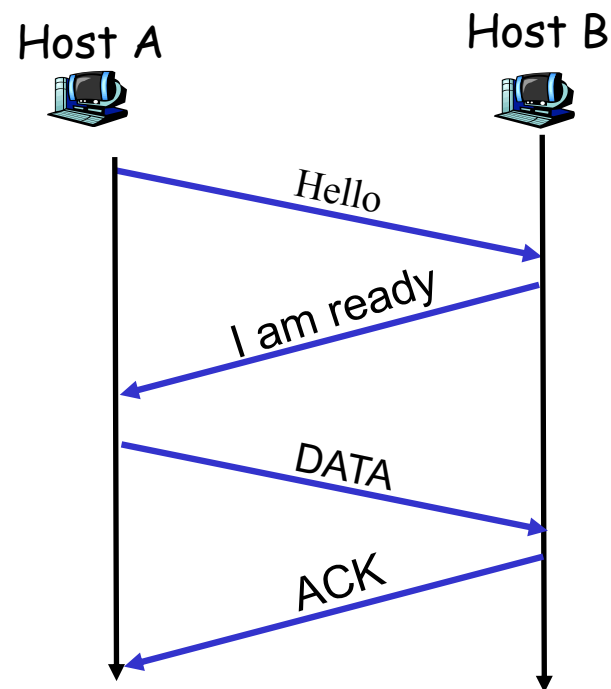
Services Provided by Transport

□ User data protocol (UDP)

- multiplexing/demultiplexing

□ Transmission control protocol (TCP)

- multiplexing/demultiplexing
- reliable data transfer
- rate control: flow control and congestion control



Big Picture: Socket

