Application Layer Protocols

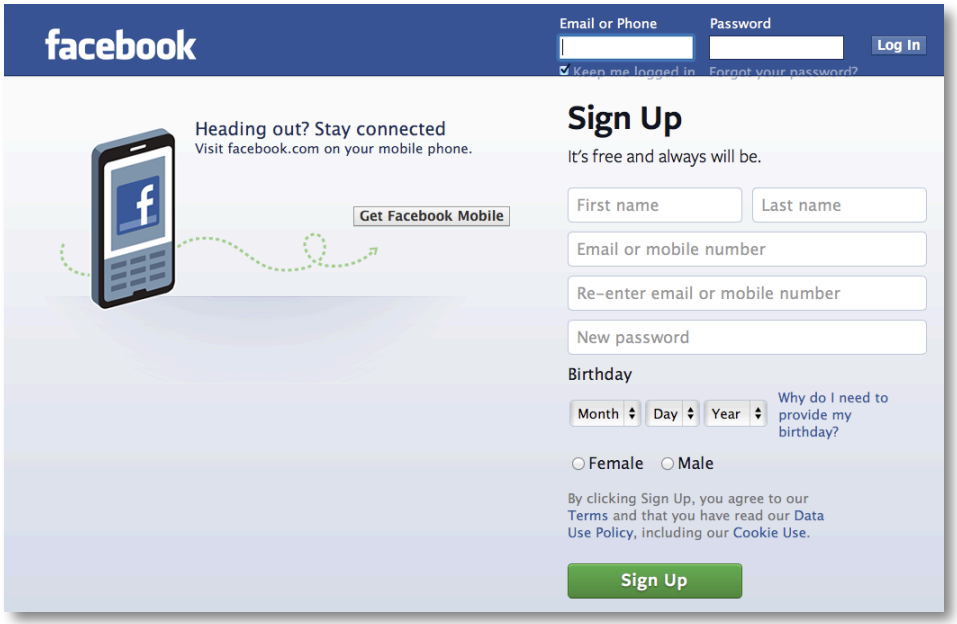
Peerapon S.

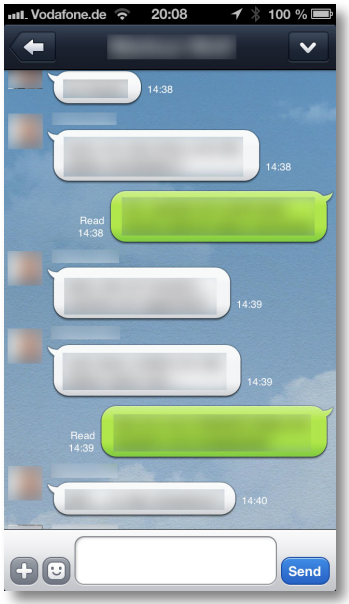
CPE 314 Computer Networks (2/61)

Topics

Network applications

Computer processes exchaning data to achieve certain goals.

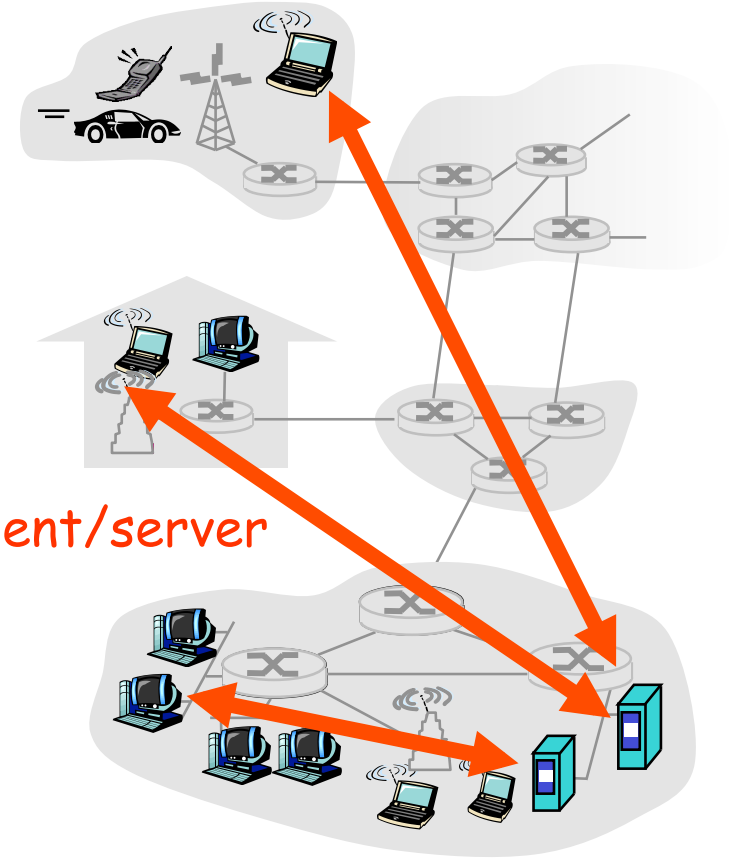




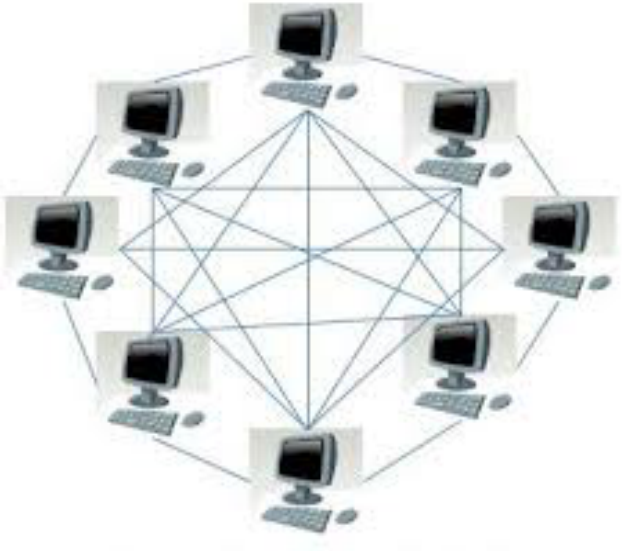
What are the differences underneath?

- Type “www.facebook.com” in the browser and hit Enter ?
- Type a message in Line and hit Send ?

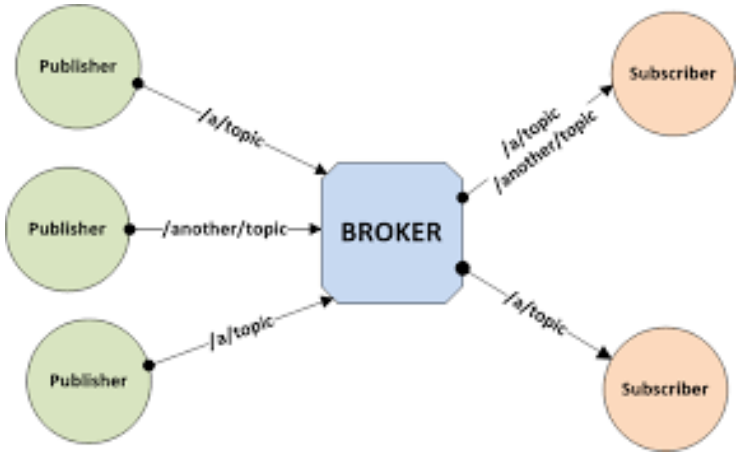
Network Application Models



client/server



peer-to-peer



Broker-based

Self-Test

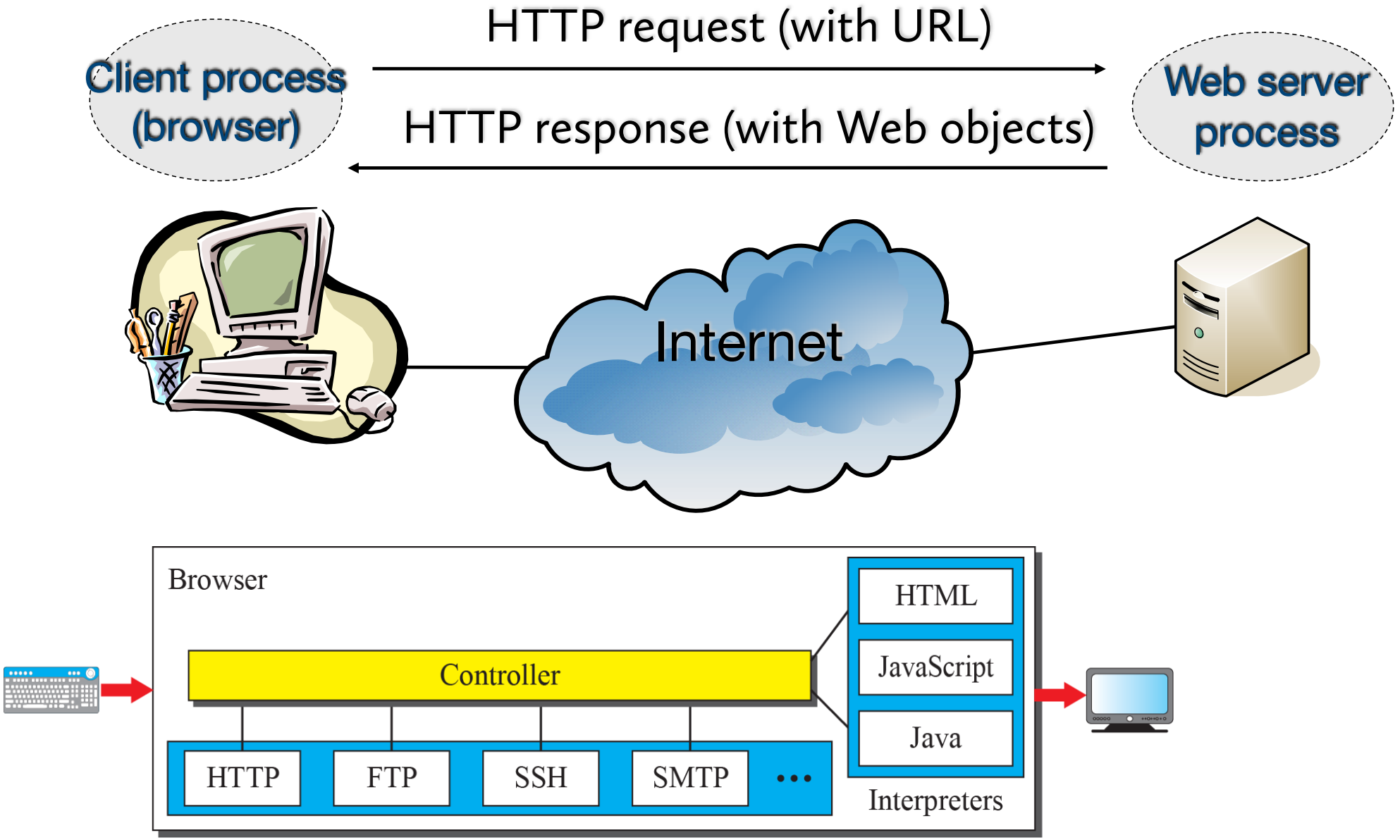
- What are resources/services provided by the following network applications ?
  - WWW
  - Email
  - Line
  - Facebook

Uniform Resource Locator (URL):

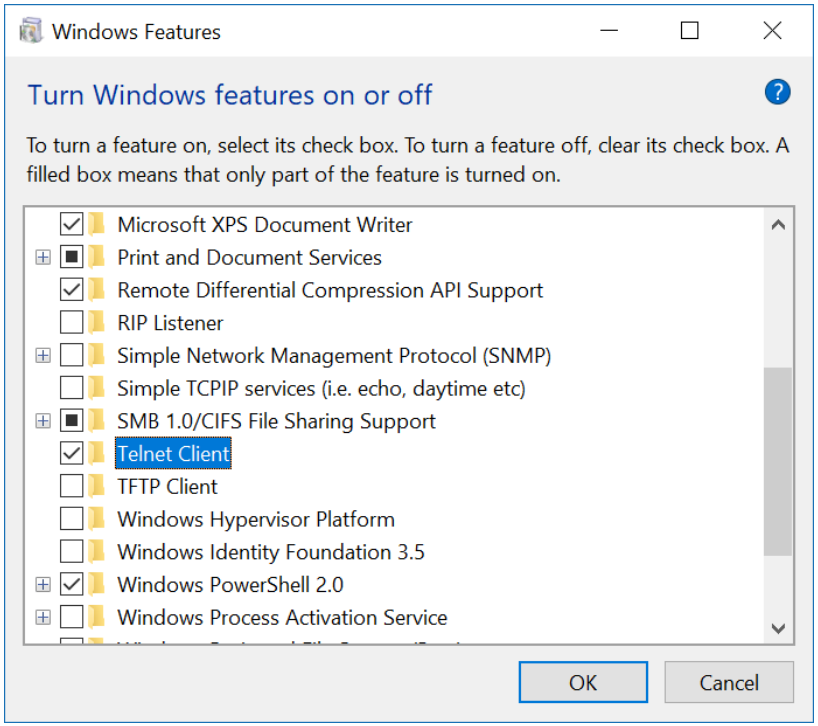
www2.kmutt.ac.th/thai/CUR_STU/index.html

HTML/JS Contents and objects addressable via Uniform Resource Locator (URL)

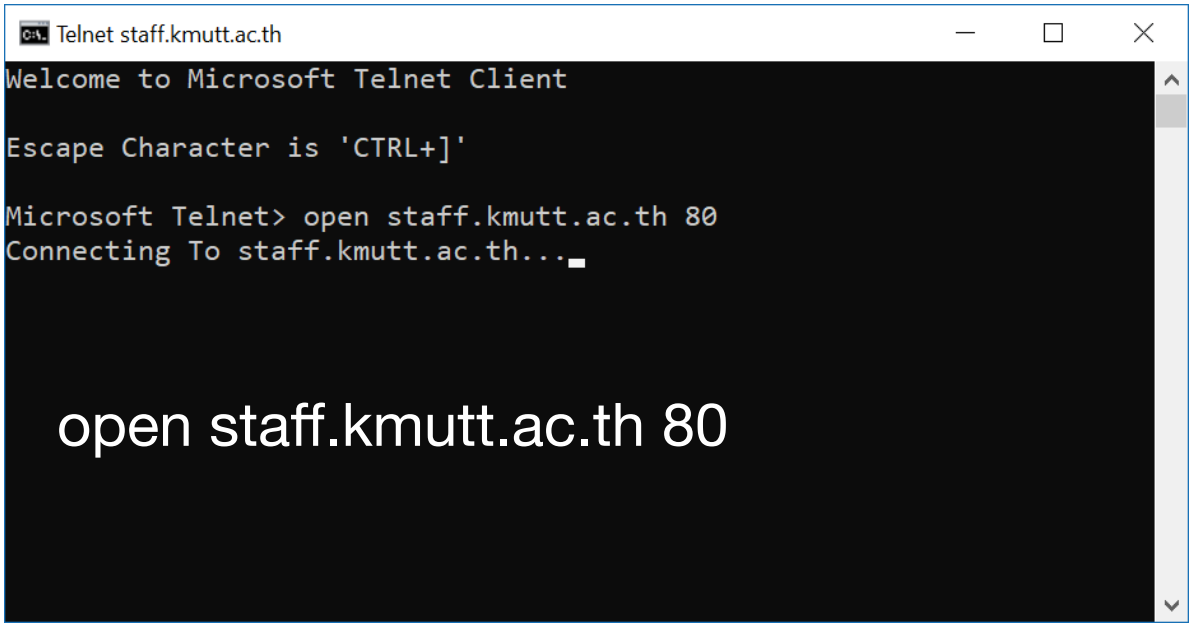
Hyper Text Transfer Protocol (HTTP)

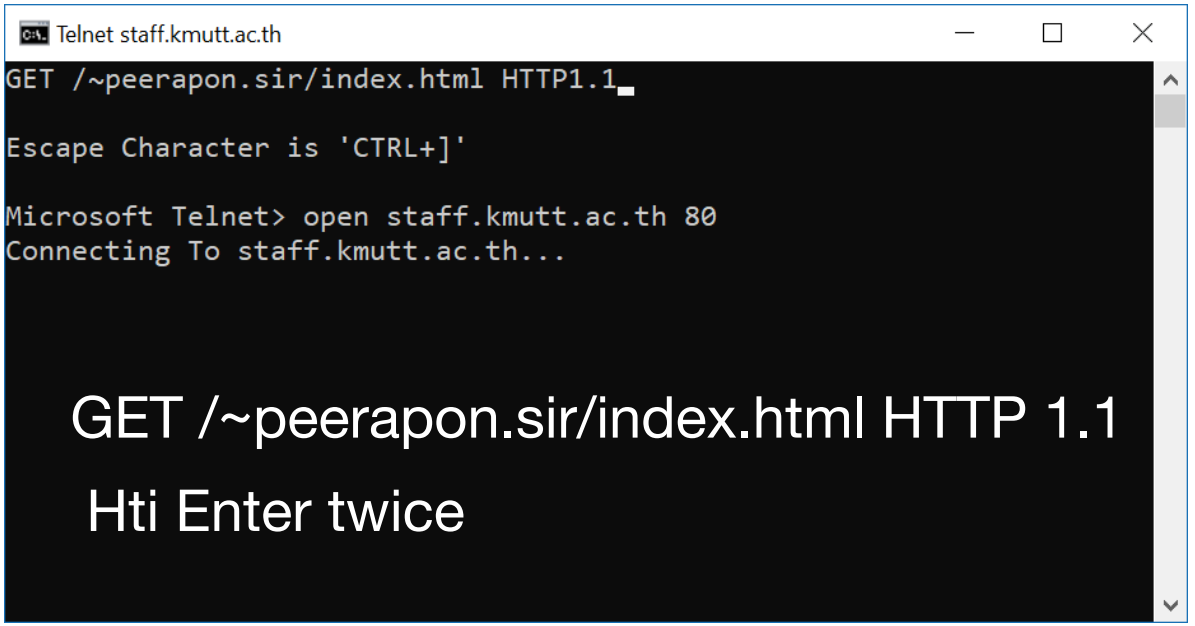


Trying Out HTTP Manually (MS Windows Client)



Enable Telnet client





Trying out HTTP Manually (MacOS Client)

- At a terminal command prompt

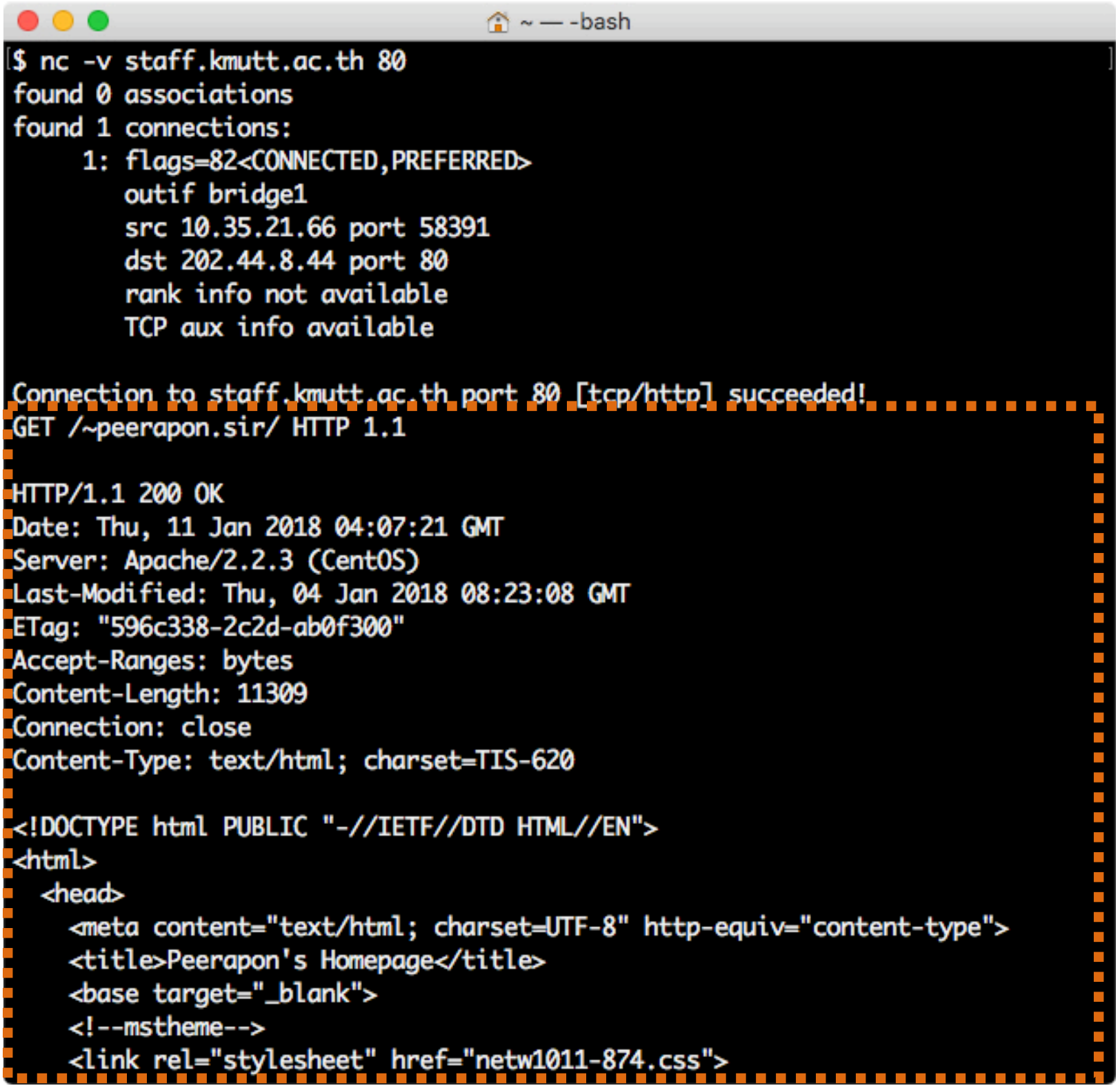
```
$ nc -v staff.kmutt.ac.th 80
```

- After the connection is established

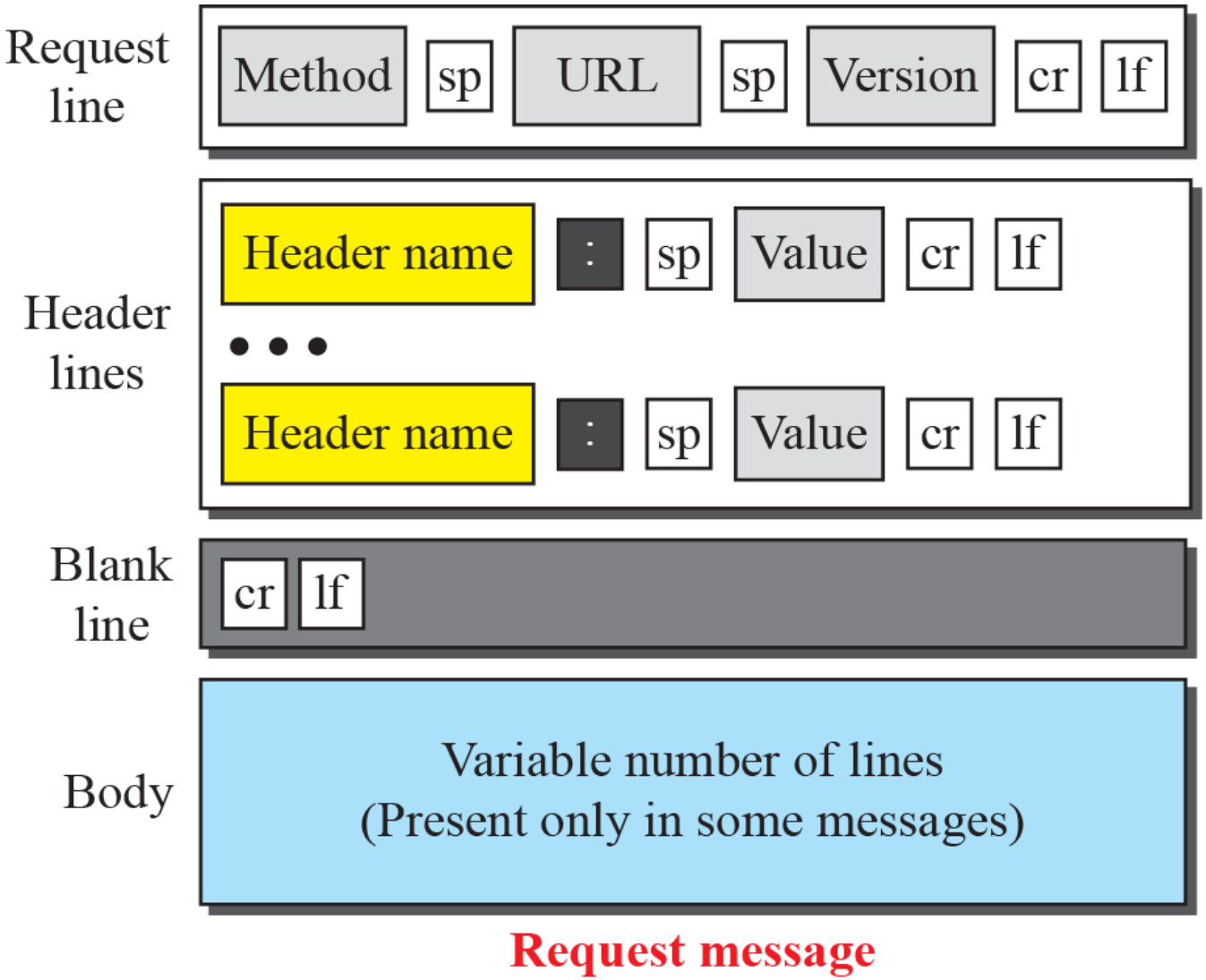
```
GET /~peerapon.sir/ HTTP 1.1
```

Hit Enter Twice
and wait a few seconds

- What information do you see in the response ?


~ — -bash
$ nc -v staff.kmutt.ac.th 80
found 0 associations
found 1 connections:
1: flags=82<CONNECTED,PREFERRED>
outif bridge1
src 10.35.21.66 port 58391
dst 202.44.8.44 port 80
rank info not available
TCP aux info available
Connection to staff.kmutt.ac.th port 80 [tcp/http] succeeded!
GET /~peerapon.sir/ HTTP 1.1
HTTP/1.1 200 OK
Date: Thu, 11 Jan 2018 04:07:21 GMT
Server: Apache/2.2.3 (CentOS)
Last-Modified: Thu, 04 Jan 2018 08:23:08 GMT
ETag: "596c338-2c2d-ab0f300"
Accept-Ranges: bytes
Content-Length: 11309
Connection: close
Content-Type: text/html; charset=TIS-620
<!DOCTYPE html PUBLIC "-//IETF//DTD HTML//EN">
<html>
<head>
<meta content="text/html; charset=UTF-8" http-equiv="content-type">
<title>Peerapon's Homepage</title>
<base target="_blank">
<!--mstheme-->
<link rel="stylesheet" href="netw1011-874.css">

HTTP Request Message



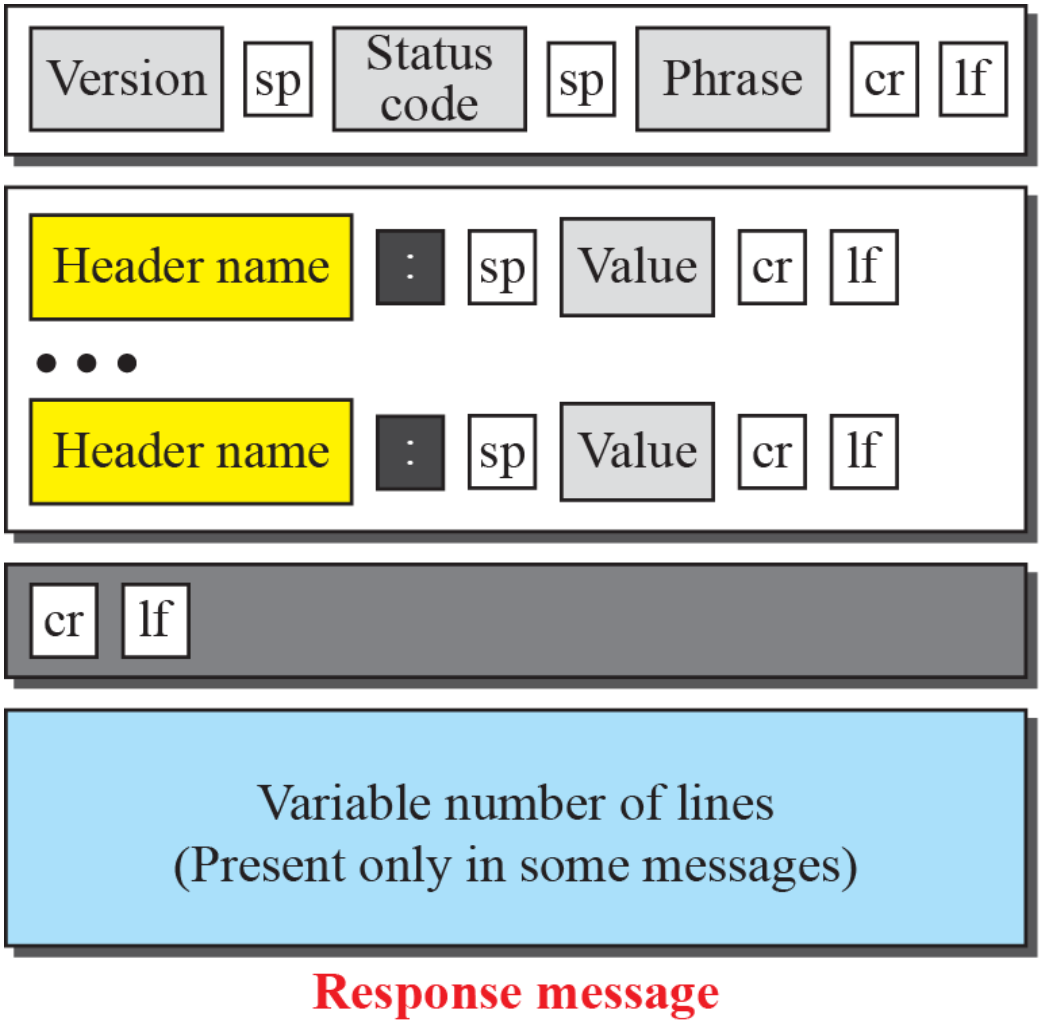
Request message

GET and POST methods

Method	*Action*
GET	Requests a document from the server
HEAD	Requests information about a document but not the document itself
PUT	Sends a document from the client to the server
POST	Sends some information from the client to the server
TRACE	Echoes the incoming request
DELETE	Removes the web page
CONNECT	Reserved
OPTIONS	Inquires about available options

```
from urllib.request import urlopen

u = urlopen("http://staff.kmutt.ac.th/~peerapon.sir/")

data = u.read()

print(data)
```

HTTP Response Message




```
GET /~peerapon.sir/ HTTP 1.1

HTTP/1.1 200 OK
Date: Wed, 24 Dec 2014 05:58:38 GMT
Server: Apache/2.2.3 (CentOS)
Last-Modified: Tue, 14 Oct 2014 03:23:27 G
ETag: "596c338-2c2c-89e0e9c0"
Accept-Ranges: bytes
Content-Length: 11308
Connection: close
Content-Type: text/html; charset=TIS-620
```

Ex: HTTP Response Status Codes

200 OK

- Request succeeded, requested object later in this message

301 Moved Permanently

- Requested object moved, new location specified later in this message (Location:)

400 Bad Request

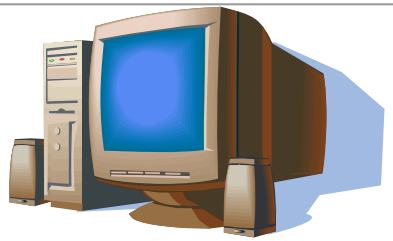
- Request message not understood by server

404 Not Found

- Requested document not found on this server

505 HTTP Version Not Supported

Cookies



client

server

Cookie file

ebay: 8734

usual http request msg

usual http response + Set-cookie: 1678

server creates ID 1678 for user

entry in backend database

Cookie file

amazon: 1678
ebay: 8734

usual http request msg cookie: 1678

usual http response msg

cookie-specific action

access

one week later:

Cookie file

amazon: 1678
ebay: 8734

usual http request msg cookie: 1678

usual http response msg

cookie-spectific action

access

- Cookies enables HTTP from being **stateless** to **stateful**.
- What cookies can bring:
  - Authorization
  - Shopping carts
  - Recommendations
  - User session state (Web e-mail)

Cookies and privacy:

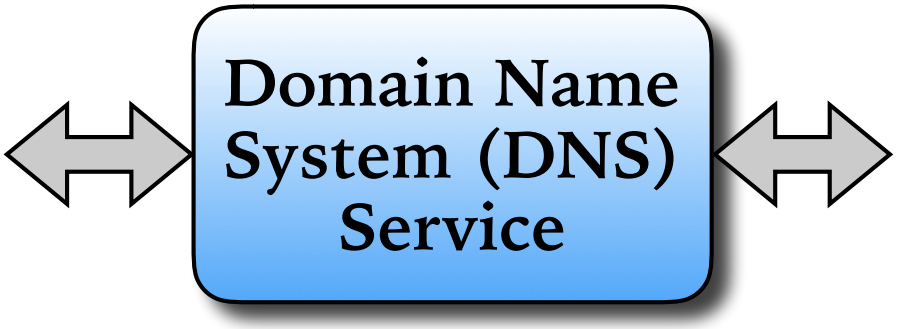
- Cookies permit sites to learn a lot about you
- You may supply name and e-mail to sites
- Search engines use redirection & cookies to learn yet more
- Advertising companies obtain info across sites

Design Lessons Learned

- Stateless protocol operations
- Text-based syntax
- TCP for reliable transfer
- Session ID if states needed

Domain Name

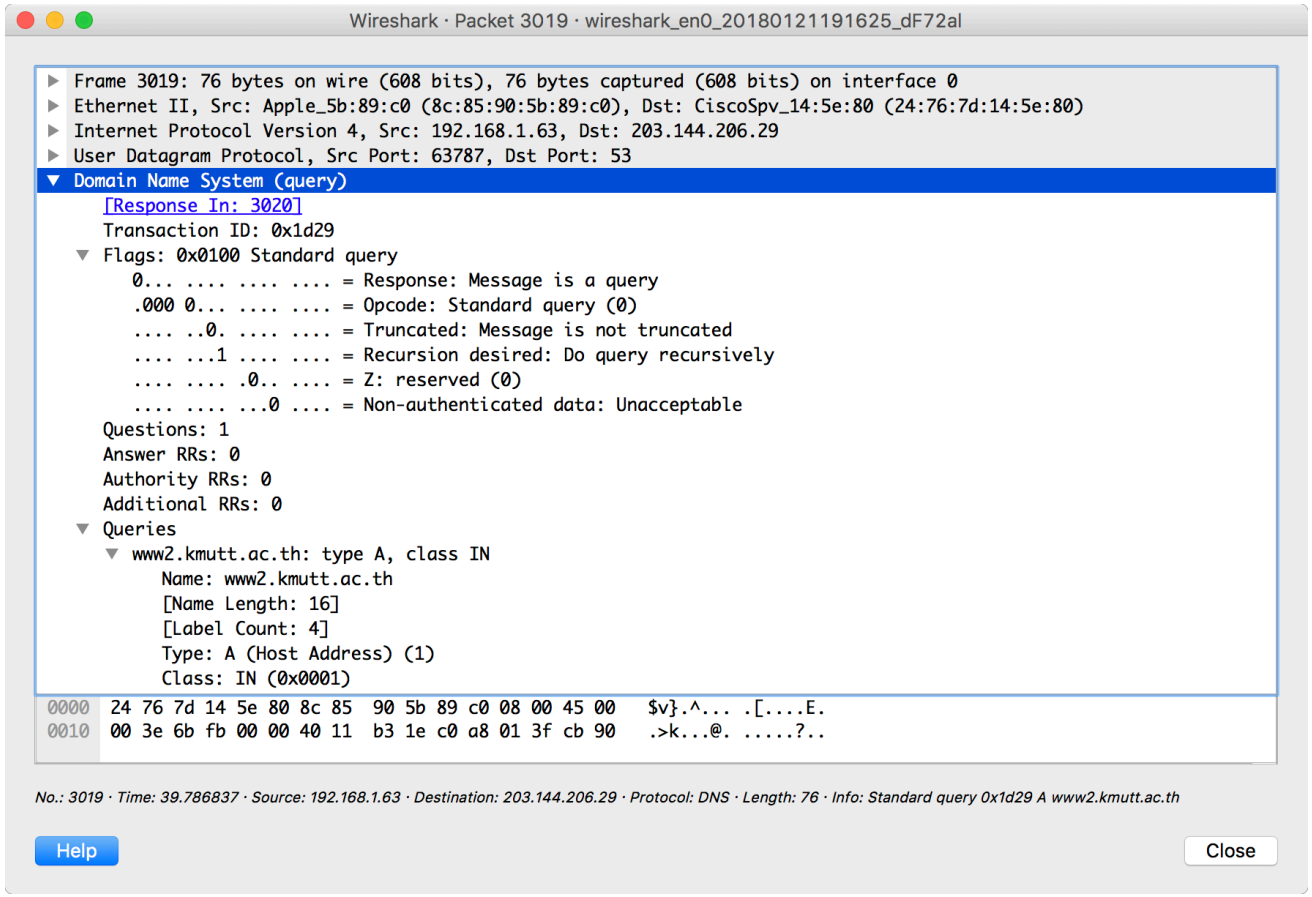
www.google.co.th
www.facebook.com
www.kmutt.ac.th
www2.kmutt.ac.th
540141@st.kmutt.ac.th
peerapon@gmail.com



64.233.81.19
157.240.10.35
202.44.88.5

gethostbyname()
nslookup
whois

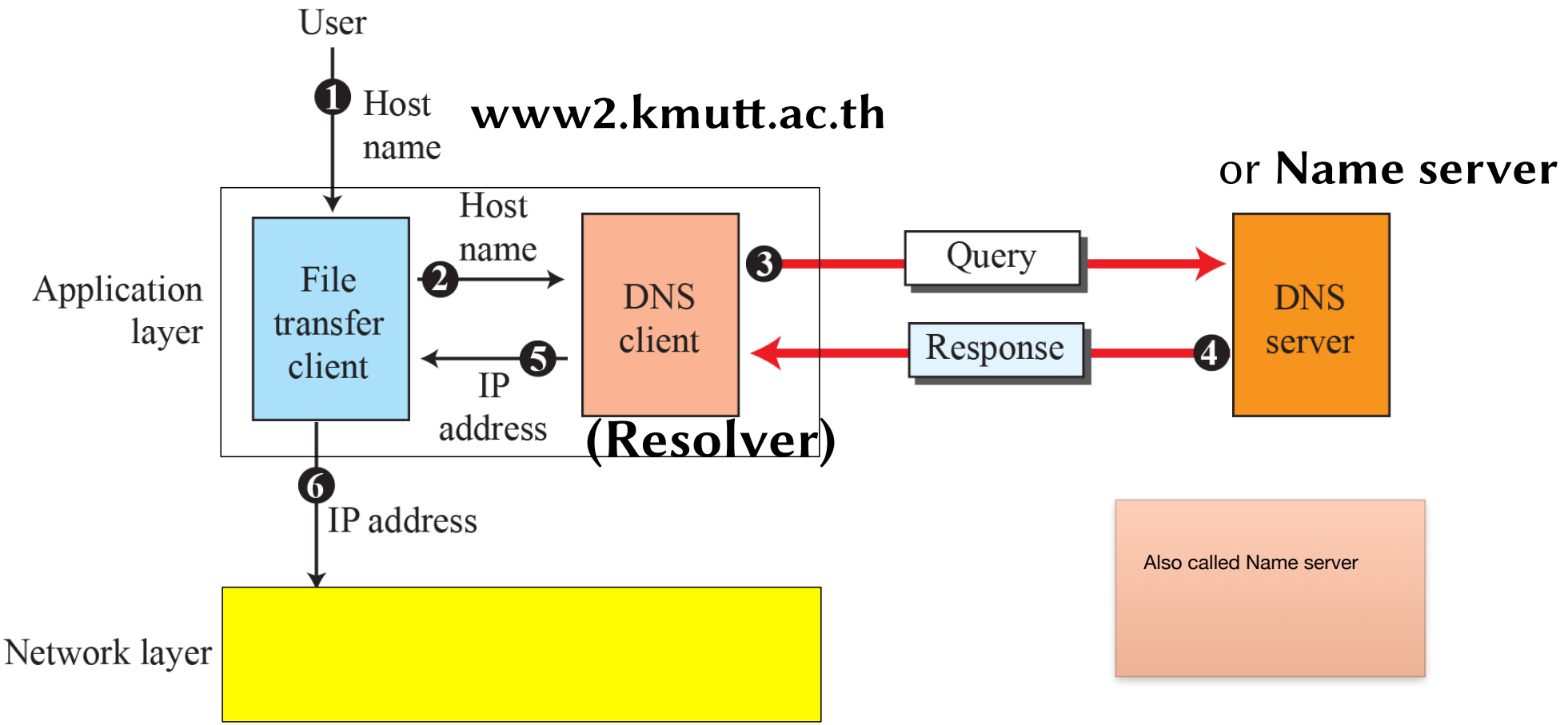
- ☐ What does the command do?
- ☐ What messages have been exchanged?
- ☐ What are the source and destination of the messages?




```
peerapon@perseus ~
$ nslookup  www2.kmutt.ac.th
Server:         203.144.206.29
Address:        203.144.206.29#53

Non-authoritative answer:
Name:   www2.kmutt.ac.th
Address: 202.44.8.55
```

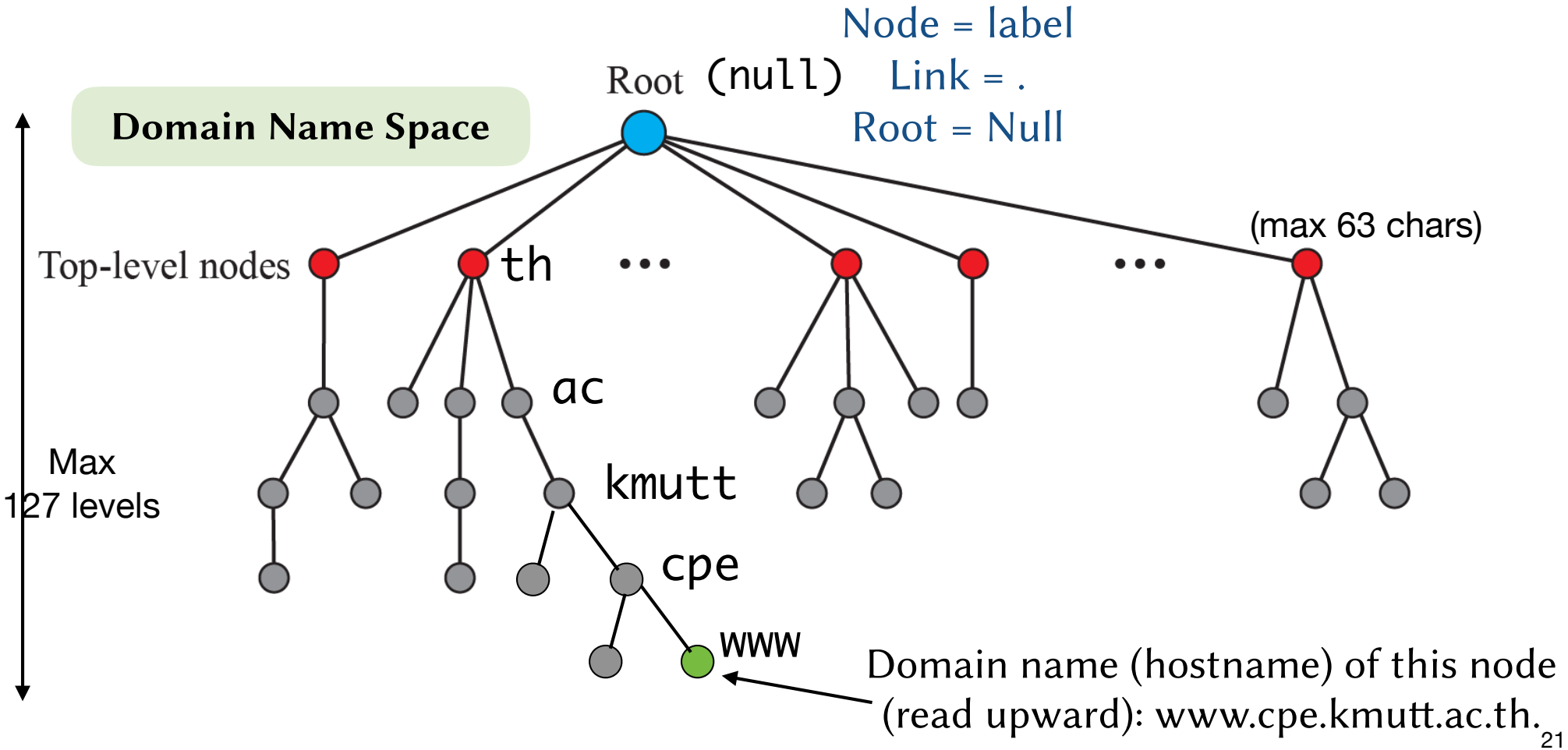
Q: Why dotted separated labels?

Name Resolution Steps



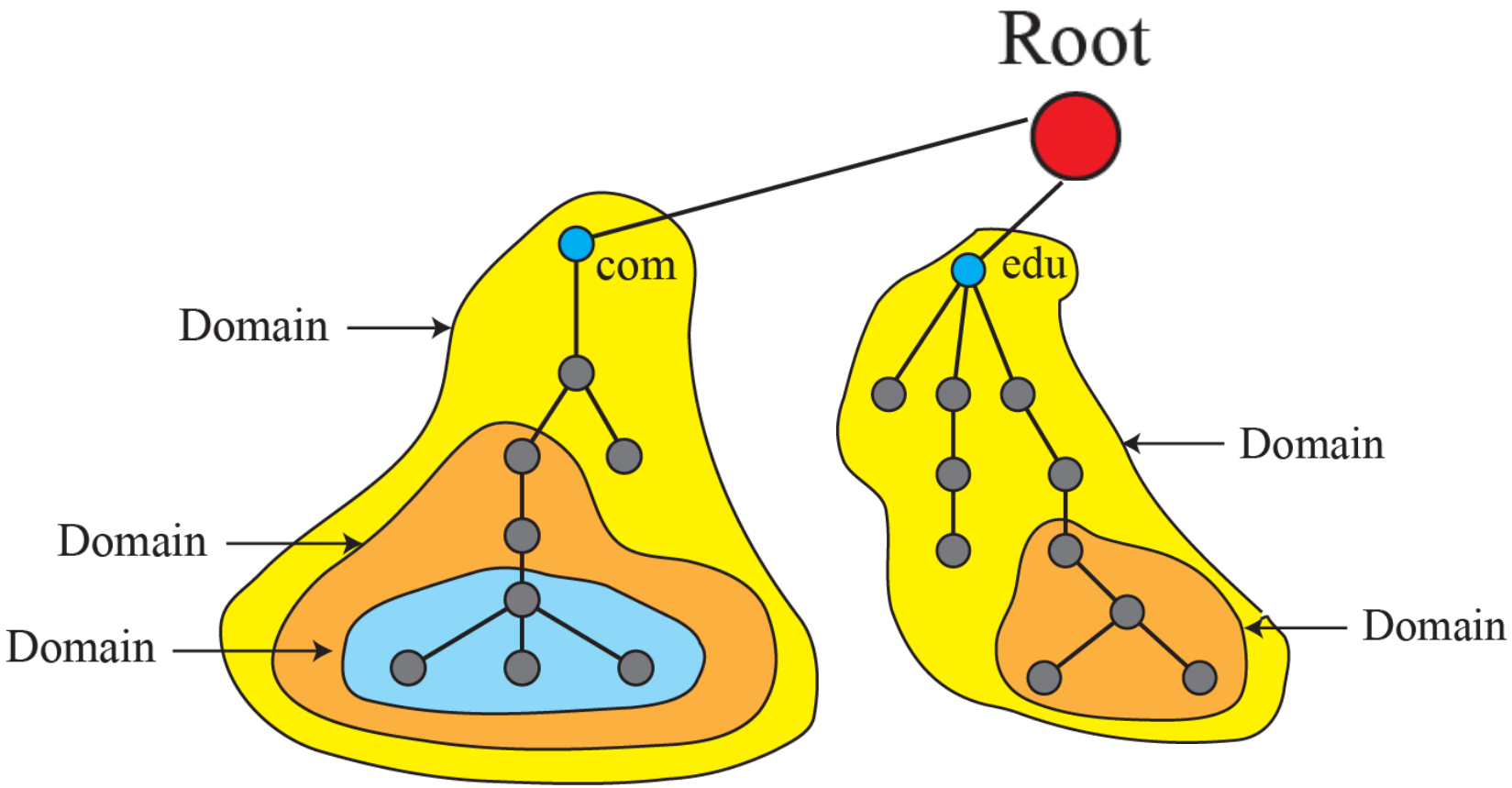
Hierarchical Domain Name

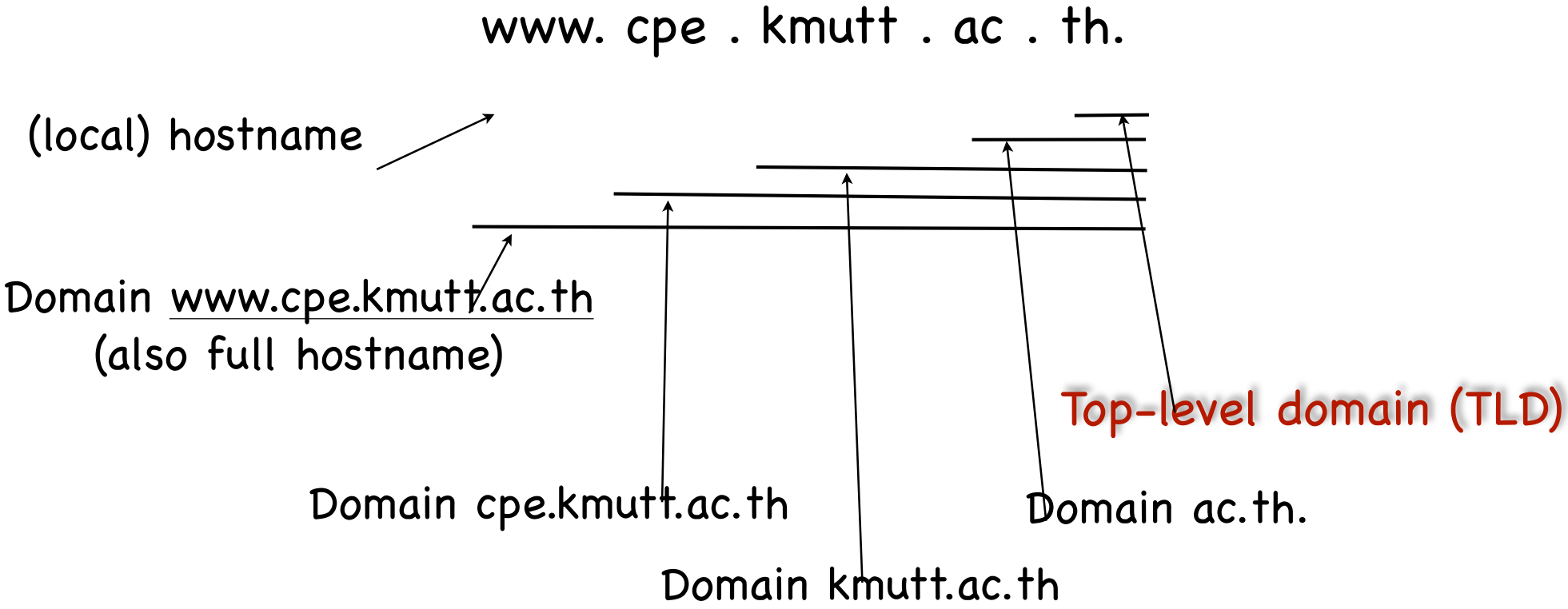
- Dotted separated labels with structure (similar to phone number)



Domain

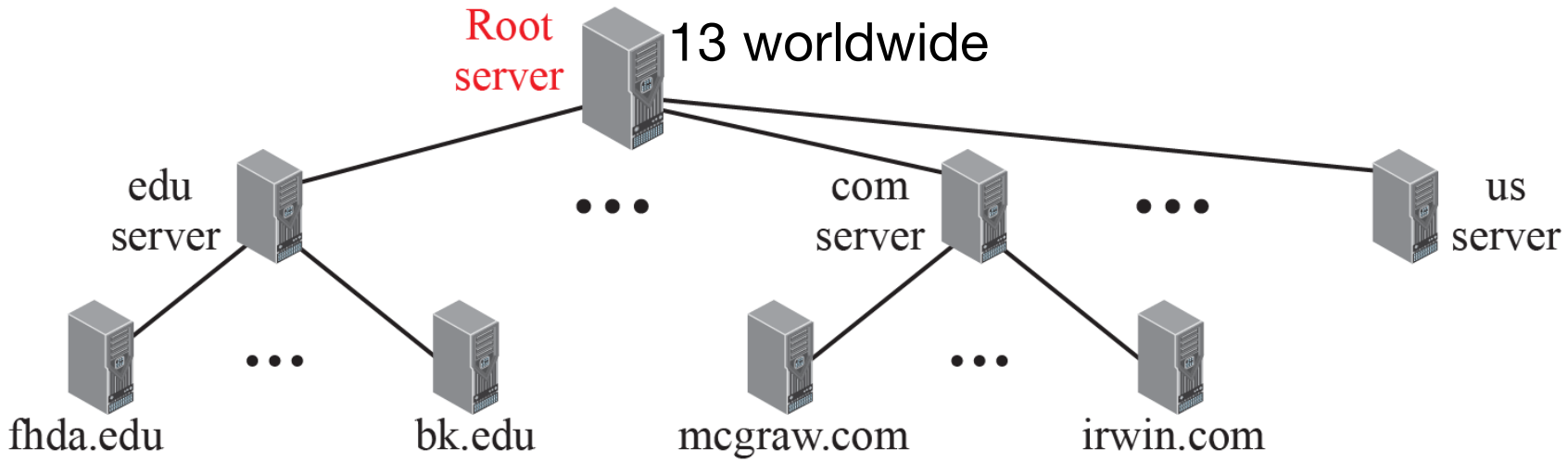
- Subtree of the domain name space rooted at a given node.




www. cpe . kmutt . ac . th.
(local) hostname
Domain www.cpe.kmutt.ac.th
(also full hostname)
Top-level domain (TLD)
Domain cpe.kmutt.ac.th
Domain ac.th.
Domain kmutt.ac.th

Distributed Name Server Implementation

- Hierarchical name structure allows distributed, hierarchical DNS database implementation
- **Types of DNS servers**
  - TLD and Authoritative server: Primary, Secondary (backup)
  - Local DNS server (not in Hierarchy)



- **Top-level domain (TLD) servers:**
  - Responsible for com, org, net, edu, etc, and all top-level country domains uk, fr, ca, jp.
  - Network Solutions maintains servers for com TLD
  - Educause for edu TLD
- **Authoritative DNS servers:**
  - Organization's DNS servers, providing authoritative hostname to IP mappings for organization's servers (e.g., Web, mail).
  - Can be maintained by organization or service provider

Resource Records

- NS stores a database of resource records (RR)
  - 5-tuple structure: {*Domain name, Type, Class, TTL, Value*}
  - *Type* defines how *Domain name* and *Value* are interpreted.
  - Ex: {www.cpe.kmutt.ac.th, A, IN, 86400, 202.44.12,85}

Type	*Interpretation of value*
A	A 32-bit IPv4 address (see Chapter 4)
NS	Identifies the authoritative servers for a zone
CNAME	Defines an alias for the official name of a host
SOA	Marks the beginning of a zone
MX	Redirects mail to a mail server
AAAA	An IPv6 address (see Chapter 4)

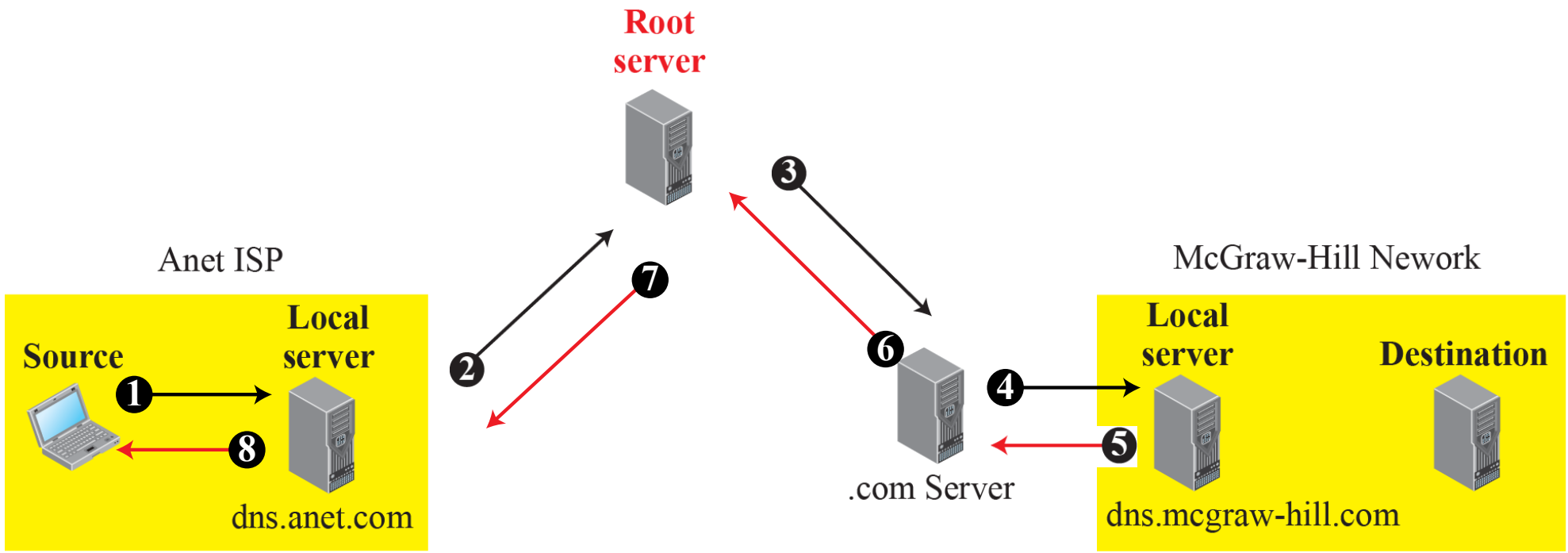
Sample Resource Record

```
$TTL 86400    ; Default TTL for a DNS record cache in seconds (1 day)
@  IN SOA ns1.cpe.net. root ( ; Must be canonical name, not alias name
          1       ; serial
          28800   ; refresh  8 hours
          14400    ; retry
          3600000 ; expire
          86400   ; TTL
        )
pc1   IN A   202.0.1.1
ns1   IN A   202.0.1.2
pc2   IN A   202.0.1.2
pc3   IN A   202.0.1.3
pc4   IN A   202.0.1.4
dns-server   IN CNAME pc2
cpe2.net     IN NS ns1
```

- Type=A
  - **name** is hostname
  - **value** is IP address
- Type=MX
  - **value** is name of mailserver associated with **name**
- Type=NS
  - **name** is domain (e.g. foo.com)
  - **value** is IP address or hostname of authoritative name server for this domain
- Type=CNAME
  - **name** is alias name for some "canonical" (the real) name www.ibm.com is really servereast.ibm.com
  - **value** is canonical name

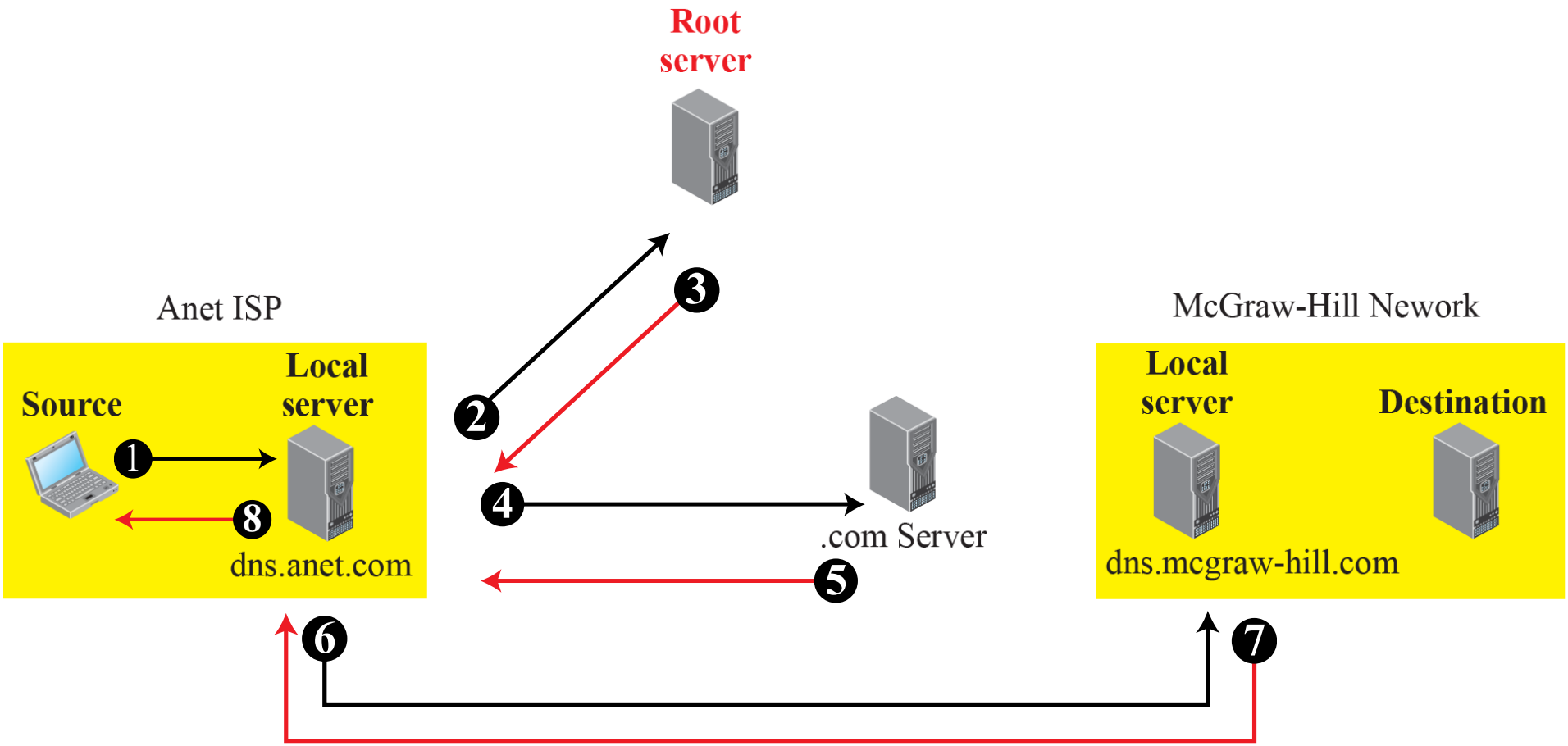
```
$ nslookup -type=NS kmutt.ac.th
Server:         172.20.10.1
Address:        172.20.10.1#53

Non-authoritative answer:
kmutt.ac.th     nameserver = sucreep.kmutt.ac.th.
kmutt.ac.th     nameserver = machanu.kmutt.ac.th.
kmutt.ac.th     nameserver = taksin.kmutt.ac.th.
kmutt.ac.th     nameserver = ongkot.kmutt.ac.th.

Authoritative answers can be found from:
```

Recursive Resolution



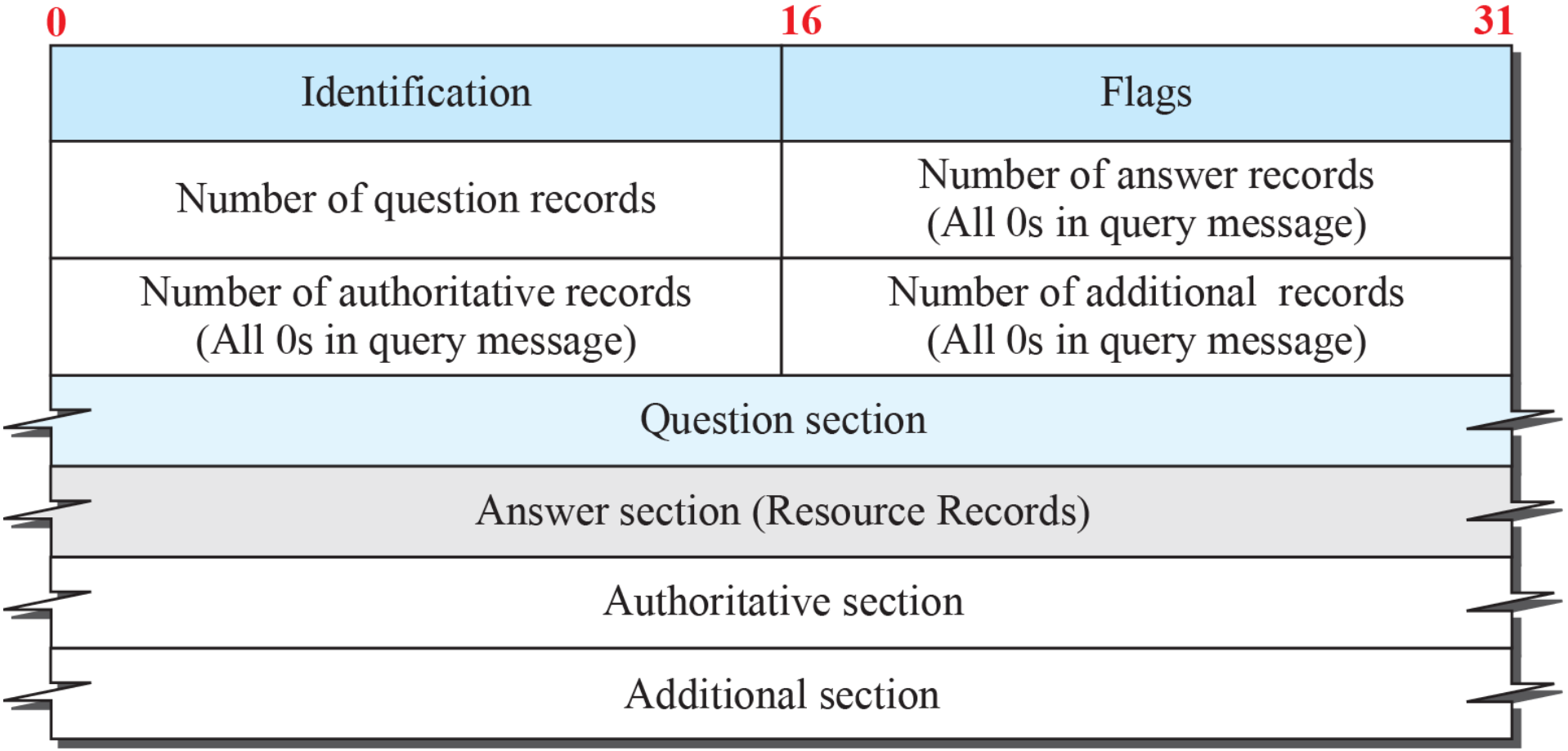
Source: some.anet.com
Destination: engineering.mcgraw-hill.com

Iterative Resolution

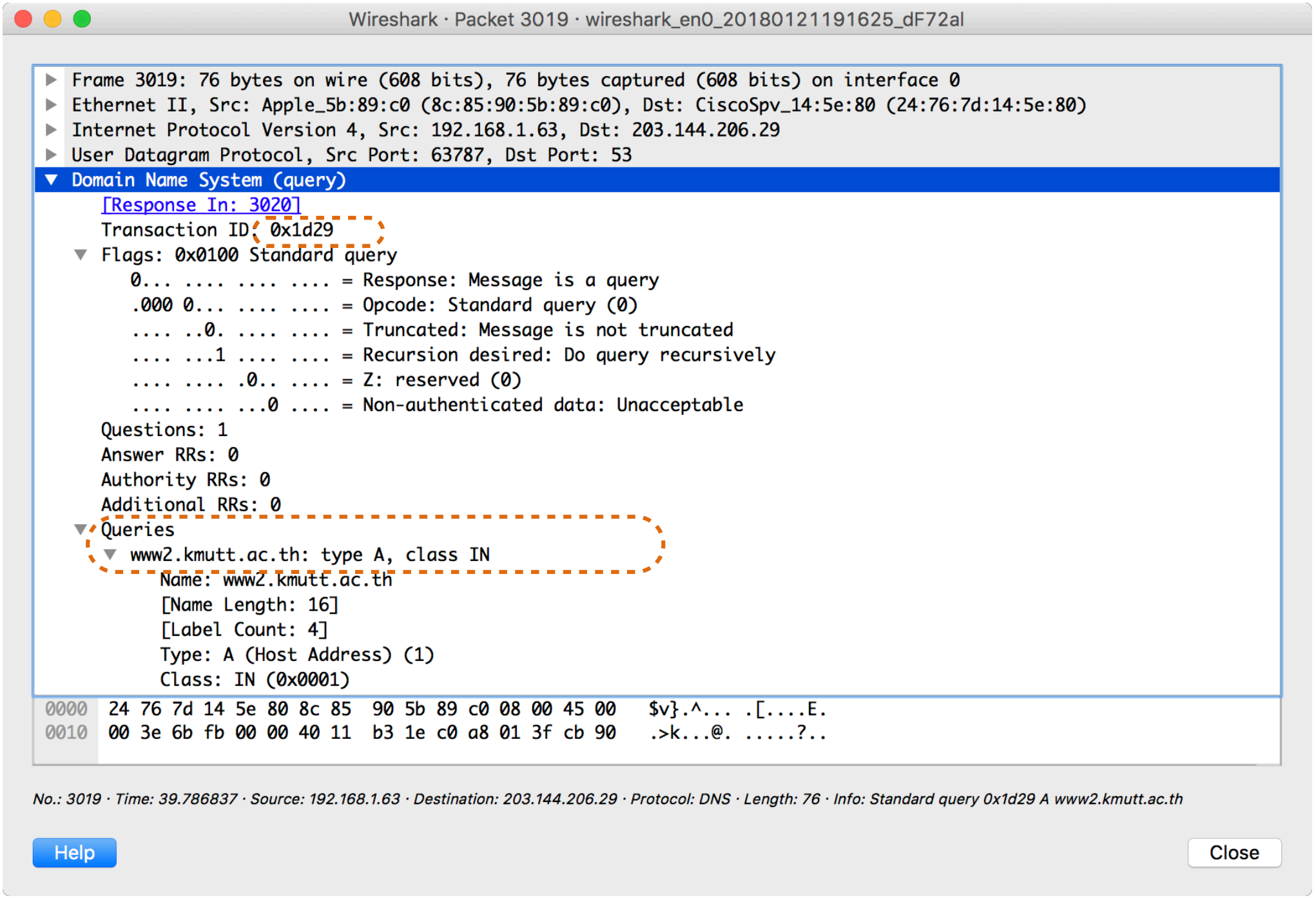


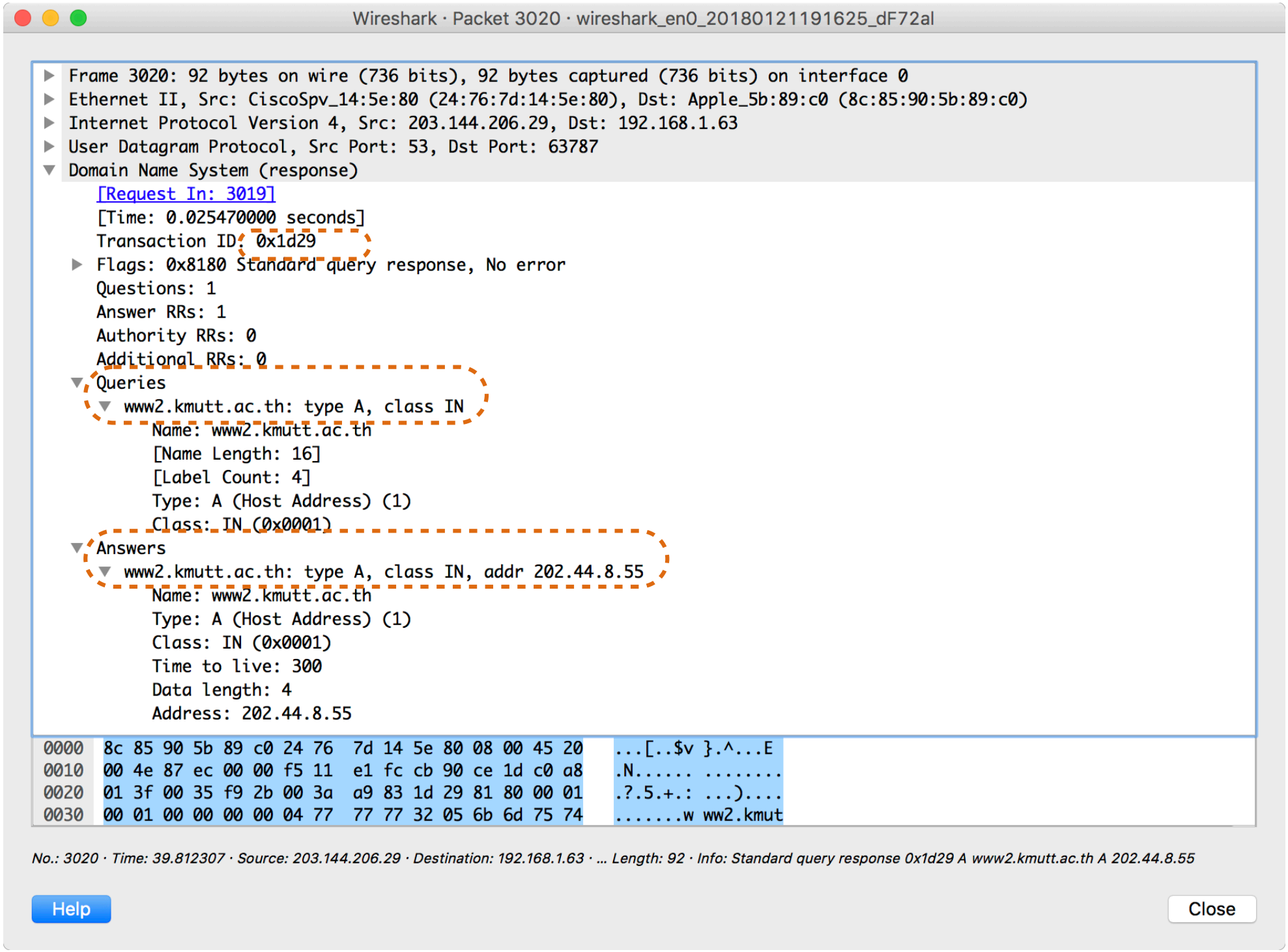
Source: some.anet.com
Destination: engineering.mcgraw-hill.com

DNS Messages (Request, Response)




```
Flags: 0x0100 Standard query
   0... .... .... .... = Response: Message is a query
   .000 0... .... .... = Opcode: Standard query (0)
   .... ..0. .... .... = Truncated: Message is not truncated
   .... ...1 .... .... = Recursion desired: Do query recursively
   .... .... .0.. .... = Z: reserved (0)
   .... .... ...0 .... = Non-authenticated data: Unacceptable
```


Wireshark · Packet 3019 · wireshark_en0_20180121191625_dF72al
Frame 3019: 76 bytes on wire (608 bits), 76 bytes captured (608 bits) on interface 0
Ethernet II, Src: Apple_5b:89:c0 (8c:85:90:5b:89:c0), Dst: CiscoSpv_14:5e:80 (24:76:7d:14:5e:80)
Internet Protocol Version 4, Src: 192.168.1.63, Dst: 203.144.206.29
User Datagram Protocol, Src Port: 63787, Dst Port: 53
Domain Name System (query)
[Response In: 3020]
Transaction ID: 0x1d29
Flags: 0x0100 Standard query
0... .... = Response: Message is a query
.000 0... .... = Opcode: Standard query (0)
.... ..0. .... = Truncated: Message is not truncated
....1 .... = Recursion desired: Do query recursively
.... .0.. .... = Z: reserved (0)
....0 .... = Non-authenticated data: Unacceptable
Questions: 1
Answer RRs: 0
Authority RRs: 0
Additional RRs: 0
Queries
www2.kmutt.ac.th: type A, class IN
Name: www2.kmutt.ac.th
[Name Length: 16]
[Label Count: 4]
Type: A (Host Address) (1)
Class: IN (0x0001)
0000 24 76 7d 14 5e 80 8c 85 90 5b 89 c0 08 00 45 00 $v}.^.... .[....E.
0010 00 3e 6b fb 00 00 40 11 b3 1e c0 a8 01 3f cb 90 .>k...@. .....?..
No.: 3019 · Time: 39.786837 · Source: 192.168.1.63 · Destination: 203.144.206.29 · Protocol: DNS · Length: 76 · Info: Standard query 0x1d29 A www2.kmutt.ac.th
Help
Close


Wireshark · Packet 3020 · wireshark_en0_20180121191625_dF72al
Frame 3020: 92 bytes on wire (736 bits), 92 bytes captured (736 bits) on interface 0
Ethernet II, Src: CiscoSpv_14:5e:80 (24:76:7d:14:5e:80), Dst: Apple_5b:89:c0 (8c:85:90:5b:89:c0)
Internet Protocol Version 4, Src: 203.144.206.29, Dst: 192.168.1.63
User Datagram Protocol, Src Port: 53, Dst Port: 63787
Domain Name System (response)
[Request In: 3019]
[Time: 0.025470000 seconds]
Transaction ID: 0x1d29
Flags: 0x8180 Standard query response, No error
Questions: 1
Answer RRs: 1
Authority RRs: 0
Additional RRs: 0
Queries
www2.kmutt.ac.th: type A, class IN
Name: www2.kmutt.ac.th
[Name Length: 16]
[Label Count: 4]
Type: A (Host Address) (1)
Class: IN (0x0001)
Answers
www2.kmutt.ac.th: type A, class IN, addr 202.44.8.55
Name: www2.kmutt.ac.th
Type: A (Host Address) (1)
Class: IN (0x0001)
Time to live: 300
Data length: 4
Address: 202.44.8.55
0000 8c 85 90 5b 89 c0 24 76 7d 14 5e 80 08 00 45 20 ...[..$v }.^...E
0010 00 4e 87 ec 00 00 f5 11 e1 fc cb 90 ce 1d c0 a8 .N..... ....
0020 01 3f 00 35 f9 2b 00 3a a9 83 1d 29 81 80 00 01 .?.5.+.: ...)....
0030 00 01 00 00 00 00 04 77 77 77 32 05 6b 6d 75 74 ....w ww2.kmut
No.: 3020 · Time: 39.812307 · Source: 203.144.206.29 · Destination: 192.168.1.63 · ... Length: 92 · Info: Standard query response 0x1d29 A www2.kmutt.ac.th A 202.44.8.55
Help
Close

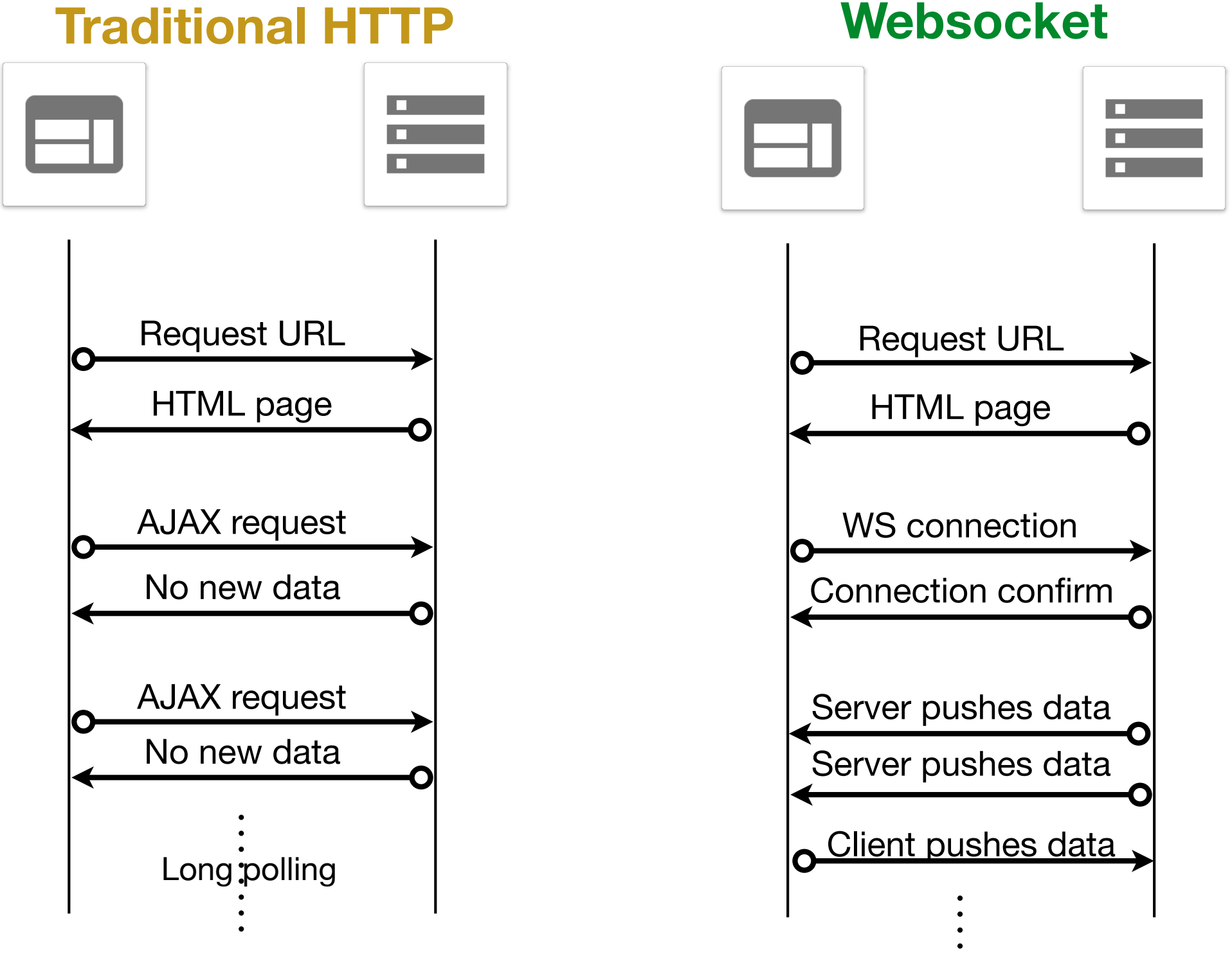
Self-Test

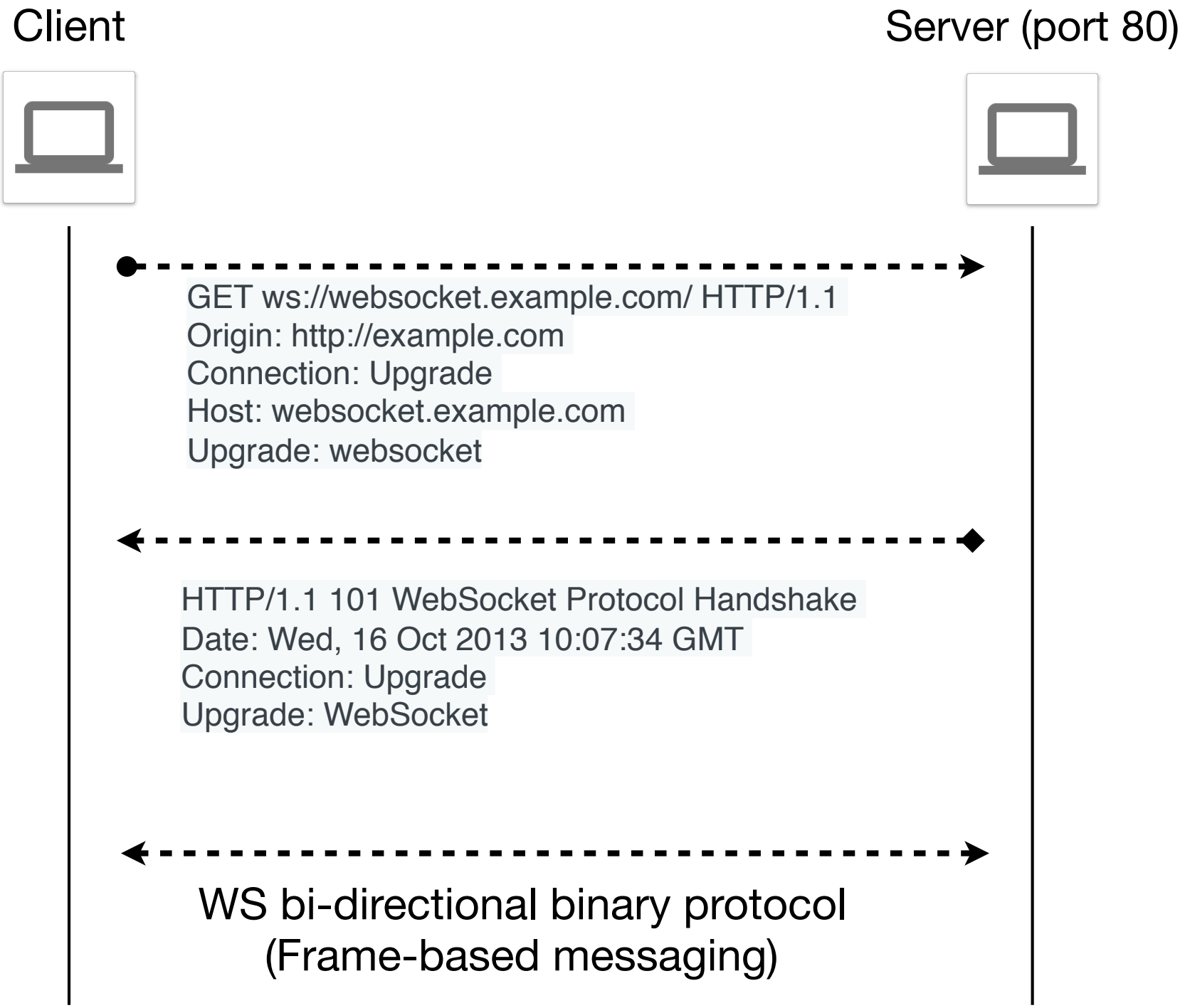
Design Lessons Learned

- Hierarchy + Distributed databases = Scalability
- Secondary server for fault tolerance
- UDP for one-short transaction
  - Why not TCP?
  - What if the records really long?

WebSocket RFC6455

- Limitations of HTTP
  - Server cannot push data to clients (e.g., real-time data update).
  - High latency and overhead if using polling.
- Websocket allows full duplex data transfer without the need for client to send a request.
  - Client first connects to server running at port 80 (ws://…)
  - HTTP handshake before switching to WS

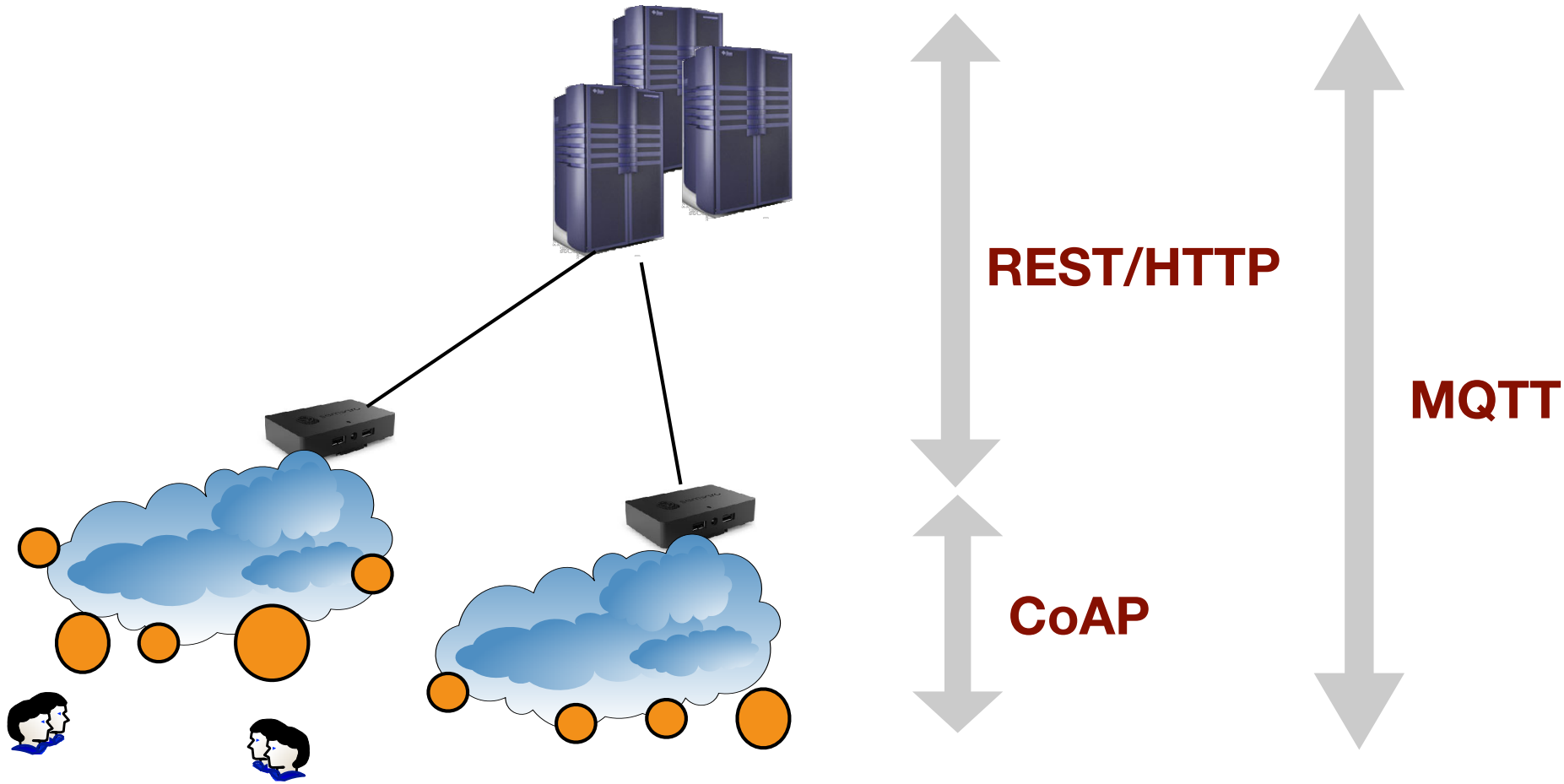

Traditional HTTP
Request URL
HTML page
AJAX request
No new data
AJAX request
No new data
Long polling
Websocket
Request URL
HTML page
WS connection
Connection confirm
Server pushes data
Server pushes data
Client pushes data


Client
Server (port 80)
GET ws://websocket.example.com/ HTTP/1.1
Origin: http://example.com
Connection: Upgrade
Host: websocket.example.com
Upgrade: websocket
HTTP/1.1 101 WebSocket Protocol Handshake
Date: Wed, 16 Oct 2013 10:07:34 GMT
Connection: Upgrade
Upgrade: WebSocket
WS bi-directional binary protocol
(Frame-based messaging)

Why Websocket?

- Lower latency and overhead, and more flexible than HTTP
- Allow port 80 to be used for proprietary service
  - Most firewalls and servers allow only well-known ports like 80 and 443 for security reasons
  - Connection easily goes through HTTP proxy

IoT Data Transport Protocols



Application data transfer (Request/response, Publish/Subscribe)
Service and device discoveries

REST: REpresentational State Transfer

- **RE**: Resources encoded as representations (XML, EXI, JSON).
  - Temperature (resource) value in decimal number (representation)
- **S**: Necessary states to complete a request provided with the request.
  - Client/server cannot rely on states to be stored on the other side.
- **T**: Representations and states can be transferred between client and servers.

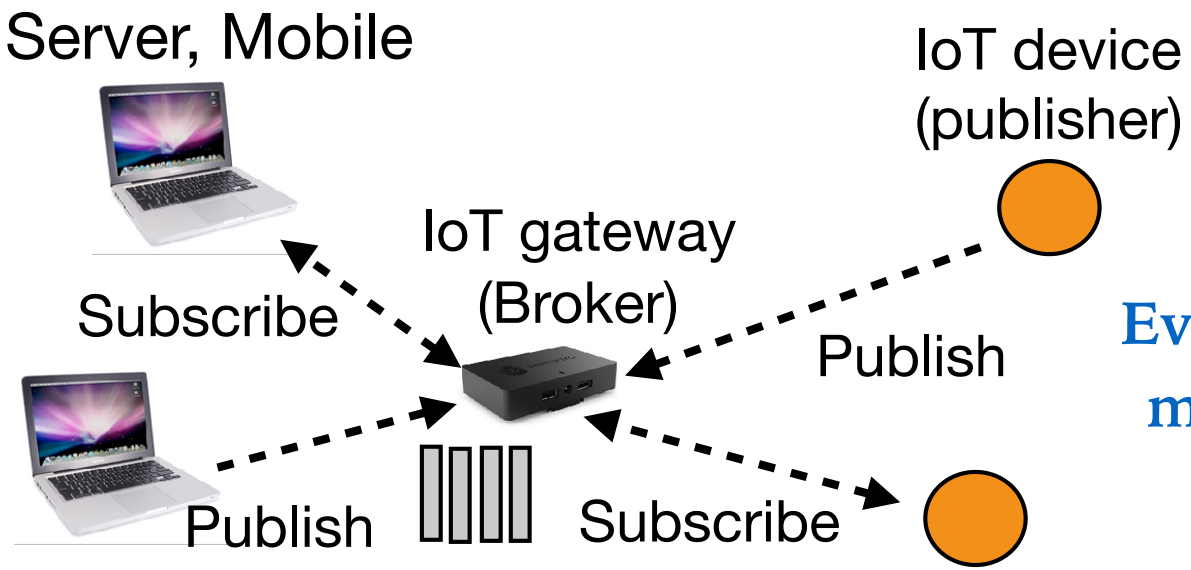
RESTful HTTP

```
GET /sensors/temperature HTTP/1.1
Content-type: application/json
```

```
HTTP/1.1 200 OK
Content-type: application/json

{"sensors":[{"name": "Temperature", "value": 26.1}]}
```

Resources accessed by URI
CRUD operations by HTTP methods

MQTT: MQ Telemetry Transport



Event-triggered or monitoring apps

Typical scenarios

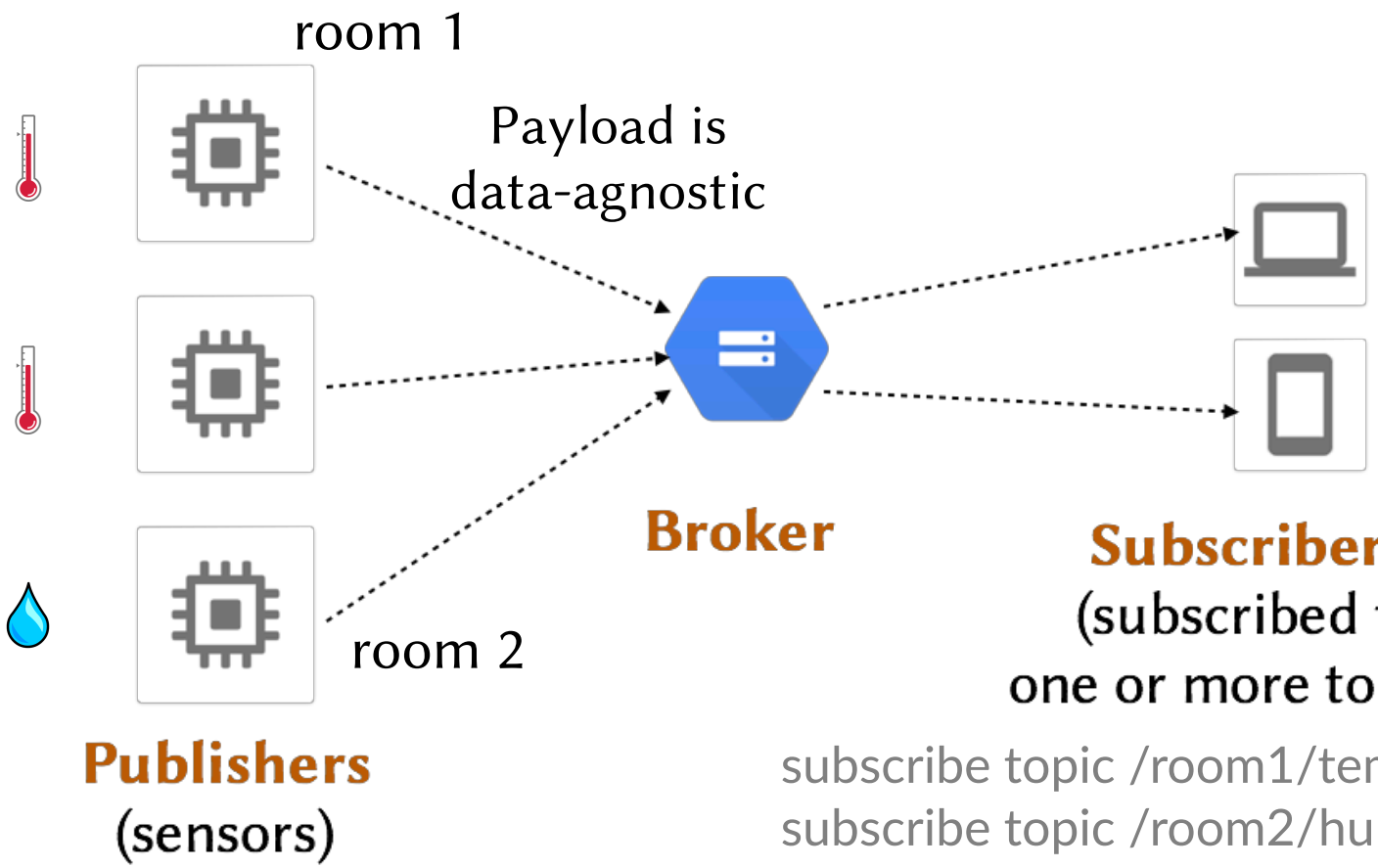
Resource-constrained environment
(Small message size, Stream-based)
Device-to-Server data collection
Remote monitoring for dumb devices with small messages on low bandwidth

2-Byte header

Trying MQTT Publisher and Subscriber

- Download and install Mosquitto on gateway.
- Open Command prompt (admin mode)
- Start Broker with **mosquitto -v**
- Start Subscriber with **mosquitto_sub -h 127.0.0.1 -i testsub -t topicA**
- Start Publisher with **mosquitto_pub -h 127.0.0.1 -i testPublish -t topicA -m "Hello world"**

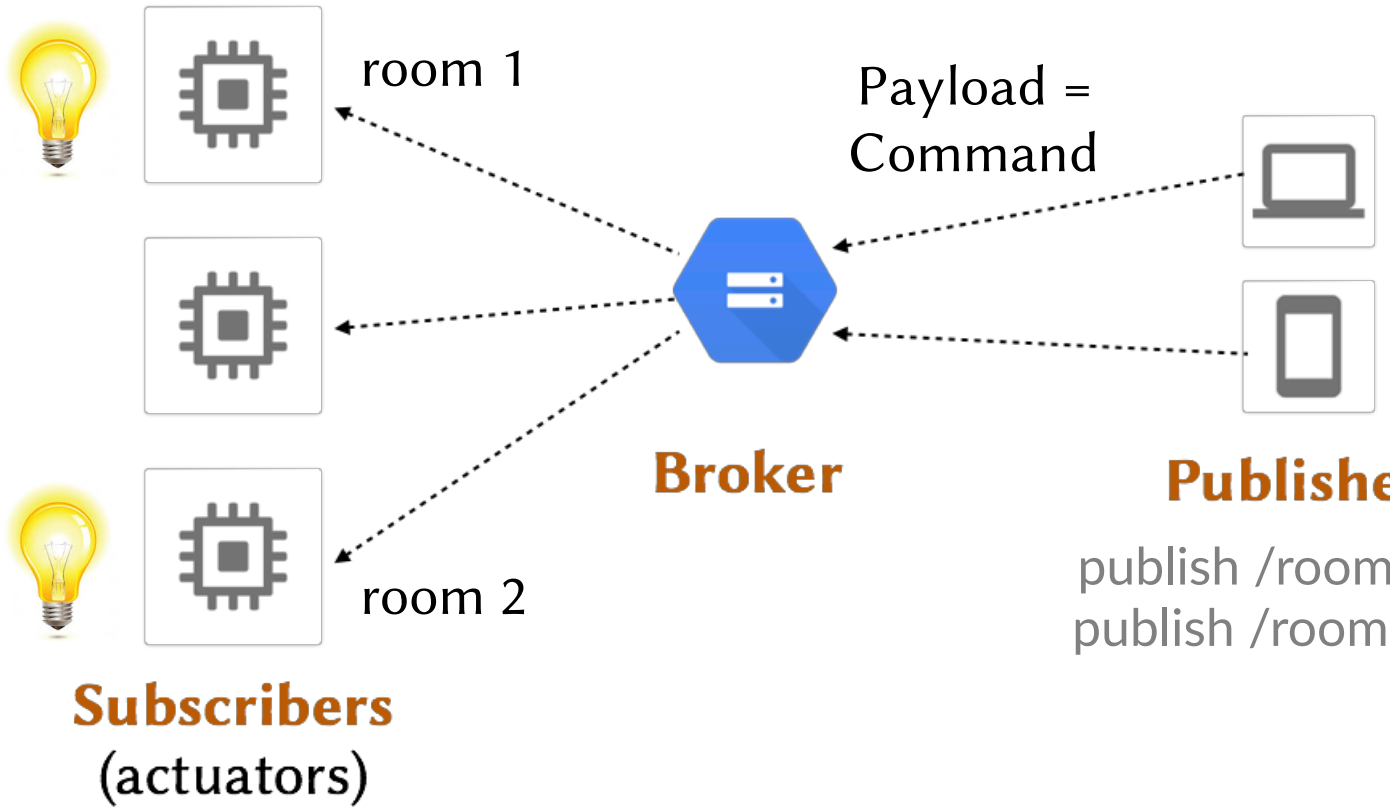
MQTT App Model — Telemetry



subscribe topic /room1/temperature
subscribe topic /room2/humidity

publish /room1/temperature value=25
publish /room2/humidity=40

MQTT App Model — Remote Control



subscribe topic /room1/light
subscribe topic /room2/light

Wildcard Matching

- Single level (+): myhome/groundfloor/+/temperature

  - ✓ myhome / groundfloor / livingroom / temperature
  - ✓ myhome / groundfloor / kitchen / temperature
  - ✗ myhome / groundfloor / kitchen / brightness
  - ✗ myhome / firstfloor / kitchen / temperature
  - ✗ myhome / groundfloor / kitchen / fridge / temperature

- Multilevel (#): myhome/groundfloor/#

  - ✓ myhome / groundfloor / livingroom / temperature
  - ✓ myhome / groundfloor / kitchen / temperature
  - ✓ myhome / groundfloor / kitchen / brightness
  - ✗ myhome / firstfloor / kitchen / temperature

Self Test

Conclusion

- Client-server paradigm (Request/Command/Response) most widely deployed
  - User applications
  - Network service applications
- AL protocols to support transport of application data
- Header fields mostly text-based for ease of human readability.

- Some key AL protocol design principles
  - Stateless vs. Stateful
  - Centralized vs. Distributed contents
  - Flat vs. Hierarchy
  - UDP vs TCP
  - Text-based vs. Binary Syntax
- Term project 1