Application Layer

Dr. Xiqun Lu

College of Computer Science

Zhejiang University

Outline

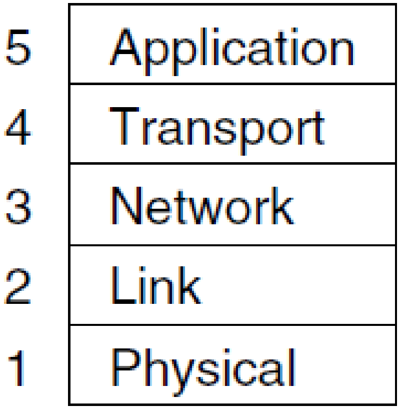
- Overview of application layer
- Important application layer protocols
  - DNS
  - FTP
  - Email
  - HTTP

Outline

- Overview of application layer
- Important application layer protocols
  - DNS
  - FTP
  - Email
  - HTTP

Where we are in the course?

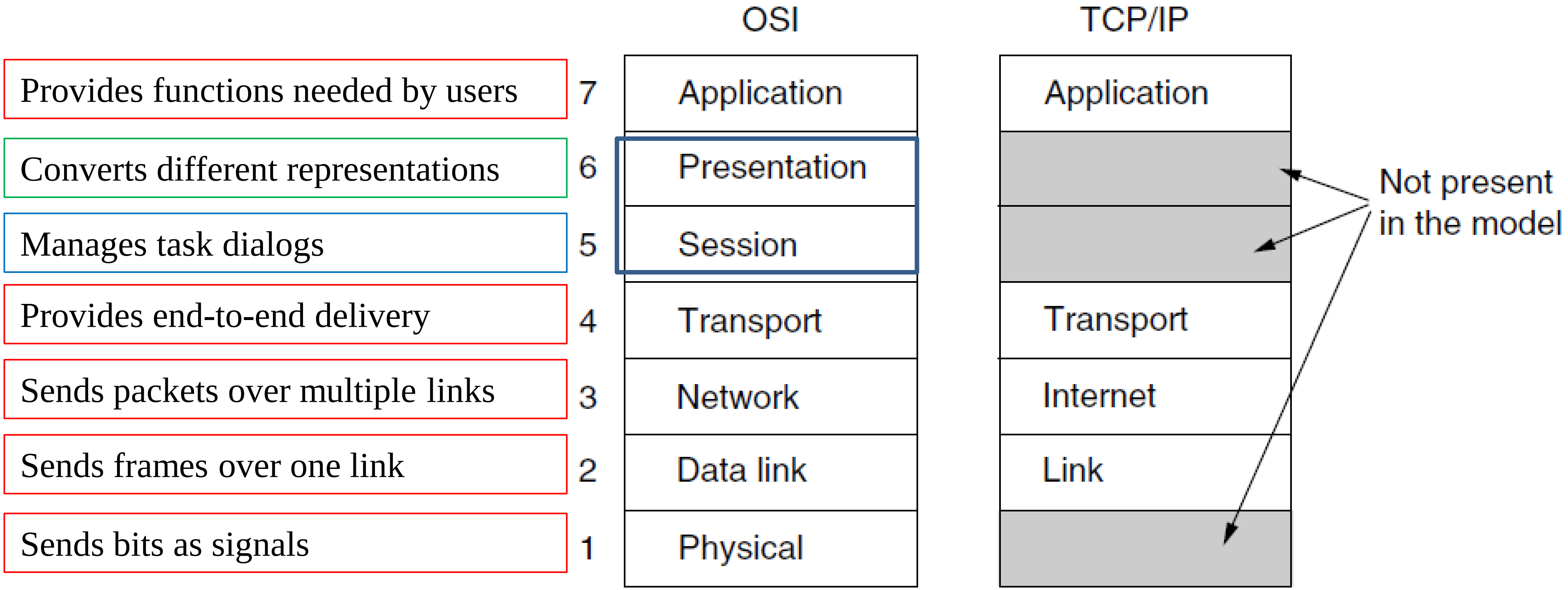
- Starting the Application Layer!
- Builds distributed "network services" (DNS, Web) on transport services.
- Application layer protocols are often part of an "application".
- Application layer messages are often split over multiple packets or may be aggregated in a packet.



5	Application
4	Transport
3	Network
2	Link
1	Physical

Figure 1-23. The reference model used in this book.

OSI Session/Presentation Layers

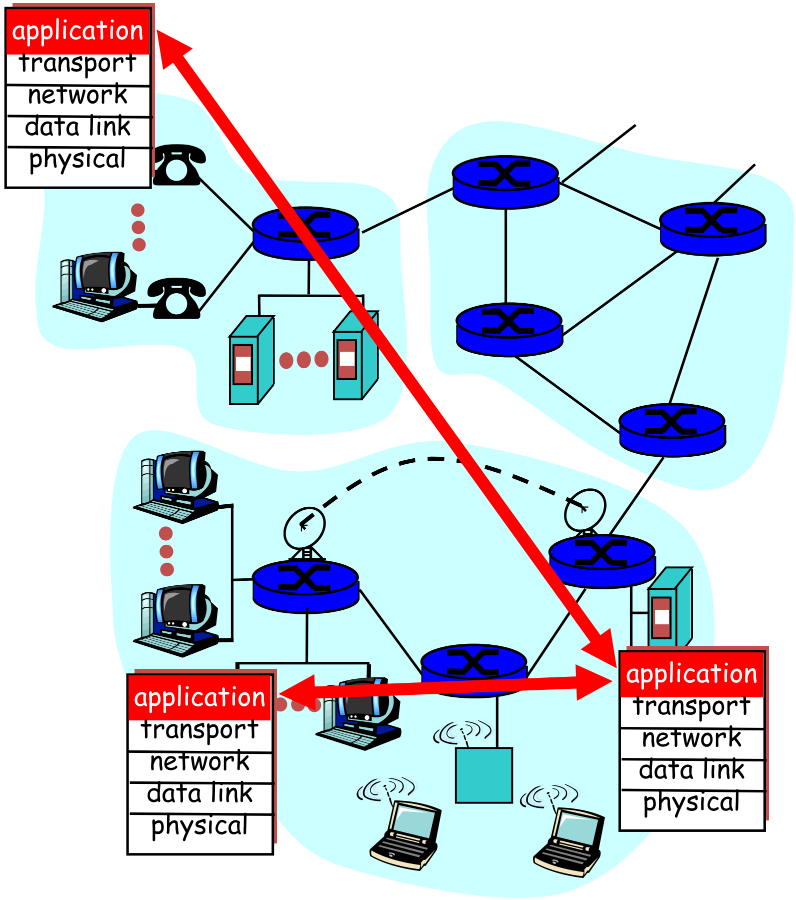


		OSI	TCP/IP
Provides functions needed by users	7	Application	Application
Converts different representations	6	Presentation	
Manages task dialogs	5	Session	
Provides end-to-end delivery	4	Transport	Transport
Sends packets over multiple links	3	Network	Internet
Sends frames over one link	2	Data link	Link
Sends bits as signals	1	Physical	

Not present in the model

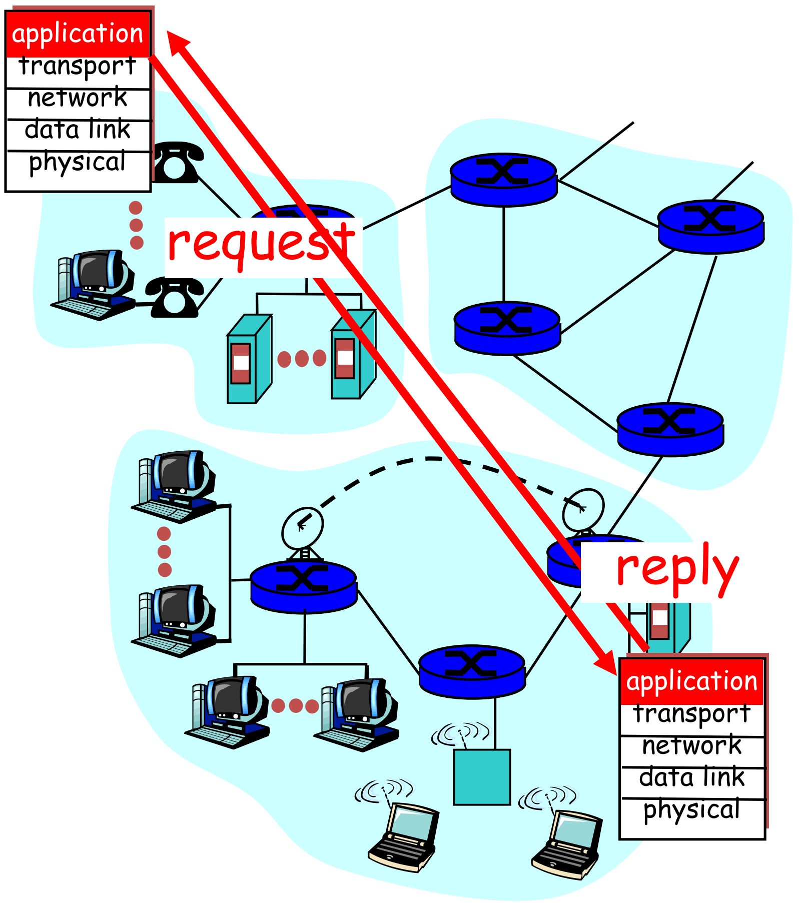
Figure 1-21. The TCP/IP reference model.

Applications and Application Level Protocols



- The three concepts
  - Protocol
  - Service model
  - Interface
- Network application is more than application level protocols
  - Client site
  - Server site
  - Application level protocol

Client/Server Paradigm



- Client
  - Initiates contact with server (speak first)
  - Typically request service from server
  - Question: identify who is/implements client in
    - Web?
    - Email?
- Server
  - Provides requested service to clients
  - Question: identify who is/implements the server counterpart in
    - Web?
    - Email?

Which Transport Service Does Application Need? - Parameters

- Data Loss
  - Loss-tolerant applications, e.g. audio/video
  - Other applications such as file transfer, telnet requires 100% reliable transmission
- Bandwidth
  - Bandwidth-sensitive applications, such as multimedia, require a maximum amount of bandwidth
  - Elastic applications: can use whatever bandwidth available
- Timing
  - Some applications such as internet telephone requires “low delay” to be effective.

Transport Service Required By Common Applications

Application	Data loss	Bandwidth	Time Sensitive
file transfer	no loss	elastic	no
e-mail	no loss	elastic	no
Web documents	loss-tolerant	elastic	no
real-time audio/video	loss-tolerant	audio: 5Kb-1Mb video:10Kb-5Mb	yes, 100’s msec
stored audio/video	loss-tolerant	same as above	yes, few secs
interactive games	loss-tolerant	few Kbps up	yes, 100’s msec
financial apps	no loss	elastic	yes and no

Internet Applications and Their Transport Layer Protocols

Application	Application layer protocol	Underlying transport protocol
e-mail	SMTP [RFC 821]	TCP
remote terminal access	telnet [RFC 854]	TCP
Web	http [RFC 2068]	TCP
file transfer	ftp [RFC 959]	TCP
streaming multimedia	proprietary (e.g. RealNetworks)	TCP or UDP
remote file server	NFS	TCP or UDP
Internet telephony	Proprietary (private) (e.g., Skype)	typically UDP

A Brief Summary of the Evolution of the Internet

Memex Conceived 1945

A Mathematical Theory of Communication 1948

Silicon Chip 1958

First Vast Computer Network Envisioned 1962

Packet Switching Invented 1964

Hypertext Invented 1965

ARPANET 1969

TCP/IP Created 1972

Internet Named and Goes TCP/IP 1984

WWW Created 1989

Mosaic Created 1993

Age of eCommerce Begins 1995

1945

1995

Outline

- Overview of application layer
- Important application layer protocols
  - DNS
  - FTP
  - Email
  - HTTP

Outline

- Overview of application layer
- Important application layer protocols
  - DNS
  - FTP
  - Email
  - HTTP

DNS — Domain Name System

- Why we need DNS?
  - Theoretically Web pages, mailboxes, and other resources are linked to the network addresses (i.e. IP) of the computers on which they are stored, these address are hard for people to remember.
  - If the Web server moves to a different machine with a different IP address, everyone needs to be told the new IP address.
  - Consequently, high-level, readable names were introduced in order to **decouple** machines names from machine addresses.

DNS—Domain Name System

- What is the DNS?
  - DNS was invented in 1983. It has been a key part of the Internet ever since.
  - DNS is defined in RFCs 1034, 1035, 2181.
  - The essence of DNS is the invention of **a hierarchical, domain-based naming scheme** and **a distributed database system** for implementing converting the machine names to network addresses.
  - **To map a name to an IP address**, an application program calls a library procedure called **the resolver**, passing it the name as parameter.
    - An example of a resolver: gethostbyname in Fig.6-6 (socket)
  - The query and response messages are sent as **UDP packets**.

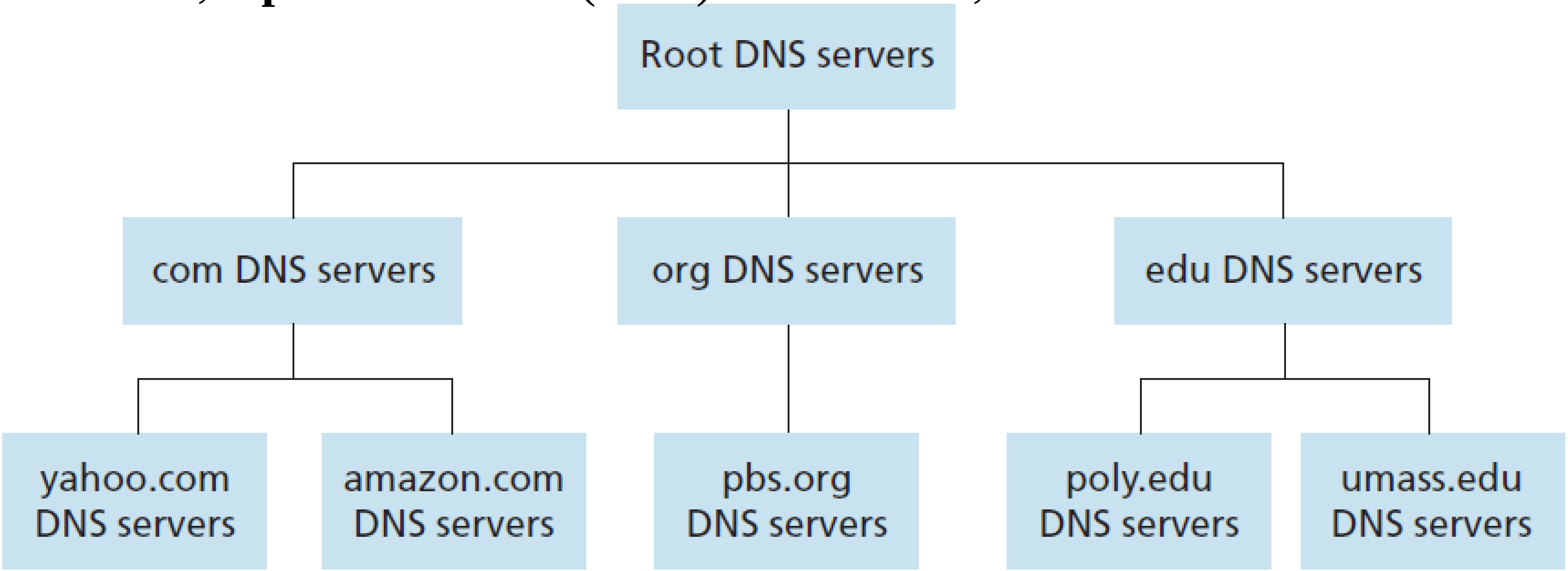
DNS – Why Not Centric?

- Single point of failure
- Traffic volume
- Distant name server means slow response
- Scalability

- History: ARPANET begins with a single **hosts.txt.**
  - hosts.txt listed all the computer names and their IP addresses. For a network of a few hundred large timesharing machines, this approach worked reasonably well.

Hierarchy of DNS Servers

- In order to deal with the issue of scale, the DNS uses a large number of servers, organized in a hierarchical fashion and distributed around the world.
  - The mappings are distributed across the DNS servers.
  - To a first approximation, there are three classes of DNS servers: **root** DNS servers, **top-level domain** (**TLD**) DNS servers, and **authoritative** DNS servers.



Root DNS servers

com DNS servers

org DNS servers

edu DNS servers

yahoo.com DNS servers

amazon.com DNS servers

pbs.org DNS servers

poly.edu DNS servers

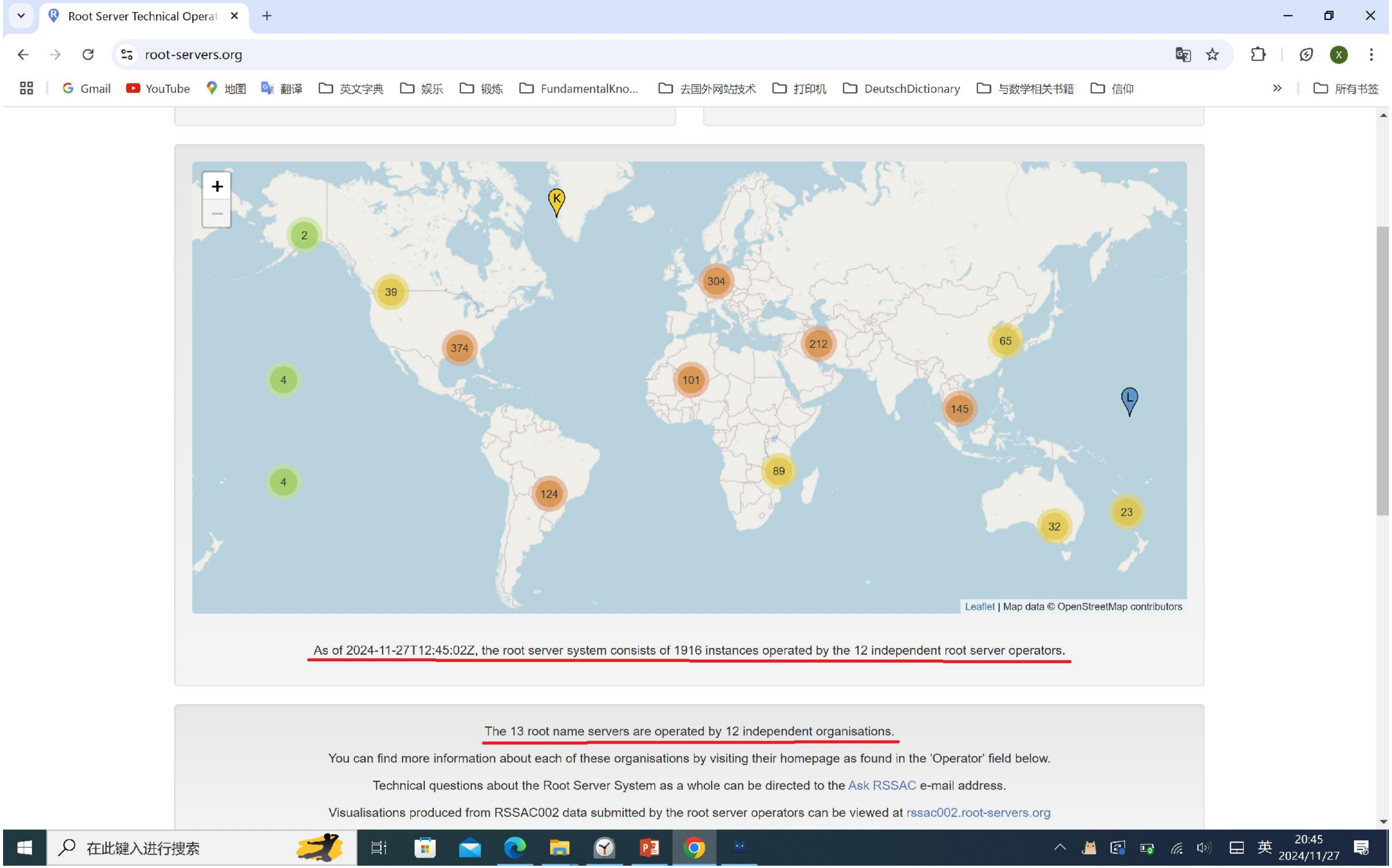
umass.edu DNS servers

Figure 2.19 ♦ Portion of the hierarchy of DNS servers

Root Nameservers

- Root (dot) is served by 13 server names
  - a.root-servers.net to m.root-servers.net
  - Each “server” is actually **a cluster of replicated servers**, for both security and reliability purposes.
  - All nameservers need root IP addresses.
  - Handled via configuration file (named.ca) (ca — cache)
- There are 1916 distributed server instances (to Nov. 27, 2024)
  - Highly reachable, reliable service
  - Most servers are reached by **IP anycast**
    - Most of the servers are present in multiple geographical locations and reached using **anycast routing**, in which a packet is delivered to the nearest instance of a destination address.
  - Servers are IPv4 and IPv6 reachable

Root Servers Deployment



As of 2024-11-27T12:45:02Z, the root server system consists of 1916 instances operated by the 12 independent root server operators.

The 13 root name servers are operated by 12 independent organisations.

You can find more information about each of these organisations by visiting their homepage as found in the 'Operator' field below.

Technical questions about the Root Server System as a whole can be directed to the Ask RSSAC e-mail address.

Visualisations produced from RSSAC002 data submitted by the root server operators can be viewed at rssac002.root-servers.org

The DNS Name Space

- Hierarchical, starting from “.”
- The top-level domains come in two flavors: generic and countries.

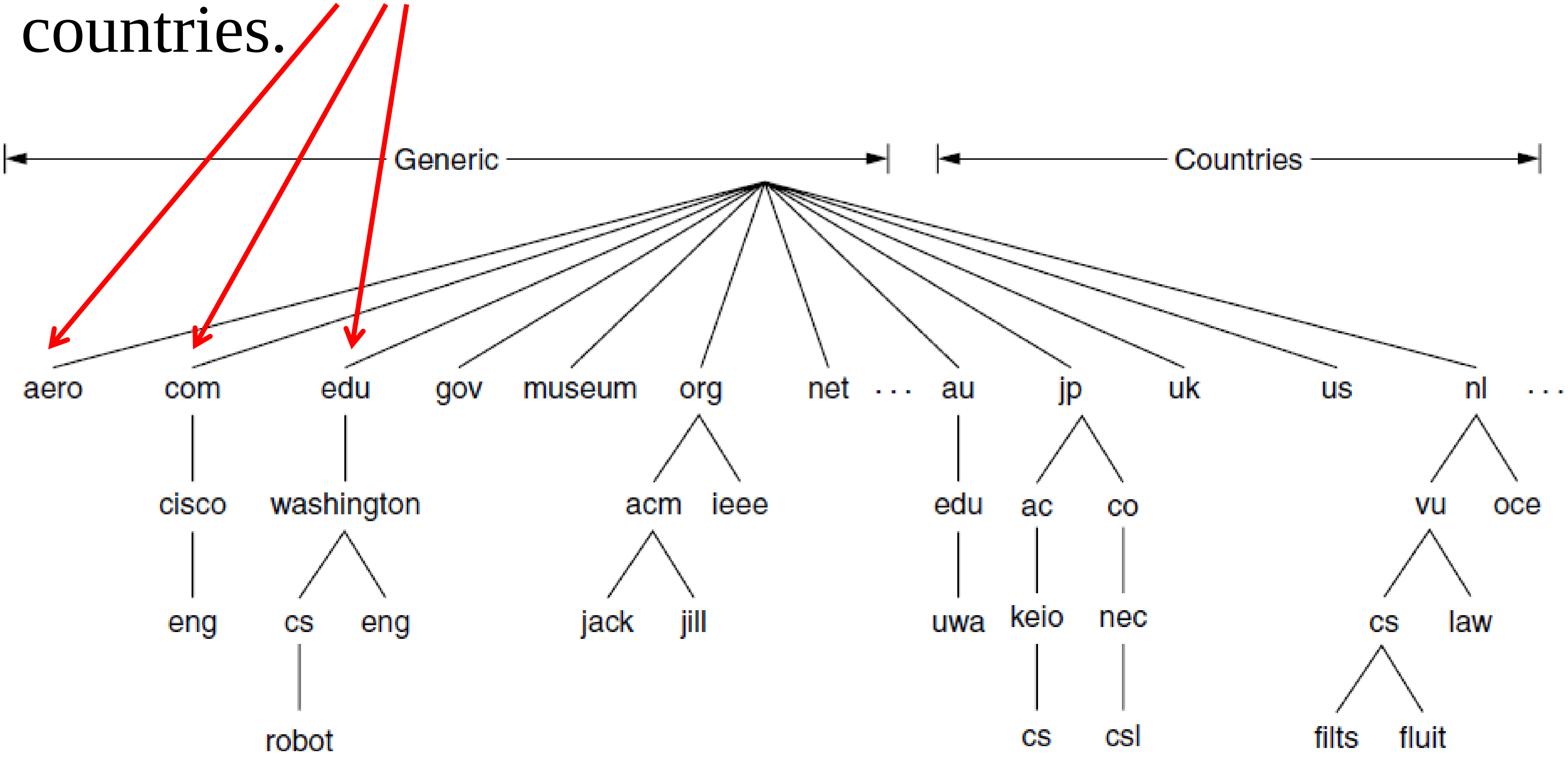


Figure 7-1. A portion of the Internet domain name space.

TLDs (Top-Level Domains)

- Run by ICANN (Internet Corp. for Assigned Names and Numbers)
  - Starting in 1998 (http://www.icann.org/)

- 22+ generic TLDs
  - Initially .com, .edu, .gov, .mil, .org, .net
  - Added .aero, .museum, etc. from 2001 through .xxx in 2011.
  - Different TLDs have different usage policies
- ~250 country code TLDs
  - Two letters, e.g., “au”, plus international characters since 2010.
  - Widely commercialized, e.g., .tv (Tuvalu 图瓦卢)
  - Many domain hacks (黑客), e.g., instagr.am (Armenia 亚美尼亚), goo.gl (Greenland)

DNS Zones

- To avoid the problems associated with having only a single source of information, the DNS name space is divided into **nonoverlapping zones**.
- A zone is a contiguous portion of the namespace. **Each zone is managed by one or more nameservers**.

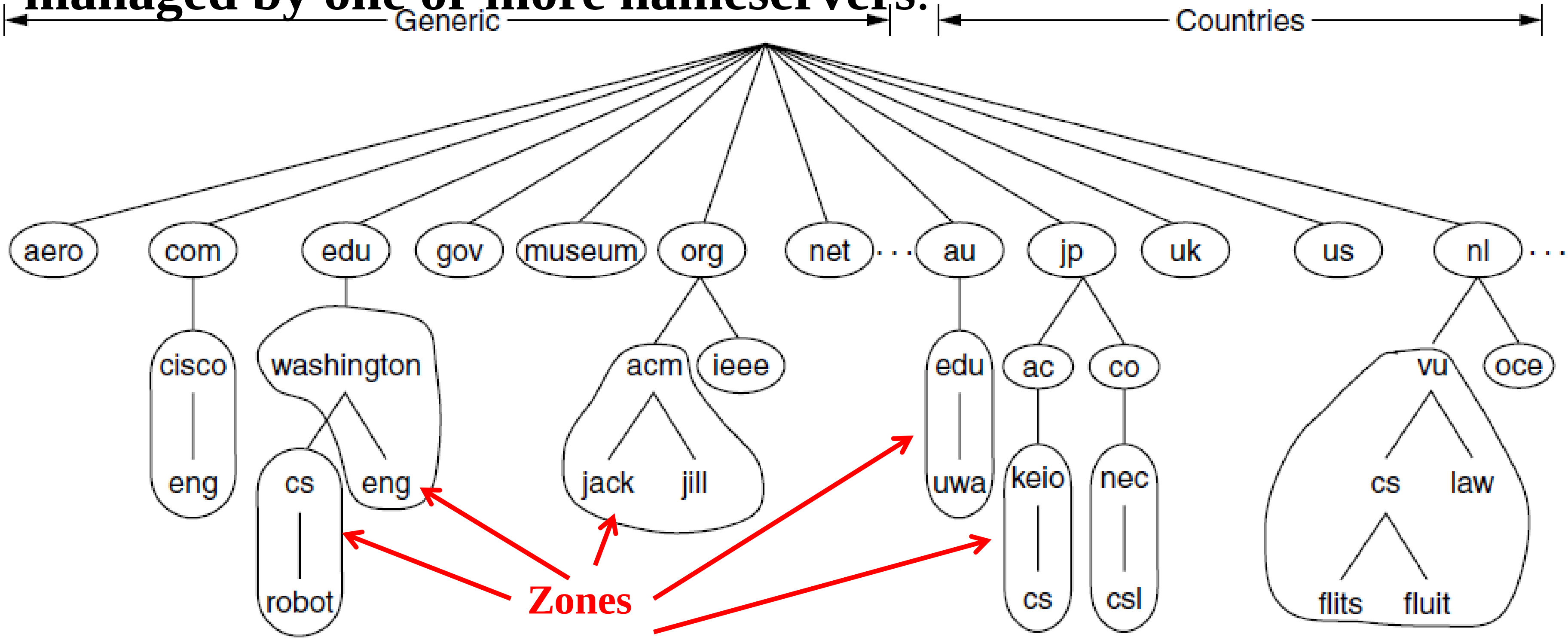
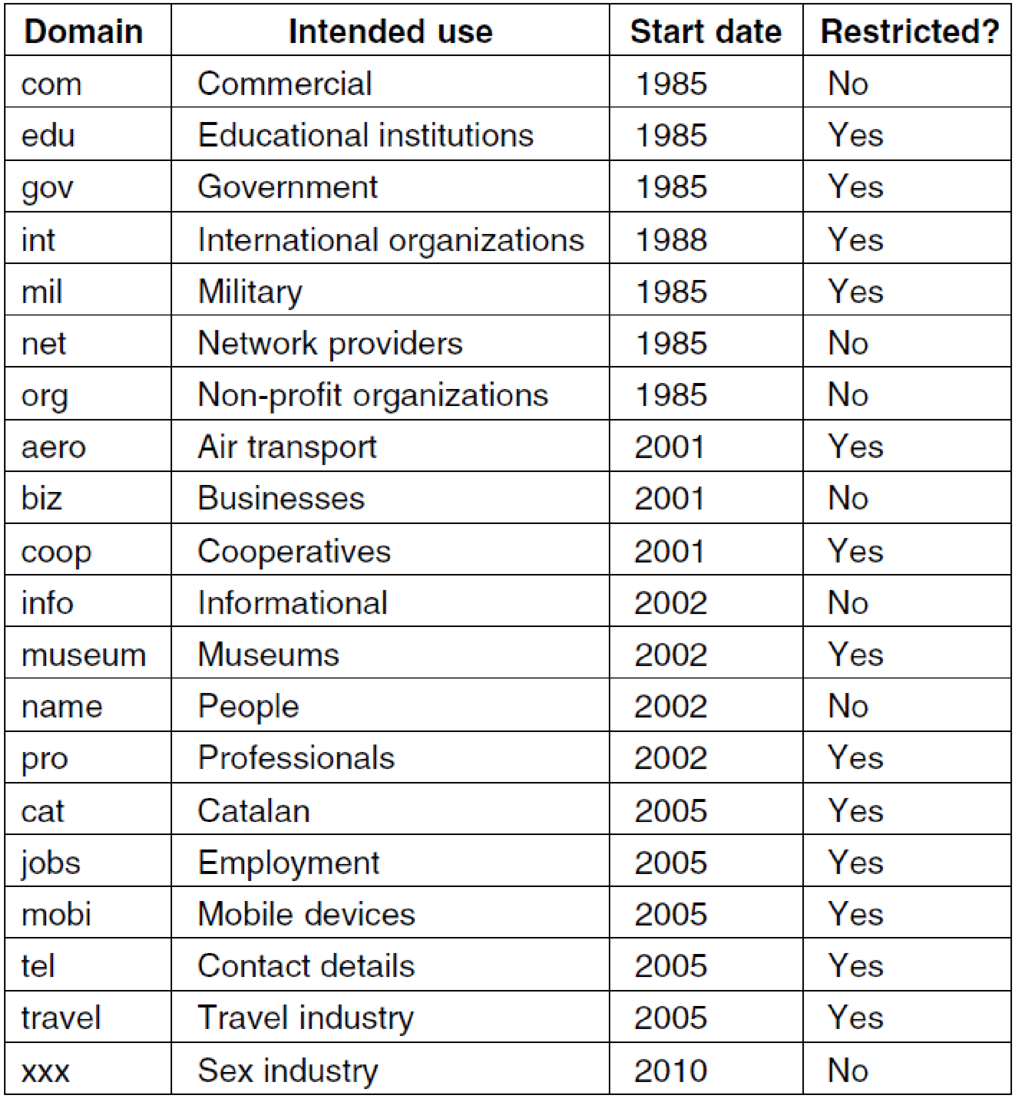


Figure 7-5. Part of the DNS name space divided into zones (which are circled).

DNS Zones (II)

- Zones are the basis for distribution
  - EDU registrar administers .edu
  - ZJU administers zju.edu.cn
  - CS administers cs.zju.edu.cn
- Each zone has one or more nameservers to contact for information about it.

The DNS Name Space



Domain	Intended use	Start date	Restricted?
com	Commercial	1985	No
edu	Educational institutions	1985	Yes
gov	Government	1985	Yes
int	International organizations	1988	Yes
mil	Military	1985	Yes
net	Network providers	1985	No
org	Non-profit organizations	1985	No
aero	Air transport	2001	Yes
biz	Businesses	2001	No
coop	Cooperatives	2001	Yes
info	Informational	2002	No
museum	Museums	2002	Yes
name	People	2002	No
pro	Professionals	2002	Yes
cat	Catalan	2005	Yes
jobs	Employment	2005	Yes
mobi	Mobile devices	2005	Yes
tel	Contact details	2005	Yes
travel	Travel industry	2005	Yes
xxx	Sex industry	2010	No

Figure 7-2. Generic top-level domains.

Domain Resource Records

- Every domain, whether it is a single host or a top-level domain, can have a set of resource records associated with it. These records are **the DNS database**.
- For a single host, the most common resource record is just its IP address, but many other kinds of resource records also exist.
- When a resolver gives a domain name to DNS, what it gets back are the resource records associated with that name.
  - The primary function of DNS is to map **domain names** onto **resource records**.

Domain Resource Records

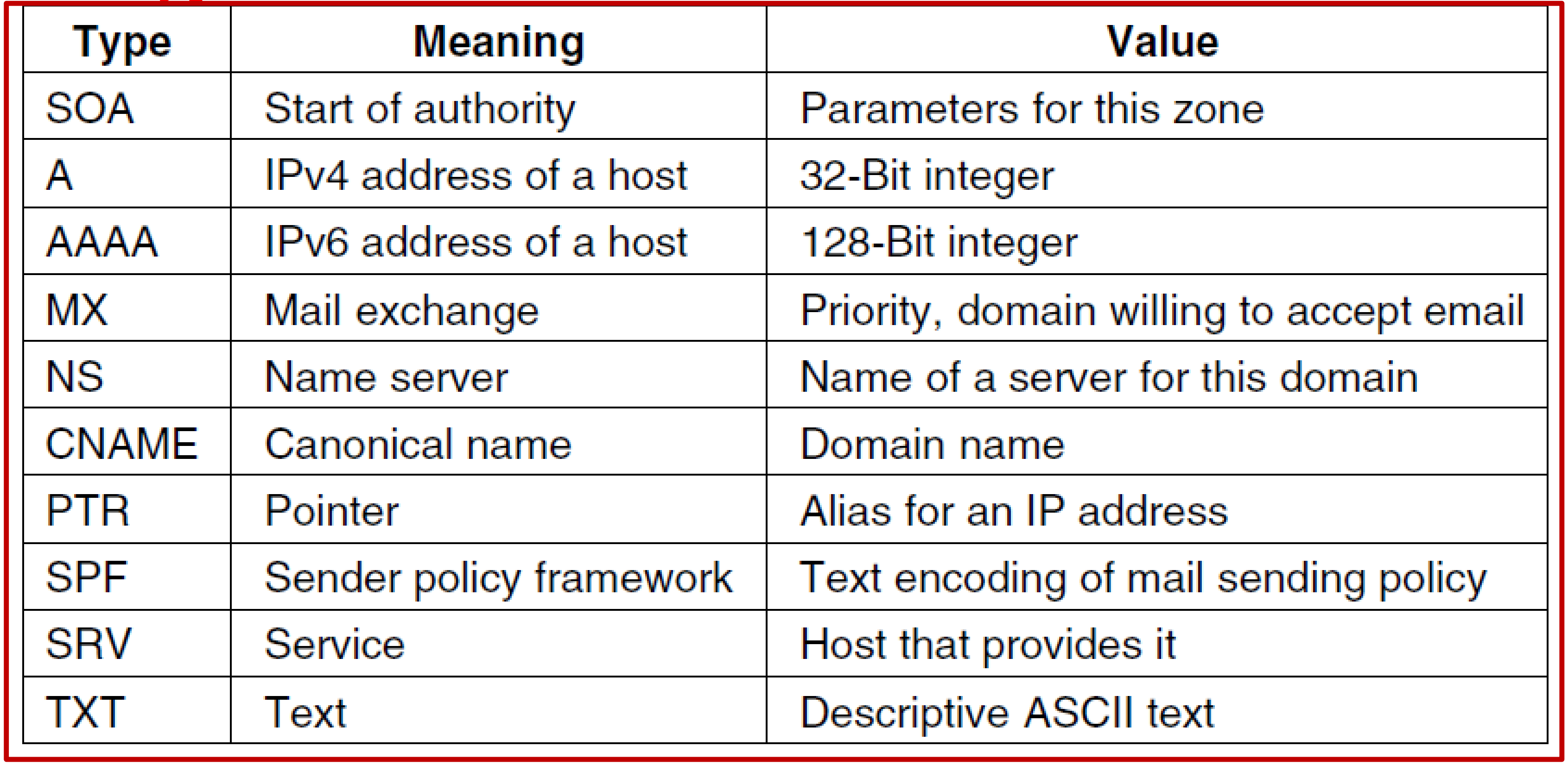
- A resource record is a five-tuple. The format is as follows:

Domain_name Time_to_live Class Type Value

- 1) The Domain_name tells the domain to which this record applies. Normally, many records exist for each domain and each copy of the database holds information about multiple domains. This field is thus **the primary search key** used to satisfy queries.
- 2) The Time_to_live field gives an indication of how **stable** the record is. Information that is highly stable is assigned a large value; information that is highly volatile is assigned a small value.

Domain Resource Records

- 3) The Class field. For Internet information, it is always IN. For non-Internet information, other codes can be used, but in practice these are rarely seen.
- 4) The Type field



Type	Meaning	Value
SOA	Start of authority	Parameters for this zone
A	IPv4 address of a host	32-Bit integer
AAAA	IPv6 address of a host	128-Bit integer
MX	Mail exchange	Priority, domain willing to accept email
NS	Name server	Name of a server for this domain
CNAME	Canonical name	Domain name
PTR	Pointer	Alias for an IP address
SPF	Sender policy framework	Text encoding of mail sending policy
SRV	Service	Host that provides it
TXT	Text	Descriptive ASCII text

Figure 7-3. The principal DNS resource record types.

Domain Resource Records

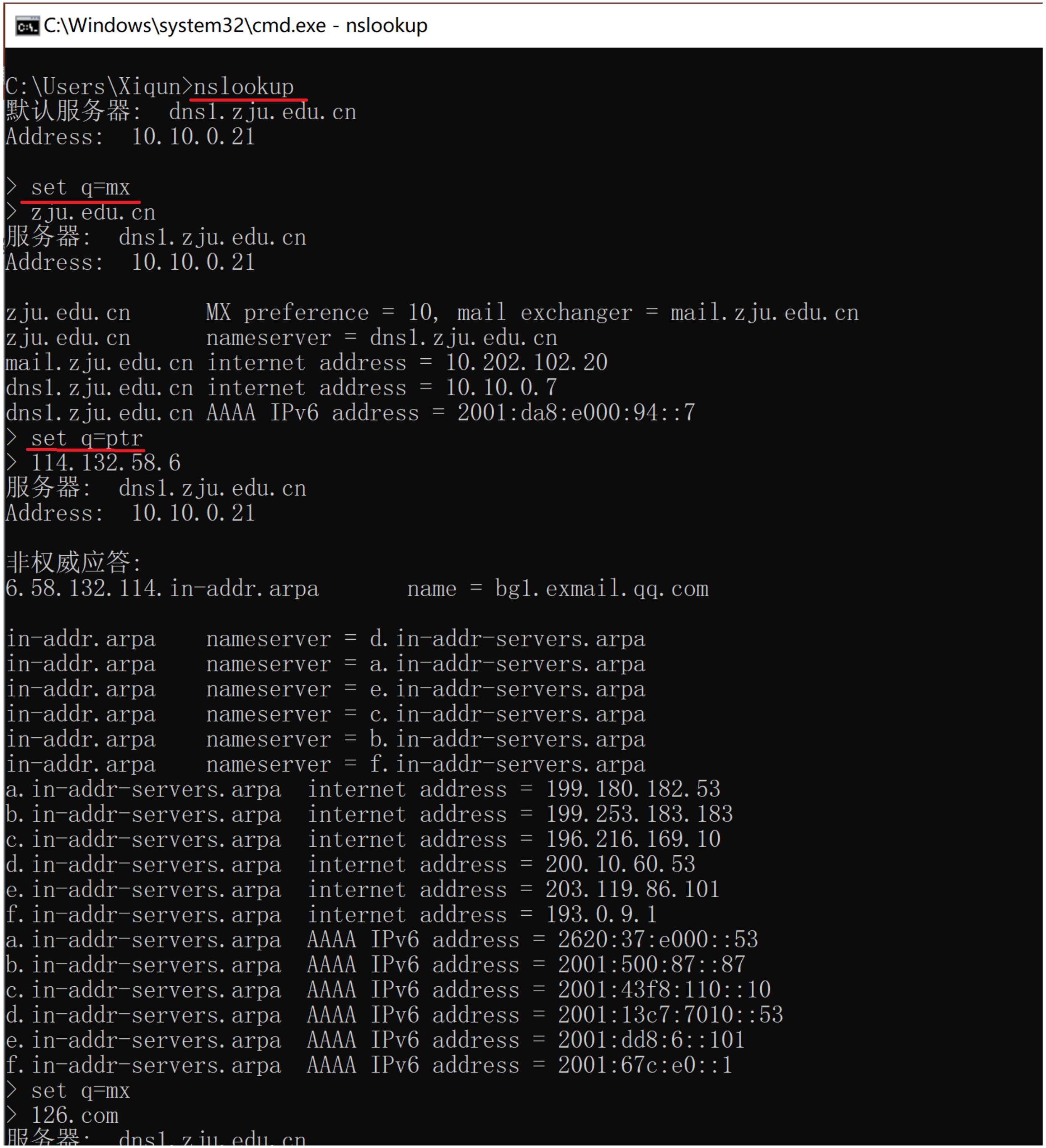
- An SOA record provides the name of the primary source of information about the name server's zone.
- The A (Address) record: a 32 bit IPv4 address of an interface for some host.
- The corresponding AAAA, or "quad A" record holds a 128-bit IPv6 address.
- The MX record, it specifies the name of the host prepared to accept email for the specific domain.
- The NS record specifies a name server for the domain or subdomain. This is a host that has a copy of the database for a domain.

Domain Resource Records

- CNAME records allow aliases to be created (macro definition).[7]
- PTR points to another name. it is nearly always used to associated a name with an IP address to allow lookups of the IP address and return the name of the corresponding machine. — **reverse lookups**
- SRV is a newer type of record that allows a host to be identified for a given service in a domain. This record generalizes the MX record that performs the same task but it is just for mail servers.
- SPF is also a newer type of record. It lets a domain encode information about what machines in the domain will send mail to the rest of the Internet. This helps receiving machines check that mail is valid.

Domain Resource Records

- TXT records were originally provided to allow domains to identify themselves in arbitrary ways.
- 5) The Value field. This field can be a number, a domain name, or an ASCII string. The semantics depends on the record type.



C:\Windows\system32\cmd.exe - nslookup

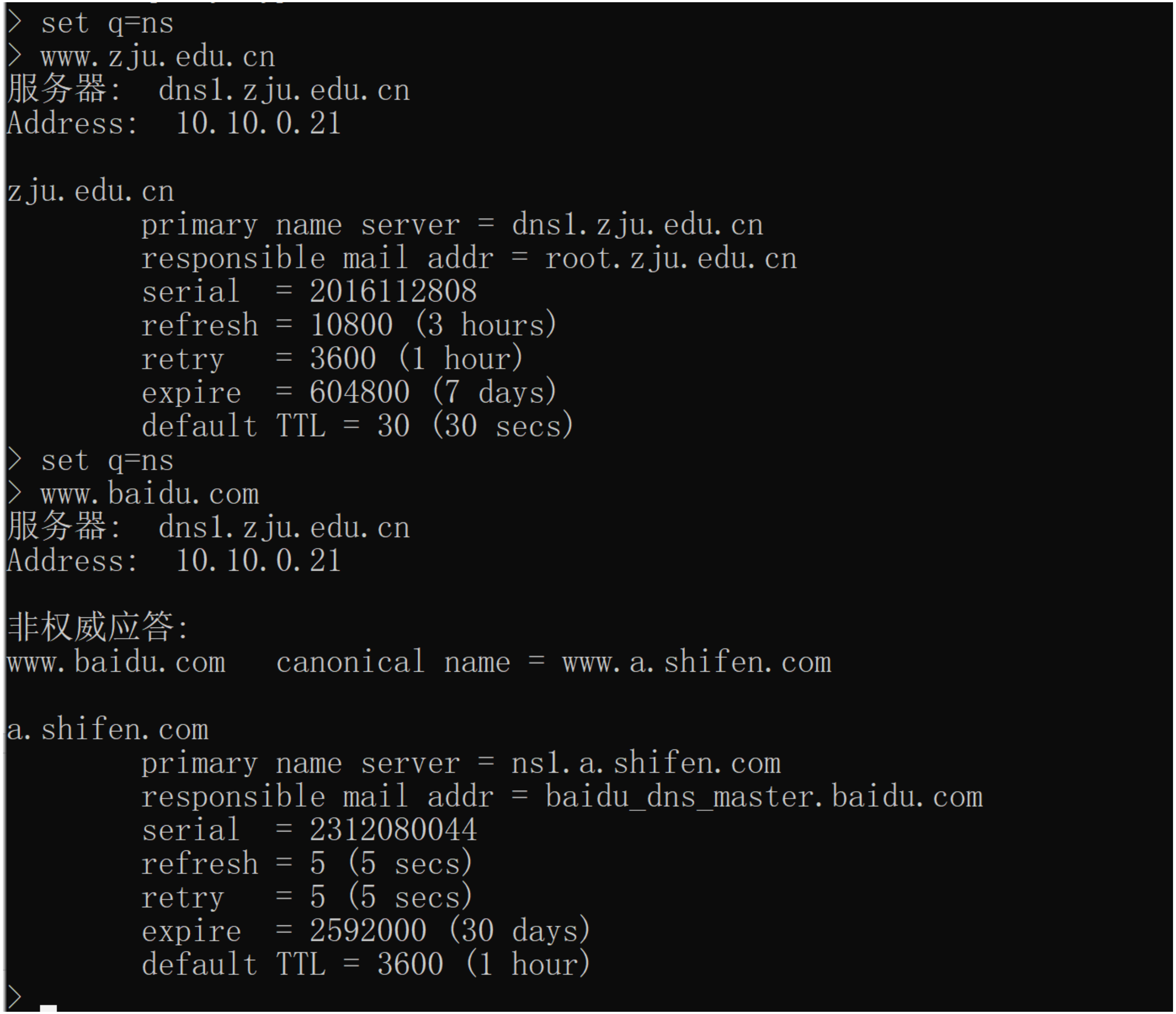
```
C:\Users\Xiqun>nslookup
默认服务器:  dns1.zju.edu.cn
Address:  10.10.0.21

> set q=mx
> zju.edu.cn
服务器:  dns1.zju.edu.cn
Address:  10.10.0.21

zju.edu.cn      MX preference = 10, mail exchanger = mail.zju.edu.cn
zju.edu.cn      nameserver = dns1.zju.edu.cn
mail.zju.edu.cn internet address = 10.202.102.20
dns1.zju.edu.cn internet address = 10.10.0.7
dns1.zju.edu.cn AAAA IPv6 address = 2001:da8:e000:94::7
> set q=ptr
> 114.132.58.6
服务器:  dns1.zju.edu.cn
Address:  10.10.0.21

非权威应答:
6.58.132.114.in-addr.arpa       name = bg1.exmail.qq.com

in-addr.arpa    nameserver = d.in-addr-servers.arpa
in-addr.arpa    nameserver = a.in-addr-servers.arpa
in-addr.arpa    nameserver = e.in-addr-servers.arpa
in-addr.arpa    nameserver = c.in-addr-servers.arpa
in-addr.arpa    nameserver = b.in-addr-servers.arpa
in-addr.arpa    nameserver = f.in-addr-servers.arpa
a.in-addr-servers.arpa  internet address = 199.180.182.53
b.in-addr-servers.arpa  internet address = 199.253.183.183
c.in-addr-servers.arpa  internet address = 196.216.169.10
d.in-addr-servers.arpa  internet address = 200.10.60.53
e.in-addr-servers.arpa  internet address = 203.119.86.101
f.in-addr-servers.arpa  internet address = 193.0.9.1
a.in-addr-servers.arpa  AAAA IPv6 address = 2620:37:e000::53
b.in-addr-servers.arpa  AAAA IPv6 address = 2001:500:87::87
c.in-addr-servers.arpa  AAAA IPv6 address = 2001:43f8:110::10
d.in-addr-servers.arpa  AAAA IPv6 address = 2001:13c7:7010::53
e.in-addr-servers.arpa  AAAA IPv6 address = 2001:dd8:6::101
f.in-addr-servers.arpa  AAAA IPv6 address = 2001:67c:e0::1
> set q=mx
> 126.com
服务器:  dns1.zju.edu.cn
```



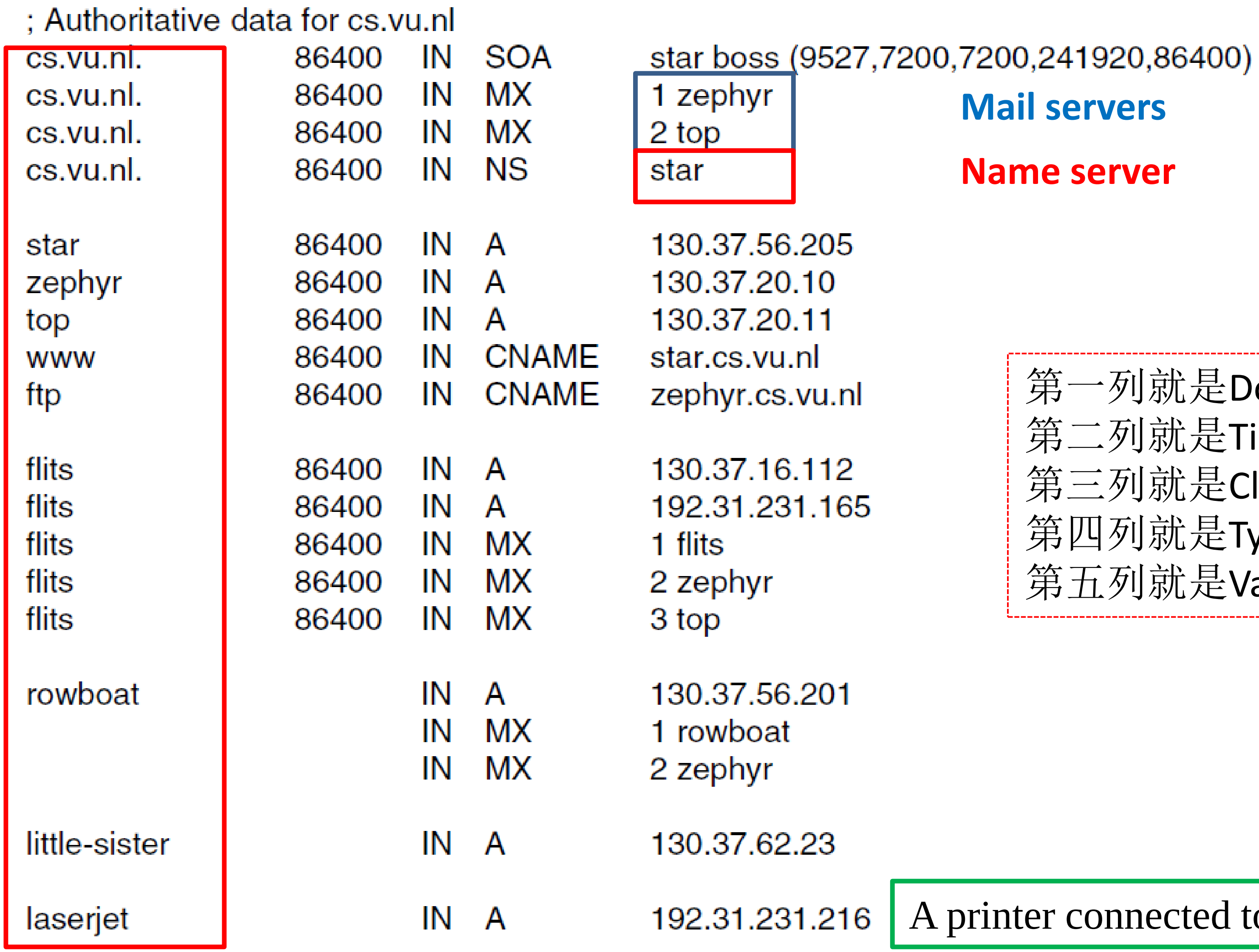
```
> set q=ns
> www.zju.edu.cn
服务器:  dns1.zju.edu.cn
Address:  10.10.0.21

zju.edu.cn
        primary name server = dns1.zju.edu.cn
        responsible mail addr = root.zju.edu.cn
        serial  = 2016112808
        refresh = 10800 (3 hours)
        retry   = 3600 (1 hour)
        expire  = 604800 (7 days)
        default TTL = 30 (30 secs)
> set q=ns
> www.baidu.com
服务器:  dns1.zju.edu.cn
Address:  10.10.0.21

非权威应答:
www.baidu.com   canonical name = www.a.shifen.com

a.shifen.com
        primary name server = ns1.a.shifen.com
        responsible mail addr = baidu_dns_master.baidu.com
        serial  = 2312080044
        refresh = 5 (5 secs)
        retry   = 5 (5 secs)
        expire  = 2592000 (30 days)
        default TTL = 3600 (1 hour)
> _
```

Domain Resource Records



```
; Authoritative data for cs.vu.nl
cs.vu.nl.       86400  IN  SOA    star boss (9527,7200,7200,241920,86400)
cs.vu.nl.       86400  IN  MX     1 zephyr
cs.vu.nl.       86400  IN  MX     2 top
cs.vu.nl.       86400  IN  NS     star

star            86400  IN  A      130.37.56.205
zephyr          86400  IN  A      130.37.20.10
top             86400  IN  A      130.37.20.11
www             86400  IN  CNAME  star.cs.vu.nl
ftp             86400  IN  CNAME  zephyr.cs.vu.nl

flits           86400  IN  A      130.37.16.112
flits           86400  IN  A      192.31.231.165
flits           86400  IN  MX     1 flits
flits           86400  IN  MX     2 zephyr
flits           86400  IN  MX     3 top

rowboat                IN  A      130.37.56.201
                       IN  MX     1 rowboat
                       IN  MX     2 zephyr

little-sister          IN  A      130.37.62.23

laserjet               IN  A      192.31.231.216
```

Figure 7-4. A portion of a possible DNS database for *cs.vu.nl*.

Mail servers

Name server

第一列就是Domain_name,
第二列就是Time_to_live，
第三列就是Class，
第四列就是Type，
第五列就是Value

A printer connected to the Internet

DNS Resolution

- DNS protocol lets a host resolve any host name (domain) to IP address
- If unknown, can start with the root nameserver and work down zones.

Example: flits.cs.vu.nl resolves robot.cs.washington.edu

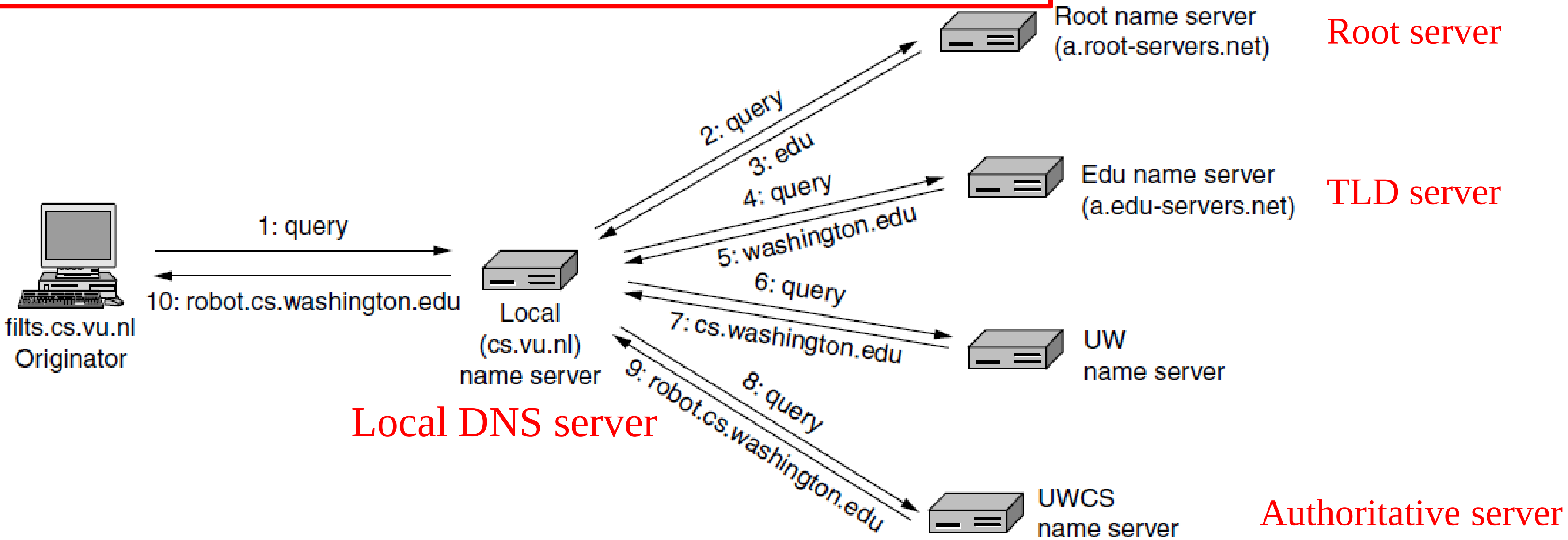


Figure 7-6. Example of a resolver looking up a remote name in 10 steps.

Iterative Queries (I)

- **Definition**: An iterative DNS query is a process in which the DNS resolver (usually a client - side DNS server) makes a series of requests to different DNS servers until it gets the answer it needs. When a DNS resolver sends an iterative query, it starts from the root DNS servers.
- Process Example: Suppose a local DNS resolver wants to resolve the domain name "www.cs.washington.edu". see Fig. 7-6 the right part behind the local DNS server.
- 1) The local DNS server first contacts a root DNS server. But The root DNS server doesn't know the IP address of www.cs.washington.edu directly, but it knows the IP addresses of the top-level domain (TLD) servers (such as .edu servers). So, it responds to the local DNS resolver with the IP addresses of the relevant TLD servers.

Iterative Queries (II)

- **Definition**: An iterative DNS query is a process in which the DNS resolver (usually a client - side DNS server) makes a series of requests to different DNS servers until it gets the answer it needs. When a DNS resolver sends an iterative query, it starts from the root DNS servers.
- Process Example: Suppose a local DNS resolver wants to resolve the domain name "www.cs.washington.edu". see Fig. 7-6 the right part behind the local DNS server.
- 2) The local DNS resolver then contacts the TLD server. The TLD server, in turn, provides the IP addresses of the authoritative DNS servers for the domain "washtington.edu".
- 3) Finally, the local DNS resolver contacts the authoritative DNS server, which provides the IP address of "www.cs.washington.edu".

Iterative Queries (III)

- Advantages:
  - It reduces the load on DNS servers other than the root and TLD servers because the local DNS resolver is doing most of the work in terms of following up on the referrals.
  - The local DNS server can cache over a pool of clients for better performance
- Disadvantages:
  - The process can be **slower** because the resolver has to make multiple requests and wait for responses from different servers. Also, each step in the process may introduce additional latency.

Recursive Queries (I)

- Definition: A recursive DNS query is a query where the DNS resolver (usually a client - side DNS server) asks another DNS server to handle the entire resolution process. The client DNS server sends a single query to a recursive DNS server and waits for the final answer.
- Process Example: see Fig. 7-6 the left part between the client and the local DNS server (the recursive DNS server).

Recursive Queries (II)

- Advantages:
  - It simplifies the process for the client - side DNS resolver. The resolver only needs to send a single query and wait for the response, without having to handle referrals or make multiple requests.
  - It can potentially provide a faster response because the recursive DNS server can optimize the query process by using its **cache**. If it has previously resolved the domain name or has relevant information in its cache, it can return the answer more quickly.

Recursive Queries (III)

- Disadvantages:
  - Recursive DNS servers can be *overloaded* if they receive a large number of requests, especially if they have to perform a full resolution process for each query.
  - There is a *security risk* associated with recursive DNS servers. If a malicious actor can control a recursive DNS server, they can manipulate the results of DNS resolutions, leading to issues such as *phishing* (网络钓鱼) or *redirecting* users to malicious websites.

DNS Caching vs. Freshness

- Caching reduces DNS resolution latency
  - Previous resolutions cut out most of the process
- Caching reduces server load
- Caching delays updates
- The cache will expire after some time
  - Information is cached between 5 minutes and 72 hours (TTL: Time-to-Live)
- Update/notify mechanism is defined by IETF RFC 2136

Local Nameservers

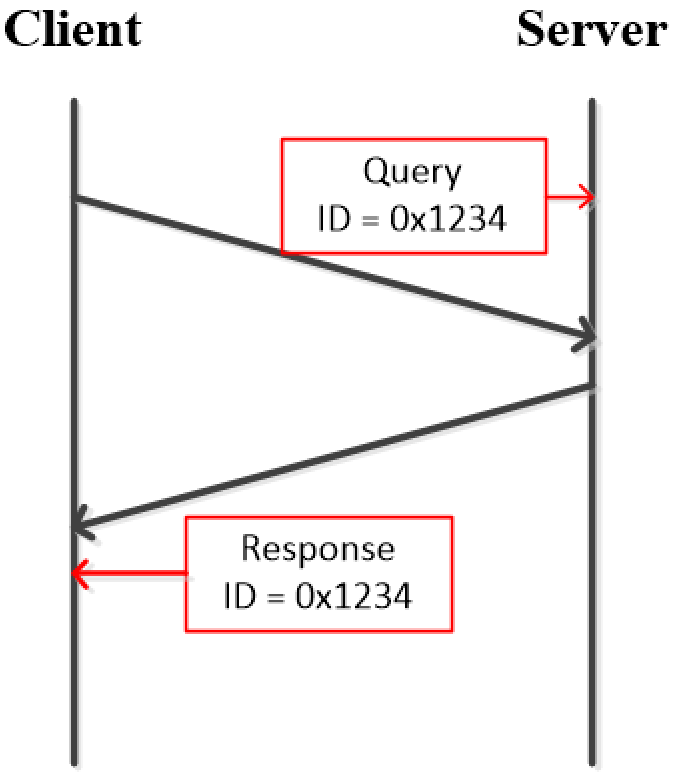
- Local nameservers typically run by IT (enterprise, university, ISP)
  - But may be your host or AP
  - Or alternatives, e.g. , Google public DNS
- Clients need to be able to contact their local nameservers
  - Typically configured via DHCP

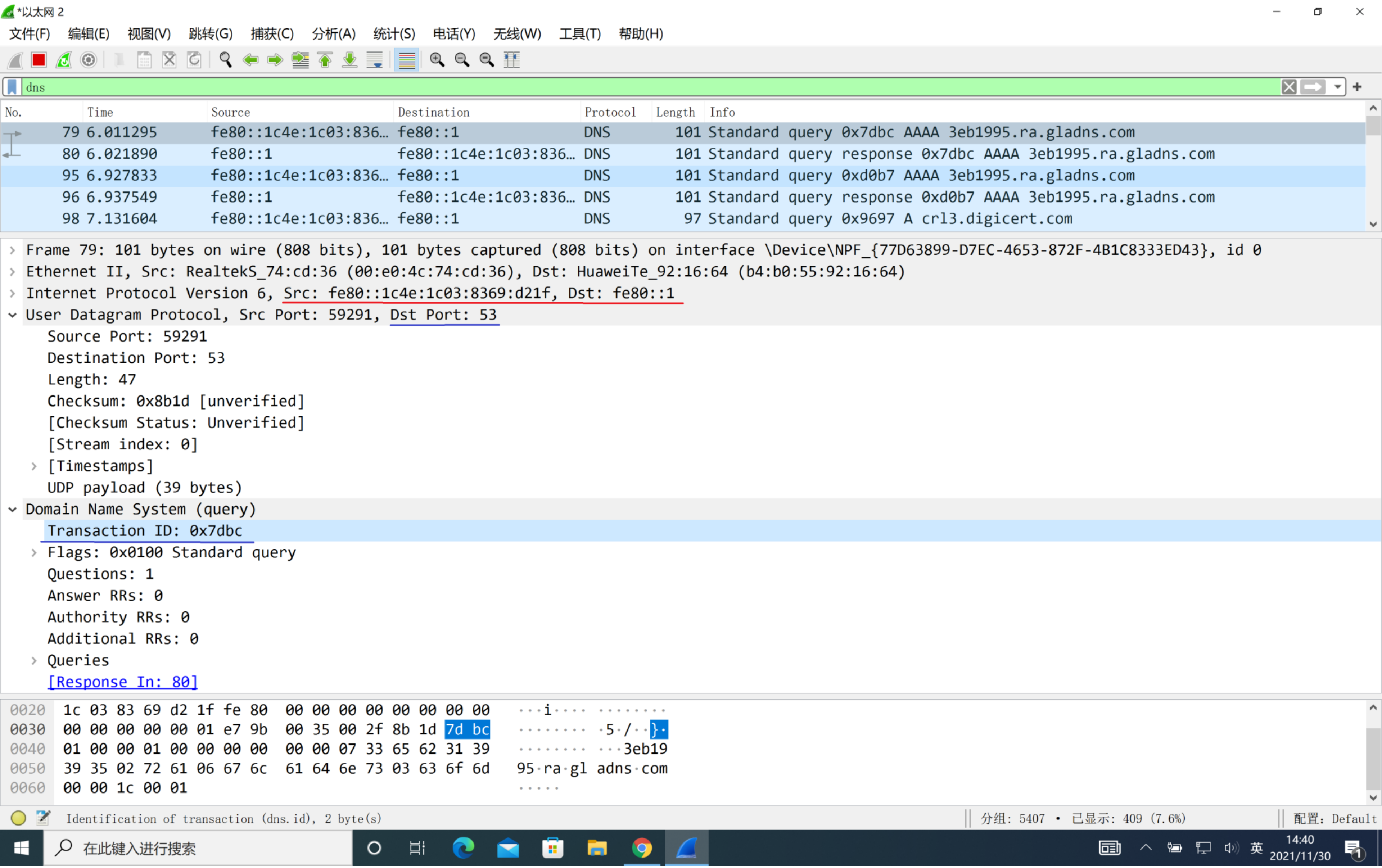
Name Servers

- There are 13 root DNS servers. Most of the root servers are present in multiple geographical locations and reached using **anycast routing**, in which a packet is delivered to the nearest instance of a destination address.
- Running on top of **UDP**
- Port number: **53**
- User utilities: dig, http://www.netliner.com/dig.html (UNIX), "nslookup" (Windows)

DNS Protocol

- Query and response messages
  - Built on **UDP** messages, port **53**
  - ARQ for reliability; server is stateless!
  - Messages linked by **a 16-bit ID field**
- Service reliability via replicas
  - Run multiple nameservers for domain
  - Return the list; clients use one answer
  - Help to distribute load too.
- Security is a major issue
  - Not part of initial protocols
  - DNSSEC (DNS Security Extensions)




*以太网 2
文件(F) 编辑(E) 视图(V) 跳转(G) 捕获(C) 分析(A) 统计(S) 电话(Y) 无线(W) 工具(T) 帮助(H)
dns
在此键入进行搜索
英
14:40
2021/11/30

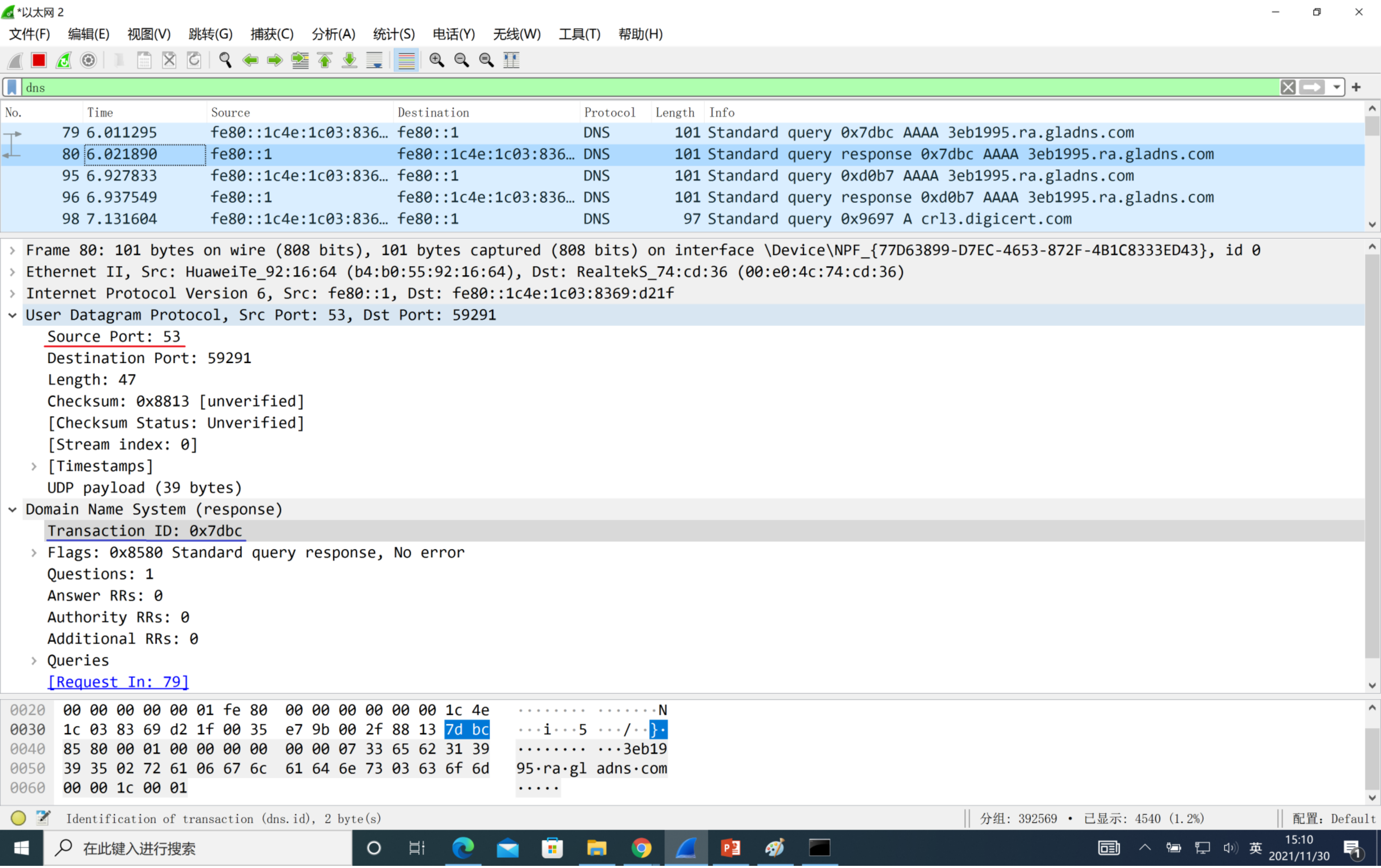
```
No.  Time      Source                  Destination             Protocol  Length Info
  79 6.011295  fe80::1c4e:1c03:836…    fe80::1                 DNS          101 Standard query 0x7dbc AAAA 3eb1995.ra.gladns.com
  80 6.021890  fe80::1                 fe80::1c4e:1c03:836…    DNS          101 Standard query response 0x7dbc AAAA 3eb1995.ra.gladns.com
  95 6.927833  fe80::1c4e:1c03:836…    fe80::1                 DNS          101 Standard query 0xd0b7 AAAA 3eb1995.ra.gladns.com
  96 6.937549  fe80::1                 fe80::1c4e:1c03:836…    DNS          101 Standard query response 0xd0b7 AAAA 3eb1995.ra.gladns.com
  98 7.131604  fe80::1c4e:1c03:836…    fe80::1                 DNS           97 Standard query 0x9697 A crl3.digicert.com
```

```
> Frame 79: 101 bytes on wire (808 bits), 101 bytes captured (808 bits) on interface \Device\NPF_{77D63899-D7EC-4653-872F-4B1C8333ED43}, id 0
> Ethernet II, Src: RealtekS_74:cd:36 (00:e0:4c:74:cd:36), Dst: HuaweiTe_92:16:64 (b4:b0:55:92:16:64)
> Internet Protocol Version 6, Src: fe80::1c4e:1c03:8369:d21f, Dst: fe80::1
v User Datagram Protocol, Src Port: 59291, Dst Port: 53
    Source Port: 59291
    Destination Port: 53
    Length: 47
    Checksum: 0x8b1d [unverified]
    [Checksum Status: Unverified]
    [Stream index: 0]
  > [Timestamps]
    UDP payload (39 bytes)
v Domain Name System (query)
    Transaction ID: 0x7dbc
  > Flags: 0x0100 Standard query
    Questions: 1
    Answer RRs: 0
    Authority RRs: 0
    Additional RRs: 0
  > Queries
    [Response In: 80]
```

```
0020  1c 03 83 69 d2 1f fe 80  00 00 00 00 00 00 00 00   ···i···· ········
0030  00 00 00 00 00 01 e7 9b  00 35 00 2f 8b 1d 7d bc   ········ ·5·/··}·
0040  01 00 00 01 00 00 00 00  00 00 07 33 65 62 31 39   ········ ···3eb19
0050  39 35 02 72 61 06 67 6c  61 64 6e 73 03 63 6f 6d   95·ra·gl adns·com
0060  00 00 1c 00 01                                     ·····
```

```
Identification of transaction (dns.id), 2 byte(s)    分组: 5407 · 已显示: 409 (7.6%)    配置: Default
```

DNS query Example: 2021年11月30日在家捕获的一个DNS的query包，传输层为UDP，目的端口号为53，Transaction ID：0x7dbc (16 bit)


*以太网 2
文件(F) 编辑(E) 视图(V) 跳转(G) 捕获(C) 分析(A) 统计(S) 电话(Y) 无线(W) 工具(T) 帮助(H)
在此键入进行搜索
英
15:10
2021/11/30

dns

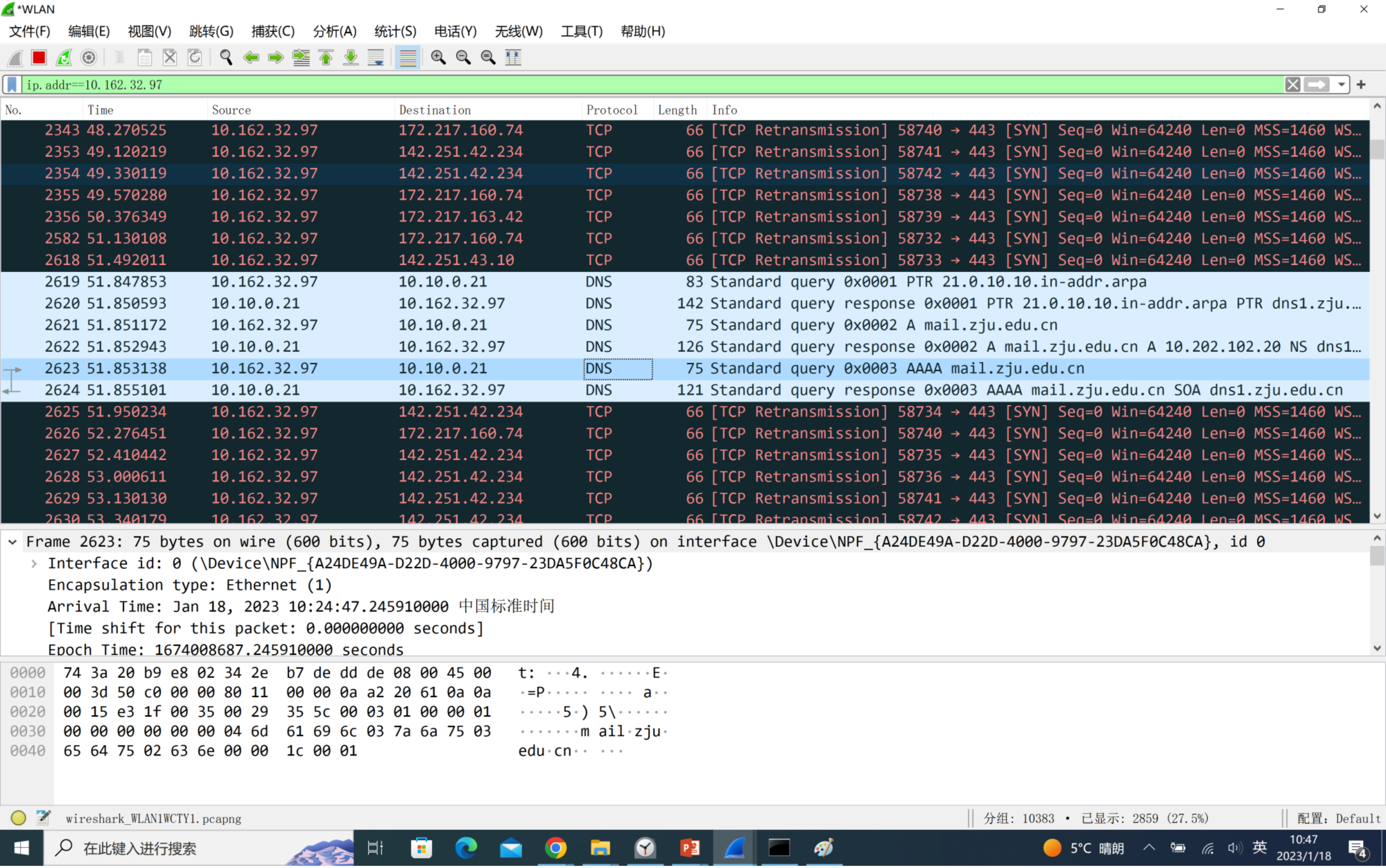
```
No.   Time      Source                Destination           Protocol  Length Info
   79 6.011295  fe80::1c4e:1c03:836…  fe80::1               DNS          101 Standard query 0x7dbc AAAA 3eb1995.ra.gladns.com
   80 6.021890  fe80::1               fe80::1c4e:1c03:836…  DNS          101 Standard query response 0x7dbc AAAA 3eb1995.ra.gladns.com
   95 6.927833  fe80::1c4e:1c03:836…  fe80::1               DNS          101 Standard query 0xd0b7 AAAA 3eb1995.ra.gladns.com
   96 6.937549  fe80::1               fe80::1c4e:1c03:836…  DNS          101 Standard query response 0xd0b7 AAAA 3eb1995.ra.gladns.com
   98 7.131604  fe80::1c4e:1c03:836…  fe80::1               DNS           97 Standard query 0x9697 A crl3.digicert.com
```

```
> Frame 80: 101 bytes on wire (808 bits), 101 bytes captured (808 bits) on interface \Device\NPF_{77D63899-D7EC-4653-872F-4B1C8333ED43}, id 0
> Ethernet II, Src: HuaweiTe_92:16:64 (b4:b0:55:92:16:64), Dst: RealtekS_74:cd:36 (00:e0:4c:74:cd:36)
> Internet Protocol Version 6, Src: fe80::1, Dst: fe80::1c4e:1c03:8369:d21f
v User Datagram Protocol, Src Port: 53, Dst Port: 59291
    Source Port: 53
    Destination Port: 59291
    Length: 47
    Checksum: 0x8813 [unverified]
    [Checksum Status: Unverified]
    [Stream index: 0]
  > [Timestamps]
    UDP payload (39 bytes)
v Domain Name System (response)
    Transaction ID: 0x7dbc
  > Flags: 0x8580 Standard query response, No error
    Questions: 1
    Answer RRs: 0
    Authority RRs: 0
    Additional RRs: 0
  > Queries
    [Request In: 79]
```

```
0020  00 00 00 00 00 01 fe 80  00 00 00 00 00 00 1c 4e   ········ ·······N
0030  1c 03 83 69 d2 1f 00 35  e7 9b 00 2f 88 13 7d bc   ···i···5 ···/··}·
0040  85 80 00 01 00 00 00 00  00 00 07 33 65 62 31 39   ········ ···3eb19
0050  39 35 02 72 61 06 67 6c  61 64 6e 73 03 63 6f 6d   95·ra·gl adns·com
0060  00 00 1c 00 01                                     ·····
```

Identification of transaction (dns.id), 2 byte(s) || 分组：392569 • 已显示：4540 (1.2%) || 配置：Default

DNS response Example: 2021年11月30日在家捕获的一个DNS的response包，传输层为UDP，源端口号为53，Transaction ID：0x7dbc (16 bit)



ip. addr==10. 162. 32. 97

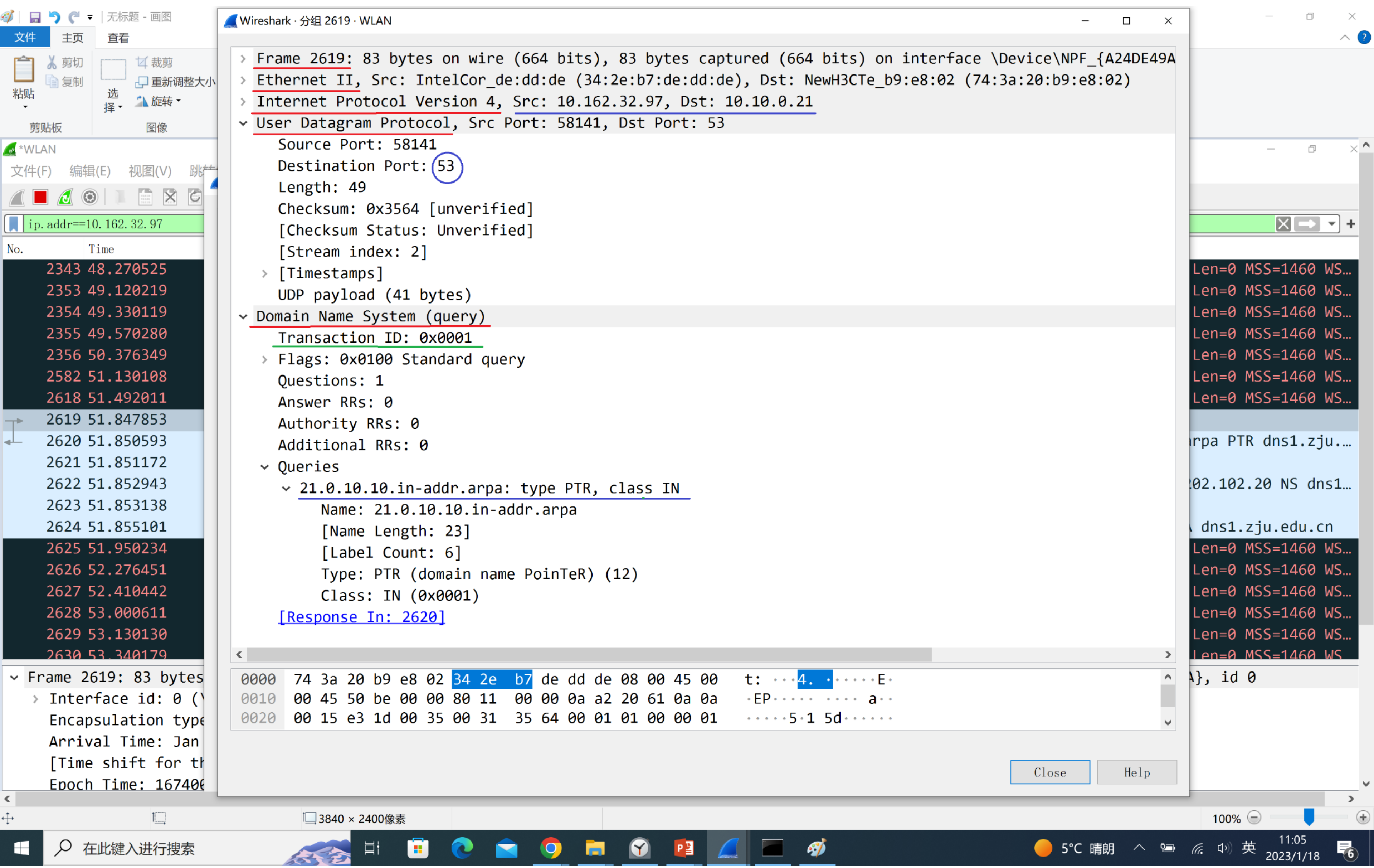
```
No.  Time       Source         Destination     Protocol Length Info
2343 48.270525  10.162.32.97   172.217.160.74  TCP       66 [TCP Retransmission] 58740 → 443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS…
2353 49.120219  10.162.32.97   142.251.42.234  TCP       66 [TCP Retransmission] 58741 → 443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS…
2354 49.330119  10.162.32.97   142.251.42.234  TCP       66 [TCP Retransmission] 58742 → 443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS…
2355 49.570280  10.162.32.97   172.217.160.74  TCP       66 [TCP Retransmission] 58738 → 443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS…
2356 50.376349  10.162.32.97   172.217.163.42  TCP       66 [TCP Retransmission] 58739 → 443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS…
2582 51.130108  10.162.32.97   172.217.160.74  TCP       66 [TCP Retransmission] 58732 → 443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS…
2618 51.492011  10.162.32.97   142.251.43.10   TCP       66 [TCP Retransmission] 58733 → 443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS…
2619 51.847853  10.162.32.97   10.10.0.21      DNS       83 Standard query 0x0001 PTR 21.0.10.10.in-addr.arpa
2620 51.850593  10.10.0.21     10.162.32.97    DNS      142 Standard query response 0x0001 PTR 21.0.10.10.in-addr.arpa PTR dns1.zju.…
2621 51.851172  10.162.32.97   10.10.0.21      DNS       75 Standard query 0x0002 A mail.zju.edu.cn
2622 51.852943  10.10.0.21     10.162.32.97    DNS      126 Standard query response 0x0002 A mail.zju.edu.cn A 10.202.102.20 NS dns1…
2623 51.853138  10.162.32.97   10.10.0.21      DNS       75 Standard query 0x0003 AAAA mail.zju.edu.cn
2624 51.855101  10.10.0.21     10.162.32.97    DNS      121 Standard query response 0x0003 AAAA mail.zju.edu.cn SOA dns1.zju.edu.cn
2625 51.950234  10.162.32.97   142.251.42.234  TCP       66 [TCP Retransmission] 58734 → 443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS…
2626 52.276451  10.162.32.97   172.217.160.74  TCP       66 [TCP Retransmission] 58740 → 443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS…
2627 52.410442  10.162.32.97   142.251.42.234  TCP       66 [TCP Retransmission] 58735 → 443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS…
2628 53.000611  10.162.32.97   142.251.42.234  TCP       66 [TCP Retransmission] 58736 → 443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS…
2629 53.130130  10.162.32.97   142.251.42.234  TCP       66 [TCP Retransmission] 58741 → 443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS…
2630 53.340179  10.162.32.97   142.251.42.234  TCP       66 [TCP Retransmission] 58742 → 443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS
```

```
˅ Frame 2623: 75 bytes on wire (600 bits), 75 bytes captured (600 bits) on interface \Device\NPF_{A24DE49A-D22D-4000-9797-23DA5F0C48CA}, id 0
  › Interface id: 0 (\Device\NPF_{A24DE49A-D22D-4000-9797-23DA5F0C48CA})
    Encapsulation type: Ethernet (1)
    Arrival Time: Jan 18, 2023 10:24:47.245910000 中国标准时间
    [Time shift for this packet: 0.000000000 seconds]
    Epoch Time: 1674008687.245910000 seconds
```

```
0000  74 3a 20 b9 e8 02 34 2e  b7 de dd de 08 00 45 00   t: ···4. ······E·
0010  00 3d 50 c0 00 00 80 11  00 00 0a a2 20 61 0a 0a   ·=P····· ···· a··
0020  00 15 e3 1f 00 35 00 29  35 5c 00 03 01 00 00 01   ·····5·) 5\······
0030  00 00 00 00 00 00 04 6d  61 69 6c 03 7a 6a 75 03   ·······m ail·zju·
0040  65 64 75 02 63 6e 00 00  1c 00 01                  edu·cn·· ···
```

wireshark_WLAN1WCTY1. pcapng　　分组: 10383 · 已显示: 2859 (27. 5%)　　配置: Default

通过WireShark捕获的DNS数据包



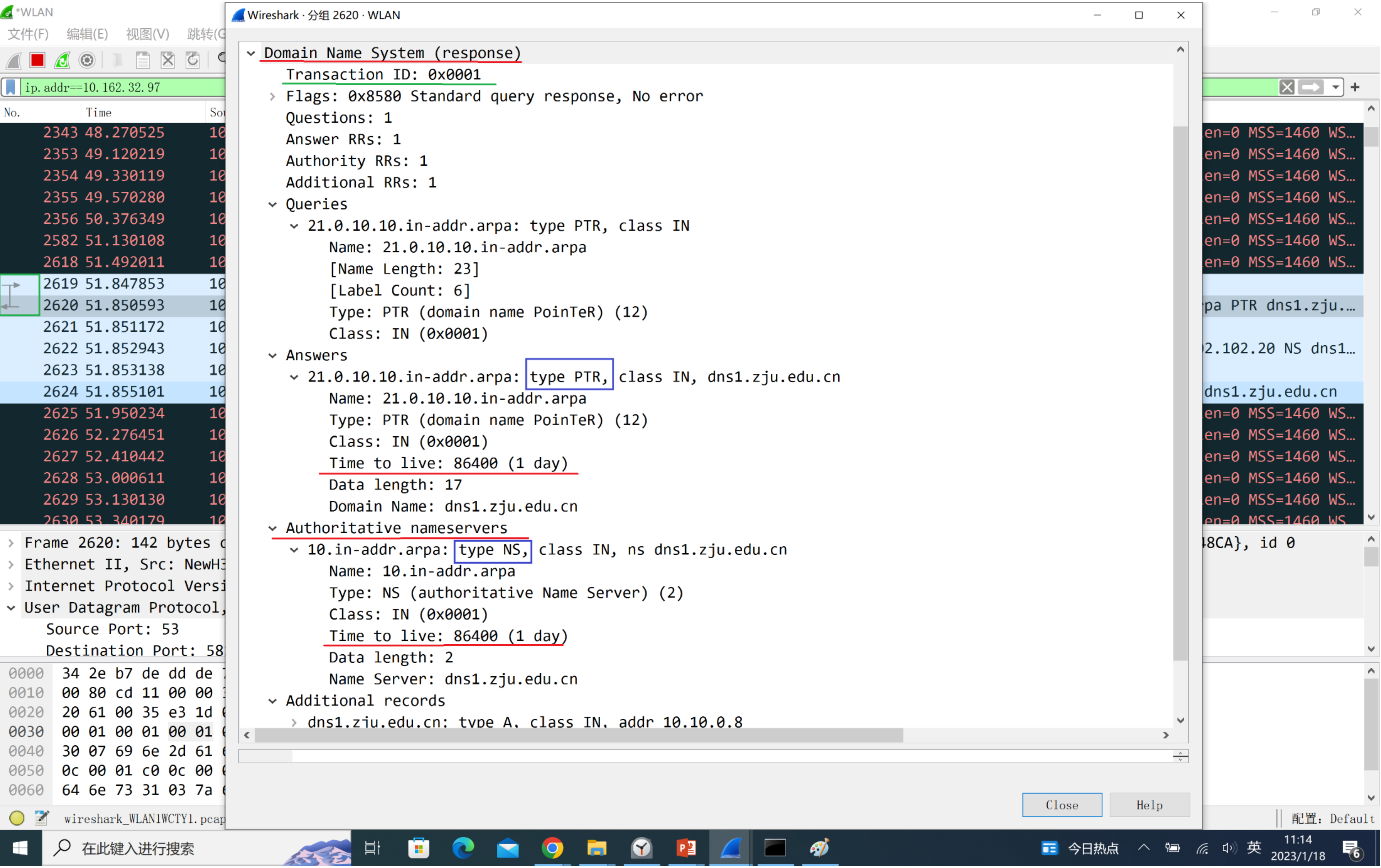
```
> Frame 2619: 83 bytes on wire (664 bits), 83 bytes captured (664 bits) on interface \Device\NPF_{A24DE49A
> Ethernet II, Src: IntelCor_de:dd:de (34:2e:b7:de:dd:de), Dst: NewH3CTe_b9:e8:02 (74:3a:20:b9:e8:02)
> Internet Protocol Version 4, Src: 10.162.32.97, Dst: 10.10.0.21
v User Datagram Protocol, Src Port: 58141, Dst Port: 53
    Source Port: 58141
    Destination Port: 53
    Length: 49
    Checksum: 0x3564 [unverified]
    [Checksum Status: Unverified]
    [Stream index: 2]
  > [Timestamps]
    UDP payload (41 bytes)
v Domain Name System (query)
    Transaction ID: 0x0001
  > Flags: 0x0100 Standard query
    Questions: 1
    Answer RRs: 0
    Authority RRs: 0
    Additional RRs: 0
  v Queries
    v 21.0.10.10.in-addr.arpa: type PTR, class IN
        Name: 21.0.10.10.in-addr.arpa
        [Name Length: 23]
        [Label Count: 6]
        Type: PTR (domain name PointeR) (12)
        Class: IN (0x0001)
    [Response In: 2620]

0000  74 3a 20 b9 e8 02 34 2e  b7 de dd de 08 00 45 00   t: ···4. ·····E·
0010  00 45 50 be 00 00 80 11  00 00 0a a2 20 61 0a 0a   ·EP····· ···· a··
0020  00 15 e3 1d 00 35 00 31  35 64 00 01 01 00 00 01   ·····5·1 5d·····
```

```
2343 48.270525    Len=0 MSS=1460 WS…
2353 49.120219    Len=0 MSS=1460 WS…
2354 49.330119    Len=0 MSS=1460 WS…
2355 49.570280    Len=0 MSS=1460 WS…
2356 50.376349    Len=0 MSS=1460 WS…
2582 51.130108    Len=0 MSS=1460 WS…
2618 51.492011    Len=0 MSS=1460 WS…
2619 51.847853
2620 51.850593    rpa PTR dns1.zju.…
2621 51.851172
2622 51.852943    02.102.20 NS dns1…
2623 51.853138
2624 51.855101    dns1.zju.edu.cn
2625 51.950234    Len=0 MSS=1460 WS…
2626 52.276451    Len=0 MSS=1460 WS…
2627 52.410442    Len=0 MSS=1460 WS…
2628 53.000611    Len=0 MSS=1460 WS…
2629 53.130130    Len=0 MSS=1460 WS…
2630 53.340179    Len=0 MSS=1460 WS…

v Frame 2619: 83 bytes                            }, id 0
  > Interface id: 0 (
    Encapsulation type
    Arrival Time: Jan
    [Time shift for th
    Epoch Time: 167400
```

DNS query 数据包1:注意这里type PTR, class IN, 21.0.10.10.in-addr.arpa (浙大域名解析服务器IP地址顺序是颠倒的！)其中 “in-addr.arpa” 是用于反向 IP 地址解析的顶级域名， 数字部分从右到左依次代表 IP 地址的各个部分， 这样就构建了一个与正向域名解析类似的层次结构， 方便 DNS 服务器进行查找和管理。



Wireshark · 分组 2620 · WLAN

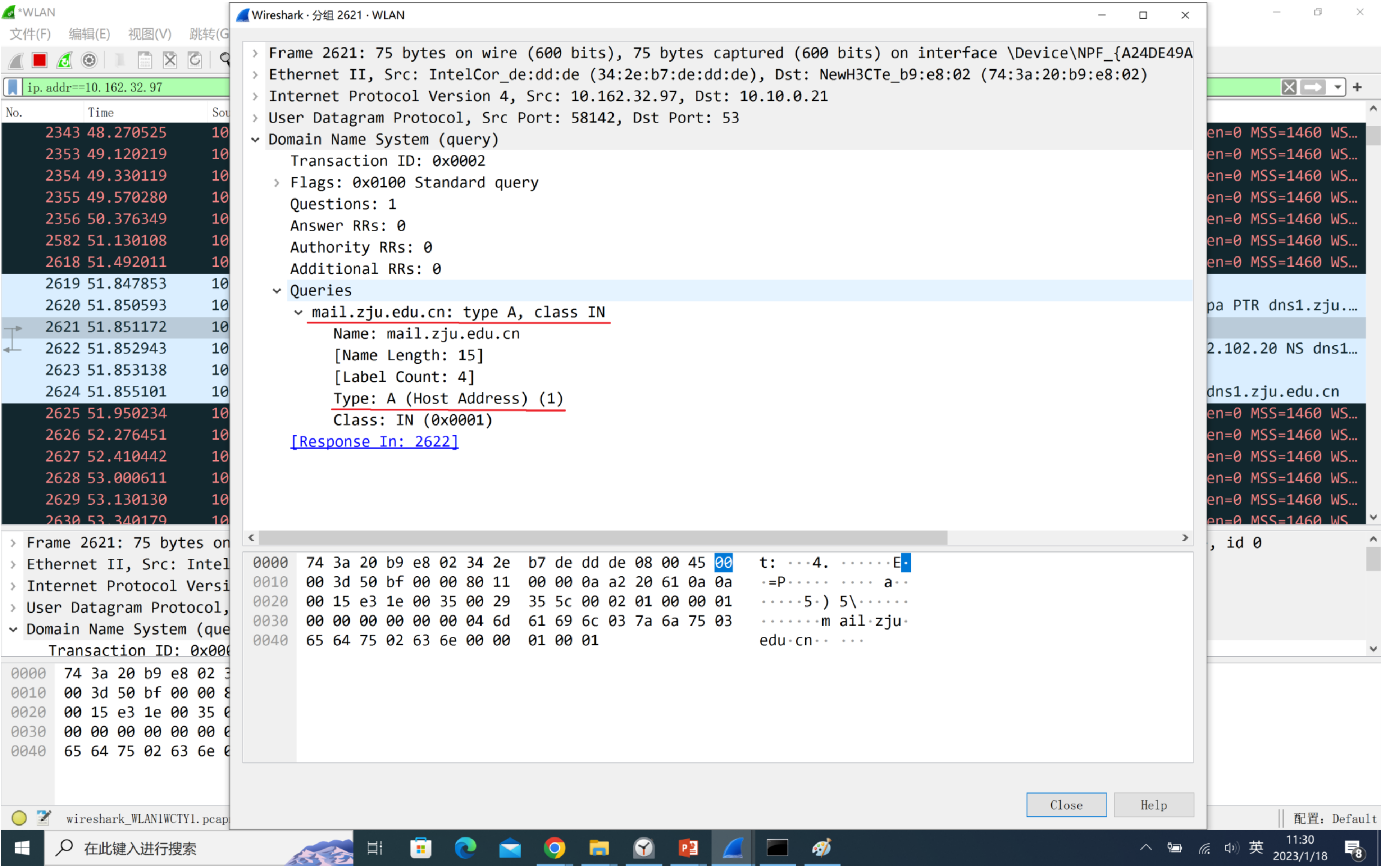
```
⌄ Domain Name System (response)
    Transaction ID: 0x0001
  › Flags: 0x8580 Standard query response, No error
    Questions: 1
    Answer RRs: 1
    Authority RRs: 1
    Additional RRs: 1
  ⌄ Queries
    ⌄ 21.0.10.10.in-addr.arpa: type PTR, class IN
        Name: 21.0.10.10.in-addr.arpa
        [Name Length: 23]
        [Label Count: 6]
        Type: PTR (domain name PoinTeR) (12)
        Class: IN (0x0001)
  ⌄ Answers
    ⌄ 21.0.10.10.in-addr.arpa: type PTR, class IN, dns1.zju.edu.cn
        Name: 21.0.10.10.in-addr.arpa
        Type: PTR (domain name PoinTeR) (12)
        Class: IN (0x0001)
        Time to live: 86400 (1 day)
        Data length: 17
        Domain Name: dns1.zju.edu.cn
  ⌄ Authoritative nameservers
    ⌄ 10.in-addr.arpa: type NS, class IN, ns dns1.zju.edu.cn
        Name: 10.in-addr.arpa
        Type: NS (authoritative Name Server) (2)
        Class: IN (0x0001)
        Time to live: 86400 (1 day)
        Data length: 2
        Name Server: dns1.zju.edu.cn
  ⌄ Additional records
    › dns1.zju.edu.cn: type A, class IN, addr 10.10.0.8
```

```
2343 48.270525  10   en=0 MSS=1460 WS…
2353 49.120219  10   en=0 MSS=1460 WS…
2354 49.330119  10   en=0 MSS=1460 WS…
2355 49.570280  10   en=0 MSS=1460 WS…
2356 50.376349  10   en=0 MSS=1460 WS…
2582 51.130108  10   en=0 MSS=1460 WS…
2618 51.492011  10   en=0 MSS=1460 WS…
2619 51.847853  10
2620 51.850593  10   pa PTR dns1.zju.…
2621 51.851172  10
2622 51.852943  10   2.102.20 NS dns1…
2623 51.853138  10
2624 51.855101  10   dns1.zju.edu.cn
2625 51.950234  10   en=0 MSS=1460 WS…
2626 52.276451  10   en=0 MSS=1460 WS…
2627 52.410442  10   en=0 MSS=1460 WS…
2628 53.000611  10   en=0 MSS=1460 WS…
2629 53.130130  10   en=0 MSS=1460 WS…
2630 53.340179  10   en=0 MSS=1460 WS

› Frame 2620: 142 bytes c                 8CA}, id 0
› Ethernet II, Src: NewH3
› Internet Protocol Versi
⌄ User Datagram Protocol,
    Source Port: 53
    Destination Port: 58

0000  34 2e b7 de dd de
0010  00 80 cd 11 00 00
0020  20 61 00 35 e3 1d
0030  00 01 00 01 00 01
0040  30 07 69 6e 2d 61
0050  0c 00 01 c0 0c 00
0060  64 6e 73 31 03 7a
```

DNS response 数据包1



Wireshark · 分组 2621 · WLAN
*WLAN
文件(F) 编辑(E) 视图(V) 跳转(G
ip. addr==10. 162. 32. 97
Close
Help
wireshark_WLAN1WCTY1. pcap
配置: Default
在此键入进行搜索
英
11:30
2023/1/18

```
No.   Time        Sou
 2343 48.270525   10                      en=0 MSS=1460 WS…
 2353 49.120219   10                      en=0 MSS=1460 WS…
 2354 49.330119   10                      en=0 MSS=1460 WS…
 2355 49.570280   10                      en=0 MSS=1460 WS…
 2356 50.376349   10                      en=0 MSS=1460 WS…
 2582 51.130108   10                      en=0 MSS=1460 WS…
 2618 51.492011   10                      en=0 MSS=1460 WS…
 2619 51.847853   10
 2620 51.850593   10                      pa PTR dns1.zju.…
 2621 51.851172   10
 2622 51.852943   10                      2.102.20 NS dns1…
 2623 51.853138   10
 2624 51.855101   10                      dns1.zju.edu.cn
 2625 51.950234   10                      en=0 MSS=1460 WS…
 2626 52.276451   10                      en=0 MSS=1460 WS…
 2627 52.410442   10                      en=0 MSS=1460 WS…
 2628 53.000611   10                      en=0 MSS=1460 WS…
 2629 53.130130   10                      en=0 MSS=1460 WS…
 2630 53.340179   10                      en=0 MSS=1460 WS
```

```
> Frame 2621: 75 bytes on wire (600 bits), 75 bytes captured (600 bits) on interface \Device\NPF_{A24DE49A
> Ethernet II, Src: IntelCor_de:dd:de (34:2e:b7:de:dd:de), Dst: NewH3CTe_b9:e8:02 (74:3a:20:b9:e8:02)
> Internet Protocol Version 4, Src: 10.162.32.97, Dst: 10.10.0.21
> User Datagram Protocol, Src Port: 58142, Dst Port: 53
v Domain Name System (query)
    Transaction ID: 0x0002
  > Flags: 0x0100 Standard query
    Questions: 1
    Answer RRs: 0
    Authority RRs: 0
    Additional RRs: 0
  v Queries
    v mail.zju.edu.cn: type A, class IN
        Name: mail.zju.edu.cn
        [Name Length: 15]
        [Label Count: 4]
        Type: A (Host Address) (1)
        Class: IN (0x0001)
    [Response In: 2622]
```

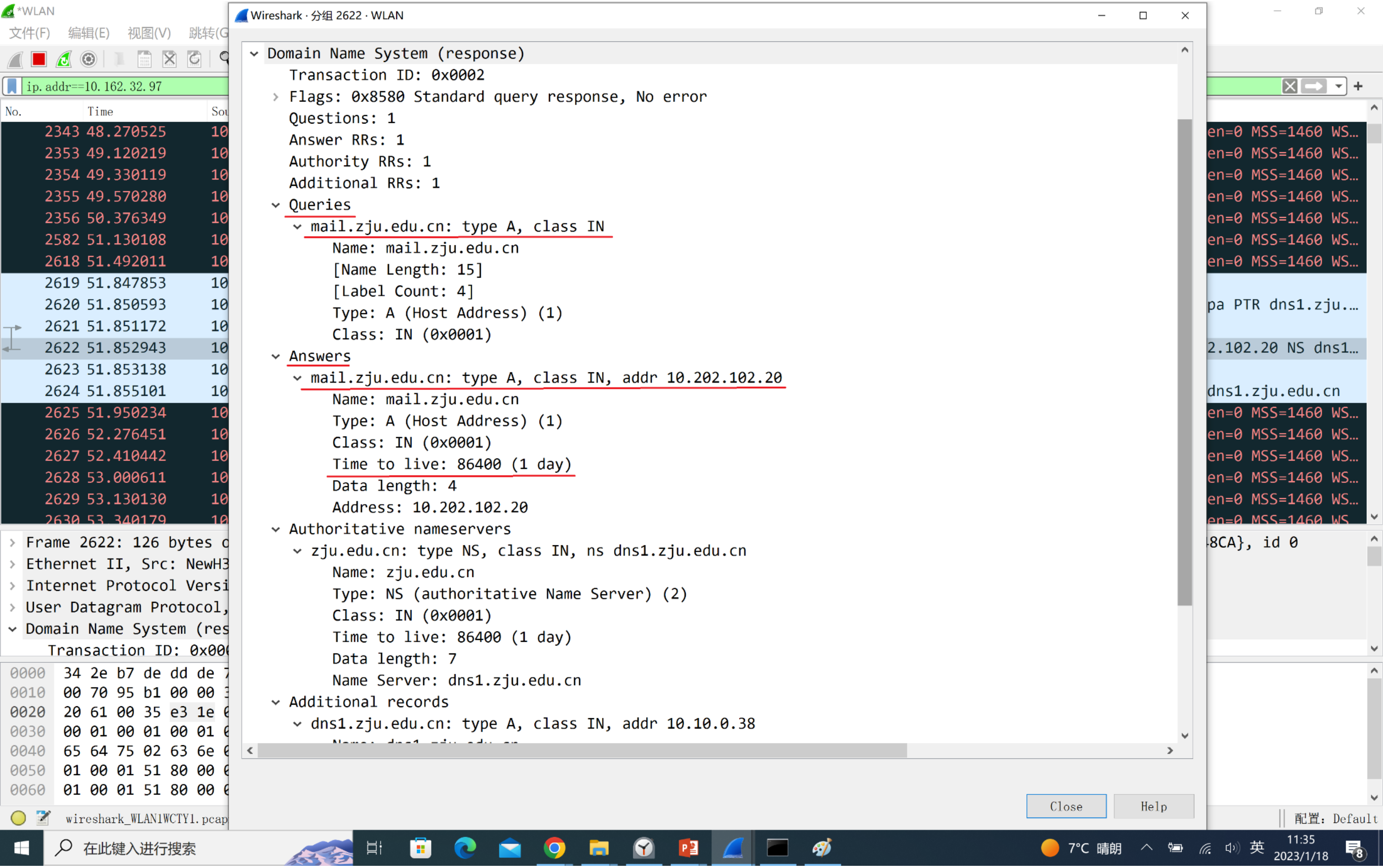
```
0000  74 3a 20 b9 e8 02 34 2e  b7 de dd de 08 00 45 00   t: ···4. ······E·
0010  00 3d 50 bf 00 00 80 11  00 00 0a a2 20 61 0a 0a   ·=P····· ···· a··
0020  00 15 e3 1e 00 35 00 29  35 5c 00 02 01 00 00 01   ·····5·) 5\······
0030  00 00 00 00 00 00 04 6d  61 69 6c 03 7a 6a 75 03   ·······m ail·zju·
0040  65 64 75 02 63 6e 00 00  01 00 01                  edu·cn·· ···
```

```
> Frame 2621: 75 bytes on
> Ethernet II, Src: Intel
> Internet Protocol Versi
> User Datagram Protocol,
v Domain Name System (que
    Transaction ID: 0x000
```

```
0000  74 3a 20 b9 e8 02 3
0010  00 3d 50 bf 00 00 8
0020  00 15 e3 1e 00 35 0
0030  00 00 00 00 00 00 0
0040  65 64 75 02 63 6e 0
```

, id 0

DNS query 数据包2



*WLAN
文件(F) 编辑(E) 视图(V) 跳转(G
Wireshark · 分组 2622 · WLAN
Close
Help
wireshark_WLAN1WCTY1.pcap
配置: Default
在此键入进行搜索
7°C 晴朗
英
11:35
2023/1/18

ip.addr==10.162.32.97

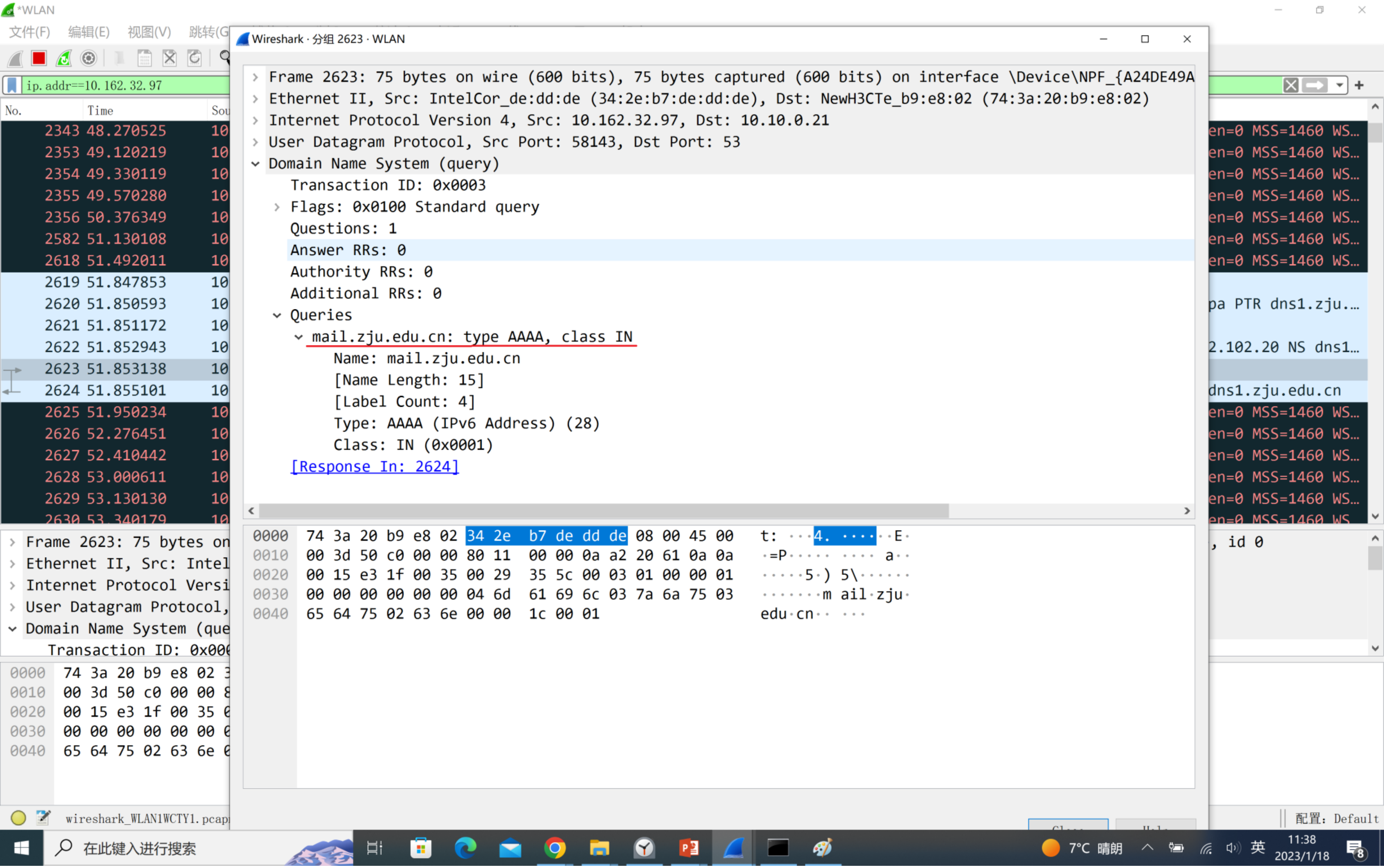
```
No.   Time       Sou
2343 48.270525   10   en=0 MSS=1460 WS…
2353 49.120219   10   en=0 MSS=1460 WS…
2354 49.330119   10   en=0 MSS=1460 WS…
2355 49.570280   10   en=0 MSS=1460 WS…
2356 50.376349   10   en=0 MSS=1460 WS…
2582 51.130108   10   en=0 MSS=1460 WS…
2618 51.492011   10   en=0 MSS=1460 WS…
2619 51.847853   10
2620 51.850593   10   pa PTR dns1.zju.…
2621 51.851172   10
2622 51.852943   10   2.102.20 NS dns1…
2623 51.853138   10
2624 51.855101   10   dns1.zju.edu.cn
2625 51.950234   10   en=0 MSS=1460 WS…
2626 52.276451   10   en=0 MSS=1460 WS…
2627 52.410442   10   en=0 MSS=1460 WS…
2628 53.000611   10   en=0 MSS=1460 WS…
2629 53.130130   10   en=0 MSS=1460 WS…
2630 53.340179   10   en=0 MSS=1460 WS…
```

```
> Frame 2622: 126 bytes o                8CA}, id 0
> Ethernet II, Src: NewH3
> Internet Protocol Versi
> User Datagram Protocol,
v Domain Name System (res
    Transaction ID: 0x00
```

```
0000  34 2e b7 de dd de 7
0010  00 70 95 b1 00 00 3
0020  20 61 00 35 e3 1e 0
0030  00 01 00 01 00 01 0
0040  65 64 75 02 63 6e 0
0050  01 00 01 51 80 00 0
0060  01 00 01 51 80 00 0
```

```
v Domain Name System (response)
    Transaction ID: 0x0002
  > Flags: 0x8580 Standard query response, No error
    Questions: 1
    Answer RRs: 1
    Authority RRs: 1
    Additional RRs: 1
  v Queries
    v mail.zju.edu.cn: type A, class IN
        Name: mail.zju.edu.cn
        [Name Length: 15]
        [Label Count: 4]
        Type: A (Host Address) (1)
        Class: IN (0x0001)
  v Answers
    v mail.zju.edu.cn: type A, class IN, addr 10.202.102.20
        Name: mail.zju.edu.cn
        Type: A (Host Address) (1)
        Class: IN (0x0001)
        Time to live: 86400 (1 day)
        Data length: 4
        Address: 10.202.102.20
  v Authoritative nameservers
    v zju.edu.cn: type NS, class IN, ns dns1.zju.edu.cn
        Name: zju.edu.cn
        Type: NS (authoritative Name Server) (2)
        Class: IN (0x0001)
        Time to live: 86400 (1 day)
        Data length: 7
        Name Server: dns1.zju.edu.cn
  v Additional records
    v dns1.zju.edu.cn: type A, class IN, addr 10.10.0.38
        Name: dns1.zju.edu.cn
```

DNS response 数据包2: 返回mail.zju.edu.cn服务器的ip address 10.202.102.20



```
2343 48.270525  10   en=0 MSS=1460 WS…
2353 49.120219  10   en=0 MSS=1460 WS…
2354 49.330119  10   en=0 MSS=1460 WS…
2355 49.570280  10   en=0 MSS=1460 WS…
2356 50.376349  10   en=0 MSS=1460 WS…
2582 51.130108  10   en=0 MSS=1460 WS…
2618 51.492011  10   en=0 MSS=1460 WS…
2619 51.847853  10
2620 51.850593  10   pa PTR dns1.zju.…
2621 51.851172  10
2622 51.852943  10   2.102.20 NS dns1…
2623 51.853138  10
2624 51.855101  10   dns1.zju.edu.cn
2625 51.950234  10   en=0 MSS=1460 WS…
2626 52.276451  10   en=0 MSS=1460 WS…
2627 52.410442  10   en=0 MSS=1460 WS…
2628 53.000611  10   en=0 MSS=1460 WS…
2629 53.130130  10   en=0 MSS=1460 WS…
2630 53.340179  10   en=0 MSS=1460 WS
```

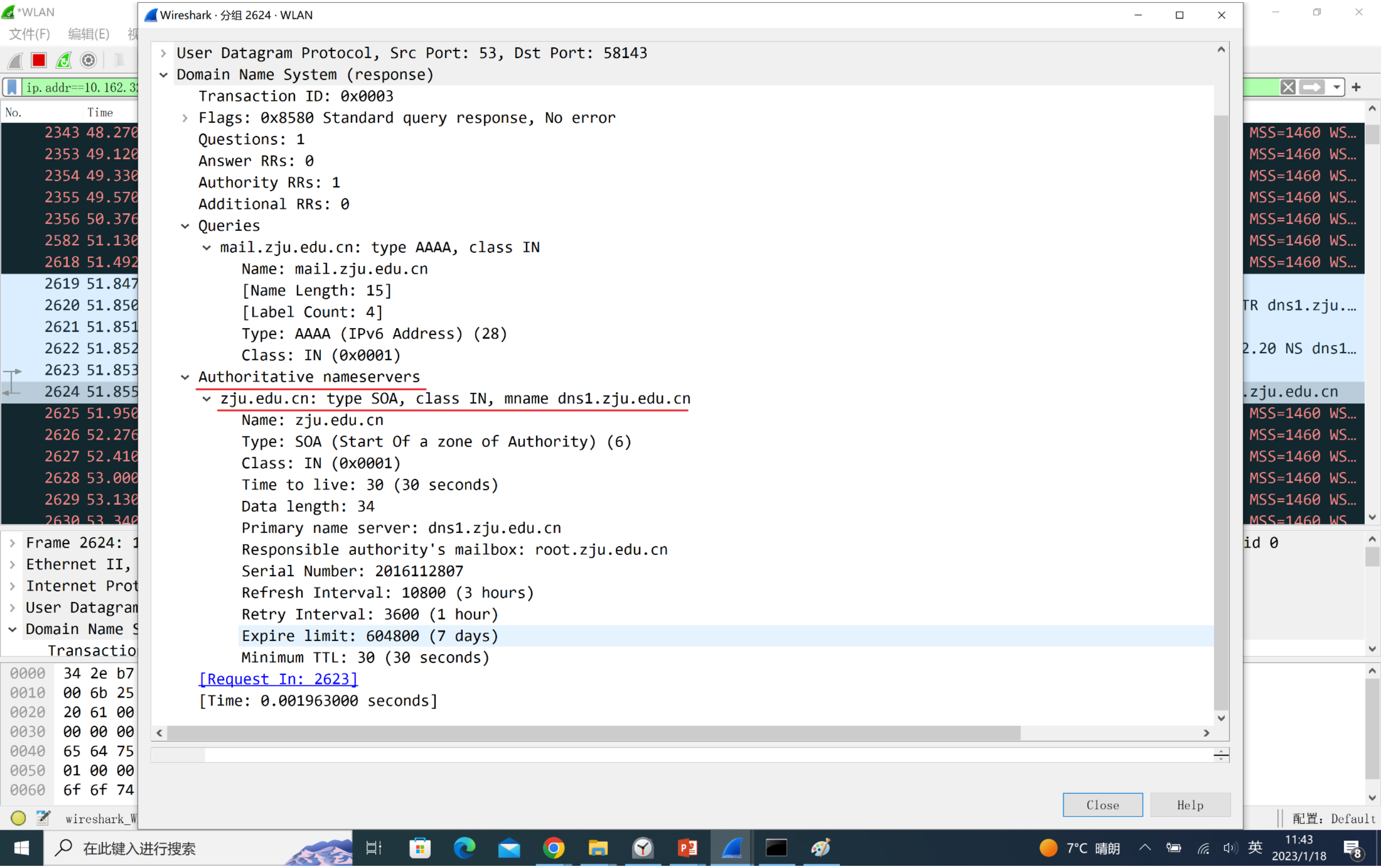
```
> Frame 2623: 75 bytes on wire (600 bits), 75 bytes captured (600 bits) on interface \Device\NPF_{A24DE49A
> Ethernet II, Src: IntelCor_de:dd:de (34:2e:b7:de:dd:de), Dst: NewH3CTe_b9:e8:02 (74:3a:20:b9:e8:02)
> Internet Protocol Version 4, Src: 10.162.32.97, Dst: 10.10.0.21
> User Datagram Protocol, Src Port: 58143, Dst Port: 53
v Domain Name System (query)
    Transaction ID: 0x0003
  > Flags: 0x0100 Standard query
    Questions: 1
    Answer RRs: 0
    Authority RRs: 0
    Additional RRs: 0
  v Queries
    v mail.zju.edu.cn: type AAAA, class IN
        Name: mail.zju.edu.cn
        [Name Length: 15]
        [Label Count: 4]
        Type: AAAA (IPv6 Address) (28)
        Class: IN (0x0001)
    [Response In: 2624]
```

```
0000  74 3a 20 b9 e8 02 34 2e  b7 de dd de 08 00 45 00   t: ···4. ······E·
0010  00 3d 50 c0 00 00 80 11  00 00 0a a2 20 61 0a 0a   ·=P····· ···· a··
0020  00 15 e3 1f 00 35 00 29  35 5c 00 03 01 00 00 01   ·····5·) 5\······
0030  00 00 00 00 00 00 04 6d  61 69 6c 03 7a 6a 75 03   ·······m ail·zju·
0040  65 64 75 02 63 6e 00 00  1c 00 01                  edu·cn·· ···
```

```
> Frame 2623: 75 bytes on
> Ethernet II, Src: Intel
> Internet Protocol Versi
> User Datagram Protocol,
v Domain Name System (que
    Transaction ID: 0x000
, id 0
```

```
0000  74 3a 20 b9 e8 02 3
0010  00 3d 50 c0 00 00 8
0020  00 15 e3 1f 00 35 0
0030  00 00 00 00 00 00 0
0040  65 64 75 02 63 6e 0
```

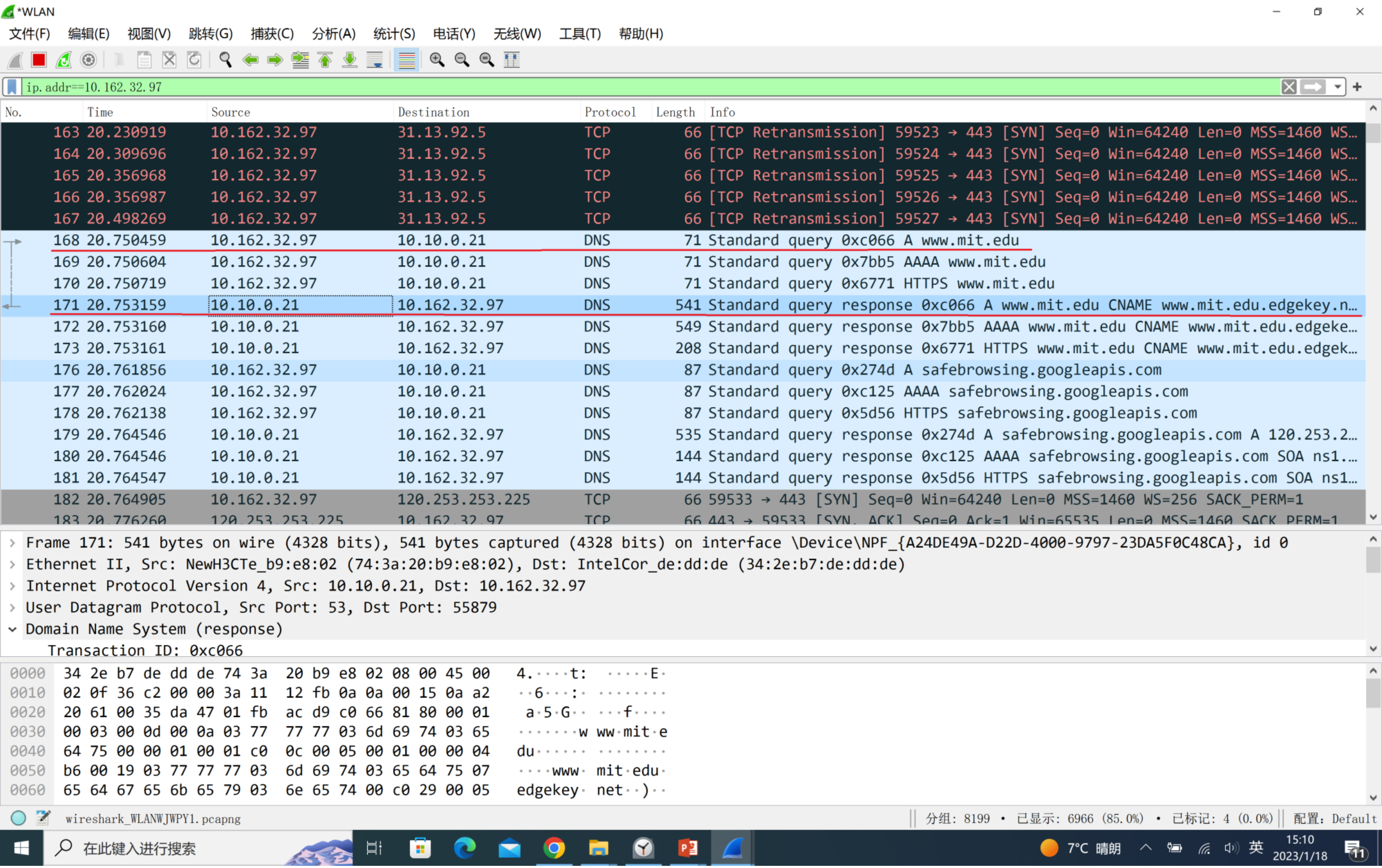
DNS query 数据包3: 询问mail.zju.edu.cn服务器的IPv6地址



Wireshark · 分组 2624 · WLAN

```
> User Datagram Protocol, Src Port: 53, Dst Port: 58143
v Domain Name System (response)
    Transaction ID: 0x0003
  > Flags: 0x8580 Standard query response, No error
    Questions: 1
    Answer RRs: 0
    Authority RRs: 1
    Additional RRs: 0
  v Queries
    v mail.zju.edu.cn: type AAAA, class IN
        Name: mail.zju.edu.cn
        [Name Length: 15]
        [Label Count: 4]
        Type: AAAA (IPv6 Address) (28)
        Class: IN (0x0001)
  v Authoritative nameservers
    v zju.edu.cn: type SOA, class IN, mname dns1.zju.edu.cn
        Name: zju.edu.cn
        Type: SOA (Start Of a zone of Authority) (6)
        Class: IN (0x0001)
        Time to live: 30 (30 seconds)
        Data length: 34
        Primary name server: dns1.zju.edu.cn
        Responsible authority's mailbox: root.zju.edu.cn
        Serial Number: 2016112807
        Refresh Interval: 10800 (3 hours)
        Retry Interval: 3600 (1 hour)
        Expire limit: 604800 (7 days)
        Minimum TTL: 30 (30 seconds)
    [Request In: 2623]
    [Time: 0.001963000 seconds]
```

DNS response 数据包3: 并没有直接给出服务器的IPv6地址， 而是给出SOA (Start of Authority)


*WLAN
文件(F) 编辑(E) 视图(V) 跳转(G) 捕获(C) 分析(A) 统计(S) 电话(Y) 无线(W) 工具(T) 帮助(H)
在此键入进行搜索
7°C 晴朗
英
15:10
2023/1/18

```
ip.addr==10.162.32.97
```

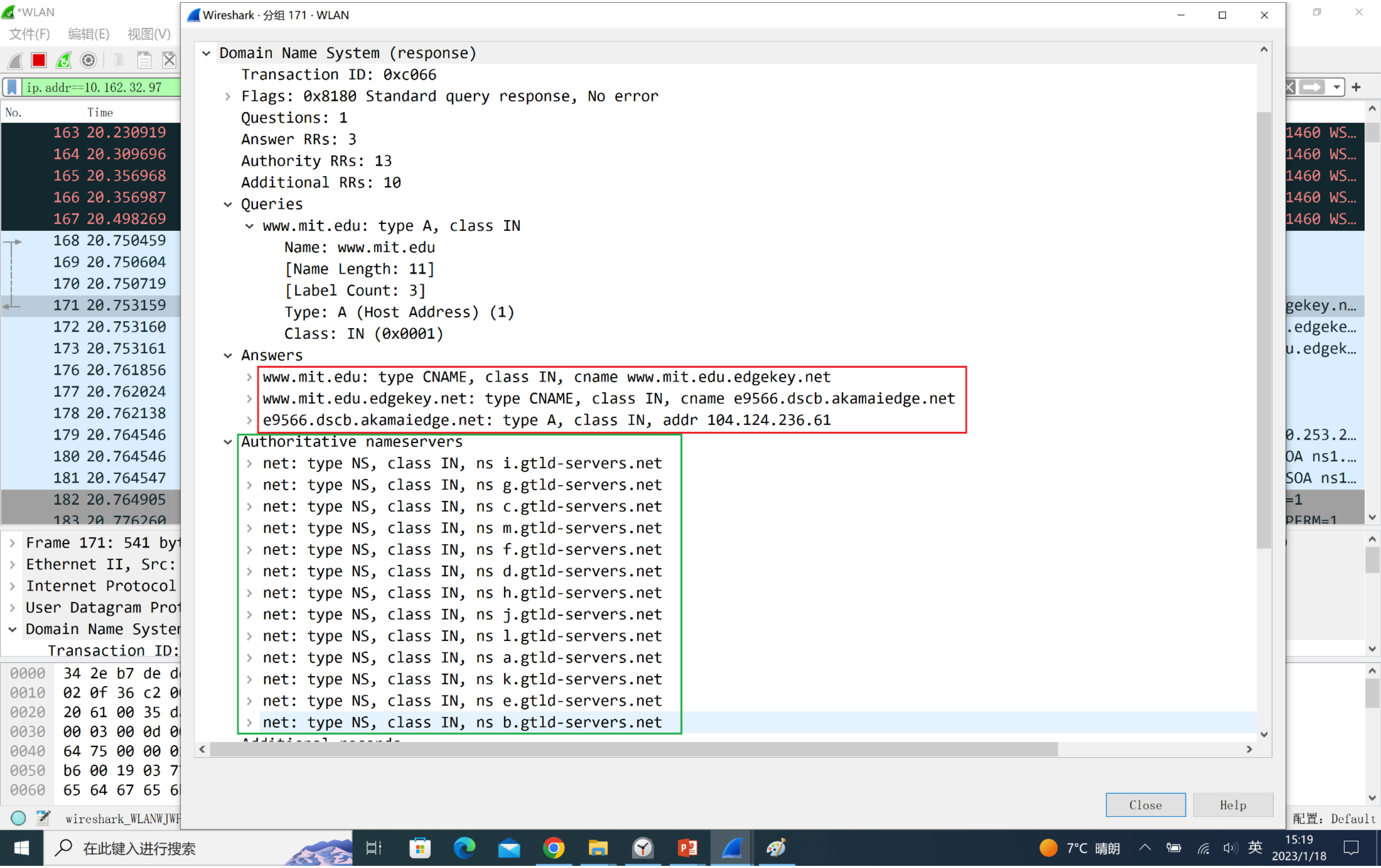
```
No.  Time       Source           Destination      Protocol Length Info
163 20.230919  10.162.32.97     31.13.92.5       TCP      66 [TCP Retransmission] 59523 → 443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS…
164 20.309696  10.162.32.97     31.13.92.5       TCP      66 [TCP Retransmission] 59524 → 443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS…
165 20.356968  10.162.32.97     31.13.92.5       TCP      66 [TCP Retransmission] 59525 → 443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS…
166 20.356987  10.162.32.97     31.13.92.5       TCP      66 [TCP Retransmission] 59526 → 443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS…
167 20.498269  10.162.32.97     31.13.92.5       TCP      66 [TCP Retransmission] 59527 → 443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS…
168 20.750459  10.162.32.97     10.10.0.21       DNS      71 Standard query 0xc066 A www.mit.edu
169 20.750604  10.162.32.97     10.10.0.21       DNS      71 Standard query 0x7bb5 AAAA www.mit.edu
170 20.750719  10.162.32.97     10.10.0.21       DNS      71 Standard query 0x6771 HTTPS www.mit.edu
171 20.753159  10.10.0.21       10.162.32.97     DNS     541 Standard query response 0xc066 A www.mit.edu CNAME www.mit.edu.edgekey.n…
172 20.753160  10.10.0.21       10.162.32.97     DNS     549 Standard query response 0x7bb5 AAAA www.mit.edu CNAME www.mit.edu.edgeke…
173 20.753161  10.10.0.21       10.162.32.97     DNS     208 Standard query response 0x6771 HTTPS www.mit.edu CNAME www.mit.edu.edgek…
176 20.761856  10.162.32.97     10.10.0.21       DNS      87 Standard query 0x274d A safebrowsing.googleapis.com
177 20.762024  10.162.32.97     10.10.0.21       DNS      87 Standard query 0xc125 AAAA safebrowsing.googleapis.com
178 20.762138  10.162.32.97     10.10.0.21       DNS      87 Standard query 0x5d56 HTTPS safebrowsing.googleapis.com
179 20.764546  10.10.0.21       10.162.32.97     DNS     535 Standard query response 0x274d A safebrowsing.googleapis.com A 120.253.2…
180 20.764546  10.10.0.21       10.162.32.97     DNS     144 Standard query response 0xc125 AAAA safebrowsing.googleapis.com SOA ns1.…
181 20.764547  10.10.0.21       10.162.32.97     DNS     144 Standard query response 0x5d56 HTTPS safebrowsing.googleapis.com SOA ns1…
182 20.764905  10.162.32.97     120.253.253.225  TCP      66 59533 → 443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1
183 20.776260  120.253.253.225  10.162.32.97     TCP      66 443 → 59533 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1460 SACK_PERM=1
```

```
> Frame 171: 541 bytes on wire (4328 bits), 541 bytes captured (4328 bits) on interface \Device\NPF_{A24DE49A-D22D-4000-9797-23DA5F0C48CA}, id 0
> Ethernet II, Src: NewH3CTe_b9:e8:02 (74:3a:20:b9:e8:02), Dst: IntelCor_de:dd:de (34:2e:b7:de:dd:de)
> Internet Protocol Version 4, Src: 10.10.0.21, Dst: 10.162.32.97
> User Datagram Protocol, Src Port: 53, Dst Port: 55879
v Domain Name System (response)
    Transaction ID: 0xc066
```

```
0000  34 2e b7 de dd de 74 3a  20 b9 e8 02 08 00 45 00   4.····t: ·····E·
0010  02 0f 36 c2 00 00 3a 11  12 fb 0a 0a 00 15 0a a2   ··6···:· ········
0020  20 61 00 35 da 47 01 fb  ac d9 c0 66 81 80 00 01    a·5·G·· ···f····
0030  00 03 00 0d 00 0a 03 77  77 77 03 6d 69 74 03 65   ·······w ww·mit·e
0040  64 75 00 00 01 00 01 c0  0c 00 05 00 01 00 00 04   du······ ········
0050  b6 00 19 03 77 77 77 03  6d 69 74 03 65 64 75 07   ····www· mit·edu·
0060  65 64 67 65 6b 65 79 03  6e 65 74 00 c0 29 00 05   edgekey· net··)··
```

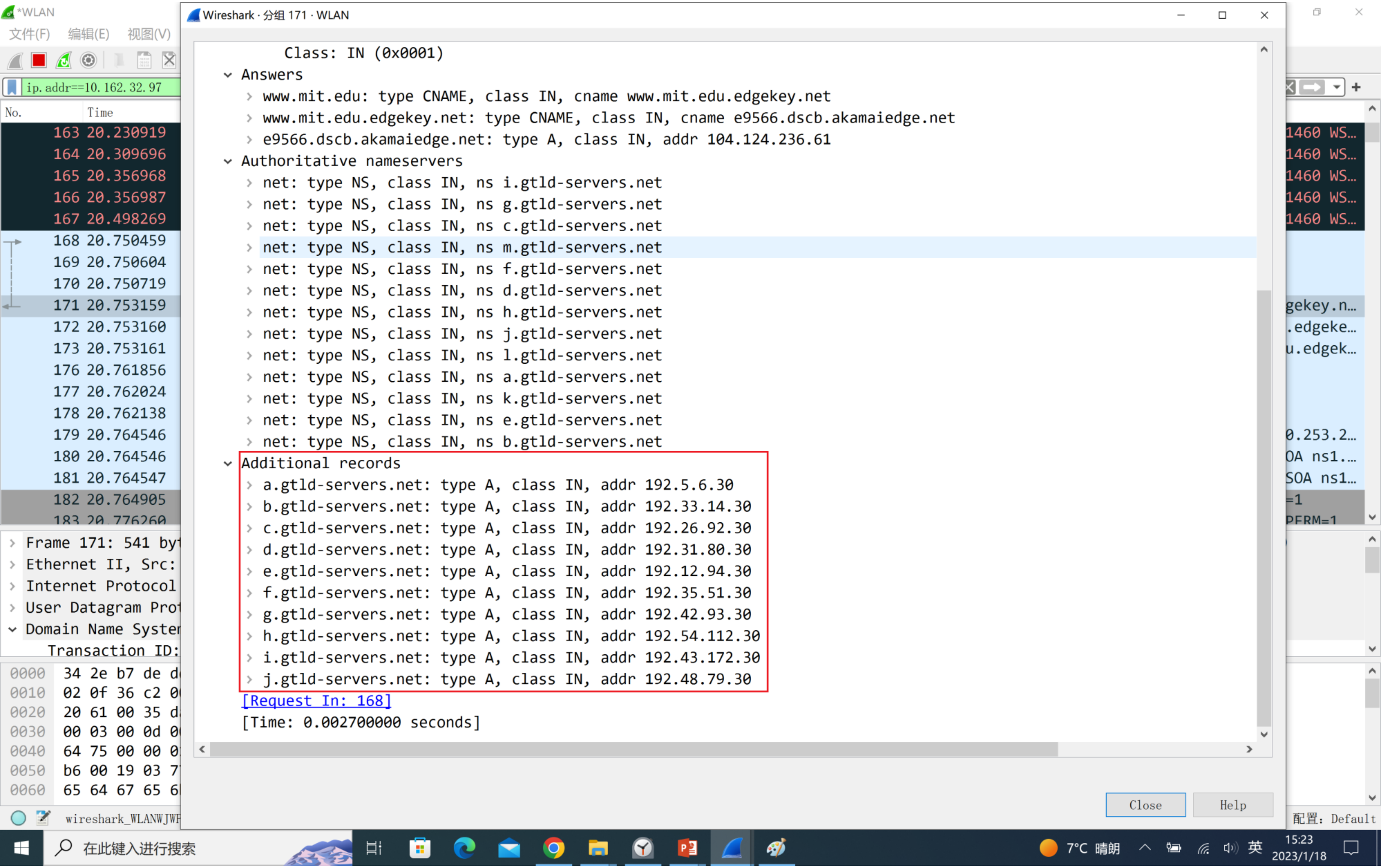
```
wireshark_WLANWJWPY1.pcapng    分组: 8199 • 已显示: 6966 (85.0%) • 已标记: 4 (0.0%)    配置: Default
```

这是打开www.mit.edu网页时DNS数据包



```
Domain Name System (response)
    Transaction ID: 0xc066
  > Flags: 0x8180 Standard query response, No error
    Questions: 1
    Answer RRs: 3
    Authority RRs: 13
    Additional RRs: 10
  v Queries
    v www.mit.edu: type A, class IN
        Name: www.mit.edu
        [Name Length: 11]
        [Label Count: 3]
        Type: A (Host Address) (1)
        Class: IN (0x0001)
  v Answers
    > www.mit.edu: type CNAME, class IN, cname www.mit.edu.edgekey.net
    > www.mit.edu.edgekey.net: type CNAME, class IN, cname e9566.dscb.akamaiedge.net
    > e9566.dscb.akamaiedge.net: type A, class IN, addr 104.124.236.61
  v Authoritative nameservers
    > net: type NS, class IN, ns i.gtld-servers.net
    > net: type NS, class IN, ns g.gtld-servers.net
    > net: type NS, class IN, ns c.gtld-servers.net
    > net: type NS, class IN, ns m.gtld-servers.net
    > net: type NS, class IN, ns f.gtld-servers.net
    > net: type NS, class IN, ns d.gtld-servers.net
    > net: type NS, class IN, ns h.gtld-servers.net
    > net: type NS, class IN, ns j.gtld-servers.net
    > net: type NS, class IN, ns l.gtld-servers.net
    > net: type NS, class IN, ns a.gtld-servers.net
    > net: type NS, class IN, ns k.gtld-servers.net
    > net: type NS, class IN, ns e.gtld-servers.net
    > net: type NS, class IN, ns b.gtld-servers.net
```

www.mit.edu官网有13个域名服务器



```
ip.addr==10.162.32.97

No.  Time
163 20.230919
164 20.309696
165 20.356968
166 20.356987
167 20.498269
168 20.750459
169 20.750604
170 20.750719
171 20.753159
172 20.753160
173 20.753161
176 20.761856
177 20.762024
178 20.762138
179 20.764546
180 20.764546
181 20.764547
182 20.764905
183 20.776260
```

```
        Class: IN (0x0001)
  ⌄ Answers
    › www.mit.edu: type CNAME, class IN, cname www.mit.edu.edgekey.net
    › www.mit.edu.edgekey.net: type CNAME, class IN, cname e9566.dscb.akamaiedge.net
    › e9566.dscb.akamaiedge.net: type A, class IN, addr 104.124.236.61
  ⌄ Authoritative nameservers
    › net: type NS, class IN, ns i.gtld-servers.net
    › net: type NS, class IN, ns g.gtld-servers.net
    › net: type NS, class IN, ns c.gtld-servers.net
    › net: type NS, class IN, ns m.gtld-servers.net
    › net: type NS, class IN, ns f.gtld-servers.net
    › net: type NS, class IN, ns d.gtld-servers.net
    › net: type NS, class IN, ns h.gtld-servers.net
    › net: type NS, class IN, ns j.gtld-servers.net
    › net: type NS, class IN, ns l.gtld-servers.net
    › net: type NS, class IN, ns a.gtld-servers.net
    › net: type NS, class IN, ns k.gtld-servers.net
    › net: type NS, class IN, ns e.gtld-servers.net
    › net: type NS, class IN, ns b.gtld-servers.net
  ⌄ Additional records
    › a.gtld-servers.net: type A, class IN, addr 192.5.6.30
    › b.gtld-servers.net: type A, class IN, addr 192.33.14.30
    › c.gtld-servers.net: type A, class IN, addr 192.26.92.30
    › d.gtld-servers.net: type A, class IN, addr 192.31.80.30
    › e.gtld-servers.net: type A, class IN, addr 192.12.94.30
    › f.gtld-servers.net: type A, class IN, addr 192.35.51.30
    › g.gtld-servers.net: type A, class IN, addr 192.42.93.30
    › h.gtld-servers.net: type A, class IN, addr 192.54.112.30
    › i.gtld-servers.net: type A, class IN, addr 192.43.172.30
    › j.gtld-servers.net: type A, class IN, addr 192.48.79.30
    [Request In: 168]
    [Time: 0.002700000 seconds]
```

```
› Frame 171: 541 by
› Ethernet II, Src:
› Internet Protocol
› User Datagram Prot
⌄ Domain Name Syster
    Transaction ID:

0000  34 2e b7 de d
0010  02 0f 36 c2 0
0020  20 61 00 35 d
0030  00 03 00 0d 0
0040  64 75 00 00 0
0050  b6 00 19 03 7
0060  65 64 67 65 6

wireshark_WLANWJWF
```

```
1460 WS…
1460 WS…
1460 WS…
1460 WS…
1460 WS…
gekey.n…
.edgeke…
u.edgek…
0.253.2…
OA ns1.…
SOA ns1…
=1
PERM=1
```

www.mit.edu官网一些域名服务器IP地址，因为信息很多，所以response有好几个数据包。

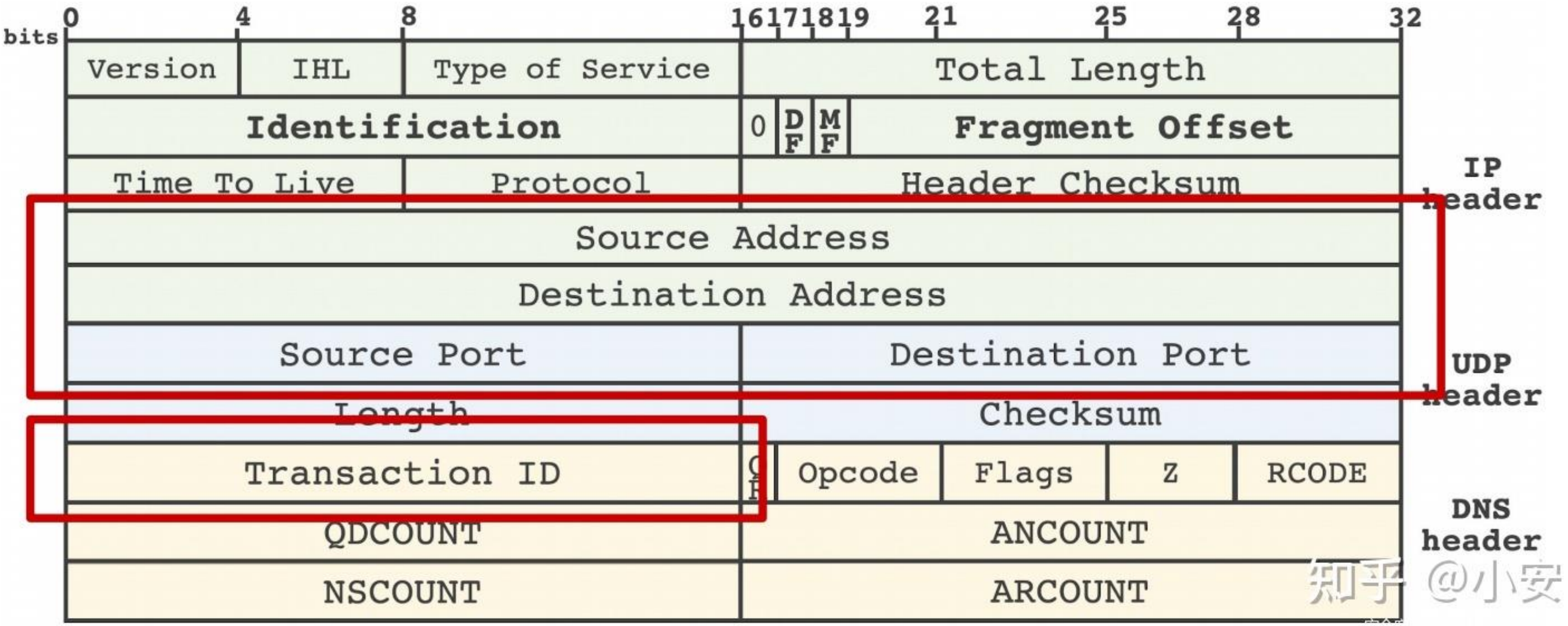
Security of DNS

- Initially, the transaction ID was only 16 bits, and the queries and responses were not secured.
  - This design choice left DNS vulnerable to a variety attacks including “**cache poisoning attack**”. (chapter 8)
    - Forging attack (伪造攻击，目的是制造一个恶意DNS响应，并欺骗递归解析器去接受它。当DNS响应数据包中的一些字段和DNS请求数据包中的字段相匹配时，DNS响应数据包就会被解析器所接受，这些字段是：Question section查询问题，DNS transaction ID, source/destination address, port numbers)
    - Example: The Kaminsky attack (发生在2008年，通过伪造DNS响应包来攻击，当时影响了几乎所有的DNS软件和设备)

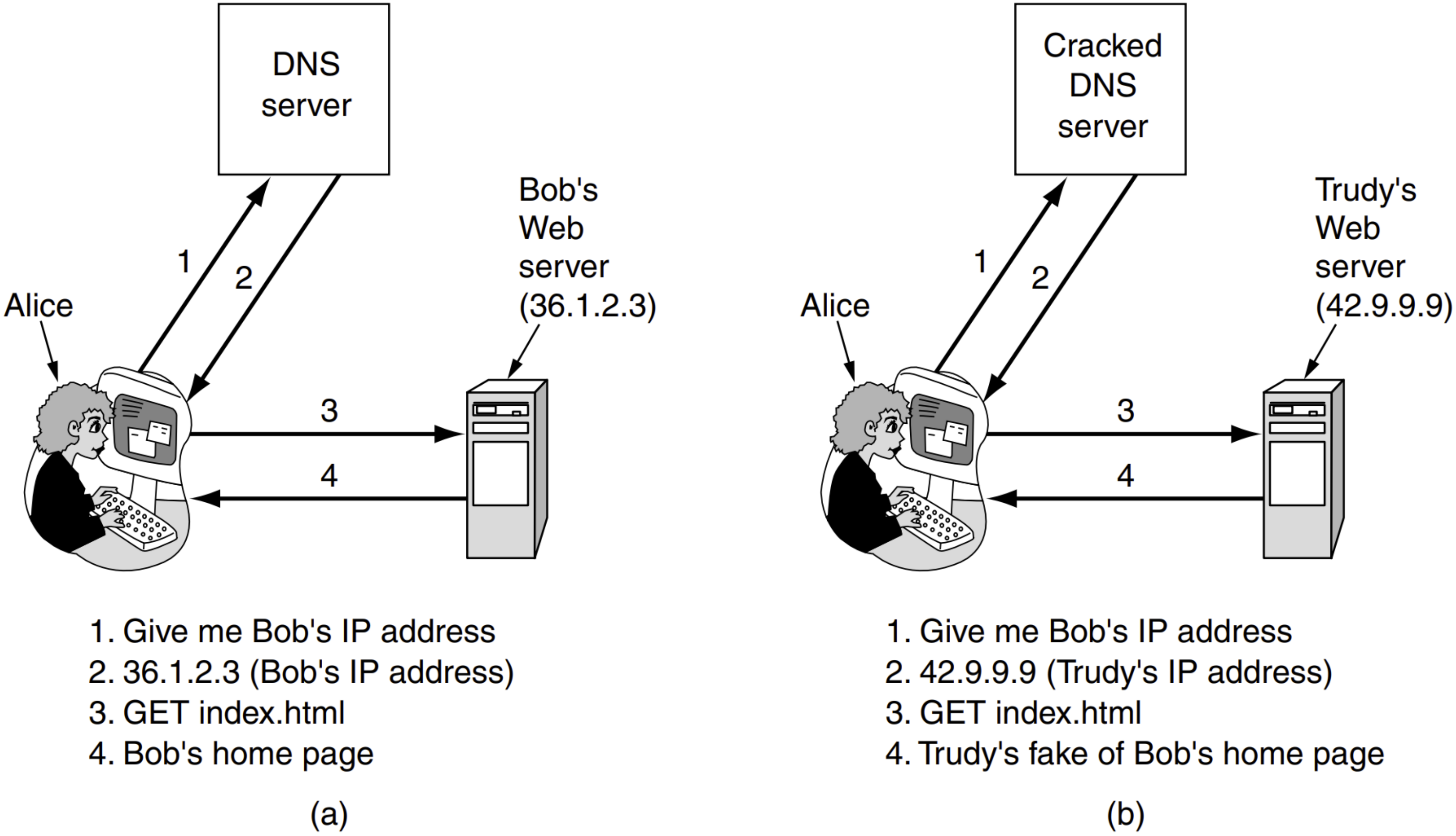
DNS Header [7]

- Sensitive fields to be forged:
  - 1) Question session
  - 2) DNS Transaction ID
  - 3) source/destination addresses
  - 4) port numbers

在Kaminsky攻击之前，DNS报文中的大多数源端口都不是采用随机化的方式分配的，通常一些DNS解析器会直接采用53作为源端口号，或是操作系统中的一个固定值。



DNS Spoofing



1. Give me Bob's IP address
2. 36.1.2.3 (Bob's IP address)
3. GET index.html
4. Bob's home page

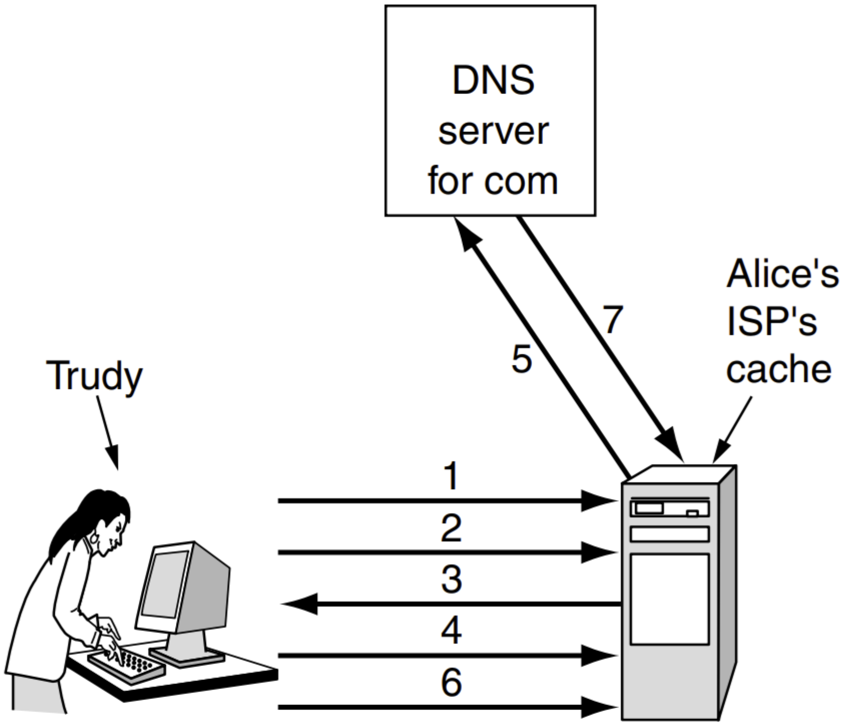
(a)

1. Give me Bob's IP address
2. 42.9.9.9 (Trudy's IP address)
3. GET index.html
4. Trudy's fake of Bob's home page

(b)

Figure 8-46. (a) Normal situation. (b) An attack based on breaking into a DNS server and modifying Bob's record.

DNS Spoofing — man-in-the-middle attack



1. Look up foobar.trudy-the-intruder.com
   (to force it into the ISP's cache)
2. Look up www.trudy-the-intruder.com
   (to get the ISP's next sequence number)
3. Request for www.trudy-the-intruder.com
   (Carrying the ISP's next sequence number, n)
4. Quick like a bunny, look up bob.com
   (to force the ISP to query the com server in step 5)
5. Legitimate query for bob.com with seq = n+1
6. Trudy's forged answer: Bob is 42.9.9.9, seq = n+1
7. Real answer (rejected, too late)

Figure 8-47. How Trudy spoofs Alice's ISP.

Outline

- Overview of application layer
- Important application layer protocols
  - DNS
  - FTP
  - Email
  - HTTP

Outline

- Overview of application layer
- Important application layer protocols
  - DNS
  - FTP
  - Email
  - HTTP

File Transfer: FTP [5] (I)

- FTP: to transfer files to and from a remote host.
  - RFC 959

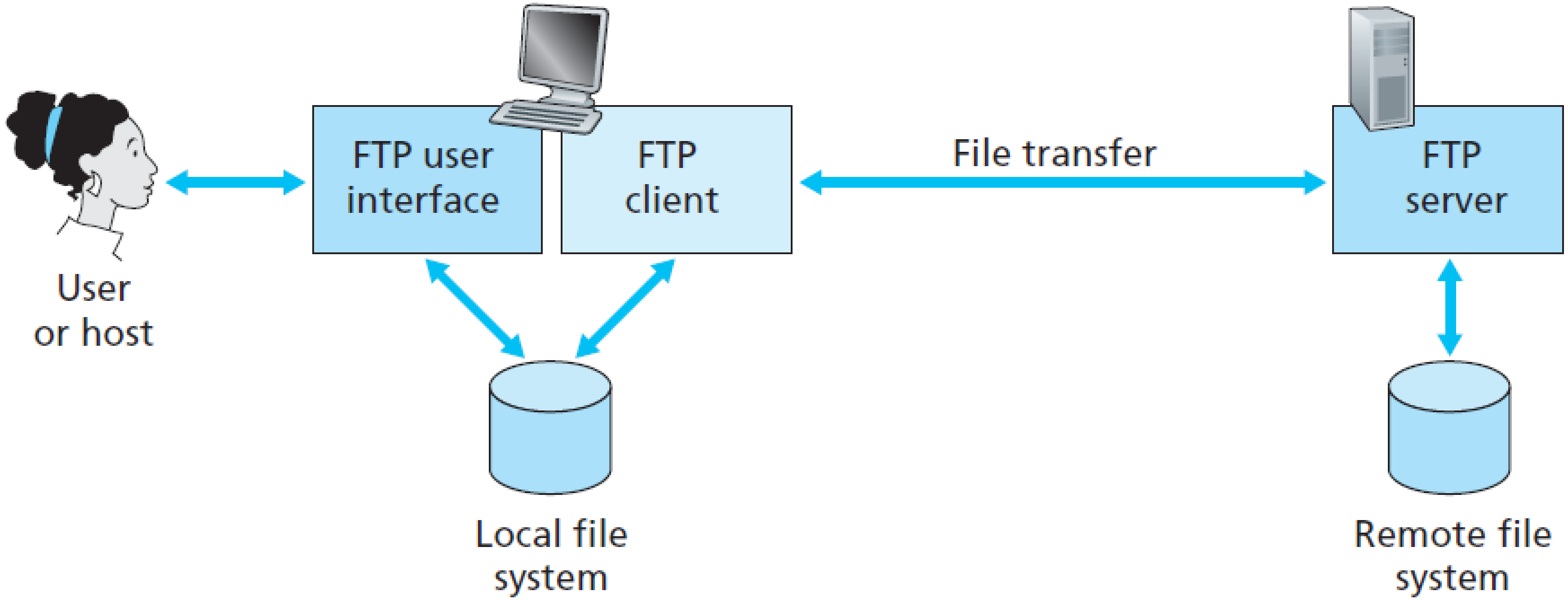


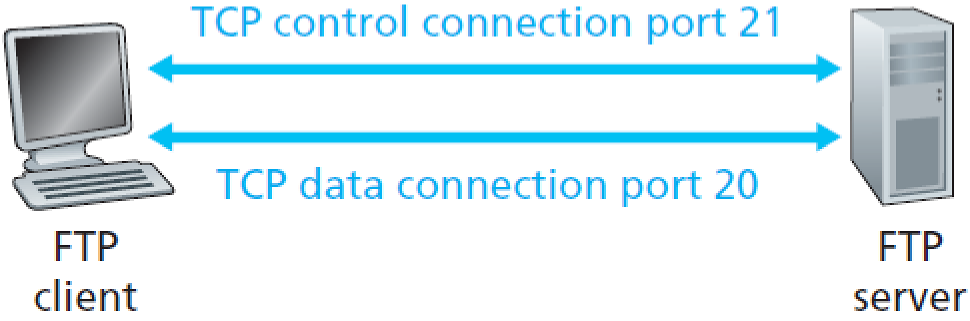
Figure 2.14 ♦ FTP moves files between local and remote file systems

File Transfer: FTP [5] (II)

- Steps:
  - 1) The user first provides the hostname of the remote host, causing the FTP client process in the local host to establish **a TCP connection** with the FTP server process in the remote host.
  - 2) The user then provides the user identification and password, which are sent over the TCP connection as part of FTP commands.
  - 3) Once the server has authorized the user, the user copies one or more files stored in the local file system into the remote file system (or vice versa).

File Transfer: FTP [5] (III)

- FTP uses **two parallel TCP connections** to transfer a file, a control connection and a data connection.
  - The control connection is used for sending control information between the two hosts — information such as user identification, password, commands to change remote directory, and commands to "put" and "get" files.
    - FTP is said to send its control information **out-of-band**. (Because of this control connection (separate) FTP is "out-of-band" .)
    - The commands, from client to server, and replies, from server to client, are sent across the control connection <u>in 7-bit ASCII format</u>.
  - The data connection is used to actually send a file.



Outline

- Overview of application layer
- Important application layer protocols
  - DNS
  - FTP
  - Email
  - HTTP

Outline

- Overview of application layer
- Important application layer protocols
  - DNS
  - FTP
  - Email
  - HTTP

Electronic Mail

- Email is an **asynchronous** communication medium.
- Three major components
  - User agents
  - Message transfer agents (mail servers)
  - Simple Mail Transfer Protocol: **SMTP**

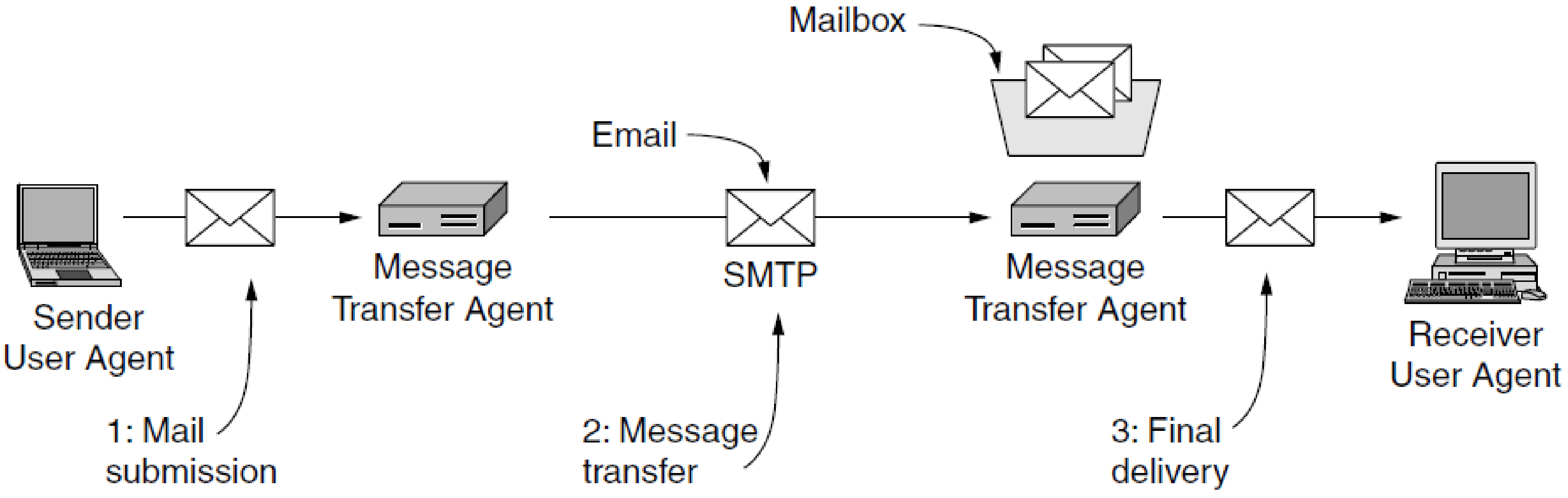


Figure 7-7. Architecture of the email system.

SMTP [5] (I)

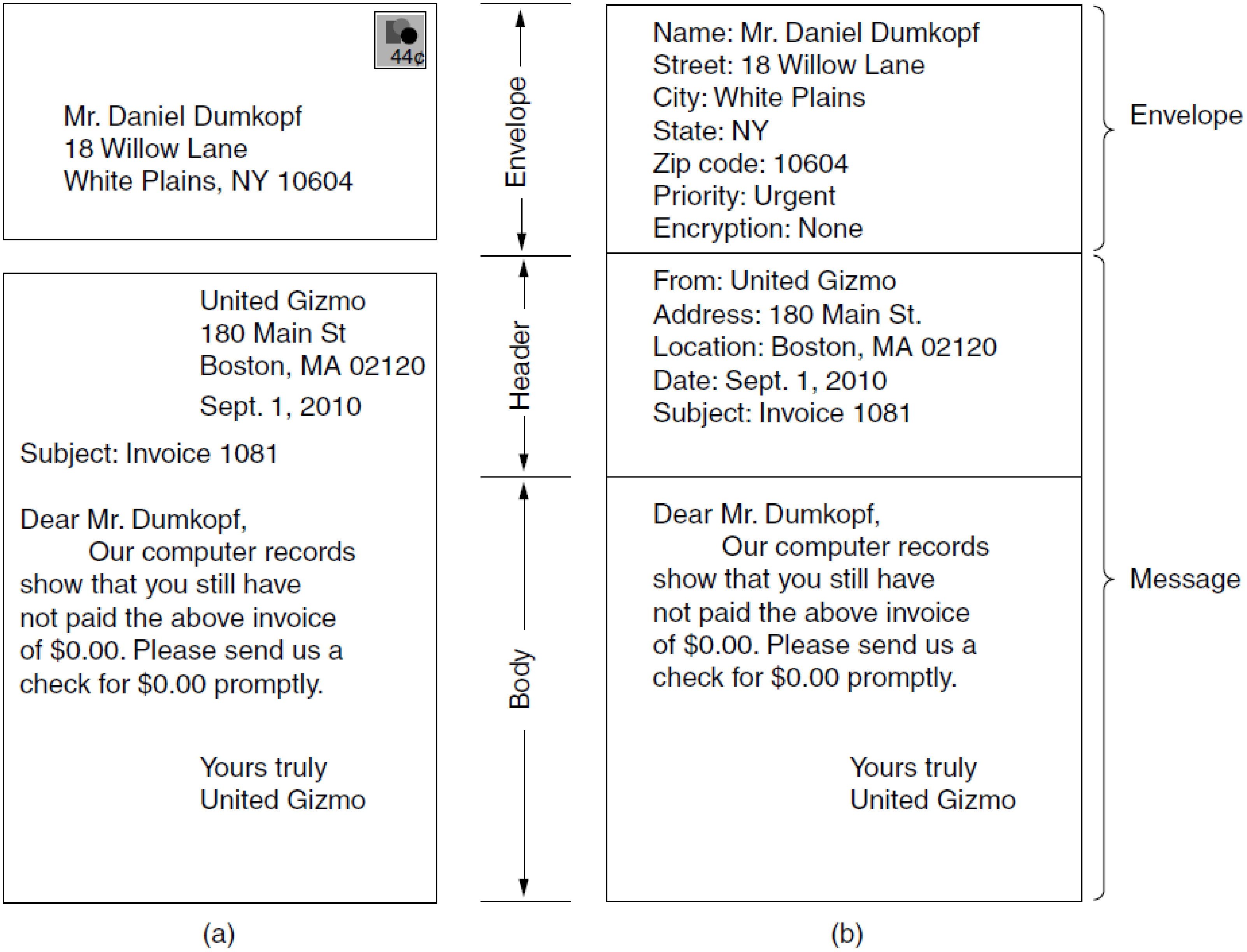
- SMTP is the principal application layer protocol for Internet electronic mail.
  - RFC 5321
  - It uses the reliable data transfer service of TCP to transfer mail from the sender’s mail server to the recipient’s mail server.
  - As with most application-layer protocols, SMTP has two sides: a client side, which executes on the sender’s mail server, and a server side, which executes on the recipient’s mail server.
    - Both the client and server sides of SMTP run on every mail server.
  - It restricts the body (not just the headers) of all mail messages to simple **7-bit ASCII**.
    - It requires binary multimedia data to be encoded to ASCII before being sent over SMTP; and it requires the corresponding ASCII message to be decoded back to binary after SMTP transport.

SMTP [5] (II)

- It is important to observe that SMTP does not normally use intermediate mail servers for sending mail, even when the two mail servers are located at opposite ends of the world.
- How SMTP transfers a message from a sending mail server to a receiving mail server?
  - First, the client SMTP (running on the sending mail server host) has **TCP** establish a connection to **port 25** at the server SMTP (running on the receiving mail server host).
    - If the server is down, the client tries again later.
  - Once this connection is established, the server and client perform some application-layer handshaking.
    - During this SMTP handshaking phase, the SMTP client indicates the email address of the sender and the email address of the recipient.
  - Once the SMTP client and server have introduced themselves to each other, the client sends the message.
    - The client then repeats this process over the same TCP connection if it has other messages to send to the server.
  - Otherwise, it instructs TCP to close the connection.

Email Message Format

- Mail is sent between message transfer agents in a standard format.
- The original format, **RFC 822**, has been revised to the current **RFC5322** and extended with support for *multimedia content*.
- A key idea in the message format is the distinction between the envelope and its contents.
  - The envelope encapsulates the message. It contains all the information needed for transporting the message, such as the destination address, priority, and security level. The message transport agents use the envelope for routing, just as the post office does.
  - The message inside the envelope consists of two separate parts: **the header** and **the body**.
    - The header contains control information for the user agents
    - The body is entirely for the human recipient.



Mr. Daniel Dumkopf
18 Willow Lane
White Plains, NY 10604

United Gizmo
180 Main St
Boston, MA 02120

Sept. 1, 2010

Subject: Invoice 1081

Dear Mr. Dumkopf,
Our computer records show that you still have not paid the above invoice of $0.00. Please send us a check for $0.00 promptly.

Yours truly
United Gizmo

Name: Mr. Daniel Dumkopf
Street: 18 Willow Lane
City: White Plains
State: NY
Zip code: 10604
Priority: Urgent
Encryption: None

From: United Gizmo
Address: 180 Main St.
Location: Boston, MA 02120
Date: Sept. 1, 2010
Subject: Invoice 1081

Dear Mr. Dumkopf,
Our computer records show that you still have not paid the above invoice of $0.00. Please send us a check for $0.00 promptly.

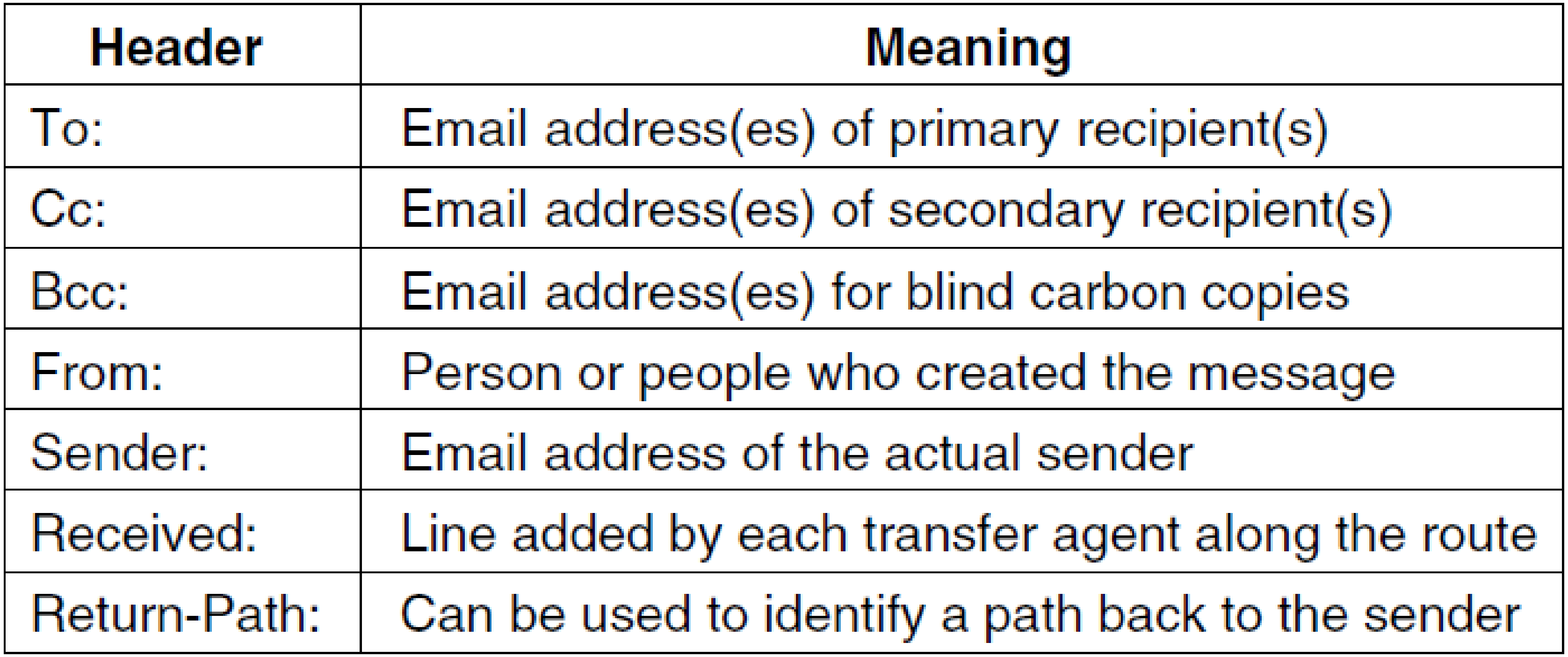
Yours truly
United Gizmo

Figure 7-8. Envelopes and messages. (a) Paper mail. (b) Electronic mail.

Electronic Mail: User Agent

- A user agent is a program (sometimes called an “email reader”)
- Composing, receiving and replying to messages, as well as for manipulating mailboxes
- There are many popular user agents, including Google gmail, Microsoft Outlook, etc.
- Outgoing, incoming messages stored on server

RFC5322 — the Internet Message Format



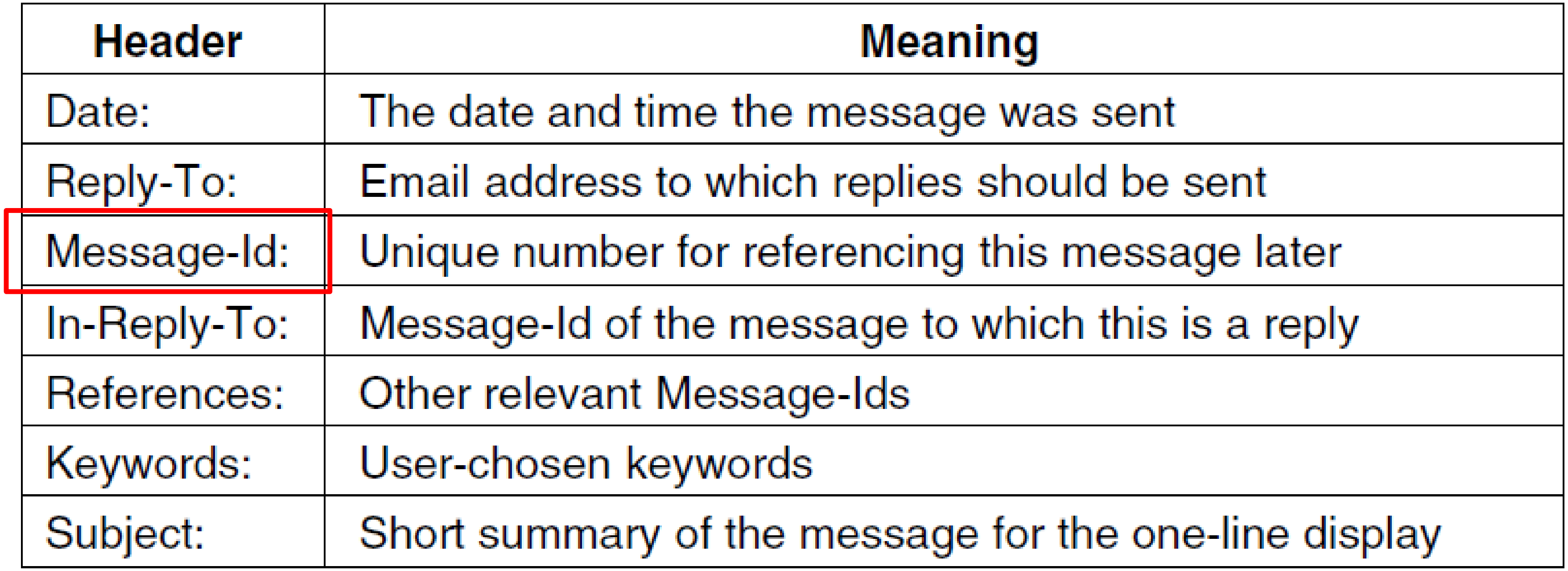
Header	Meaning
To:	Email address(es) of primary recipient(s)
Cc:	Email address(es) of secondary recipient(s)
Bcc:	Email address(es) for blind carbon copies
From:	Person or people who created the message
Sender:	Email address of the actual sender
Received:	Line added by each transfer agent along the route
Return-Path:	Can be used to identify a path back to the sender

Figure 7-10. RFC 5322 header fields related to message transport.

The sender may be different from the “from”, e.g.:

from: a company manager
sender: his secretary

RFC5322 — the Internet Message Format

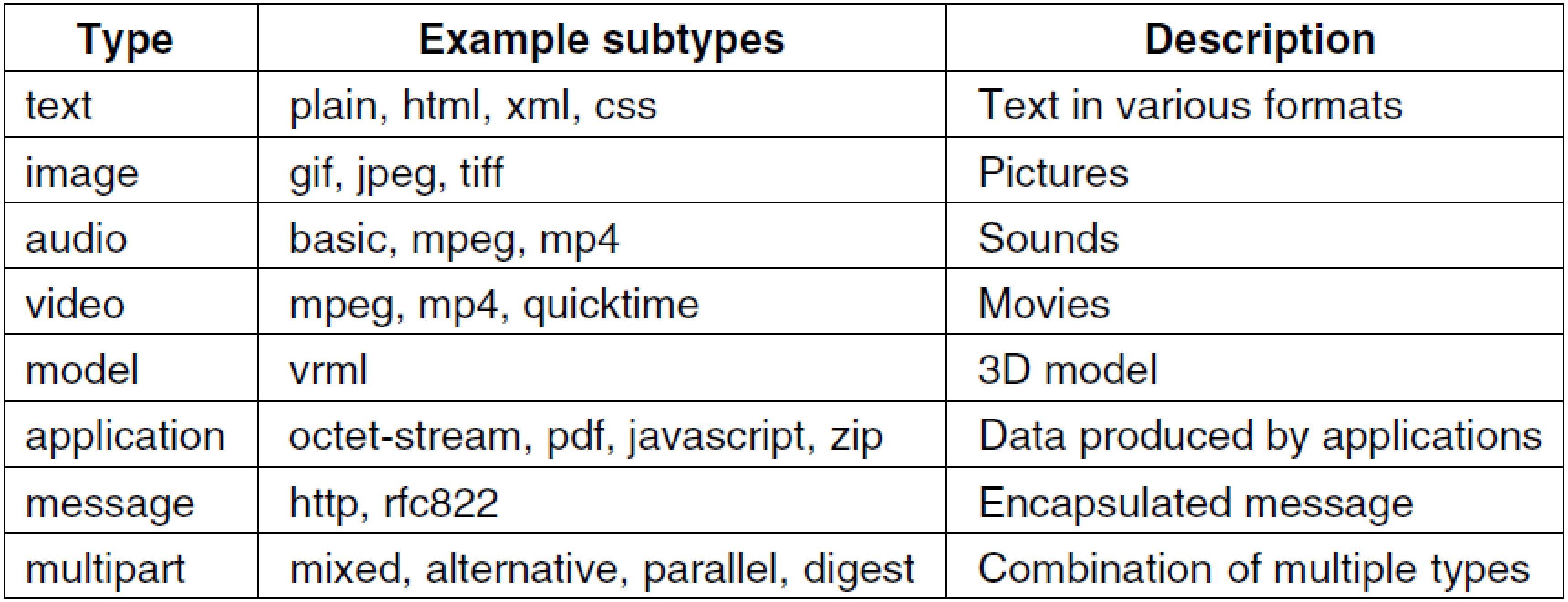


Header	Meaning
Date:	The date and time the message was sent
Reply-To:	Email address to which replies should be sent
Message-Id:	Unique number for referencing this message later
In-Reply-To:	Message-Id of the message to which this is a reply
References:	Other relevant Message-Ids
Keywords:	User-chosen keywords
Subject:	Short summary of the message for the one-line display

Figure 7-11. Some fields used in the RFC 5322 message header.

MIME — The Multimedia Internet Mail Extension

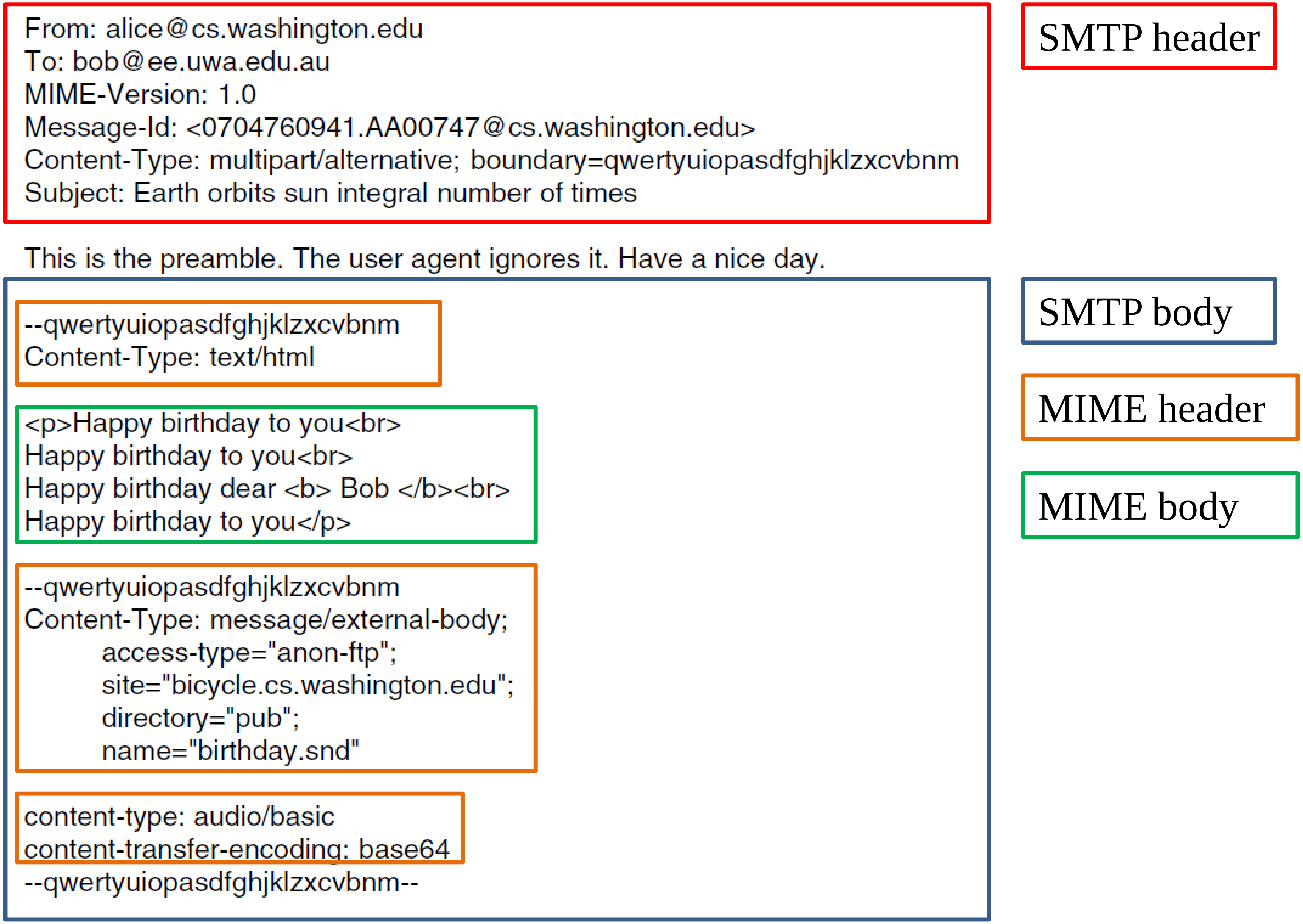
- MIME is described in RFCs 2045-2047, 4288, 4289, and 2049.
- Content-Types: type/subtype; parameters



Type	Example subtypes	Description
text	plain, html, xml, css	Text in various formats
image	gif, jpeg, tiff	Pictures
audio	basic, mpeg, mp4	Sounds
video	mpeg, mp4, quicktime	Movies
model	vrml	3D model
application	octet-stream, pdf, javascript, zip	Data produced by applications
message	http, rfc822	Encapsulated message
multipart	mixed, alternative, parallel, digest	Combination of multiple types

Figure 7-13. MIME content types and example subtypes.

- base64编码方法如下：
  - 先将24bit的代码划分为4个6位组。
  - 6bit组的二进制代码共有64种不同的值，从0到63。
  - 用A表示0，B表示1，等等。26个大写字母排列完毕后，接下去再排26个小写字母，再后面是10个数字，最后用“+”表示62，而用“/”表示63。再用两个连在一起的等号“==”和一个等号“=”分别表示最后一组的代码只有8或16比特。
    - A-00,B-01,C-02,D-03,E-04,F-05,G-06,H-07,I-08,J-09,K-10,L-11,M-12,N-13,O-14,P-15,Q-16,R-17,S-18,T-19,U-20,V-21,W-22,X-23,Y-24,Z-25, a-26,b-27,c-28,d-29,e-30,f-31,g-32,h-33,i-34,j-35,k-36,l-37,m-38,n-39,o-40,p-41,q-42,r-43,s-44,t-45,u-46,v-47,w-48,x-49,y-50,z-51, 0-52,1-53,2-54,3-55,4-56,5-57,6-58,7-59,8-60,9-61,+-62,/-63
  - 回车和换行都忽略，它们可在任何地方插入。
  - 作为base64编码的例子，假设有二进制代码，共24bit：01001001 00110001 01111001。先划分为4个6bit组，即010010 010011 000101 111001,对应的十进制值为18,19,5,57。对应的base64编码为：STF5。



```
From: alice@cs.washington.edu
To: bob@ee.uwa.edu.au
MIME-Version: 1.0
Message-Id: <0704760941.AA00747@cs.washington.edu>
Content-Type: multipart/alternative; boundary=qwertyuiopasdfghjklzxcvbnm
Subject: Earth orbits sun integral number of times

This is the preamble. The user agent ignores it. Have a nice day.

--qwertyuiopasdfghjklzxcvbnm
Content-Type: text/html

<p>Happy birthday to you<br>
Happy birthday to you<br>
Happy birthday dear <b> Bob </b><br>
Happy birthday to you</p>

--qwertyuiopasdfghjklzxcvbnm
Content-Type: message/external-body;
        access-type="anon-ftp";
        site="bicycle.cs.washington.edu";
        directory="pub";
        name="birthday.snd"

content-type: audio/basic
content-transfer-encoding: base64
--qwertyuiopasdfghjklzxcvbnm--
```

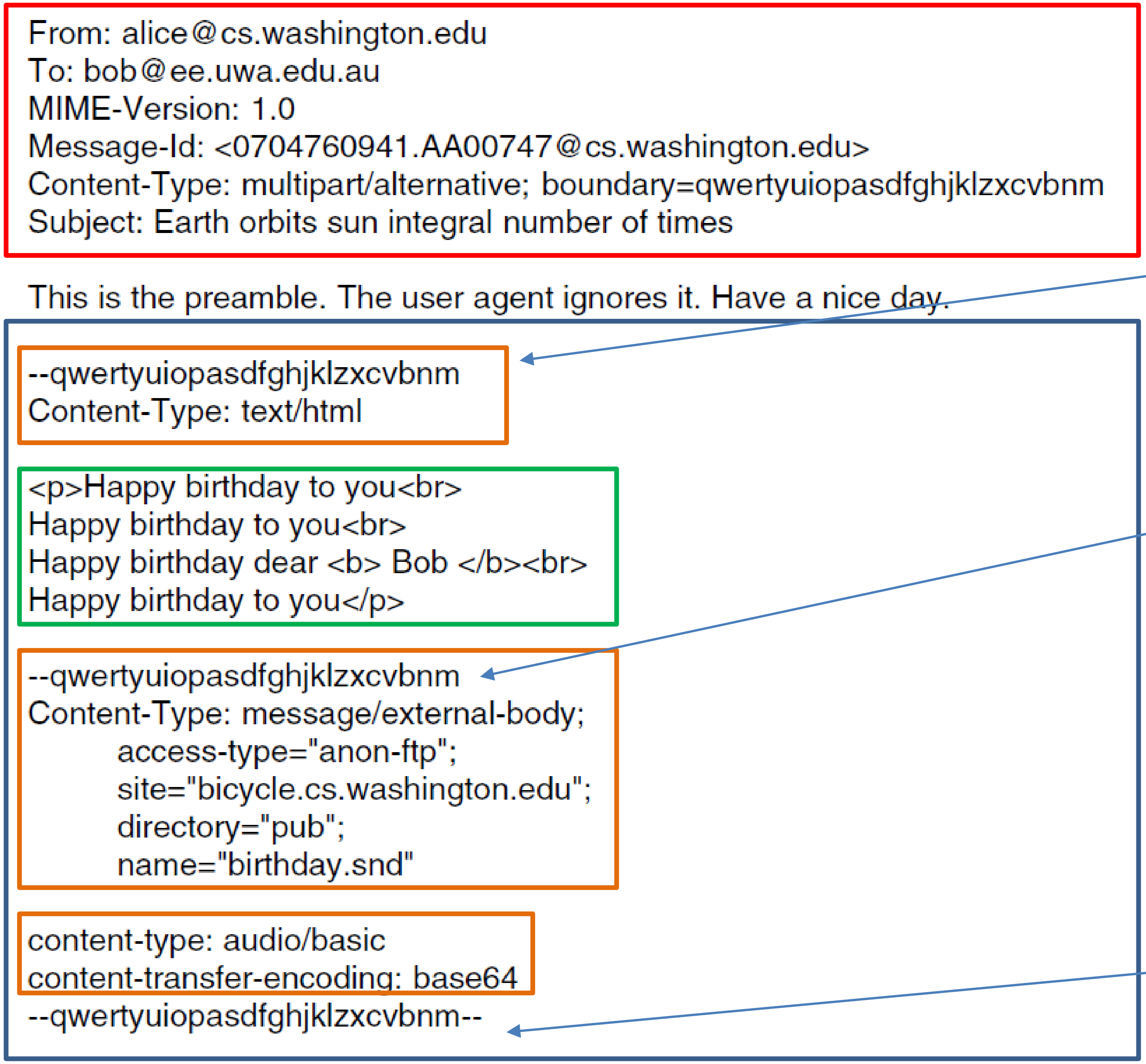
SMTP header

SMTP body

MIME header

MIME body

Figure 7-14. A multipart message containing HTML and audio alternatives.



```
From: alice@cs.washington.edu
To: bob@ee.uwa.edu.au
MIME-Version: 1.0
Message-Id: <0704760941.AA00747@cs.washington.edu>
Content-Type: multipart/alternative; boundary=qwertyuiopasdfghjklzxcvbnm
Subject: Earth orbits sun integral number of times

This is the preamble. The user agent ignores it. Have a nice day.

--qwertyuiopasdfghjklzxcvbnm
Content-Type: text/html

<p>Happy birthday to you<br>
Happy birthday to you<br>
Happy birthday dear <b> Bob </b><br>
Happy birthday to you</p>

--qwertyuiopasdfghjklzxcvbnm
Content-Type: message/external-body;
        access-type="anon-ftp";
        site="bicycle.cs.washington.edu";
        directory="pub";
        name="birthday.snd"

content-type: audio/basic
content-transfer-encoding: base64
--qwertyuiopasdfghjklzxcvbnm--
```

Figure 7-14. A multipart message containing HTML and audio alternatives.

在电子邮件中，当内容类型（Content - Type）被指定为“multipart”（如“multipart/mixed”“multipart/alternative”等）时，需要一种方式来区分消息体中的不同部分。这就是边界（**boundary**）的作用。它本质上是一个字符串，用于标记每个部分的开始和结束。

Content-Type: multipart/alternative

- 含义：是在电子邮件或其他基于 HTTP 等协议的消息体中的一个内容类型（MIME 类型）的标识。它表明消息内容是由多个部分组成的，并且这些部分是提供同一内容的不同表示形式，通常用于在不同的显示环境或客户端能力下提供替代的呈现方式。
- 当一个消息被标记为“multipart/alternative”时，它通常包含一个或多个子部分，每个子部分都有自己独立的 MIME 类型和内容。这些子部分按照优先级顺序排列，一般来说，最复杂或者功能最丰富的表示形式放在最后。例如，一个包含文本和 HTML 两种格式内容的电子邮件可能会这样构建：
  - 首先是纯文本格式的部分，这部分内容简单，任何文本客户端都可以显示。
  - 然后是 HTML 格式的部分，这部分内容可以包含丰富的格式，如字体样式、颜色、图片等。
- 在电子邮件通信中，这种类型非常有用。因为不同的电子邮件客户端对内容的显示能力不同。有些客户端可能只能显示纯文本，而有些则可以很好地显示 HTML 内容。

其它的一些Content-Type

- “Content - Type: message/external - body”是一种 MIME（多用途互联网邮件扩展）类型。它用于表示邮件的内容实际上是引用自外部的资源，而不是直接包含在邮件内部。这就好比是一个指针，指向邮件内容真正所在的位置。
- “Content - Type: audio/basic” 是一种 MIME（多用途互联网邮件扩展）类型，用于表示音频内容。它表明消息体中的内容是基本的音频数据格式。这种类型主要用于在网络通信（如电子邮件、网页音频嵌入等）场景中识别音频数据，以便接收端能够正确地处理和播放音频。

Mail Access Protocols [5]

- SMTP has been designed for pushing email form one host to another.
  - Alice’s user agent uses SMTP to **push** the email message into her mail server.
  - Then, Alice’s mail server uses SMTP to **relay** the email message to Bob’s mail server.
  - Note that Bob’s user agent cannot use SMTP to obtain the messages because obtaining the messages is a **pull** operation, whereas **SMTP is a push protocol**.

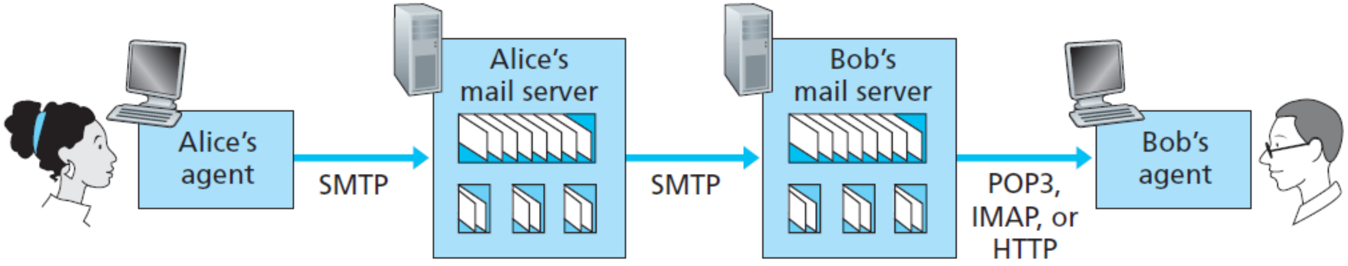
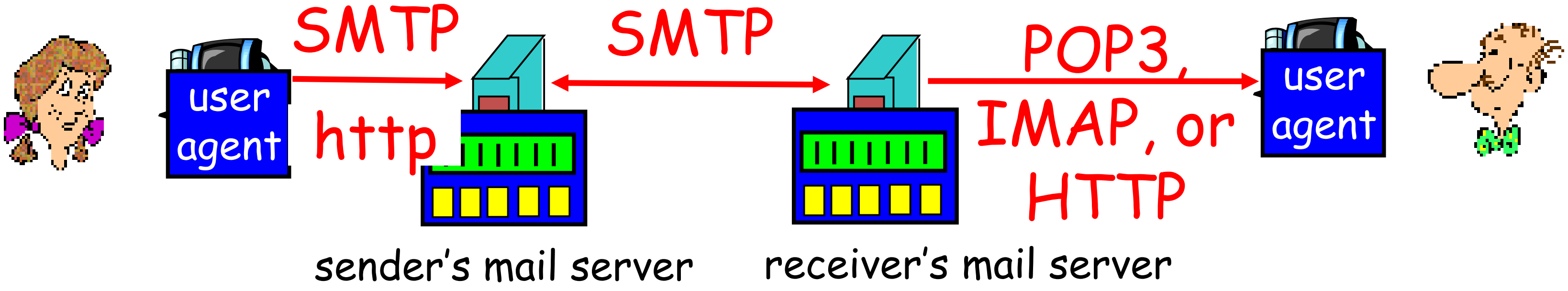


Figure 2.18 ♦ E-mail protocols and their communicating entities

Mail Access Protocol – Final Delivery



- SMTP: delivery/storage to receiver's server
- Mail access protocol: retrieval from server
  - POP: Post Office Protocol [RFC 1939] (port 110)
    - authorization (agent ↔ server) and download
    - Does not maintain state across POP sessions
    - Cannot manipulate emails at the server side
  - IMAP: Internet Mail Access Protocol [RFC 3501] (port 143)
    - more features (more complex)
    - manipulation of stored messages on server
    - Maintain state for the user
  - HTTP: Hotmail , Yahoo! Mail, etc.
    - Slow

Outline

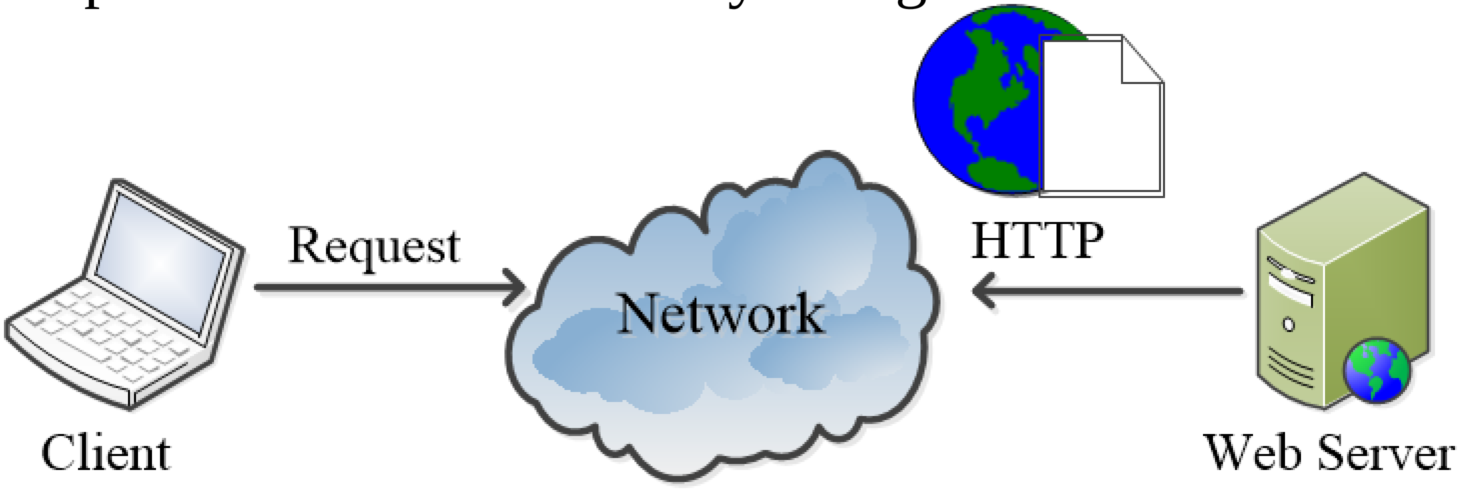
- Overview of application layer
- Important application layer protocols
  - DNS
  - FTP
  - Email
  - HTTP

Outline

- Overview of application layer
- Important application layer protocols
  - DNS
  - FTP
  - Email
  - HTTP

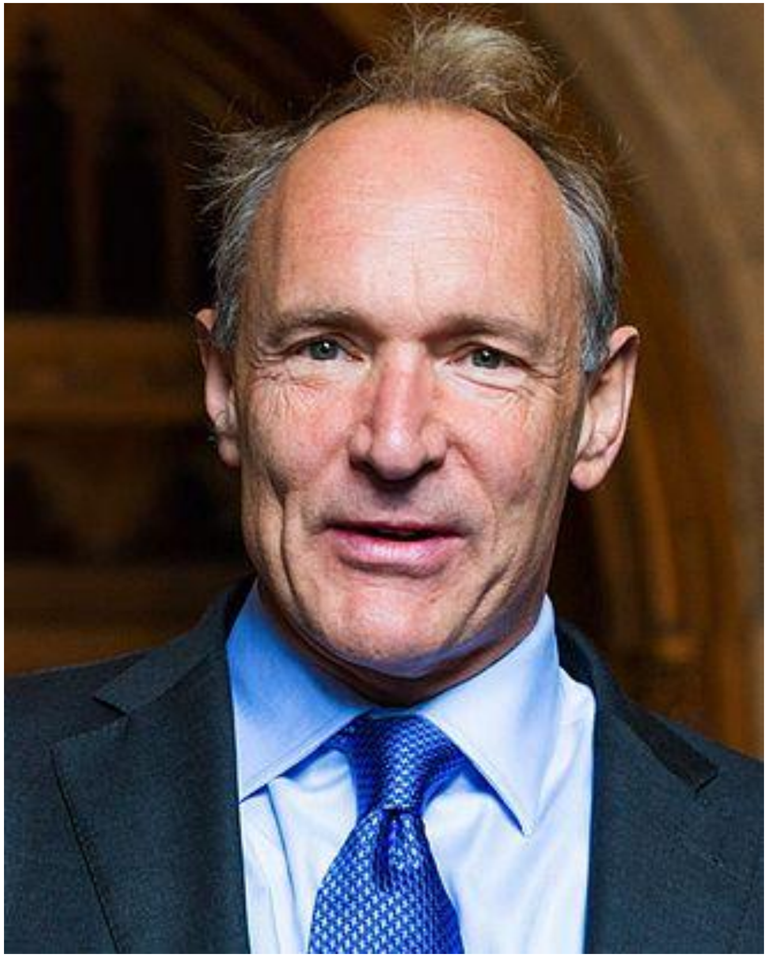
HTTP — The HyperText Transfer Protocol

- The HTTP, the Web's application-layer protocol, is at the heart of the Web.
  - [RFC1945, RFC2616]
- HTTP is a simple request-response protocol that normally runs over **TCP**.
  - HTTP has nothing to do with how a Web page is interpreted by a client.
- A **Web page** (also called a document) consists of objects. An object is simply a file — such as HTML file, a JPEG image, a Java applet, or a video clip — that is addressable by a single URL.



Sir Tim Berners-Lee

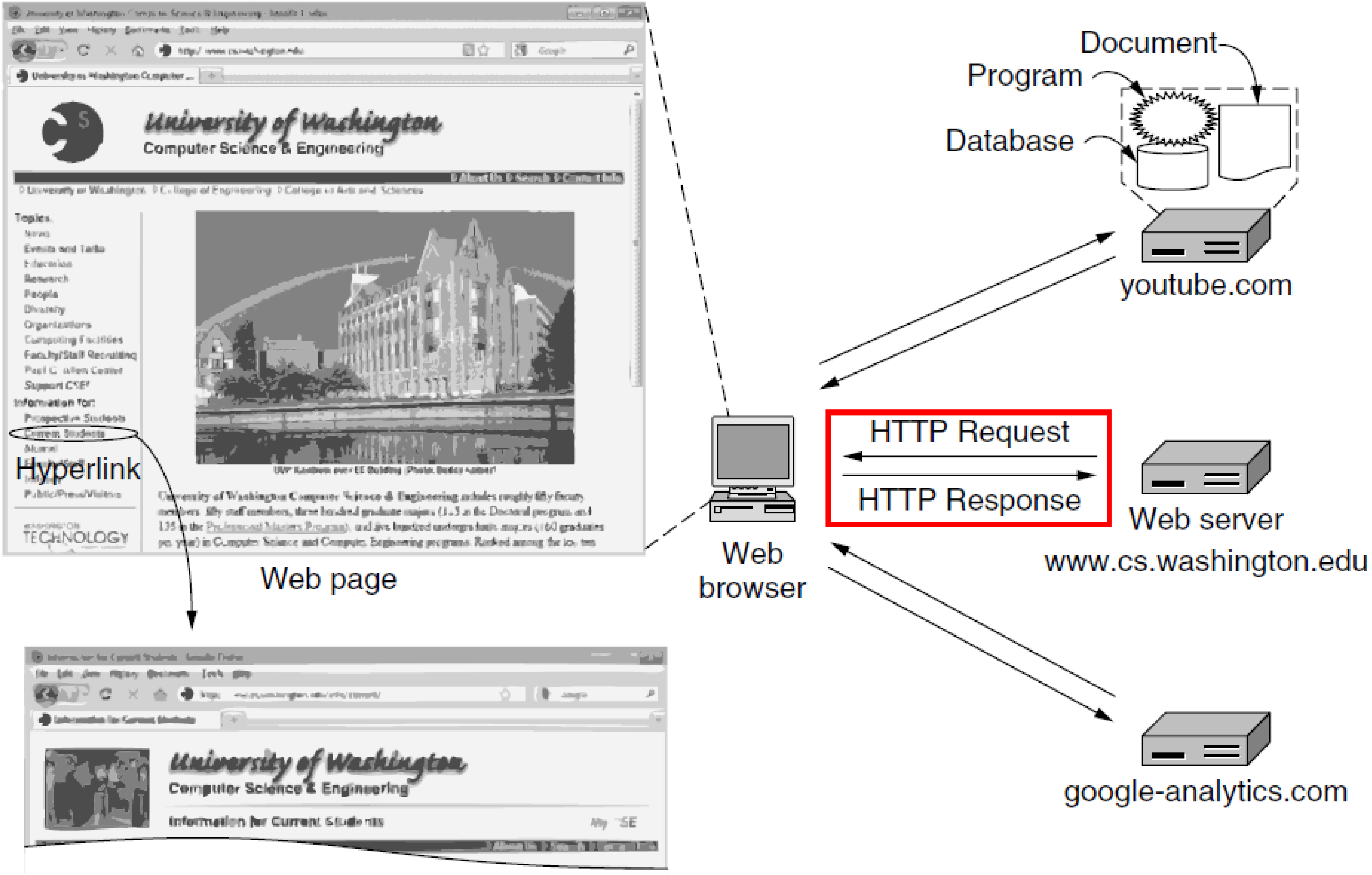
- Inventor of the Web
  - Dominant Internet application since mid 1990s
  - He now directs the W3C
- Developed Web at CERN in 1989
  - Browser, server and first HTTP protocol
  - Popularized via Mosaic (the first graphical browser developed by **Marc Andreessen** in1993), Netscape
  - First WWW conference in 1994…
  - Received the 2016 A. M. Turing Award "for inventing the World Wide Web, the first web browser, and the fundamental protocols and algorithms allowing the Web to scale."
  - The idea of have one page point to another, now call **hypertext**, was invented by a visionary MIT professor of EE, Vannevar Bush, in 1945.



Jun. 8, 1955 -

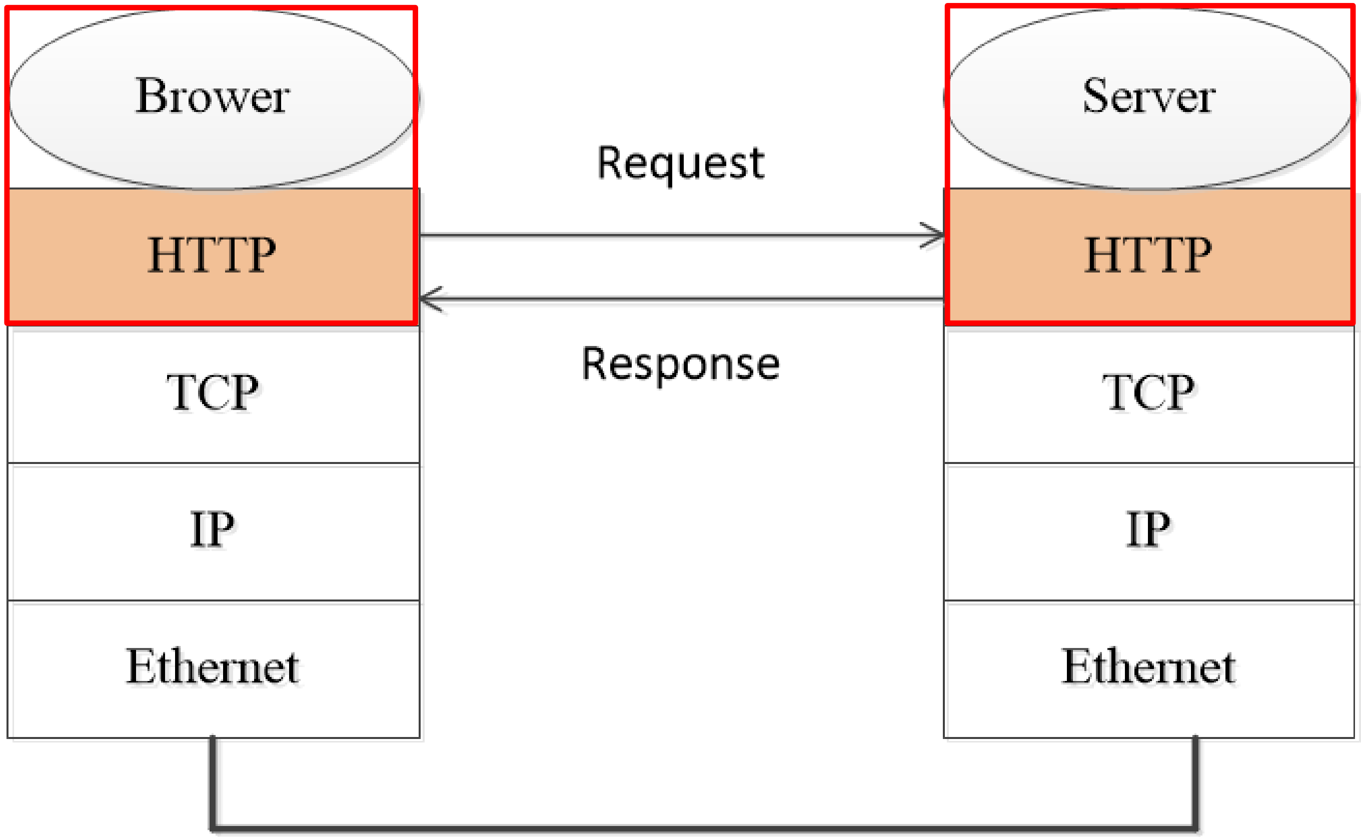
Web Context

Web page as a set of related Web resources — The content from these different servers is integrated for display by the browser.



HTTP Context

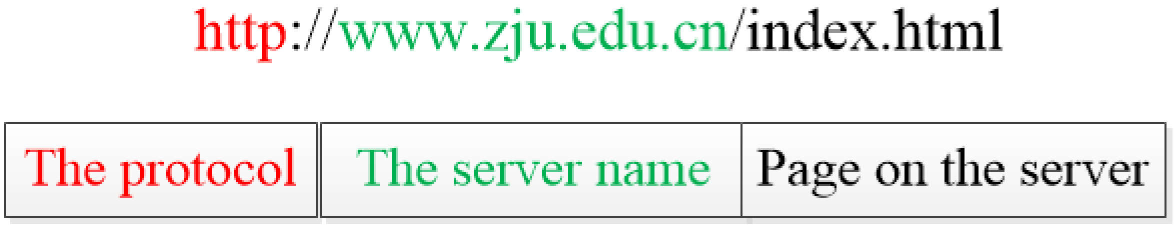
- HTTP is a request/response protocol for fetching Web resources
  - Runs on **TCP**, typically **port 80**
  - Part of browser/server applications, and Web browsers (such as Internet Explorer, and Google) implement the client side of HTTP.
  - HTTP is a **stateless** protocol. (The server sends requested files to clients without storing any state information about the client.) [5]



A Web server is always on, with a fixed IP address, and it services requests from potential millions of different browsers.

Fetching a Web page with HTTP (the Client Side)

- Start with the page URL

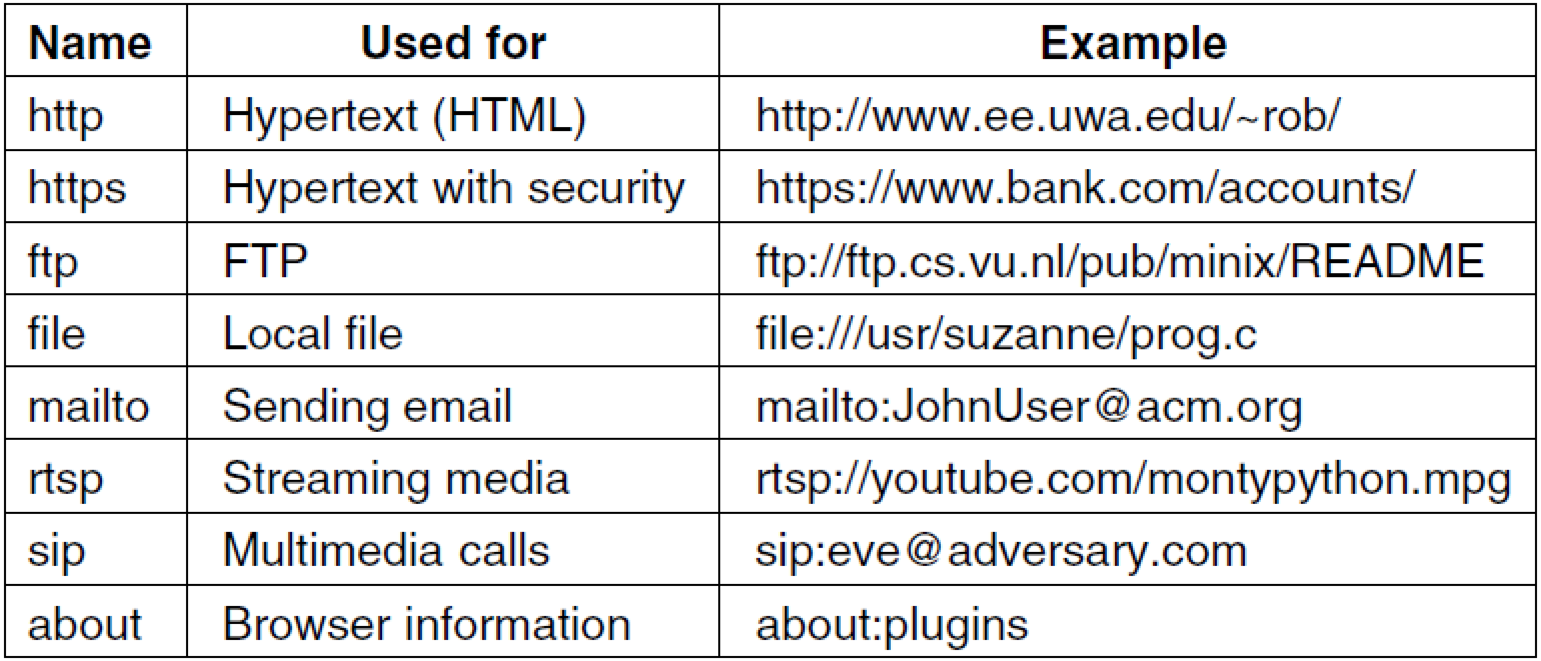


http://www.zju.edu.cn/index.html

- Steps:
  - Resolve the server name to IP address (DNS)
  - Set up TCP connection to the server
  - Send HTTP request for the page
  - (Await HTTP response for the page)
  - Execute / fetch embedded resources /render (不只是展示网页中内容，可能还要运行程序等。)
  - Clean up an idle TCP connections

The Browser (Client) Side

- The URL design is open-ended in the sense that it is straightforward to have browsers use multiple protocols to get different kinds of resources



Name	Used for	Example
http	Hypertext (HTML)	http://www.ee.uwa.edu/~rob/
https	Hypertext with security	https://www.bank.com/accounts/
ftp	FTP	ftp://ftp.cs.vu.nl/pub/minix/README
file	Local file	file:///usr/suzanne/prog.c
mailto	Sending email	mailto:JohnUser@acm.org
rtsp	Streaming media	rtsp://youtube.com/montypython.mpg
sip	Multimedia calls	sip:eve@adversary.com
about	Browser information	about:plugins

Figure 7-19. Some common URL schemes.

ftp: The Web makes it easy to obtain files placed on numerous FTP servers throughout the world by providing a simple, clickable interface instead of a command-line interface.

mailto: it allows users to send email from a Web browser. Most browser will respond when a mailto link is followed by starting the user’s mail agent to compose a message with the address field already filled in.

The Server Side

- Steps:

  1. Accept a TCP connection from a client (a browser).
  2. Get the path to the page, which is the name of the file requested.
  3. Get the file (from disk).
  4. Send the contents of the file to the client.
  5. Release the TCP connection.

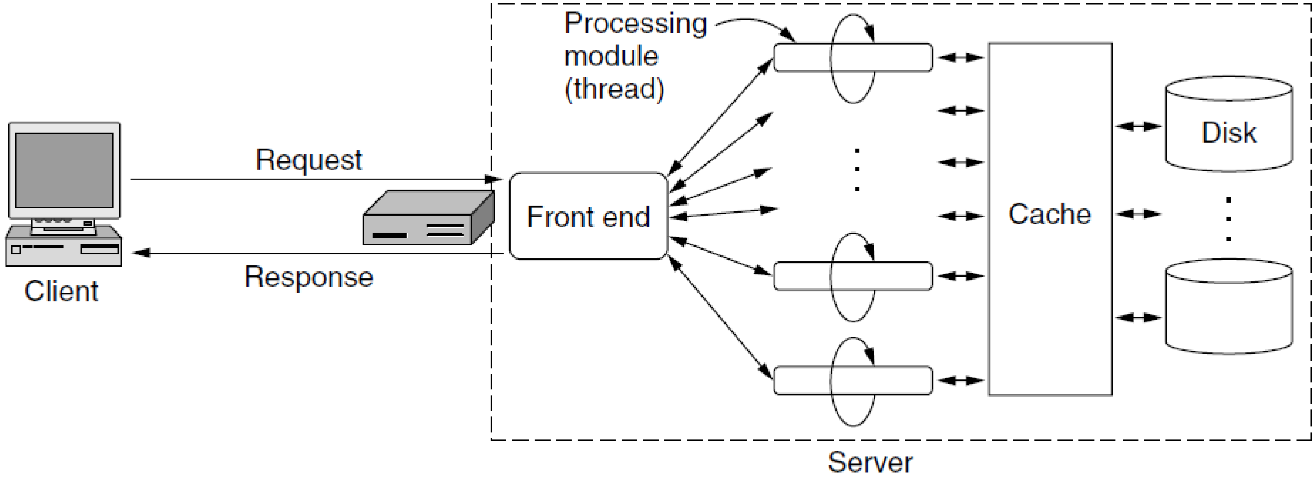
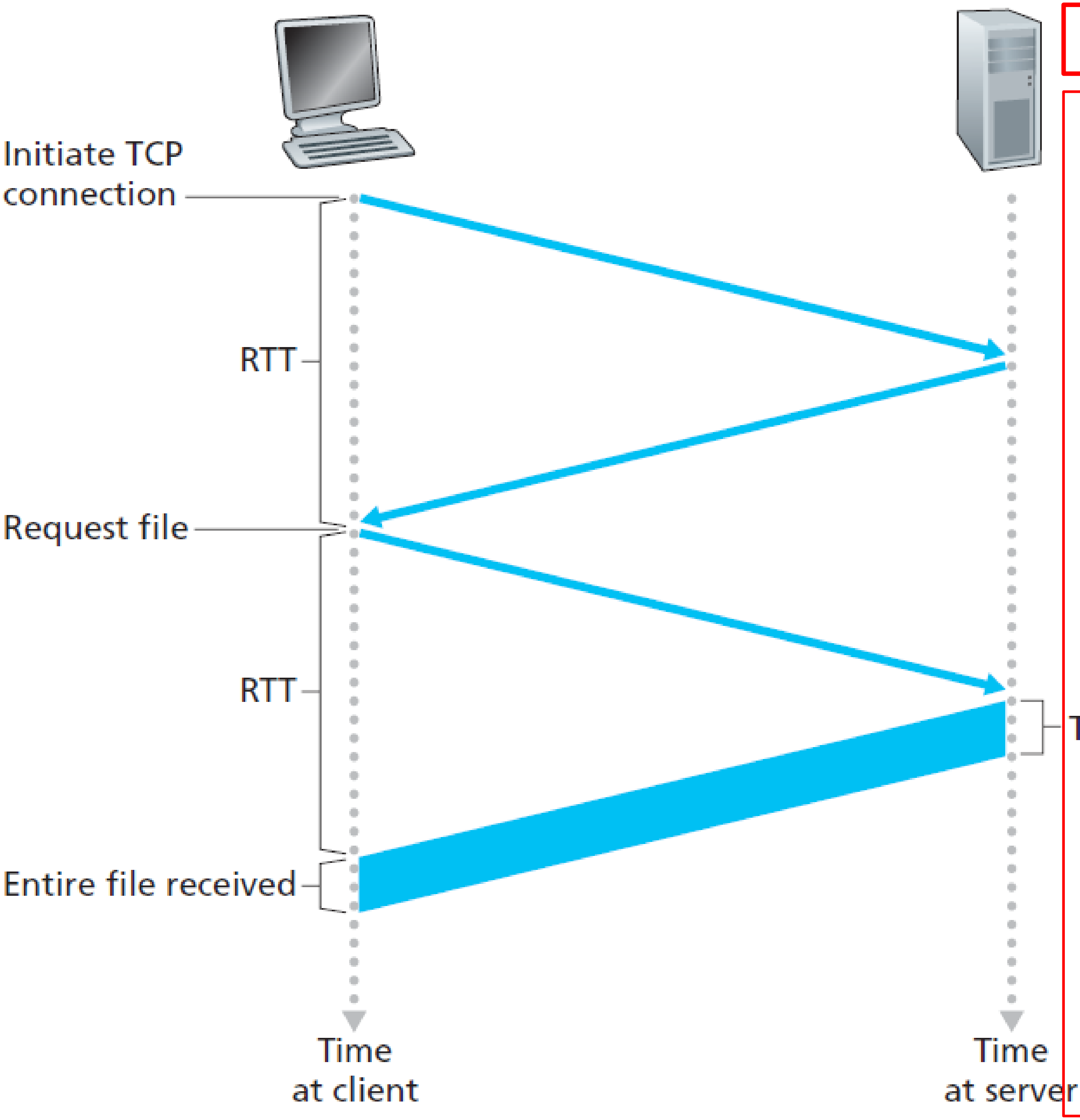


Figure 7-21. A multithreaded Web server with a front end and processing modules.



“Three-way Handshake”

1) The browser (client) initiates a TCP connection by sending a small TCP segment to the server.
2) The server acknowledges and responds with a small TCP segment.
3) The client sends the HTTP request message combined with the third part of the three-way handshake (the acknowledgement into the TCP connection).

Cookies (I)

- HTTP is a **stateless** protocol. [5]
- The simple request/response is not adequate when there are interactions between the users and Web sites. It is often desirable for a Web site to identify users.
  - Registration
  - E-commerce
- Solutions:
  - 1) IP address (but sometimes it does not work because of NAT, DHCP)
  - 2) Cookies (first implemented in the Netscape browser 1994, RFC2109, RFC2965)

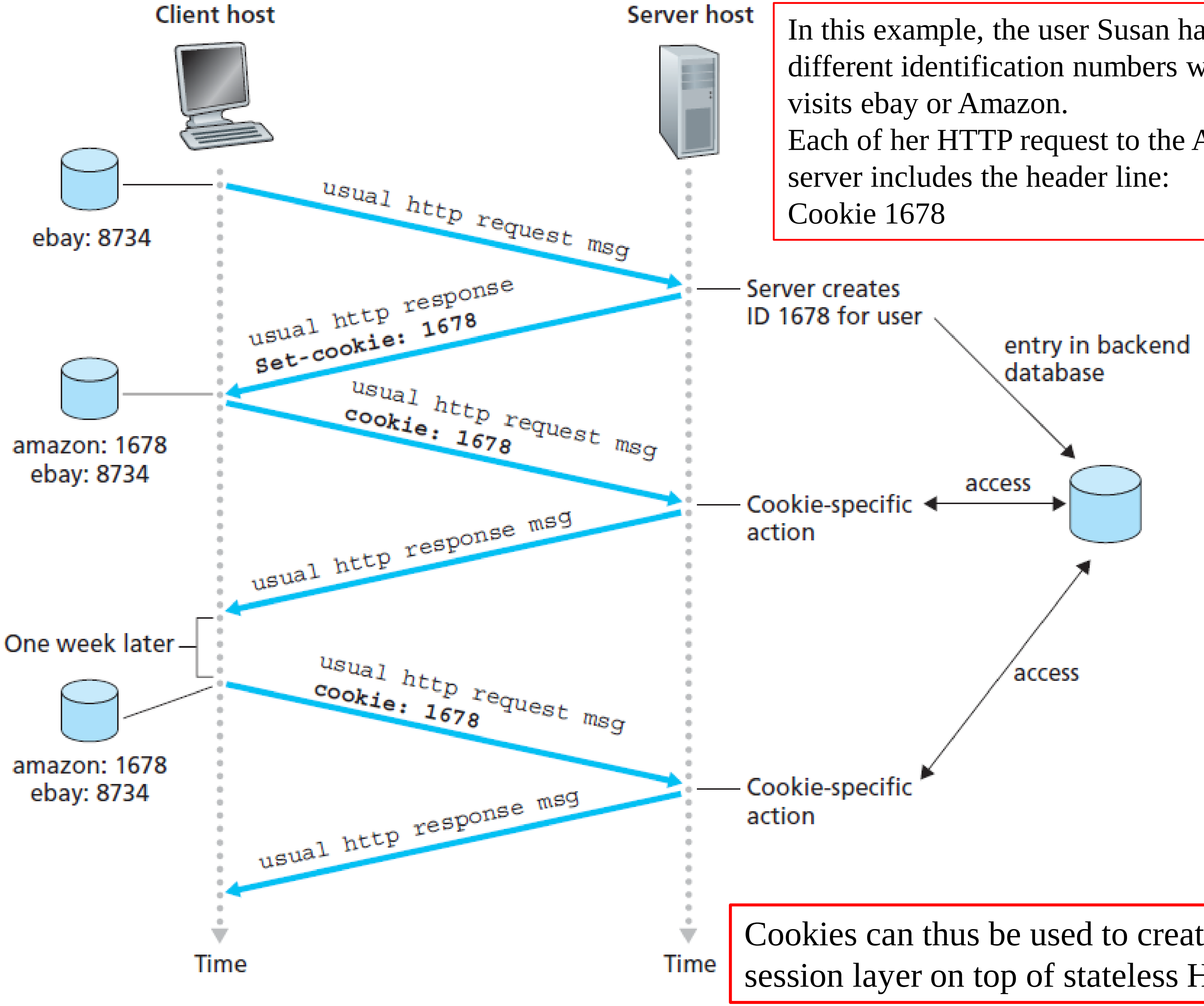
Cookies (II)

- An HTTP cookie (web cookie, browser cookie) is a small piece of data (at most 4KB) that a server sends to the user’s web browser
  - Typically, it is used to tell if two requests came from the same browser —— keeping a user logged-in
  - It remembers stateful information for the stateless HTTP protocol.

- Cookies are mainly used for three purposes:
  - Session management
    - Logins, shopping carts, game scores, or anything else the server should remember.
  - Personalization
    - User preferences, themes, and other settings
  - Tracking
    - Recording and analyzing user behavior (DoubleClick, Google Analytics专门从事Web tracking生意的)

Cookies (III)

- A cookie may contain up to five fields
  - The **Domain** tells where the cookie came from
  - The **Path** is a path in the server's directory structure that identifies which parts of the server's file tree may use the cookie.
  - The **Content** field is where the cookie's content is stored.
  - The **Expires** field specifies when the cookie expires.
    - If this field is absent, the browser discards the cookie when it exits. (**nonpersistent cookie**)
    - If a time and date are supplied, the cookie is said to be a **persistent cookie**.
    - To remove a cookie from a client's hard disk, a server just sends it again, but with an expiration time in the past.
  - The **Secure** field can be sent to indicate that the browser may only return the cookie to a server using a secure transport. This feature is used for e-commerce.



In this example, the user Susan has different identification numbers when she visits ebay or Amazon.
Each of her HTTP request to the Amazon server includes the header line:
Cookie 1678

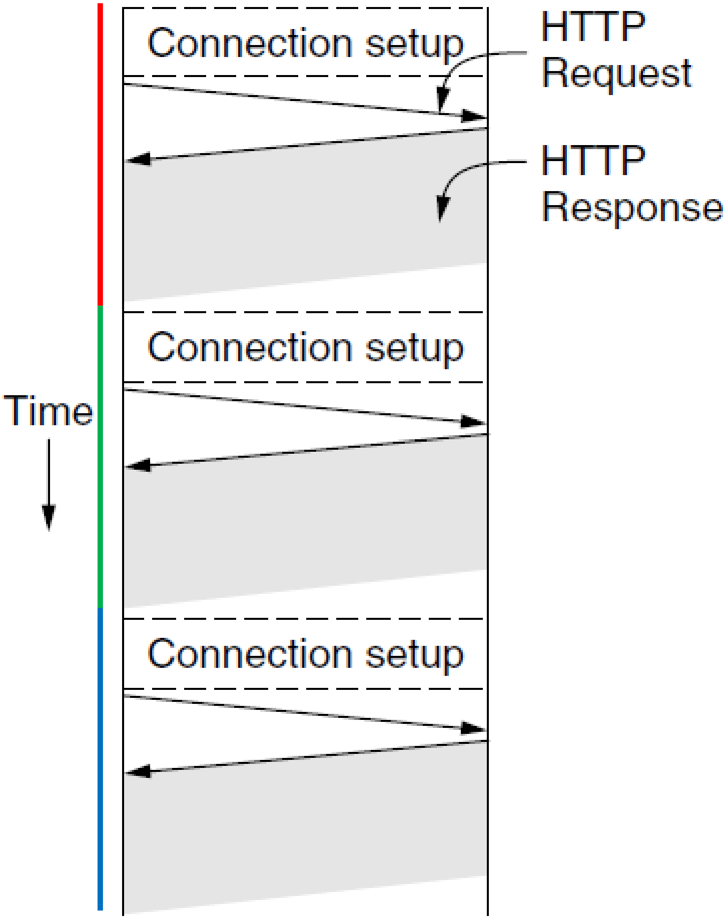
Cookies can thus be used to create a user session layer on top of stateless HTTP.

HTTP Performance

- PLT (Page Load Time) is the key measure of web performance
  - From click until user sees page
  - Small increases in PLT decrease sales
- PLT depends on many factors
  - Structure of page/content
  - HTTP (and TCP!) protocol
  - Network RTT and bandwidth

Early Performance (I)

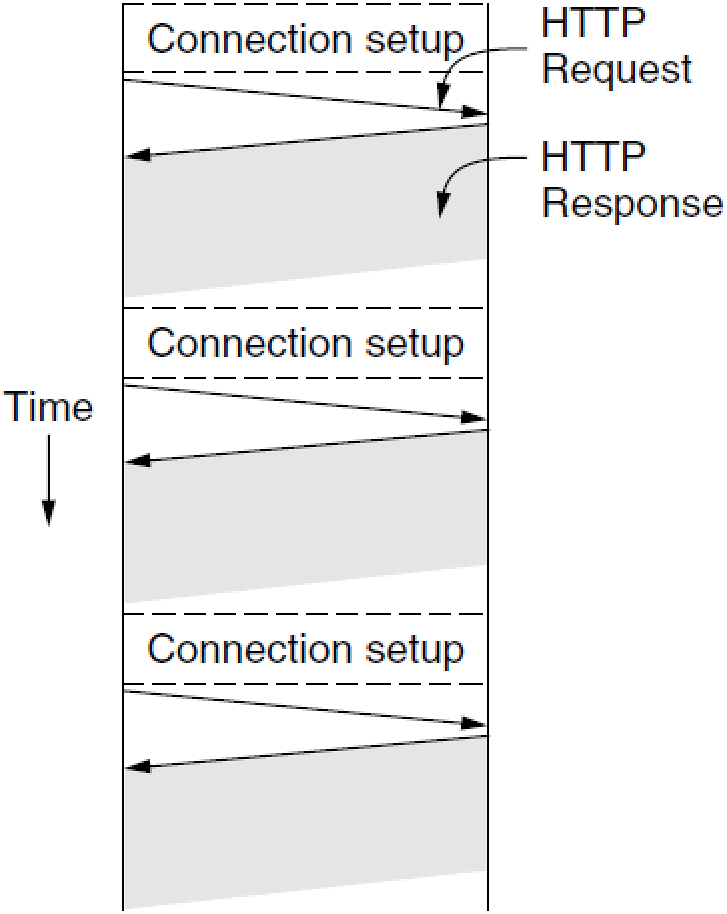
- HTTP/1.0 uses one TCP connection to fetch one web resource
  - Multiple connections and sequential requests
  - Made HTTP very easy to build
  - But gave fairly poor PLT …



One request per connection

Early Performance (II)

- Many reasons why PLT is larger than necessary
  - Sequential requests/responses, even when to *different servers*
  - Multiple TCP connection setups to the same server
  - Multiple TCP slow-start phases
- Network is not used effectively
  - Worse with many small resources/page



Ways to Decrease PLT

- Reduce content size for transfer
  - Compression techniques
    - Smaller images, gzip
- Change HTTP to make better use of available bandwidth
- Change HTTP to avoid repeated transfers of the same content
  - Caching, and proxies
- Move content closer to client
  - CDNs (Content Distribute Networks)

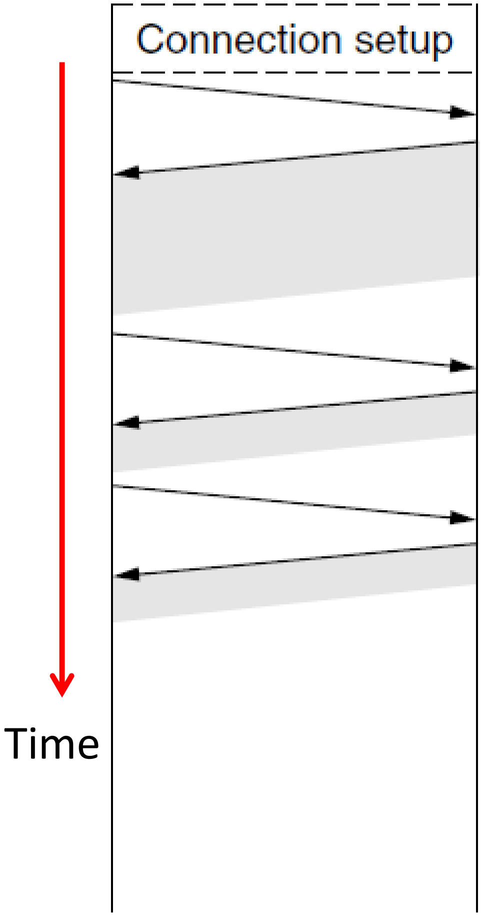
Change HTTP: **Parallel Connections**

- One simple way to reduce PLT
  - Browser runs **multiple HTTP**
  - Sever is unchanged; already handled concurrent requests for many clients
- How does this help?
  - Single HTTP wasn’t using network much efficiently
  - So parallel connections aren’t slowed much
  - But it has the same disadvantage as sequential connections — *extra overhead*
    - Each TCP connection requires at least one round-trip time to establish
    - TCP connection release cost
  - And parallel connections *compete* with each other for network resources
    - Because TCP performs congestion control for each connection independently
    - As a consequence, the connections compete against each other, causing added packet loss, and in aggregate are more aggressive users of the network than an individual connection.
      - Exacerbates network bursts and loss

Change HTTP: **Persistent Connections (I)**

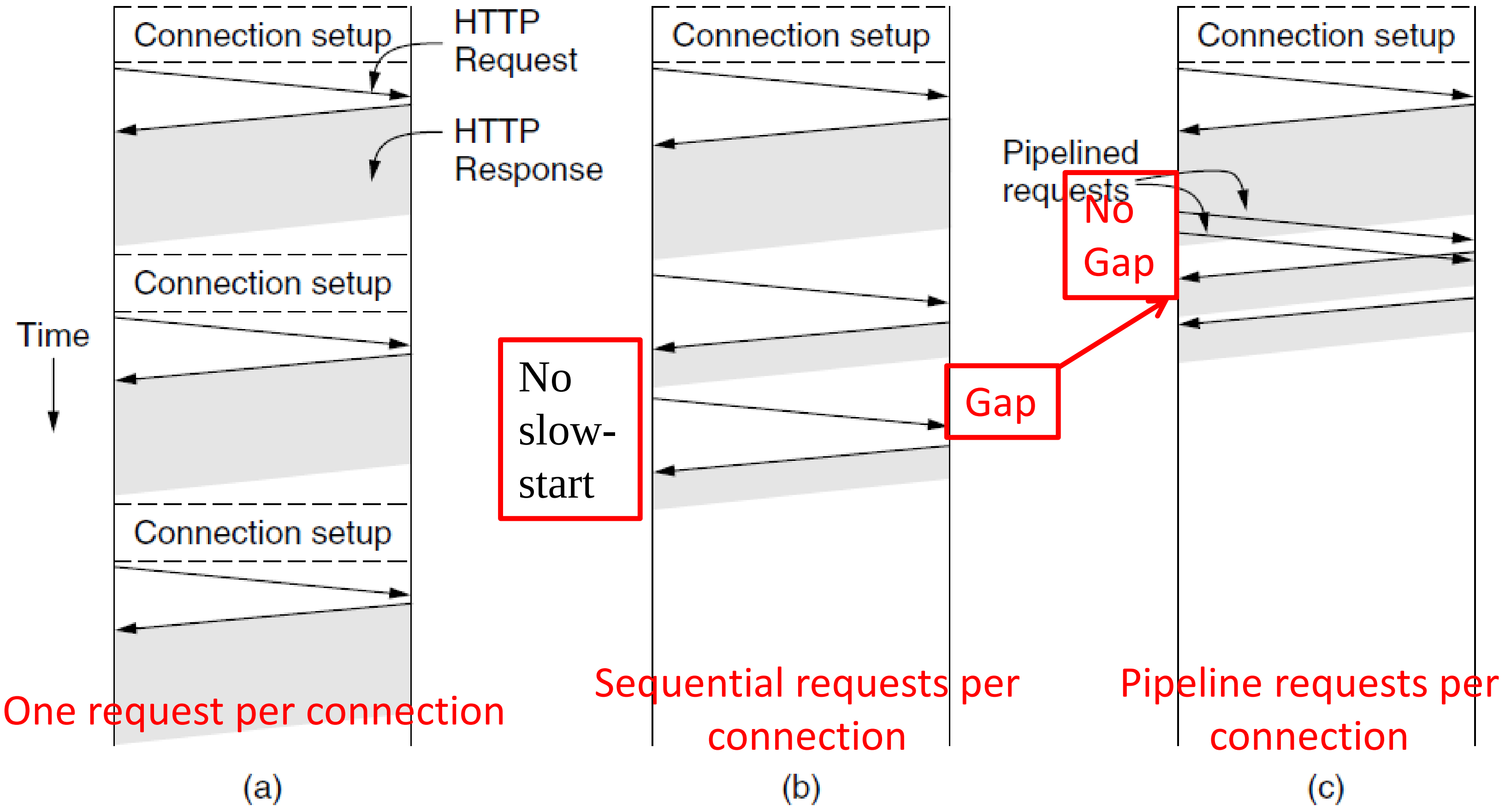
- HTTP1.1 uses Persistent connection (connection reuse)
  - Make 1 TCP connection to 1 server
  - Use it for **multiple HTTP requests**
  - Possible to **pipeline requests**, that is, send request 2 before the response of request 1 has arrived.
  - PLT benefits depending on page structure, but easy on network.

- Issues with persistent connections
  - How long to keep TCP connection?
    - Until they have been idle for a short time (e.g. 60seconds)
    - They have a large number of open connections and need to close some.
  - Can it be slower? (Yes.)



Persistent connection

Change HTTP: **Persistent Connections (II)**



No slow-start

Gap

No Gap

One request per connection

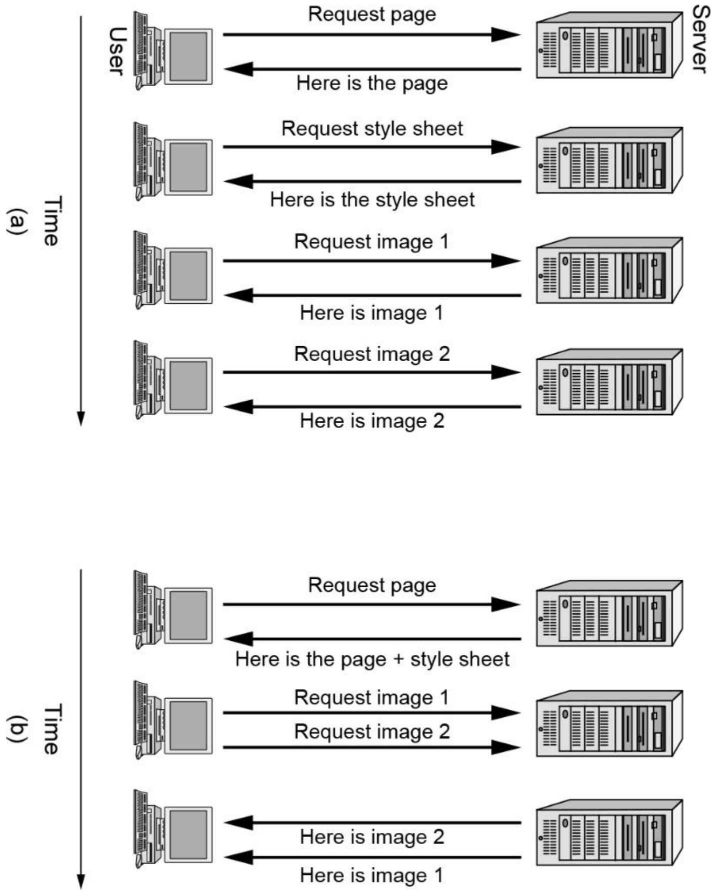
Sequential requests per connection

Pipeline requests per connection

Figure 7-36. HTTP with (a) multiple connections and sequential requests. (b) A persistent connection and sequential requests. (c) A persistent connection and pipelined requests.

Change HTTP: **HTTP/1.1 vs HTTP/2**

- 1) Server push
  - HTTP/2 allows the server to push out files that it knows will be needed but which the client may not know initially.
- 2) In HTTP/1.1, multiple requests can be sent consecutively over the same TCP connection, but the rules are that they must be processed in order and the results sent back in order. Whereas in HTTP/2, the responses can come back in any order.



(a) Getting a Web page in HTTP/1.1.
(b) Getting the same page in HTTP/2.

HTTP/1.1	HTTP/2
每个请求都需要单独建立一个 TCP 连接（虽然有 Keep-Alive 头字段可以在一定程度上保持连接复用，但效果有限）。	HTTP/2 采用了多路复用（Multiplexing）技术，它允许在一个 TCP 连接上同时发送多个请求和接收多个响应，而不需要像 HTTP/1.1 那样为每个请求单独建立连接。
HTTP/1.1 的请求和响应头部信息通常是未经压缩的文本格式，每次请求和响应都要完整地发送这些头部信息。	HTTP/2 采用了 HPACK 头部压缩算法，对请求和响应的头部信息进行高效压缩。它可以根据之前传输过的头部信息以及一些预设的规则，对重复出现的部分进行压缩处理，大大减少了头部信息占用的网络带宽。
HTTP/1.1 传输的数据格式是基于文本的，采用 ASCII 码进行编码。	HTTP/2 引入了二进制分帧层（Binary Framing Layer），它将所有传输的数据（包括请求、响应以及它们的头部和主体部分）都转换为二进制格式进行传输。
HTTP/1.1 没有明确的请求优先级机制。当浏览器同时发送多个请求（比如加载一个网页时，同时请求图像、脚本、样式表等资源），服务器会按照接收到请求的先后顺序来处理，无法根据资源的重要性或紧急程度进行有针对性的处理。	HTTP/2 具备明确的请求优先级机制。客户端（如浏览器）可以在发送请求时为不同的请求设置优先级，服务器收到这些请求后，会根据设置的优先级来安排处理顺序，优先处理重要性高、紧急程度高的请求，从而更合理地分配资源，提高用户体验。
HTTP/1.1 没有完善的流控制机制。	HTTP/2 建立了完善的流控制机制，通过窗口大小调整等方式来控制数据传输的速度。在网络拥塞时，它可以根据实际情况适当缩小窗口大小，减少数据传输量，避免过度占用网络资源；在网络状况良好时，又可以适当扩大窗口大小，加快数据传输速度，保证了网络传输的稳定性和高效性。

Change HTTP: **HTTP/3**

- HTTP/3: HTTP-over-**QUIC**
- The major distinction for HTTP/3 is the transport protocol that it used to support the HTTP messages: rather than relying on TCP, it relies on an augmented version of UDP called QUIC.

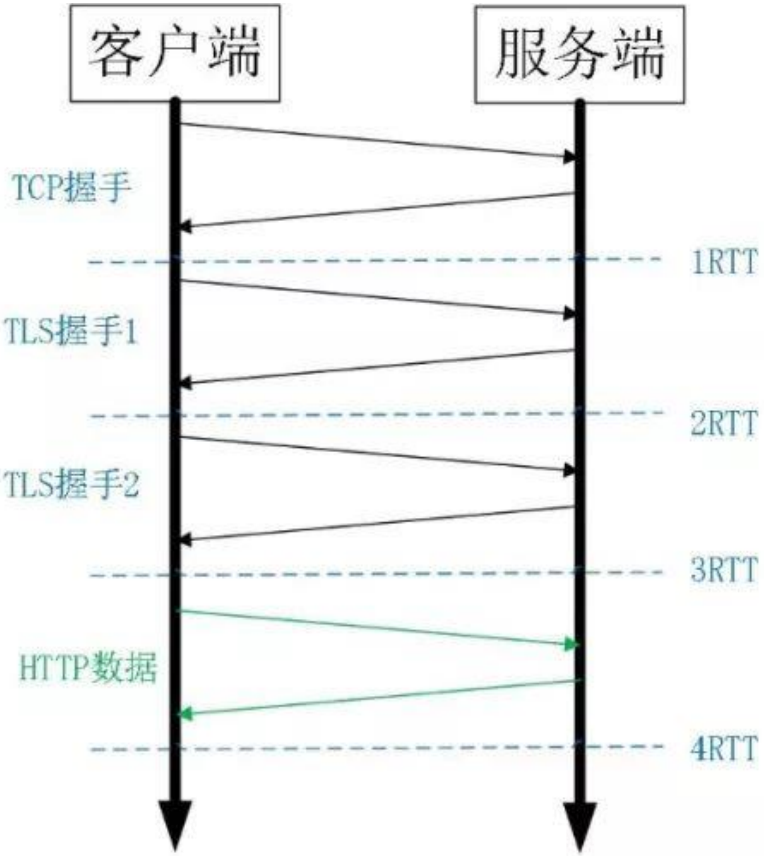
QUIC (Quick UDP Internet Connections)[9]

- QUIC 全称 (Quick UDP Internet Connection)，中文翻译成“快速 UDP 互联网连接”，是由 Google 提出的使用 UDP 进行多路并发传输的协议。

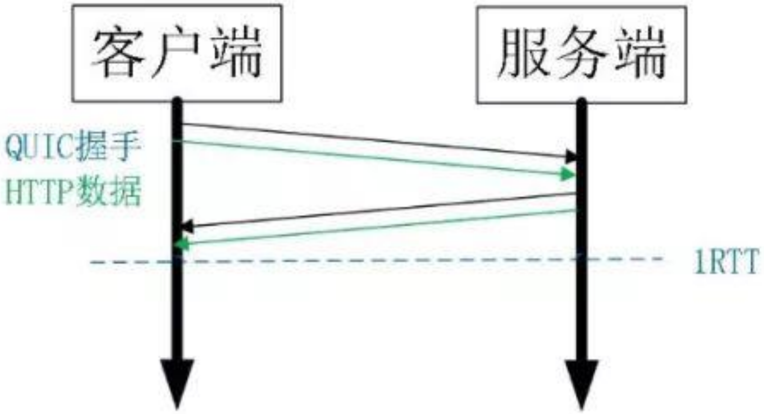
- QUIC 相比现在广泛应用的 http2+tcp+tls 协议有如下优势:
  - 1. 减少了 TCP 三次握手及 TLS 握手时间。
  - 2. 改进的拥塞控制。
  - 3. 避免队头阻塞的多路复用。(Multiplexing)
  - 4. 连接迁移。(handoff)
  - 5. 前向冗余纠错。

QUIC (Quick UDP Internet Connections) [9]

- 1. QUIC: Low latency to establish connection.



HTTPS



1) QUIC建立在 UDP 的基础上
2) 在实现前向加密的基础上，并且 0 RTT 的成功率相比 TLS 的 Session Ticket要高很多

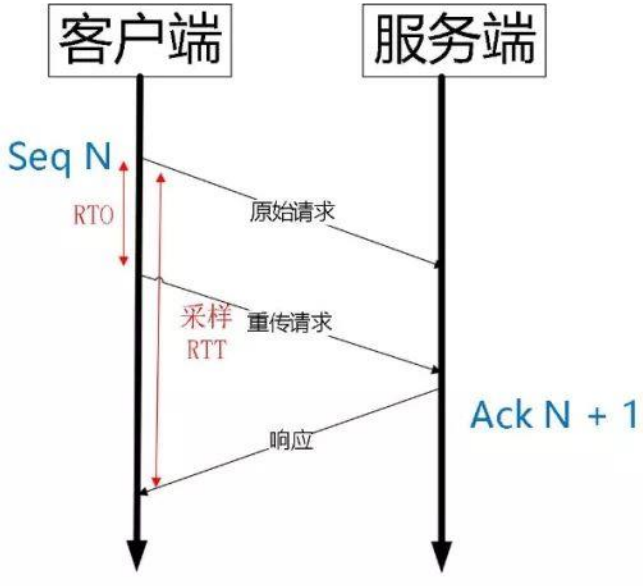
QUIC (Quick UDP Internet Connections) [9]

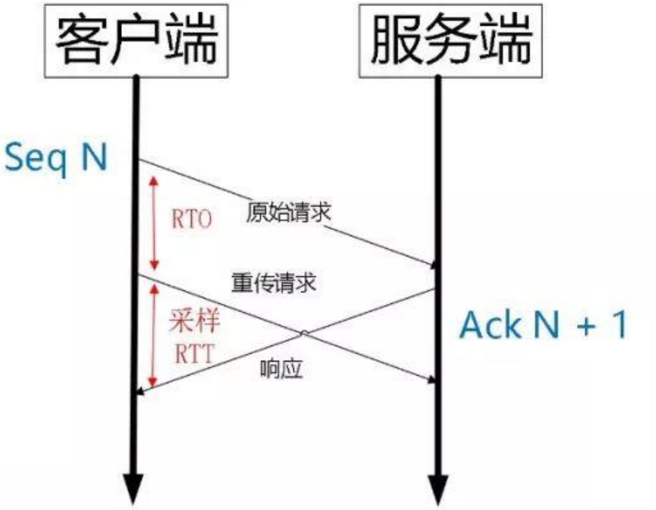
- 2. QUIC: Improved congestion control scheme, and a plug-and -play protocol
  - Reno
  - CUBIC [2]
  - BBR [3]

- Traditional TCP congestion control includes four key algorithms:
  - 1) Slow-start
  - 2) Congestion avoidance (ssthresh)
  - 3) Fast Retransmission (three duplicated ACKs trigger retransmission before time-out)
  - 4) Fast Recovery (Reno, the congestion window not slow-start after "packet loss" but additive increase from the new ssthresh = cwnd/2, pretend further duplicate ACKs are the expected ACKs)

QUIC (Quick UDP Internet Connections) [9]

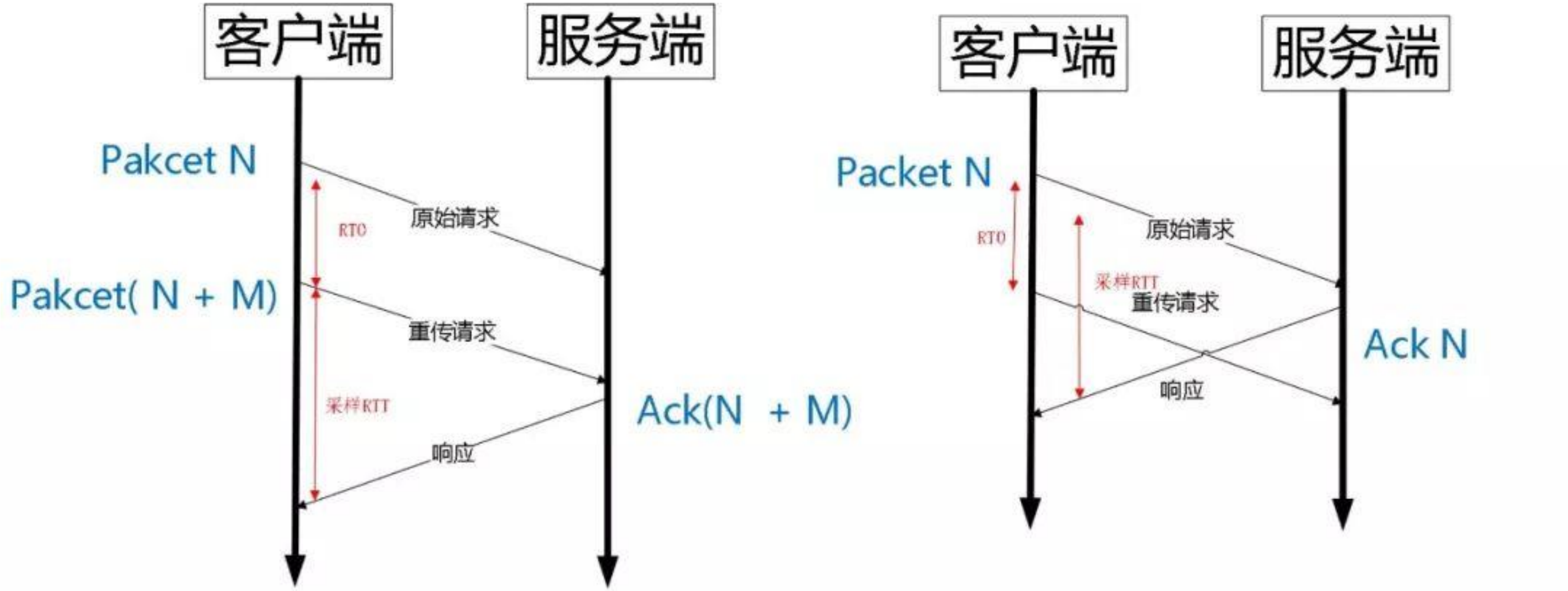
- 3. QUIC: reliable transmission based on **monotonically increased packed number**.
- To ensure reliable transmission, TCP counts on **the sequence number** and **ACK** of each segment.
  - There exists **ambiguity** of ACK when retransmission (ACK belong to the original segment or to the retransmission one)
  - This ambiguity will induce the inaccurate estimation of RTT (Karn’s algorithm)





QUIC (Quick UDP Internet Connections) [9]

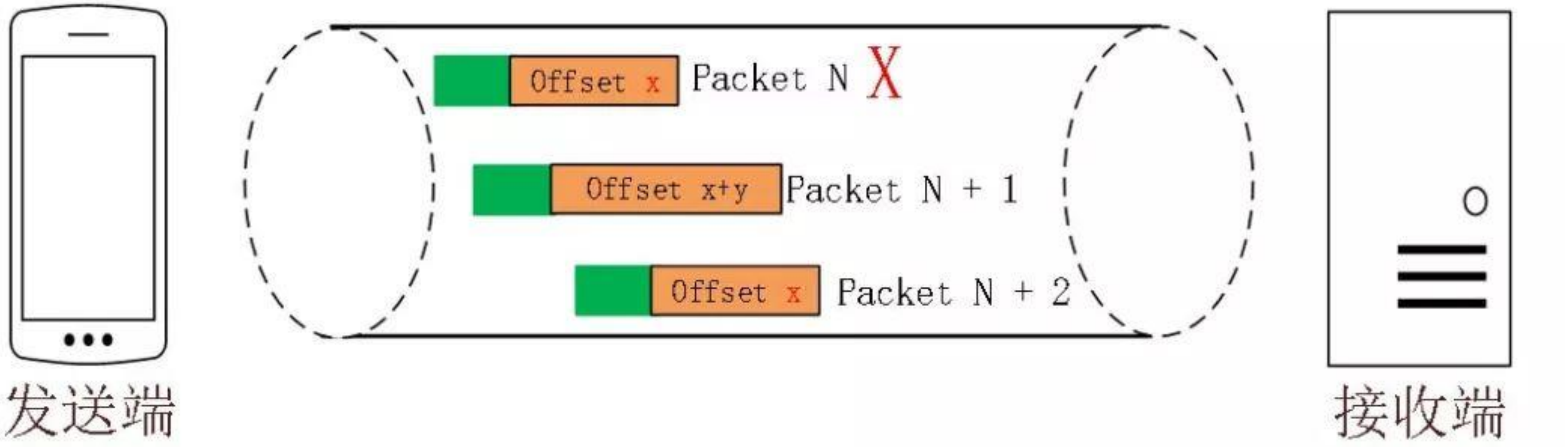
- 3. QUIC: reliable transmission based on **monotonically increased packet number**.
  - But just based on packet number only cannot ensure to receive data in order and to transmit reliably.
  - **Stream offset**.



There is no ambiguity of ACK in QUIC when retransmission!

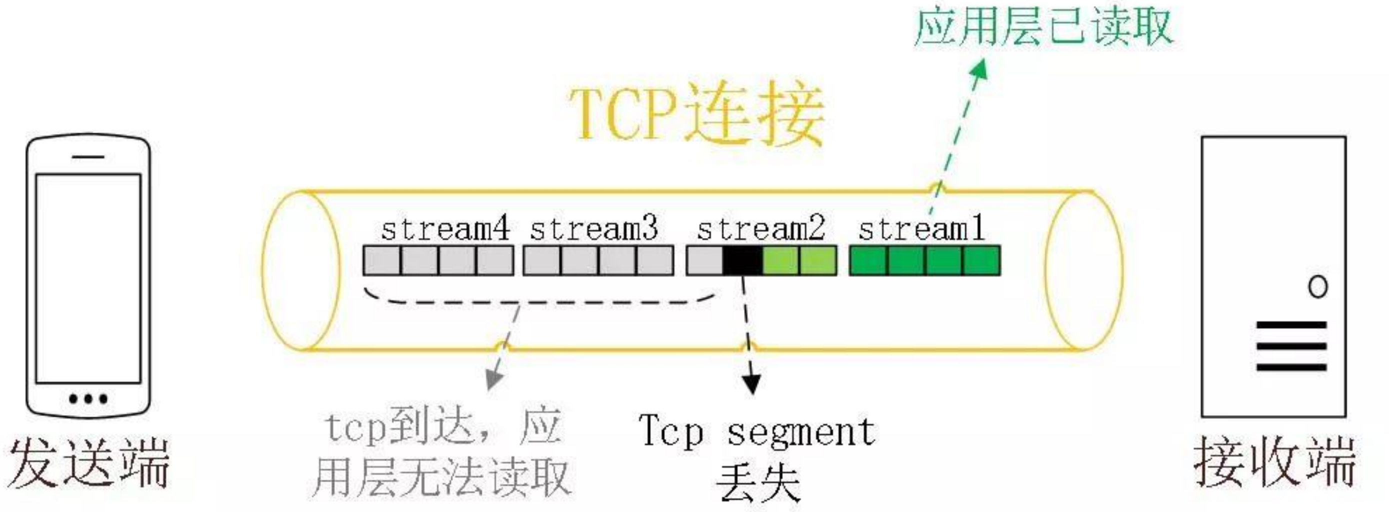
QUIC (Quick UDP Internet Connections) [9]

- 3. QUIC: reliable transmission based on **monotonically increased packet number**.
  - But just based on packet number only cannot ensure to receive data in order and to transmit reliably.
  - **Stream offset**.



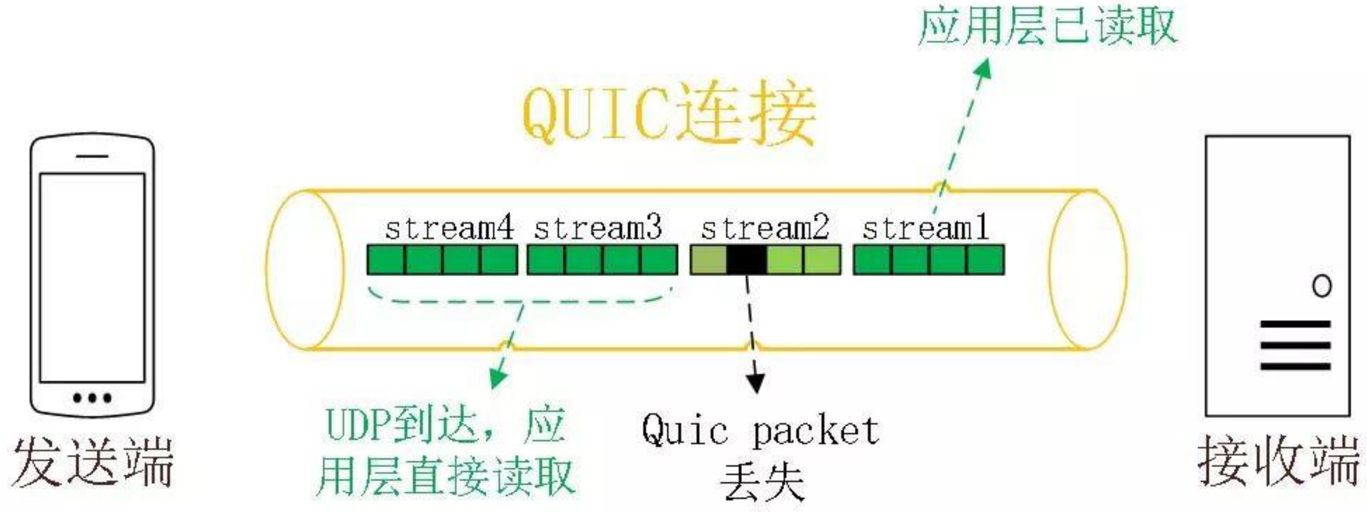
QUIC (Quick UDP Internet Connections)[9]

- 4. QUIC: removal of the **“Head-of-Line blocking” (HOL blocking) problem (队头阻塞问题)**
  - QUIC 的多路复用和 HTTP2 类似。在一条 QUIC 连接上可以并发发送多个 HTTP 请求 (stream)。
  - 多路复用是 HTTP2 最强大的特性，能够将多条请求在一条 TCP 连接上同时发出去。但也恶化了 TCP 的一个问题，队头阻塞，如下图示。
  - 不仅如此，由于 HTTP2 强制使用 TLS，还存在一个 TLS 协议层面的队头阻塞



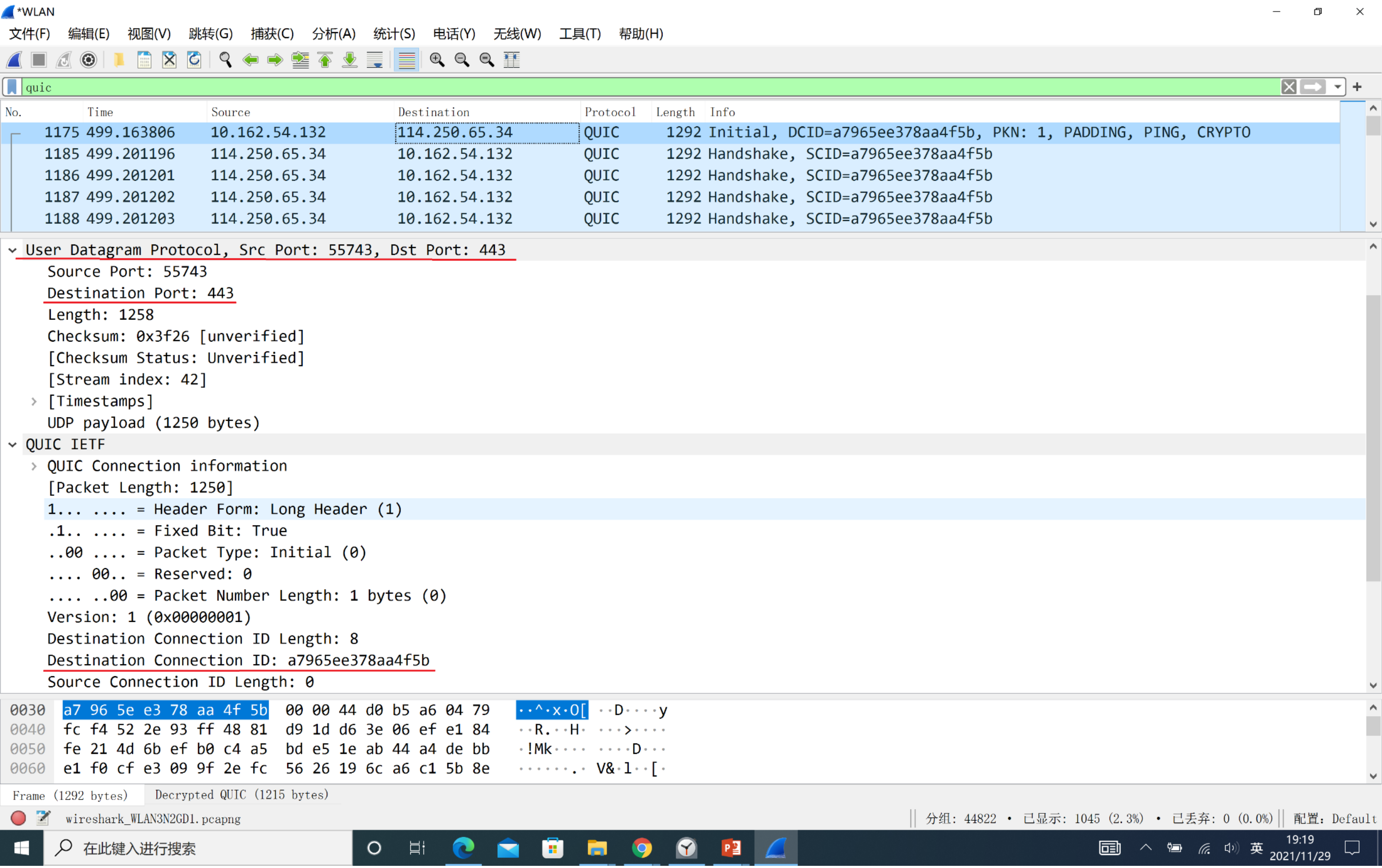
QUIC (Quick UDP Internet Connections) [9]

- 4. QUIC: removal of the **“Head-of-Line blocking” (HOL blocking) problem** (**队头阻塞问题**)
  - QUIC 最基本的传输单元是 Packet，不会超过 MTU 的大小，整个加密和认证过程都是基于 Packet 的，不会跨越多个 Packet。这样就能避免 TLS 协议存在的队头阻塞。
  - Stream 之间相互独立，比如 Stream2 丢了一个 Packet，不会影响 Stream3 和 Stream4。不存在 TCP 队头阻塞。



QUIC (Quick UDP Internet Connections)[9]

- 5. QUIC: **连接迁移**
  - 一条 TCP 连接 是由四元组标识的（源 IP，源端口，目的 IP，目的端口）
  - 什么叫连接迁移呢？就是当其中任何一个元素发生变化时，这条连接依然维持着，能够保持业务逻辑不中断。当然这里面主要关注的是客户端的变化，因为客户端不可控并且网络环境经常发生变化，而服务端的 IP 和端口一般都是固定的。
    - 比如大家使用手机在 WiFi 和 4G 移动网络切换时，客户端的 IP 肯定会发生变化，需要重新建立和服务端的 TCP 连接。
    - 又比如大家使用公共 NAT 出口时，有些连接竞争时需要重新绑定端口，导致客户端的端口发生变化，同样需要重新建立 TCP 连接。
  - 任何一条 QUIC 连接不再以 IP 及端口四元组标识，而是以一个 64 位的随机数作为 ID 来标识，这样就算 IP 或者端口发生变化时，只要 ID 不变，这条连接依然维持着，上层业务逻辑感知不到变化，不会中断，也就不需要重连。
    - 由于这个 ID 是客户端随机产生的，并且长度有 64 位，所以冲突概率非常低。



*WLAN
文件(F) 编辑(E) 视图(V) 跳转(G) 捕获(C) 分析(A) 统计(S) 电话(Y) 无线(W) 工具(T) 帮助(H)
quic

```
No.   Time        Source          Destination     Protocol  Length  Info
1175  499.163806  10.162.54.132   114.250.65.34   QUIC      1292    Initial, DCID=a7965ee378aa4f5b, PKN: 1, PADDING, PING, CRYPTO
1185  499.201196  114.250.65.34   10.162.54.132   QUIC      1292    Handshake, SCID=a7965ee378aa4f5b
1186  499.201201  114.250.65.34   10.162.54.132   QUIC      1292    Handshake, SCID=a7965ee378aa4f5b
1187  499.201202  114.250.65.34   10.162.54.132   QUIC      1292    Handshake, SCID=a7965ee378aa4f5b
1188  499.201203  114.250.65.34   10.162.54.132   QUIC      1292    Handshake, SCID=a7965ee378aa4f5b
```

```
User Datagram Protocol, Src Port: 55743, Dst Port: 443
    Source Port: 55743
    Destination Port: 443
    Length: 1258
    Checksum: 0x3f26 [unverified]
    [Checksum Status: Unverified]
    [Stream index: 42]
  > [Timestamps]
    UDP payload (1250 bytes)
QUIC IETF
  > QUIC Connection information
    [Packet Length: 1250]
    1... .... = Header Form: Long Header (1)
    .1.. .... = Fixed Bit: True
    ..00 .... = Packet Type: Initial (0)
    .... 00.. = Reserved: 0
    .... ..00 = Packet Number Length: 1 bytes (0)
    Version: 1 (0x00000001)
    Destination Connection ID Length: 8
    Destination Connection ID: a7965ee378aa4f5b
    Source Connection ID Length: 0
```

```
0030  a7 96 5e e3 78 aa 4f 5b  00 00 44 d0 b5 a6 04 79   ··^·x·O[ ··D····y
0040  fc f4 52 2e 93 ff 48 81  d9 1d d6 3e 06 ef e1 84   ··R.··H· ···>····
0050  fe 21 4d 6b ef b0 c4 a5  bd e5 1e ab 44 a4 de bb   ·!Mk···· ····D···
0060  e1 f0 cf e3 09 9f 2e fc  56 26 19 6c a6 c1 5b 8e   ·······. V&·l··[·
```

Frame (1292 bytes)   Decrypted QUIC (1215 bytes)

wireshark_WLAN3N2GD1.pcapng   分组：44822 • 已显示：1045 (2.3%) • 已丢弃：0 (0.0%)   配置：Default

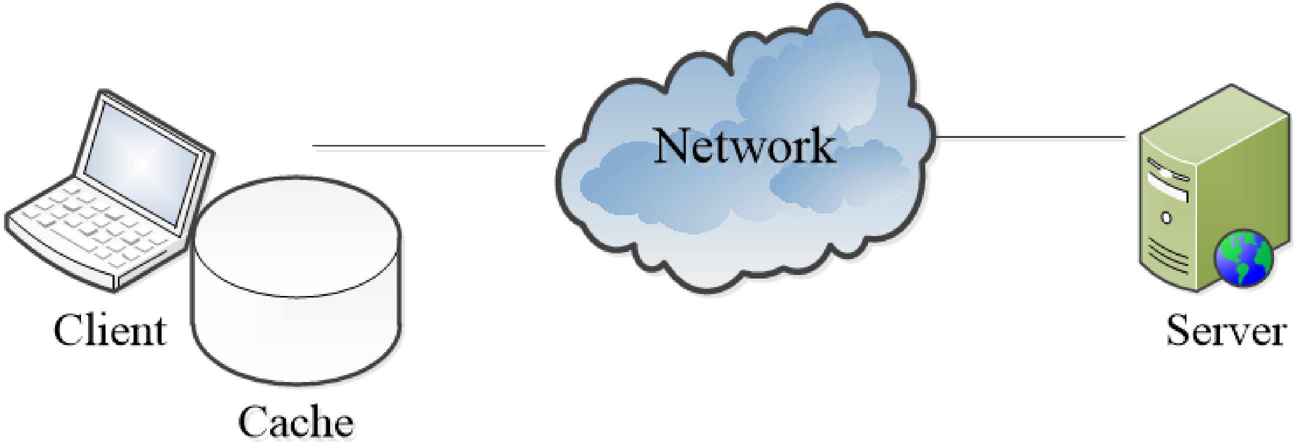
Port 443: HTTPS, Destination Connection ID: a7965ee378aa4f5b (64bit), Google browser.

Ways to Decrease PLT

- Reduce content size for transfer
  - Compression techniques
    - Smaller images, gzip
- Change HTTP to make better use of available bandwidth
- Change HTTP to avoid repeated transfers of the same content
  - Caching, and proxies
- Move content closer to client
  - CDNs (Content Distribute Networks)

Web Caching: avoid repeated transfers of the same content (I)

- Users often revisit web pages, and it is big win if we can reuse local copy.
- HTTP has built-in support to help clients identify when they can safely reuse pages.
  - **Caching**
  - This support improves performance by reducing both network traffic and latency.
  - The key question is <u>when is it OK to reuse the local copy</u>?



Web Caching: avoid repeated transfers of the same content (II)

- HTTP uses two strategies to tackle this problem
  - 1) **Page validation** — locally determine copy is still valid
    - Based on expiry information such as the "**Expires**" header from server
      - The Expires header returned when the cached page was originally fetched and the current date and time can be used to make determination
    - Or use a heuristic to guess (cacheable, freshly valid, not modified recently)
      - The **Last-Modified** header
      - The cacheability of a page may vary wildly over time.
        - » For example, the stock market might have closed for the day so that the page will not change for hours, but it will change rapidly once the next trading session starts.
    - The advantage is that content is then available right away

Web Caching: avoid repeated transfers of the same content (III)

- 2) Revalidate copy with remote server
  - Based on timestamp of copy such as "**Last-Modified**" header from server
    - If the client has the time a cached page was last updated from the "Last-Modified" header. It can send this time to the server using the **If-Modified-Since** header to ask for the page only if it has been changed in the meantime.
    - **A conditional GET**

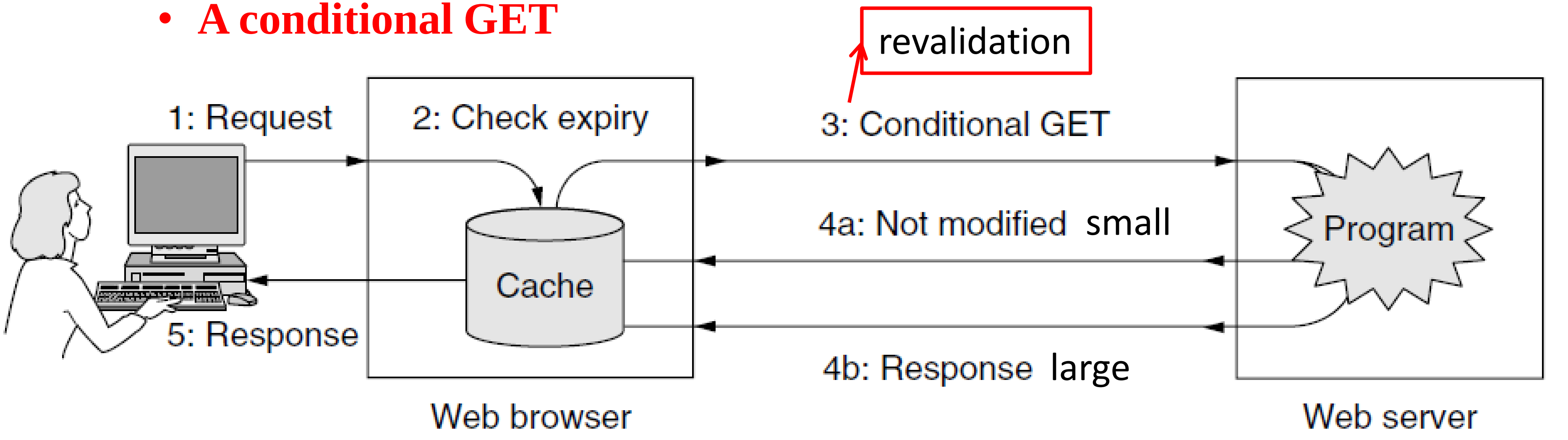


Figure 7-40. HTTP caching.

Web Caching: avoid repeated transfers of the same content (IV)

- 2) Revalidate copy with remote server
  - Or based on content of copy such as “**Etag**” header from server
    - The “Etag” is a short name for the content of the page, like a checksum but better. (It can be a cryptographic hash)
  - The client can validate cached copies by sending the server an “if-None-Match” header listing the tags of the cached copies.
    - If any of the tags match the content that the server would respond with, the corresponding cached copy may be used.
    - This method can be used when it is not convenient or useful to determine freshness.
      - For example, a server may return different content for the same URL depending on what languages and MIME types are preferred.
- The advantage is that content is available after one RTT.

Web Caching: avoid repeated transfers of the same content (V)

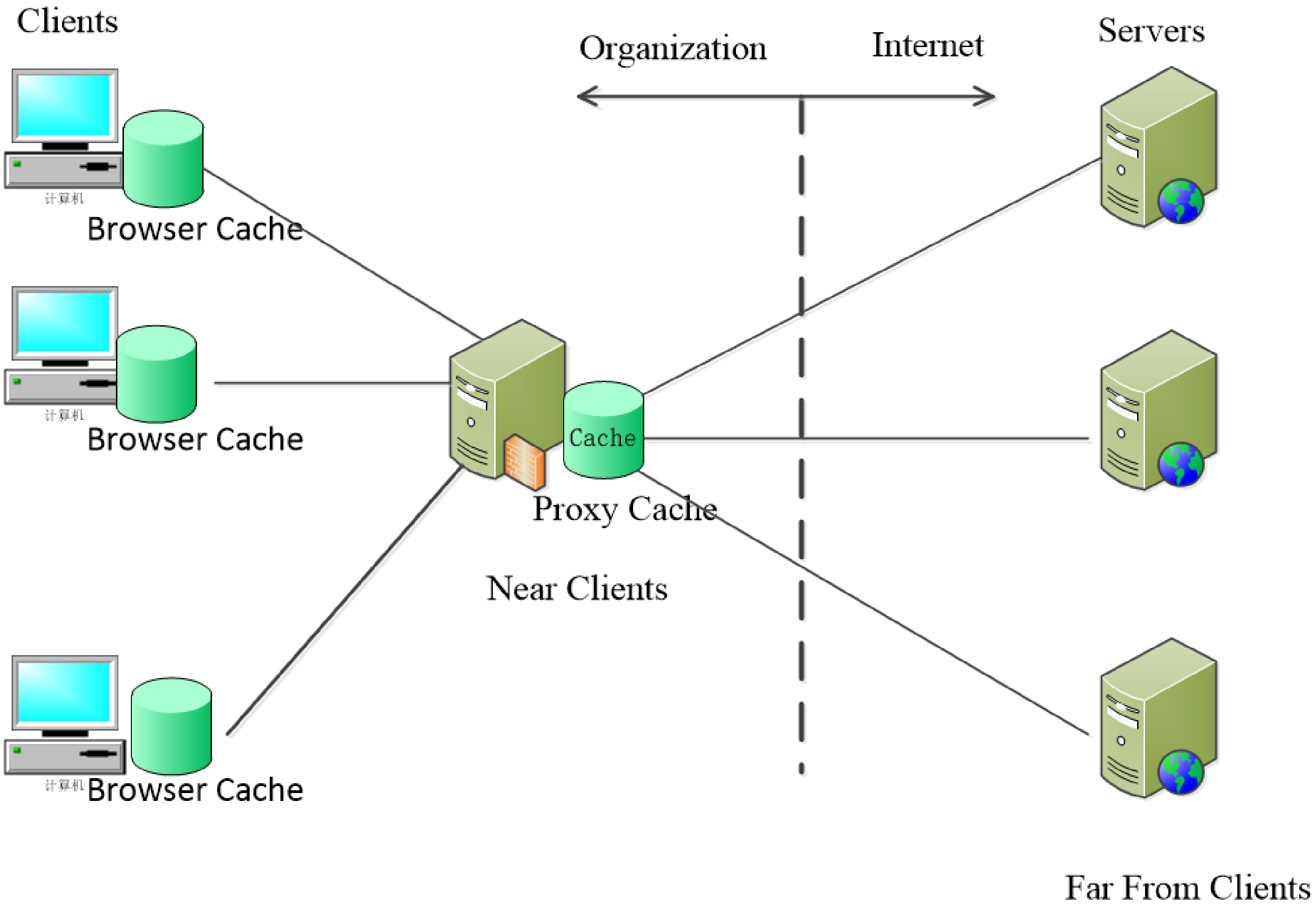
- Both of these caching strategies are overridden by the directives carried in the "Cache-Control" header. These directives can be used to restrict caching when it is **not** appropriate:
  - A dynamic page
  - Pages that required authorization are also not cached.

Web Proxies: avoid repeated transfers of the same content (I)

- Place intermediary between pool of clients and external web servers
  - Benefits for clients include greater caching and security checking
  - Organizational access policies too!
- Proxy Caching
  - Clients benefit from larger, shared cache
  - Benefits limited by secure/dynamic content, as well as “long tail”
    - Here the “long tail” is unpopular documents.

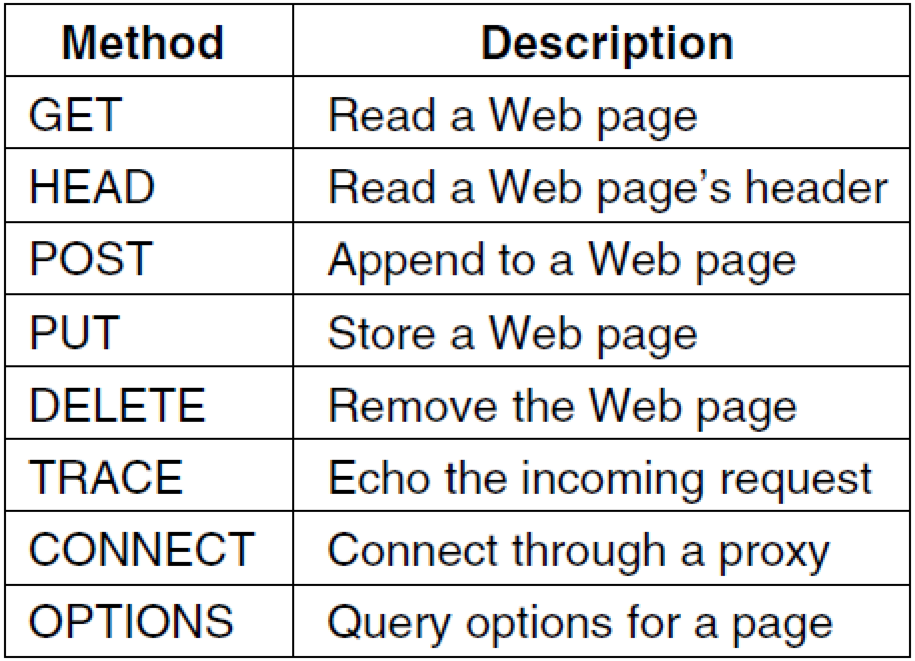
Web Proxies: avoid repeated transfers of the same content (II)

- Clients contact proxy; proxy contacts server
  - A Web proxy is both a server and a client at the same time.



HTTP Message Format: Request

- Originally a simple protocol, with many options added over time
  - Text-based (ASCII) commands: request lines, header lines
  - The request line has three fields: the **method** field, the **URL** field, and the **HTTP version** field.
- <u>Methods used in the **request**</u>



Method	Description
GET	Read a Web page
HEAD	Read a Web page’s header
POST	Append to a Web page
PUT	Store a Web page
DELETE	Remove the Web page
TRACE	Echo the incoming request
CONNECT	Connect through a proxy
OPTIONS	Query options for a page

Figure 7-37. The built-in HTTP request methods.

- The **GET** method requests the server to send the page.
- The **POST** method is used when a user fills out *a form*. It uploads the data to the server. The server then does something with the data that depends on the URL.
- The **PUT** method allows a user to upload an object to a specific path (directory) on a specific Web server.

HTTP Message Format

- The request line (e.g. the line with the GET method) may be followed by additional lines with more information. They are called **request headers**.
  - This information can be compared to the parameters of a procedure call.
  - Responses may also have response headers.

Summary of Fig 7-39

Function	Example Headers
Browser capabilities (client → server)	User-agent, Accept, Accept-Charset, Accept-Encoding, Accept-Language
Cache related (mixed directions)	If-Modified-Since, If-None-Match, Last-Modified, Expires, Date, Cache-Control, Etag
Browser context (client → server)	Host, Authorization, Referer, Cookie
Content delivery (server → client)	Content-Encoding, Content-Language, Content-Length, Content-Type, Content-Range, Set-Cookie

Request Headers (I)

- The **User-Agent** header allows the client to inform the server about its browser implementation (e.g. Mozilla/5.0 and Chrome/5.0.375.125).
  - This information is useful to let server tailor their responses to the browser, since different browsers can have widely varying capabilities and behaviors.
- The four **Accept** headers tell the server that the client is willing to accept in the event that it has a limited repertoire of what is acceptable.
  - Accept: MIME types
  - Accept-Charset: the character set (ISO-8859-5 or Unicode-1-1)
  - Accept-Encoding: deal with compression methods (e.g., gzip)
  - Accept-Language

Request Headers (II)

- The **If-Modified-Since** and **If-None-Match** headers are used with caching.
  - They let the client ask for a page to be sent only if the cached copy is no longer valid.
- The **Host** header names the server. It is taken from the URL. This header is mandatory.
  - It is used because some IP addresses may serve multiple DNS names and the server needs some way to tell which host to had the request to.
- The **Authorization** header is needed for pages that are protected.

Request Headers (III)

- The **Referer** header: the client uses the misspelled Referer (早期HTTP规范中拼写错误，为了保持向后兼容就将错就错了。) header to give the URL that referred to the URL that is now requested.
  - It tells servers how a client arrived at the page. (Referer会告诉服务器我是从哪个页面链接过来的，服务器借此可以获得一些信息用于处理。)
- The **Set-Cookie** header is how servers send cookies to clients.
  - The client is expected to save the cookie and return it on subsequent request to the server by using the Cookie header.
- The **Last-Modified** header tells when the page was last modified.
  - Related to page caching
- The **Expires** header tells how long the page will remain valid.
  - Related to page caching

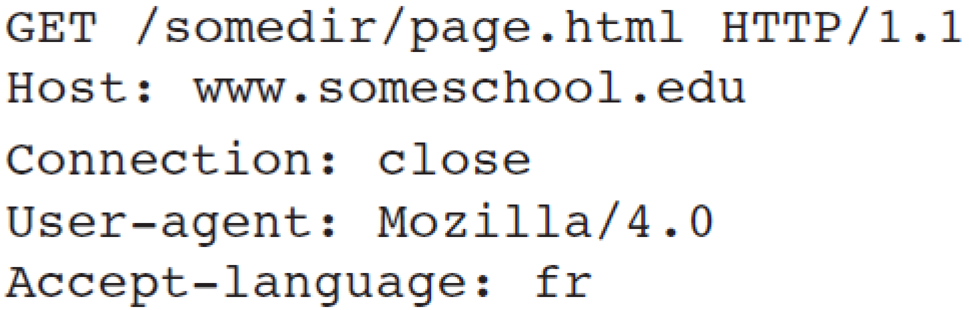
Request Headers (IV)

- The **Location** header is used by the server to inform the client that it should try a different URL.
  - This can be used if the page has moved or allow multiple URLs to refer to the same page (possibly on different servers).
  - It is also used for companies that have a main Web page in the com domain but redirect clients to a national or regional page based on their IP addresses or preferred language.
- The **Accept-Ranges** header: if a page is very large, a small client may not want it all at once. Some servers will accept requests for byte ranges, so the page can be fetched in multiple small units.

Request Headers (V)

- The **Date** header can be used in *both directions* and contains the time and date the message was sent.
- The **Range** header tells the byte range of the page that is provided by the response.
- The **ETag** header gives a short tag that serves as a name for the content of the page. It is used for *caching*.
- The **Cache-Control** header gives other explicit instructions about how to *cache pages*.
- The **Upgrade** header is used for switching to a new communication protocol.

HTTP Request Message Example



```
GET /somedir/page.html HTTP/1.1
Host: www.someschool.edu
Connection: close
User-agent: Mozilla/4.0
Accept-language: fr
```

1) **The request line** has three fields: *the method field, the URL field, and the HTTP version field.*

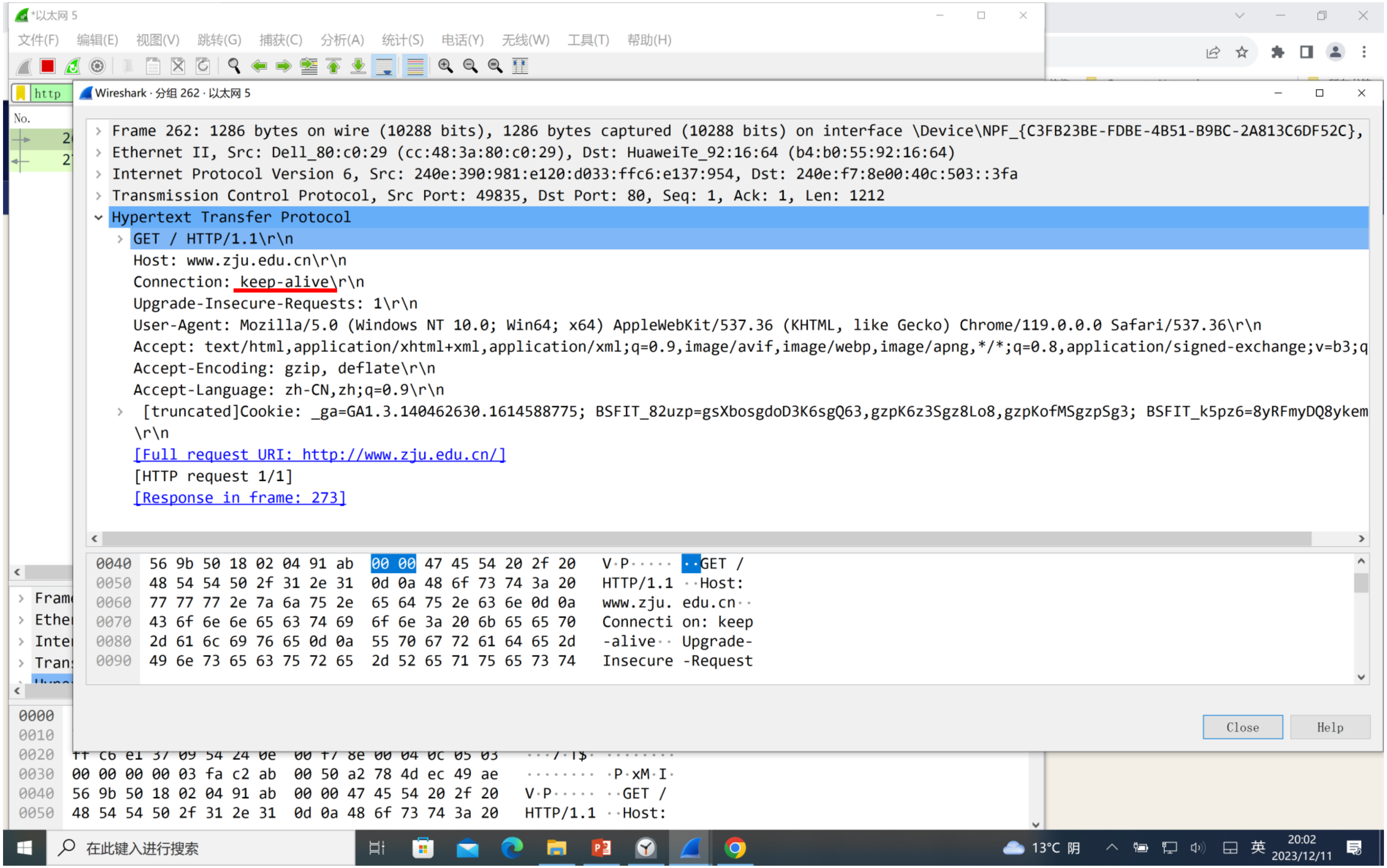
GET /somedir/page.html HTTP/1.1

2) The subsequent lines are called **header lines**

♦ Host: www.someschool.edu (specifies the host on which the object resides)

♦ Connection: close (the browser is telling the server that it doesn’t want to bother with persistent connections; it wants the server to close the connection after sending the request object.)

♦ User-agent: Mozilla/4.0 (specifies the user agent, that is, the browser type that is making the request to the server.)


*以太网 5
文件(F) 编辑(E) 视图(V) 跳转(G) 捕获(C) 分析(A) 统计(S) 电话(Y) 无线(W) 工具(T) 帮助(H)
http
No.
Wireshark · 分组 262 · 以太网 5
Close
Help
在此键入进行搜索
13°C 阴
英
20:02
2023/12/11

```
> Frame 262: 1286 bytes on wire (10288 bits), 1286 bytes captured (10288 bits) on interface \Device\NPF_{C3FB23BE-FDBE-4B51-B9BC-2A813C6DF52C},
> Ethernet II, Src: Dell_80:c0:29 (cc:48:3a:80:c0:29), Dst: HuaweiTe_92:16:64 (b4:b0:55:92:16:64)
> Internet Protocol Version 6, Src: 240e:390:981:e120:d033:ffc6:e137:954, Dst: 240e:f7:8e00:40c:503::3fa
> Transmission Control Protocol, Src Port: 49835, Dst Port: 80, Seq: 1, Ack: 1, Len: 1212
v Hypertext Transfer Protocol
  > GET / HTTP/1.1\r\n
    Host: www.zju.edu.cn\r\n
    Connection: keep-alive\r\n
    Upgrade-Insecure-Requests: 1\r\n
    User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/119.0.0.0 Safari/537.36\r\n
    Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q
    Accept-Encoding: gzip, deflate\r\n
    Accept-Language: zh-CN,zh;q=0.9\r\n
  >  [truncated]Cookie: _ga=GA1.3.140462630.1614588775; BSFIT_82uzp=gsXbosgdoD3K6sgQ63,gzpK6z3Sgz8Lo8,gzpKofMSgzpSg3; BSFIT_k5pz6=8yRFmyDQ8ykem
    \r\n
    [Full request URI: http://www.zju.edu.cn/]
    [HTTP request 1/1]
    [Response in frame: 273]
```

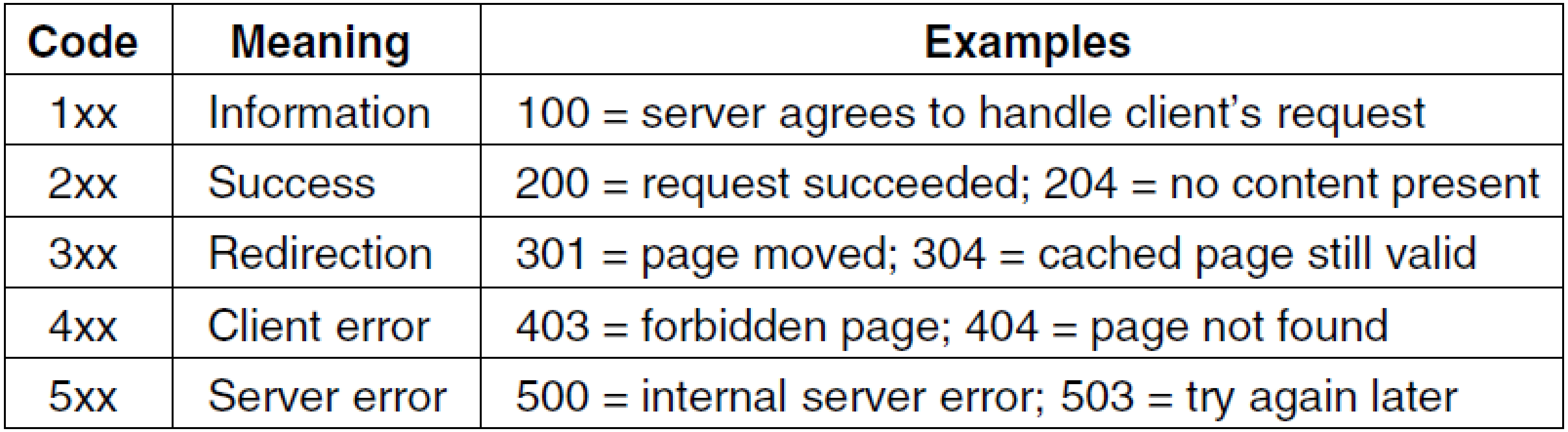
```
0040  56 9b 50 18 02 04 91 ab  00 00 47 45 54 20 2f 20   V·P·····  ··GET /
0050  48 54 54 50 2f 31 2e 31  0d 0a 48 6f 73 74 3a 20   HTTP/1.1  ··Host:
0060  77 77 77 2e 7a 6a 75 2e  65 64 75 2e 63 6e 0d 0a   www.zju.  edu.cn··
0070  43 6f 6e 6e 65 63 74 69  6f 6e 3a 20 6b 65 65 70   Connecti  on: keep
0080  2d 61 6c 69 76 65 0d 0a  55 70 67 72 61 64 65 2d   -alive··  Upgrade-
0090  49 6e 73 65 63 75 72 65  2d 52 65 71 75 65 73 74   Insecure  -Request
```

```
0000
0010
0020  ff c6 e1 37 09 54 24 0e  00 f7 8e 00 04 0c 05 03   ···7·T$·  ········
0030  00 00 00 00 03 fa c2 ab  00 50 a2 78 4d ec 49 ae   ········  ·P·xM·I·
0040  56 9b 50 18 02 04 91 ab  00 00 47 45 54 20 2f 20   V·P·····  ··GET /
0050  48 54 54 50 2f 31 2e 31  0d 0a 48 6f 73 74 3a 20   HTTP/1.1  ··Host:
```

keep-alive表示用persistent connections。

HTTP Message Format: Response

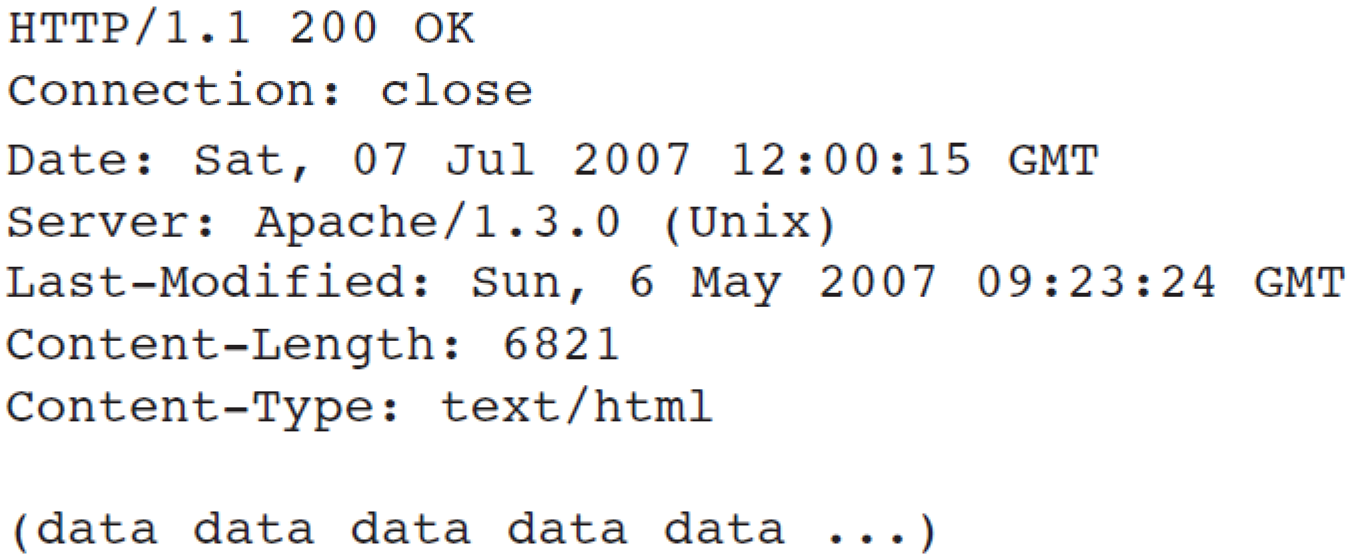
- Each request gets a **response** consisting of **a status line**, and possibly additional information.
- The status line contains a three-digit status code telling whether the request was satisfied and, if not, why not. The 1st digit is used to divide the responses into **five** major groups



Code	Meaning	Examples
1xx	Information	100 = server agrees to handle client's request
2xx	Success	200 = request succeeded; 204 = no content present
3xx	Redirection	301 = page moved; 304 = cached page still valid
4xx	Client error	403 = forbidden page; 404 = page not found
5xx	Server error	500 = internal server error; 503 = try again later

Figure 7-38. The status code response groups.

HTTP Response Message Example



```
HTTP/1.1 200 OK
Connection: close
Date: Sat, 07 Jul 2007 12:00:15 GMT
Server: Apache/1.3.0 (Unix)
Last-Modified: Sun, 6 May 2007 09:23:24 GMT
Content-Length: 6821
Content-Type: text/html

(data data data data data ...)
```

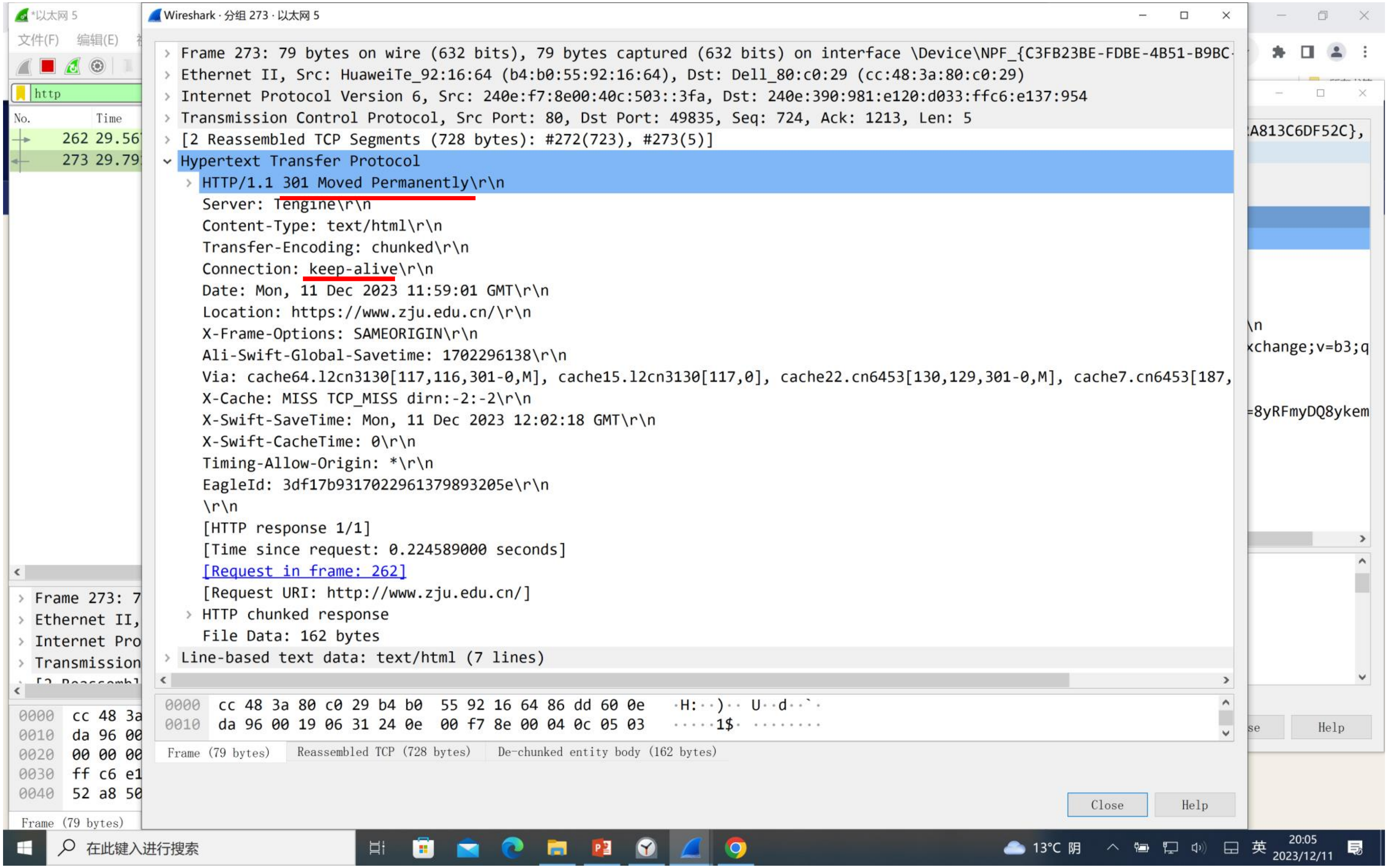
The response message has *three sections*: An initial **status line**, six **header lines** and then **the entire body**.

♦The status line has three fields: the protocol version, a status code, and a corresponding status message. (HTTP/1.1 200 OK)

♦The header lines (Connection, Date, Server, Last-Modified, Content-Length, Content-Type)

For example, the server uses the “Connection: Close” header line to tell the client that it is going to close the TCP connection after sending the message.

♦ The entire body: data



```
> Frame 273: 79 bytes on wire (632 bits), 79 bytes captured (632 bits) on interface \Device\NPF_{C3FB23BE-FDBE-4B51-B9BC-
> Ethernet II, Src: HuaweiTe_92:16:64 (b4:b0:55:92:16:64), Dst: Dell_80:c0:29 (cc:48:3a:80:c0:29)
> Internet Protocol Version 6, Src: 240e:f7:8e00:40c:503::3fa, Dst: 240e:390:981:e120:d033:ffc6:e137:954
> Transmission Control Protocol, Src Port: 80, Dst Port: 49835, Seq: 724, Ack: 1213, Len: 5
> [2 Reassembled TCP Segments (728 bytes): #272(723), #273(5)]
v Hypertext Transfer Protocol
  > HTTP/1.1 301 Moved Permanently\r\n
    Server: Tengine\r\n
    Content-Type: text/html\r\n
    Transfer-Encoding: chunked\r\n
    Connection: keep-alive\r\n
    Date: Mon, 11 Dec 2023 11:59:01 GMT\r\n
    Location: https://www.zju.edu.cn/\r\n
    X-Frame-Options: SAMEORIGIN\r\n
    Ali-Swift-Global-Savetime: 1702296138\r\n
    Via: cache64.l2cn3130[117,116,301-0,M], cache15.l2cn3130[117,0], cache22.cn6453[130,129,301-0,M], cache7.cn6453[187,
    X-Cache: MISS TCP_MISS dirn:-2:-2\r\n
    X-Swift-SaveTime: Mon, 11 Dec 2023 12:02:18 GMT\r\n
    X-Swift-CacheTime: 0\r\n
    Timing-Allow-Origin: *\r\n
    EagleId: 3df17b9317022961379893205e\r\n
    \r\n
    [HTTP response 1/1]
    [Time since request: 0.224589000 seconds]
    [Request in frame: 262]
    [Request URI: http://www.zju.edu.cn/]
  > HTTP chunked response
    File Data: 162 bytes
> Line-based text data: text/html (7 lines)
```

```
0000  cc 48 3a 80 c0 29 b4 b0  55 92 16 64 86 dd 60 0e   ·H:··)·· U··d··`·
0010  da 96 00 19 06 31 24 0e  00 f7 8e 00 04 0c 05 03   ·····1$· ········
```

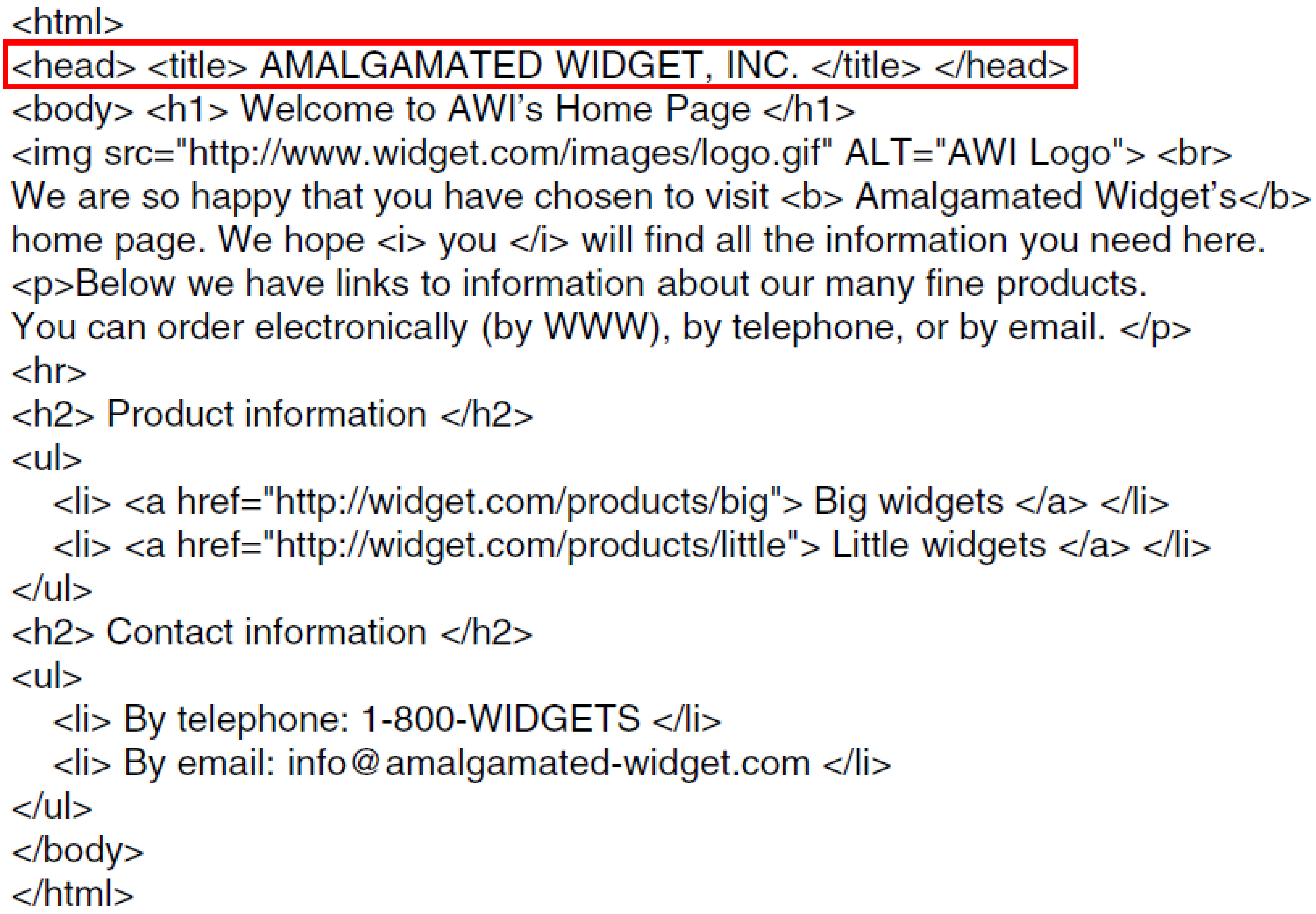
Frame (79 bytes)   Reassembled TCP (728 bytes)   De-chunked entity body (162 bytes)

301表示page moved permanently

Static Web Pages

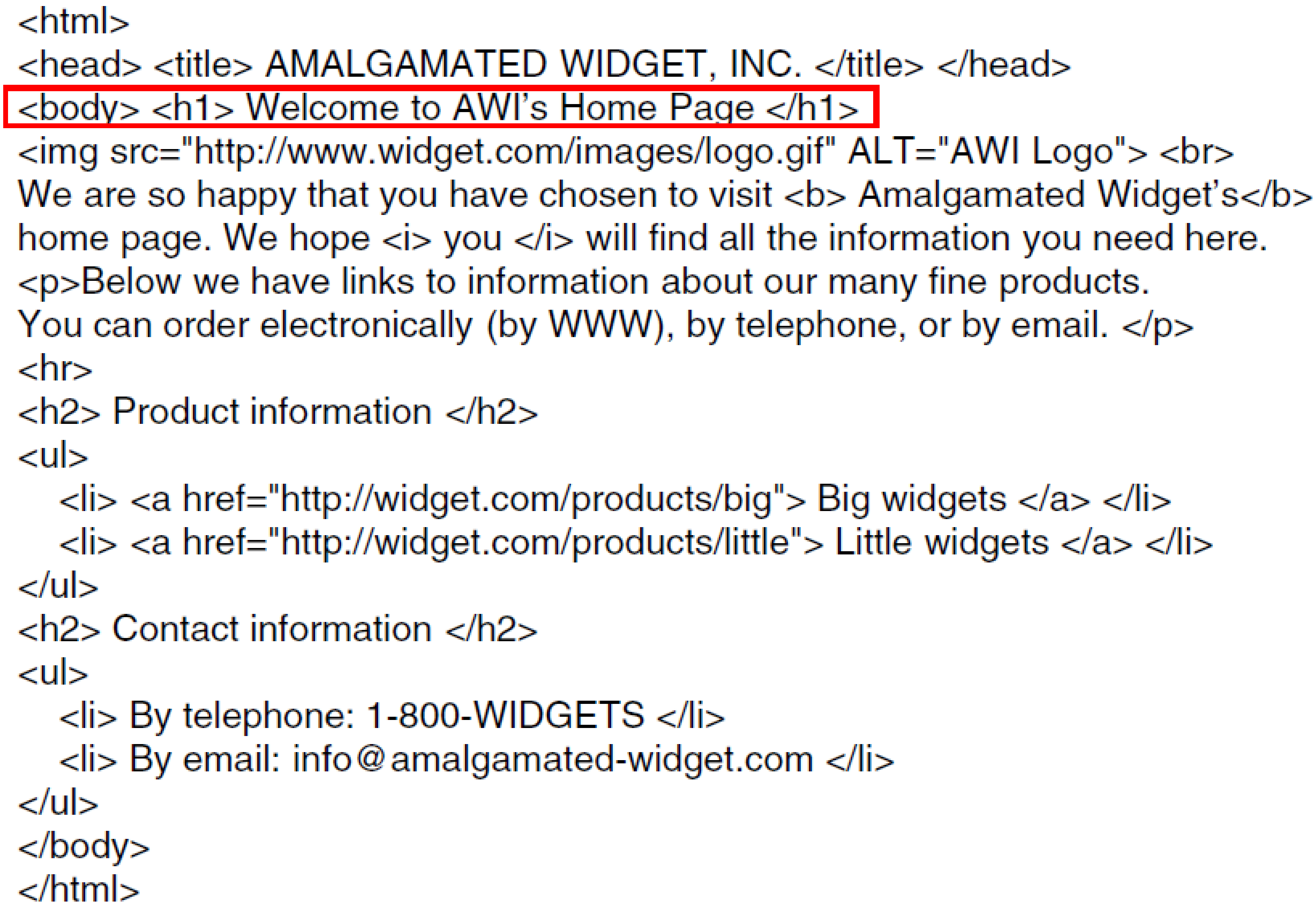
- Static web page is a file contents, e.g., image
  - A page containing a video can be a static Web page.
- **HTML** is a makeup language, or language for describing how documents are to be formatted.
  - Makeup languages contain explicit commands for formatting.
  - Other examples: LaTex and Tex
  - The key advantage of a makeup language over one with no explicit makeup is that it separates content from how it should be presented.
  - The browser simply has to understand the makeup commands and apply them to the content. It makes possible for any Web browser to reformat any Web page.

```
<html>
<head> <title> AMALGAMATED WIDGET, INC. </title> </head>
<body> <h1> Welcome to AWI’s Home Page </h1>
<img src="http://www.widget.com/images/logo.gif" ALT="AWI Logo"> <br>
We are so happy that you have chosen to visit <b> Amalgamated Widget’s</b>
home page. We hope <i> you </i> will find all the information you need here.
<p>Below we have links to information about our many fine products.
You can order electronically (by WWW), by telephone, or by email. </p>
<hr>
<h2> Product information </h2>
<ul>
   <li> <a href="http://widget.com/products/big"> Big widgets </a> </li>
   <li> <a href="http://widget.com/products/little"> Little widgets </a> </li>
</ul>
<h2> Contact information </h2>
<ul>
   <li> By telephone: 1-800-WIDGETS </li>
   <li> By email: info@amalgamated-widget.com </li>
</ul>
</body>
</html>
```



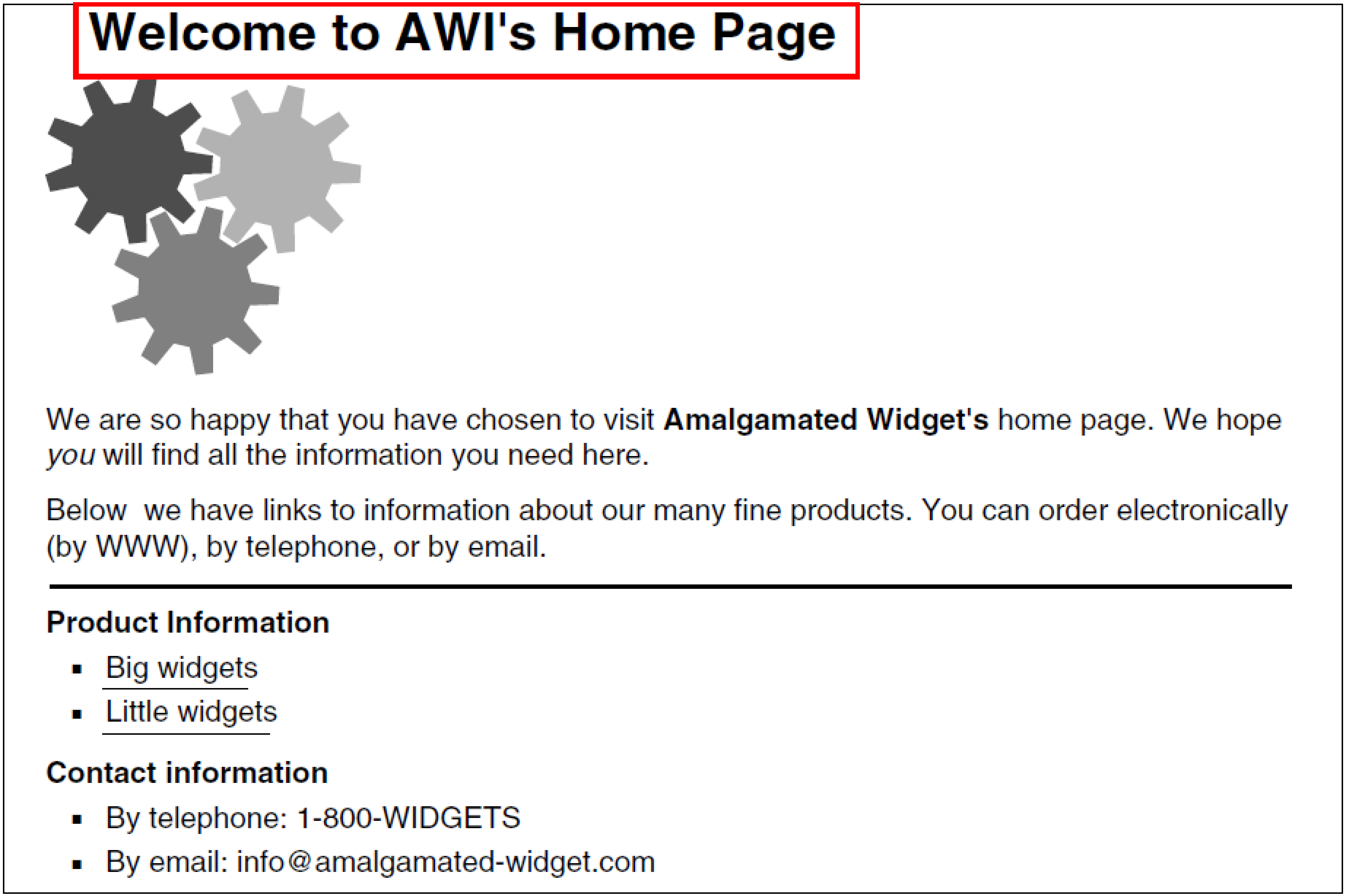
```
<html>
<head> <title> AMALGAMATED WIDGET, INC. </title> </head>
<body> <h1> Welcome to AWI's Home Page </h1>
<img src="http://www.widget.com/images/logo.gif" ALT="AWI Logo"> <br>
We are so happy that you have chosen to visit <b> Amalgamated Widget's</b>
home page. We hope <i> you </i> will find all the information you need here.
<p>Below we have links to information about our many fine products.
You can order electronically (by WWW), by telephone, or by email. </p>
<hr>
<h2> Product information </h2>
<ul>
   <li> <a href="http://widget.com/products/big"> Big widgets </a> </li>
   <li> <a href="http://widget.com/products/little"> Little widgets </a> </li>
</ul>
<h2> Contact information </h2>
<ul>
   <li> By telephone: 1-800-WIDGETS </li>
   <li> By email: info@amalgamated-widget.com </li>
</ul>
</body>
</html>
```

The main item in the head is the title, delimited by <title> and </title>. The title itself is **not** displayed on the page. Some browsers use it to label the page's window.



```
<html>
<head> <title> AMALGAMATED WIDGET, INC. </title> </head>
<body> <h1> Welcome to AWI's Home Page </h1>
<img src="http://www.widget.com/images/logo.gif" ALT="AWI Logo"> <br>
We are so happy that you have chosen to visit <b> Amalgamated Widget's</b>
home page. We hope <i> you </i> will find all the information you need here.
<p>Below we have links to information about our many fine products.
You can order electronically (by WWW), by telephone, or by email. </p>
<hr>
<h2> Product information </h2>
<ul>
   <li> <a href="http://widget.com/products/big"> Big widgets </a> </li>
   <li> <a href="http://widget.com/products/little"> Little widgets </a> </li>
</ul>
<h2> Contact information </h2>
<ul>
   <li> By telephone: 1-800-WIDGETS </li>
   <li> By email: info@amalgamated-widget.com </li>
</ul>
</body>
</html>
```

Each heading is generated by an <h*n*> tag, where *n* is a digit in the range 1 to 6. Thus, <h1> is the most important heading; <h6> is the least important heading.



Welcome to AWI's Home Page

We are so happy that you have chosen to visit **Amalgamated Widget's** home page. We hope *you* will find all the information you need here.

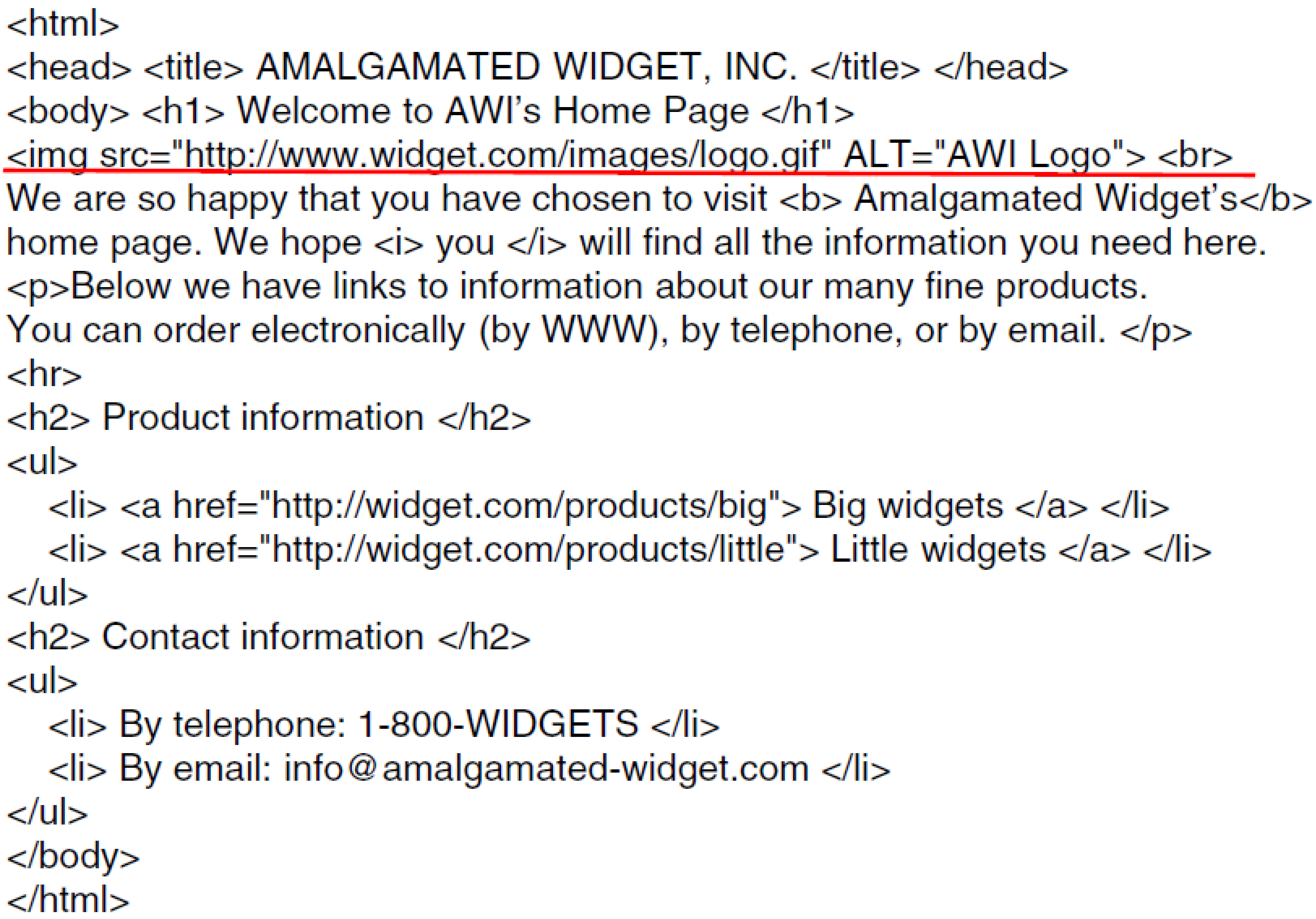
Below we have links to information about our many fine products. You can order electronically (by WWW), by telephone, or by email.

Product Information

- Big widgets
- Little widgets

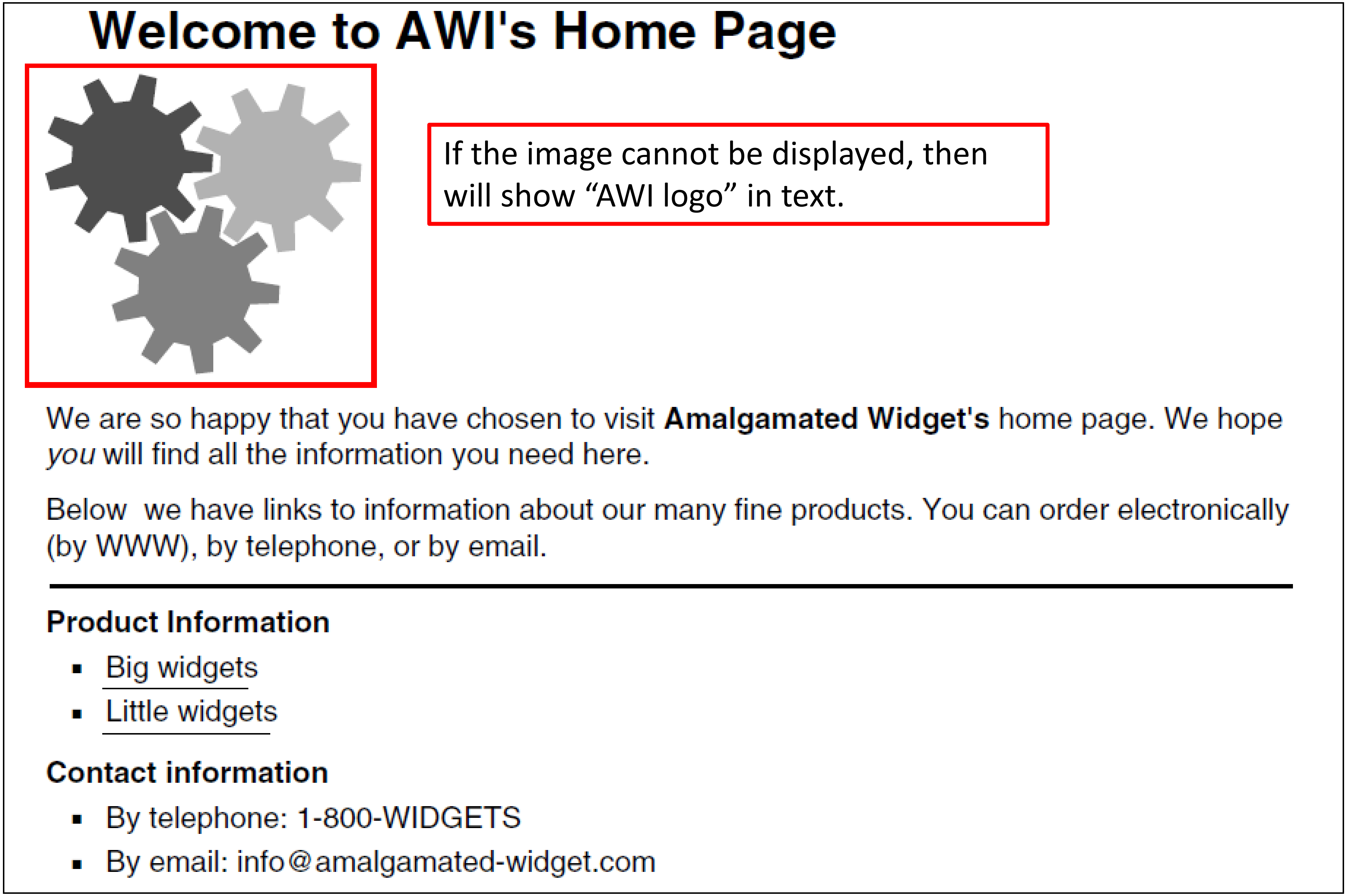
Contact information

- By telephone: 1-800-WIDGETS
- By email: info@amalgamated-widget.com



```
<html>
<head> <title> AMALGAMATED WIDGET, INC. </title> </head>
<body> <h1> Welcome to AWI’s Home Page </h1>
<img src="http://www.widget.com/images/logo.gif" ALT="AWI Logo"> <br>
We are so happy that you have chosen to visit <b> Amalgamated Widget’s</b>
home page. We hope <i> you </i> will find all the information you need here.
<p>Below we have links to information about our many fine products.
You can order electronically (by WWW), by telephone, or by email. </p>
<hr>
<h2> Product information </h2>
<ul>
   <li> <a href="http://widget.com/products/big"> Big widgets </a> </li>
   <li> <a href="http://widget.com/products/little"> Little widgets </a> </li>
</ul>
<h2> Contact information </h2>
<ul>
   <li> By telephone: 1-800-WIDGETS </li>
   <li> By email: info@amalgamated-widget.com </li>
</ul>
</body>
</html>
```

The <img> tag is used for including an image inline with the text. It has two attributes: **src** and **alt**. **src** gives the URL for the images. **Alt** gives alternative text to use if the image cannot be displayed. Here the <br> tag forces the browser to break and start a new line.



Welcome to AWI's Home Page

If the image cannot be displayed, then will show “AWI logo” in text.

We are so happy that you have chosen to visit **Amalgamated Widget's** home page. We hope *you* will find all the information you need here.

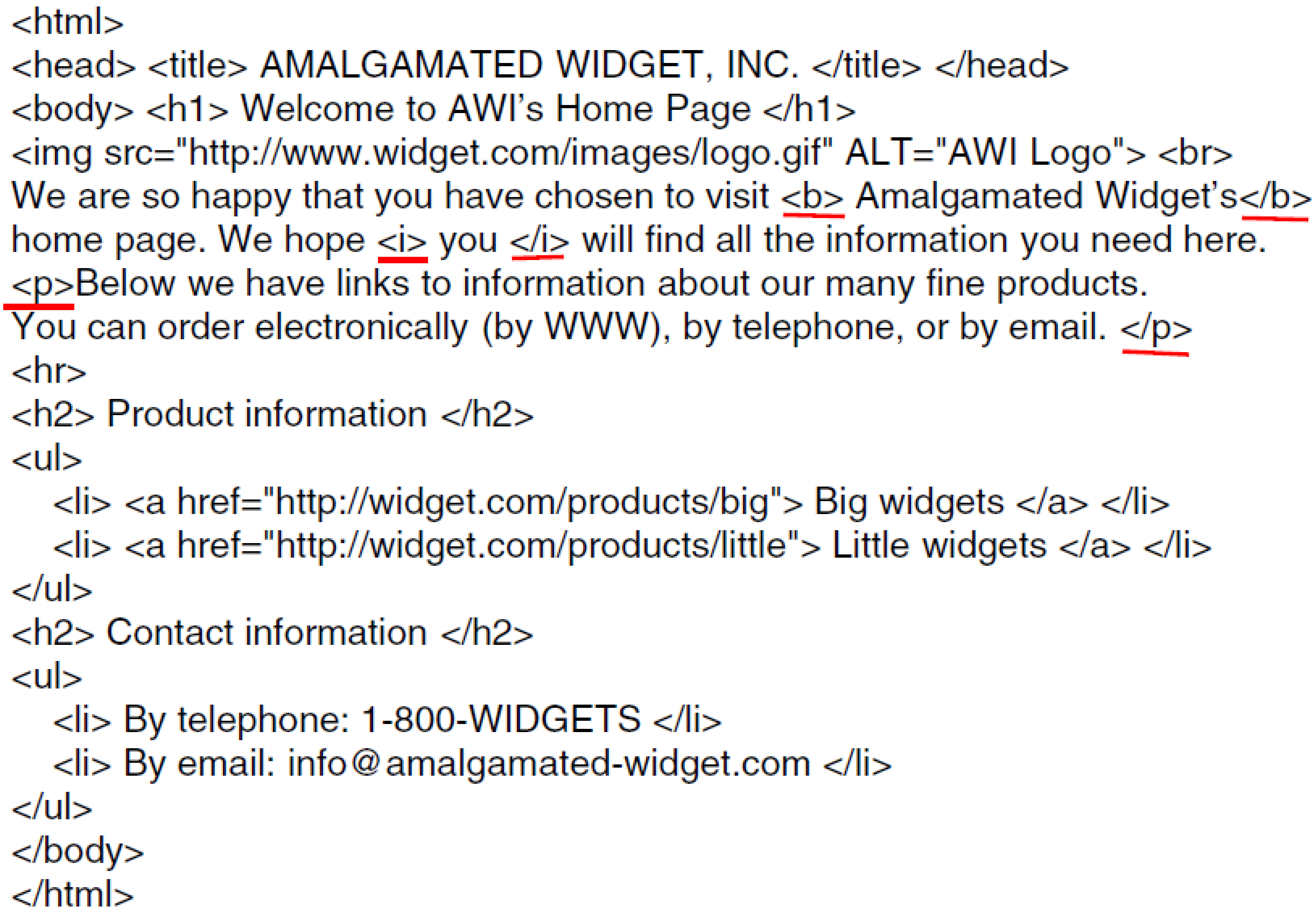
Below we have links to information about our many fine products. You can order electronically (by WWW), by telephone, or by email.

Product Information

- Big widgets
- Little widgets

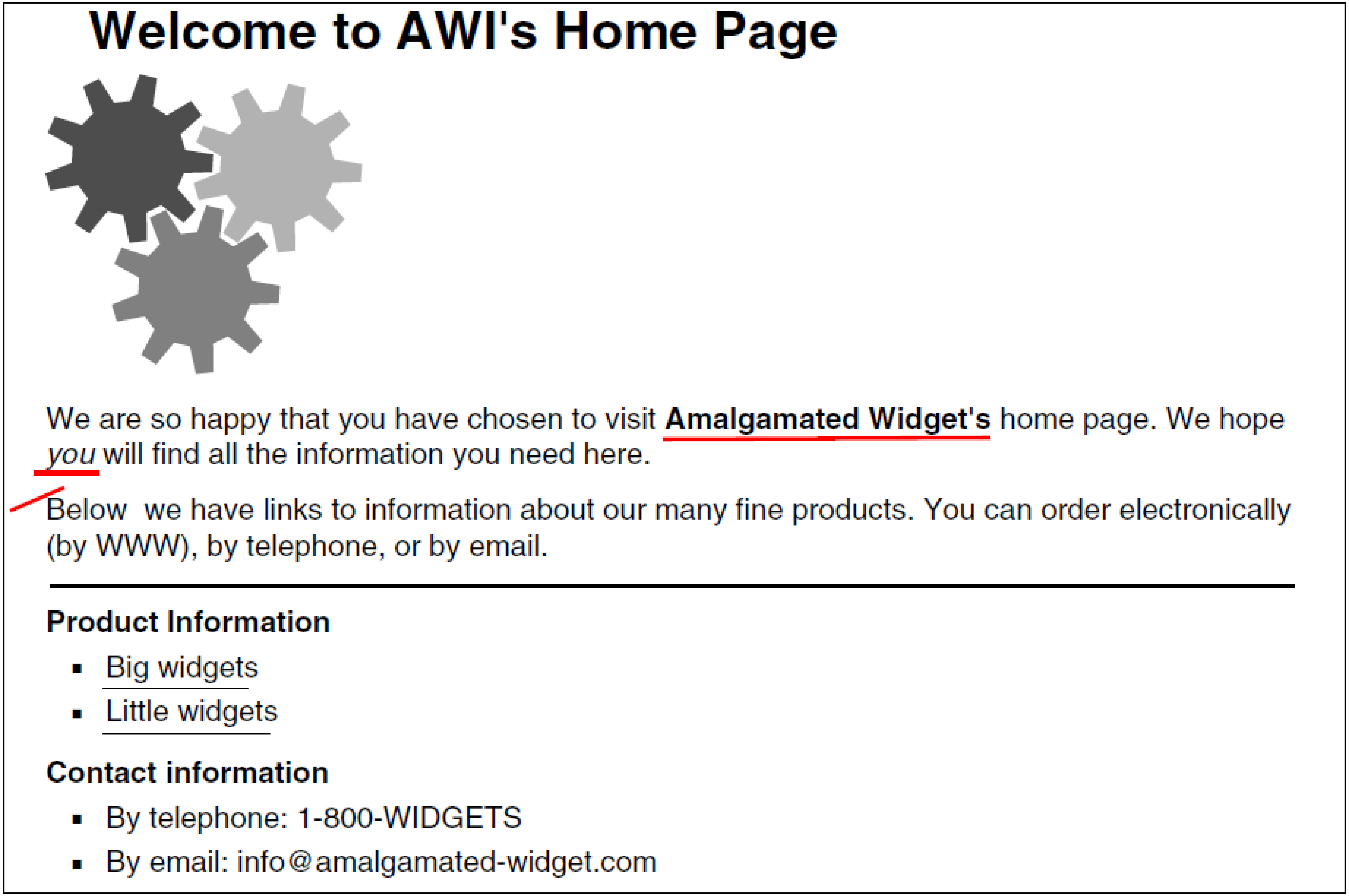
Contact information

- By telephone: 1-800-WIDGETS
- By email: info@amalgamated-widget.com



```
<html>
<head> <title> AMALGAMATED WIDGET, INC. </title> </head>
<body> <h1> Welcome to AWI’s Home Page </h1>
<img src="http://www.widget.com/images/logo.gif" ALT="AWI Logo"> <br>
We are so happy that you have chosen to visit <b> Amalgamated Widget’s</b>
home page. We hope <i> you </i> will find all the information you need here.
<p>Below we have links to information about our many fine products.
You can order electronically (by WWW), by telephone, or by email. </p>
<hr>
<h2> Product information </h2>
<ul>
   <li> <a href="http://widget.com/products/big"> Big widgets </a> </li>
   <li> <a href="http://widget.com/products/little"> Little widgets </a> </li>
</ul>
<h2> Contact information </h2>
<ul>
   <li> By telephone: 1-800-WIDGETS </li>
   <li> By email: info@amalgamated-widget.com </li>
</ul>
</body>
</html>
```

The tags <b> and </b> are used to enter boldface mode. And <i> and </i> are for italics
The tag <p> starts a paragraph, </p> marks the end of the paragraph.



Welcome to AWI's Home Page

We are so happy that you have chosen to visit **Amalgamated Widget's** home page. We hope *you* will find all the information you need here.

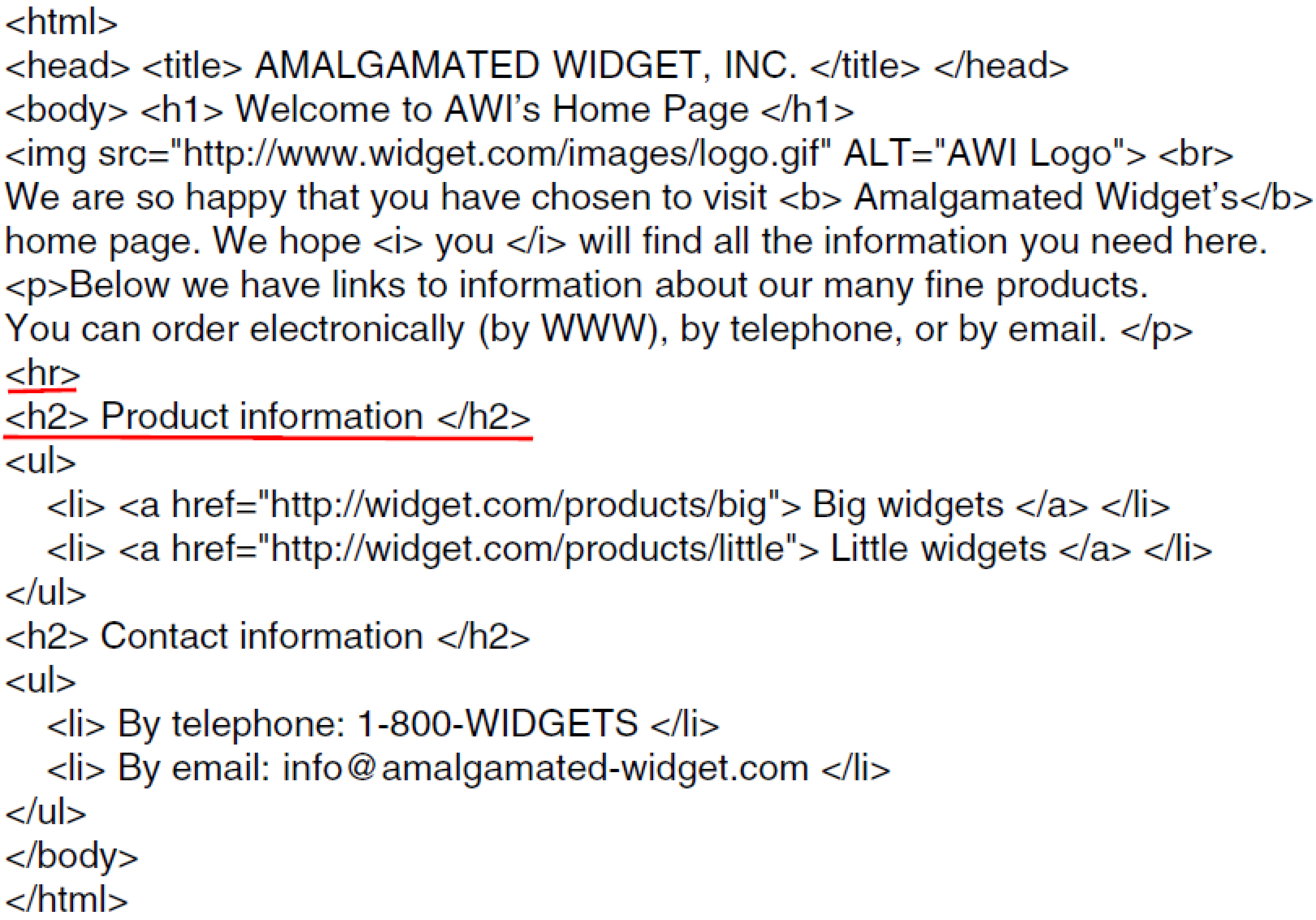
Below we have links to information about our many fine products. You can order electronically (by WWW), by telephone, or by email.

Product Information

- Big widgets
- Little widgets

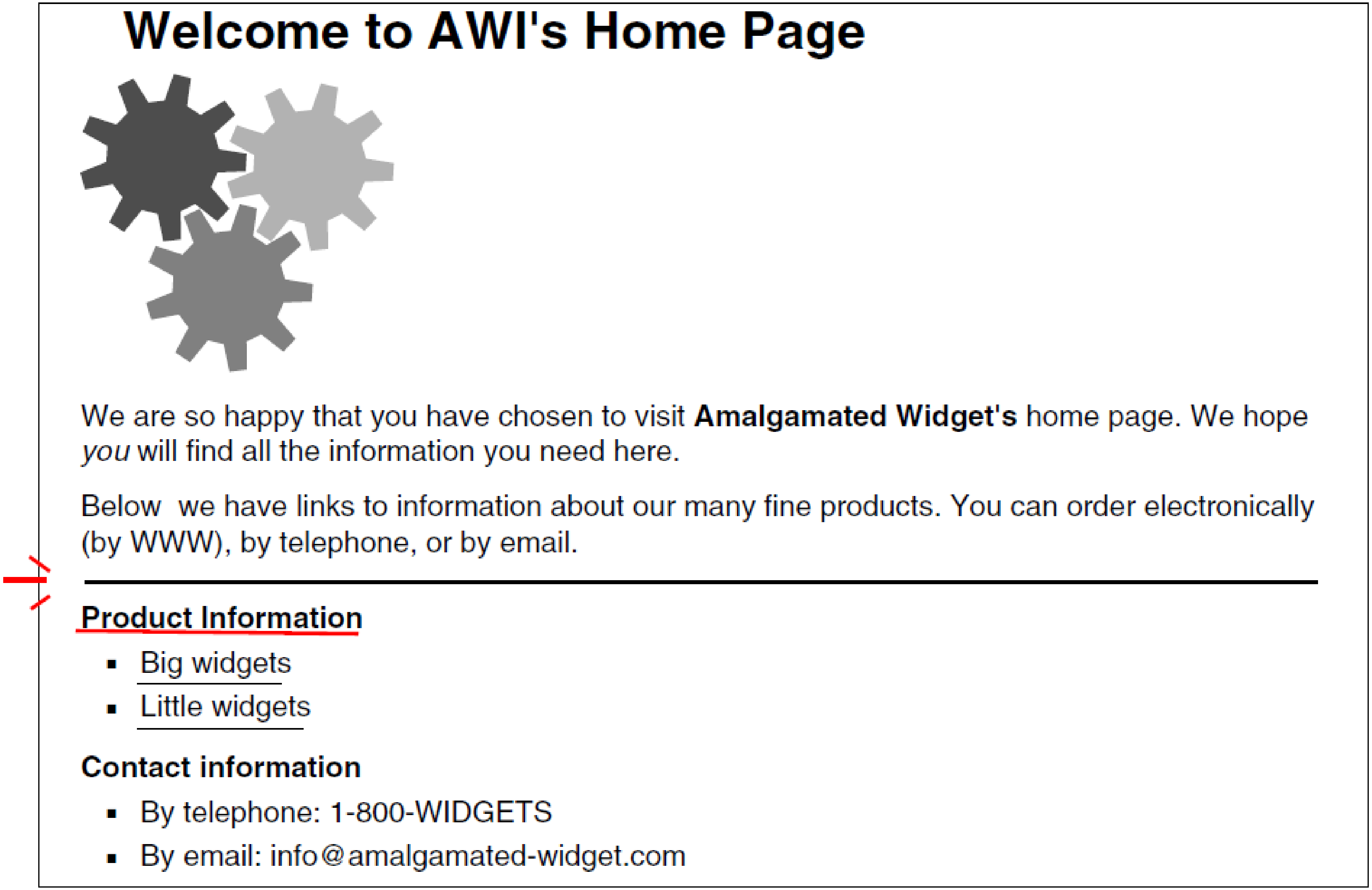
Contact information

- By telephone: 1-800-WIDGETS
- By email: info@amalgamated-widget.com



```
<html>
<head> <title> AMALGAMATED WIDGET, INC. </title> </head>
<body> <h1> Welcome to AWI’s Home Page </h1>
<img src="http://www.widget.com/images/logo.gif" ALT="AWI Logo"> <br>
We are so happy that you have chosen to visit <b> Amalgamated Widget’s</b>
home page. We hope <i> you </i> will find all the information you need here.
<p>Below we have links to information about our many fine products.
You can order electronically (by WWW), by telephone, or by email. </p>
<hr>
<h2> Product information </h2>
<ul>
   <li> <a href="http://widget.com/products/big"> Big widgets </a> </li>
   <li> <a href="http://widget.com/products/little"> Little widgets </a> </li>
</ul>
<h2> Contact information </h2>
<ul>
   <li> By telephone: 1-800-WIDGETS </li>
   <li> By email: info@amalgamated-widget.com </li>
</ul>
</body>
</html>
```

The <hr> tag forces a break and draws a horizontal line across the display.
<h2> and </h2> denotes the 2nd most important heading.



Welcome to AWI's Home Page

We are so happy that you have chosen to visit **Amalgamated Widget's** home page. We hope *you* will find all the information you need here.

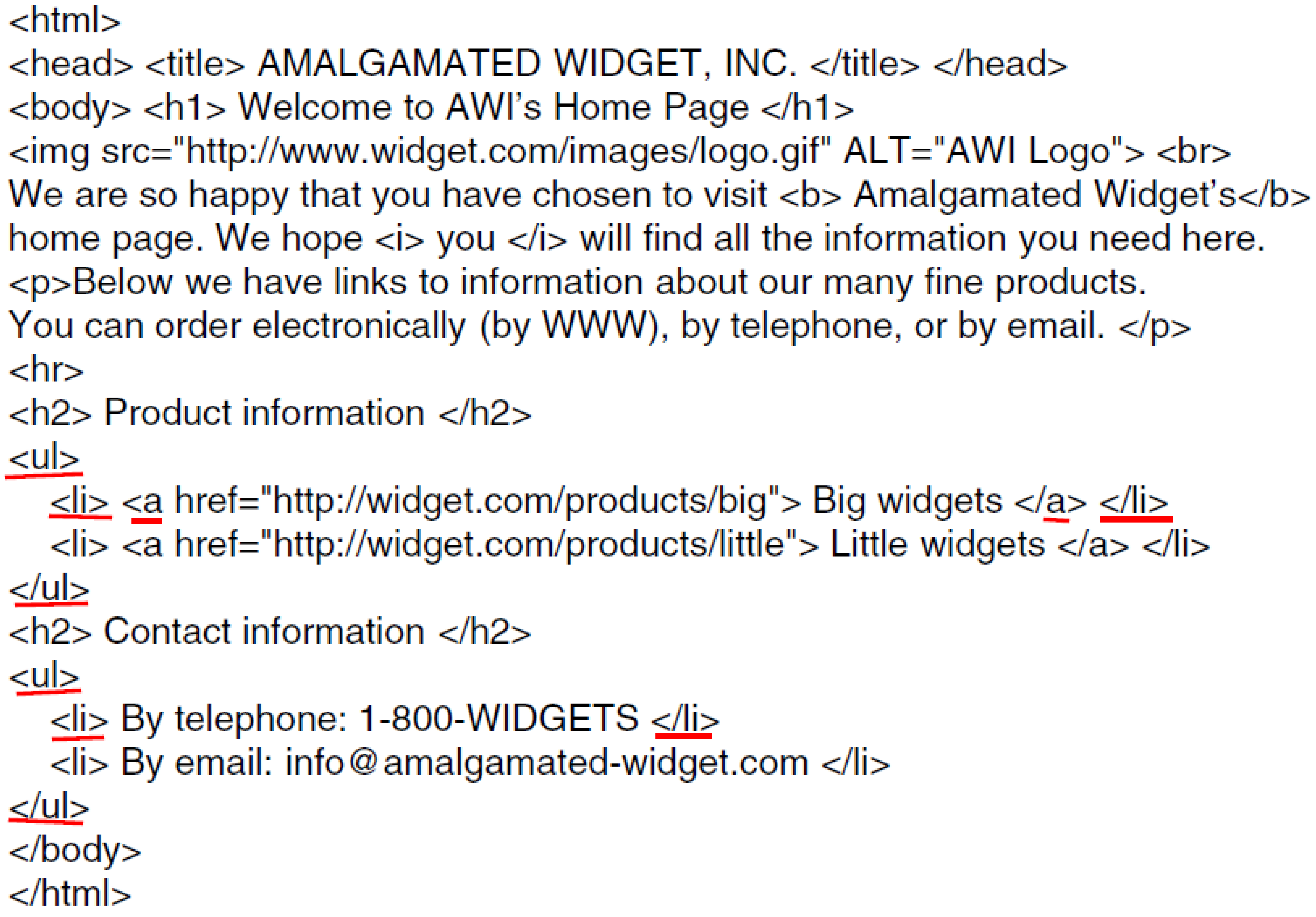
Below we have links to information about our many fine products. You can order electronically (by WWW), by telephone, or by email.

Product Information

- Big widgets
- Little widgets

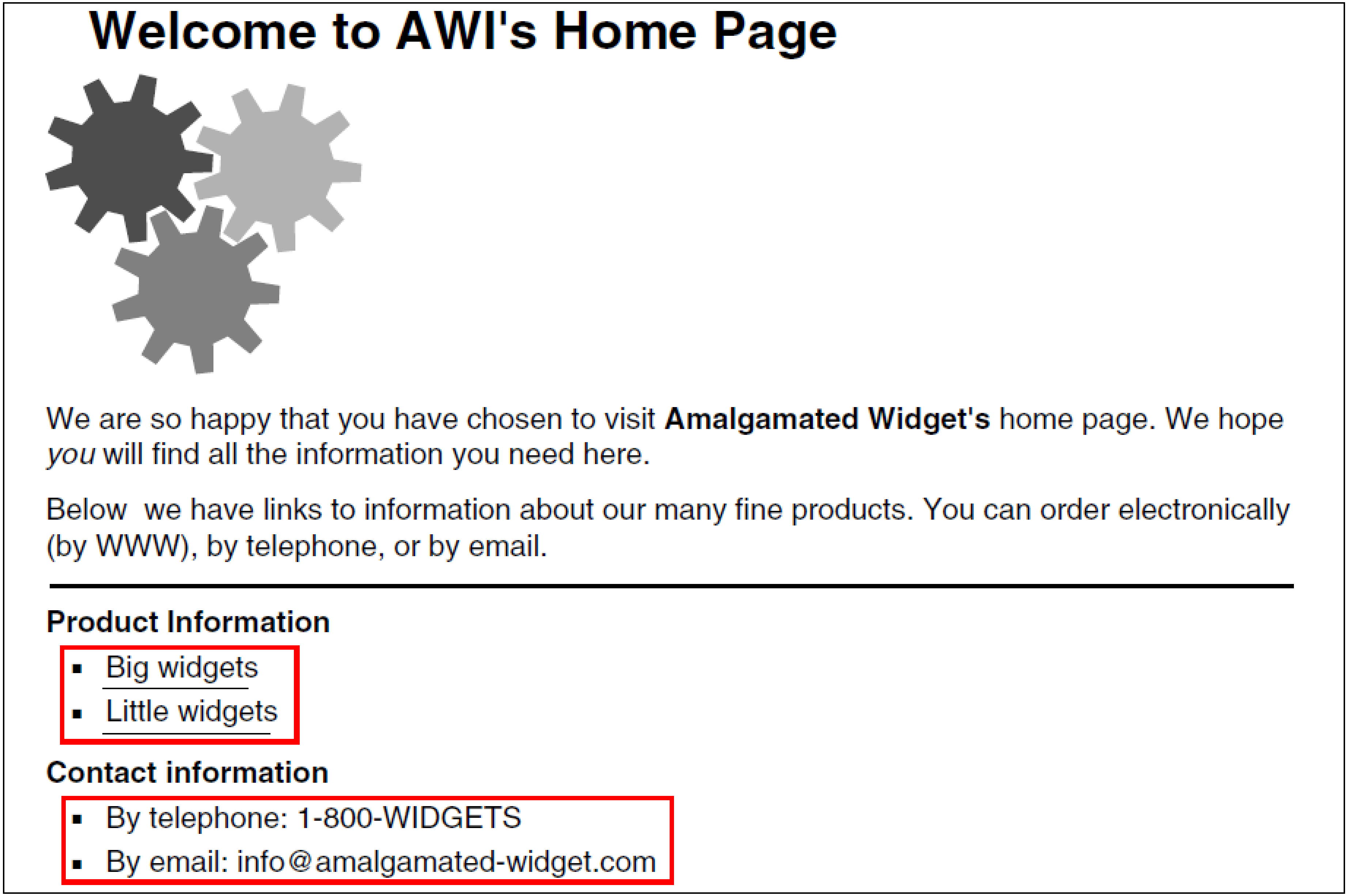
Contact information

- By telephone: 1-800-WIDGETS
- By email: info@amalgamated-widget.com



```
<html>
<head> <title> AMALGAMATED WIDGET, INC. </title> </head>
<body> <h1> Welcome to AWI's Home Page </h1>
<img src="http://www.widget.com/images/logo.gif" ALT="AWI Logo"> <br>
We are so happy that you have chosen to visit <b> Amalgamated Widget's</b>
home page. We hope <i> you </i> will find all the information you need here.
<p>Below we have links to information about our many fine products.
You can order electronically (by WWW), by telephone, or by email. </p>
<hr>
<h2> Product information </h2>
<ul>
    <li> <a href="http://widget.com/products/big"> Big widgets </a> </li>
    <li> <a href="http://widget.com/products/little"> Little widgets </a> </li>
</ul>
<h2> Contact information </h2>
<ul>
    <li> By telephone: 1-800-WIDGETS </li>
    <li> By email: info@amalgamated-widget.com </li>
</ul>
</body>
</html>
```

The tag <a> and </a> are used for hyperlinks
The tags <ul> and </ul>, <li> and </li> are used to mark the start of items. (li – list)
The tag <ol> and </ol> are used to start an ordered list.



Welcome to AWI's Home Page

We are so happy that you have chosen to visit **Amalgamated Widget's** home page. We hope *you* will find all the information you need here.

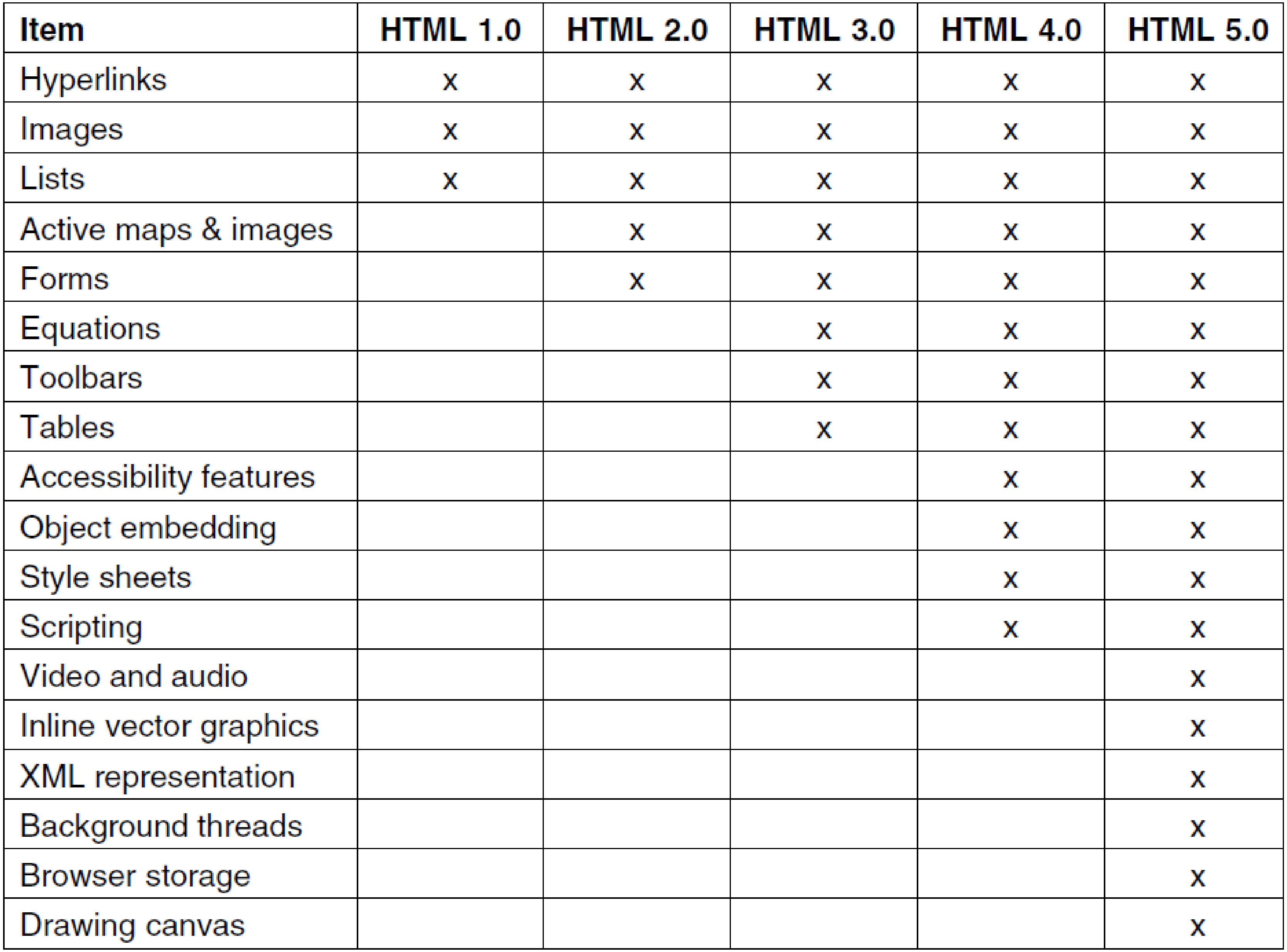
Below we have links to information about our many fine products. You can order electronically (by WWW), by telephone, or by email.

Product Information

- Big widgets
- Little widgets

Contact information

- By telephone: 1-800-WIDGETS
- By email: info@amalgamated-widget.com



Item	HTML 1.0	HTML 2.0	HTML 3.0	HTML 4.0	HTML 5.0
Hyperlinks	x	x	x	x	x
Images	x	x	x	x	x
Lists	x	x	x	x	x
Active maps & images		x	x	x	x
Forms		x	x	x	x
Equations			x	x	x
Toolbars			x	x	x
Tables			x	x	x
Accessibility features				x	x
Object embedding				x	x
Style sheets				x	x
Scripting				x	x
Video and audio					x
Inline vector graphics					x
XML representation					x
Background threads					x
Browser storage					x
Drawing canvas					x

Figure 7-24. Some differences between HTML versions.

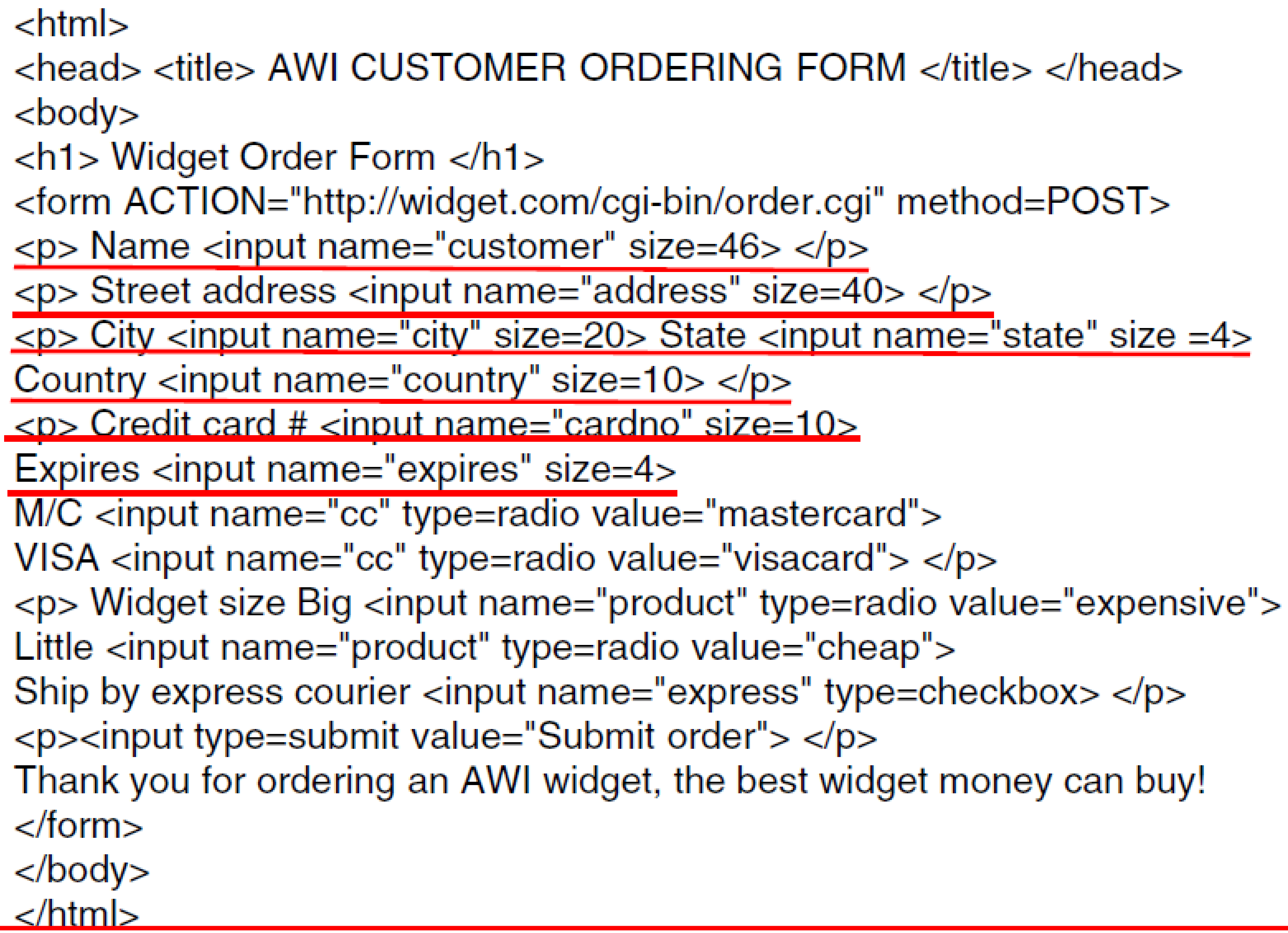
HTML — Input and Forms (I)

- HTML 1.0 was basically one-way
  - – Users could fetch pages from information providers, but it was difficult to send information back the other way.
- It quickly became apparent that there was a need for *two-way traffic* to allow orders for products to be placed via Web pages, registration cards to be filled out online.
- Sending input from the user to the server (via the browser) requires two kinds of support.
  - – The 1st requirement is that HTTP be able to carry data in that direction
    - The **POST** method
  - – The 2nd requirement is to be able to present user interface elements that gather and package up the input
    - **Forms** were included with this functionality in HTML 2.0.

HTML — Input and Forms (II)

- Forms contain **boxes** or **buttons** that allow users to fill in information or make choices and then send the information back to the page’s owner.

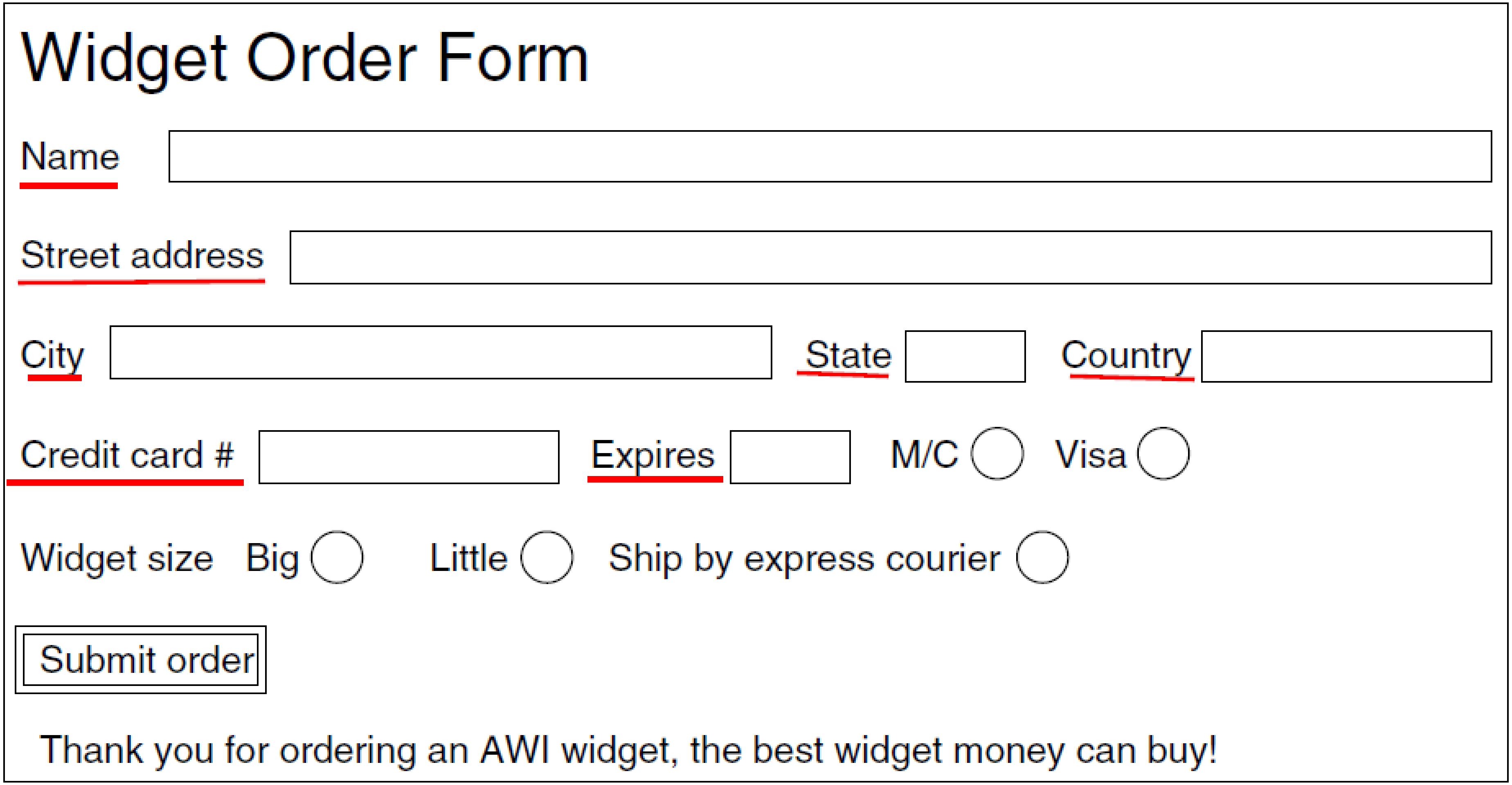
```
<html>
<head> <title> AWI CUSTOMER ORDERING FORM </title> </head>
<body>
<h1> Widget Order Form </h1>
<form ACTION="http://widget.com/cgi-bin/order.cgi" method=POST>
<p> Name <input name="customer" size=46> </p>
<p> Street address <input name="address" size=40> </p>
<p> City <input name="city" size=20> State <input name="state" size =4>
Country <input name="country" size=10> </p>
<p> Credit card # <input name="cardno" size=10>
Expires <input name="expires" size=4>
M/C <input name="cc" type=radio value="mastercard">
VISA <input name="cc" type=radio value="visacard"> </p>
<p> Widget size Big <input name="product" type=radio value="expensive">
Little <input name="product" type=radio value="cheap">
Ship by express courier <input name="express" type=checkbox> </p>
<p><input type=submit value="Submit order"> </p>
Thank you for ordering an AWI widget, the best widget money can buy!
</form>
</body>
</html>
```



```
<html>
<head> <title> AWI CUSTOMER ORDERING FORM </title> </head>
<body>
<h1> Widget Order Form </h1>
<form ACTION="http://widget.com/cgi-bin/order.cgi" method=POST>
<p> Name <input name="customer" size=46> </p>
<p> Street address <input name="address" size=40> </p>
<p> City <input name="city" size=20> State <input name="state" size =4>
Country <input name="country" size=10> </p>
<p> Credit card # <input name="cardno" size=10>
Expires <input name="expires" size=4>
M/C <input name="cc" type=radio value="mastercard">
VISA <input name="cc" type=radio value="visacard"> </p>
<p> Widget size Big <input name="product" type=radio value="expensive">
Little <input name="product" type=radio value="cheap">
Ship by express courier <input name="express" type=checkbox> </p>
<p><input type=submit value="Submit order"> </p>
Thank you for ordering an AWI widget, the best widget money can buy!
</form>
</body>
</html>
```

Three kinds of input boxes are used in this form, each of which uses the <input> tag.

The first kind of input box is **a text box**.



Widget Order Form

Name

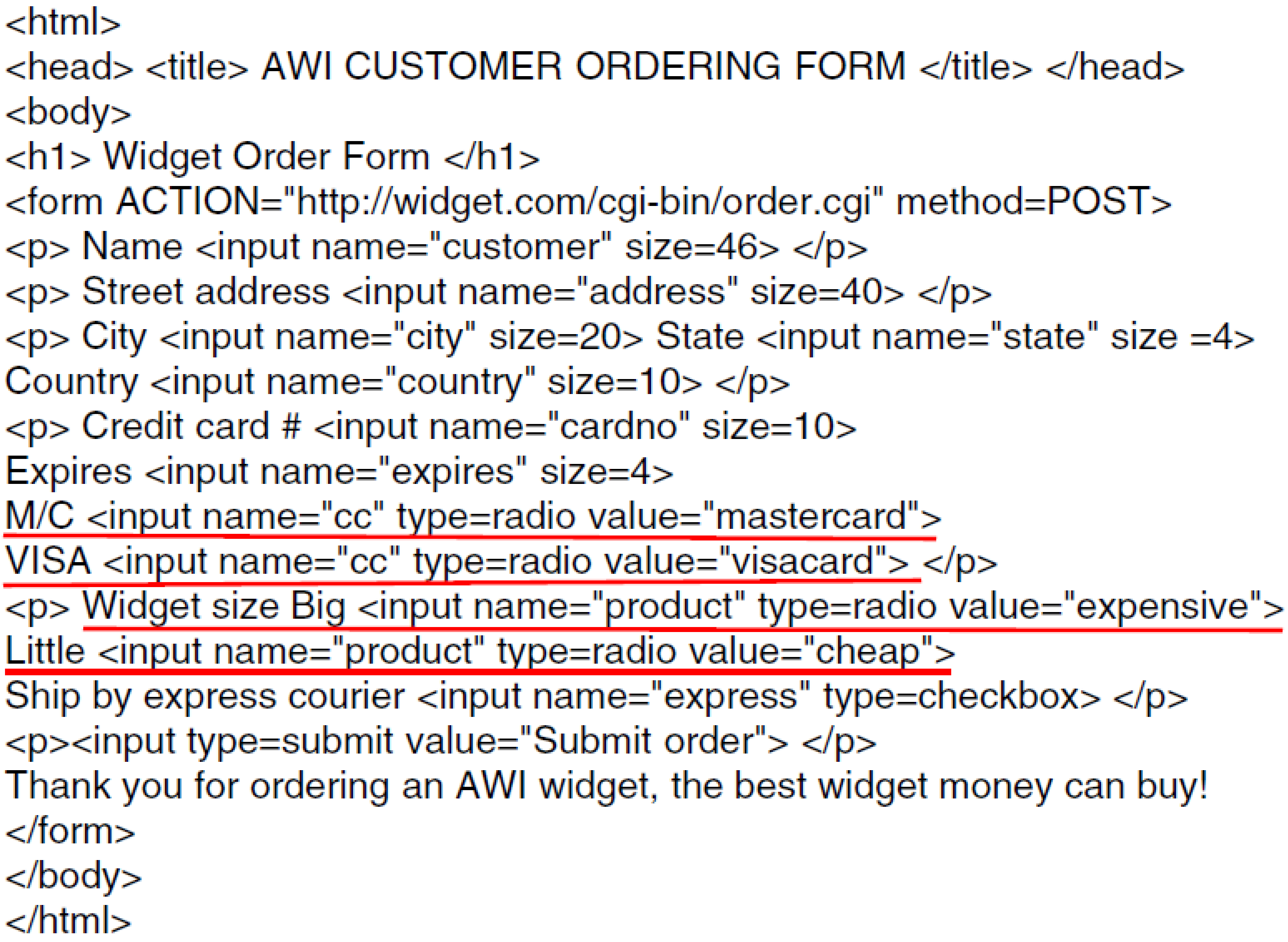
Street address

City State Country

Credit card # Expires M/C ○ Visa ○

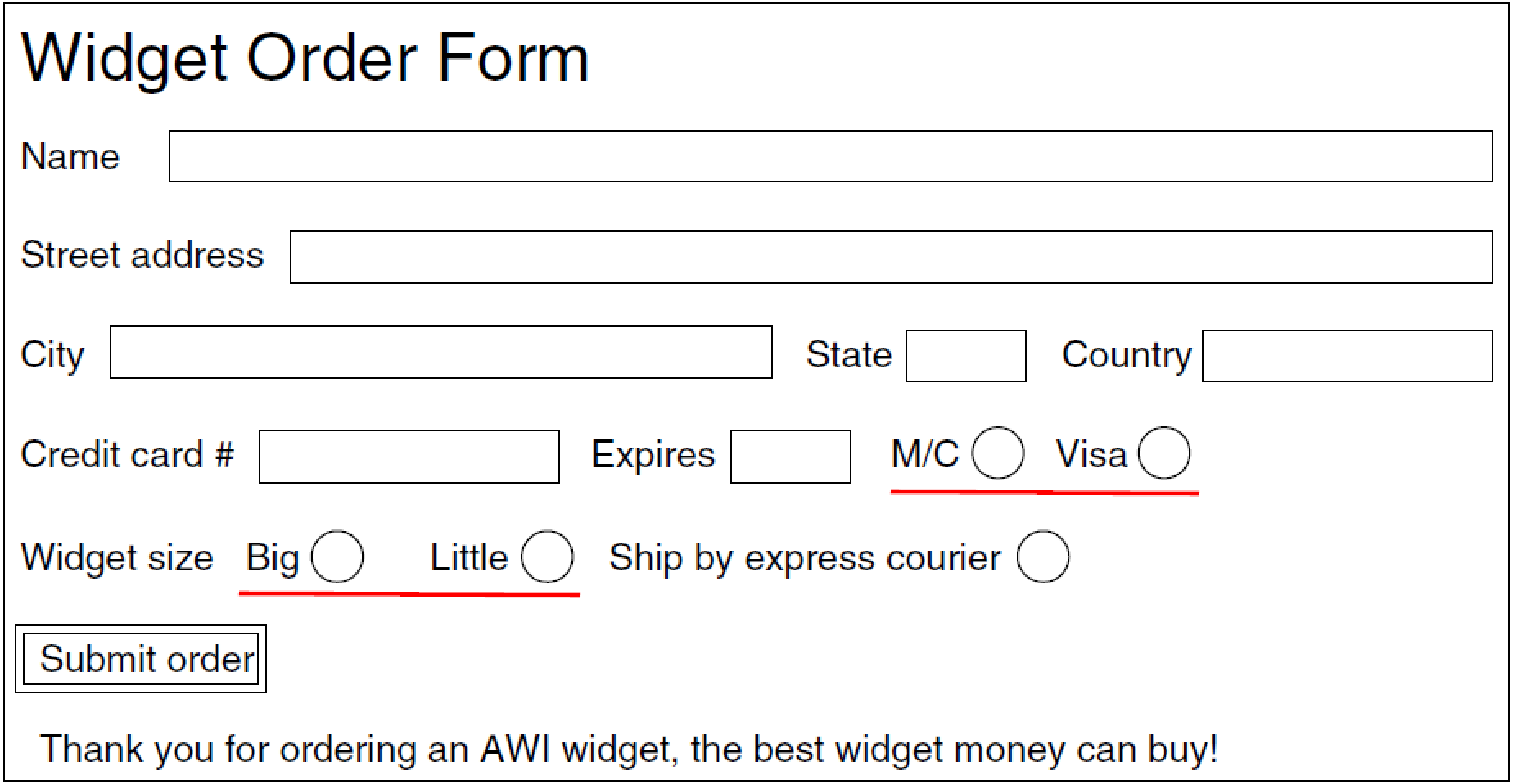
Widget size Big ○ Little ○ Ship by express courier ○

Thank you for ordering an AWI widget, the best widget money can buy!



```
<html>
<head> <title> AWI CUSTOMER ORDERING FORM </title> </head>
<body>
<h1> Widget Order Form </h1>
<form ACTION="http://widget.com/cgi-bin/order.cgi" method=POST>
<p> Name <input name="customer" size=46> </p>
<p> Street address <input name="address" size=40> </p>
<p> City <input name="city" size=20> State <input name="state" size =4>
Country <input name="country" size=10> </p>
<p> Credit card # <input name="cardno" size=10>
Expires <input name="expires" size=4>
M/C <input name="cc" type=radio value="mastercard">
VISA <input name="cc" type=radio value="visacard"> </p>
<p> Widget size Big <input name="product" type=radio value="expensive">
Little <input name="product" type=radio value="cheap">
Ship by express courier <input name="express" type=checkbox> </p>
<p><input type=submit value="Submit order"> </p>
Thank you for ordering an AWI widget, the best widget money can buy!
</form>
</body>
</html>
```

Radio buttons: these are used when a choice must be made among two or more alternatives. Clicking on one button turns off all the other ones in the same group.



Widget Order Form

Name

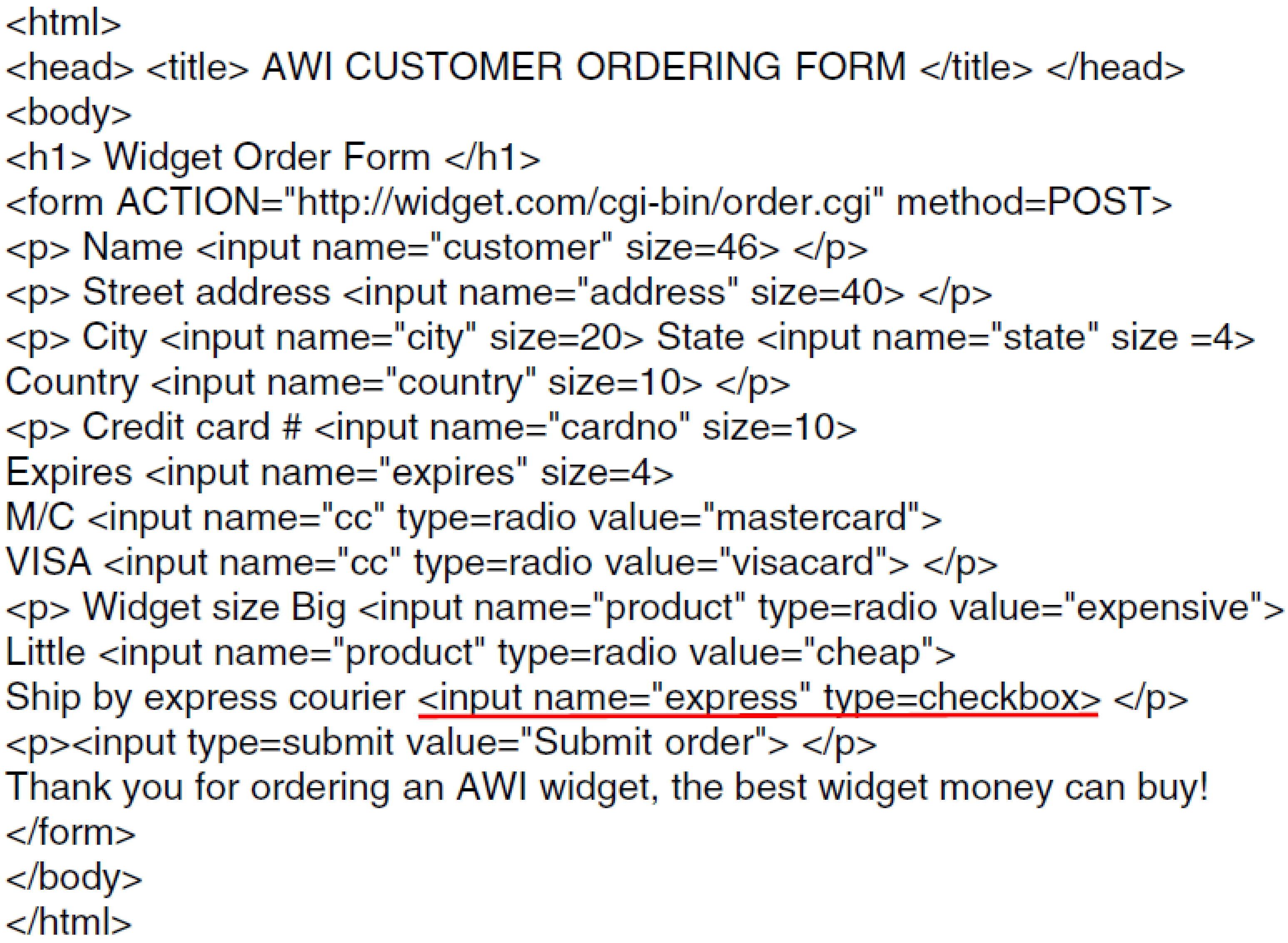
Street address

City State Country

Credit card # Expires M/C ○ Visa ○

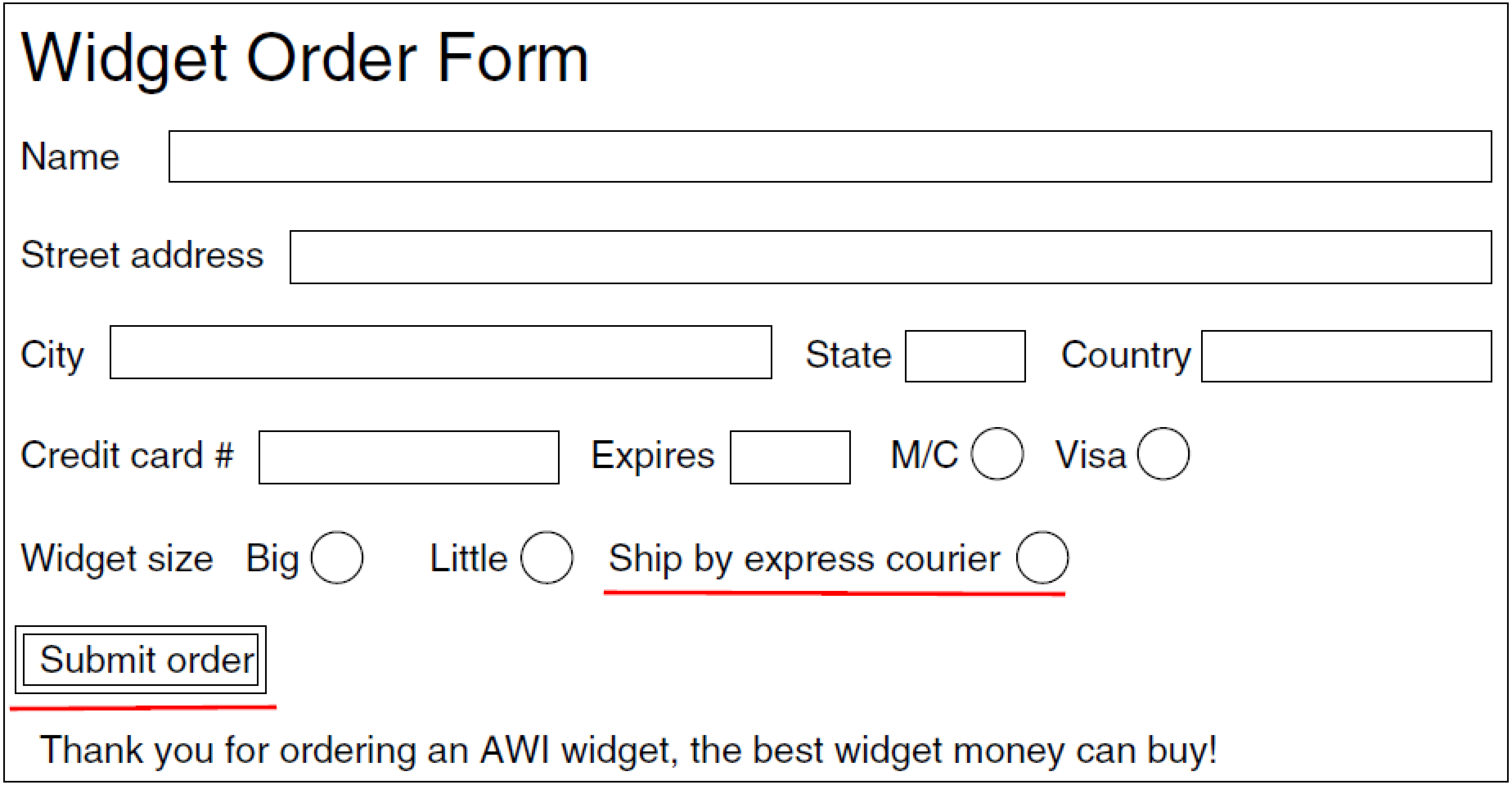
Widget size Big ○ Little ○ Ship by express courier ○

Thank you for ordering an AWI widget, the best widget money can buy!



```
<html>
<head> <title> AWI CUSTOMER ORDERING FORM </title> </head>
<body>
<h1> Widget Order Form </h1>
<form ACTION="http://widget.com/cgi-bin/order.cgi" method=POST>
<p> Name <input name="customer" size=46> </p>
<p> Street address <input name="address" size=40> </p>
<p> City <input name="city" size=20> State <input name="state" size =4>
Country <input name="country" size=10> </p>
<p> Credit card # <input name="cardno" size=10>
Expires <input name="expires" size=4>
M/C <input name="cc" type=radio value="mastercard">
VISA <input name="cc" type=radio value="visacard"> </p>
<p> Widget size Big <input name="product" type=radio value="expensive">
Little <input name="product" type=radio value="cheap">
Ship by express courier <input name="express" type=checkbox> </p>
<p><input type=submit value="Submit order"> </p>
Thank you for ordering an AWI widget, the best widget money can buy!
</form>
</body>
</html>
```

The 3rd kind of input boxes is **checkbox**. Each box of type checkbox can be on or off, independently of all the others.



Widget Order Form

Name

Street address

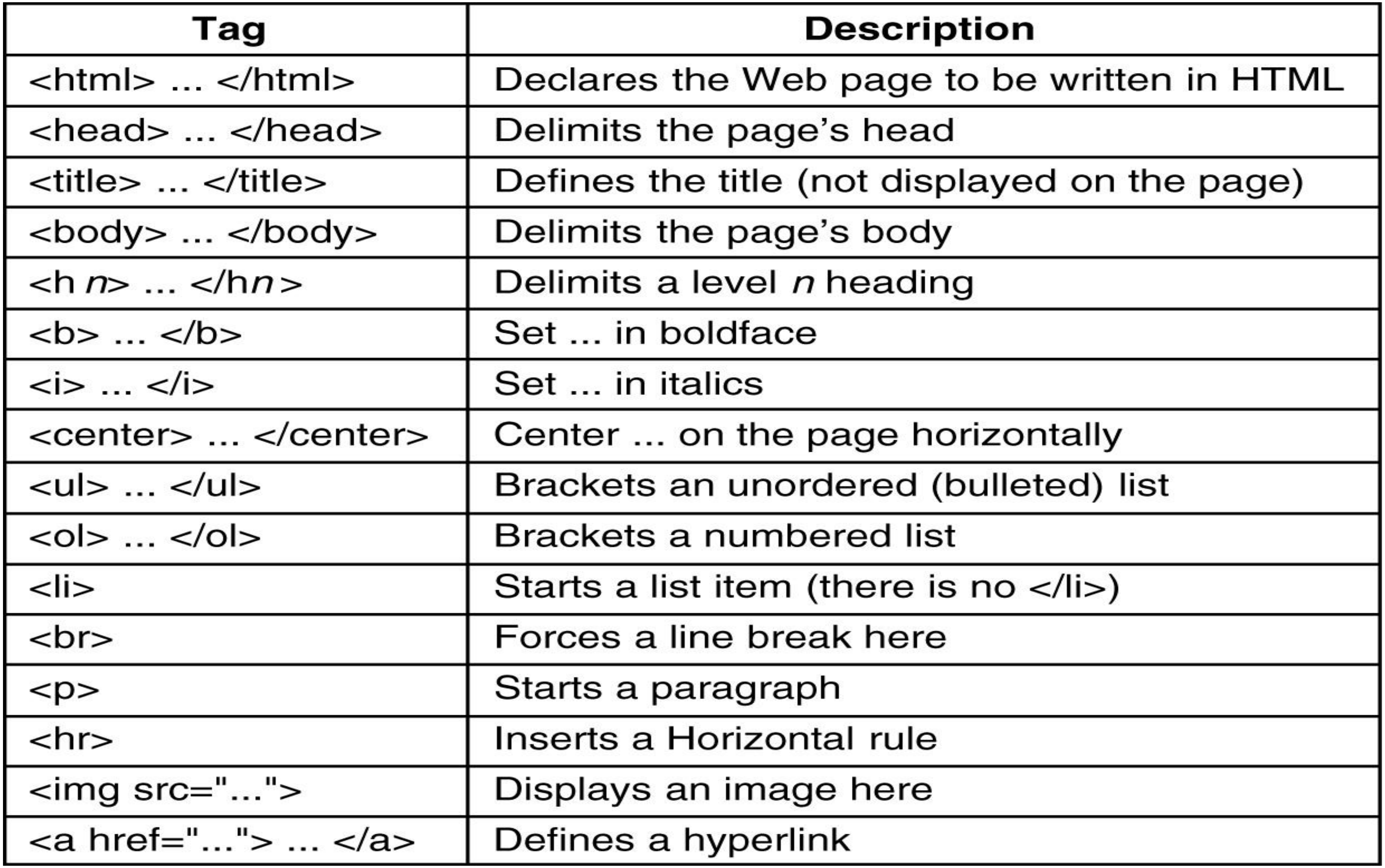
City State Country

Credit card # Expires M/C ○ Visa ○

Widget size Big ○ Little ○ Ship by express courier ○

Thank you for ordering an AWI widget, the best widget money can buy!

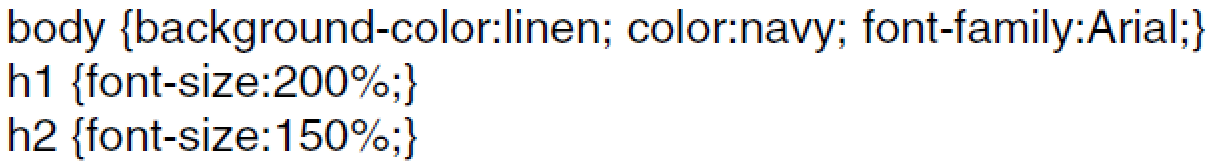
When the user clicks the submit button, the browser packages the collected information into a single long line and sends it back to the server to the URL provided as part of the <**form**> tag.



Tag	Description
<html> … </html>	Declares the Web page to be written in HTML
<head> … </head>	Delimits the page’s head
<title> … </title>	Defines the title (not displayed on the page)
<body> … </body>	Delimits the page’s body
<h *n*> … </h*n* >	Delimits a level *n* heading
<b> … </b>	Set … in boldface
<i> … </i>	Set … in italics
<center> … </center>	Center … on the page horizontally
<ul> … </ul>	Brackets an unordered (bulleted) list
<ol> … </ol>	Brackets a numbered list
<li>	Starts a list item (there is no </li>)
<br>	Forces a line break here
<p>	Starts a paragraph
<hr>	Inserts a Horizontal rule
<img src="…">	Displays an image here
<a href="…"> … </a>	Defines a hyperlink

HTML — CSS (Cascading Style Sheets)

- The original goal of HTML was to specify the structure of the document, not its appearance.
- CSS introduced **style sheets** to the Web with HTML 4.0.
- CSS defines a simple language for describing rules that control the appearance of tagged content.
- The CSS definition example:



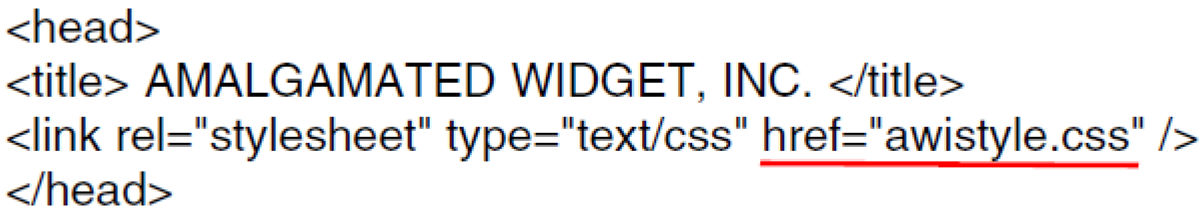
```
body {background-color:linen; color:navy; font-family:Arial;}
h1 {font-size:200%;}
h2 {font-size:150%;}
```

Figure 7-27. CSS example.

- Any style parameters that are not defined are filled with defaults by the browser.

HTML — CSS (II)

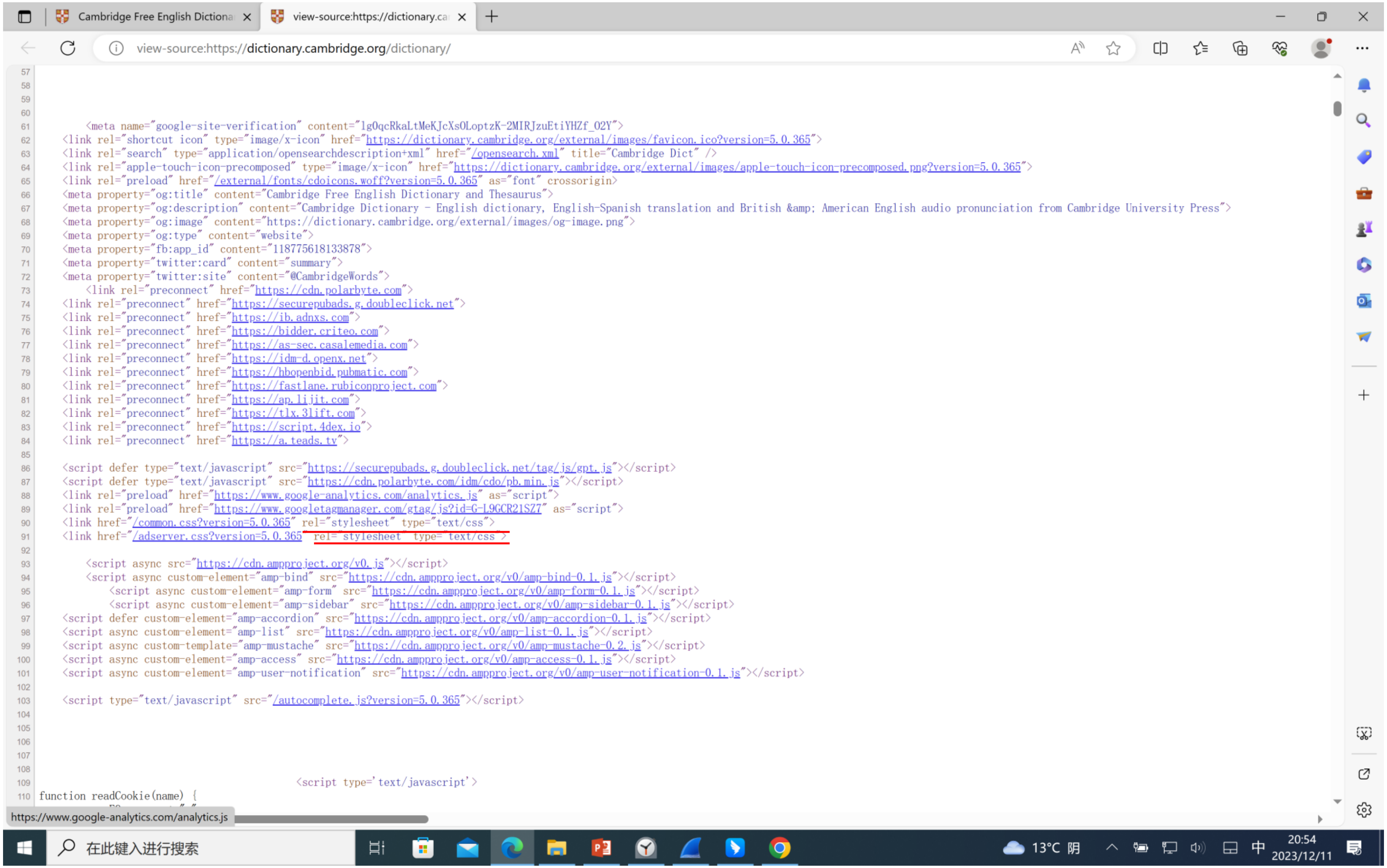
- Style sheets can be placed in an HTML file (e.g., using **the <style> tag**), but it is more common to place them in a separate file and reference them.



```
<head>
<title> AMALGAMATED WIDGET, INC. </title>
<link rel="stylesheet" type="text/css" href="awistyle.css" />
</head>
```

Figure 7-28. Including a CSS style sheet.

- This strategy has two advantages.
  - It lets one set of styles be applied to many pages on a Web site.
    - ~ #include file in a C program
  - It keeps the HTML files that are downloaded small.


Cambridge Free English Dictiona
view-source:https://dictionary.ca
view-source:https://dictionary.cambridge.org/dictionary/
https://www.google-analytics.com/analytics.js
在此键入进行搜索
13°C 阴
中
20:54
2023/12/11

```
        <meta name="google-site-verification" content="lg0qcRkaLtMeKJcXsOLoptzK-2MIRJzuEtiYHZf_02Y">
    <link rel="shortcut icon" type="image/x-icon" href="https://dictionary.cambridge.org/external/images/favicon.ico?version=5.0.365">
    <link rel="search" type="application/opensearchdescription+xml" href="/opensearch.xml" title="Cambridge Dict" />
    <link rel="apple-touch-icon-precomposed" type="image/x-icon" href="https://dictionary.cambridge.org/external/images/apple-touch-icon-precomposed.png?version=5.0.365">
    <link rel="preload" href="/external/fonts/cdoicons.woff?version=5.0.365" as="font" crossorigin>
    <meta property="og:title" content="Cambridge Free English Dictionary and Thesaurus">
    <meta property="og:description" content="Cambridge Dictionary - English dictionary, English-Spanish translation and British &amp; American English audio pronunciation from Cambridge University Press">
    <meta property="og:image" content="https://dictionary.cambridge.org/external/images/og-image.png">
    <meta property="og:type" content="website">
    <meta property="fb:app_id" content="118775618133878">
    <meta property="twitter:card" content="summary">
    <meta property="twitter:site" content="@CambridgeWords">
        <link rel="preconnect" href="https://cdn.polarbyte.com">
    <link rel="preconnect" href="https://securepubads.g.doubleclick.net">
    <link rel="preconnect" href="https://ib.adnxs.com">
    <link rel="preconnect" href="https://bidder.criteo.com">
    <link rel="preconnect" href="https://as-sec.casalemedia.com">
    <link rel="preconnect" href="https://idm-d.openx.net">
    <link rel="preconnect" href="https://hbopenbid.pubmatic.com">
    <link rel="preconnect" href="https://fastlane.rubiconproject.com">
    <link rel="preconnect" href="https://ap.lijit.com">
    <link rel="preconnect" href="https://tlx.3lift.com">
    <link rel="preconnect" href="https://script.4dex.io">
    <link rel="preconnect" href="https://a.teads.tv">

    <script defer type="text/javascript" src="https://securepubads.g.doubleclick.net/tag/js/gpt.js"></script>
    <script defer type="text/javascript" src="https://cdn.polarbyte.com/idm/cdo/pb.min.js"></script>
    <link rel="preload" href="https://www.google-analytics.com/analytics.js" as="script">
    <link rel="preload" href="https://www.googletagmanager.com/gtag/js?id=G-L9GCR21SZ7" as="script">
    <link href="/common.css?version=5.0.365" rel="stylesheet" type="text/css">
    <link href="/adserver.css?version=5.0.365" rel="stylesheet" type="text/css">

        <script async src="https://cdn.ampproject.org/v0.js"></script>
        <script async custom-element="amp-bind" src="https://cdn.ampproject.org/v0/amp-bind-0.1.js"></script>
            <script async custom-element="amp-form" src="https://cdn.ampproject.org/v0/amp-form-0.1.js"></script>
            <script async custom-element="amp-sidebar" src="https://cdn.ampproject.org/v0/amp-sidebar-0.1.js"></script>
    <script defer custom-element="amp-accordion" src="https://cdn.ampproject.org/v0/amp-accordion-0.1.js"></script>
    <script async custom-element="amp-list" src="https://cdn.ampproject.org/v0/amp-list-0.1.js"></script>
    <script async custom-template="amp-mustache" src="https://cdn.ampproject.org/v0/amp-mustache-0.2.js"></script>
    <script async custom-element="amp-access" src="https://cdn.ampproject.org/v0/amp-access-0.1.js"></script>
    <script async custom-element="amp-user-notification" src="https://cdn.ampproject.org/v0/amp-user-notification-0.1.js"></script>

    <script type="text/javascript" src="/autocomplete.js?version=5.0.365"></script>






                                            <script type='text/javascript'>
function readCookie(name) {
```

Dynamic Web Pages

- Dynamic web page is the result of program execution
  - E-commerce, library catalogs, stock market, reading and sending email.
    - For example, a map service that lets user to enter a street address and presents a corresponding map of the location.

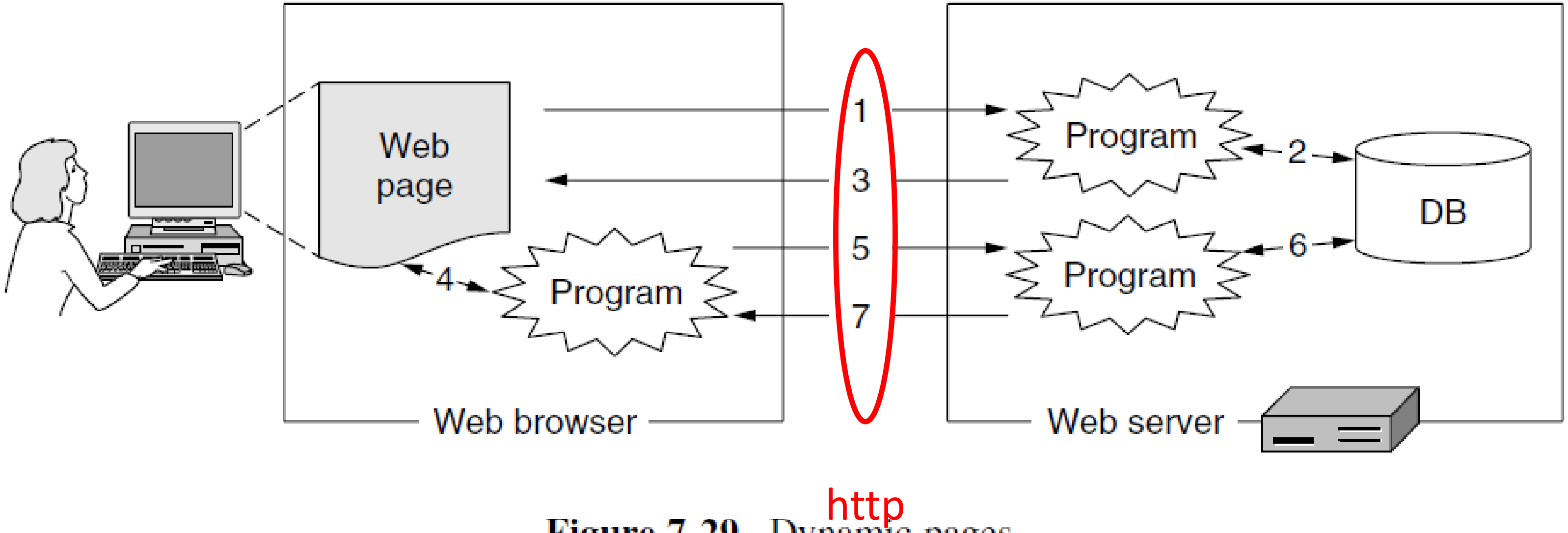


Figure 7-29. Dynamic pages.

1. request; 2. consults a database to generate the appropriate page; 3. return it to the browser; 4. update the page (zoom in or out) need more data; 5. request to the server; 6. retrieve more information; 7. return a response.

Server-Side Dynamic Web Page Generation

- Several APIs (Application Programming Interface) for handling dynamic page requests
  - CGI (Common Gateway Interface) provides an interface to allow Web servers to talk to back-end programs and scripts that can accept input (e.g., from forms) and generate HTML pages in response. — CGI scripts
    - RFC 3875
    - These programs usually be written in a script language, Python, Ruby, Perl.
  - To embed little scripts inside HTML pages and have them be executed by the server itself to generate the page. — embedded PHP
    - **PHP** (In PHP, after the user clicked on the submit button, the browser collects the information into a long string and sends it off to the server as a request for a PHP page.)
  - JSP (JavaServer Pages) is similar to PHP but written in Java programming language.

Client-Side Dynamic Web Page Generation

- Neither PHP nor CGI can respond to mouse movements or interact with users directly. For this purpose, it is necessary to have scripts embedded in HTML pages that are executed on the client machine rather than the server machine.
    - Starting with HTML 4.0, such scripts are permitted using **the tag <script>** — **dynamic HTML** (example Fig. 7-31)
- The most popular scripting language for the client side is **JavaScript**.
    - JavaScript has almost nothing to do with the Java programming language.
- VBScript (随着 Web 技术的发展，VBScript 的使用逐渐减少。主要原因是它的浏览器兼容性问题，因为它主要是由微软的 Internet Explorer 浏览器支持，在其他浏览器如 Firefox、Chrome 等支持较差。)
- Applets (These are small Java programs that have been compiled into machine instructions for a virtual computer called the JVM (Java Virtual Machine))

AJAX — Asynchronous Javascript and XML

- AJAX is not a language. It is a set of technologies that work together to enable Web applications
  - – 1. HTML and CSS to present information as pages.
  - – 2. DOM (Document Object Model, 文档对象模型) to change parts of pages while they are viewed.
    - A representation of an HTML page, and is structured as a tree and reflects the structure of the HTML block.
    - To change parts of the page, there is no need to rewrite the entire page. Only the node that contains the changes needs to be replaced.
  - – 3. XML (eXtensible Makeup Language) to let programs exchange application data with the server.
  - – 4. An asynchronous way for programs to send and retrieval XML data.
  - – 5. JavaScript as a language to bind all this functionality together.

DOM (Document Object Model)

- DOM is a representation of an HTML page that is accessible to programs.
- This representation is structured as a tree that reflects the structure of the HTML elements.
  - At the root is an html element that represents the entire HTML block.
- The significance of the DOM model is that it provides programs with a straightforward way to change parts of the page.
  - There is no need to rewrite the entire page. Only the node that contains the change needs to be replaced.
- The DOM is a powerful method for producing pages that can **evolve**.

DOM (Document Object Model)

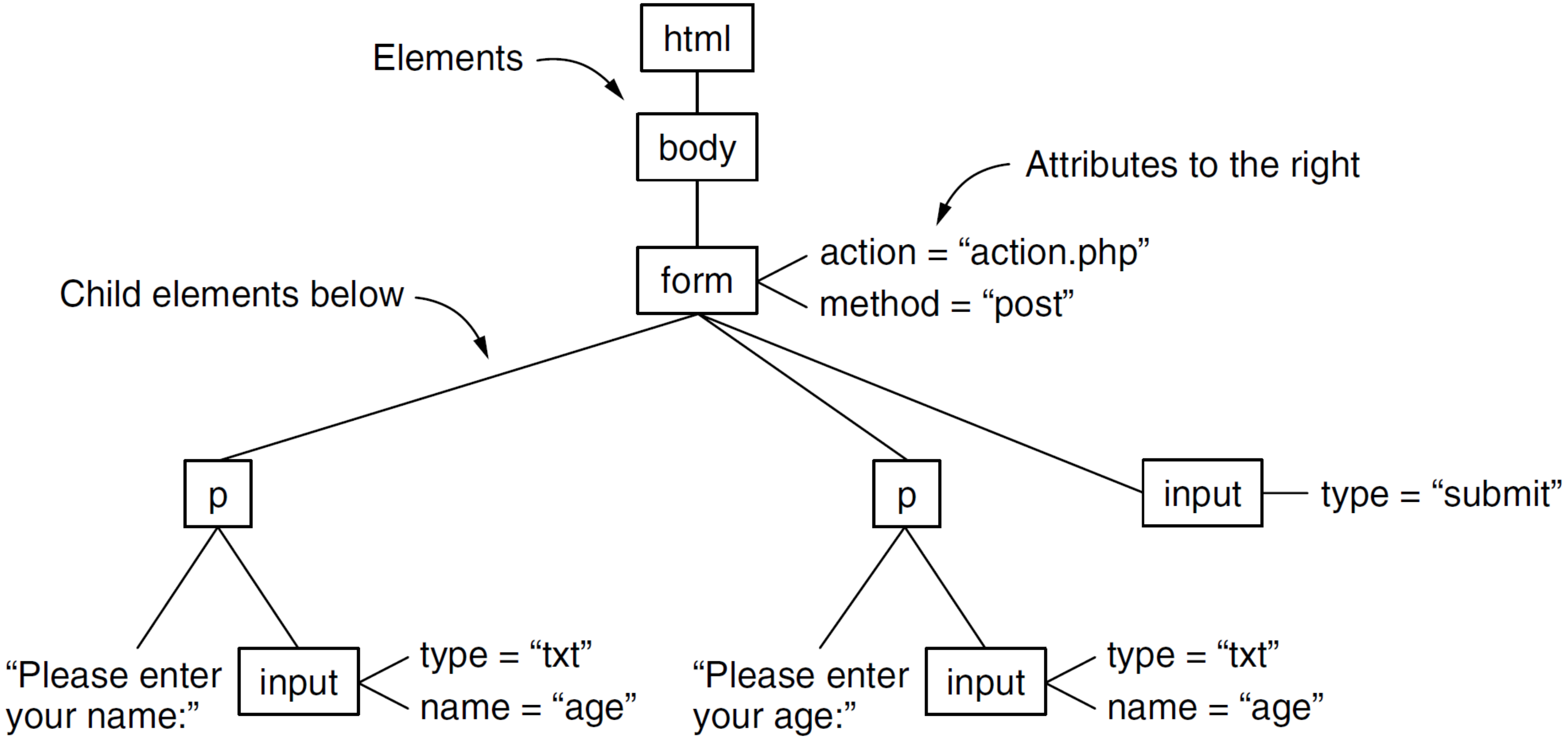


Figure 7-33. The DOM tree for the HTML in Fig. 7-30(a).

Technologies to generate dynamic Web pages

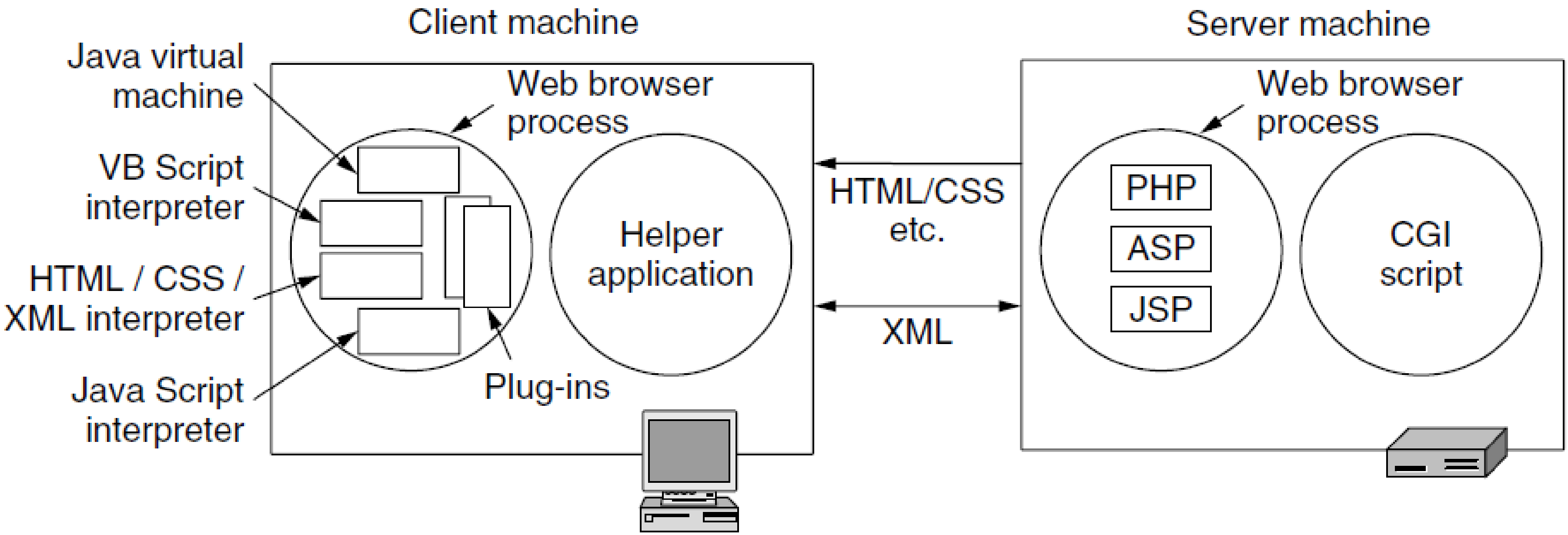


Figure 7-35. Various technologies used to generate dynamic pages.

References

- [1] A.S. Tanenbaum, and D.J. Wetherall, Computer Networks, 5th Edition, Prentice Hall, 2011.
- [2] http://www.evolutionoftheweb.com/
- [3] https://root-servers.org/
- [4] https://www.w3.org/ (http://www.chinaw3c.org/)
- [5] J. F. Kurose and K.W. Ross, Computer Networking — A Top-down Approach, 5th Edition, Pearson Education Inc., 2010.
- [6] https://zhuanlan.zhihu.com/p/53958870 (国内外公共DNS)
- [7] https://blog.csdn.net/qq_24433609/article/details/126548112
- [8] https://zhuanlan.zhihu.com/p/383254776
- [9] https://zhuanlan.zhihu.com/p/32553477
- [10] J. Gehtland, B. Galbraith, and D. Almaer, Progmatic Ajax — A Web 2.0 Primer, 电子工业出版社 (徐锋，胡冰译) 2006.