Internet Applications

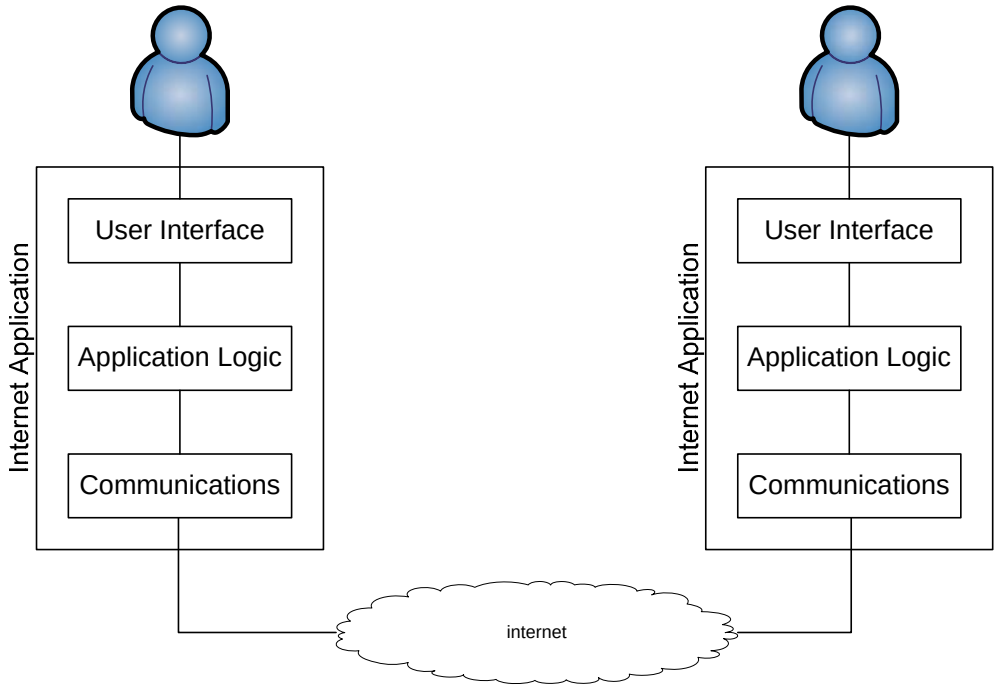
Dr Steve Gordon
ICT, SIIT

Contents

- What are Internet Applications?
- Implementing Internet Applications
- Selected Applications and Protocols
  - Naming and DNS
  - Web Access and HTTP
  - Email, SMTP and POP

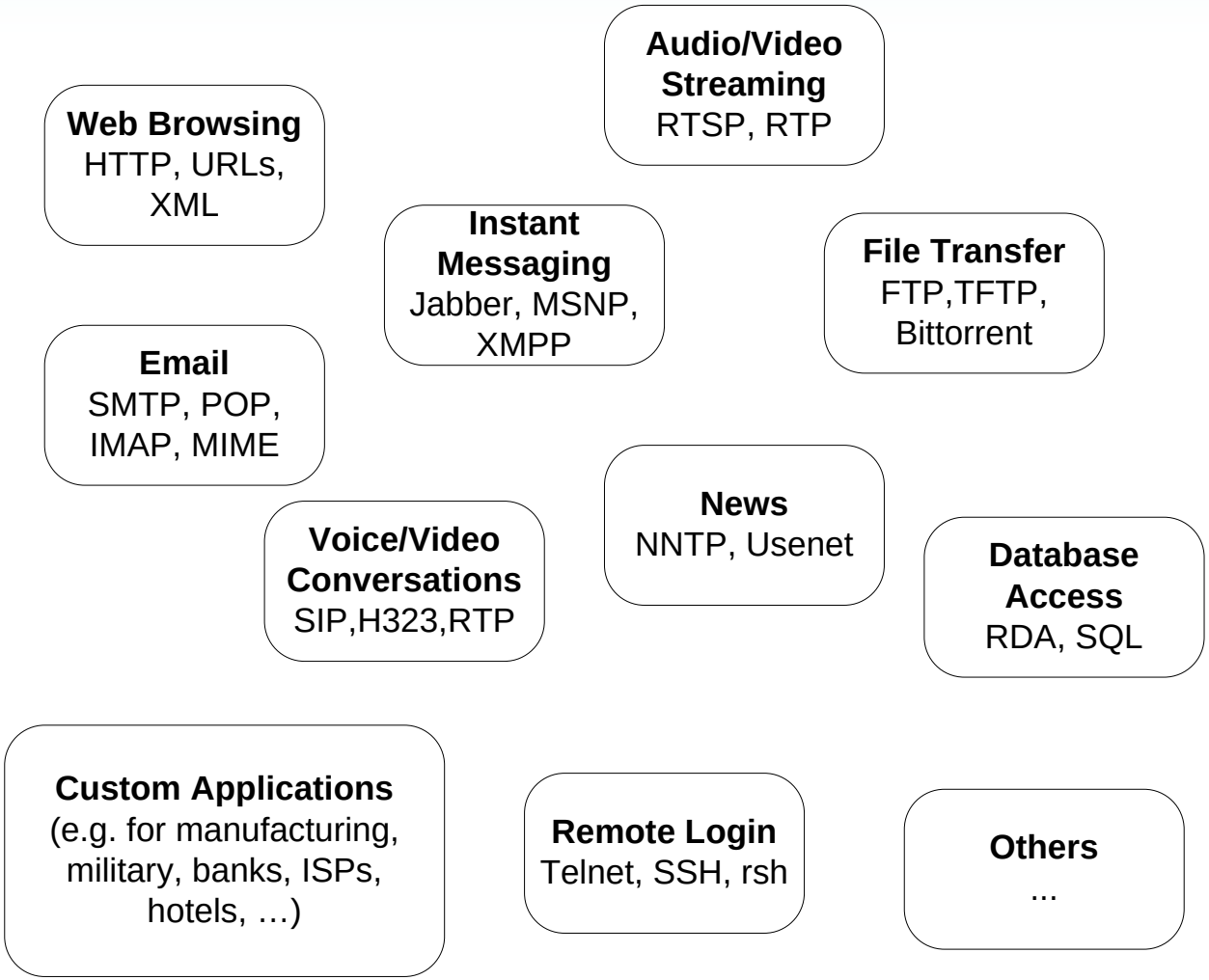
What are Internet Applications?

- Software applications that involve communication with other applications over the Internet
- Internet applications have three basic functions:
  - User interface
  - Application logic
  - Communications
- Example: web browser
  - UI: display web page on screen; process clicks on links; …
  - Logic: parse web page to generate graphical output; save files; …
  - Comms: request and receive web pages



- Communications follow an *application layer protocol*
  - Allows different implementations of applications to communicate
  - E.g. Microsoft Internet Explorer communicates with Apache Web Server

Types of Internet Applications



Client/Server Applications

- Most Internet applications follow a client/server model
  - Client initiates communication
    - Sends Transport layer segments to a server port
  - Server waits for client to initiate communication
    - “Listens” on a well-known port
  - Once the communication is initiated, data can flow in both directions (client to server and server to client)

- Other models:
  - Pre-collection of information: a software process runs all the time and collects information. When a user application needs some information, it uses the pre-collected information. Inefficient use of network and CPU
  - Peer-to-peer: in fact, almost all “peer-to-peer applications” follow a client/server model, however an application may act as both client and server

Implementing Internet Applications

- Internet applications are implemented as user-level software processes
- Application layer protocols make use of Transport layer for communications
  - Remember Transport and Network layer are part of operating system
- How do you write an Internet application that can use TCP (or UDP or other) to send packets?
  - There is an Application Programming Interface (API) commonly referred to as *Sockets*
    - Based on original implementation of TCP/UDP communications in Unix
    - Similar API is used by almost all operating systems, and in many different programming languages

Example Sockets Interface

- A socket is an abstraction for a communication end-point
- For example, using TCP:
  - A server will associate a socket with a port, and listen on that socket
  - A client will create a socket, and connect it to the socket at the server
  - The client can then send (or write) to its socket, and TCP will deliver the data to the socket at the server, where the server receives or reads the data
- Standard socket API (some operating systems/languages may use slightly different syntax)
  - Create a socket: `socketid = socket (protocol_family, comms_type, protocol)`
  - Bind socket to address: `bind (socketid, addr, addr_length)`
  - Connect socket to a destination : `connect (socketid, destaddr, addr_length)`
  - Prepare socket for incoming connections: `listen (socketid, queue_length)`
  - Wait for incoming connection: `newsocketid = accept (socketid,addr, addr_length)`
  - Sending and receiving data:
    ```
    write (socketid, data, length)
    read (socketid, buffer, length)
    ```
  - Close a connection: `close (socketid)`

Socket Example

- Client Application

```
s = socket(PF_INET,SOCK_STREAM,0);
connect(s, 172.17.3.12:23, len);

write(s, “Request”, 7);


read(s, buffer, 5);
close(s);
```

- Server Application

```
s = socket(PF_INET,SOCK_STREAM,0);
bind(s, 172.17.3.12:23, len);
listen(s, 5);
while(1) {

   snew = accept(s,clientaddr,len);

   read(snew, buffer, 7);
   write(snew, “Reply”, 5);
   close(snew);
}
```

Naming and DNS

Identifying Computers on the Internet

- We need some method of identifying resources (such as computers and files) in networks
  - Should be consistent, unique and user-friendly
- IP addresses are used to identify computer (interfaces)
  - If a user knows the IP address of a computer, then they can communicate with that computer
  - However IP addresses are not user-friendly
- *Domain names* are user-friendly way to identify computers
  - Domain names follow a hierarchical structure: an organisation manages a domain, and can allocate sub-domains to other organisations
    - E.g. THNIC manages .th domain (including .ac.th)
    - Thammasat University obtains .tu.ac.th from THNIC
    - SIIT obtains .siit.tu.ac.th from TU
    - The SIIT computer centre allocates names for different computers: www.siit.tu.ac.th, reg.siit.tu.ac.th, ict.siit.tu.ac.th, …

Example Domains

- Top-Level Domains (TLDs)
  - .biz .com .info .name .net .org .pro
  - .aero .asia .cat .coop .edu .gov .int .jobs .mil .mobi .museum .tel .travel
  - These TLDs are managed by commercial and government organisations, called Domain Name Registries (e.g. VeriSign)
- Country Code Top-Level Domains (ccTLD)
  - Two letter identifiers for countries, e.g. .th .us .au .uk .eu …
  - ccTLDs are managed by national registries
  - Typically sub-domains are created by the national registries
    - E.g. .ac.th, .co.th, .go.th, .or.th, .mi.th, .net.th, .in.th, .or.th

Identifying Files on the Internet

- Uniform Resource Locators (URLs) are used to identify files (resources) on the Internet
  - URLs are in fact a specific form of Uniform Resource Identifier (URI)
  - Format of URI:

```
scheme : user @ host : port  path ? query
```

    - `scheme`: identifies the application protocol used to access the resource, e.g. http, ftp, https, dns, ipp, news, sip, …; often followed by //
      - IANA official schemes: http://www.iana.org/assignments/uri-schemes.html
      - Unofficial schemes: http://en.wikipedia.org/wiki/URI_scheme
    - `user`: identifies the user that is accessing the resources; password is optional
    - `host`: identifies the host that stores the resources; typically a domain name or IP address
    - `port`: identifies the port number of the application on the host; if not given, the default value for the scheme will be used, e.g. http (80); https (443)
    - `path`: pathname of the file where the resource is located
    - `query`: additional identification information for the resource location; typically in attribute-value pairs, e.g. key=value
  - Most parts are optional and there are exceptions!

Example URLs

- Web pages

```
http://www.example.com/dir/file.html
http://73.16.0.4:40240/dir/file.html
https://www.example.com/dir/file.html
http://example.com/dir/file?id=6&name=steve
```

- Email addresses

```
mailto:steve@example.com
mailto:steve@example.com?subject=test
```

- Remote login

```
telnet://steve@example.com
telnet://steve:mypassword@example.com:46
ssh://steve@example.com
```

Domain Name System

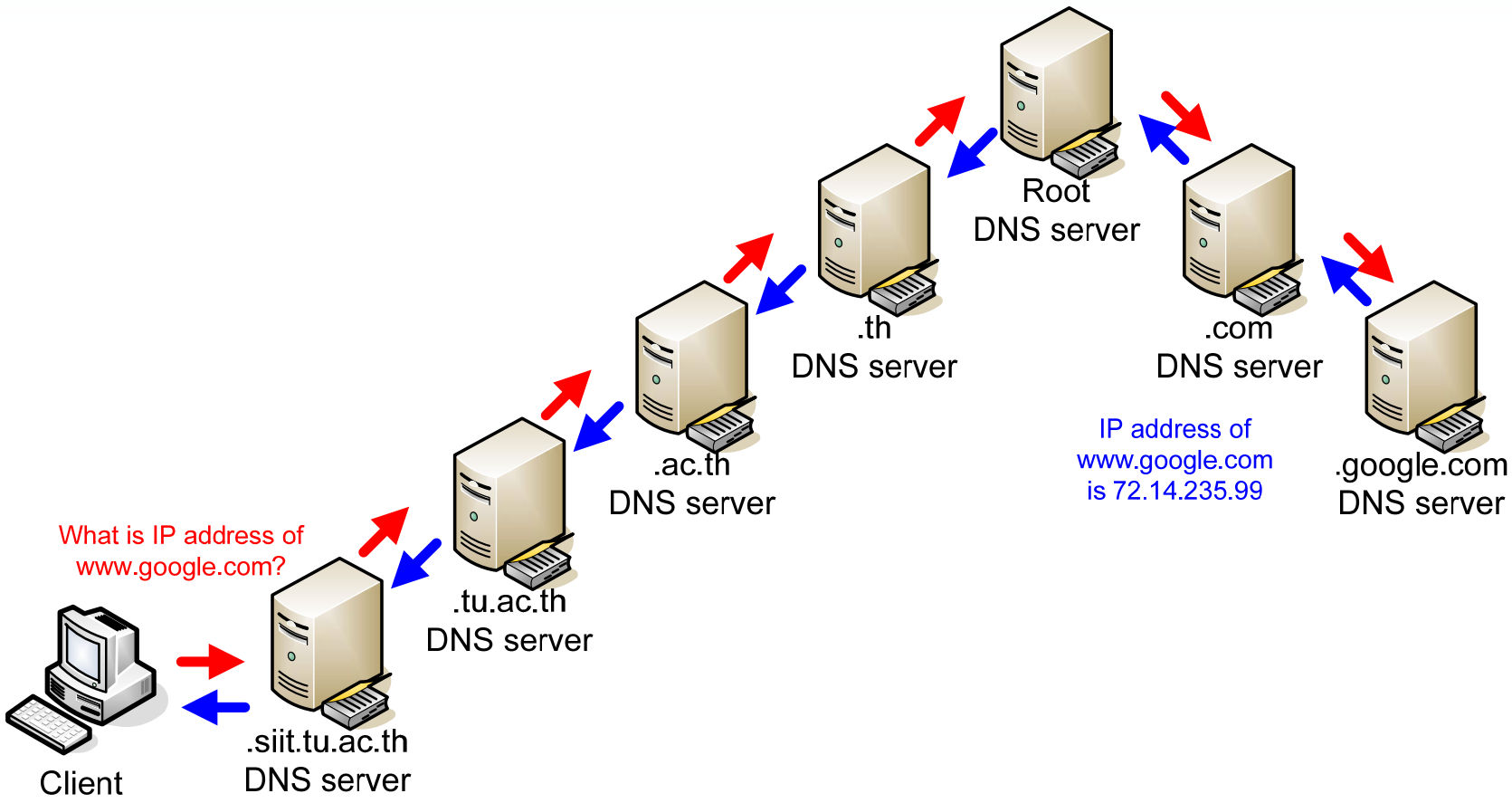
- Domain Name System (DNS) specifies:
  - The format and structure of domain names
  - How to map domain names to IP addresses
    - Computers use IP addresses to communicate; users use domain names as a user-friendly way to communicate; need to join the two approaches
- DNS Protocol
  - Domain names and their corresponding IP address are registered at DNS servers
    - Registration may be manual (e.g. if the IP/domain does not change often) or automatic (e.g. if your IP address changes often, such as on a home ADSL internet connection)
  - When applications have a domain name, the application uses DNS protocol to retrieve the corresponding IP address from the DNS server
  - Then the IP address is used by TCP or UDP to send data to the destination computer
- There is a hierarchy of DNS servers across the globe so that your requests are fast
- DNS may send messages using UDP (most cases) or TCP (when large responses expected)

Recursive DNS

- Assume each DNS Server knows of domain names/IP addresses within its domain
  - Google.com DNS Server knows IP address of www.google.com, code.google.com, mobile.google.com, …
  - .th DNS Server knows IP address of .ac.th DNS Server, .go.th DNS Server, .mi.th DNS Server, …
  - Root DNS Server knows IP address of .com DNS Server, .th DNS Server, …
- Query for IP address is sent from client to its local DNS server for the domain
  - Each DNS Server sends query up to parent domain DNS server (up to root), and then down to destination domain server (google.com)
- Response is sent back via the DNS servers, eventually to the client

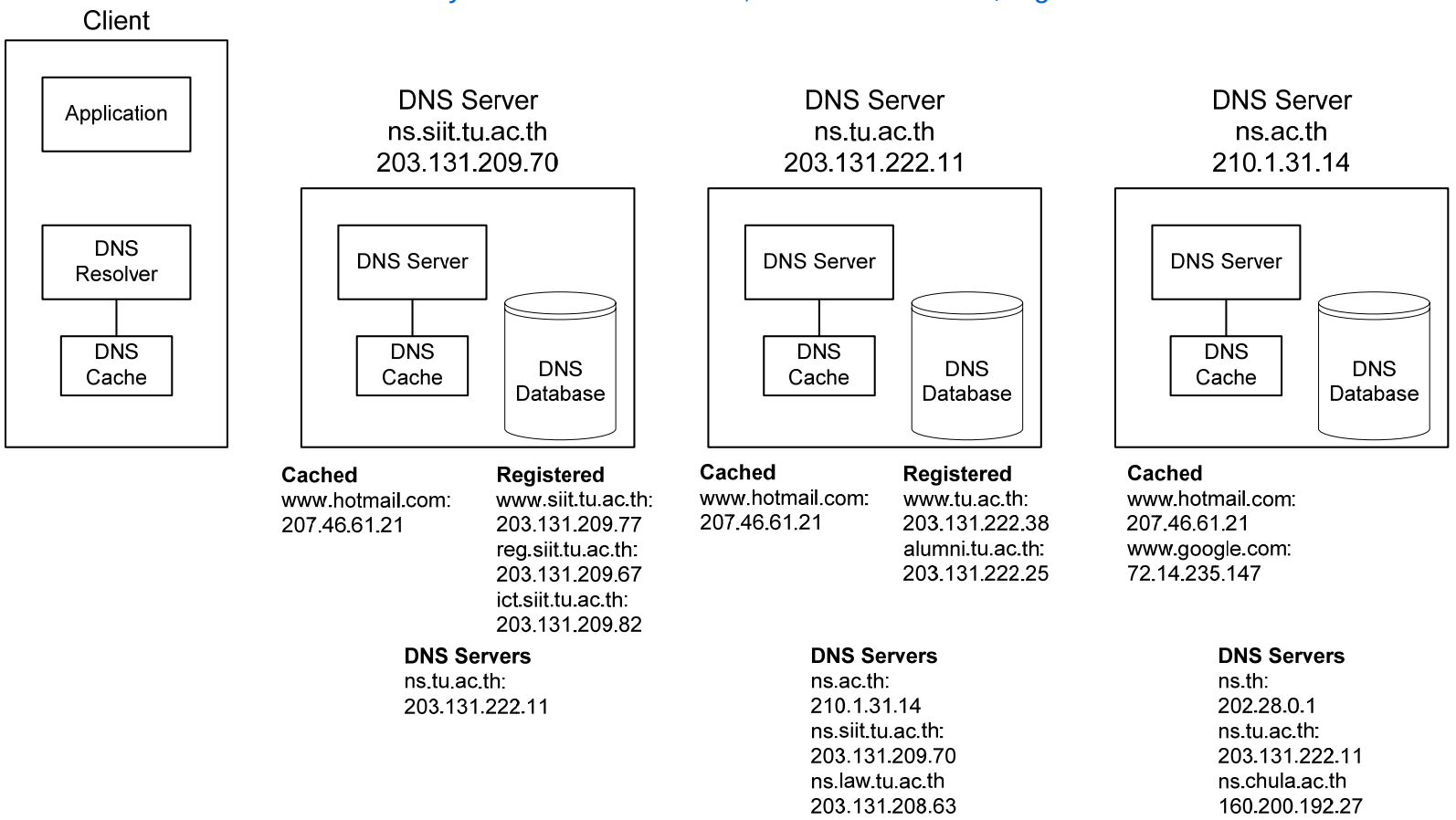
- Performance is improved via caching
  - Each DNS server (and client) caches recently requested domain names (and their IP addresses)
  - If DNS server has the domain name in cache, then can respond immediately

Recursive DNS



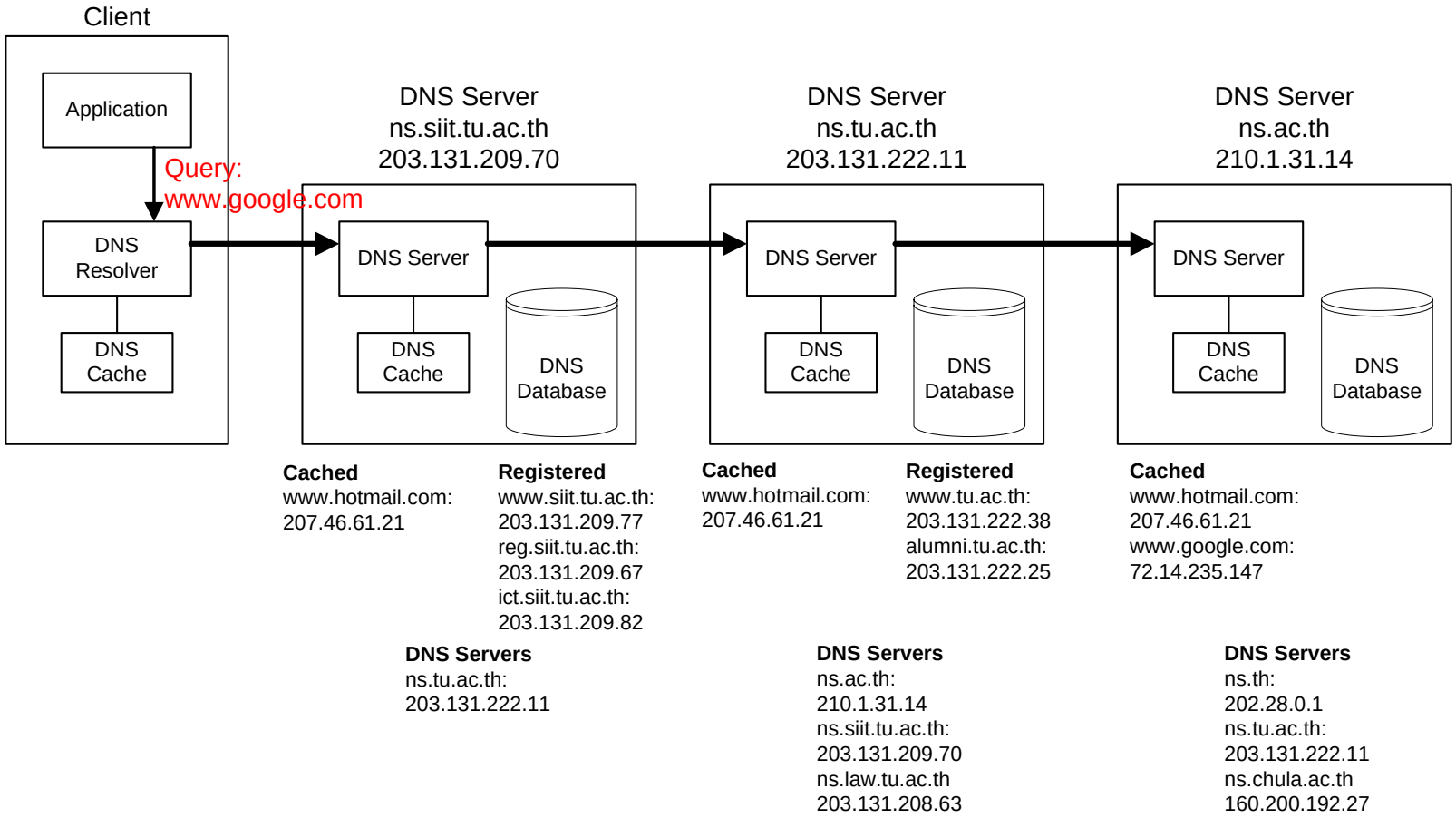
Recursive DNS with Caching

- Initial state of selected DNS Servers
  - DNS Database stores the domain names (and IP address) registered with that server
  - DNS Cache temporarily stores domain names (and IP address) requested in the past
  - A DNS Server also stores the list of other DNS Servers it knows about (its parent and children)
    - Each DNS Server may have a domain name; often called ns. …, e.g. DNS Server at SIIT is ns.siit.tu.ac.th



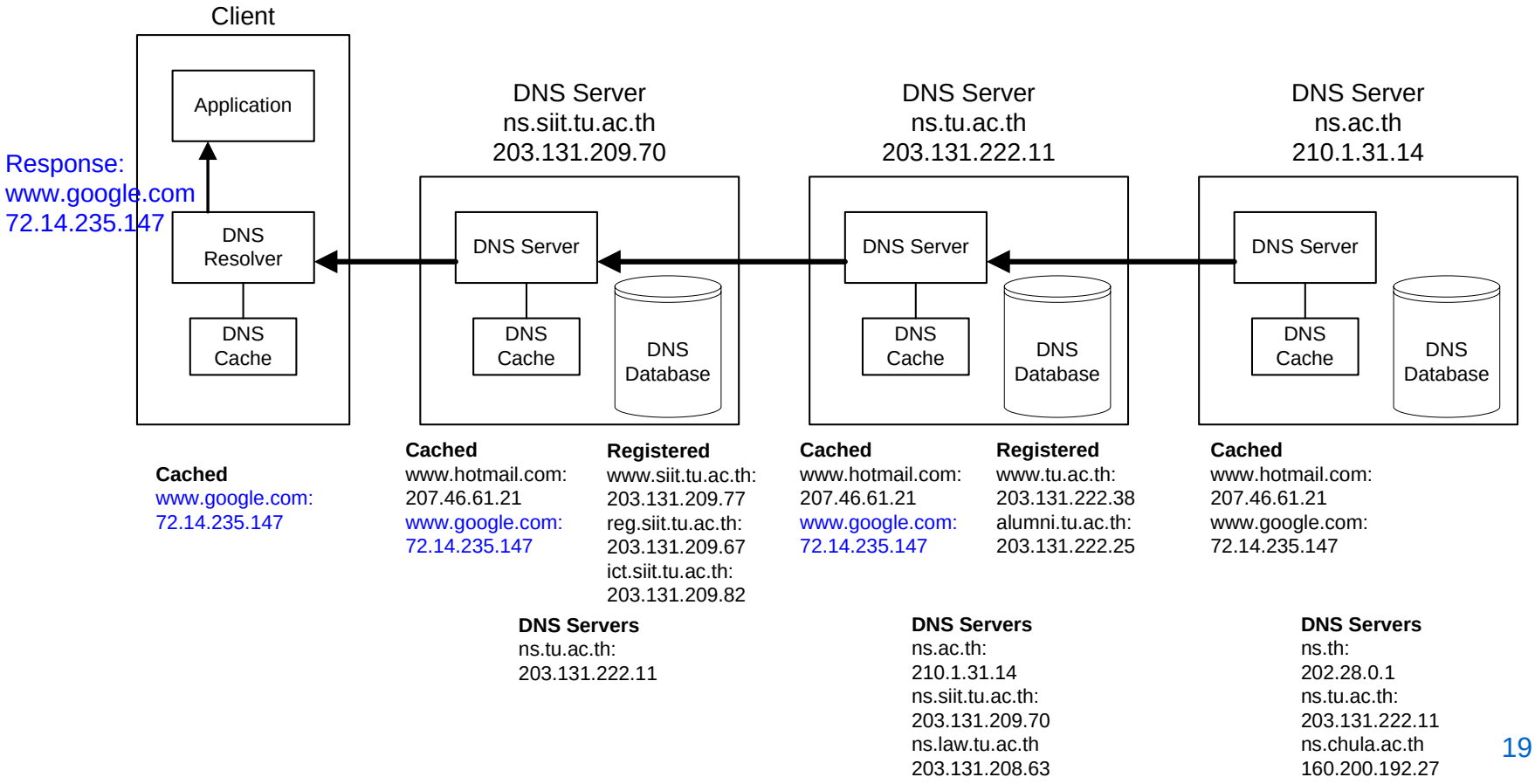
Recursive DNS with Caching

- User of application on client wants to send data to www.google.com
  - Application uses DNS Resolver (a DNS client application) to send a Query for the domain name www.google.com
  - If a DNS Server does not know the IP address, then Query sent to next DNS Server



Recursive DNS with Caching

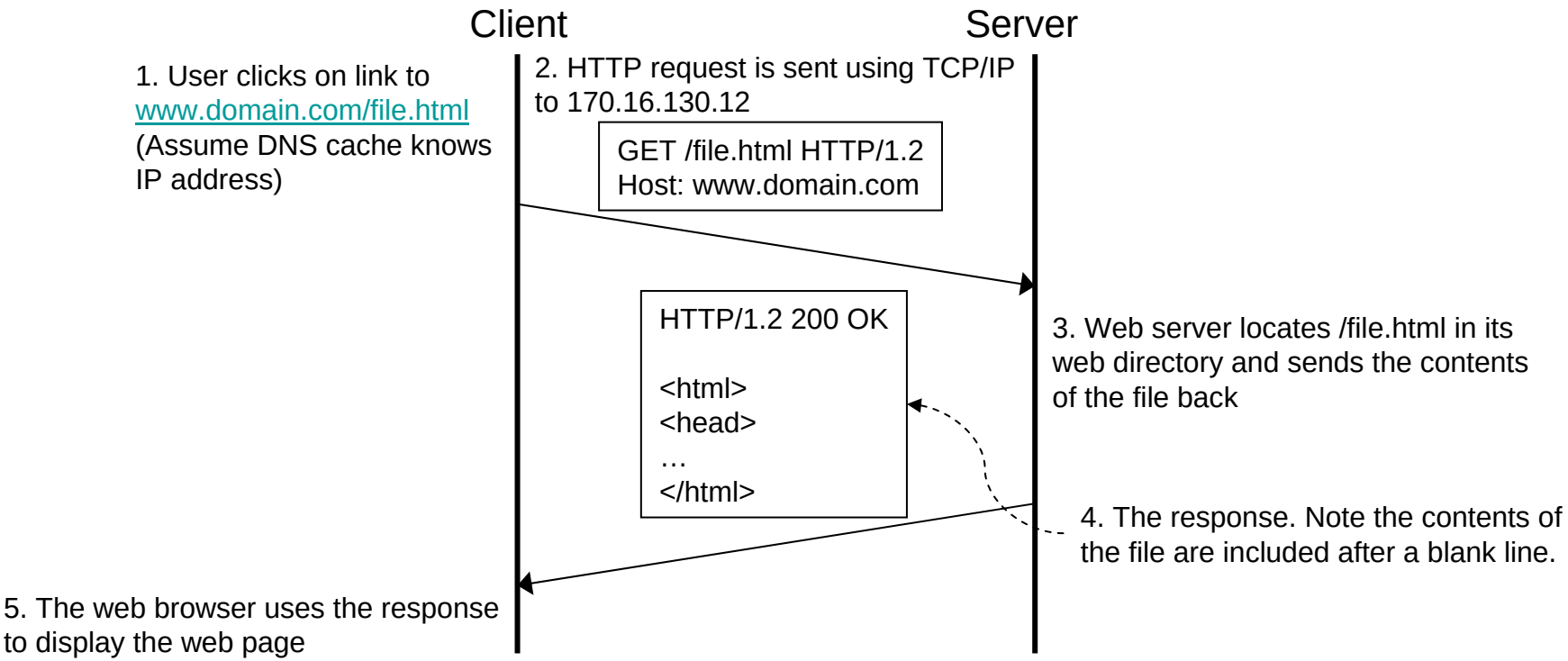
- ns.ac.th has www.google.com in its cache, hence returns a DNS Response
  - The DNS Servers and Client that receive the response, also update their caches
  - IP address eventually delivered to client application, which can now send data to 72.14.235.147 (that is, www.google.com)



Web Access and HTTP

Web Access

- HyperText Transfer Protocol (HTTP)
  - Request/response protocol using TCP: default port 80 for servers
  - Client is called User Agent (e.g. web browser) and server is Web (or HTTP) Server



HTTP Messages

- HTTP is Request/Response protocol
  - Client sends a Request, and server responds with a single Response message
- HTTP is state-less
  - HTTP maintains no state between requests
  - The application that uses HTTP may maintain state
- The format of a generic HTTP message is:

```
Start line
Optional header lines
<empty line>
Optional message body
```

HTTP Request Messages

- Clients send HTTP Requests
  - Start line: `Method URL Version`
    - Example: `GET http://example.com/index.html HTTP/1.1`
  - Methods:
    - GET – retrieve the resource at the specific URL
    - HEAD – same as GET, except do not return message body (only header)
    - OPTIONS – retrieve options available for resource or server
    - POST – asks server to accept and process the attached data at the resource
    - PUT – asks server to put the attached data at the resource
    - DELETE – asks the server to delete the resource
    - TRACE – server replies with the Request in the message body; used for client to see what exactly is received by server (for example, if goes via proxy)
    - CONNECT – used for creating secure connection to proxy
  - Version: version of HTTP, e.g. HTTP/1.0, HTTP/1.1

HTTP Response Messages

- Server sends HTTP Response
  - Start line: `HTTPVersion StatusCode StatusReason`
    - Example: HTTP/1.1 200 OK
  - Status Codes and Reasons:
    - 1xx: Informational
      - 100: Continue (the client should continue with its request)
    - 2xx: Success
      - 200: OK (the request succeeded)
    - 3xx: Redirection
      - 301: Moved Permanently (the requested resource has a new URL)
      - 304: Not Modified (resource hasn't changed since last request, client should use cached copy)
    - 4xx: Client error
      - 401: Unauthorized (request must include user authentication)
      - 403: Forbidden (request was understood, but server refuses to process it)
      - 404: Not Found (server cannot find resource at requested URL)
    - 5xx: Server error
      - 503: Service Unavailable (server currently unable to handle request, e.g. server is too busy)

HTTP Headers

- Header Format: `field-name: field-value`
- General Header Fields
  - Date: data and time of message generation
- Request Header Fields
  - Host: domain name of host of resource (means relative URLs can be used)
  - Accept: indicates the types of media the client can accept (or prefer) in response
    - Example: `Accept: text/plain; q=0.5; text/html, text/x-dvi; q=0.8, text/x-c`
    - "text/html and text/x-c are the preferred media types, but if they do not exist, then send text/x-dvi, and if that does not exist, send text/plain"
  - Accept-Charset, Accept-Encoding, Accept-Language: indicate the character sets, encodings and languages that client can accept
  - Authorization: include user credentials (e.g. username, password) if authorization is required
  - If-Modified-Since: specifies a date/time; server should only return resource if it has been modified since the date/time (otherwise return 304 Not Modified)
  - User-Agent: indicates information about the client (user agent), e.g. web browser
  - Cookie: a HTTP cookie previously sent by server
  - Referrer: URL from which this request came from

HTTP Headers

- Response Header Fields
  - Location: contains the new URL if redirection is used
  - Set-Cookie: a HTTP cookie
  - Cache-Control: used to control how the resource is cached (e.g. no-cache)
- Header Fields about Message Body (may be in Request/Response depending on type)
  - Content-Encoding: encoding or compression, e.g. gzip
  - Content-Length: length of message body on bytes
  - Content-Type: the type of content in message body
  - Last-Modified: indicates data/time when content was last modified on server
  - Expires: data/time when content in message body is considered ‘stale’ or no longer relevant

Web Access Example

- Web Server software (Apache httpd) is running on host sandilands.info (IP: 125.25.85.14)
  - Base directory of web server is: `/var/www/html/`
  - Contents of directory `/var/www/html/its323/`

```
web@basil:/var/www/html/its323$ ls -l
total 16
-rw-r--r-- 1 web www 7399 2006-09-22 14:41 image.gif
-rw-r--r-- 1 web www  156 2008-09-13 15:50 index.html
-rw-r--r-- 1 web www  132 2008-09-13 15:42 test.html
```

- Web browser (Firefox) is running on host 192.168.1.2
  - User enters the following URL in browser:

```
http://sandilands.info/its323/
```

  - Then user clicks on a link in the page to `test.html`
  - Then user enters following URL in browser:

```
http://sandilands.info/its323/nopage.html
```

ITS323 Example - Mozilla Firefox

File Edit View History Bookmarks Tools Help

http://sandilands.info/its323/

Google

This is a example web page for ITS323. Link to Test page

Done

ITS323 Test Page - Mozilla Firefox

File Edit View History Bookmarks Tools Help

http://sandilands.info/its323/test.html

Google

Test page with image:

Done

404 Not Found - Mozilla Firefox

File Edit View History Bookmarks Tools Help

http://sandilands.info/its323/nopage.html

Google

Not Found

The requested URL /its323/nopage.html was not found on this server.

Apache/2.2.8 (Ubuntu) DAV/2 SVN/1.4.6 PHP/5.2.4-2ubuntu5.1 with Suhosin-Patch mod_ssl/2.2.8 OpenSSL/0.9.8g Server at sandilands.info Port 80

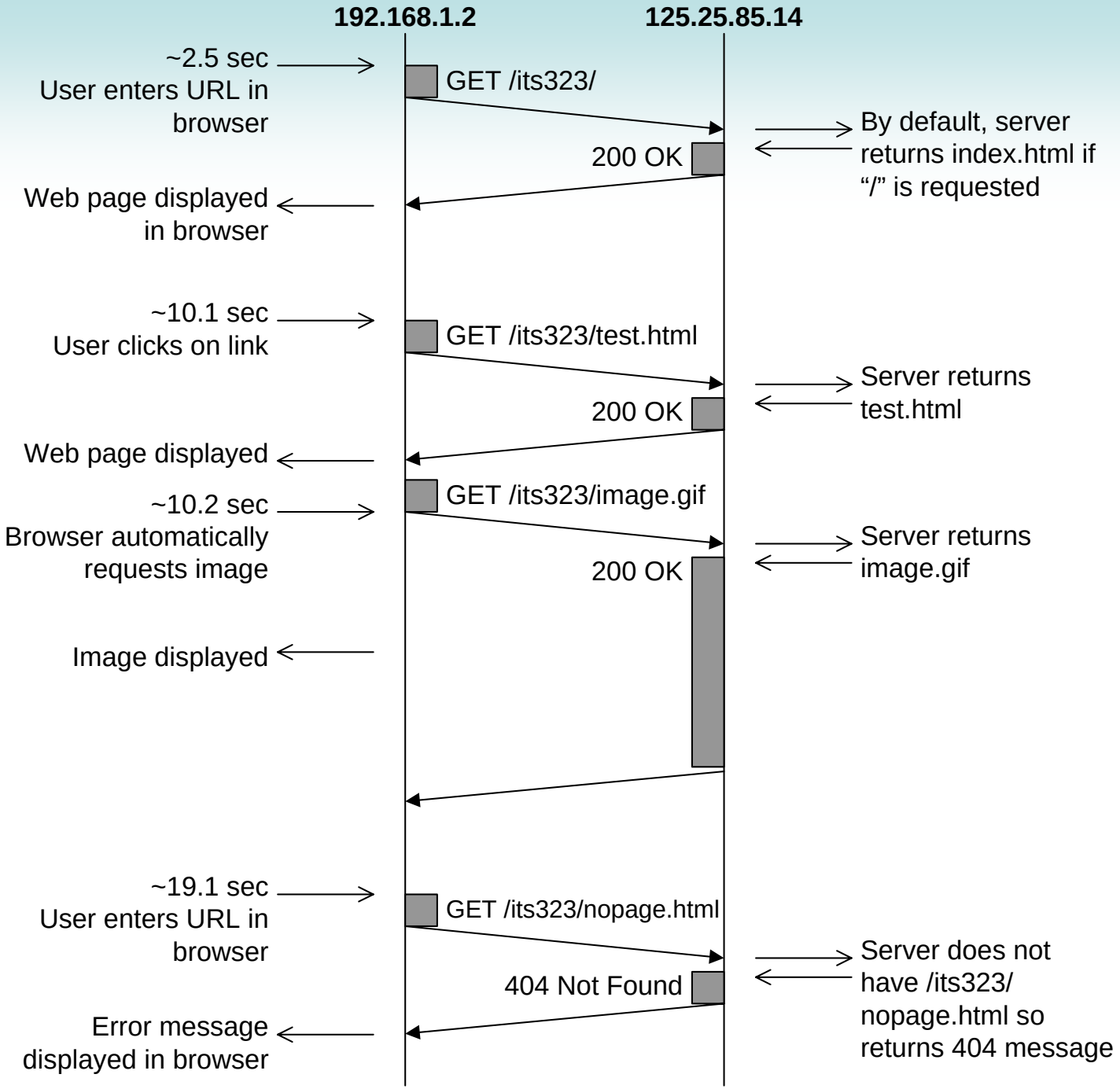
Done

Capture of HTTP Messages

- The messages sent between client and server were recorded (captured)
  - Start time of capture is 0 seconds
  - 8 HTTP messages captured shown below

	Time	Source	Destination	Short Description
1	2.569691	192.168.1.2	125.25.85.14	GET /its323/ HTTP/1.1
2	2.574261	125.25.85.14	192.168.1.2	HTTP/1.1 200 OK (text/html)
3	10.182447	192.168.1.2	125.25.85.14	GET /its323/test.html HTTP/1.1
4	10.185977	125.25.85.14	192.168.1.2	HTTP/1.1 200 OK (text/html)
5	10.212269	192.168.1.2	125.25.85.14	GET /its323/image.gif HTTP/1.1
6	10.224590	125.25.85.14	192.168.1.2	HTTP/1.1 200 OK (GIF89a)
7	19.156073	192.168.1.2	125.25.85.14	GET /its323/nopage.html HTTP/1.1
8	19.159985	125.25.85.14	192.168.1.2	HTTP/1.1 404 Not Found (text/html)

  - User entered URL `http://sandilands.info/its323/` at approx. 2.5 seconds
  - User clicked on link to `test.html` at approx. 10.1 seconds
  - User entered URL `http://sandilands.info/its323/nopage.html` at approx. 19.1 seconds


192.168.1.2
125.25.85.14
~2.5 sec
User enters URL in browser
GET /its323/
By default, server returns index.html if "/" is requested
200 OK
Web page displayed in browser
~10.1 sec
User clicks on link
GET /its323/test.html
Server returns test.html
200 OK
Web page displayed
~10.2 sec
Browser automatically requests image
GET /its323/image.gif
Server returns image.gif
200 OK
Image displayed
~19.1 sec
User enters URL in browser
GET /its323/nopage.html
Server does not have /its323/ nopage.html so returns 404 message
404 Not Found
Error message displayed in browser

First HTTP Request

```
GET /its323/ HTTP/1.1
Host: sandilands.info
User-Agent: Mozilla/5.0 (Windows; U; Windows NT 5.1; en-GB; rv:1.9.0.1)
                                    Gecko/2008070208 Firefox/3.0.1
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-gb,en;q=0.5
Accept-Encoding: gzip,deflate
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
Keep-Alive: 300
Connection: keep-alive
Cookie: SESS3fda6fb4416e6a89c4fcdeb88b2b8525=b7b16364a416dcf1e6bfa4fcfb7a5ec4;
        SESS780c4f49efc943a475d912e4fd3eb8bc=bad79640552be2dac128f66f7d8bba43
If-Modified-Since: Sat, 13 Sep 2008 08:37:54 GMT
If-None-Match: "c2804-a2-456c2eb0d3c80"
Cache-Control: max-age=0
```

Length of HTTP message: 645 bytes

First HTTP Response

```
HTTP/1.1 200 OK
Date: Sat, 13 Sep 2008 08:50:28 GMT
Server: Apache/2.2.8 (Ubuntu) DAV/2 SVN/1.4.6 PHP/5.2.4-2ubuntu5.1 with
                  Suhosin-Patch mod_ssl/2.2.8 OpenSSL/0.9.8g
Last-Modified: Sat, 13 Sep 2008 08:50:13 GMT
ETag: "c2804-9c-456c317197b40"
Accept-Ranges: bytes
Content-Length: 156
Keep-Alive: timeout=15, max=100
Connection: Keep-Alive
Content-Type: text/html

<html>
<head>
<title>ITS323 Example</title>
</head>
<body>
This is a example web page for ITS323
<a href="test.html">Link to Test page</a>
</body>
</html>
```

Length of HTTP message: 531 bytes

Email, SMTP and POP

Email

- A User Agent creates an email and sends to a local Mail Transfer Agent (MTA)
  - Example User Agents: Microsoft Outlook, Thunderbird, Eudora, LotusNotes and web mail clients (Hotmail, Gmail, Yahoo Mail, …)
  - Format of email is defined (originally in IETF RFC 822), with extensions such as MIME
  - Simple Mail Transfer Protocol (SMTP) is used to send to MTA
- The local MTA sends the email to the destination MTA
  - SMTP is used to transfer emails between MTAs
  - A MTA generally called an “Mail Server”
- Destination MTA stores email in mailbox (e.g. as a file), and destination User Agent retrieves email and displays to user
  - Post Office Protocol (POP) or Internet Mail Access Protocol (IMAP) is used by User Agent at destination to retrieve the email

Email Format

- Original format (RFC822)
  - Header lines of the format: `field-name: field-value`
  - <blank line>
  - Message body (optional)

  - Email (headers and body) was in 7-bit ASCII text

- Header Fields
  - From, To, CC, BCC, Subject, Date, Message-ID, Reply-To, …
  - List of fields at: http://www.iana.org/assignments/message-headers/perm-headers.html

- Email Addresses
  - URI
    - username @ host

ASCII Table

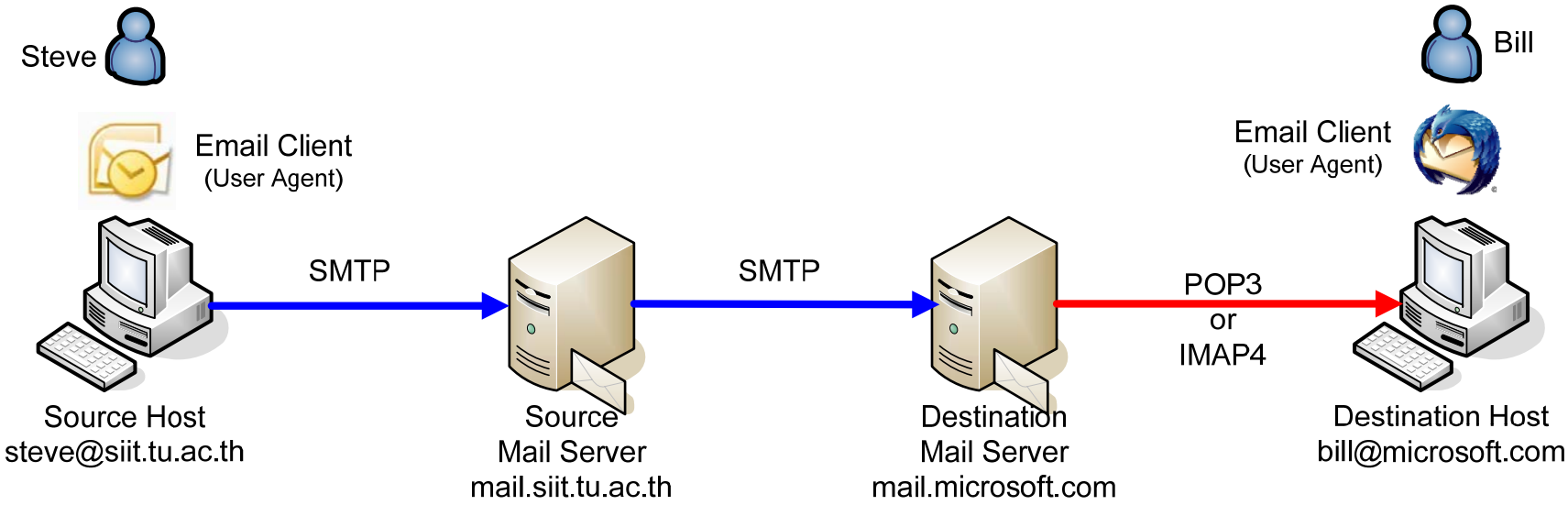
Hex	Dec	Char	Hex	Dec	Char	Hex	Dec	Char	Hex	Dec	Char
0	0	Null	20	32	Space	40	64	@	60	96	`
1	1	Start of heading	21	33	!	41	65	A	61	97	a
2	2	Start of text	22	34	“	42	66	B	62	98	b
3	3	End of text	23	35	#	43	67	C	63	99	c
4	4	End trans.	24	36	$	44	68	D	64	100	d
5	5	Enquiry	25	37	%	45	69	E	65	101	e
6	6	Ack	26	38	&	46	70	F	66	102	f
7	7	Bell	27	39	‘	47	71	G	67	103	g
8	8	Backspace	28	40	(	48	72	H	68	104	h
9	9	Horiz. Tab	29	41	)	49	73	I	69	105	I
A	10	Line feed	2A	42	*	4A	74	J	6A	106	j
B	11	Vert. Tab	2B	43	+	4B	75	K	6B	107	k
C	12	New page	2C	44	,	4C	76	L	6C	108	l
D	13	Carriage ret.	2D	45	-	4D	77	M	6D	109	m
E	14	Shift out	2E	46	.	4E	78	N	6E	110	n
F	15	Shift in	2F	47	/	4F	79	O	6F	111	o
10	16	Data link escape	30	48	0	50	80	P	70	112	p
11	17	Device control 1	31	49	1	51	81	Q	71	113	q
12	18	Device control 2	32	50	2	52	82	R	72	114	r
13	19	Device control 3	33	51	3	53	83	S	73	115	s
14	20	Device control 4	34	52	4	54	84	T	74	116	t
15	21	Neg. ACK	35	53	5	55	85	U	75	117	u
16	22	Synch idle	36	54	6	56	86	V	76	118	v
17	23	End trans. block	37	55	7	57	87	W	77	119	w
18	24	Cancel	38	56	8	58	88	X	78	120	x
19	25	End of medium	39	57	9	59	89	Y	79	121	y
1A	26	Substitute	3A	58	:	5A	90	Z	7A	122	z
1B	27	Escape	3B	59	;	5B	91	[	7B	123	{
1C	28	Field separator	3C	60	<	5C	92	\	7C	124	|
1D	29	Group Sep.	3D	61	=	5D	93	]	7D	125	}
1E	30	Record Sep.	3E	62	>	5E	94	^	7E	126	~
1F	31	Unit Sep.	3F	63	?	5F	95	_	7F	127	Delete

Multipurpose Internet Mail Extensions

- Motivation:
  - Email messages sent using SMTP contain only 7-bit ASCII characters
- MIME was introduced to support:
  - Message bodies and headers encoded with different character sets
  - Non-text attachments
  - Message bodies with multiple parts
  - New headers fields, including:
    - Content-Transfer-Encoding: specifies how the message is encoded
    - Content-Type: specifies the type of media in format type/subtype
- MIME and Content-Types are now used by applications other than email (Web access, databases, instant messaging, …)
  - Example types: text/plain, text/html, text/javascript, image/gif, image/jpeg, audio/mpeg, audio/x-wav, video/mp4, video/quicktime, application/zip, application/msword, …

Mail Transfer

- Email Transfer is “Store-and-forward” operation
  - Email Client (User Agent) sends email to Source Mail Server (Mail Transfer Agent) using SMTP
  - Email is stored at Source Mail Server
  - Source Mail Server sends email to Destination Mail server using SMTP
  - Email is stored at Destination Mail Server
  - Destination Email Client retrieves email from Destination Mail Server



Protocols using in Email Transfer

- SMTP
  - SMTP Server listens on port 25; communication using TCP
  - SMTP Client connects to server and follows SMTP rules to transfer email
    - Client tells Server who the email is from, who it is to and sends the email (headers + message body)
- DNS
  - A special use of DNS is recording mail servers. For example:
    - Source Mail Server has an email to forward to bill@microsoft.com
    - Normally, the mail server for a domain will have its own hostname, e.g. mail.microsoft.com
    - Source Mail Server uses DNS to find the IP address for the *Mail Server* at microsoft.com: DNS returns mail.microsoft.com (IP 130.107.1.71), which was registered as the Mail Server for domain microsoft.com
- POP3 (or IMAP4):
  - Normally, users do not have their own mail server on their computer (since mail server must be running all the time)
  - Destination User Agent retrieves emails from local Mail Server on demand
    - E.g. when you start your email client, or press “Get Mail”, or every 5 minutes
  - POP3 generally used to download emails from server, deleting from server
  - IMAP4 generally used to download emails from server, leaving copies on server