University of Glasgow

Session Layer and DNS

Networked Systems 3
Lecture 16

Lecture Outline

- Higher layer protocols
- The session layer
  - Managing connections
  - Middleboxes and caches
  - Naming users and resources
- The domain name system (DNS)

Higher Layer Protocols

- The OSI reference model defines three layers above the transport layer:
  - Session layer
  - Presentation layer
  - Application layer
- All typically implemented within an application or library; poorly-defined boundaries between layers

Function of the Higher Layers

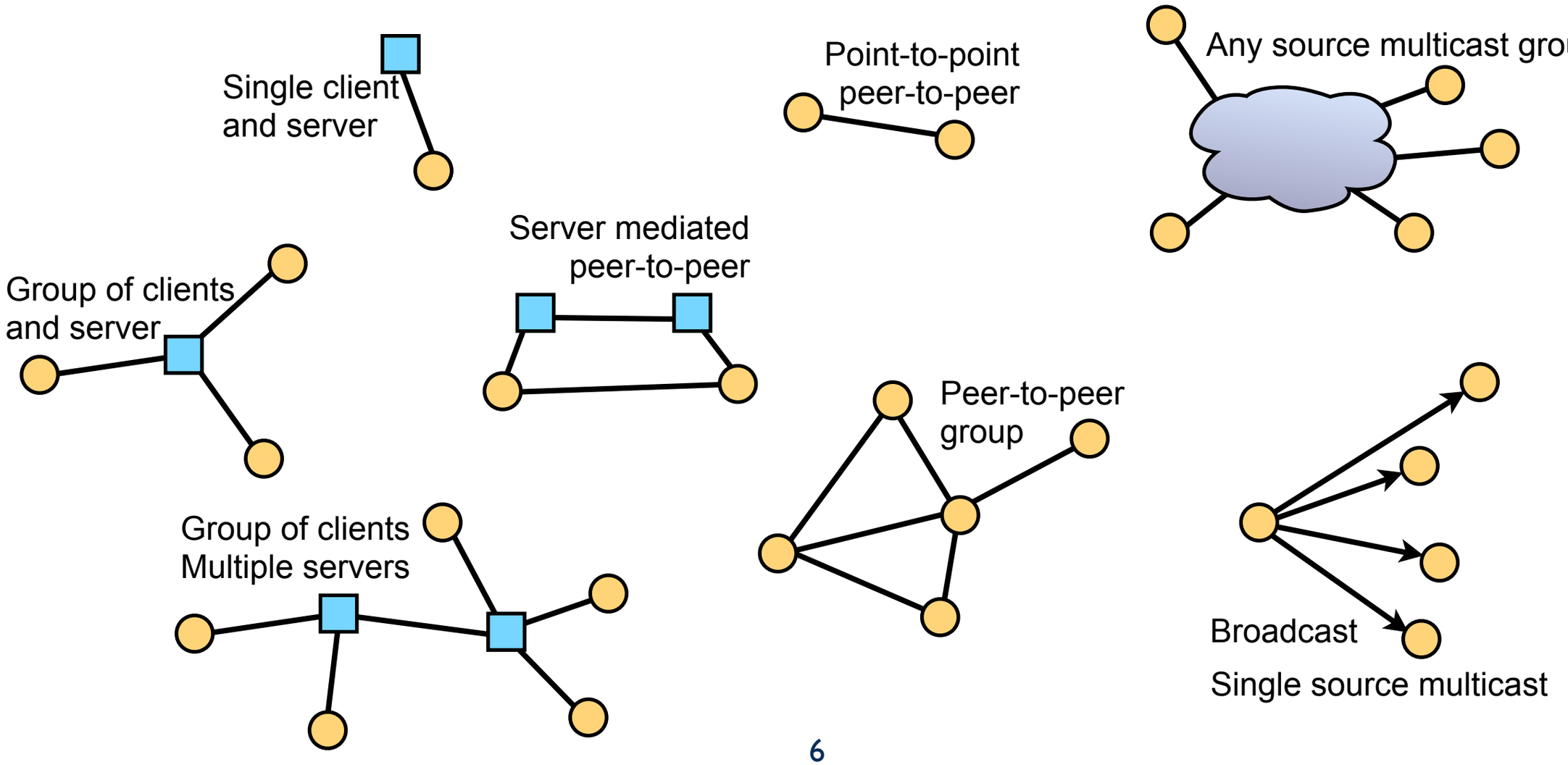
- To support the needs of the application:
  - Setup and manage transport layer connections
  - Name and locate application-level resources
  - Negotiate supported data formats, performing format conversion as needed
  - Present data in an appropriate manner
  - To implement application semantics

The Session Layer

- Responsible for managing connections:
  - Find users/resources; create transport connections
  - Middleboxes and caches
- Responsible for naming resources:
  - Uniform resource identifiers
  - The Domain Name System (DNS)

Managing Connections

- What connections does the application need?



Managing Connections

- How to find participants?
  - Look-up name in a directory (e.g. DNS, web search engine)
  - Server mediated connection (e.g. instant messenger, VoIP call)
- How to setup connections?
  - Direct connection to named host (→ NAT issues)
  - Mediated service discovery, followed by peer-to-peer connection
    - E.g. VoIP using SIP and RTP with ICE
- How does session membership change?
  - Does the group size vary greatly? How rapidly do participants join and leave? Are all participants aware of other group members?

User and Resource Mobility

- IP addresses encode location → mobility breaks transport layer connections
- Session layer must find new location, establish new connections
  - Old location might redirect – e.g., HTTP
  - Users might register new location
    - Updating a DNS name to point to the new IP address
    - Via an application-specific server – e.g., SIP proxy for VoIP calls

Example: HTTP redirect

HTTP request

```
GET /index.html HTTP/1.1
Host: www.google.com
```

HTTP response

```
HTTP/1.1 302 Moved Temporarily
Location: http://www.google.co.uk/index.html
Cache-Control: private
Content-Type: text/html
Server: gws
Content-Length: 231
Date: Sun, 17 Feb 2008 23:23:30 GMT

<HTML><HEAD><meta http-equiv="content-type" content="text/html;charset=utf-8">
<TITLE>302 Moved</TITLE></HEAD><BODY>
<H1>302 Moved</H1>
The document has moved
<A HREF="http://www.google.co.uk/index.html">here</A>.
</BODY></HTML>
```

302 response code indicates the content has moved, the “Location:” header specifies where it’s moved to.

Multiple Connections

- A single session may span multiple transport connections
  - E.g., retrieving a web page containing images – one connection for the page, then one per image
  - E.g., a peer-to-peer file sharing application, building a distributed hash table
- Session layer responsible for co-ordinating the connections

Middleboxes and Caches

- Some protocols rely on middleboxes or caches
  - Web cache – optimise performance, moving popular content closer to hosts
  - Email server – supports disconnected operation by holding mail until user connects
  - SIP proxy servers and instant messaging servers – locate users, respond for offline users
- The end-to-end argument applies, once again
  - Only add middleboxes when absolutely necessary

How to Find the Middlebox?

- Manual configuration
- Look-up in central directory service
  - E.g., DNS MX records to find email servers
- Multicast service discovery
- “Transparent” redirection
  - E.g., Wi-Fi hotspots that grab web traffic, and redirect to a payment server

Naming

- How to identify resources used or referenced by an application?
  - Files, email addresses, phone numbers, objects in a database, books, parcels being shipped, etc.
  - Use a *uniform resource identifier*
    - Uniform resource name (URN) – a unique resource name; no information on where to find, or how to access, the resource
    - Uniform resource locator (URL) – a unique resource name, plus location and access method
    - Directory service used for URN → URL mapping

Uniform Resource Identifier

A general mechanism for naming arbitrary resources

scheme:authority/path?query#fragment (authority, query and fragment optional)

ftp://ftp.is.co.za/rfc/rfc1808.txt

http://news.bbc.co.uk/2/hi/europe/7249034.stm#map

ldap://[2001:db8::7]/c=GB?objectClass=one

mailto:John.Doe@example.com

news:comp.infosystems.www.servers.unix

tel:+1-816-555-1212

telnet://192.0.2.16:80/

urn:oasis:names:specification:docbook:dtd:xml:4.1.2

Syntax is extremely flexible

Wide range of schemes defined

Some can be directly accessed, others require a look-up to map from the URI to a URL

Domain Name System

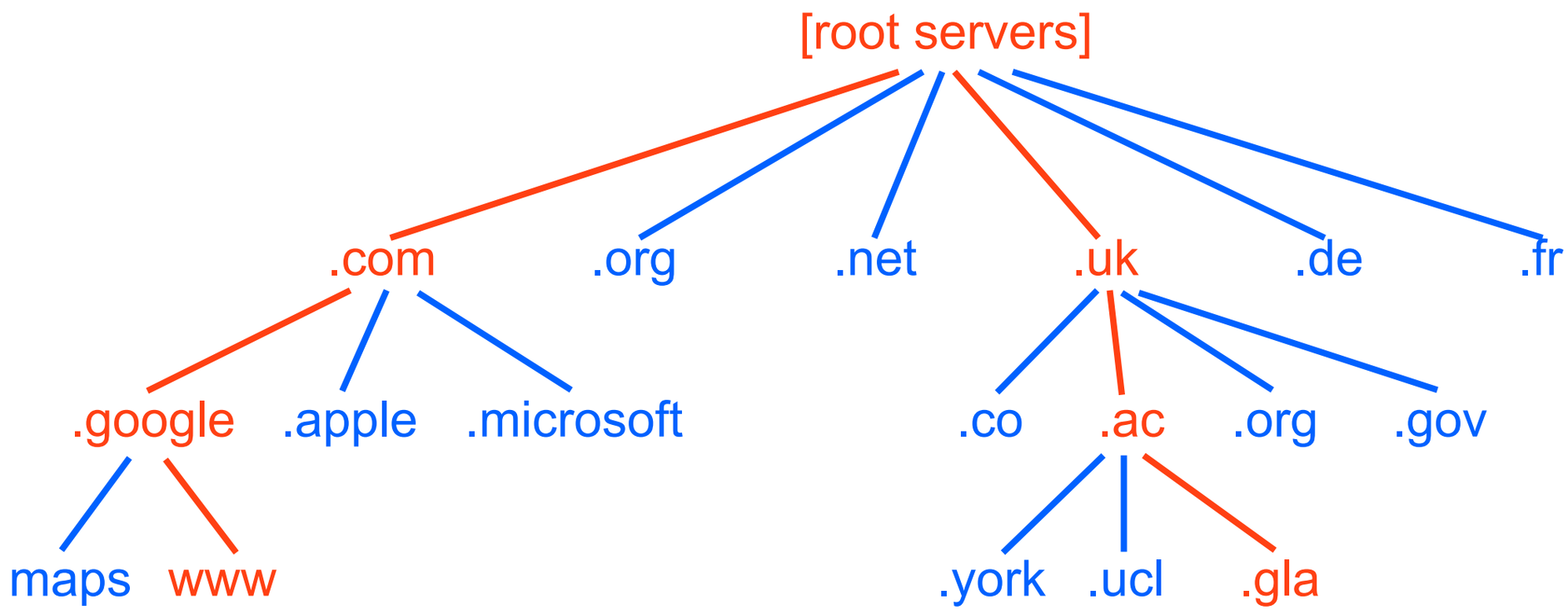
- URIs often refer to a host on the network
  - Want to use a human-readable hostname in URIs, rather than an IP address
  - The *domain name system* (DNS) translates from the hostname to an IP address
    - www.dcs.gla.ac.uk → 130.209.240.1
    - DNS is an application layer protocol, running over the network
    - Not necessary for the correct operation of the transport or network layers, or lower

History of the DNS

- Early Internet didn't use DNS
  - Flat file `hosts.txt` listing all host names and addresses
  - Maintained by central NIC; updated by email every few days; manually installed in hosts
- DNS proposed in 1983 as distributed database of host names
  - Solve scaling problems with `hosts.txt`

Paul Mockapetris

Operation of the DNS



- Hierarchy of DNS zones
- One logical server per zone
- Delegation follows hierarchy
- Hop-by-hop name look-up, follows hierarchy via root
- Results have TTL, cached at intermediate servers
- getaddrinfo()

Contents of a DNS Zone

```
$TTL 3600          ; 1 hour
example.org.     IN       SOA      ns1.example.org. admin.example.org. (
                                    2006051501       ; Serial
                                    10800            ; Refresh
                                    3600             ; Retry
                                    604800           ; Expire
                                    86400            ; Minimum TTL
                                   )
; DNS Servers
                 IN       NS       ns1.example.org.
                 IN       NS       ns2.example.org.
; MX Records
                 IN       MX 10    mx.example.org.
                 IN       MX 20    mail.example.org.
; Machine Names
ns1              IN       A        192.168.1.2
ns2              IN       A        192.168.1.3
mx               IN       A        192.168.1.4
mail             IN       A        192.168.1.5
mail             IN       AAAA     2001:200:1000:0:25f:23ff:fe80:1234
server1          IN       A        192.168.1.10
server2          IN       A        192.168.1.11
; Aliases
www              IN       CNAME    server1
```

Source: adapted from The FreeBSD Handbook

How to Perform DNS Lookups

- Prefer using DNS names to raw IP addresses
  - Use `getaddrinfo()` to look-up name in DNS
  - Returns a linked list of `struct addrinfo` values, representing addresses of the host:

```
struct addrinfo {
    int              ai_flags;     // input flags
    int              ai_family;    // AF_INET, AF_INET6, ...
    int              ai_socktype;  // IPPROTO_TCP, IPPROTO_UDP
    int              ai_protocol;  // SOCK_STREAM, SOCK_DRAM, ...
    socklen_t        ai_addrlen;   // length of socket-address
    struct sockaddr *ai_addr;      // socket-address for socket
    char            *ai_canonname; // canonical name of host
    struct addrinfo *ai_next;      // pointer to next in list
};
```

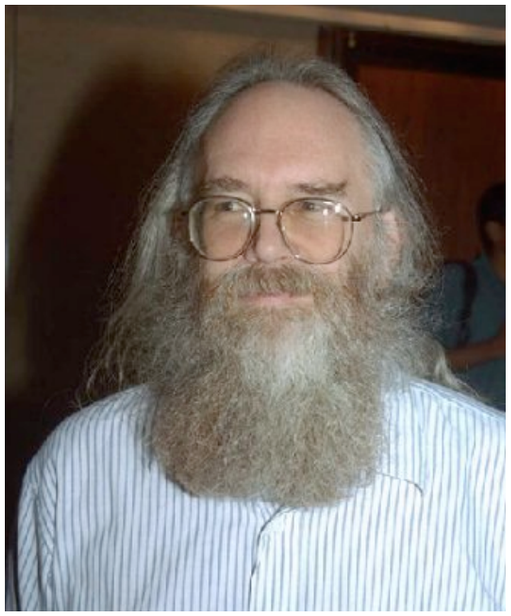
(`#include <netdb.h>` for definition of `struct addrinfo`)

Connecting via a DNS Query

```
struct addrinfo   hints, *ai, *ai0;
int i;

memset(&hints, 0, sizeof(hints));
hints.ai_family    = PF_UNSPEC;
hints.ai_socktype  = SOCK_STREAM;
if ((i = getaddrinfo(“www.google.com”, "80", &hints, &ai0)) != 0) {
    printf("Unable to look up IP address: %s", gai_strerror(i));
    ...
}

for (ai = ai0; ai != NULL; ai = ai->ai_next) {
    fd = socket(ai->ai_family, ai->ai_socktype, ai->ai_protocol);
    if (fd == -1) {
        perror("Unable to create socket");
        continue;
    }

    if (connect(fd, ai->ai_addr, ai->ai_addrlen) == -1) {
        perror("Unable to connect");
        close(fd);
        continue;
    }
    ...success, use the connection
    break;
}
if (ai == NULL) {
    // Connection failed, handle the failure...
}
```

DNS Politics

- The DNS was administered by IANA
  - Jon Postel *was* IANA from its creation until his death in 1998
  - http://www.ietf.org/rfc/rfc2468.txt – “I remember IANA”
- DNS now managed into ICANN
  - The US government asserts ultimate control over ICANN, and hence the DNS
  - Significant attempts to move control of national domains to the UN, and hence to the countries concerned
  - Other attempts to set up alternate roots for the DNS, with different namespaces → significant technical problems



Jon Postel

Summary

- Higher layer protocols
- The session layer
  - Managing connections
  - Middleboxes and caches
  - Naming users and resources
- The domain name system (DNS)
  - Performing DNS lookups
  - Politics