

CSCI 491-01

Topics: Internet Programming

Fall 2008

Application Layer

Derek Leonard

Hendrix College

September 19, 2008

Original slides copyright © 1996-2007 J.F Kurose and K.W. Ross

Chapter 2: Roadmap

2.1 Principles of network applications

2.2 Web and HTTP

2.3 FTP

2.4 Electronic Mail

- SMTP, POP3, IMAP

2.5 DNS

2.6 P2P file sharing

2.7 Socket programming with TCP

2.8 Socket programming with UDP

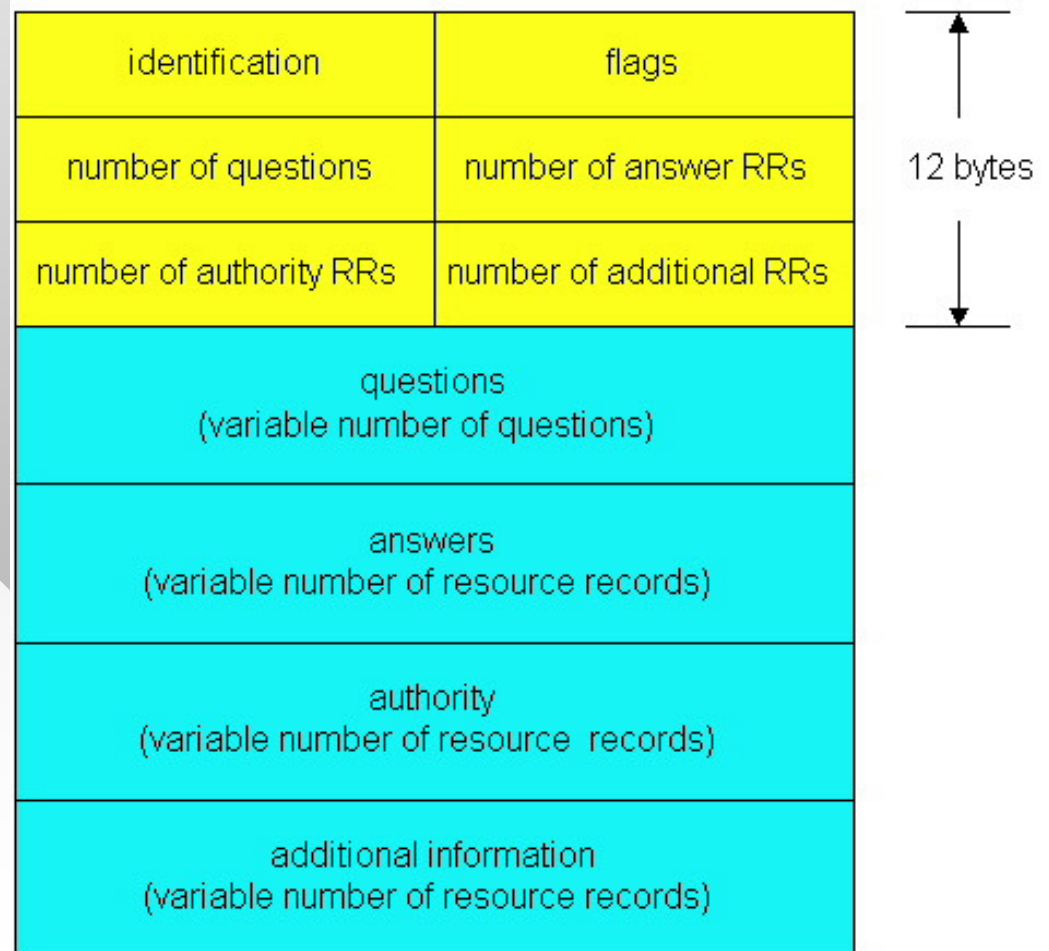
2.9 Building a Web server

DNS Protocol, Messages

DNS protocol : *query* and *reply* messages, both with same *message format*

Message header

- **Identification:** 16 bit # for query, reply to query uses same #
- **Flags:**
 - Query or reply
 - Recursion desired
 - Recursion available
 - Reply is authoritative



DNS Protocol, Messages

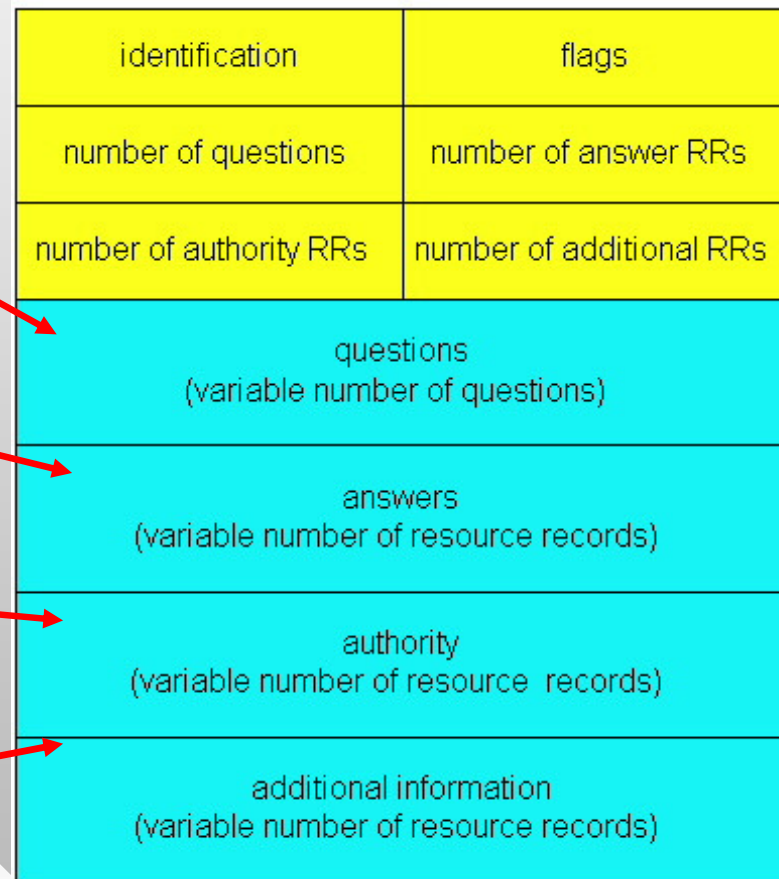
www.hendrix.edu A, PTR, MX

Name, type fields
for a query

RRs in response
to query

SOA and NS records
from authoritative server

Additional "helpful"
RRs that may be used



↑
12 bytes
↓

Do not forget that all numbers are in network byte order (MSB); use htons() for all 2-byte numbers

DNS Flags



Format of flags (# of bits):

QR(1): 0=query, 1=response

Opcode(4): 0=query

AA(1): 1=authoritative answer

TC(1): 1=truncated response

RD(1): 1=recursion desired

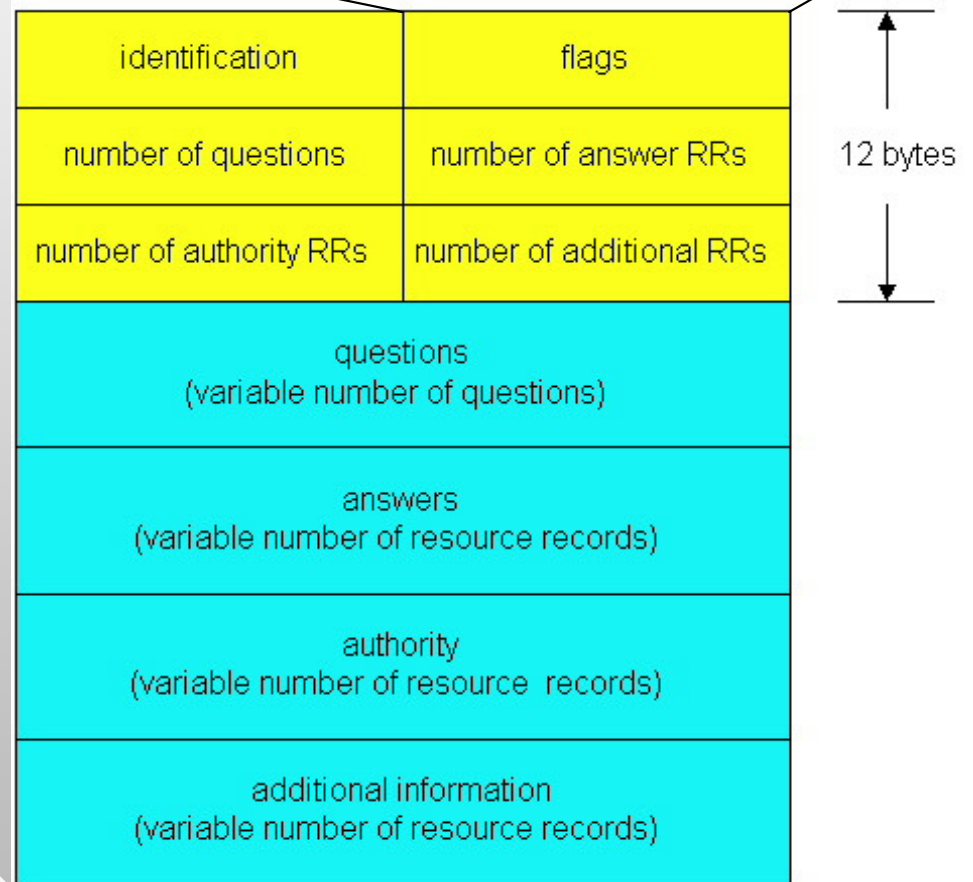
RA(1): 1=recursion available

Rcode(4): 0 = no error

1 = format error

2 = server failure

3 = no DNS name



See <http://www.networksorcery.com/enp/protocol/dns.htm>

Inserting Records Into DNS

- Example: just created startup “Network Utopia”
- Register name networkutopia.com at a registrar (e.g., Network Solutions)
 - Need to provide registrar with names and IP addresses of your authoritative name server (primary and secondary)
 - Registrar inserts four RRs into the `com` TLD server:
 - `(networkutopia.com, dns1.networkutopia.com, NS)`
 - `(dns1.networkutopia.com, 212.212.212.1, A)`
 - `(networkutopia.com, dns2.networkutopia.com, NS)`
 - `(dns2.networkutopia.com, 212.212.212.2, A)`
- Put in authoritative server Type A record for `www.networkutopia.com` and Type MX record for domain `networkutopia.com`

Chapter 2: Roadmap

2.1 Principles of network applications

2.2 Web and HTTP

2.3 FTP

2.4 Electronic Mail

- SMTP, POP3, IMAP

2.5 DNS

2.6 P2P file sharing

2.7 Socket programming with TCP

2.8 Socket programming with UDP

2.9 Building a Web server

P2P File Sharing

Operation

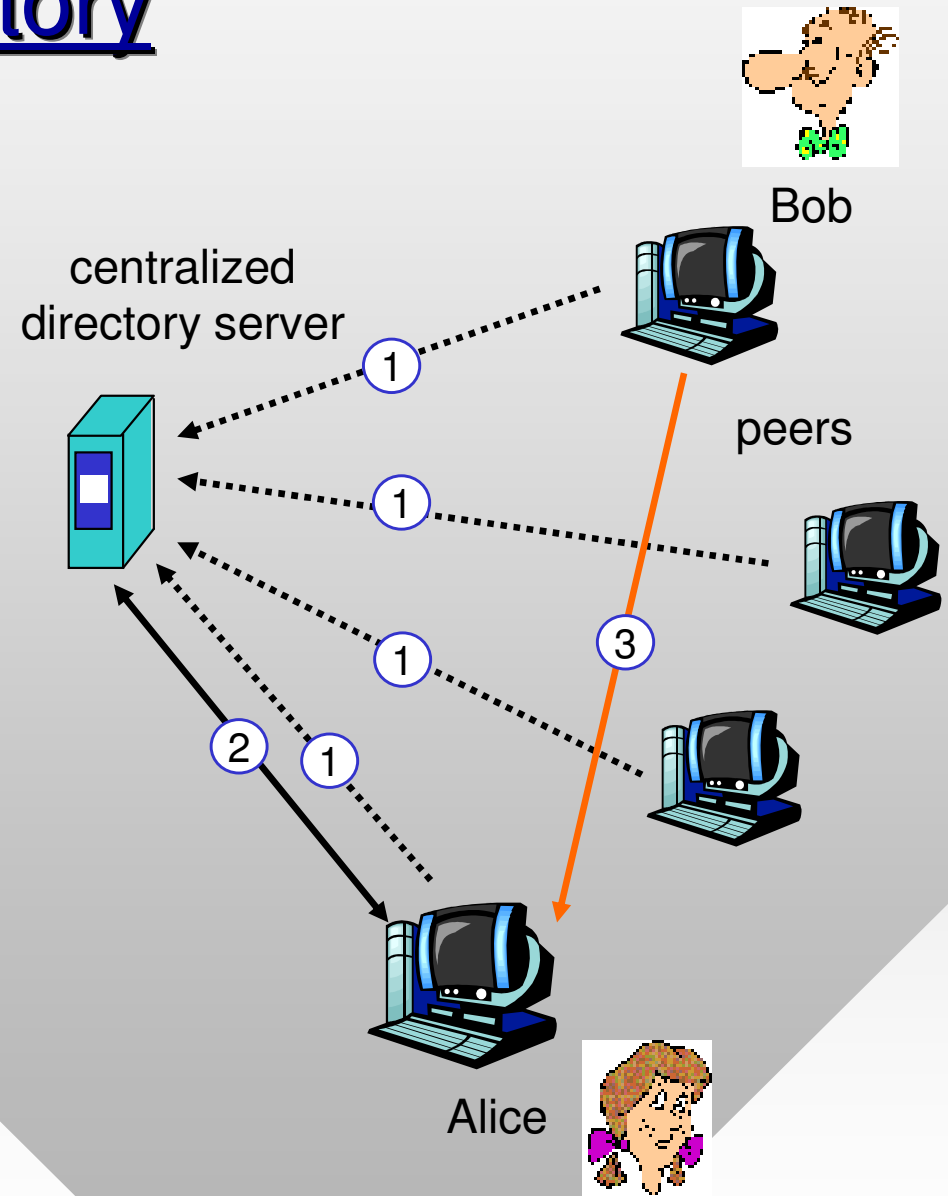
- Alice runs P2P client application on her notebook computer
- Intermittently connects to Internet; gets new IP address for each connection
- Asks for “Hey Jude”
- Application displays other peers that have copy of Hey Jude
- Alice chooses one of the peers, Bob
- File is copied from Bob’s PC to Alice’s notebook
- While Alice downloads, other users upload from Alice
- Alice’s peer is both a client and a transient server

All peers are servers = highly scalable!

P2P: Centralized Directory

Original “Napster” design

- 1) When peer connects, it informs central server:
 - Its [IP:port] combination
 - List of shared files
- 2) Alice queries for “Hey Jude” and receives a list of matches including Bob
- 3) Alice requests file directly from Bob



P2P: Problems With Centralized Directory

- Single point of failure
- Performance bottleneck
- Copyright infringement

File transfer is decentralized, but search is highly centralized

Query Flooding: Original Gnutella

- Fully distributed
 - No central server
- Public domain protocol
- Many Gnutella clients implementing protocol
- Problem: how to find content?

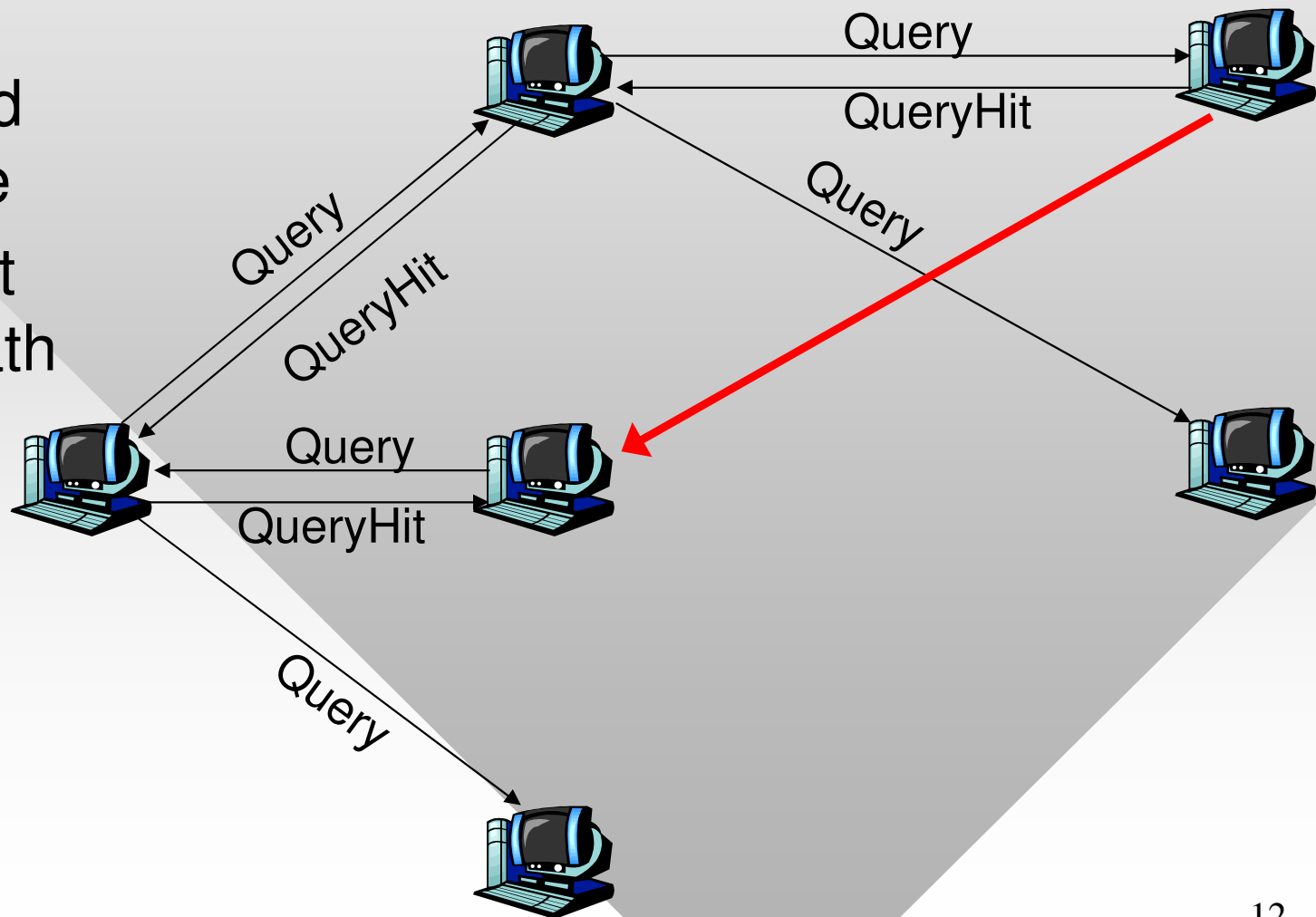
Construct a graph:

- Edge between peer X and Y if there's a TCP connection
- All active peers and edges is **overlay network**
- Edge is not a physical link
- A peer will typically be connected with < 30 overlay neighbors

Original Gnutella: Protocol

- ❑ Query message sent over existing TCP connections
- ❑ Peers forward Query message
- ❑ QueryHit sent over reverse path

File transfer:
HTTP



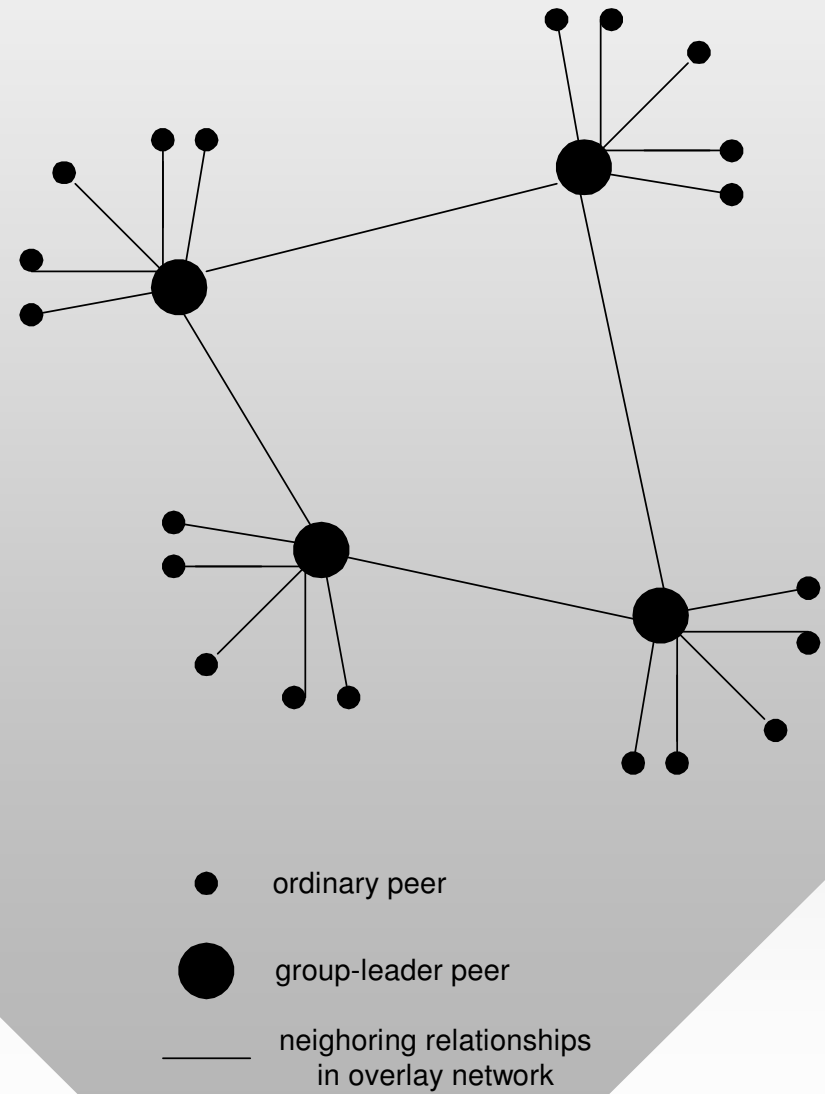
Scalability:
Limited-scope
flooding

Gnutella: Peer Joining

1. Joining peer X must find some other peer in Gnutella network: use list of candidate peers
2. X sequentially attempts to make TCP with peers on list until connection setup with Y
3. X sends ping message to Y; Y forwards ping message
4. All peers receiving ping message respond with pong message
5. X receives many pong messages. It can then setup additional TCP connections

Exploiting Heterogeneity: Kazaa

- Each peer is either a group leader or assigned to a group leader
 - TCP connection between peer and its group leader
 - TCP connections between some pairs of group leaders
- Group leader tracks the content in all its children



Kazaa: Querying

- Client sends keyword query to its group leader
- Group leader responds with matches:
 - For each match: **metadata, hash, IP address**
- If group leader forwards query to other group leaders, they respond with matches
- Client then selects files for downloading
 - **HTTP requests using hash as identifier sent to peers holding desired file**

Other Algorithms

- Limitations on simultaneous uploads and request queuing
 - Rate-limiting based on how many parallel request a user wishes to satisfy
- Incentive priorities
 - Peers who share files are given more priority
- Parallel downloading
 - Chunks of a file are loaded simultaneously from multiple alternative peers
 - Hash index is used to verify that the files are identical