

CSCI 491-01

Topics: Internet Programming

Fall 2008

## Network Layer

Derek Leonard

Hendrix College

November 7, 2008

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

# Chapter 4: Roadmap

4.1 Introduction

4.2 Virtual circuit and datagram networks

4.3 What's inside a router

4.4 IP: Internet Protocol

- Datagram format
- IPv4 addressing
- ICMP
- **IPv6**

4.5 Routing algorithms

4.6 Routing in the Internet

4.7 Broadcast and multicast routing

# IPv6

16-byte IP, e.g.,

FEBC:A574:382B:23C1:AA49:4592:4EFE:9982

- **Initial motivation:** 32-bit address space soon to be completely allocated
- Additional motivation:
  - Simpler header format helps speed up forwarding
  - Header changes to facilitate QoS and extensions

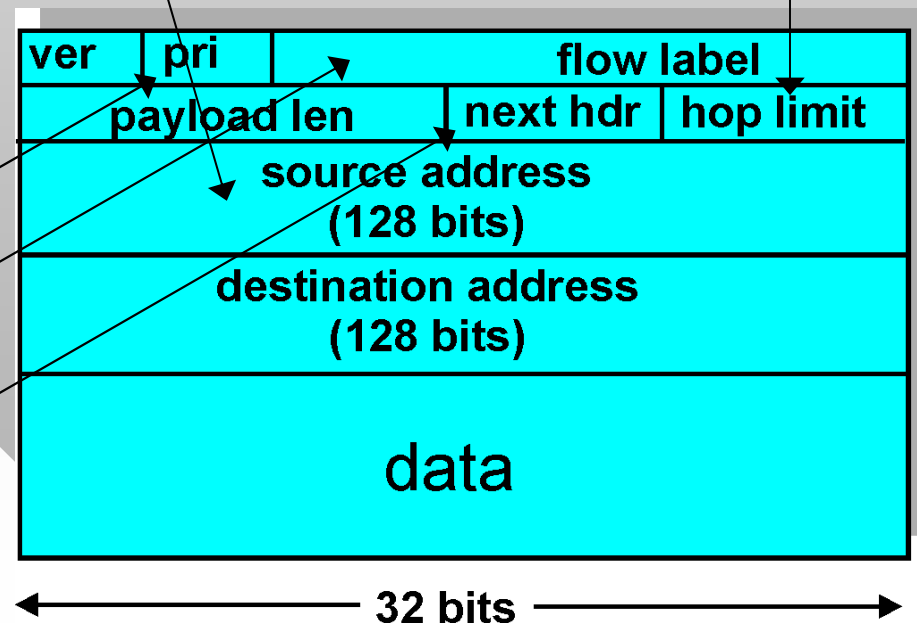
## IPv6 datagram format:

- Fixed-length 40 byte header
- No fragmentation allowed

priority of packet (QoS)

flow ID (not well defined)

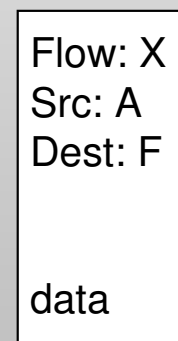
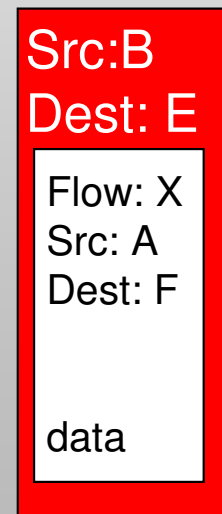
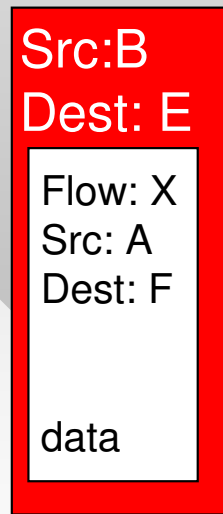
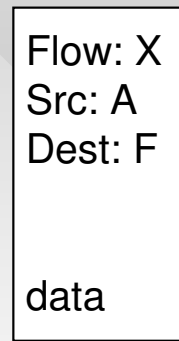
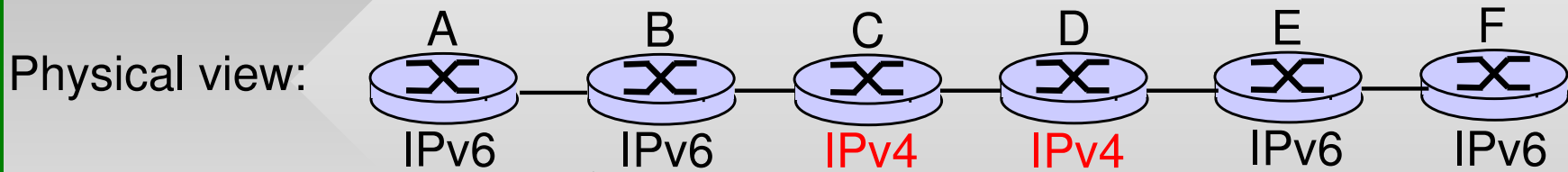
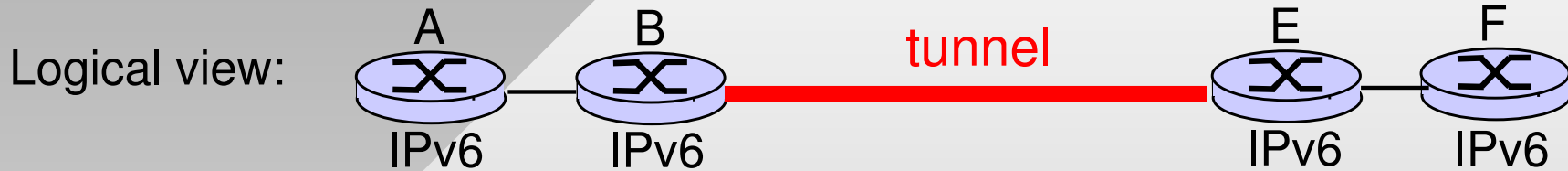
upper-layer protocol  
(e.g., TCP, ICMP)



# IPv6 Notes

- *Checksum*: removed entirely to reduce processing time at each hop
  - IPv4 checksums the header only
- *Options*: allowed, but outside of header, indicated by “Next Header” field
- All routers cannot be upgraded simultaneously
  - How will the network operate with mixed IPv4 and IPv6 routers?
- *Tunneling*: IPv6 carried as payload in IPv4 datagram among IPv4 routers

# Tunneling



A-to-B:  
IPv6

B-to-C:  
IPv6 inside  
IPv4

D-to-E:  
IPv6 inside  
IPv4

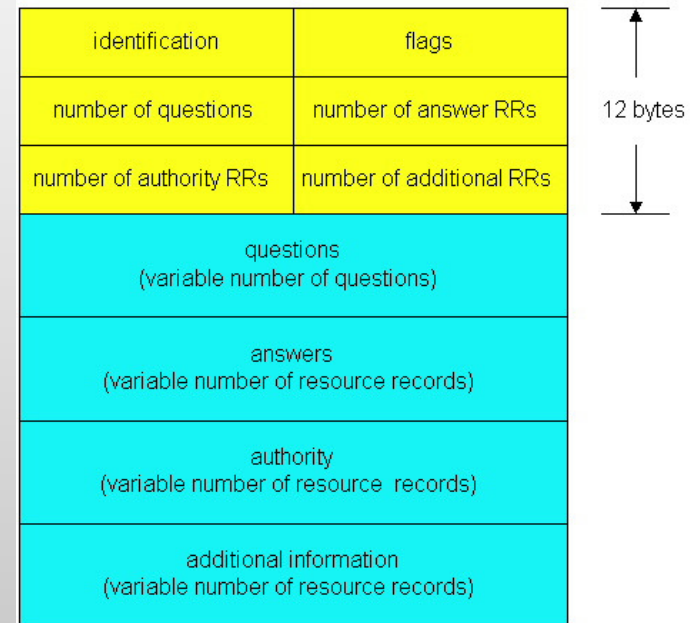
E-to-F:  
IPv6

Q: how does E know the packet has encapsulated IPv6 data?

A: protocol field (often 41)

# Homework #2

- Query your local DNS server
  - Get IP address from /etc/resolv.conf
- Create classes for fixed headers
  - Fill in the values (flags: DNS\_QUERY and DNS\_RD)
  - Allocate memory for the packet
  - Memcpy the various fields into the packet



```
class fixedDNSheader {
    u_short ID;
    u_short flags;
    u_short questions;
    ...
};
```

```
class queryHeader {
    u_short class;
    u_short type;
};
```

## Homework #2

- Pseudocode:

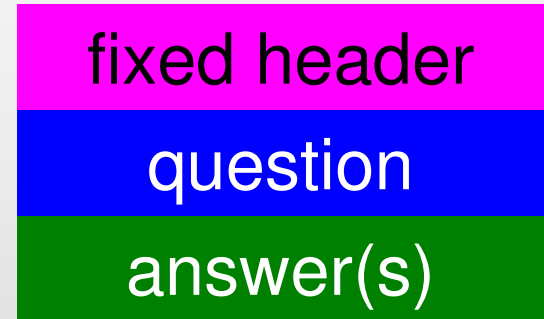
```
fixedDNSheader dns_header;
queryHeader query_header;
... // fixed field initialization

Question q;           // your class
string host = "www.google.com";

q.CreateQuestion(host);
// q.rawQuerybuffer now holds a raw request

int size = q.size() + sizeof (fixedDNSheader) +
           sizeof(queryHeader);
u_char *pkt = new u_char [size];
q.MakePacket (pkt, &dns_header, &query_header);
sendto (sock, pkt, ...);
delete pkt; q.freeRawbuffer();
```

## Homework #2



- Formation of questions:

```
u_char *rawbuffer = new char[...];  
while(words left to copy){  
    rawbuffer[i++] = size_of_next_word;  
    memcpy (rawbuffer+i, next_word, size_of_next_word);  
    i += size_of_next_word;  
}  
rawbuffer[i] = 0; // last word NULL-terminated
```

- All answers start with an RR name followed by a fixed DNS reply header, followed by the answer

- Uncompressed answer

```
0x5 "ozark" 0x7 "hendrix" 0x3 "edu" 0x00  
<DNS_REPLY_HEADER> <ANSWER>
```

- Compressed

```
0xC0 0x0C <DNS_REPLY_HEADER> <ANSWER>
```

- For A-type questions, the answer is a 4-byte IP

## Homework #2

- To check the header
  - Hex printout on screen
  - Use Wireshark
- Notice that `sizeof(DNSAnswer)` does not return the correct size
  - The actual size is 10 bytes, but most structures in C/C++ are aligned/padded to a 4 or 8 byte boundary
- Remember to disable the default alignment in C++:

```
class DNSAnswer {  
    u_short type;  
    u_short class;  
    u_int ttl;  
    u_short len;  
};
```

```
#pragma pack(push)  
#pragma pack(1)  
//      define class here  
#pragma pack(pop)
```