CSC465 – Computer Networks
Spring 2004
Dr. J. Harrison

These slides were produced almost entirely from material by Behrouz Forouzan for the text “TCP/IP Protocol Suite (2nd Edition)”, McGraw Hill Publisher

Chapter 8

Internet Protocol (IP)

CONTENTS
- DATAGRAM
- FRAGMENTATION
- OPTIONS
- CHECKSUM
- IP PACKAGE

Position of IP in TCP/IP protocol suite

IP
- Unreliable
- Connectionless
- Datagrams can follow different paths
- Can arrive out of order or be lost

8.1 DATAGRAM
IP datagram

**IP**
- Version (VER): defines IP version 4-bits = 4
- Header length (HLEN): datagram header
  - In 4-byte words; range is 5 to 15 (5 if no options)
- Differentiated Services (8-bit field)
  - Old interpretation is “Service Type”
  - First 3-bits Precedence: not used
  - Bit 4: D - Minimize delays
  - Bit 5: T – Maximize throughput
  - Bit 6: R – Maximize reliability
  - Bit 7: C – Minimize cost
  - Bit 8: Not used

<table>
<thead>
<tr>
<th>Protocol</th>
<th>TOS Bits</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ICMP</td>
<td>0000</td>
<td>Normal</td>
</tr>
<tr>
<td>SNMP</td>
<td>0010</td>
<td>Maximize Reliability</td>
</tr>
<tr>
<td>FTP (Data)</td>
<td>0100</td>
<td>Maximize Throughput</td>
</tr>
<tr>
<td>FTP (Command)</td>
<td>1000</td>
<td>Minimize delay</td>
</tr>
<tr>
<td>DNS (UDP query)</td>
<td>1000</td>
<td>Minimize delay</td>
</tr>
<tr>
<td>DNS (TCP query)</td>
<td>0000</td>
<td>Normal</td>
</tr>
<tr>
<td>Telnet</td>
<td>1000</td>
<td>Minimize delay</td>
</tr>
<tr>
<td>Interior Gateway</td>
<td>0010</td>
<td>Maximize reliability</td>
</tr>
</tbody>
</table>

**Codepoint Values**

<table>
<thead>
<tr>
<th>Category</th>
<th>Codepoint</th>
<th>Assigning Authority</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>XXXXX0</td>
<td>Internet</td>
</tr>
<tr>
<td>2</td>
<td>XXXX11</td>
<td>Local</td>
</tr>
<tr>
<td>3</td>
<td>XXXX01</td>
<td>Temp/Experimental</td>
</tr>
</tbody>
</table>

**Total Length**
- 16-bit value with length of header plus data
- IP Datagram limited to 64K bytes
  - Some networks can’t encapsulate a 64K datagram into their frames
  - Datagram must be fragmented
- Length field not normally needed
  - Ethernet frame holds data sizes 46 to 1500 bytes
  - If IP datagram is < 46 bytes padding will be added to meet minimum requirement of DL layer
  - Receiving IP layer needs to know breakdown
Encapsulation of a small datagram in an Ethernet frame

Time to Live

- Constrains “lifetime” of packet
- Stops “run-a-way” packets due to corrupt router tables
  - Protects network resources
  - Avoids confusing TCP
- Field decremented by routers
- Can limit packet journey
  - Set to one to keep traffic on LAN

Protocol

- 8-bits
- Defines higher-level protocol using IP layer
- Field specifies final destination protocol for delivery
- Helps in the IP demultiplexing process

Higher-level Protocol Field Values

<table>
<thead>
<tr>
<th>Value</th>
<th>Protocol</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>ICMP</td>
</tr>
<tr>
<td>2</td>
<td>IGMP</td>
</tr>
<tr>
<td>6</td>
<td>TCP</td>
</tr>
<tr>
<td>17</td>
<td>UDP</td>
</tr>
<tr>
<td>89</td>
<td>Open Shortest-Path First (OSPF) Routing</td>
</tr>
</tbody>
</table>

Example 1

An IP packet has arrived with the first 8 bits as: 01000010.
The receiver discards the packet. Why?

An IP packet has arrived with the first 8 bits as shown:

.readValue(0x01000010)
The receiver discards the packet. Why?

There is an error in this packet. The 4 left-most bits (0100) show the version, which is correct.
The next 4 bits (0010) show the header length, which means (2 \times 4 \text{ byte words} = 8 \text{ bytes}), which is wrong. The minimum number of bytes in the header must be 20. The packet has been corrupted in transmission.
Example 2

In an IP packet, the value of HLEN is 1000 in binary. How many bytes of options are being carried by this packet?

Solution

In an IP packet, the value of HLEN is 1000 in binary. How many bytes of options are being carried by this packet?

The HLEN value is 8, which means the total number of bytes in the header is $8 \times 4$ byte words or 32 bytes.

The first 20 bytes are the main header, the next 12 bytes are the options.

Example 3

In an IP packet, the value of HLEN is $5_{16}$ and the value of the total length field is $0028_{16}$. How many bytes of data are being carried by this packet?

Solution

In an IP packet, the value of HLEN is $5_{16}$ and the value of the total length field is $0028_{16}$. How many bytes of data are being carried by this packet?

The HLEN value is 5, which means the total number of bytes in the header is $5 \times 4$ or 20 bytes (no options). The total length is 40 bytes, which means the packet is carrying 20 bytes of data (40−20).

Example 4

An IP packet arrives with the first few hexadecimal digits as shown below:

\[
\text{45000028000100000102} \cdots \cdots
\]

How many hops can this packet travel before being dropped? The data belong to what upper layer protocol?

Solution

An IP packet has arrived with the first few hexadecimal digits as shown below:

\[
\text{45000028000100000102} \cdots \cdots
\]

How many hops can this packet travel before being dropped? The data belong to what upper layer protocol?

To find the time-to-live field, we should skip 8 bytes (16 hexadecimal digits). The time-to-live field is the ninth byte, which is 01. This means the packet can travel only one hop. The protocol field is the next byte (02), which means that the upper layer protocol is IGMP.
Fragmentation

- Datagram travels through different networks
- Router decapsulates frame, and encapsulates
- Format & Size of frame depends on network
- Each DL layer specifies maximum frame size (Maximum Transfer Unit)

### MTUs for Different Networks

<table>
<thead>
<tr>
<th>Protocols</th>
<th>MTU</th>
</tr>
</thead>
<tbody>
<tr>
<td>HyperChannel</td>
<td>65,536</td>
</tr>
<tr>
<td>Token Ring (16 Mbps)</td>
<td>17,914</td>
</tr>
<tr>
<td>Token Ring (4 Mbps)</td>
<td>4,464</td>
</tr>
<tr>
<td>Fiber Distributed Data Interface</td>
<td>4,352</td>
</tr>
<tr>
<td>Ethernet</td>
<td>1,500</td>
</tr>
<tr>
<td>X.25</td>
<td>576</td>
</tr>
<tr>
<td>PPP</td>
<td>296</td>
</tr>
</tbody>
</table>

### IP Fragmentation

- IP Should be independent of physical network
- IP Maximum Transfer Unit (MTU) = 64K bytes
- Transmission most efficient with DLs with 64K MTUs
- For networks with smaller MTUs, IP packet must be fragmented
- Each fragment has own header (with most fields)
- Fragmented packet may again be fragmented if it encounters a network with a still smaller MTU

### IP Fragmentation (con’t)

- IP packet can be fragmented by any host or any router encountered along the path of travel
- Reassembly of the datagram is performed at the destination (each datagram is independent)
  - Only place we can expect that all IP packets will eventually arrive
- Each fragment need most of header fields
  - Option field may be copied also
- Host or router that fragments must change flags, fragmentation offset and total length (& checksum); rest of fields copied

### Fragmentation Fields

- Identification
  - IP Addresses, ID → Datagram
  - All datagram fragments retain datagram ID
  - Counter maintained to ensure uniqueness
  - ID used to reassemble datagram from fragments
Flag Field

- If “Do Not Fragment” bit set, machines are instructed to drop packet if MTU too large:
  - ICMP error message sent to host
- More Fragments:
  - Indicates that the fragment is not the last

Fragmentation Offset

- 13-bit field shows relative position of fragment w.r.t. entire datagram
- Offset of data in original datagram measured in units of 8-bytes
- 13-bits: $8192 \times 8 = 65,536$ (Max MTU size)
- Forces routers and hosts that fragment to choose the size of each fragment so that the first byte number is divisible by 8

Fragmentation Example

Size = 4020 (includes header)
ID → 14,567

Reassembly Strategy

1. The 1st fragment has an offset field value of 0
2. Divide the length of this fragment by 8. The 2nd fragment has an offset value equal to that result.
3. Divide the total length of the 1st & 2nd fragments by 8. The 3rd fragment has an offset value equal to that result.
4. Continue the process; last fragment has a more bit value of zero

Example 5

A packet has arrived with an $M$ bit value of 0. Is this the first fragment, the last fragment, or a middle fragment? Do we know if the packet was fragmented?
If the $M$ bit is 0, it means that there are no more fragments; the fragment is the last one. However, we cannot say if the original packet was fragmented or not. A non-fragmented packet is considered the last fragment.

Example 6

A packet has arrived with an $M$ bit value of 0. Is this the first fragment, the last fragment, or a middle fragment? Do we know if the packet was fragmented?

Solution

A packet has arrived with an $M$ bit value of 0. Is this the first fragment, the last fragment, or a middle fragment? Do we know if the packet was fragmented?

If the $M$ bit is 0, it means that there are no more fragments; the fragment is the last one. However, we cannot say if the original packet was fragmented or not. A non-fragmented packet is considered the last fragment.

Example 7

A packet has arrived with an $M$ bit value of 1. Is this the first fragment, the last fragment, or a middle fragment? Do we know if the packet was fragmented?

If the $M$ bit is 1, it means that there is at least one more fragment. This fragment can be the first one or a middle one, but not the last one. We don’t know if it is the first one or a middle one; we need more information (the value of the fragmentation offset). However, we can definitely say the original packet has been fragmented because the $M$ bit value is 1.

Example 8

A packet has arrived with an $M$ bit value of 1. Is this the first fragment, the last fragment, or a middle fragment?

Because the $M$ bit is 1, it is either the first fragment or a middle one. Because the offset value is 0, it is the first fragment.

Solution

A packet has arrived with an $M$ bit value of 1 and a fragmentation offset value of zero. Is this the first fragment, the last fragment, or a middle fragment?

A packet has arrived with an $M$ bit value of 1 and a fragmentation offset value of zero. Is this the first fragment, the last fragment, or a middle fragment?

Because the $M$ bit is 1, it is either the first fragment or a middle one. Because the offset value is 0, it is the first fragment.

Example 9

A packet has arrived in which the offset value is 100. What is the number of the first byte? Do we know the number of the last byte?

A packet has arrived in which the offset value is 100. What is the number of the first byte? Do we know the number of the last byte?
To find the number of the first byte, we multiply the offset value by 8. This means that the first byte number is 800. We cannot determine the number of the last byte unless we know the length of the data.

A packet has arrived in which the offset value is 100. What is the number of the first byte? Do we know the number of the last byte?

To find the number of the first byte, we multiply the offset value by 8. This means that the first byte number is 800. We cannot determine the number of the last byte unless we know the length of the data.

A packet has arrived in which the offset value is 100, the value of HLEN is 5 and the value of the total length field is 100. What is the number of the first byte and the last byte?

The first byte number is 100 × 8 = 800. The total length is 100 bytes and the header length is 20 bytes (5 × 4), which means that there are 80 bytes in this datagram. If the first byte number is 800, the last byte number must be 879.

IP Datagram Header comprised of two parts:
- Fixed: 20 bytes (previously discussed)
- Variable: comprises “options” (max 40 bytes)
- Options used for network testing and debugging
- Options not required to appear in IP packet but option processing mandatory in all IP implementations

Only 6 option types are being used
**Categories of options**

- Multi-byte options require length and data fields

**No operation option**

- Code: 1 (0000000)
  - a. No operation option
  - b. Used to align beginning of an option
  - c. Used to align the next option

**End of option option**

- Code: 0 00000000
  - a. End of option
  - b. Used for padding
  - Only one End-of-Option (EoO) can be used
  - EoO must be used as last field of IP Option field
  - After EoO, packet receiver expects payload data

**Record Route Option**

- Records the internet routers that handle the datagram
- Can store at most 9 router’s IP addresses
  - 40 byte options – (9 * 4-bytes) – Code (1 byte) – Length (1 byte) – Pointer (1 byte) = 1
  - If (Pointer > Length) then Options area is full & no further changes are made
  - Otherwise, router inserts its outgoing IP address
  - Each routers increments pointer field by 4 (bytes) to point to next free entry for IP address (to prepare for next router’s address)

**Record route option**

3 bytes →

<table>
<thead>
<tr>
<th>Code</th>
<th>Length (Total length)</th>
<th>Pointer (init at 4)</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

9 IP addr maximum (9 * 4-byte addr = 36 bytes) + 3 = 39
40 byte maximum for Options field

**Record route concept**

Length field includes code, length and data
Code is 7; Ptr increments
**Strict Source Route**

- Used to specify a predetermined route through the Internet
- Useful to specify a preferred route based on type of service, e.g., min delay, max throughput
- All routers specified must be visited
- If packet received by router not in list, packet dropped and error generated
- If packet reaches destination without visiting all routers, packet dropped and error generated

**Strict source route option**

<table>
<thead>
<tr>
<th>Code</th>
<th>Length (Total length)</th>
<th>Pointer</th>
</tr>
</thead>
<tbody>
<tr>
<td>First IP address (optional when unused)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Second IP address (optional when unused)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Last IP address (optional when unused)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Strict source route concept**

Incoming router IP addressed used in datagram

**Loose Source Route**

- Similar to Strict Source Routing
- Datagram can visit routers not in option list

**Timestamp**

- Used to record the time that the datagram was processed by the router
- Helps users and managers track the behavior of Internet routers
- Determine time required for datagram to travel from one router to the next
- Amount is estimate since no router’s clocks cannot be guaranteed to be in sync

**Timestamp option**

<table>
<thead>
<tr>
<th>Code</th>
<th>Length (Total length)</th>
<th>Pointer</th>
<th>O-Flow</th>
<th>Flags</th>
</tr>
</thead>
<tbody>
<tr>
<td>First IP address</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Second IP address</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Last IP address</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
**Use of flag in timestamp**

1. **Flag 0**
   - Enter timestamps only
   - Enter IP addresses and timestamps
   - IP addresses given, enter timestamps

<table>
<thead>
<tr>
<th>Flag</th>
<th>No operation</th>
<th>End of option</th>
<th>Record route</th>
<th>Strict source route</th>
<th>Loose source route</th>
<th>Timestamp</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

2. **Flag 1**
   - 140.10.6.3
   - 280.14.7.9
   - 138.9.22.26

3. **Flag 2**
   - 140.10.6.3
   - 280.14.7.9
   - 138.9.22.26

**Example 10**

Which of the six options must be copied to each fragment?

- No operation
- End of option
- Record route
- Strict source route
- Loose source route
- Timestamp

**Solution**

We look at the first (left-most) bit of the code for each option.

- **No operation**: Code is 00000001; no copy.
- **End of option**: Code is 00000000; no copy.
- **Record route**: Code is 00000000; no copy.
- **Strict source route**: Code is 10001001; copied.
- **Loose source route**: Code is 10000011; copied.
- **Timestamp**: Code is 01000100; no copy.

**Example 11**

Which of the six options are used for datagram control and which are used for debugging and management?

**Solution**

We look at the second and third (left-most) bits of the code.

- **No operation**: Code is 00000001; control.
- **End of option**: Code is 00000000; control.
- **Record route**: Code is 00000111; control.
- **Strict source route**: Code is 10001001; control.
- **Loose source route**: Code is 10000011; control.
- **Timestamp**: Code is 01000100; debugging
Checksum

- Error detection mechanism
- Protects against packet corruption during transmission
- Used by most TCP/IP Protocols
- Checksum calculated at sender; sent with packet
- Receiver repeats the same calculation

To create the checksum the sender does the following:

1. The packet is divided into \( k \) sections, each of \( n \) bits.
2. All sections are added together using one’s complement arithmetic.
3. The final result is complemented to make the checksum.

Checksum concept

Checksum in one’s complement arithmetic

\[ T - \bar{T} = -0 \]

Example of checksum calculation in binary
Example of checksum calculation in hexadecimal

<table>
<thead>
<tr>
<th>4</th>
<th>5</th>
<th>0</th>
<th>28</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>17</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>10.12.14.5</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12.6.7.9</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

4, 5, and 0 → 4 5 0 0
28 → 0 0 1 C
1 → 0 0 0 1
0 and 0 → 0 0 0 0
4 and 17 → 0 4 1 1
0 → 0 0 0 0
10.12 → 0 A 0 C
14.5 → 0 E 0 5
12.6 → 0 C 0 6
7.9 → 0 7 0 9
Sum → 7 4 4 E
Checksum → 8 B B 1

Checksum on Header
- Only applied to IP header
- Higher-level protocols have own error detection
- Checksum recomputed at each router
- Faster to only compute checksum on the (smaller) header data

Check Appendix C for a detailed description of checksum calculation and the handling of carries.

8.5

IP PACKAGE

Reassembly table

<table>
<thead>
<tr>
<th>St.</th>
<th>S. A.</th>
<th>D. I.</th>
<th>T. O.</th>
<th>F.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Interface MIB

- Interface states
- Interface errors
- Interface counters