Chapter 8

Internet Protocol (IP)
CONTENTS

• DATAGRAM
• FRAGMENTATION
• OPTIONS
• CHECKSUM
• IP PACKAGE
### Fragmentation

![Diagram of IP packet structure](image)

#### Fixed values

- **VER**: 4 bits
- **HLEN**: 4 bits
- **Service type**: 8 bits
- **Total length**: 16 bits
- **Identification**: 16 bits
- **Flags**: 3 bits
- **Fragmentation offset**: 13 bits
- **Time to live**: 8 bits
- **Protocol**: 8 bits
- **Header checksum**: 16 bits

#### Source IP address

#### Destination IP address

#### Option

Length: Minimum 46 bytes

- **Header**
- **Data < 46 bytes**
- **Padding**
- **Trailer**
# 8.3 OPTIONS

- Used for network testing and debugging
- Each option follows the TLV (Type-Length-Value) format

<table>
<thead>
<tr>
<th>Code</th>
<th>Length</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>8 bits</td>
<td>8 bits</td>
<td>Variable length</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Copy</th>
<th>Class</th>
<th>Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 bit</td>
<td>2 bits</td>
<td></td>
</tr>
</tbody>
</table>

- **Copy**
  - 0: Copy only in first fragment
  - 1: Copy into all fragments

- **Class**
  - 00: Datagram control
  - 01: Reserved
  - 10: Debugging and management
  - 11: Reserved

- **Number**
  - 00000: End of option
  - 00001: No operation
  - 00011: Loose source route
  - 00100: Timestamp
  - 00111: Record route
  - 01001: Strict source route
**Record route option**

*How did it arrive?*

<table>
<thead>
<tr>
<th>Code: 7 00000111</th>
<th>Length (Total length)</th>
<th>Pointer</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>First IP address</td>
<td></td>
</tr>
<tr>
<td></td>
<td>(Empty when started)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Second IP address</td>
<td></td>
</tr>
<tr>
<td></td>
<td>(Empty when started)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Last IP address</td>
<td></td>
</tr>
<tr>
<td></td>
<td>(Empty when started)</td>
<td></td>
</tr>
</tbody>
</table>

Up to 9 placeholders created by Source host
Record route concept
Strict source route option
How should it travel?

- Route strictly dictated by Source
  - For performance, reliability, testing, etc.
- All pre-specified routers MUST be visited, and nothing else.
  - Otherwise: discard datagram and send error message.
Strict source route example
Figure after corrections
**Loose source route option**

- Minimum portion of Route is dictated by Source
- All pre-specified routers MUST be visited, and possibly others as well.
  - Otherwise: discard datagram and send error message.

<table>
<thead>
<tr>
<th>Code: 131 10000011</th>
<th>Length (Total length)</th>
<th>Pointer</th>
</tr>
</thead>
<tbody>
<tr>
<td>First IP address</td>
<td>(Filled when started)</td>
<td></td>
</tr>
<tr>
<td>Second IP address</td>
<td>(Filled when started)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Last IP address</td>
<td>(Filled when started)</td>
<td></td>
</tr>
</tbody>
</table>
**Timestamp option**

When did each visited router process a datagram?

<table>
<thead>
<tr>
<th>Code: 68 01000100</th>
<th>Length (Total length)</th>
<th>Pointer</th>
<th>O-Flow 4 bits</th>
<th>Flags 4 bits</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>First IP address</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Second IP address</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Last IP address</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Use of flag in timestamp

- **Enter timestamps only**
  - Flag: 0

- **Enter IP addresses and timestamps**
  - Flag: 1
  - 140.10.6.3
  - 200.14.7.9
  - 138.6.22.26

- **IP addresses given, enter timestamps**
  - Flag: 3

**Outgoing**
Timestamp concept
Example 10

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

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 0000111; no copy.

Strict source route: Code is 10001001; copied.

Loose source route: Code is 10000111; 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 0000001; control.
End of option: Code is 0000000; control.
Record route: Code is 0000111; control.
Strict source route: Code is 1001001; control.
Loose source route: Code is 1000011; control.
Timestamp: Code is 01000100; debugging
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

Sender
- Section 1: $n$ bits
- Section 2: $n$ bits
- Checksum: All 0s
- Section $k$: $n$ bits

Receiver
- Section 1: $n$ bits
- Section 2: $n$ bits
- Checksum: $n$ bits
- Section $k$: $n$ bits

Checksum

Packet

Sum
- $n$ bits
- Complement
- $n$ bits
- Checksum

Result
- $n$ bits
- If the result is 0, keep; otherwise, discard.
Checksum in one’s complement arithmetic

\[ T - T = -0 \]
Example of checksum calculation in binary

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>5</td>
<td>0</td>
<td>28</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>17</td>
<td></td>
<td>0</td>
</tr>
</tbody>
</table>

10.12.14.5

12.6.7.9

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>4, 5, and 0</td>
<td>28</td>
<td>1</td>
<td>0 and 0</td>
<td>4 and 17</td>
</tr>
<tr>
<td>0</td>
<td>0000000000</td>
<td>00000000</td>
<td>00000000</td>
<td>00000000</td>
</tr>
<tr>
<td></td>
<td>0000000000</td>
<td>00011100</td>
<td>00000001</td>
<td>00010001</td>
</tr>
<tr>
<td></td>
<td>0000000000</td>
<td>00000000</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>00001010</td>
<td>00001100</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>00001110</td>
<td>00000101</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>00001100</td>
<td>00000110</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>00000111</td>
<td>00001001</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sum</td>
<td>01110100</td>
<td>01001110</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Checksum</td>
<td>10001011</td>
<td>10110001</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
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>
</tbody>
</table>

10.12.14.5

12.6.7.9

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
8.5 IP PACKAGE

From upper-layer protocol

IP

Data and destination address

Header-adding module

IP packet

Processing module

To upper-layer protocol

Data

Reassembly module

Reassembly table

Routing module

Routing table

MTU table

Fragmentation module

IP packet, next hop, interface

IP packet, next hop

From data link layer

To data link layer
### MTU table

<table>
<thead>
<tr>
<th>Interface Number</th>
<th>MTU</th>
</tr>
</thead>
<tbody>
<tr>
<td>●●●●●●●●</td>
<td>●●●●●●●●</td>
</tr>
</tbody>
</table>

### Reassembly table

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>●●●●●●</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Sorted Linked-List of fragments of the same datagram