IPv6 Extension Headers and Network Security

Bill Cerveny, Internet2

Andrew Lake, Albion College

Summer 2005 Joint-Techs

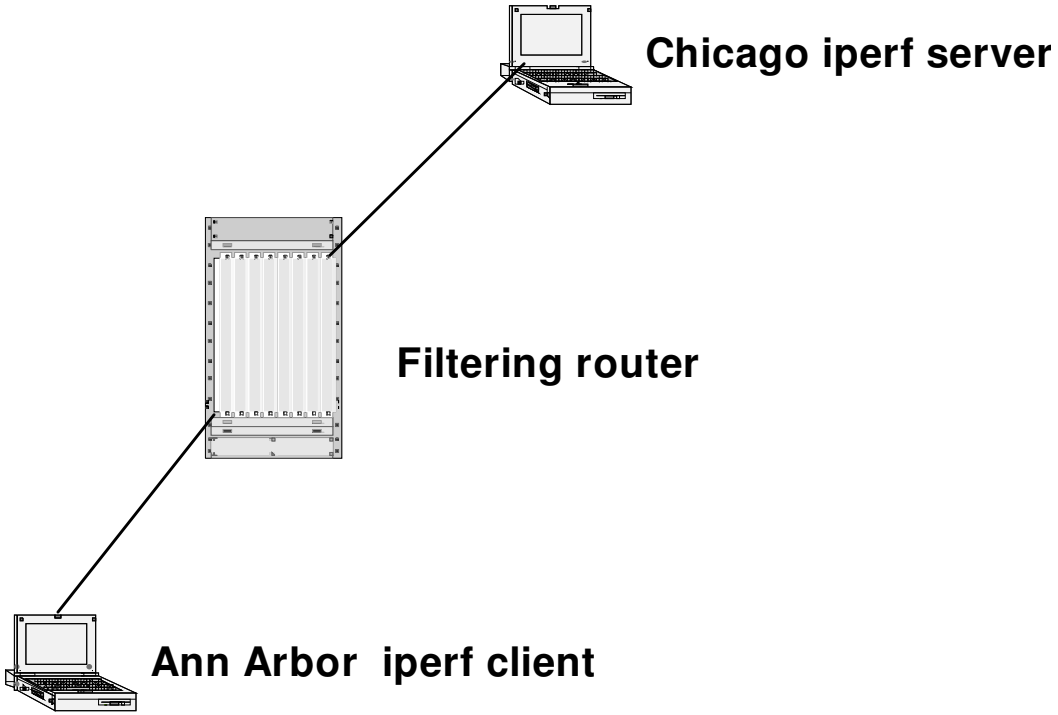
Vancouver, BC

Outline

- Description of event
- Breakdown and diagnosis of what happened
- Recommendations

Initial Environment

Client: iperf –c chicago –b 10m –u –V –p xxxx
Server: iperf –s –u –V –p xxxx

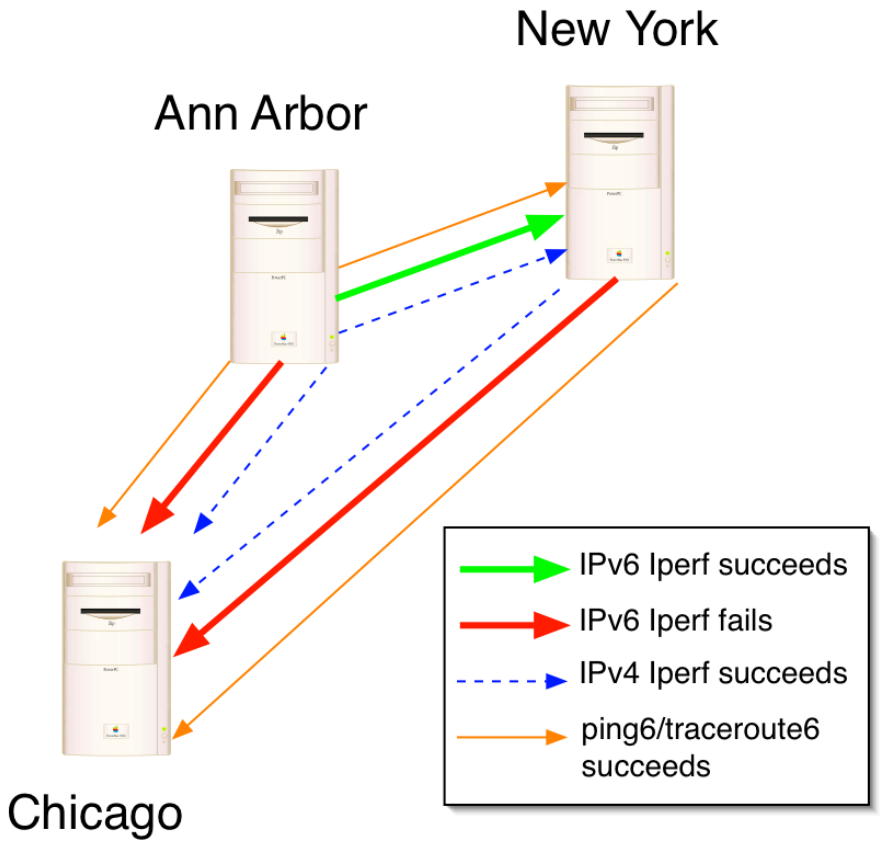


Problem and Diagnosis

- In a nutshell, a straightforward IPv6 iperf test wasn't working
  - UDP port on filtering router was opened up
  - Still didn't work
- Ran iperf from Ann Arbor to a different server in New York
  - Worked as expected

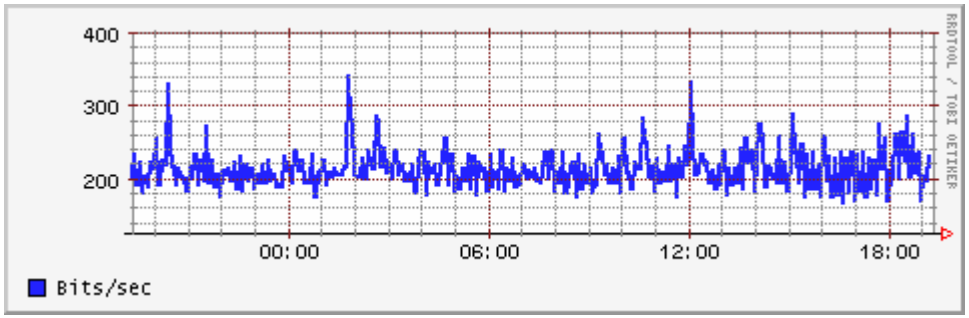
Iperf Test Failure Analysis

Client: iperf –c chicago –u –V –p xxxx
Server: iperf –s –u –V –p xxxx

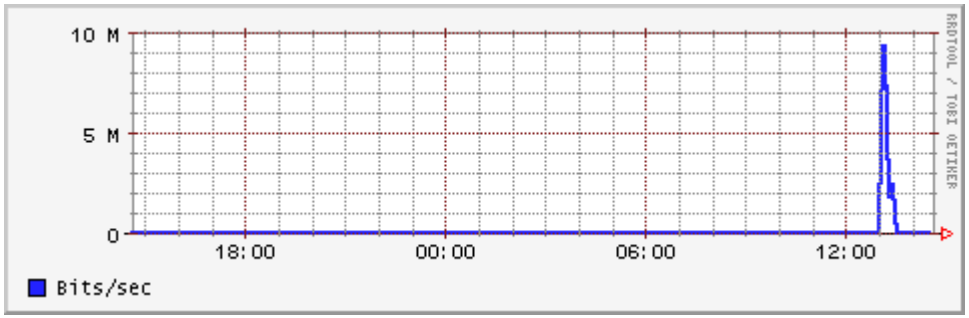


Abilene Traffic Graphs

- "v6-udp" graph



- "v6-other" graph



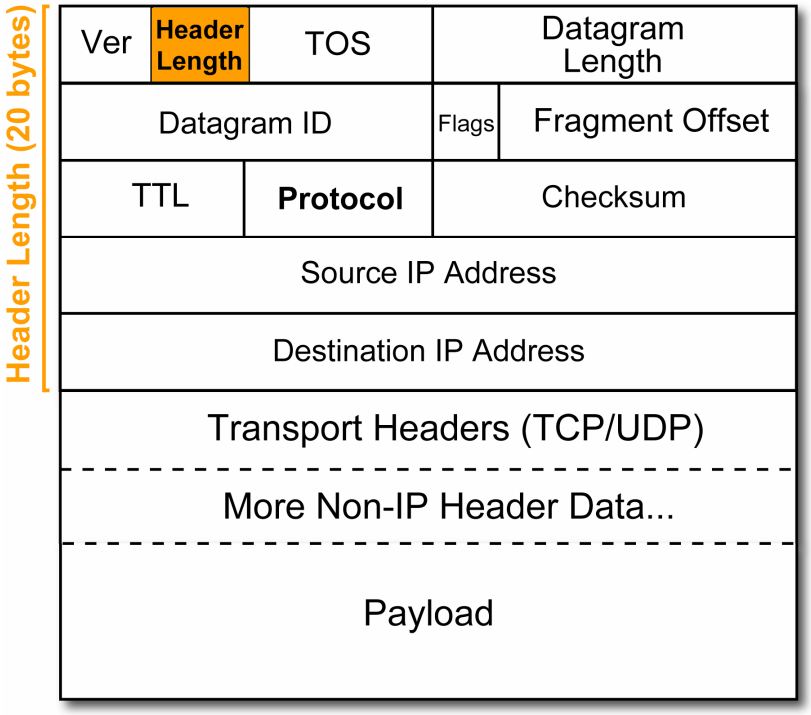
Pseudocode fragment:

```
filter v6filter {
if multicast
        Count v6-multi
    elseif tcp
        Count v6-tcp
    elseif udp
        Count v6-udp
    elseif otherheader
        Count v6-other
      }
```

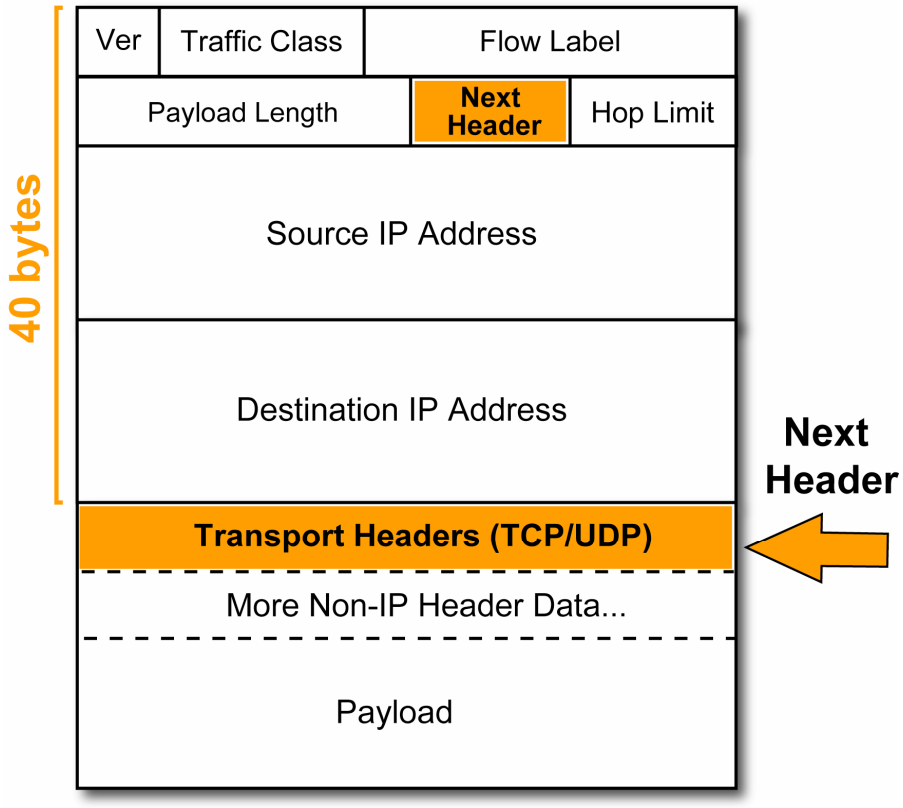
http://vixen.grnoc.iu.edu/jfirewall-viz/v6_index.html

IPv4 vs IPv6
The IP Packet

IPv4 Packet (No Options)

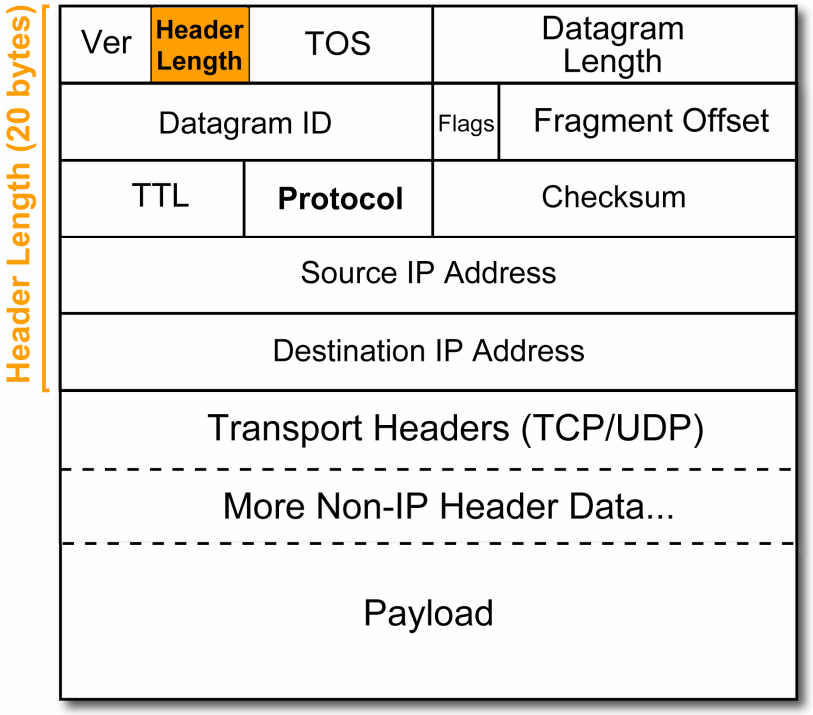


IPv6 Packet (No Extensions)

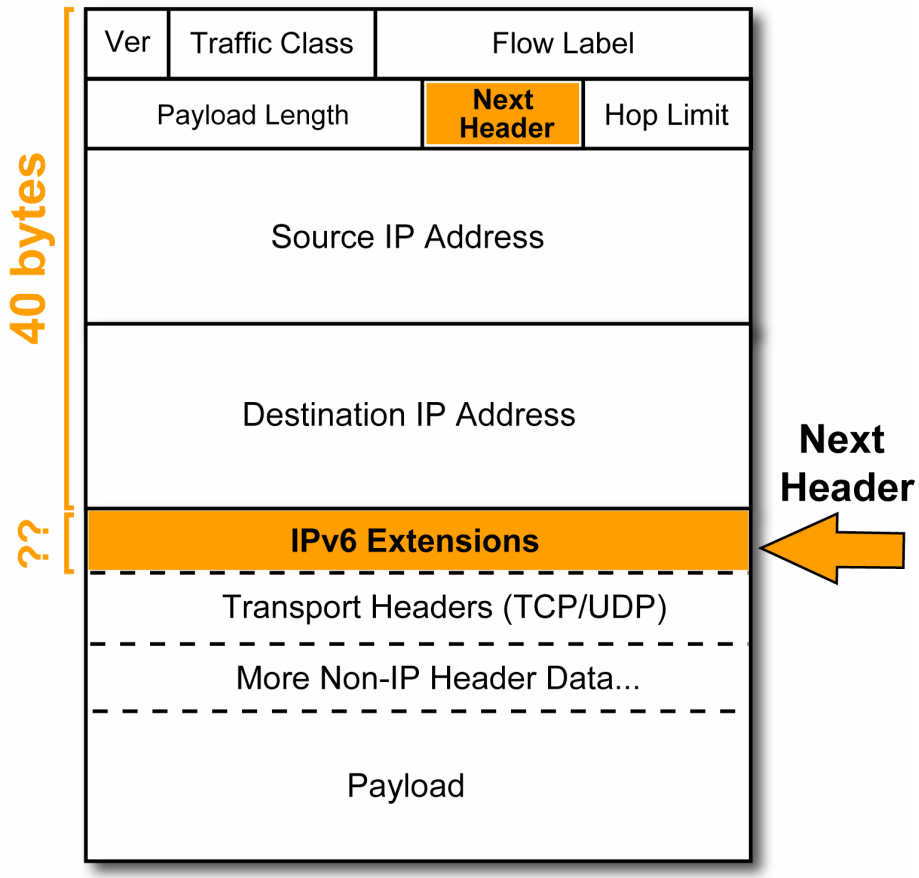


IPv4 vs IPv6
IPv6 Extensions

IPv4 Packet (no Options)



IPv6 Packet (with Extensions)

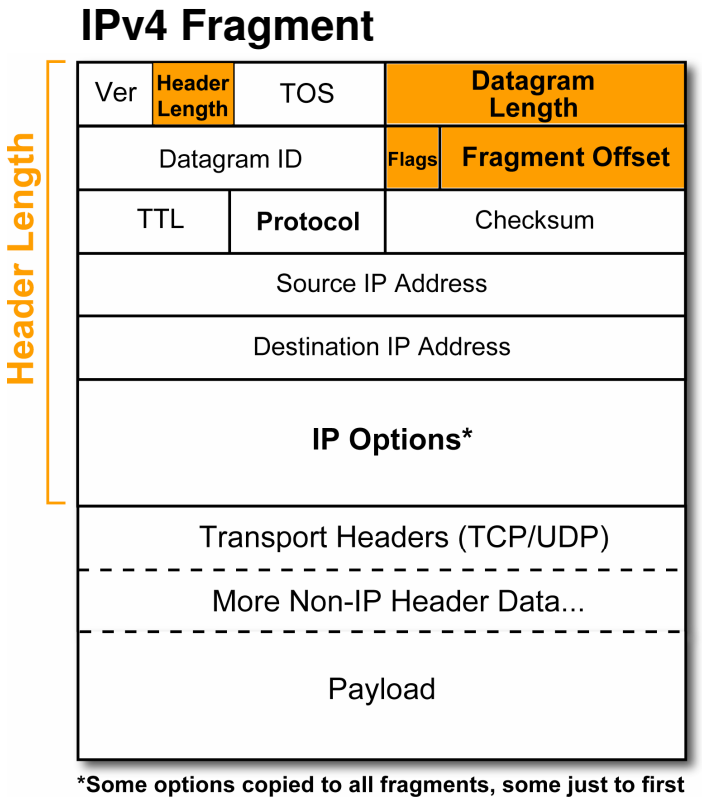


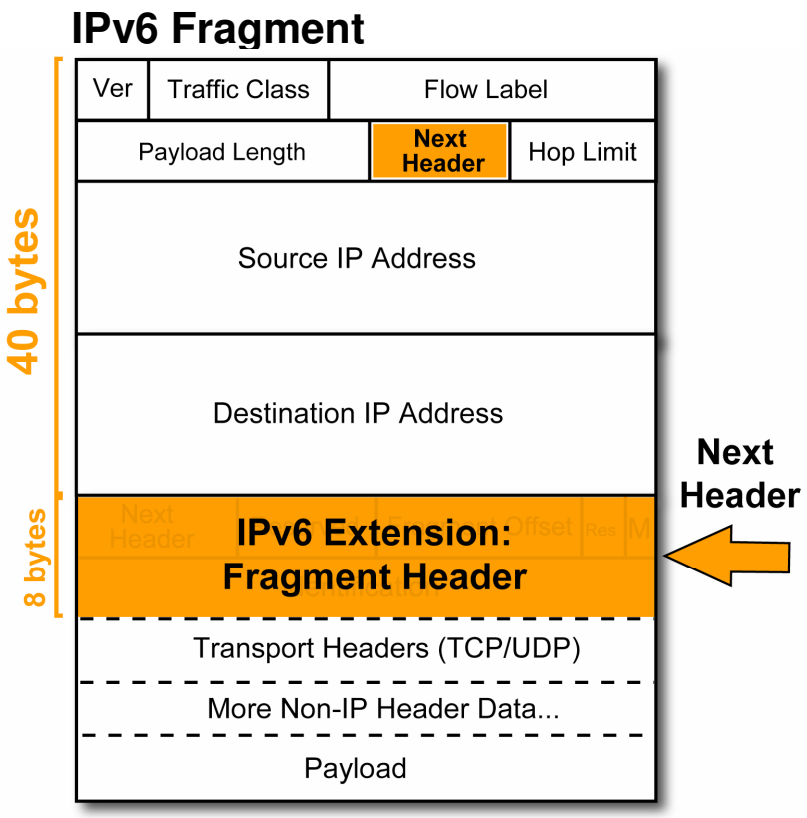
Recommended order of headers in an IPv6 packet (RFC 2460)

Recommended order of headers in an IPv6 packet:

1. IPv6 header (40 bytes)
2. Hop-by-hop options header (variable)
3. Destination options header (1) (variable)
4. Routing header (variable)
5. Fragment header (8 bytes)
6. Authentication header (variable)
7. Encapsulation Security Payload header (variable)
8. Destination options header (2) (variable)
9. Upper-layer header (for example, TCP or UDP)

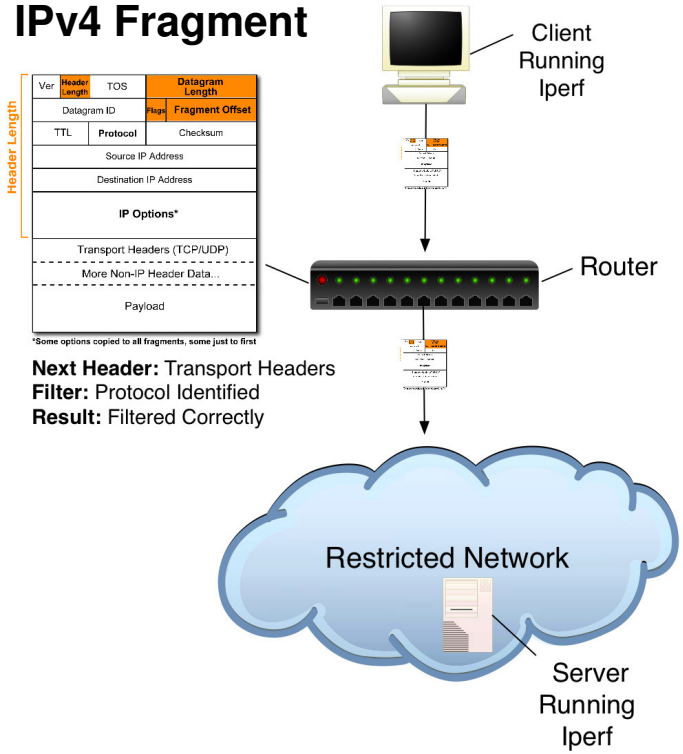
IPv4 vs IPv6
Fragmentation

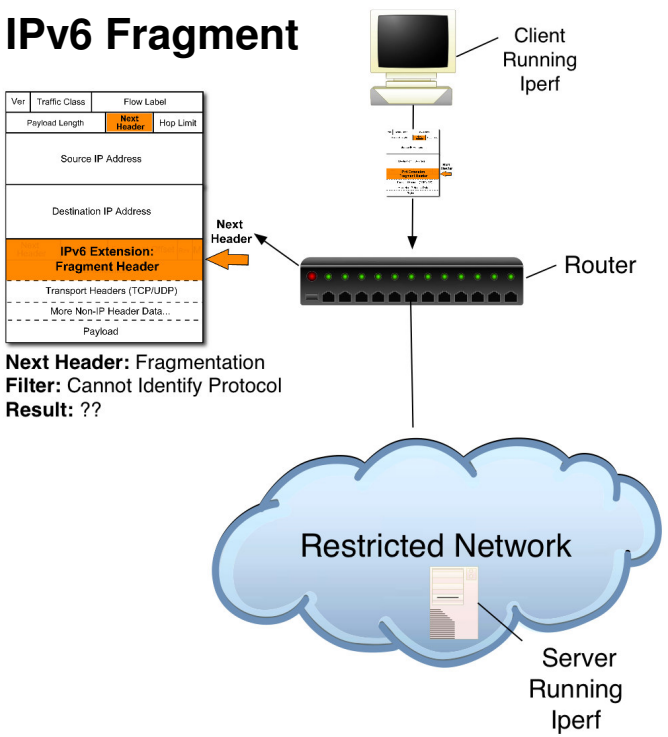




Routers and Filter Packet Handling

- When the router sees an IPv4 packet, it looks for transport layer information (like whether the packet is TCP or UDP) at the point that is “header length” away from the start of the IP header
- As currently implemented, when the router looks at the IPv6 packet, it tries to characterize the packet by looking in the default next header field.





IPv6 Fragmented vs. Unfragmented Datagram

Unfragmented

```
▷ Ethernet II, Src:
▽ Internet Protocol Version 6
    Version: 6
    Traffic class: 0x00
    Flowlabel: 0x00000
    Payload length: 1458
    Next header: UDP (0x11)
    Hop limit: 64
    Source address:
    Destination address: 2001:468:1
▽ User Datagram Protocol, Src Port:
    Source port: 33427 (33427)
    Destination port:
    Length: 1458
    Checksum: 0xe027 (correct)
  Data (1450 bytes)
```

Fragmented

```
▷ Ethernet II, Src:
▽ Internet Protocol Version 6
    Version: 6
    Traffic class: 0x00
    Flowlabel: 0x00000
    Payload length: 1456
    Next header: IPv6 fragment (0x2c)
    Hop limit: 64
    Source address:
    Destination address:
▽ Fragmentation Header
    Next header: UDP (0x11)
    Offset: 0
    More fragments: Yes
    Identification: 0x00000709
▽ User Datagram Protocol, Src Port: 33427
    Source port: 33427 (33427)
    Destination port:
    Length: 1478
    Checksum: 0x5a48
  Data (1440 bytes)
```

IPv6 Fragmented vs. Unfragmented Datagram

Unfragmented

```
▷ Ethernet II, Src:
▽ Internet Protocol Version 6
    Version: 6
    Traffic class: 0x00
    Flowlabel: 0x00000
    Payload length: 1458
    Next header: UDP (0x11)
    Hop limit: 64
    Source address:
    Destination address: 2001:468:
▽ User Datagram Protocol, Src Port
    Source port: 33427 (33427)
    Destination port:
    Length: 1458
    Checksum: 0xe027 (correct)
  Data (1450 bytes)
```

Fragmented

```
▷ Ethernet II, Src:
▽ Internet Protocol Version 6
    Version: 6
    Traffic class: 0x00
    Flowlabel: 0x00000
    Payload length: 1456
    Next header: IPv6 fragment (0x2c)
    Hop limit: 64
    Source address:
    Destination address:
▽ Fragmentation Header
    Next header: UDP (0x11)
    Offset: 0
    More fragments: Yes
    Identification: 0x00000709
▽ User Datagram Protocol, Src Port: 33427
    Source port: 33427 (33427)
    Destination port:
    Length: 1478
    Checksum: 0x5a48
  Data (1440 bytes)
```

IPv6 Fragmented vs. Unfragmented Datagram

Unfragmented

```
▷ Ethernet II, Src:
▽ Internet Protocol Version 6
    Version: 6
    Traffic class: 0x00
    Flowlabel: 0x00000
    Payload length: 1458
    Next header: UDP (0x11)
    Hop limit: 64
    Source address:
    Destination address: 2001:468:14
▽ User Datagram Protocol, Src Port:
    Source port: 33427 (33427)
    Destination port:
    Length: 1458
    Checksum: 0xe027 (correct)
  Data (1450 bytes)
```

Fragmented

```
▷ Ethernet II, Src:
▽ Internet Protocol Version 6
    Version: 6
    Traffic class: 0x00
    Flowlabel: 0x00000
    Payload length: 1456
    Next header: IPv6 fragment (0x2c)
    Hop limit: 64
    Source address:
    Destination address:
▽ Fragmentation Header
    Next header: UDP (0x11)
    Offset: 0
    More fragments: Yes
    Identification: 0x00000709
▽ User Datagram Protocol, Src Port: 33427
    Source port: 33427 (33427)
    Destination port:
    Length: 1478
    Checksum: 0x5a48
  Data (1440 bytes)
```

What caused the fragmentation

- The default setting in iperf for datagram size is 1470 bytes.
- Given typical Ethernet network with 1500-byte MTUs, IPv6 packets will fragment 1470-byte datagrams, whereas IPv4 packets will not
  - IPv4 IP Header + UDP header + 1470 < 1500
  - IPv6 IP Header + UDP header + 1470 > 1500

2nd and subsequent packets

- The second and subsequent packets of fragmented datagrams don't contain any transport header information.
- This is true for both IPv6 *and* IPv4

IPv4 and IP Options

- A similar problem can occur with the IP options field in IPv4 and the location of the transport layer header is moved deeper into the packet
- Routers tend to drop packets with IP options and developers have been sensitized to avoid making use of IP options in their applications.

Software Suggestions

- Unless testing network performance with fragmented IPv6 packets, don't send UDP packets that must be fragmented
  - For iperf udp packets, largest datagram size should be 1450 bytes (-l option). This is actually suggested deep in the iperf documentation.
  - We propose that iperf default IPv6 UDP datagram size be changed from 1470 to 1450 bytes
  - *Actually, the packet size should be automatically computed to avoid/prevent fragmentation.*

Implications & Security Risks

- Many high-throughput WAN routers may not figure out layer 3 or 4 header details if there are any extension headers
- Those routers that do extension header analysis may suffer performance hits
- Filtering on layer 3 or 4 header details could be hit or miss
  - Avoid filters/acls that filter layer 3/4 detail and where final option is “allow any”

Unanswered Questions

- How do various router platforms handle this?
- How hard a problem is this to solve?
- At what speed does it become impossible to evaluate extension headers?

Summary

- With IPv6 extension headers, it is trivial to defeat router level 3/4 filters with “allow any” type filters.
- Evaluating level 3/4 headers with extension headers at high speed is hard
- Avoid sending datagrams which are likely to be fragmented or use other extension headers
- This is one of those bumps in the road to IPv6.

Acknowledgements

- Ben Eater, Juniper Networks
- Tony Hain, Cisco
- Jim Ferguson, NLANR
- Bill Owens, NYSERnet
- Internet2:
  - Charles Yun
  - Matt Zekauskas
  - Richard Carlson


INTERNET®
www.internet2.edu