The Network Layer
IPv4 and IPv6
Part 3
Jean-Yves Le Boudec
2017
Contents

9. Proxy ARP
10. Fragmentation
11. Interworking h4-h6 with NATs

Textbook
Chapter 5: The Network Layer
Reminder: IP principle says one subnet = one LAN

Assume you want to cheat with this principle, e.g. **12.12.12.4** is in the wrong place, but you want to keep it that way. You can support this configuration with /32 entries (with IPv4) in forwarding tables at R1, R2 and R3 (“Host Routes”)
Does this solve the problem?

A. Yes because R2 now send packets to \textbf{12.12.12.4} on the correct interface

B. No this does not work because subnets cannot be split across multiple locations

C. No it does not work for other reasons

D. I don’t know
sic500cs should be a ....

A. Router
B. Bridge
C. Application Layer gateway
D. I don’t know

- All subnets have 255.255.255.0 netmask
- Subnet 84 is on both sides of sic500cs
Assume you want to have sic500cs operating as a router

This deviates from the IP principle: one subnet = one LAN

Therefore we need a second deviation (at least) to compensate

We can use PROXY-ARP

- sic500cs answers for ARP REQs / NDP NSs on behalf of greenmac

1. All subnets have 255.255.255.0 netmask
2. Subnet 84 is on both sides of sic500cs
ed0-ext has a packet to 128.178.84.133 and sends an ARP REQ

A. sic500cs replies with own MAC address
B. sic500cs replies with the MAC address of greenmac
C. Greenmac replies with his MAC address
D. Greenmac replies with the MAC address of sic500cs
E. I don’t know

1. All subnets have 255.255.255.0 netmask
2. Subnet 84 is on both sides of sic500cs
3. sic500cs does PROXY-ARP on behalf of greenmac
10. Fragmentation

Link-layer networks have different maximum frame lengths

MTU (maximum transmission unit) = maximum frame size usable for an IP packet

MAC layer options and tunnels make this worse

<table>
<thead>
<tr>
<th>Link-layer Network</th>
<th>MTU</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ethernet, WiFi</td>
<td>1500</td>
</tr>
<tr>
<td>VPN Tunnel in IPv4</td>
<td>1400</td>
</tr>
<tr>
<td>Token Ring 4 Mb/s</td>
<td>4464</td>
</tr>
<tr>
<td>Token Ring 16 Mb/s</td>
<td>17914</td>
</tr>
<tr>
<td>FDDI</td>
<td>4352</td>
</tr>
<tr>
<td>X.25</td>
<td>576</td>
</tr>
<tr>
<td>Frame Relay</td>
<td>1600</td>
</tr>
<tr>
<td>ATM with AAL5</td>
<td>9180</td>
</tr>
<tr>
<td>Hyperchannel</td>
<td>65535</td>
</tr>
<tr>
<td>PPP</td>
<td>296 to 1500</td>
</tr>
</tbody>
</table>

Z:\>netsh interface ipv6 show subinterfaces

<table>
<thead>
<tr>
<th>MTU</th>
<th>MediaSenseState</th>
<th>Bytes In</th>
<th>Bytes Out</th>
<th>Interface</th>
</tr>
</thead>
<tbody>
<tr>
<td>4294967295</td>
<td>1</td>
<td>0</td>
<td>28980</td>
<td>Loopback Pseudo-Interface 1</td>
</tr>
<tr>
<td>1500</td>
<td>5</td>
<td>0</td>
<td>0</td>
<td>Wireless Network Connection 2</td>
</tr>
<tr>
<td>1400</td>
<td>1</td>
<td>19419</td>
<td>14328</td>
<td>VPN EPFL</td>
</tr>
<tr>
<td>1500</td>
<td>1</td>
<td>126682357</td>
<td>17501036</td>
<td>Local Area Connection</td>
</tr>
<tr>
<td>1280</td>
<td>5</td>
<td>0</td>
<td>536</td>
<td>Local Area Connection* 12</td>
</tr>
</tbody>
</table>
IPv4 Fragmentation

IPv4 hosts or routers may have IP datagrams larger than MTU. Fragmentation is performed when IP datagram too large. Re-assembly is only at destination, never at intermediate points. Fragmentation is in principle avoided with TCP.
IPv4 Fragmentation (2)

IP datagram is *fragmented* if

MTU of interface < datagram total length

all fragments are self-contained IP packets

fragmentation controlled by fields: Identification, Flag and Fragment Offset in IPv4 header;

IP *datagram* = original ; IP *packet* = fragments or complete datagram

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2a</th>
<th>2b</th>
<th>2c</th>
</tr>
</thead>
<tbody>
<tr>
<td>Length</td>
<td>1420</td>
<td>620</td>
<td>620</td>
<td>220</td>
</tr>
<tr>
<td>Identification</td>
<td>567</td>
<td>567</td>
<td>567</td>
<td>567</td>
</tr>
<tr>
<td>More Fragment flag</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Offset</td>
<td>0</td>
<td>0</td>
<td>75</td>
<td>150</td>
</tr>
<tr>
<td>8 * Offset</td>
<td>0</td>
<td>0</td>
<td>600</td>
<td>1200</td>
</tr>
</tbody>
</table>

Fragment data size (here 600) is always a multiple of 8
Identification given by source
IPv6 Fragmentation

Same as IPv4 except

- Routers never fragment – drop packet if too large
- Fragmentation header is a «next header»
- Minimum MTU is 1280 Bytes (vs 68 for IPv4)
Fragmentation Considered Harmful

Fragmentation requires re-assembly at destination; this may cause deadlocks and identification wrapping problems. Unit of loss is smaller than unit of re-transmission: can worsen congestion (avalanche effect).

Solution = avoid fragmentation as much as possible by discovering "Path MTU" (= minimum MTU along one path)
Methods for setting Path MTU

Path MTU Discovery (PMTUD)
1. Host sets “Don’t Fragment” bit on all datagrams (IPv4 only – with IPv6 routers never fragment)
2. sets PathMTU to local MTU
3. routers send ICMP message: “destination unreachable/fragmentation needed” if MTU is too large
4. host reduces PathMTU estimate to next smallest value
5. after timeout, host increases PMTU estimate

Packetization Layer Path MTU Discovery (PLPMTUD)
- does not require routers to send ICMP messages
- relies on TCP making Path MTU probes from time to time; estimate the largest possible Path-MTU that works by observing packets that were acknowledged

Path-MTU is kept in destination cache by operating system
IPv6 example

:\>netsh interface ipv6 show destinationcache

Interface 12: Local Area Connection

<table>
<thead>
<tr>
<th>PMTU Destination Address</th>
<th>Next Hop Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>1500 2a00:1450:4001:c02::54</td>
<td>fe80::208:e3ff:feff:fc50</td>
</tr>
<tr>
<td>1500 2a00:1450:4002:801::100c</td>
<td>fe80::208:e3ff:feff:fc50</td>
</tr>
<tr>
<td>1500 2a00:1450:4003:803::100f</td>
<td>fe80::208:e3ff:feff:fc50</td>
</tr>
<tr>
<td>1500 2a00:1450:400a:804::1008</td>
<td>fe80::208:e3ff:feff:fc50</td>
</tr>
<tr>
<td>1500 2a00:1450:400a:805::100d</td>
<td>fe80::208:e3ff:feff:fc50</td>
</tr>
<tr>
<td>1500 2a00:1450:400a:805::100f</td>
<td>fe80::208:e3ff:feff:fc50</td>
</tr>
<tr>
<td>1500 2a00:1450:400a:807::1018</td>
<td>fe80::208:e3ff:feff:fc50</td>
</tr>
<tr>
<td>1500 2a00:1450:400a:807::101f</td>
<td>fe80::208:e3ff:feff:fc50</td>
</tr>
<tr>
<td>1500 2a03:2880:f006:101:face:b00c:0:1</td>
<td>fe80::208:e3ff:feff:fc50</td>
</tr>
<tr>
<td>1500 2a02:26f0:12:198::eed</td>
<td>fe80::208:e3ff:feff:fc50</td>
</tr>
</tbody>
</table>
TCP, UDP and Fragmentation

The UDP service interface accepts a datagram

- up to 64 KB (by default) or up to 4GB (with IPv6 “jumbo payload” extension header
- UDP datagram passed to the IP service interface as one SDU
- is fragmented at the source if resulting IP datagram is too large to fit on Path MTU

The TCP service interface is stream oriented

- packetization is done by TCP – fragmentation should in principle be avoided
- TCP connections negotiate an MSS (maximum segment size)
  Host operating system should verify that
  \[ \text{MSS} \leq \text{PathMTU} - \text{IP header size} - \text{TCP header size} \]
When a host generates UDP traffic, the port number is always present in all packets

A. True  
B. False  
C. True with IPv4, false with IPv6  
D. True with IPv6, false with IPv4  
E. I don’t know
One TCP segment is contained in one IPv4 datagram that is fragmented by a router on its way from source to destination. One of the fragments is lost. What will TCP re-transmit?

A. The bytes that were in the missing fragment
B. The bytes that were in all fragments of the datagram, missing or not
C. It depends whether the loss is detected by fast retransmit or by timeout
D. I don’t know
11. IPv4 and IPv6 Interworking with NATs

IPv4 and IPv6 are incompatible

- v4 only host cannot handle IPv6 packets
- v6 only host cannot handle IPv4 packets

What needs to be solved:

interworking: h6 to h4
like-to-like access

- 6 to 6 over 4
- 4 to 4 over 6

In this module we study
interworking
like-to-like access is studied in “tunnels”
Reminder:
Dual Stack Application Layer Gateways (ALG46s) can be used to solve h4 to h6 Interworking

Application layer gateway (e.g. web proxy) relays HTTP questions / answers.
ALG must be dual stack
Simple but performance is not optimal (store and forward)
Plus dependency on application
NAT64 (RFC6146) is an alternative solution to solve h6 to h4 interworking

NAT64 translates an IPv4 packet into an IPv6 packet and vice-versa; no encapsulation
NAT uses an IPv4 address pool to translate IPv6 address (No need for IPv6 private pool)
Port translation is used (as in any NAT) to save number of IPv4 addresses of the pool

To h6, h4 appears under an IPv6 translated address
To h4, h6 appears under an IPv4 translated address
Translation from IPv6 address to IPv4 translated address is stateful

NAT owns pool 120.130.26/24
Packet from h6 is translated by NAT to next available IPv4 source address /port number combination – this is the usual NAT’s job
Need to keep table of all existing mapping – this is called “stateful” operation
Translation from IPv4 address to IPv6 translated address is stateless

IPv4 addresses are mapped to valid IPv6 addresses such as 64:ff9b::c000:201

Such an address is an IPv4-embedded-IPv6 address

The block 64:ff9b::/96 is a well know prefix reserved for that use; other prefixes (ISP specific) may be used

This is called “stateless” operation
How is the v4 address 192.0.2.1 translated to the v6 address 64:ff9b::c000:201?

A. The NAT keeps a translation table
B. It is algorithmic
C. None of the above
D. I don’t know
How does client6 knows that NAT64 should be used?

DNS64 is used in combination with stateful NAT64
DNS64 responds with translated IPv6 address if no AAAA record is found
This is transparent to client6
NAT64, putting things together

IPv6 only host

IPv6

IPv4

NAT64

DNS64

DNS

To: 64:ff9b::c000:201
From: 2001:620:618:dede::baba
Next Header: tcp
Source port: 2345
Dest Port: 80
GET /

To: 192.0.2.1
From: 120.130.26.33
Protocol type: tcp
Source port: 1763
Dest Port: 80
GET /

infosciences.epfl.ch AAAA 64:ff9b::c000:201

A and AAAA infosciences.epfl.ch?
How does NAT64 translate 120.130.26.33 to 2001:620:618:dede::baba?

A. Algorithmically using the 64:ff9b::/96 prefix
B. Algorithmically using a different prefix
C. Using its NAT table and with the help of the port numbers
D. I don’t know
client4 to server6 with NAT64

NAT64 and DNS64 support client6 ↔ server4

The reverse client4 ↔ server6 works with NAT64 with static configuration of NAT64 mapping from server IPv6 address to server’s IPv4 translated address is stateful and requires ad-hoc configuration of NAT64; must be put in DNS as A record. mapping from client4 IPv4 address to client’s IPv6 translated address is stateless

<table>
<thead>
<tr>
<th>IPv6 address and port</th>
<th>IPv4 address and port</th>
<th>NAT64 table</th>
</tr>
</thead>
<tbody>
<tr>
<td>2001:620:618:dede::baba ; port x</td>
<td>120.130.26.33 : port x, static</td>
<td></td>
</tr>
</tbody>
</table>
Pros and Cons of NAT64

+ 
—
All links are Ethernet v2 with MTU = 1500 Bytes. Assume all hosts perform Path-MTU and discover the best possible Path-MTU value. What is the value of Path-MTU at server4 for the interaction with client6?

A. 1500 Bytes
B. 1480 Bytes
C. 1460 Bytes
D. I don’t know
Conclusion

Proxy ARP / NDP is a trick used to solve the problems caused by a subnet present at different locations.

Fragmentation is due to different MAC layers having different packet sizes; MTU is by definition the max length of an IP packet at a given interface. When a packet is too large, it can be either discarded or fragmented. Fragmentation occurs only at IPv6 hosts, IPv4 hosts or IPv4 routers. Re-assembly is never done by routers. Fragmentation may cause problems and should be avoided if possible.

Translation between IPv4 and IPv6 is a possible solution to the h4-h6 interworking problem. It uses NAT64. Translation is stateless between an IPv4 address and a translated IPv6 address; it is stateful between an IPv6 address and a translated IPv4 address. NAT64 is automatic for client6<-> server4 and static for server6 <-> client4. Both require some manipulation of DNS.