Computer Networks

Lecture 7: IP Fragmentation, IPv6, NAT

IP Fragmentation & Reassembly

Network links have MTU (maximum transmission unit) – the largest possible link-level frame

- different link types, different MTUs
- not including frame header/trailer
- but including any and all headers above the link layer

Large IP datagrams are split up ("fragmented") in the network

- each with its own IP header
- fragments are "reassembled" only at final destination (why?)
- IP header bits used to identify and order related fragments

IPv4 Packet Header Format

<table>
<thead>
<tr>
<th>4 bits</th>
<th>4 bits</th>
<th>8 bits</th>
<th>16 bits</th>
</tr>
</thead>
<tbody>
<tr>
<td>version</td>
<td>hdr len (bytes)</td>
<td>Type of Service (TOS)</td>
<td>Total length (bytes)</td>
</tr>
<tr>
<td>Identification</td>
<td>3-bit flags</td>
<td>13-bit Fragment Offset</td>
<td>20-byte Header</td>
</tr>
<tr>
<td>Source IP Address</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Destination IP Address</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Options (if any)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Payload (e.g., TCP/UDP packet, max size?)</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Example: 4000-byte datagram

MTU = 1500 bytes

One large datagram becomes several smaller datagrams

- all but the last fragments must be in multiple of 8 bytes
- offsets are specified in unit of 8-byte chunks
- IP header = 20 bytes (1 header becomes 3 in this example)
Fragmentation Considered Harmful

Reason 1: lose 1 fragment, lose whole packet:
- kernel has limited buffer space
- but IP doesn't know number of fragments per packet

For example:
- sender sends two packets, L and S
  - L is fragmented into 8 fragments
  - S is fragmented into 2 fragments
- receiver has 8 buffer slots
- suppose fragments arrive in the following order:
  - L1, L2, L3, L4, L5, L6, L7, S1, L8, S2
- receiver's buffer fills up after S1, both packets thrown away when reassembly timer times out

Analysis:
- IP doesn't have control over number of fragments
- TCP can do buffer management better because it has more information

Alternatives to fragmentation:
- send only small datagrams (why not?)
- do path MTU discovery and let TCP send the appropriate segment sizes
  - set DF flag
  - router returns ICMP error message (type 3, code 4)
    if fragmentation becomes necessary
- IPv6 enforces minimum MTU of 1280 bytes (576 bytes for IPv4), fragmentation requires fragmentation header

Reason 2: inefficient transmission
Example:
- 10 KB of data
  - sent as 1024 byte TCP segments
  - uses 10 IP packets, each 1064 bytes (TCP/IP headers, each 20 bytes)
  - suppose MTU is 1006 bytes
  - each TCP segment is fragmented into 2 IP packets, of 1,004 bytes and 80 bytes respectively
  - ends up sending 20 packets
- If TCP had sent 960-byte segments, only need to send 11 packets

IPv6

Initial motivation:
32-bit address space exhaustion, increases address size

Additional motivation:
- efficient header format helps speed processing/forwarding
  - header length: removed, use fixed-length 40-byte header (0.07% overhead even for 576-byte packets)
  - header checksum: removed to reduce processing time at each hop
  - options: allowed, but outside of header, indicated by "next header" field
IPv4 Packet Header Format

<table>
<thead>
<tr>
<th>Field</th>
<th>Length (bytes)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Version</td>
<td>4 bits</td>
</tr>
<tr>
<td>Type of Service (TOS)</td>
<td>8 bits</td>
</tr>
<tr>
<td>Total length</td>
<td>16 bits</td>
</tr>
<tr>
<td>Identification</td>
<td>4 bits</td>
</tr>
<tr>
<td>Fragment Offset</td>
<td>12 bits</td>
</tr>
<tr>
<td>Time to Live (TTL)</td>
<td>8 bits</td>
</tr>
<tr>
<td>Protocol</td>
<td>4 bits</td>
</tr>
<tr>
<td>Source IP Address</td>
<td>128 bits</td>
</tr>
<tr>
<td>Destination IP Address</td>
<td>128 bits</td>
</tr>
<tr>
<td>Options</td>
<td>0-128 bytes</td>
</tr>
<tr>
<td>Payload</td>
<td>0-65535 bytes</td>
</tr>
</tbody>
</table>

IPv6

Additional motivation:
- header changes to facilitate Quality of Service (QoS)
  - priority: set priority amongst datagrams in flow (ToS bit)
  - flow label: identify datagrams in the same “flow” (concept of “flow” not well defined, originally these were “reserved” bits)

Next header identifies “upper layer” protocol or IPv6 options:
- hop-by-hop option, destination option, routing, fragmentation, authentication, encryption

IPv6 Address

What does an IPv6 address look like?
- 128 bits written as 8 16-bit integers separated by ‘:’
- each 16-bit integer is represented by 4 hex digits

Example:

Abbreviations:
- actual - 1080:0:0:0:0:0:0:0:0:0:0:0:0:0:0:0:8:800:200C:417A
- skip leading 0’s - 1080:0:0:0:8:800:200C:417A
- double ‘:’ - 1080::8:800:200C:417A
- but not ::BA98:7654::

IPv6 Address Format

<table>
<thead>
<tr>
<th>Field</th>
<th>Length (bytes)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Subnet prefix</td>
<td>64 bits</td>
</tr>
<tr>
<td>Interface identifier</td>
<td>64 bits</td>
</tr>
</tbody>
</table>

Interface identifier: MAC address (globally unique!)
- MAC addresses are 48 bits: add FFE between the 2 halves
- loopback: ::1/128 (only 1 address, not a whole class A block (127/8) as in IPv4)

Subnet prefix: automatically obtained from router
- /32 assigned to Internet Registries (ARIN/RIPE/APNIC), which then dish out smaller address blocks
IPv6 Special Subnet Prefixes

- **Link-local prefix**: FE80::/10 (flush left), not forwarded by router
- **Unique Local Addresses (ULA)**: FC00::/7 routed within a set of cooperating subnets (e.g., networks of the same organization)
- **Multicast addresses**: FF00::/8
- **IPv4 addresses**: ::/96, e.g., IPv4’s 10.0.0.1 can be written as 0:0:0:0:0:A00:1 or ::10.0.0.1

NAT: Network Address Translation

**Motivation**: a stop-gap measure to handle the IPv4 address exhaustion problem

- Share a limited number (\(\geq 1\)) of global, static addresses among a number of local hosts
- Local to global address binding done per connection, on-demand

All datagrams leaving local network have the same source NAT IP address: 138.76.29.7, different (new) source port numbers

Datagrams with source and destination in this network have 10.0.0/24 addresses for source and destination (as usual)

Tunneling

Not all routers can be upgraded simultaneously

- No “flag days”
- How will the network operate with mixed IPv4 and IPv6 routers?

**Tunneling**: IPv6 packets carried as payload in IPv4 datagrams among IPv4 routers

NAT: Example

1: Host 10.0.0.1 sends datagram to 128.119.40.186:80
2: NAT box changes datagram source address from 10.0.0.1:3345 to 138.76.29.7:5001, updates table
3: Reply arrives destination address: 138.76.29.7:5001
4: NAT box changes datagram destination address from 138.76.29.7:5001 to 10.0.0.1:3345
A NAT Box’s Functions

1. Replaces <sourceIP, port#> of every outgoing datagram to <NATIP, newport#>
   • update header checksum
   • remote hosts use <NATIP, newport#> as destination address

2. In NAT translation table, record every mapping of
   <sourceIP, port#> to <NATIP, newport#>

3. Replaces <NATIP, newport#> in destination field of every incoming datagram with corresponding
   <sourceIP, port#> stored in the NAT table
   • update header checksum

4. Forwards modified datagrams into the local network

Why new port#?
Why not simply use the original source port#?

IP Address Space for Private Internets

Three blocks of the IP address space have been reserved for private internets [RFC 1981]:

- 10.0.0.0 - 10.255.255.255 (10/8 prefix)
- 172.16.0.0 - 172.31.255.255 (172.16/12)
- 192.168.0.0 - 192.168.255.255 (192.168/16)

Why must private Internets use reserved address spaces?

Types of NAT

NAT table maps iAddr+iPort of a local host to its eAddr+ePort

1. Full-cone NAT:
   • any remote host can send packets intended for iAddr+iPort to eAddr+ePort

2. IP-restricted NAT:
   • a remote host (rAddr) can send packets to eAddr+ePort only if iAddr+iPort has contacted rAddr (at any remote port, rPort)

3. Port-restricted NAT:
   • a remote host can send packets to eAddr+ePort only using an rPort that iAddr+iPort has contacted at rAddr

Symmetric NAT: eAddr+ePort can only be used by a pre-specified connection, iAddr+iPort+rAddr+rPort

NAT Type Connectivity

<table>
<thead>
<tr>
<th></th>
<th>open</th>
<th>full-cone</th>
<th>IP-restricted</th>
<th>port-restricted</th>
<th>symmetric</th>
<th>UDP-disabled</th>
</tr>
</thead>
<tbody>
<tr>
<td>open</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>full-cone</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>IP-restricted</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>port-restricted</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>symmetric</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>UDP-disabled</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>

table is symmetric along the diagonal

[Shami '09]
NAT Type Distribution

<table>
<thead>
<tr>
<th>Type</th>
<th>Distribution</th>
</tr>
</thead>
<tbody>
<tr>
<td>open</td>
<td></td>
</tr>
<tr>
<td>full-cone</td>
<td></td>
</tr>
<tr>
<td>IP-restricted</td>
<td></td>
</tr>
<tr>
<td>port-restricted</td>
<td></td>
</tr>
<tr>
<td>symmetric</td>
<td></td>
</tr>
<tr>
<td>UDP-disabled</td>
<td></td>
</tr>
</tbody>
</table>

[Shami '09]

NAT Traversal

**STUN (Session Traversal Utilities for NAT):**
- an open server that returns to NATted host the eAddr+ePort used by its NAT box
- also returns the type of the NAT box

**UPnP (Universal Plug and Play):**
- allows internal hosts to add static entries into a UPnP-speaking NAT box's mapping table
- used to traverse full-cone NAT
- NAT box returns eAddr+ePort that internal host can advertise publicly, e.g., when registering with BitTorrent Tracker

**TURN (Traversal Using Relays around NAT):**
- an open server that serves as a relay for a host behind a symmetric NAT to accept connection (from a single host only, i.e., not for NATted host to act as server)
- also useful to traverse traffic-restrictive firewalls

**NAT: Pros**

Can change address of devices in local network without notifying outside world

Devices inside local network not explicitly addressable by or visible to the outside world (security through obscurity)
NAT: Cons

Devices inside local network not explicitly addressable by or visible to the outside world, making peer-to-peer networking that much harder

- routers should only process up to layer 3 (port#'s are application layer objects!)
- port#'s are meant to identify sockets, not end hosts!

Address shortage should be solved by IPv6, instead NAT hinders the adoption of IPv6!

NAT: Lesson

Be careful what you propose as a “temporary” patch

“Temporary” solutions have a tendency to stay around beyond expiration date