Last Time

- RFID
- Some security notions
- IPv6
This Time

- IPv6

IPv6 Packet

- IPv6 Header
- Extension Header
- Extension Header
- Extension Header
- Transport-level PDU

Packet header
40 bytes
always present

0 or more extension headers:
Recommended sequence:
hop-by-hop options, destination options 1, routing, fragment, authentication, ESP, destination options 2

Message data
IPv6 Header

- causes packet delays over network
- can start a cascade of worsening behaviour
  - delayed ACK causes
  - senders to resend resulting in
  - more congestion
- implicit congestion notification:
  - used in IPv4 networks
  - TCP infers existence of congestion and tries to do
    the right thing
- what if were possible to notify explicitly?

Congestion
Explicit Congestion Notification

- routers can alert end-nodes of congestion
  - these nodes can then do the right thing
- quicker response to onset of congestion
  - better response thereto
- reduces retransmissions
  - knowing congestion present, don’t need to re-xmit until infer congestion occurred and then back off
  - congested routers will drop packets: ECN tries to let end nodes know before congestion is so bad that drops occur
- need change to IP to make this do-able
  - also change TCP header

IPv6 Header

<table>
<thead>
<tr>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>VERS</td>
<td>DS</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- Explicit congestion notification[2]:
  - 00 Not ECT — packet not using ECN
  - 01 ECT(1) — set by sender: says TCP
  - 10 ECT(0) — endpoints are ECN capable
  - 11 CE — set by router: tells end node congestion is experienced
Wouldn’t It Be Nice...

- as a sender, you may know certain things about your traffic
  - e.g., how you’d like to be handled by routers along path
  - how can you suggest to routers a preferred way of treating your traffic?
Wouldn’t It Be Nice...

- how can you suggest to routers a preferred way of treating your traffic?
  - put special handling instructions in each packet
  - makes all packets bigger

Handle With ...

- suppose:
  - sequence of packets to travel from src \rightarrow dest
    - e.g., a TCP connection
  - if all these packets are to be handled same way, could put special handling instructions in just one
    - but need way for subsequent packets to be recognized so as to receive same special handling
  - this can happen, e.g.,
    - single app opens \geq 1 TCP connection
Go With The Flow

- IPv6 supports idea of a **flow**: a sequence of packets
  - from single source
  - to a unicast or multicast destination
  - whose sender wants special handling for those packets
- a flow is uniquely defined by combining:
  - source IP address
  - dest IP address
  - flow label

IPv6 Header

```
<table>
<thead>
<tr>
<th></th>
<th></th>
<th>flow label[20]: with src &amp; dest IP addresses uniquely identifies a flow 0 if flow not being used</th>
</tr>
</thead>
<tbody>
<tr>
<td>VERS</td>
<td>DS</td>
<td>ECN</td>
</tr>
</tbody>
</table>
```

---

IT4 41 Network Servers & Infrastructure 21 Nov 2005 © C. M. Snow 2005 12.15
IPv6 Addressing

- all addresses are 128 bits long
- *unicast*: from one sending interface to a destination interface
- *multicast*: a packet sent to a multicast address is delivered to all interfaces indicated by the address
- *anycast*: a packet sent to an anycast address is delivered to one of the interfaces indicated by the address
  - “nearest” one in router-space
- who’s missing?

IPv6 Addresses

- expanded address space not the only reason
  - aim to provide better structuring of addresses so routing tables can be smaller
- addresses are assigned to interfaces, not nodes
  - as with IPv4
- one interface can have multiple IPv6 addresses
  - of any kind: unicast, multicast or anycast
- one IPv6 address can be assigned to multiple interfaces
  - used for load sharing: requires hardware support
### IPv6 Address Format

- **Global Routing Prefix r-bits**
- **Subnet ID s-bits**
- **Interface ID (128 - r - s) bits**

- address range assigned to a site, or, special address e.g., multicast
- a link within a site
- an interface on a link: must be unique on that link

### IPv6 Unicast Addresses

- called “aggregate global unicast address”
- prefix starting with 001 designates globally unique unicast address
- one goal of prefix was to make routing more efficient
  - in part by making routing tables smaller
- invent *top-level aggregation identifier* for use in highest level routing
  - core routers need only 1 entry per TLA
IPv6 Unicast Addresses

- globally unique unicast address
- top level aggregation identifier[13]: identifies 1 of 8192 possible top-level routes assigned as in rfc2450
- next level aggregation identifier[24]: identifies routes established by providers who are expected to further structure this field for optimization of route topology
- site level aggregation identifier[16]: assigned to individual orgs for use in internal routing

IPv6 Multicast Addresses

- any IPv6 address starting with FF

<table>
<thead>
<tr>
<th>flags[4]:</th>
<th>scope[4]:</th>
</tr>
</thead>
<tbody>
<tr>
<td>bits 0-3: RFU/MBZ</td>
<td>0⇒ reserved</td>
</tr>
<tr>
<td>bit 4: 0⇒ well-known mcast addr</td>
<td>1⇒ node-local</td>
</tr>
<tr>
<td>1⇒ temp multicast addr</td>
<td>2⇒ link-local</td>
</tr>
<tr>
<td>3⇒ unassigned</td>
<td>8⇒ org-local</td>
</tr>
<tr>
<td>4⇒ unassigned</td>
<td>9⇒ D ⇒ unassigned</td>
</tr>
<tr>
<td>5⇒ site-local</td>
<td>E ⇒ global</td>
</tr>
<tr>
<td>F ⇒ unassigned</td>
<td></td>
</tr>
</tbody>
</table>
**IPv6 Multicast Example**

- well-known multicast addresses: rfc2375
- an example: group defined for all NTP servers: 101

```
FF01::0:0:0:0:0:101 All NTP servers on same node as sender
FF02::0:0:0:0:0:101 All NTP servers on same link as sender
FF05::0:0:0:0:0:101 All NTP servers at same site as sender
FF06::0:0:0:0:0:101 All NTP servers on Internet
```

---

**Local Addresses**

- IPv4 provided a ‘private’ address space
  - e.g., class C had 192.168.x.x
  - addresses never to be forwarded onto public Internet
- IPv6 provides two categories:
  - link-local address:
    - for use on single link
    - should not cross a router
    - prefix = FE80

```
1111 1110 10 0 interface ID
```

---
Local Addresses

- *site-local* address:
  - for use within a single site
  - contains subnet information
  - can be routed within the site, but not externally
  - prefix = FEC0

<table>
<thead>
<tr>
<th>Subnet ID</th>
<th>interface ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>38</td>
</tr>
<tr>
<td>16</td>
<td>64</td>
</tr>
</tbody>
</table>

IPv6 loopback

- the 127.0.0.1 IPv4 address is valuable for development and testing
- **IPv6 provides analogous address**: 0:0:0:0:0:0:0:1
  - abbreviated ::1
### IPv6 Header

<table>
<thead>
<tr>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
</tr>
</thead>
<tbody>
<tr>
<td>VERS</td>
<td>DS</td>
<td>ECN</td>
<td>FLOW LABEL</td>
<td>PAYLOAD LENGTH</td>
<td>NEXT HDR</td>
<td>HOP LIMIT</td>
<td>EXTENSION HEADER</td>
<td>DATA</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Option Headers in IPv6

- **EXTENSION HEADER**
- **NEXT HDR**
- **EXT HDR LEN**
- **OPTIONS ...**

- identifies next header uses RFC1700 numbering
- length of this option in 64-bit words not including first 64 bits
Option Headers in IPv6

EXTENSION HEADER

NEXT HDR  EXT HDR LEN  OPTIONS ...

OPT TYPE  OPT LEN

can option data change en-route?
0 no
1 yes

if option type is unrecognized do:
00 skip this option, keep processing header
01 discard packet
10 discard packet and send ICMP msg to source
11 as 10 but only if dest addr was not multicast addr
Hop by Hop Option Header

- every node along delivery path must examine
- identified with Next Header value of 0
- 4 option types defined:
  - PAD1: add 1 byte of padding
  - PADN: add \( n \) bytes of padding
  - JUMBO: packet is bigger than 64K
    - so overall length field in header must be 0
    - 32-bit size, so packet can be up to 232 bytes long
  - ROUTER ALERT: packet contents of interest to router
    - if absent, tells router to route without further packet parsing

```
<table>
<thead>
<tr>
<th>NEXT HDR</th>
<th>EXT HDR LEN</th>
<th>OPT TYPE</th>
</tr>
</thead>
</table>
```

Hop by Hop Option Header

- for Pad1, no option length or data bytes
  - what is EXT HDR LEN value?

```
<table>
<thead>
<tr>
<th>NEXT HDR</th>
<th>EXT HDR LEN</th>
<th>OPT TYPE</th>
</tr>
</thead>
</table>
```

- for PadN, inserts \( N \geq 2 \) padding bytes
  - use this whenever inserting more than 1 pad byte

```
<table>
<thead>
<tr>
<th>NEXT HDR</th>
<th>EXT HDR LEN</th>
<th>OPT TYPE</th>
<th>OPT LEN=N-2</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

N-2 0-valued bytes...
Routing Extension Header

- source routing provides a way to specify nodes packet is supposed to visit on its route to dest
  - not guaranteed
- IPv6 achieves with Routing Extension Header
  - identified by Next Header = 43 in preceding header

<table>
<thead>
<tr>
<th>NEXT_HDR</th>
<th>EXT HDR LEN</th>
<th>ROUTING TYPE</th>
<th>SGMNTS LEFT</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>type-specific data</td>
</tr>
</tbody>
</table>

- data determined by routing type: variable-length but a multiple of 8 bytes
- identifies particular routing header variant (8-bit uint)
- number of intermediate nodes left to visit before arriving at dest node (8-bit uint)

Routing Type 0

- only routing type 0 is currently defined

<table>
<thead>
<tr>
<th>NEXT_HDR</th>
<th>EXT HDR LEN</th>
<th>ROUTING TYPE</th>
<th>SGMNTS LEFT</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>RFU</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>address 1</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>address 2</td>
<td></td>
</tr>
</tbody>
</table>

| *** |
Using the REH: Routing Type 0

- sender X sends packet to dest Y via A, B, C, D
- sender X puts A’s address as dest address in hdr
- nodes B, C, D, Y are listed in REH
- at each node along path:
  - take next node off list in REH and put it in hdr’s dest
  - decrement segments left
- multicast addresses not allowed
- REH is not examined except by packet whose address appears in main header dest address

REH: Unknown Routing Types

- what to do if routing type is unknown:
  - if segments_left = 0: ignore REH and process the next header
  - else if segments_left ≠ 0:
    - discard packet
    - send ICMP “Parameter Problem” msg to sender
- packet also dumped if intermediate node discovers must route onto network whose MTU is < packet size
  - discard packet
  - send ICMP “Packet Too Big” msg to sender
Fragmentation

- necessary so packets can travel along path with segments of differing MTUs
  - packet ends up no bigger than smallest MTU (path MTU)
- IPv4 requires routers to fragment-on-demand
  - and ultimate dest does re-assembly
- IPv6 is different...

IPv6 Fragmentation

- burden is on sender to generate packet no bigger than path MTU
- routers don’t do fragmentation
- source has two choices:
  - generate packets no bigger than IPv6 absolute minimum (1280 octets)
  - discover the path MTU for its traffic and fragment packet accordingly
### Fragmentation Header

- identified by Next Header value of 44

![Diagram of Fragmentation Header]

- Identification [32]: a distinct value that no other fragmented packet recently sent from this sender to the final dest node
- Fragment Offset [13]: offset in 8-octet units where data after this header fits into original packet

### More Complicated Fragmentation

- IPv6 packets, if fragmented, are fragmented by source node
- but original packet may have extension headers of importance, so view as:

![Diagram of More Complicated Fragmentation]

- The unfragmentable part appears in every fragment: unchanged or modified from original contents?
What Could Go Wrong?

1) how long should dest wait for all the fragments?

- rfc2460 says 60 seconds of first-arriving frag
- if packet cannot be reconstituted (not all frags have arrived) then:
  - discard all received fragments of the packet
  - send ICMP “Time Exceeded” error to sender
What Else Could Go Wrong?

2) if (length of fragment is not a multiple of 8) and
   (M flag = 1)
   - discard this fragment
   - send ICMP “Parameter Problem” error to sender

3) if re-assembly with this fragment makes a packet
   > 65535 bytes long then:
   - discard this fragment
   - send ICMP “Parameter Problem” error to sender
Options Not For Hop By Hop

- hop-by-hop header lets us provide information to intermediate nodes between source and dest
  - padding (1, N bytes, jumbo pkt, router alert)
- does dest node see these?
- what if we want to provide info only to dest node?

Destination Options Header

- provides option info that only the dest node(s) will examine
- identified by Next Header = 60
- same format as hop-by-hop options header

- how is use of dest options hdr different from, e.g., using fragment header?
No More

- what if there is no next header after the current one?
  - what value goes into the Next Header field of this header?

No More

- what if there is no next header after the current one?
  - what value goes into the Next Header field of this header?
  - 59 indicates there is nothing following this header
  - what if there is?
    - what if payload length field implies there must be bytes after the current header?
No More

- what if there is?
  - what if payload length field implies there must be bytes after the current header?
  - ignore them
    - forward them unchanged if packet is being forwarded

Preferred Order

- extension headers can appear in any order, but rfc2460 indicates a preferred order is:
  - hop-by-hop options
  - routing (type 0)
  - fragment
  - dest options
  - authentication
  - encapsulating security payload
IPv6 Security

- IPv6 provides built-in mechanisms for security
  - not found in IPv4
- uses extension headers to specify
- some background info first...

IPSEC

- works at IP layer
- but is connection-oriented
- each connection is a Security Association (SA)
  - simplex connection
  - need 2 SAs for two-way communication
- 2 modes:
  1. transport: ‘seen’ by nodes between sender and dest
  2. tunnel: ‘seen’ only by sender and dest
- encryption “always on”
  - but IPsec doesn’t dictate particular algorithms
  - can use “null” algorithm for no encryption (RFC 2410)
Security Associations

- ‘secure’ connection from sender to receiver
- parameters negotiated at set-up
  - e.g., key
- database entry at receiver has parms for each SA
- SAs uniquely identified by:
  1. Security Parameters Index (SPI)
  2. IP destination address
  3. security protocol identifier

SA characteristics

- sequence number: 32–bit counter
- sequence counter o’flow: does counter overflow trigger “auditable” event and lock-out this SA?
- anti-replay window: range of allowed (expected) sequence numbers
- lifetime: time interval or byte-count after which SA is no longer valid
- IPSec protocol mode: transport or tunnel
- **Path MTU**: max size that doesn’t need fragmenting
- Hdr specific parms: see header descriptions
Components of IPsec

3 main components:

1. AH: Authentication Header
   - integrity: can’t make undetected changes to msg
   - end receiver has ability to authenticate
   - eliminate spoof attacks
   - guard against replay attacks

2. ESP: Encapsulating Security Payload
   - confidentiality of msg contents
   - some traffic flow confidentiality

3. key management mechanism

AH Format
Authentication Data

- variable length
- ICV: integrity check value
- message authentication code (MAC):
  - sender & receiver share a secret key
  - sender computes a value from the message
  - encrypts value with secret key
- must be multiple of 32-bit size