IP Next Generation (IPv6)

Shivkumar Kalyanaraman
Rensselaer Polytechnic Institute
shivkuma@ecse.rpi.edu
http://www.ecse.rpi.edu/Homepages/shivkuma

Based in part upon slides of Prof. Raj Jain (OSU), S.Deering (Cisco), C. Huitema (Microsoft)
Overview

- Limitations of current Internet Protocol (IP)
- How many addresses do we need?
- IPv6 Addressing
- IPv6 header format
- IPv6 features: routing flexibility, plug-n-play, multicast support, flows
- application-layer gateways
  - difficult to deploy new internet-wide applications
  - hard to diagnose and remedy end-to-end problems
  - stateful gateways => hard to route around failures

- no global addressability
  - ad-hoc, application-specific solutions
The IP Solution ...

- internet-layer gateways & global addresses
- simple, application-independent, lowest denominator network service: best-effort datagrams
- stateless gateways could easily route around failures
- with application-specific knowledge out of gateways:
  - NSPs no longer had monopoly on new services
  - Internet: a platform for rapid, competitive innovation
The Internet Today: with NATs

- network address translators and app-layer gateways
- hard to diagnose and remedy end-to-end problems
- stateful gateways inhibit dynamic routing around failures
- no global addressability => brokered with NATs
- new Internet devices more numerous, and may not be adequately handled by NATs (e.g., mobile nodes)
Address Shortage Causes More NAT Deployment

Address exhaustion date estimate varies from 2009-2019!
IPv4 Addresses

- **Example**: 164.107.134.5
  - \(= 1010\ 0100\ : \ 0110\ 1011\ : \ 1000\ 0110\ : \ 0000\ 0101\)
  - \(= \text{A4:6B:86:05}\) (32 bits)

- Maximum number of address = \(2^{32} = 4\) Billion

- Class A Networks: 15 Million nodes
- Class B Networks: 64,000 nodes or less
- Class C Networks: 250 nodes or less
- Class B very popular…

- Total allocated address space as seen by routing: ~1Billion
How Many Addresses?

- 10 Billion people by 2020
- Each person has more than one computer
- Assuming 100 computers per person ⇒ 10^{12} computers
- More addresses may be required since
  - Multiple interfaces per node
  - Multiple addresses per interface
  - Some believe 2^6 to 2^8 addresses per host
- Safety margin ⇒ 10^{15} addresses

**IPng Requirements** ⇒ 10^{12} end systems and 10^9 networks. Desirable 10^{12} to 10^{15} networks
How big an address space?

- **H Ratio** = \( \frac{\log_{10}(\text{# of objects})}{\text{available bits}} \)
- \( 2^n \) objects with \( n \) bits: \( H\)-Ratio = \( \log_{10}2 = 0.30103 \)
- French telephone moved from 8 to 9 digits at \( 10^7 \) households \( \Rightarrow H = 0.26 \) (\( \sim3.3 \) bits/digit)
- US telephone expanded area codes with \( 10^8 \) subscribers \( \Rightarrow H = 0.24 \)
- Physics/space science net stopped at 15000 nodes using 16-bit addresses \( \Rightarrow H = 0.26 \)
- 3 Million Internet hosts currently using 32-bit addresses \( \Rightarrow H = 0.20 \)

- Huitema estimated \( H = 0.26 \) for Nov 2002
IPv6 Addresses

- 128-bit long. Fixed size
- \(2^{128} = 3.4 \times 10^{38}\) addresses
  \(\Rightarrow 665 \times 10^{21}\) addresses per sq. m of earth surface
- If assigned at the rate of \(10^6/\mu s\), it would take 20 years
- Expected to support \(8 \times 10^{17}\) to \(2 \times 10^{33}\) addresses
  \(8 \times 10^{17} \Rightarrow 1,564\) address per sq. m

- Allows multiple interfaces per host.
- Allows multiple addresses per interface
- Allows unicast, multicast, anycast
- Allows provider based, site-local, link-local
- 85% of the space is unassigned
Colon-Hex Notation

- **Dot-Decimal**: 127.23.45.88
- **Colon-Hex**: FEDC:0000:0000:0000:3243:0000:0000:ABCD
  - Can skip leading zeros of each word
  - Can skip one sequence of zero words, e.g., FEDC::3243:0000:0000:ABCD or ::3243:0000:0000:ABCD
  - Can leave the last 32 bits in dot-decimal, e.g., ::127.23.45.88
  - Can specify a prefix by /length, e.g., 2345:BA23:7::/40
## Header

### IPv6:

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Version</td>
<td>IPv6 version number</td>
</tr>
<tr>
<td>Class</td>
<td>IPv6 traffic class (0 = data, 1 = control)</td>
</tr>
<tr>
<td>Flow Label</td>
<td>IPv6 flow label</td>
</tr>
<tr>
<td>Payload Length</td>
<td>Payload length</td>
</tr>
<tr>
<td>Next Header</td>
<td>Next header type</td>
</tr>
<tr>
<td>Hop Limit</td>
<td>Hop limit</td>
</tr>
<tr>
<td>Source Address</td>
<td>Source address</td>
</tr>
<tr>
<td>Destination Address</td>
<td>Destination address</td>
</tr>
</tbody>
</table>

### IPv4:

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Version</td>
<td>IPv4 version number</td>
</tr>
<tr>
<td>IHL</td>
<td>Internet header length</td>
</tr>
<tr>
<td>Type of Service</td>
<td>Type of service</td>
</tr>
<tr>
<td>Total Length</td>
<td>Total length</td>
</tr>
<tr>
<td>Identification</td>
<td>Identification number</td>
</tr>
<tr>
<td>Flags</td>
<td>Flags</td>
</tr>
<tr>
<td>Fragment Offset</td>
<td>Fragment offset</td>
</tr>
<tr>
<td>Time to Live</td>
<td>Time to live</td>
</tr>
<tr>
<td>Protocol</td>
<td>Protocol number</td>
</tr>
<tr>
<td>Header Checksum</td>
<td>Header checksum</td>
</tr>
<tr>
<td>Source Address</td>
<td>Source address</td>
</tr>
<tr>
<td>Destination Address</td>
<td>Destination address</td>
</tr>
<tr>
<td>Options</td>
<td>Options</td>
</tr>
<tr>
<td>Padding</td>
<td>Padding</td>
</tr>
</tbody>
</table>
### The IPv4 Header

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Version</td>
<td>4 bits that indicate the version of IPv4.</td>
</tr>
<tr>
<td>Hdr Len</td>
<td>4 bits that indicate the length of the header.</td>
</tr>
<tr>
<td>Prec</td>
<td>8 bits that indicate the precedence of the packet.</td>
</tr>
<tr>
<td>TOS</td>
<td>8 bits that indicate the type of service.</td>
</tr>
<tr>
<td>Total Length</td>
<td>16 bits that indicate the total length of the packet.</td>
</tr>
<tr>
<td>Identification</td>
<td>16 bits that identify the packet.</td>
</tr>
<tr>
<td>Flags</td>
<td>8 bits that indicate flags.</td>
</tr>
<tr>
<td>Fragment Offset</td>
<td>13 bits that indicate the fragment offset.</td>
</tr>
<tr>
<td>Time to Live</td>
<td>8 bits that indicate time to live.</td>
</tr>
<tr>
<td>Protocol</td>
<td>8 bits that indicate the protocol.</td>
</tr>
<tr>
<td>Header Checksum</td>
<td>16 bits that indicate the header checksum.</td>
</tr>
<tr>
<td>Source Address</td>
<td>32 bits that indicate the source address.</td>
</tr>
<tr>
<td>Destination Address</td>
<td>32 bits that indicate the destination address.</td>
</tr>
<tr>
<td>Options</td>
<td>Fields that can be used for additional purposes.</td>
</tr>
<tr>
<td>Padding</td>
<td>Fields that are padding.</td>
</tr>
</tbody>
</table>

Shaded fields are absent from IPv6 header.
IPv6 vs IPv4

- IPv6 twice the size of IPv4 header
- Version: only field w/ same position and meaning

**Removed:**
- Header length, fragmentation fields (identification, flags, fragment offset), header checksum

**Replaced:**
- Datagram length by payload length
- Protocol type by next header
- Time to live by hop limit
- Type of service by “class” octet

**Added:** flow label

- All fixed size fields.

- No optional fields. Replaced by extension headers.

  - Idea: avoid unnecessary processing by intermediate routers w/o sacrificing the flexibility
Most extension headers are examined only at destination

- Eg: Base Header and Two Extension Headers:
  - Base Header
    - Next = Route
  - Route Header
    - Next = Auth
  - Auth Header
    - Next = TCP
  - TCP Segment
Fragmentation

- Routers cannot fragment. Only source hosts can.
  ⇒ Need path MTU discovery or tunneling
- Fragmentation requires an extension header
- Payload is divided into pieces
- A new base header is created for each fragment

<table>
<thead>
<tr>
<th>Base Header</th>
<th>Frag. 1 Header</th>
<th>Part 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>New Base Header</td>
<td>Frag. 2 Header</td>
<td>Part 2</td>
</tr>
<tr>
<td>New Base Header</td>
<td>Frag. n Header</td>
<td>Part n</td>
</tr>
</tbody>
</table>
### Initial IPv6 Prefix Allocation

<table>
<thead>
<tr>
<th>Allocation</th>
<th>Prefix</th>
<th>Allocation</th>
<th>Prefix</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reserved</td>
<td>0000 0000</td>
<td>Unassigned</td>
<td>101</td>
</tr>
<tr>
<td>Unassigned</td>
<td>0000 0001</td>
<td>Unassigned</td>
<td>110</td>
</tr>
<tr>
<td>NSAP</td>
<td>0000 001</td>
<td>Unassigned</td>
<td>1110</td>
</tr>
<tr>
<td>IPX</td>
<td>0000 010</td>
<td>Unassigned</td>
<td>1111 0</td>
</tr>
<tr>
<td>Unassigned</td>
<td>0000 011</td>
<td>Unassigned</td>
<td>1111 10</td>
</tr>
<tr>
<td>Unassigned</td>
<td>0000 1</td>
<td>Unassigned</td>
<td>1111 110</td>
</tr>
<tr>
<td>Unassigned</td>
<td>0001</td>
<td>Unassigned</td>
<td>1111 1110</td>
</tr>
<tr>
<td>Unassigned</td>
<td>001</td>
<td>Unassigned</td>
<td>1111 1110 0</td>
</tr>
<tr>
<td>Provider-based*</td>
<td>010</td>
<td>Link-Local</td>
<td>1111 1110 10</td>
</tr>
<tr>
<td>Unassigned</td>
<td>011</td>
<td>Site-Local</td>
<td>1111 1110 11</td>
</tr>
<tr>
<td>Geographic</td>
<td>100</td>
<td>Multicast</td>
<td>1111 1111</td>
</tr>
</tbody>
</table>

*Has been renamed as “*Aggregatable global unicast*”
Aggregatable Global Unicast Addresses

Address allocation: "provider-based" plan
Format: TLA + NLA + SLA + 64-bit interface ID

TLA = “Top level aggregator.”
For “backbone” providers or “exchange points”

NLA = “Next Level Aggregator”
Second tier provider and a subscriber

SLA = “Site level aggregator”

Re-numbering: change of provider => change the TLA and NLA. But have same SLA & I/f ID.
Aggregatable Global Unicast Addresses (Continued)

- Interface ID = 64 bits
  - Will be based on IEEE EUI-64 format
  - An extension of the IEEE 802 (48 bit) format.
  - Possible to derive the IEEE EUI-64 equivalent of current IEEE 802 addresses

```
001  TLA  NLA*   SLA*   interface ID
     public topology (45 bits) site topology (16 bits) interface identifier (64 bits)
```
IPv6 Routing architecture

Provider, Exchange

TOP

Next level

Next level

Next level

Site

Link

Host
Local-Use Addresses

- **Link Local**: Not forwarded outside the link, FE:80::xxx
  - Auto-configuration and when no routers are present
  
  \[
  \begin{array}{c|c|c}
  10 \text{ bits} & n \text{ bits} & 118-n \\
  \hline
  1111 & 1110 & 10 \end{array}
  \]

- **Site Local**: Not forwarded outside the site, FE:C0::xxx
  - Independence from changes of TLA / NLA*
  
  \[
  \begin{array}{c|c|c|c}
  10 \text{ bits} & n \text{ bits} & m \text{ bits} & 118-n-m \text{ bits} \\
  \hline
  1111 & 1110 & 11 & 0 \end{array}
  \]

- **Provides plug and play**
Multicast Addresses

- low-order flag indicates permanent / transient group; three other flags reserved
- scope field: 1 - node local, 2 - link-local, 5 - site-local, 8 - organization-local, B - community-local, E - global (all other values reserved)
- All IPv6 routers will support native multicast
Eg: Multicast Scoping

- **Scoping. Eg: 43 ⇒ NTP Servers**
  - FF01::43 ⇒ All NTP servers on this node
  - FF02::43 ⇒ All NTP servers on this link
  - FF05::43 ⇒ All NTP servers in this site
  - FF08::43 ⇒ All NTP servers in this org.
  - FF0F::43 ⇒ All NTP servers in the Internet

- **Structure of Group ID:**
  - First 80 bits = zero (to avoid risk of group collision, because IP multicast mapping uses only 32 bits)
Address Auto-configuration

- Allows plug and play
- BOOTP and DHCP are used in IPv4
- DHCPng will be used with IPv6
- Two Methods: Stateless and Stateful
- Stateless:
  - A system uses *link-local* address as source and multicasts to "All routers on this link"
  - Router replies and provides all the needed prefix info
Address Auto-configuration (Continued)

- All prefixes have an associated *lifetime*
- System can use link-local address permanently if no router

- **Stateful:**
  - Problem w stateless: Anyone can connect
  - Routers ask the new system to go DHCP server (by setting managed configuration bit)
  - System multicasts to "All DHCP servers"
  - DHCP server assigns an address
ICMPv6: Neighbor Discovery

- ICMPv6 combines regular ICMP, ARP, Router discovery and IGMP.

- The “neighbor discovery” is a generalization of ARP & router discovery.

- Source maintains several caches:
  - **destination cache**: dest -> neighbor mapping
  - **neighbor cache**: neighbor IPv6 -> link address
  - **prefix cache**: prefixes learnt from router advertisements
  - **router cache**: router IPv6 addresses
Neighbor Discovery (Continued)

- Old destination => look up destination cache
- If new destination, **match the prefix cache**. If match => destination local!
- Else select a router from **router cache**, use it as the next-hop (neighbor).
  - Add this neighbor address to the **destination cache**
- **Solicitation-advertisement** model:
  - Multicast **solicitation** for neighbor media address if unavailable in **neighbor cache**
  - Neighbor **advertisement** message sent to soliciting station.
IPv6 Auto-configuration: 7 problems

1. End-node acquires L3 address:
   - Use link-local address as src and multicast query for advts
   - Multiple prefixes & router addresses returned
2. Router finds L3 address of end-node: same net-ID
3. Router finds L2 address of end-node: neighbor discovery (generalization of ARP, w/ several caches)
4. End-nodes find router: solicit/listen for router advt
5. End-nodes send directly to each other: same prefix (prefix cache) => direct
6. Best router discovery: ICMPv6 redirects
7. Router-less LAN: same prefix (prefix cache) => direct. Link-local addresses + neighbor discovery if no router.

- Integrated several techniques from CLNP, IPX, Appletalk etc
Auto-Reconfiguration ("Renumbering")

- Problem: providers changed => old-prefixes given back and new ones assigned **THROUGHOUT** the site
- Solution:
  - we assume some *overlap period* between old and new, i.e., no "flash cut-over"
  - hosts learn prefix lifetimes from router advertisements
  - old TCP connections can survive until end of overlap; new TCP connections can survive beyond overlap

- Router renumbering protocol, to allow domain-interior routers to learn of prefix introduction / withdrawal
- New DNS structure to facilitate prefix changes
Other Features of IPv6

- Flow label for more efficient flow identification (avoids having to parse the transport-layer port numbers)
- Neighbor un-reachability detection protocol for hosts to detect and recover from first-hop router failure
- More general header compression (handles more than just IP+TCP)
- Security ("IPsec") & differentiated services ("diffserv") QoS features — same as IPv4
If IPv6 is so great, how come it is not there yet?

- Applications
  - Need upfront investment, stacks, etc.
- Network
  - Similar to Y2K, 32 bit vs. “clean address type”
  - Need to ramp-up investment
  - No “push-button” transition
Transition Issues: Protocol upgrades

- Most application protocols will have to be upgraded: FTP, SMTP, Telnet, Rlogin
- Several full standards revised for IPv6
- Non-IETF standards: X-Open, Kerberos, ... will be updated... Hosts, routers ... the works!

- With a suite of “fixes” to IPv4, what is compelling in IPv6?
  - **Sticks**: tight address allocation (3G going to IPv6), NAT becomes too brittle...
  - **Incentives (carrots)**: stateless autoconf simplifies mobility, if p2p and multimedia grow, then NATs may pose a problem
Transition Mechanisms

- 1. Recognize that IPv4 will co-exist with IPv6 indefinitely
- 2. Recognize that IPv6 will co-exist with NATs for a while
- **Dual-IP** Hosts, Routers, Name servers
- **Tunneling** IPv6-over-IPv4 \( (\text{6-over-4}) \), IPv4 as link \( (\text{6-to-4}) \)
- **Translation**: allow IPv6-only hosts to talk to IPv4-only hosts

![Diagram showing IPv4 and IPv6 co-existence mechanisms](image-url)
IPv4-IPv6 Co-Existence / Transition

Three categories:

1. **dual-stack techniques**, to allow IPv4 and IPv6 to co-exist in the same devices and networks

2. **tunneling techniques**, to avoid order dependencies when upgrading hosts, routers, or regions

3. **translation techniques**, to allow IPv6-only devices to communicate with IPv4-only devices

expect all of these to be used, in combination
Dual-Stack Approach

- When adding IPv6 to a system, do not delete IPv4
  - this multi-protocol approach is familiar and well-understood (e.g., for AppleTalk, IPX, etc.)
  - IPv6 will be bundled with new OS releases
- Applications (or libraries) choose IP version to use
  - when initiating, based on DNS response:
    if (dest has AAAA or A6 record) use IPv6, else use IPv4
  - when responding, based on version of initiating packet
- This allows indefinite co-existence of IPv4 and IPv6, and gradual, app-by-app upgrades to IPv6 usage
Tunnels

- Encapsulate IPv6 inside IPv4 packets (or MPLS).

  - **Manual** configuration
  - “Tunnel brokers” (using web-based service to create a tunnel)
  - “6-over-4” (intra-domain, using IPv4 multicast as virtual LAN)
  - “6-to-4” (inter-domain, using IPv4 addr as IPv6 site prefix)

- can view this as:
  - IPv6 using IPv4 as a virtual link-layer, or
  - an IPv6 VPN (virtual public network), over the IPv4 Internet
  (becoming “less virtual” over time)
6to4
Automated tunneling across IPv4…

Pure “Version 6” Internet

Original “Version 4” Internet

6to4 Site
1 v4 address = 1 v6 network
6to4 Site
6to4 addresses:

1 v4 address = 1 v6 network

- **Stateless tunnel** over the IPv4 network **without configuration**
  - The IPv6 address contains the IPv4 address
  - Entire campus infrastructure fits behind single IPv4 address

- 6to4 router derives IPv6 prefix from IPv4 address,
6to4: tunnel IPv6 over IPv4

- 6to4 router derives IPv6 prefix from IPv4 address,
- 6to4 relays advertise reachability of prefix 2002::/16
- Automatic tunneling from 6to4 routers or relays
- Single address (192.88.99.1) for all relays
Network Address Translation and Protocol Translation (NAT-PT)

IPv6-only devices

IPv4-only and dual-stack devices

NAT-PT
RSIP-based evolution leads to IPv6

IPv4
IPv4+ NAT
IPv4+ RSIP
IPv6+ RSIP
IPv6

Crisis
Broken...
Future proof...
Backbone...

Unlikely direction…
Since RSIP is not gaining traction

Shivkumar Kalyanaraman
Standards

- Core IPv6 specifications are IETF Draft Standards => well-tested & stable
  - IPv6 base spec, ICMPv6, Neighbor Discovery, Multicast Listener Discovery, PMTU Discovery, IPv6-over-Ethernet,...
- Other important specs are in good shape
  - mobile IPv6, header compression, A6 DNS support, IPv6-over-NBMA,...
- for up-to-date status: playground.sun.com / ipng
- 3GPP cellular wireless standards to mandate IPv6
6-bone, 6ren

- Experimental infrastructure: **the 6bone**
  - for testing and debugging IPv6 protocols
  - mostly IPv6-over-IPv4 tunnels
  - > 200 sites in 42 countries; mostly universities, network research labs, and IP vendors

- Production infrastructure in support of education and research: **the 6ren**
  - CAIRN, Canarie, CERNET, Chunahwa Telecom, Dante, ESnet, Internet 2, IPFNET, NTT, Renater, Singren, Sprint, SURFnet, vBNS, WIDE
  - a mixture of native and tunneled paths
  - see www.6ren.net, www.6tap.net
Incentive: Peer-to-peer applications?
Problem: Peer-to-peer RTP audio example

- With NAT:
  - Need to learn the address “outside the NAT”
  - Provide that address to peer
  - Need either NAT-aware application, or application-aware NAT
  - May need a third party registration server to facilitate finding peers
Solution: Peer-to-peer RTP audio example

- With IPv6:
  - Just use IPv6 address
Key drivers? Parting thoughts …

- **Always-on** requirement => large number of actively connected nodes online
- **3G, internet appliances**
  - large numbers of addresses needed in short order…
  - IPv6 auto-configuration and mobility model better
  - 3GPP already moving towards IPv6
- **P2P apps and multimedia** get popular and NAT/ALGs/Firewalls break enough of them
- **Multi-homed sites and traffic engineering** hacks in BGP/IPv4 make inter-domain routing un-scalable
- **Dual stack, simpler auto-conf, automatic tunneling (6to4 etc)** simplify migration path and provide installed base
  - Applications slowly start **self-selecting IPv6**
Summary

- IPv6 uses 128-bit addresses
- Allows provider-based, site-local, link-local, multicast, anycast addresses
- Fixed header size. Extension headers instead of options for provider selection, security etc
- Allows auto-configuration
- Dual-IP, 6-to-4 etc for transition