IPv6

- Background
- Technology Overview
- Deployment Strategies
- Current Status

IPv4

- IPv4 is still very much with us
- A child of the 1980s - predates the Web
- 4 billion addresses (seemed like enough at the time)
- Addressing extended by NAT
- Complex header slows routers down with re-calculations of header checksum every hop

Why not IPv6?

- Network Address Translation (NAT)
- Application Level Gateways (ALG)
- Classless InterDomain Routing (CIDR)
- DHCP for IPv4
- IPsec for IPv4
- Mobility for IPv4
- Multiprotocol Label Switching (MPLS)

Why not IPv6?

```
http://2002:09fe:fdfc:2a41:
  fe21:08c9:e133:fe01/index.html
```

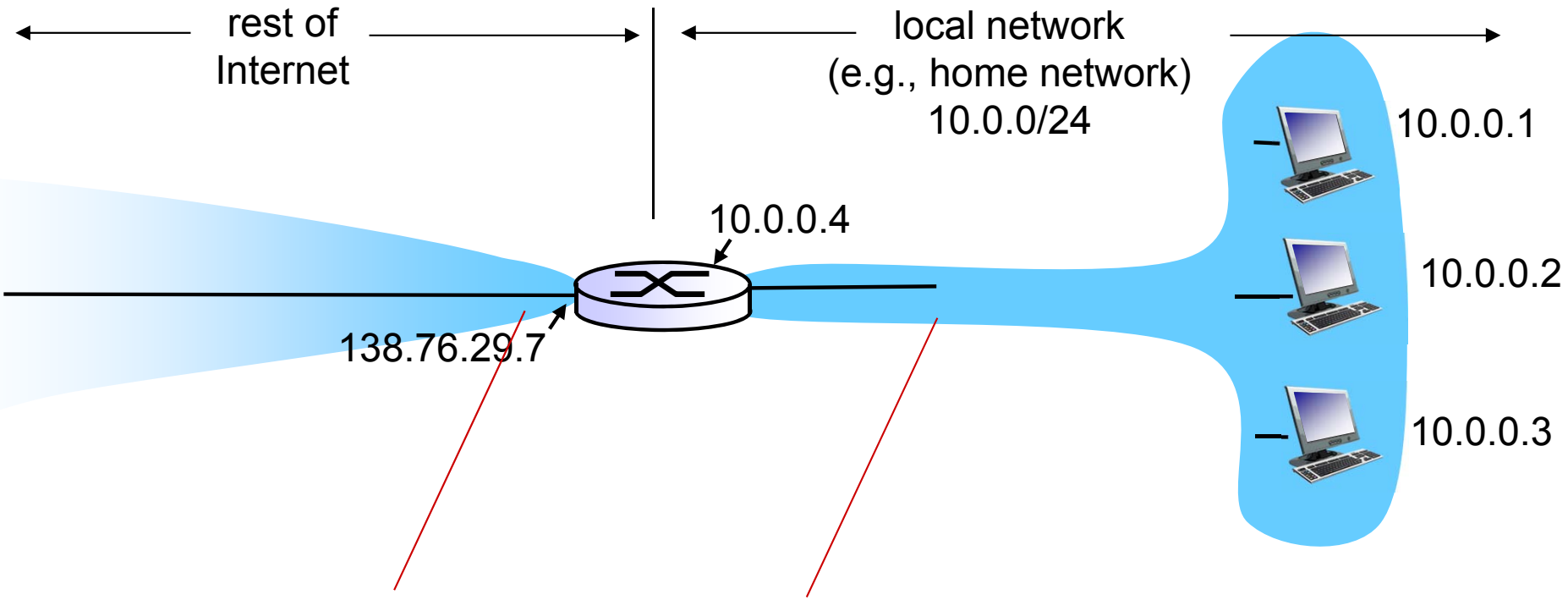
Interim Solution

- CIDR - Classless InterDomain Routing
  - Drop address classes (A, B, C)
  - Assign addresses in power-of-two chunks
    - Assign several Class C addresses instead of one Class B address
    - Assign providers large contiguous address block to be used for their customers
  - Advertise chunks instead of individual address assignments

Buying More Time

- CIDR has helped
  - Using Class C addresses more heavily
  - Reducing need for Class B addresses
  - Address aggregation slowing routing table growth
- Routing table growth still a problem

NAT: network address translation



all datagrams *leaving* local network have *same* single source NAT IP address: 138.76.29.7,different source port numbers

datagrams with source or destination in this network have 10.0.0/24 address for source, destination (as usual)

NAT: network address translation

motivation: local network uses just one IP address as far as outside world is concerned:

- range of addresses not needed from ISP: just one IP address for all devices
- can change addresses of devices in local network without notifying outside world
- can change ISP without changing addresses of devices in local network
- devices inside local net not explicitly addressable, visible by outside world (a security plus)

NAT: network address translation

implementation: NAT router must:

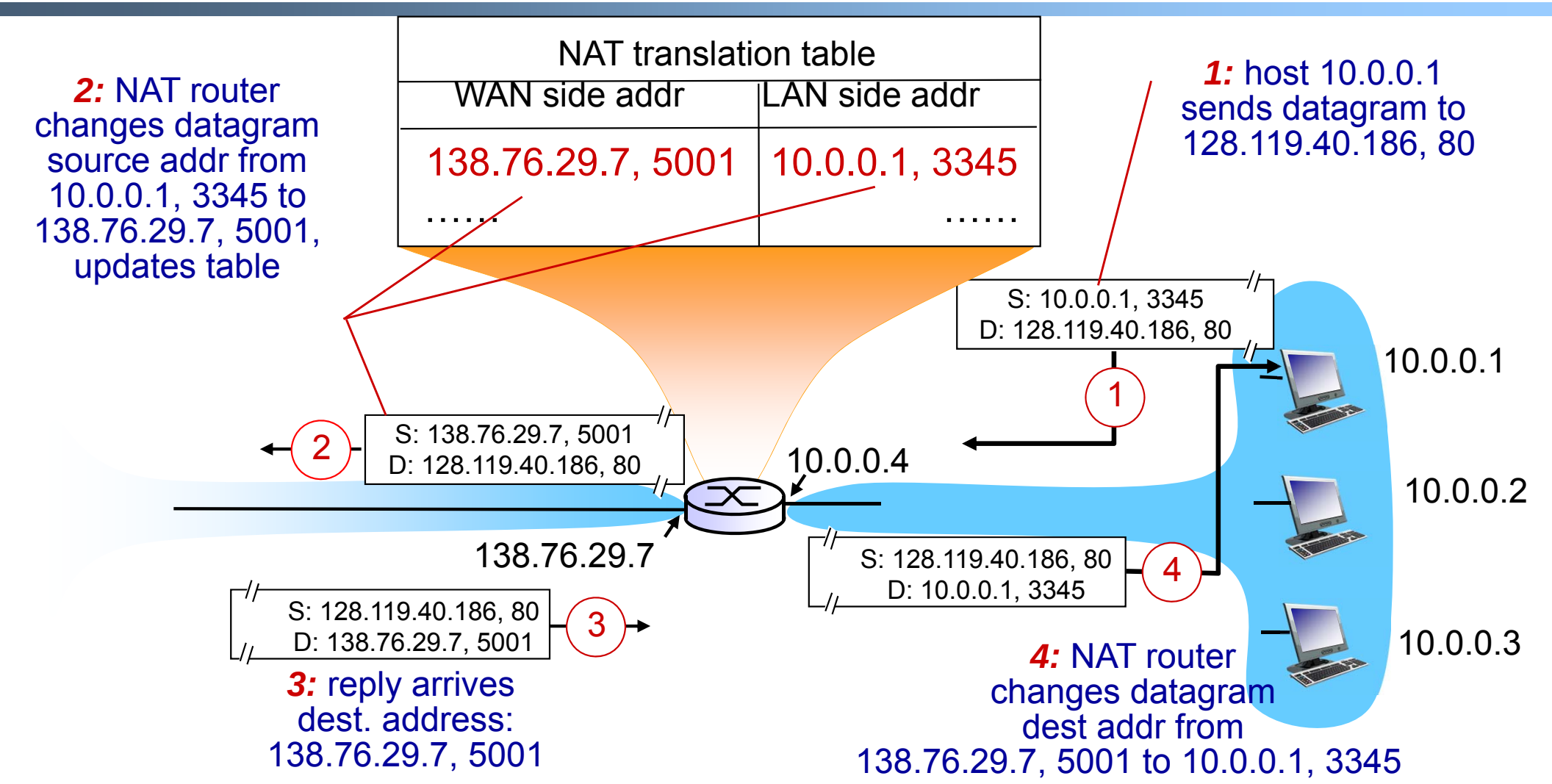
- *outgoing datagrams: replace* (source IP address, port #) of every outgoing datagram to (NAT IP address, new port #)

  . . . remote clients/servers will respond using (NAT IP address, new port #) as destination addr

- *remember (in NAT translation table)* every (source IP address, port #) to (NAT IP address, new port #) translation pair

- *incoming datagrams: replace* (NAT IP address, new port #) in dest fields of every incoming datagram with corresponding (source IP address, port #) stored in NAT table

NAT: network address translation

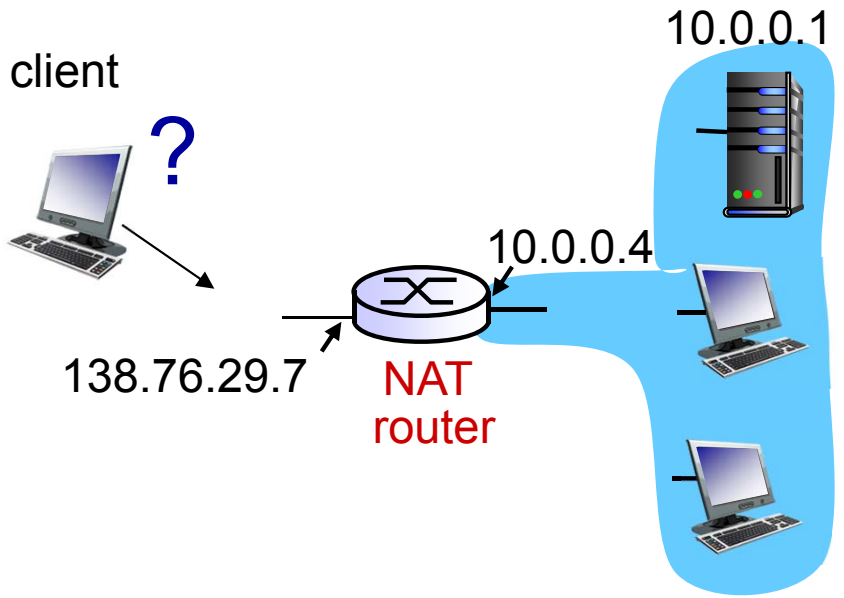


NAT: network address translation

- 16-bit port-number field:
  - 60,000 simultaneous connections with a single LAN-side address!
- NAT is controversial:
  - routers should only process up to layer 3
  - violates end-to-end argument
    - NAT possibility must be taken into account by app designers, e.g., P2P applications
  - address shortage should instead be solved by IPv6

NAT traversal problem

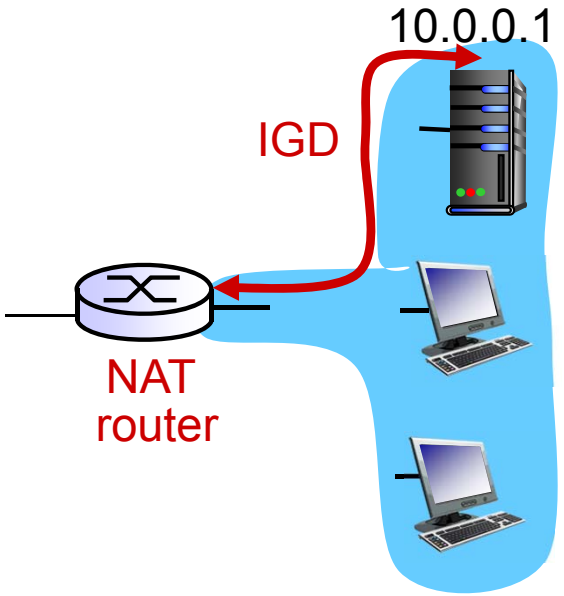
- client wants to connect to server with address 10.0.0.1
  - server address 10.0.0.1 local to LAN (client can't use it as destination addr)
  - only one externally visible NATed address: 138.76.29.7
- *solution1:* statically configure NAT to forward incoming connection requests at given port to server
  - e.g., (123.76.29.7, port 2500) always forwarded to 10.0.0.1 port 25000



NAT traversal problem

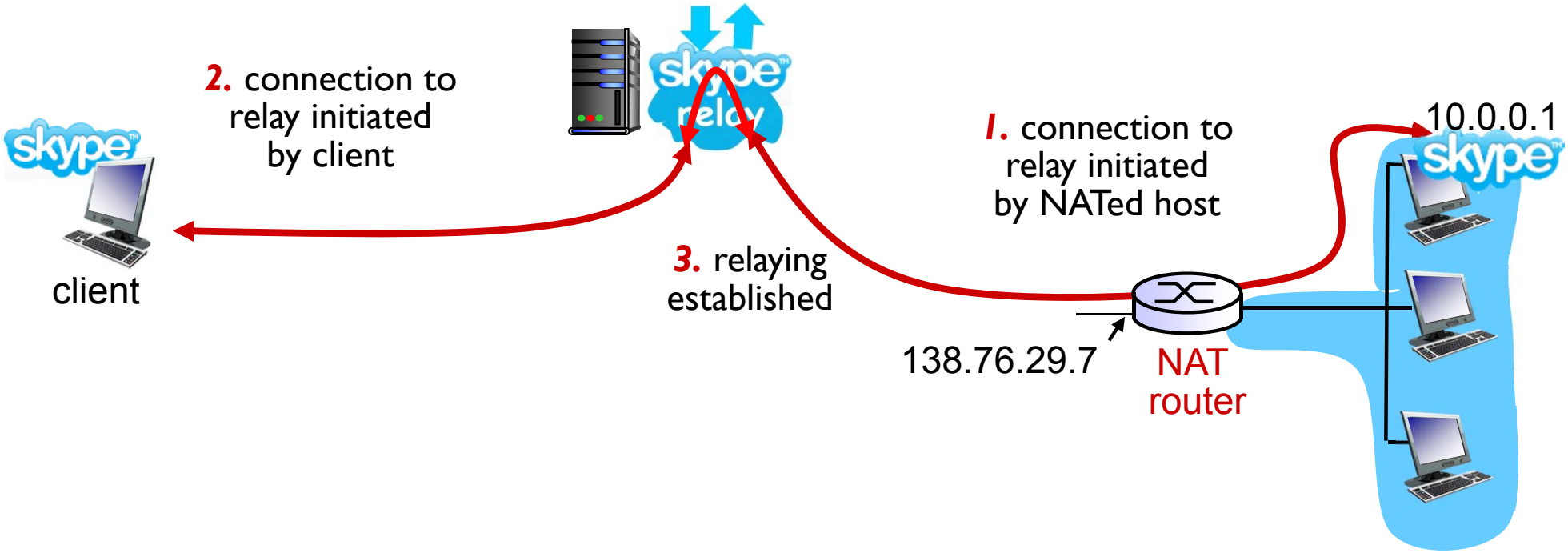
- *solution 2:* Universal Plug and Play (UPnP) Internet Gateway Device (IGD) Protocol. Allows NATed host to:
  - learn public IP address (138.76.29.7)
  - add/remove port mappings (with lease times)

  i.e., automate static NAT port map configuration



NAT traversal problem

- *solution 3:* relaying (used in Skype)
  - NATed client establishes connection to relay
  - external client connects to relay
  - relay bridges packets between to connections



IPv6

- What ever happened to IPv5? – ST, ST2
- Decades old – Coming of Age -- 1991
- 128 bit addressing
  - 340,282,366,920,938,463,463,374,607,431,768,211,456
  - 3.4×10^{38} addresses
  - 6.7×10^{17} addresses/mm²
- Addressing:
  - 2022:0000:11C2:0000:0000:09C0:876A:1B03
  - 2022:0000:11C2::09C0:876A:1B03
  - 64 bit network prefix, 64 bit interface id
- Simplified headers - IPv4's 12 fields reduced to 8
- Fixed 40 byte headers, simpler processing, faster routing, security features

Why IPv6?
(Theoretical Reasons)

Only compelling reason: **more IP addresses!**

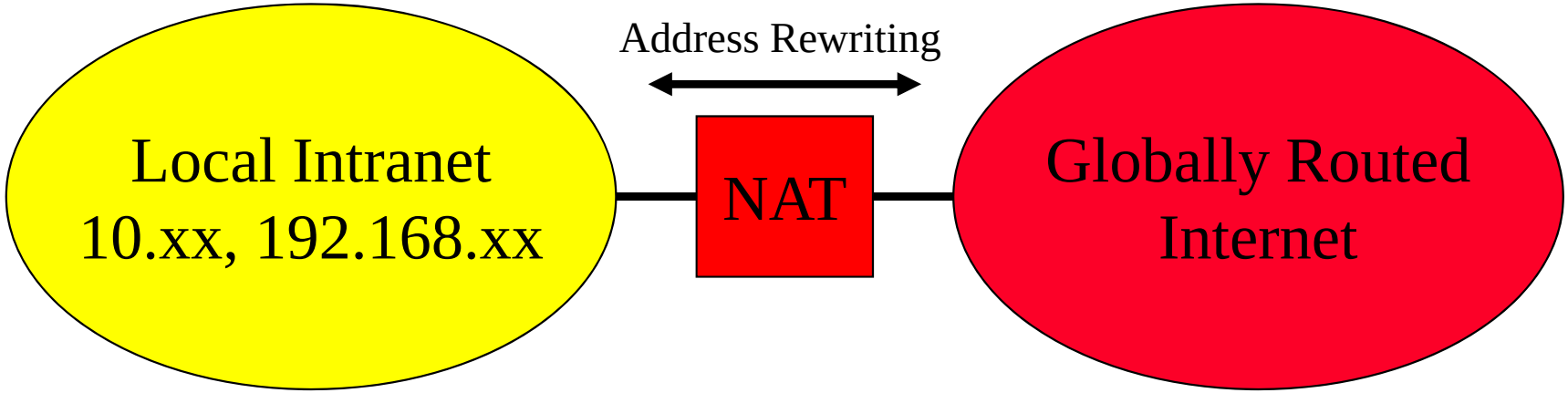
- for billions of new users (Japan, China, India,…)
- for billions of new devices (mobile phones, cars, appliances,…)
- for always-on access (cable, xDSL, ethernet-to-the-home,…)
- for applications that are difficult, expensive, or impossible to operate through NATs (IP telephony, peer-to-peer gaming, home servers,…)
- to phase out NATs to improve the robustness, security, performance, and manageability of the Internet

IP Address Allocation History

1981 - IPv4 protocol published
1985 ~ 1/16 of total space
1990 ~ 1/8 of total space
1995 ~ 1/4 of total space
2000 ~ 1/2 of total space

- this despite increasingly intense conservation efforts
  - PPP / DHCP address sharing
  - CIDR (classless inter-domain routing)
  - NAT (network address translation)
  - plus some address reclamation
- theoretical limit of 32-bit space: ~4 billion devices
  practical limit of 32-bit space: ~250 million devices
  (see RFC-3194)
  RFC 3194 - The H-Density Ratio for Address Assignment Efficiency An Update on the H ratio

What's So Bad About NAT?



- Loss of Transparency
- No Inbound Services
- Some Apps Won't Work
  (e.g. IPsec, WINS)

- Performance Limitations
- Redundancy is Hard
- Nesting is Hard
- Merger is Hard

Other Benefits of IPv6

- server-less plug-and-play possible
- end-to-end, IP-layer authentication & encryption possible
- elimination of “triangle routing” for mobile IP
- other minor improvements

NON-benefits:

- quality of service (same QoS capabilities as IPv4)
  - flow label field in IPv6 header may enable more efficient flow classification by routers, but does not add any new capability
- routing (same routing protocols as IPv4)
  - except larger address allows more levels of hierarchy
    - except customer multihoming is defeating hierarchy

Why IPv6?
(Current Business Reasons)

- demand from particular regions
  - Asia, EU
  - technical, geo-political, and business reasons
  - demand is now
- demand for particular services
  - cellular wireless (especially 3GPP[2] standards)
  - Internet gaming (e.g., Sony Playstation 2)
  - use is >= 1.5 years away
  - potential move to IPv6 by Microsoft?
  - IPv6 included in Windows XP and later

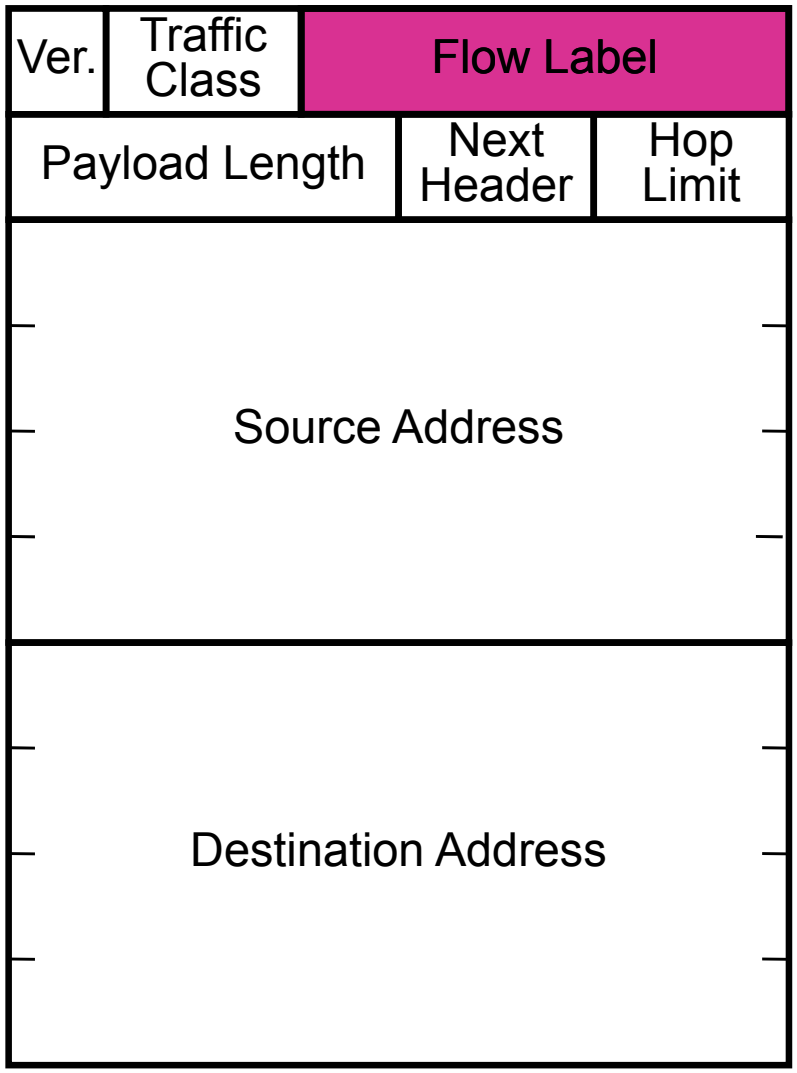
Why IPv6? - Summary

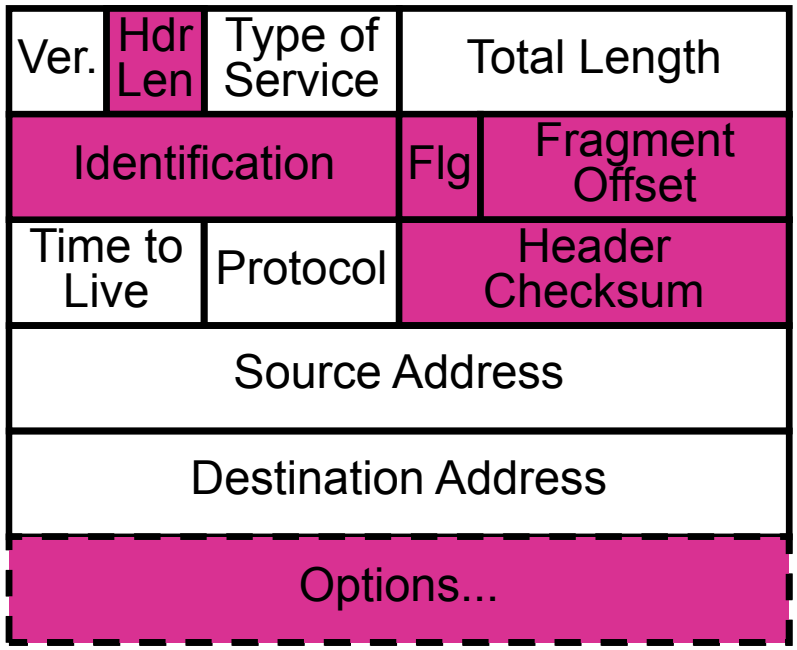
- Huge Address Space
- Address Renumbering/Hierarchy/Mobility
- Multicast/Anycast
- Security (IPsec, Source Route)
- Flow Labels
- High Performance Design
- Jumbograms (packets > 64 KB)

IPv6 Overview

- Background
- Technology Overview
- Deployment Strategies
- Current Status

IPv6 Header compared to IPv4 Header

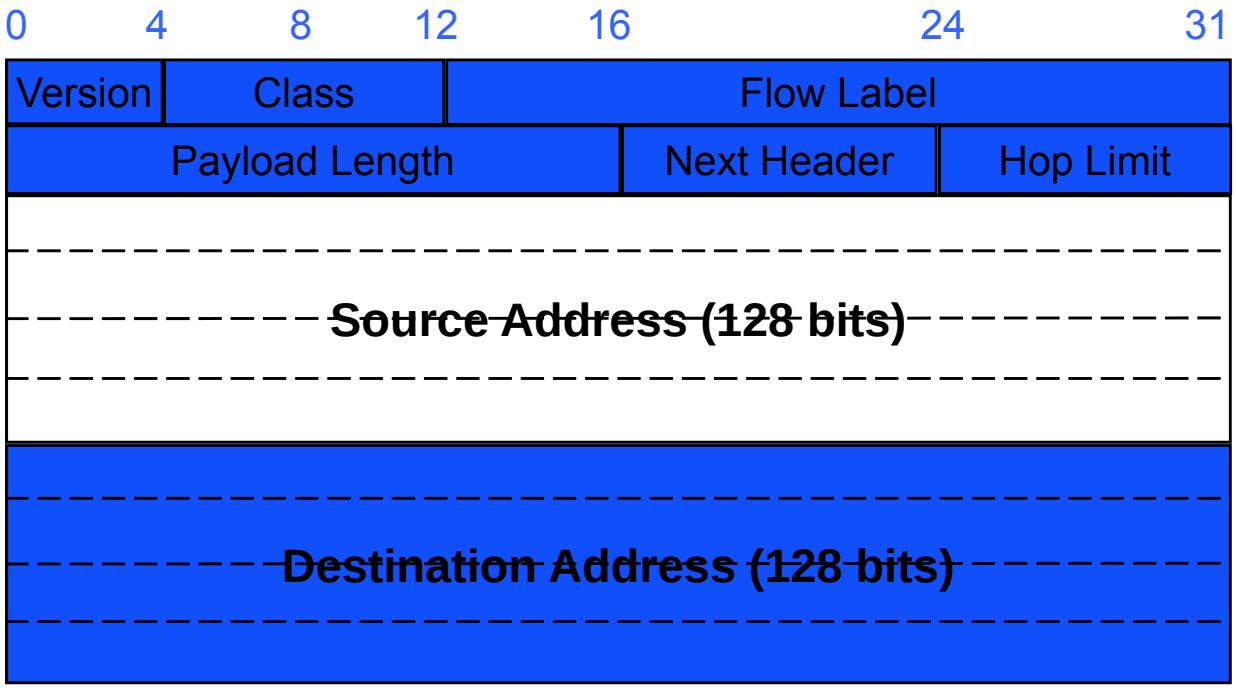




shaded fields have no equivalent in the other version

IPv6 header is twice as long (40 bytes) as IPv4 header without options (20 bytes)

Simplified Header Format



Field	Purpose
Version	IP version number (4 bits) = 6
Class	Data Classifier (8 bits)
Flow Label	Flow identifier (20 bits)
Payload Length	Length of rest of packet (16 bits)

Field	Purpose
Next Header	Header type of next header (8 bits)
Hop Limit	Packet lifetime (8 bits)
Source	Originator of packet (128 bits)
Destination	Intended recipient of packet (128 bits)

Summary of Header Changes

- Revised
  - Addresses increased 32 bits → 128 bits
  - Time to Live → Hop Limit
  - Protocol → Next Header indicates what type of header follows the IPv6 header
  - Type of Service → Traffic Class
- Streamlined
  - Fragmentation fields moved out of base header
  - IP options moved out of base header
  - Header Checksum eliminated
  - Header Length field eliminated
  - Length field excludes IPv6 header
  - Alignment changed from 32 to 64 bits
- Extended
  - Flow Label field added

Key differences in header

- No checksum
  - Bit level errors are checked for all over the place
- No length variability in header
  - Fixed format speeds processing
- No more fragmentation and reassembly in header
  - Incorrectly sized packets are dropped and message is sent to sender to reduce packet size
  - Hosts should do path MTU discovery
  - But of course we have to be able to segment packets!

IPv6 Options

- IPv6 options are placed in separate headers located between IPv6 header and transport-layer (TCP/UDP) header
- Only certain option headers need to be examined by routers
- Option headers must be ordered properly
- Option headers can be large

Current IPv6 Options

- Routing
  - Similar to IPv4 source routing
- Fragment
  - Only end nodes can fragment packets
- Hop-by-hop options (routing options)
  - Carry optional info that must be examined by every node along a packet's delivery path
  - Jumbo payload
- End-to-end options (destination options)
- Authentication
- Encapsulating Security Payload

Sample IPv6 Packets

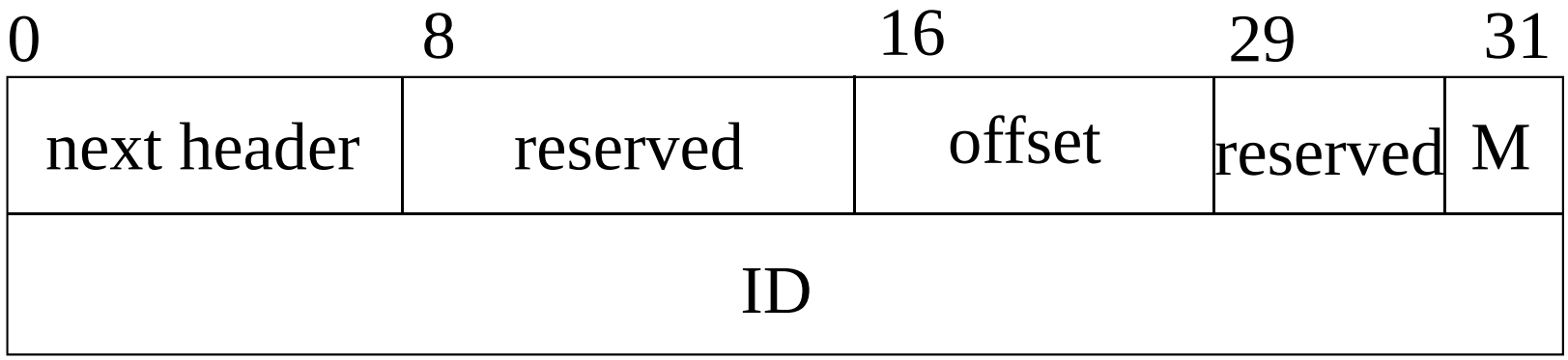
IPv6 Header *Next Header = TCP*	TCP Header *+ Data*

IPv6 Header *Next Header = Routing*	Routing Header *Next Header = TCP*	TCP Header *+ Data*

IPv6 Header *Next Header = Routing*	Routing Header *Next Header = Fragment*	Fragment Header *Next Header = TCP*	Fragment of TCP Header *+ Data*

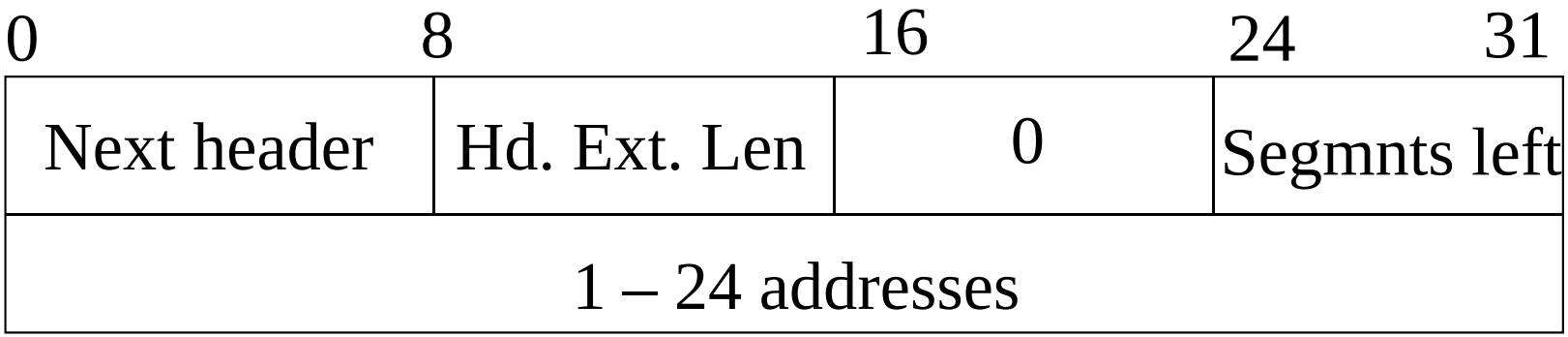
Fragmentation Extension

- Similar to v4 fragmentation
  - Implemented as an extension header
    - Placed between v6 header and data (if it is the only extension used)
  - 13 bit offset
  - Last-fragment mark (M)
  - Larger fragment ID field than v4
- Fragmentation is done on end host



Routing Extension

- Without this header, routing is essentially the same as v4
- With this header essentially same as the source routing option in v4
  - Loose or strict
- Header length is in 64-bit words
- Up to 24 addresses can be included
  - Packet will go to nearest of these in “anycast” configuration
- Segments left: tracks current target; Specifies the number of explicitly-named nodes remaining in the route until the destination



How Was IPv6 Address Size Chosen?

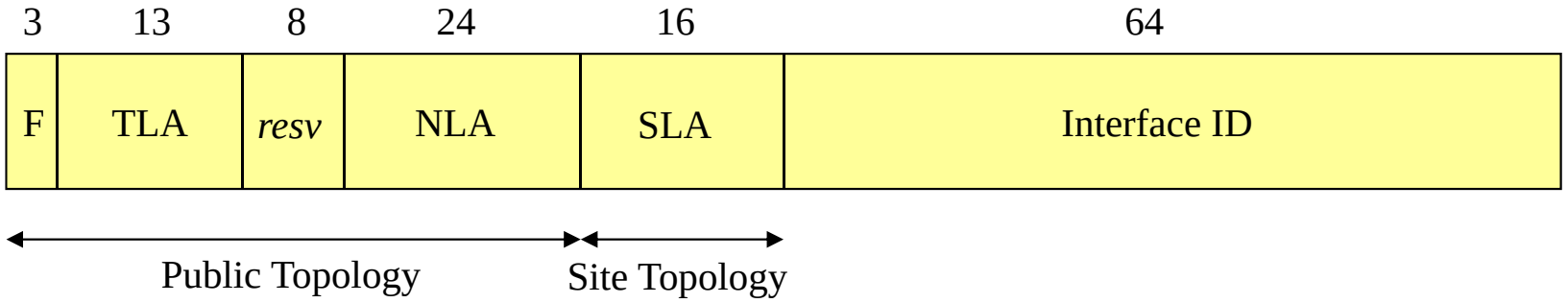
- some wanted fixed-length, 64-bit addresses
  - easily good for 10^{12} sites, 10^{15} nodes, at .0001 allocation efficiency (3 orders of mag. more than IPng requirement)
  - minimizes growth of per-packet header overhead
  - efficient for software processing
- some wanted variable-length, up to 160 bits
  - compatible with OSI NSAP addressing plans
  - big enough for auto-configuration using IEEE 802 addresses
  - could start with addresses shorter than 64 bits & grow later
- settled on fixed-length, 128-bit addresses

  (**340,282,366,920,938,463,463,374,607,431,768,211,456** in all!)

IPv6 Addressing

- Address space allocated for
  - Aggregatable global unicast addresses
  - Local use addresses
    - Link local
    - Site local
  - Multicast addresses (no broadcasting!)
  - 15% initially allocated for use, 85% unallocated

IPv6 Addressing



- Top Level and Next Level Aggregators
- Interface ID typically from MAC address
- Special site-local and link-local addresses
- Special multicast and anycast addresses
- Special IPv4 compatible addresses

Text Representation of Addresses

Written as eight 16-bit hex numbers

“preferred” form: 1080:0:FF:0:8:800:200C:417A

compressed form: FF01:0:0:0:0:0:0:43
becomes FF01::43

IPv4-embedded: 0:0:0:0:0:FFFF:13.1.68.3
or ::FFFF:13.1.68.3

Address prefixes (slash notation) are the same as v4: e.g. FEDC:BA98:7600::/40 describes a 40 bit prefix

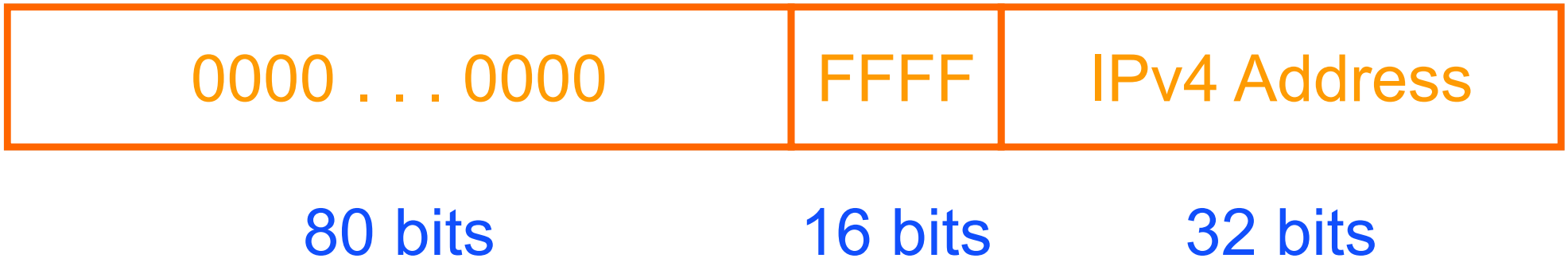
IPv4-Mapped IPv6 Address

- IPv4-Mapped addresses allow a host that supports both IPv4 and IPv6 to communicate with a host that supports only IPv4

- The IPv6 address is based completely on the IPv4 address

IPv4-Mapped IPv6 Address

- 80 bits of 0s followed by 16 bits of ones, followed by a 32 bit IPv4 Address:



Text Representation of Addresses (cont.)

address prefix: 2002:43c:476b::/48
(note: no masks in IPv6!)

zone qualifiers: FE80::800:200C:417A%3

deal with multiple NICs with link-local addresses

in URLs:
http://[3FFE::1:800:200C:417A]:8000

(square-bracket convention also used anywhere else there's a conflict with address syntax)

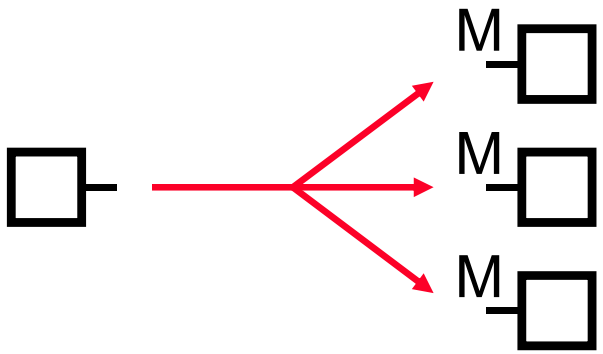
Basic Address Types

unicast:

for one-to-one communication

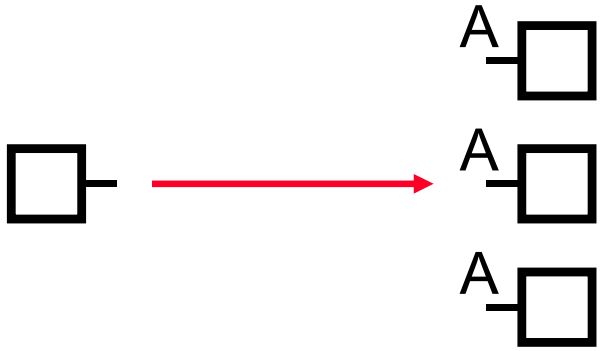
multicast:

for one-to-many communication



anycast:

for one-to-nearest communication



anycast is communication between a single sender and the nearest of several receivers in a group.

Address Prefix Assignments

0000 0000	Reserved
0000 0001	Unassigned
0000 001	Reserved for NSAP (non-IP addresses used by ISO)
0000 010	Reserved for IPX (non-IP addresses used by IPX)
0000 011	Unassigned
0000 1	Unassigned
0001	Unassigned
001	Unicast Address Space
010	Unassigned
011	Unassigned
100	Unassigned
101	Unassigned
110	Unassigned
1110	Unassigned
1111 0	Unassigned
1111 10	Unassigned
1111 110	Unassigned
1111 1110 0	Unassigned
1111 1110 10	Link Local Use addresses
1111 1110 11	Site Local Use addresses
1111 1111	Multicast addresses

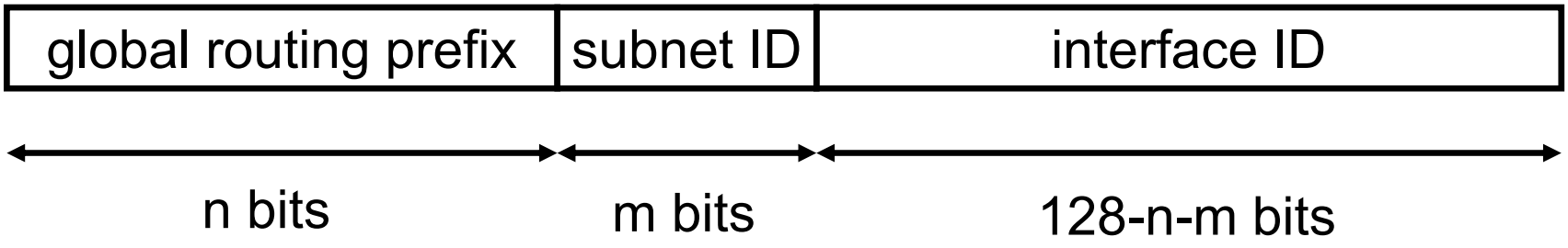
Address Type Prefixes

an address's type is determined by its leading bits:

type	binary prefix	
unspecified	0000…….0000	(128 bits)
loopback	0000…….0001	(128 bits)
multicast	11111111	(8 bits)
unicast / anycast	everything else	

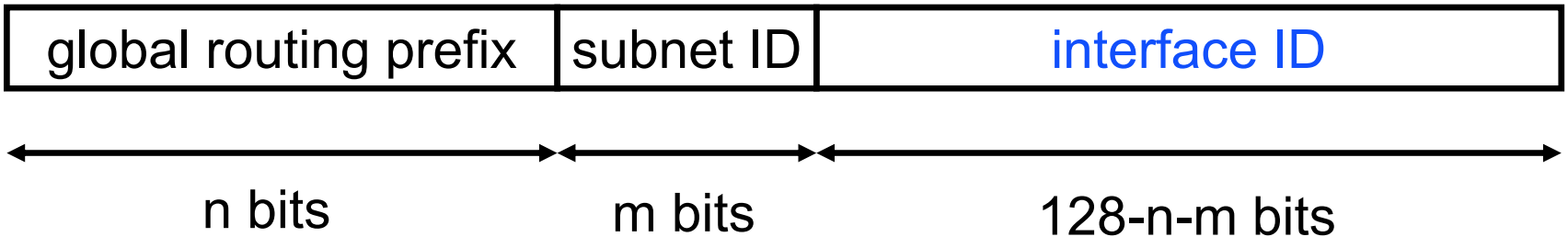
- the unspecified address indicates the absence of an address
- the loopback address is a special-case unicast address
- anycast addresses are treated as unicast

General Format of Unicast Addresses



- unicast addresses are hierarchical, just like IPv4
- the global routing prefix is itself hierarchically structured, usually
- a “subnet” is usually the same as a link, but:
  - may have more than one subnet ID for the same link
  - (proposed) a subnet ID may span multiple links

Interface ID Field of Unicast Addresses



- the interface ID is equivalent to the “host field” in an IPv4 address (but more accurately named)
- if leading bits of address = 000, interface ID may be any width
- if leading bits of address ≠ 000, interface ID is 64 bits wide

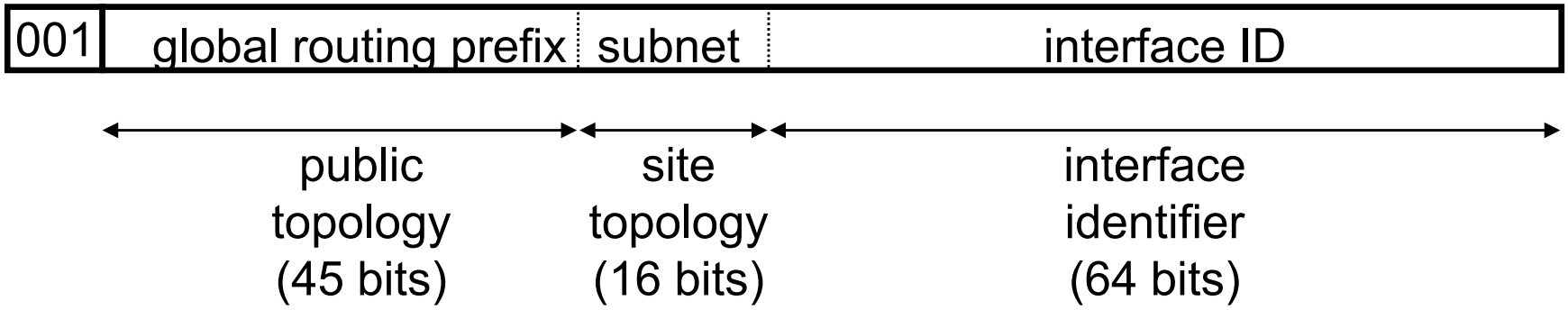
Configuring Interface IDs

there are several choices for configuring the interface ID of an address:

- manual configuration (of interface ID or whole addr)
- DHCPv6 (configures whole address)
- automatic derivation from 48-bit IEEE 802 address or 64-bit IEEE EUI-64 address
- pseudo-random generation (for client privacy)

the latter two choices enable “serverless” or “stateless” autoconfiguration, when combined with high-order part of the address learned via Router Advertisements

Global Unicast Addresses



- only 1/8th of total space (binary 001 prefix) used initially
- global routing prefix is hierarchically structured, using CIDR-type allocation and routing (at least for now!)
- agreed policy is for every subscriber site (e.g., corporate site, campus, residence, etc.) to be assigned a 48-bit prefix → 16 bits of subnet space

Why Fixed-Length, 16-bit Subnet Field?

- fixed length minimizes subscriber hassles when changing service providers or when multi-homing
- 16-bits is enough for all but the largest subscribers
- a standard size eliminates need for most subscribers to provide address space justifications and projections to ISPs

  (for more rationale, see RFC 3177, IAB / IESG Recommendations on IPv6 Address Allocations to Sites)

is remaining 45 bits enough to address all subscribers??

Unicast Assignment in v6

- Unicast address assignment is similar to CIDR
  - Unicast addresses start with 001
  - Host interfaces belong to subnets
  - Addresses are composed of a subnet prefix and a host identifier
  - Subnet prefix structure provides for aggregation into larger networks
- Provider-based plan
  - Idea is that the Internet is global hierarchy of network
  - Three levels of hierarchy – region, provider, subscriber
  - Goal is to provide route aggregation to reduce BGP overhead
    - A provider can advertise a single prefix for all of its subscribers
  - Region = 13 bits, Provider = 24 bits, Subscriber = 16 bits, Host = 80 bits
    - Eg. 001,regionID,providerID,subscriberID,subnetID,intefaceID
  - What about multi-homed subscribers?
    - No simple solution
- Anycast addresses are treated just like unicast addresses
  - It's up to the routing system to determine which server is "closest"

The HD (Host Density) Ratio (RFC-3194)

- measures “pain level” of a given level of utilization of a hierarchical address space, on a scale of 0 to 1
- HD = log (number of addressed objects) / log (total number of addresses)
- historical analysis of IPv4, US phone numbers, French phone numbers, DECnet IV, etc. shows remarkable consistency:

  HD = 0.80 manageable (51M for 32-bit space)

  HD = 0.85 painful (154M for 32-bit space)

  HD = 0.87 practical limit (240M for 32-bit space)

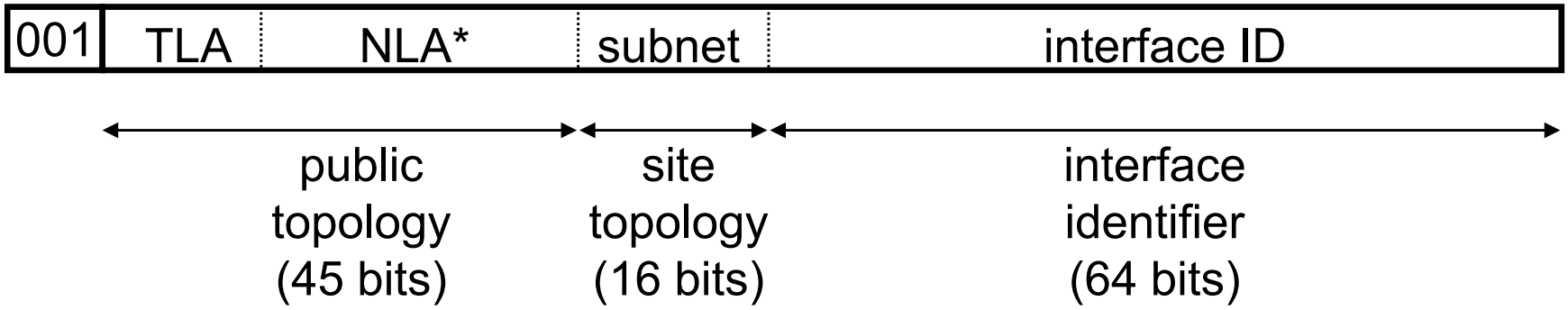
HD Ratio Applied to 45-bit Space

- 45-bit space for sites holds 35 trillion numbers
- achievable utilization, according to HD ratio:

HD = 0.80	manageable	= 70 billion
HD = 0.85	painful	= 330 billion
HD = 0.87	practical limit	= 610 billion

- current world population is 7 billion, projected to peak at 9 to 12 billion in about 2070
- remember: this is still using only 1/8th of total IPv6 address space; majority of space is being kept in reserve in case these projections miss the mark

TLA / NLA Terminology (Soon to be Obsolete!)



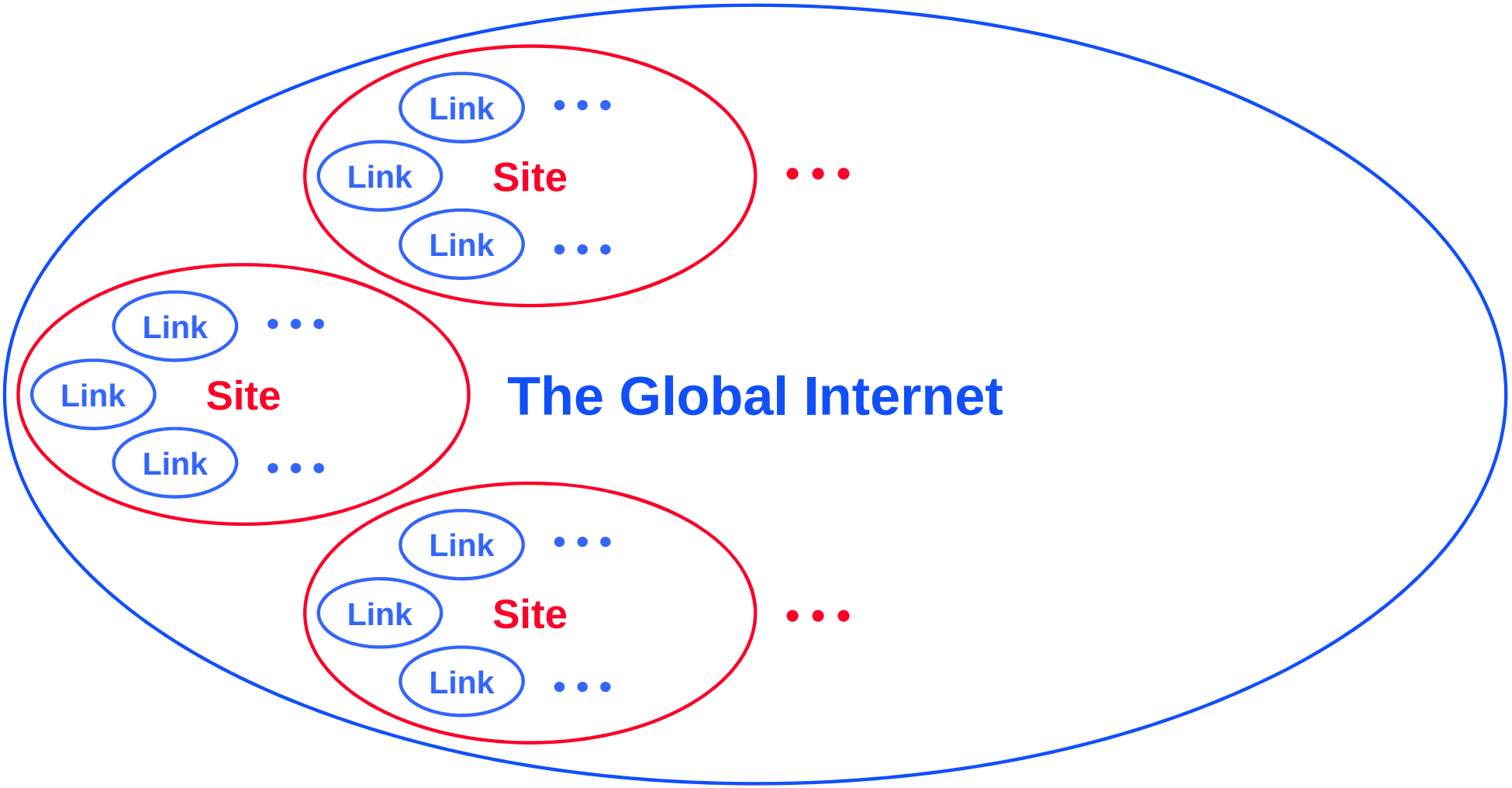
- TLA = Top-Level Aggregator
  NLA* = Next-Level Aggregator(s)
- this structure is defined in existing IPv6 Address Architecture RFCs and registry policy documents, but has been dropped in more recent revisions
- regional internet registries (RIRs) are responsible for structure/allocation of the 45-bit global routing part

Non-Global Addresses

- IPv6 includes non-global addresses, similar to IPv4 private addresses (“net 10”, etc.)
- a topological region within which such non-global addresses are used is called a zone
- zones come in different sizes, called scopes (e.g., link-local, site-local,…)
- a zone, is a connected region of topology of a given scope
- unlike in IPv4, a non-global address zone is also part of the global addressable region (the “global zone”)

  → an interface may have both global and non-global addresses

Address Zones and Scopes



Each oval is a different zone; different colors indicate different scopes

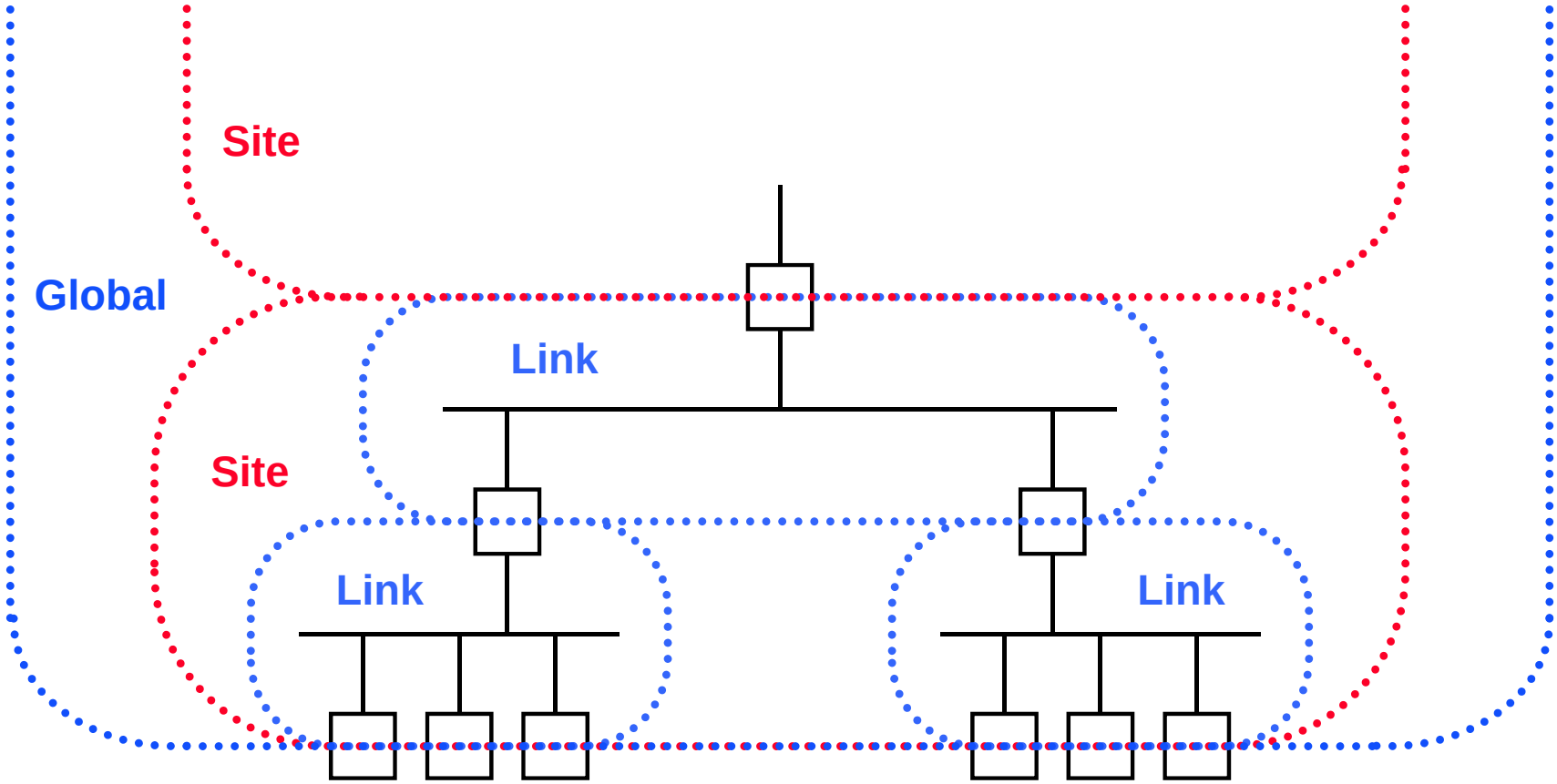
Properties of Zones and Scopes

- zones of the same scope do not overlap, e.g., two sites cannot overlap (i.e., cannot have any links in common)
- zones of smaller scope nest completely within zones of larger scope
- zones of same scope can reuse addresses of that scope (e.g., the same site-local address can occur in more than one site)

Properties of Zones and Scopes (cont.)

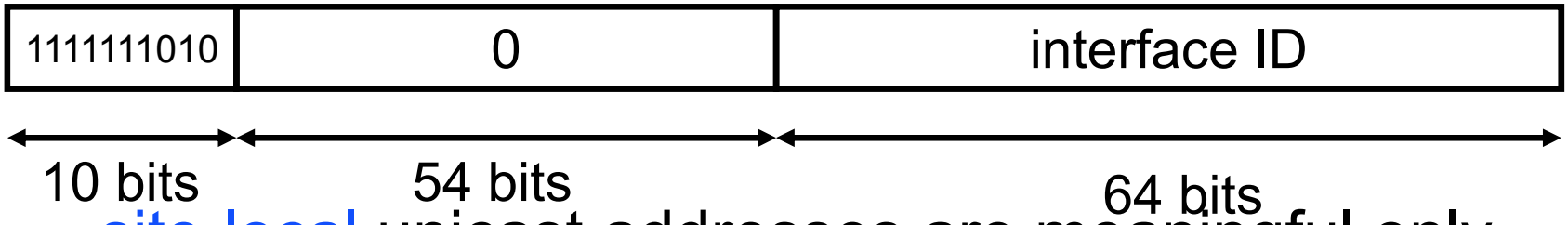
- the scope of an address is encoded in the address itself, but the zone of an address is not
  - that's why the "%zone-id" qualifier is needed, in the text representation of addresses
  - for a non-global address received in a packet, its zone is determined based on what interface it arrived on
- packets with a source or destination address of a given scope are kept within a zone of that scope
  - (enforced by zone-boundary routers)
- zone boundaries always cut through nodes, not links or interfaces

Zone Boundaries

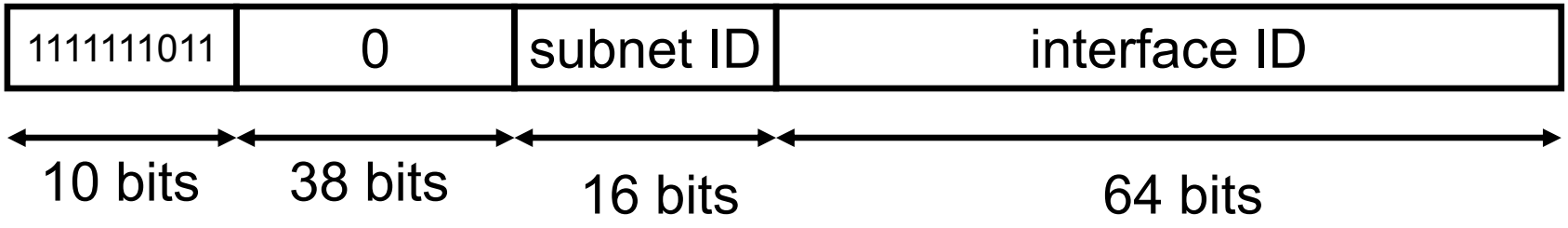


Non-Global Unicast Addresses

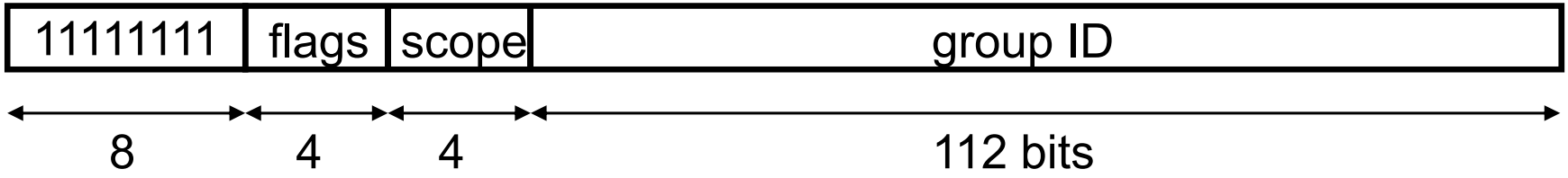
- link-local unicast addresses are meaningful only in a single link zone, and may be re-used on other links



- site-local unicast addresses are meaningful only in a single site zone, and may be re-used in other sites



Multicast Addresses

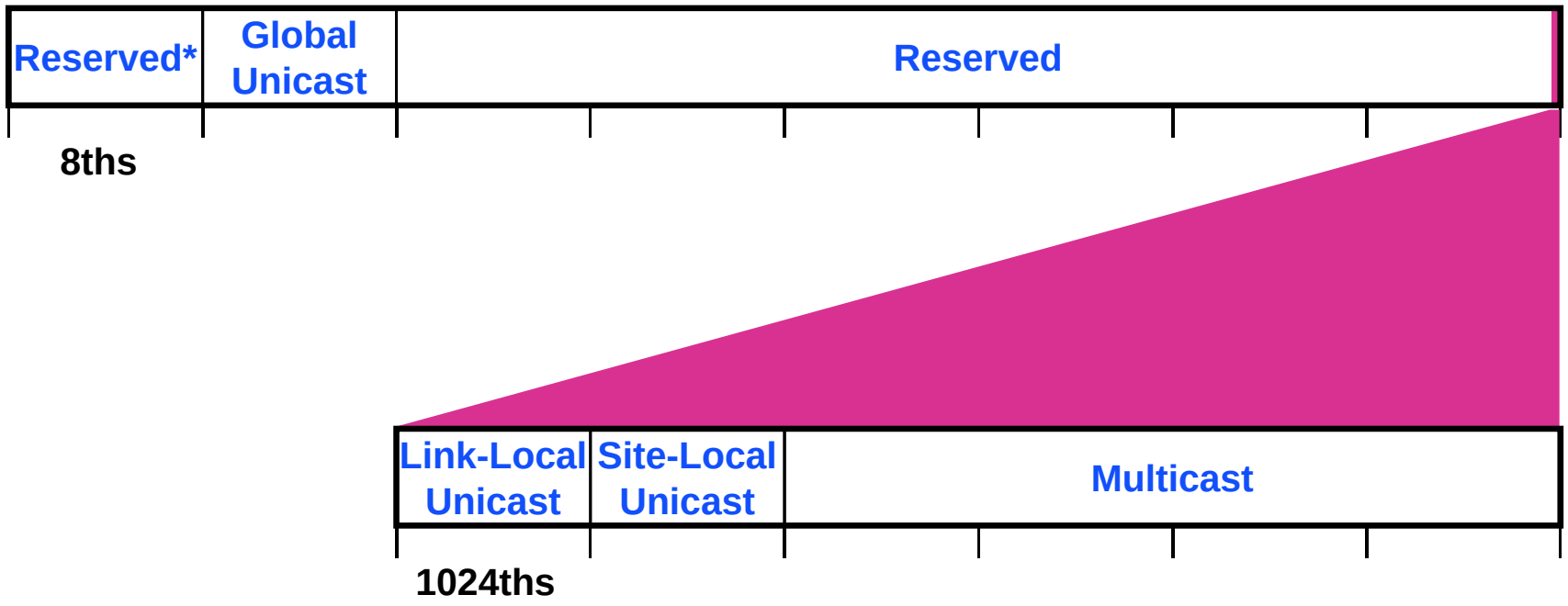


- low-order flag indicates permanent / transient group; three other flags reserved
- scope field:

1 - interface-local	(for multicast loopback)
2 - link-local	(same as unicast link-local)
3 - subnet-local	
4 - admin-local	
5 - site-local	(same as unicast site-local)
8 - organization-local	
B - community-local	
E - global	(same as unicast global)

(all other values reserved)

Address Space Layout



* Part of the first reserved 8th of space is allocated to various special-purpose addresses, currently including the Unspecified, Loopback, IPv4-Embedded, and NSAP-Embedded addresses, altogether consuming ~128th of total space.

An Interface on an IPv6 Node Can, and Usually Will, Have Many Addresses

- Link-Local
- Site-Local
- Auto-configured 6to4 (if IPv4 public address available)
- Solicited-Node Multicast
- All-Nodes Multicast
- Global anonymous
- Global published

IPv6 Routing

- uses same “longest-prefix match” routing as IPv4 CIDR
- straightforward changes to existing IPv4 routing protocols to handle bigger addresses
  - unicast: OSPF, RIP-II, IS-IS, BGP4+, …
  - multicast: MOSPF, PIM, …
- good news: minimal training required for operators
- bad news: routing is in trouble, and IPv6 doesn’t have any magic bullets — multi6 WG is grappling with this

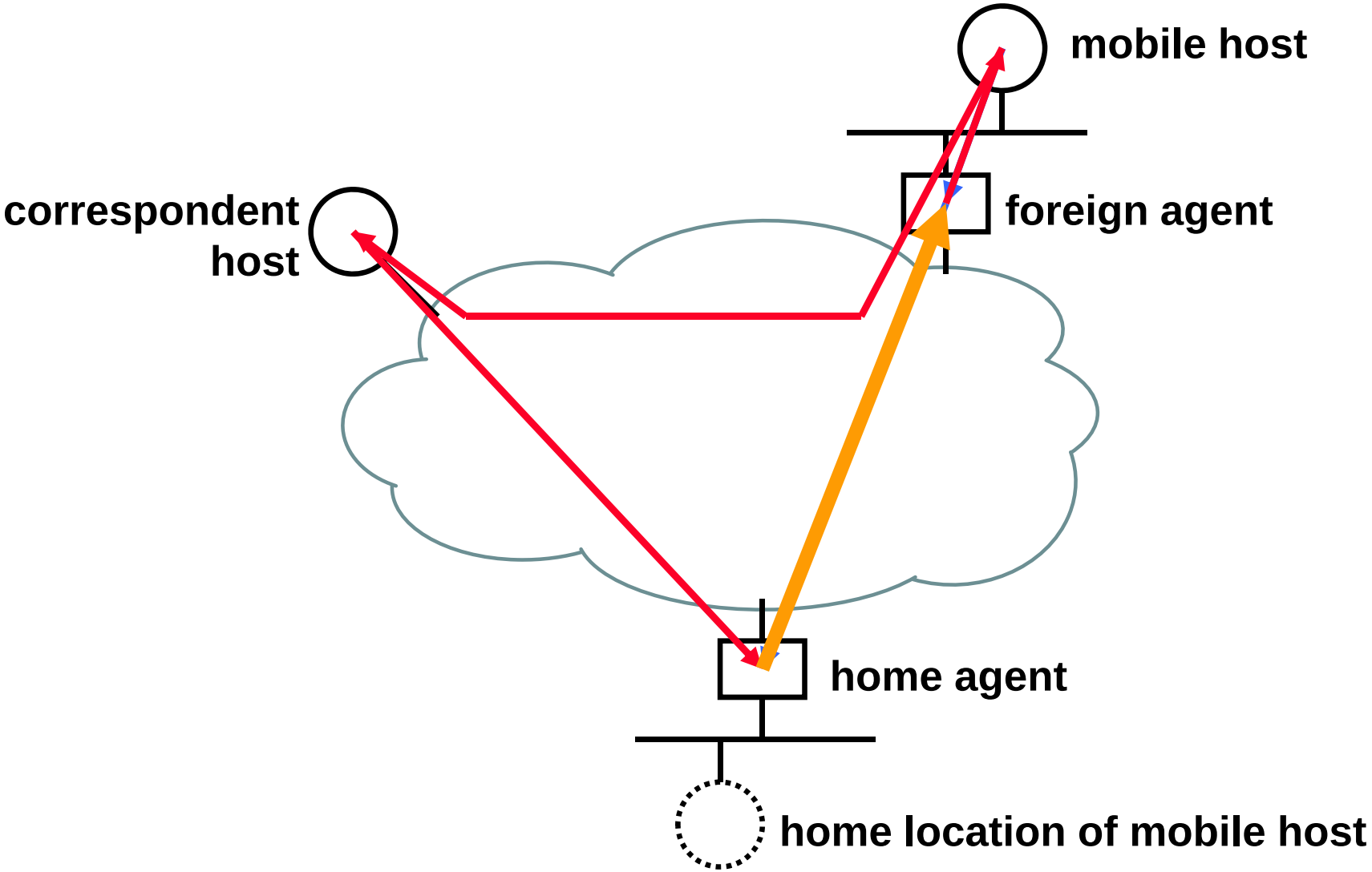
Serverless Autoconfiguration ("Plug-n-Play")

- hosts can construct their own addresses:
  - subnet prefix(es) learned from periodic multicast advertisements from neighboring router(s)
  - interface IDs generated locally, e.g., using MAC addresses
- other IP-layer parameters also learned from router adverts (e.g., router addresses, recommended hop limit, etc.)
- higher-layer info (e.g., DNS server and NTP server addresses) discovered by multicast / anycast-based service-location protocol [details still to be decided]
- DHCP also available for those who want more control (DHCP for IPv6)

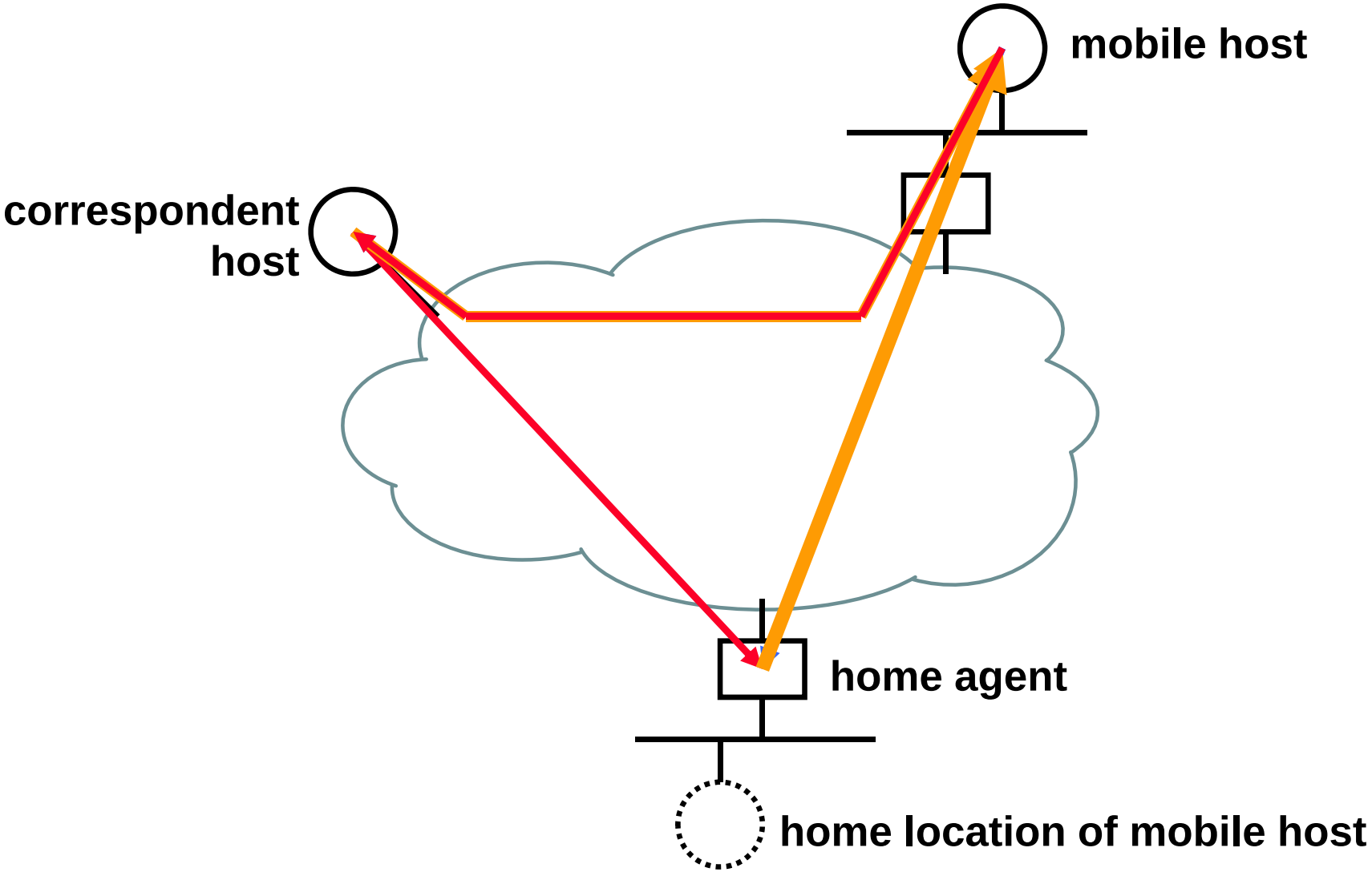
Auto-Reconfiguration ("Renumbering")

- new address prefixes can be introduced, and old ones withdrawn
  - we assume some overlap period between old and new, i.e., no "flash cut-over"
  - hosts learn prefix lifetimes and preferability 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

Mobile IP (v4 version)



Mobile IP (v6 version)



Throughput Control

- IPv6 has throughput control and management enhancements
  - 8-bit class field enables a source to identify the desired delivery characteristics of its packets
  - 20-bit flow label enables a source to label those packets for which it requests special handling by IPv6 routers for non-default quality of service or “real-time” service

Throughput Control

- Flow Label
  - A flow is a sequence of packets sent from a source to a destination requiring special handling by routers, such as guaranteed bandwidth (for voice or video)
  - How the flow identifier is obtained and how flow requirements are communicated to routers is still being resolved

Security

- IPv6 adds three security services
  - Packet authentication
  - Packet integrity
  - Packet confidentiality
- Implemented using the Authentication Header and the Encapsulating Security Payload Header

Authentication Header

- Provides cryptographic authentication and/or integrity validation
- Allows client and server to validate each other's identity
- Allows client and server to validate that data in packet wasn't changed
- By default, keyed MD5 digest algorithm is used

ESP Header

- Provides confidentiality protection of the data in a packet
- Uses two modes of operation
  - Tunnel-mode ESP
    - Entire original IP datagram is encrypted and becomes data for another IP datagram
  - Transport-mode ESP
    - ESP header added to datagram that encrypts the data in the datagram

IPv6 Overview

- Background
- Technology Overview
- Deployment Strategies
- Current Status

Why so Long?

- Everything has to change (end-to-end)
- Apps and API's have to change
- Domain Name System (DNS) changes
- Border Gateway Protocol (BGP) changes
- Routing protocol changes
- IPv4 over xxx now needs IPv6 over xxx

IPv4-IPv6 Transition / Co-Existence Techniques

a wide range of techniques have been identified and implemented, basically falling into 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

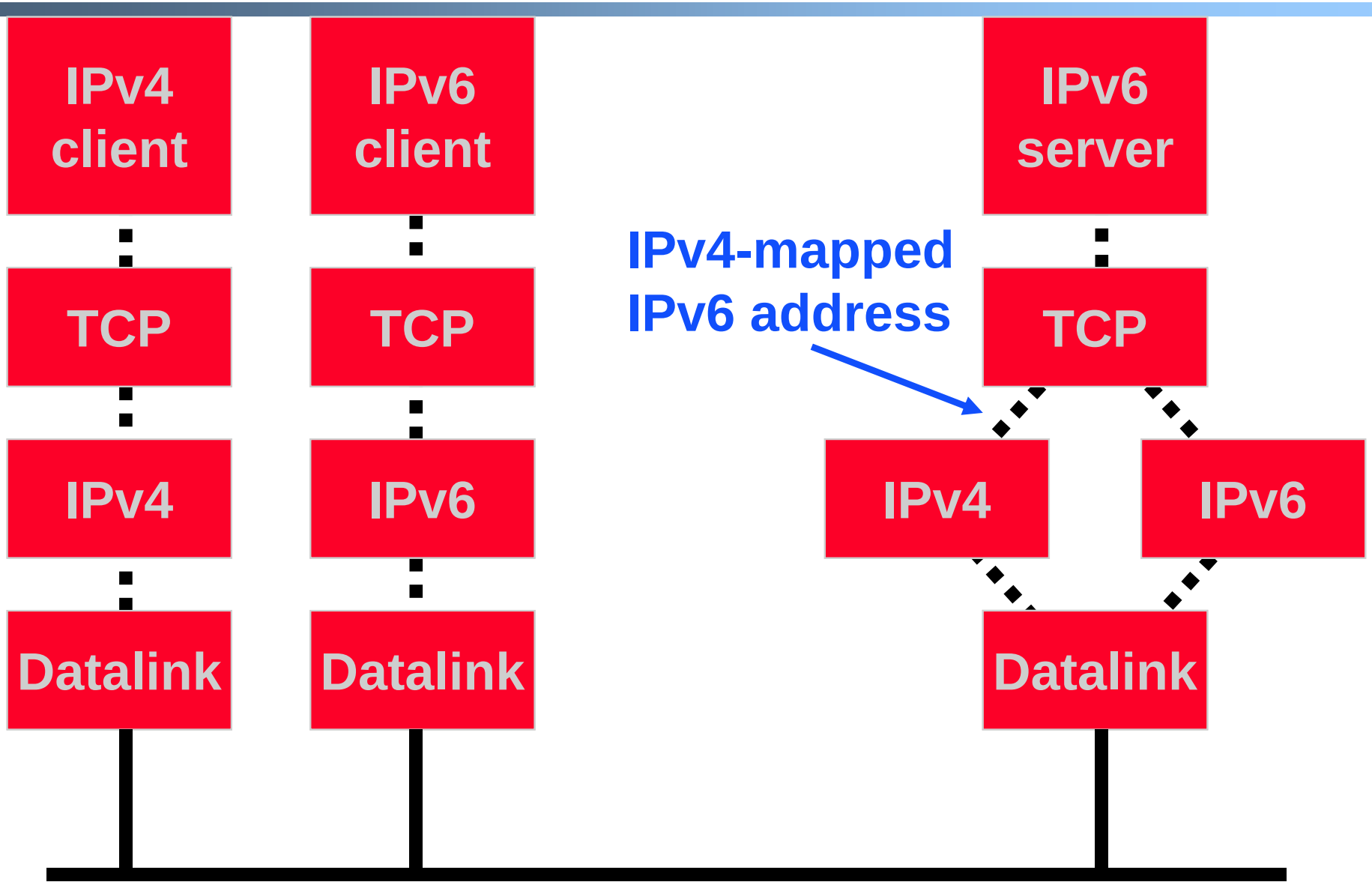
Dual-Stack Approach

- 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.)
  - note: in most cases, IPv6 will be bundled with new OS releases, not an extra-cost add-on
- 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

Dual Server

- In the future it will be important to create servers that handle both IPv4 and IPv6
- The work is handled by the O.S. (which contains protocol stacks for both v4 and v6):
  - automatic creation of IPv6 address from an IPv4 client (IPv4-mapped IPv6 address)


IPv4 client
IPv6 client
IPv6 server
IPv4-mapped IPv6 address
TCP
TCP
TCP
IPv4
IPv6
IPv4
IPv6
Datalink
Datalink
Datalink

IPv6 Clients

- If an IPv6 client specifies an IPv4 address for the server, the kernel detects and talks IPv4 to the server
- DNS support for IPv6 addresses can make everything work
  - **`getaddrinfo()`** returns an IPv4 mapped IPv6 address for hosts that only support IPv4

Works with DNS

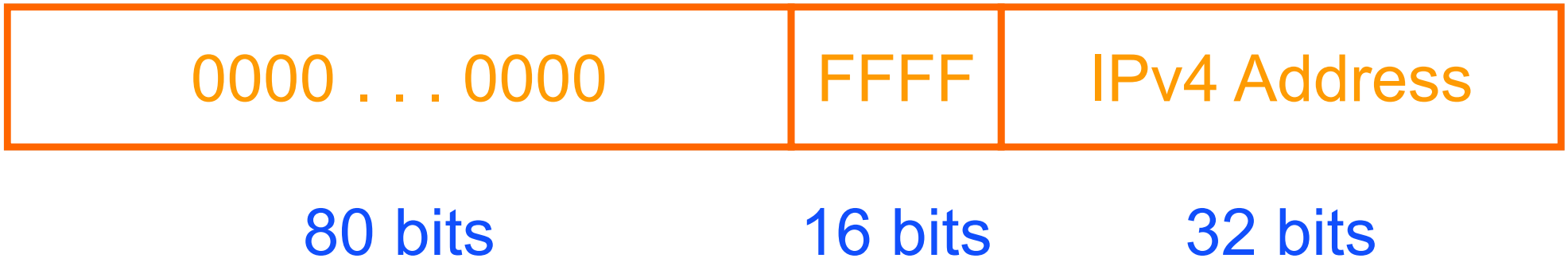
- An IPv6 application asks DNS for the address of a host, but the host only has an IPv4 address
- DNS creates the IPv4-Mapped IPv6 address automatically
- Kernel understands this is a special address and really uses IPv4 communication

IPv4-Compatible IPv6 Address

- An IPv4 compatible address allows a host supporting IPv6 to talk IPv6 even if the local router(s) don’t talk IPv6
- IPv4 compatible addresses tell endpoint software to create a tunnel by encapsulating the IPv6 packet in an IPv4 packet

IPv4-Compatible IPv6 Address

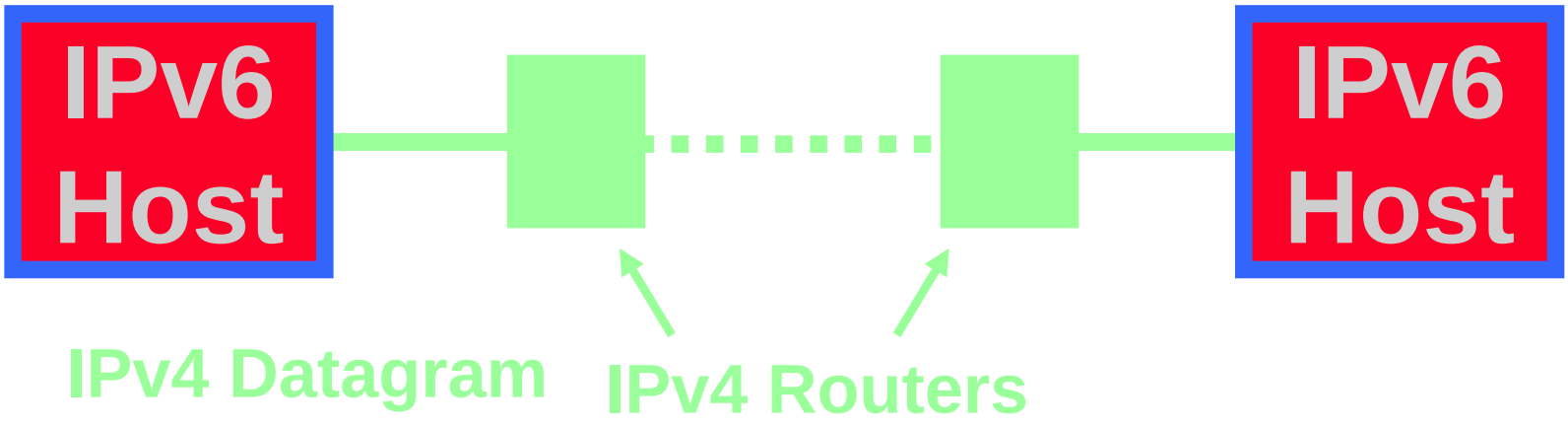
- 80 bits of 0s followed by 16 bits of 0s, followed by a 32 bit IPv4 Address:

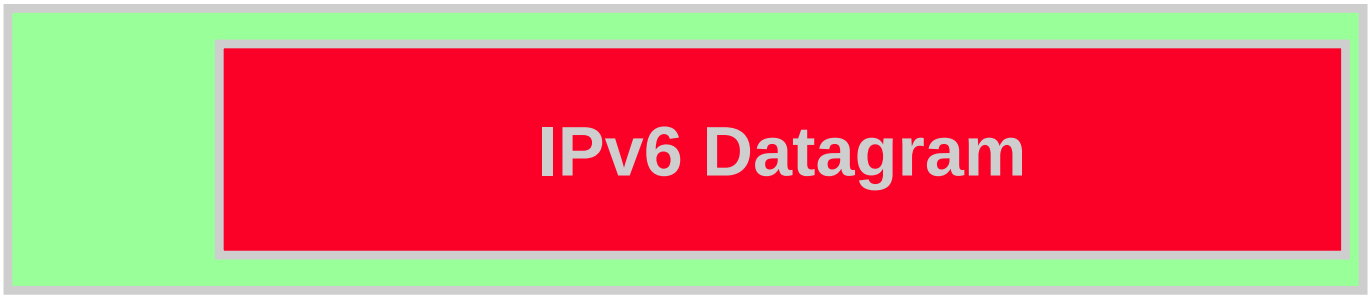


Tunnels to Get Through IPv6-Ignorant Routers

- encapsulate IPv6 packets inside IPv4 packets (or MPLS frames)
- many methods exist for establishing tunnels:
  - manual configuration
  - “tunnel brokers” (using web-based service to create a tunnel)
  - “ISATAP” (intra-domain, using IPv4 addr as IPv6 interface ID)
  - “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, we hope)

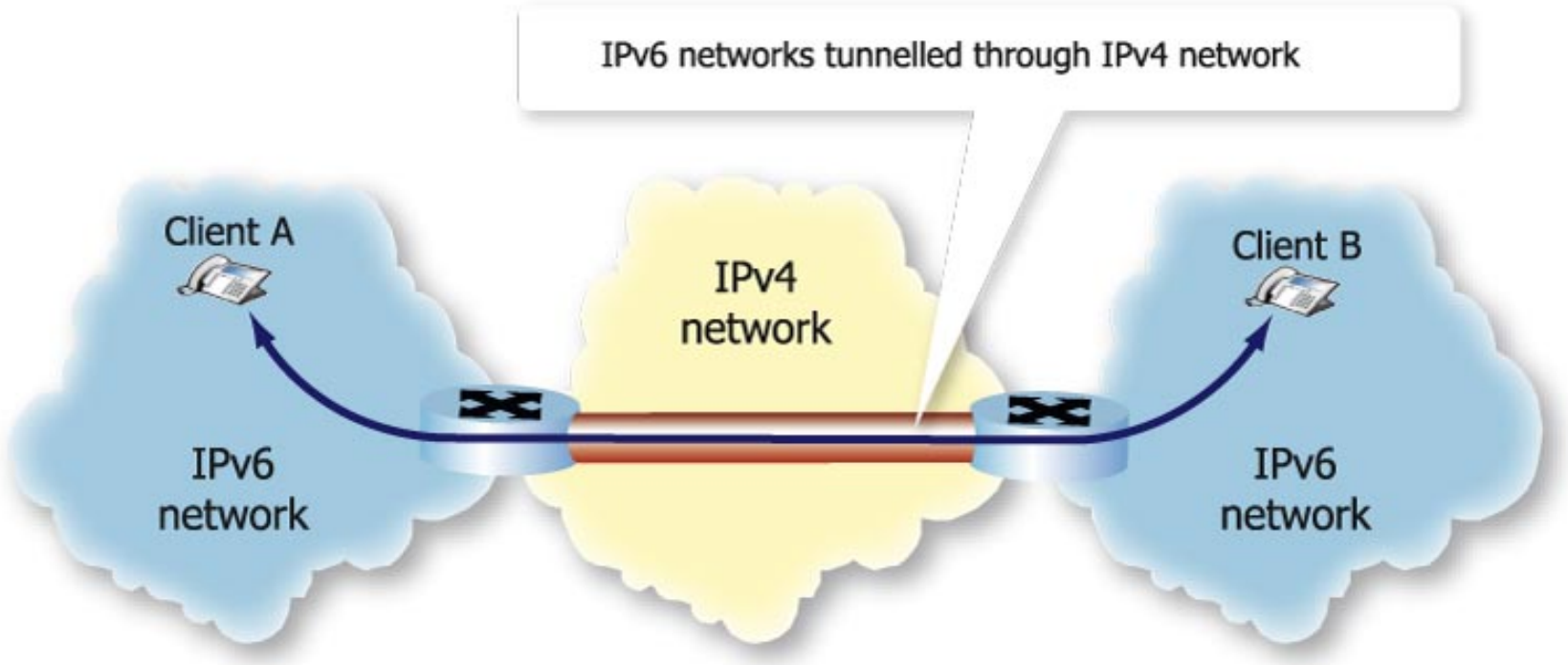
Tunneling
(done automatically by kernel when IPv4-Compatible IPv6 addresses used)





Tunnelling

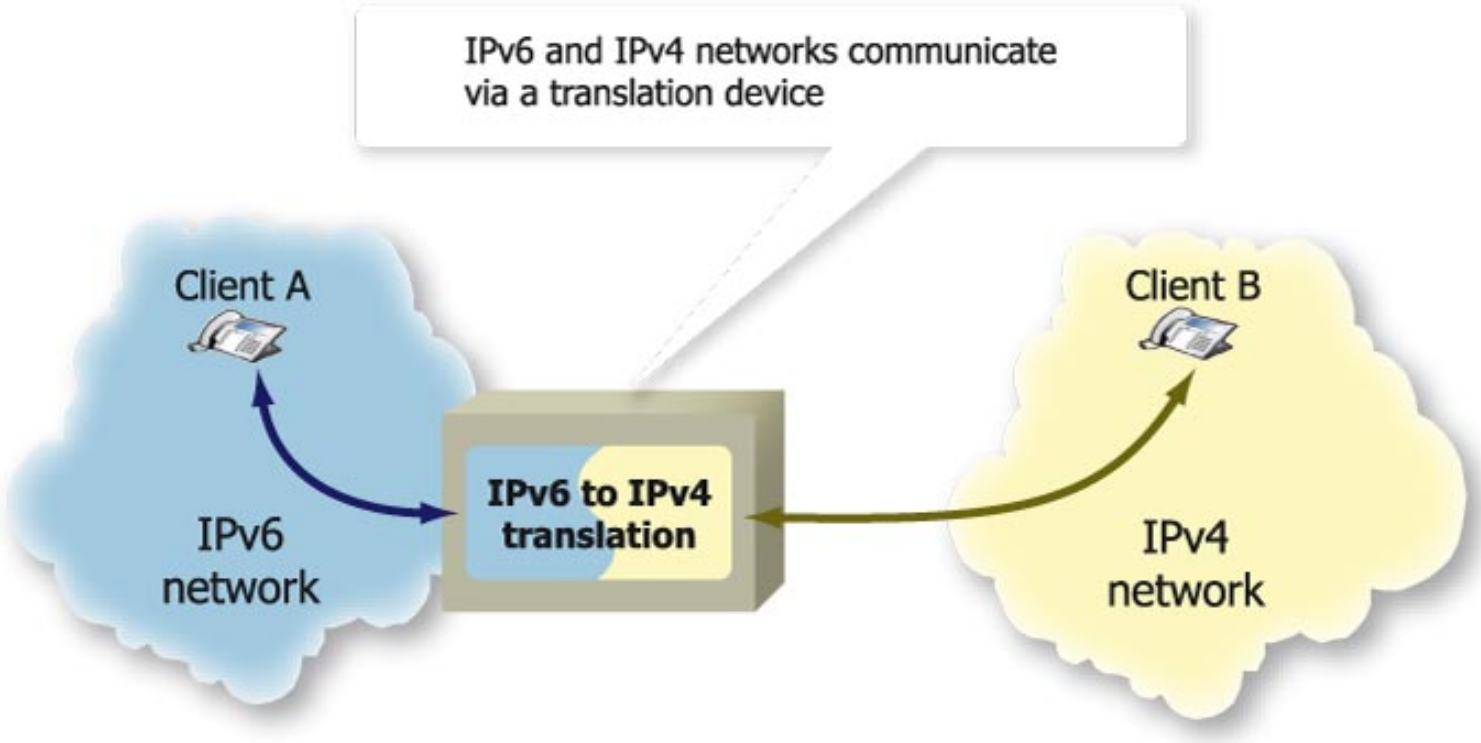
- Connecting two IPv6 Networks through an IPv4 Network



Translation

- may prefer to use IPv6-IPv4 protocol translation for:
  - new kinds of Internet devices (e.g., cell phones, cars, appliances)
  - benefits of shedding IPv4 stack (e.g., serverless autoconfig)
- this is a simple extension to NAT techniques, to translate header format as well as addresses
  - IPv6 nodes behind a translator get full IPv6 functionality when talking to other IPv6 nodes located anywhere
  - they get the normal (i.e., degraded) NAT functionality when talking to IPv4 devices
  - methods used to improve NAT functionality (e.g, RSIP) can be used equally to improve IPv6-IPv4 functionality

Protocol Translation



- Translation of addresses with RFC2766 NAPT-PT may meet traditional Internet requirements, but…
- SIP controlled multimedia requires more

What about TCP and UDP?

- No changes other than to accommodate IPv6
  - 128-bit addresses
  - Checksum's IP pseudo header format changed slightly

What about Applications?

- Most applications have only minimal changes to support 128-bit addresses
  - FTP - internet address exchanged in PORT and PASV operations
  - DNS (AAAA and DNAME resource records)
  - SNMP
  - BOOTP/DHCP
  - Routing protocols (RIP, OSPF, BGP, …)

What Does All This Mean?

- IPv6 design allows for interoperation with IPv4 and easy migration to IPv6
- Ideally no or very minor user impact
- Some burden on network and system administrators (two stacks/protocols)

IPv6 Overview

- Background
- Technology Overview
- Deployment Strategies
- Current Status

Standards

- core IPv6 specifications are IETF Draft Standards
  → well-tested & stable
  - IPv6 base spec, ICMPv6, Neighbor Discovery, PMTU Discovery, IPv6-over-Ethernet, IPv6-over-PPP,...
- other important specs are further behind on the standards track, but in good shape
  - mobile IPv6, header compression,...
  - for up-to-date status: playground.sun.com/ipng
- 3GPP UMTS Release 5 cellular wireless standards mandate IPv6; also being considered by 3GPP2

Implementations

- IPv6 is shipping as a standard feature on most major IP platforms today
  - BSD Unix (all flavors), Cisco, Compaq, Ericsson, HP, IBM, Juniper, Linux, Microsoft, Nokia, Sun, and many more
- in many cases, still missing major pieces
  - e.g., IPsec for IPv6, mobility, multicast, QoS,…
- implementations have been well-tested at frequent multi-vendor events

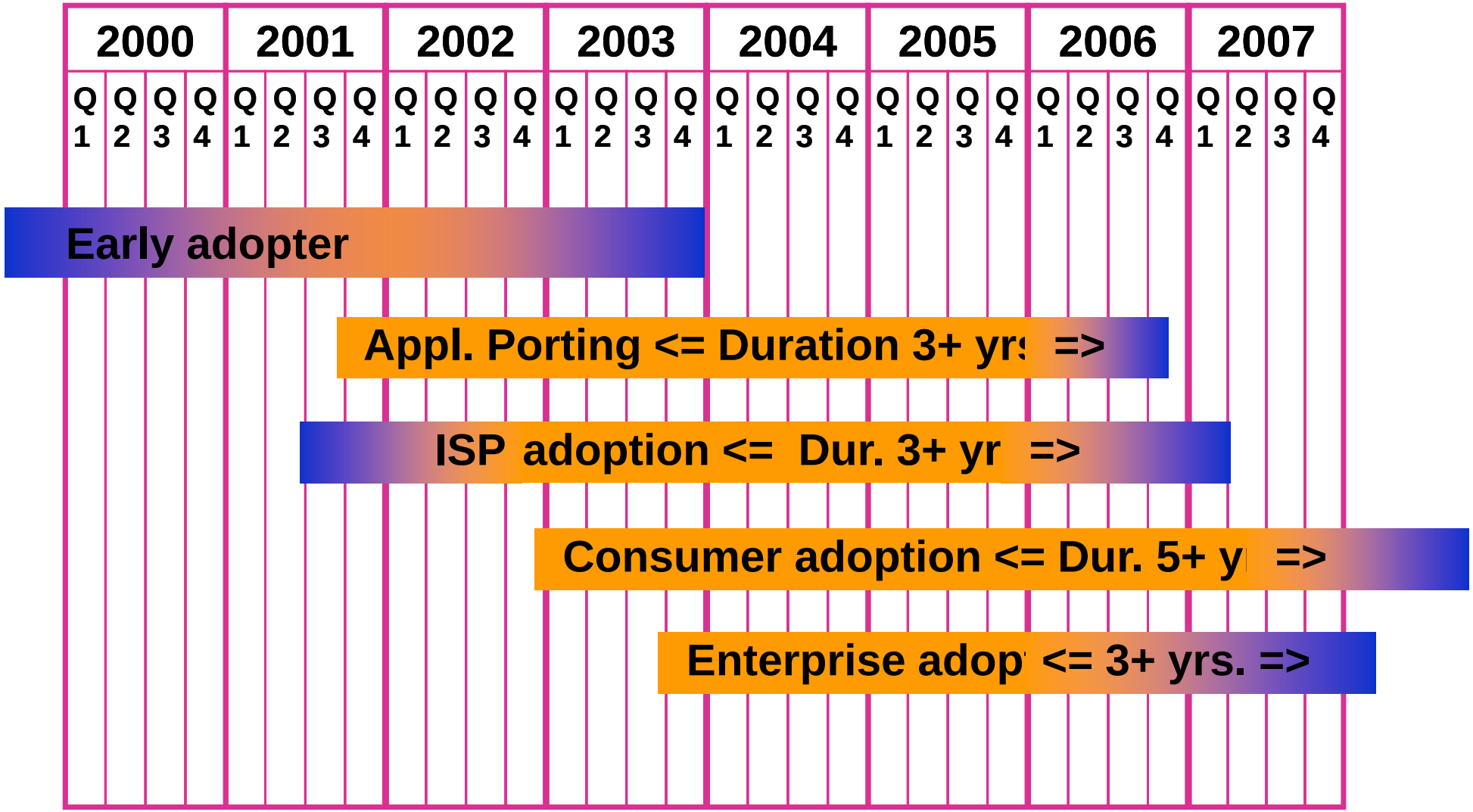
Deployment

- experimental infrastructure: the 6bone
  - for testing and debugging IPv6 protocols and operations (see www.6bone.net)
- 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,…
    (see www.6ren.net, www.6tap.net)
- commercial infrastructure
  - a few ISPs (IIJ, NTT, Telia,…) have deployed commercial IPv6 service, more announced, mainly in Japan and Korea

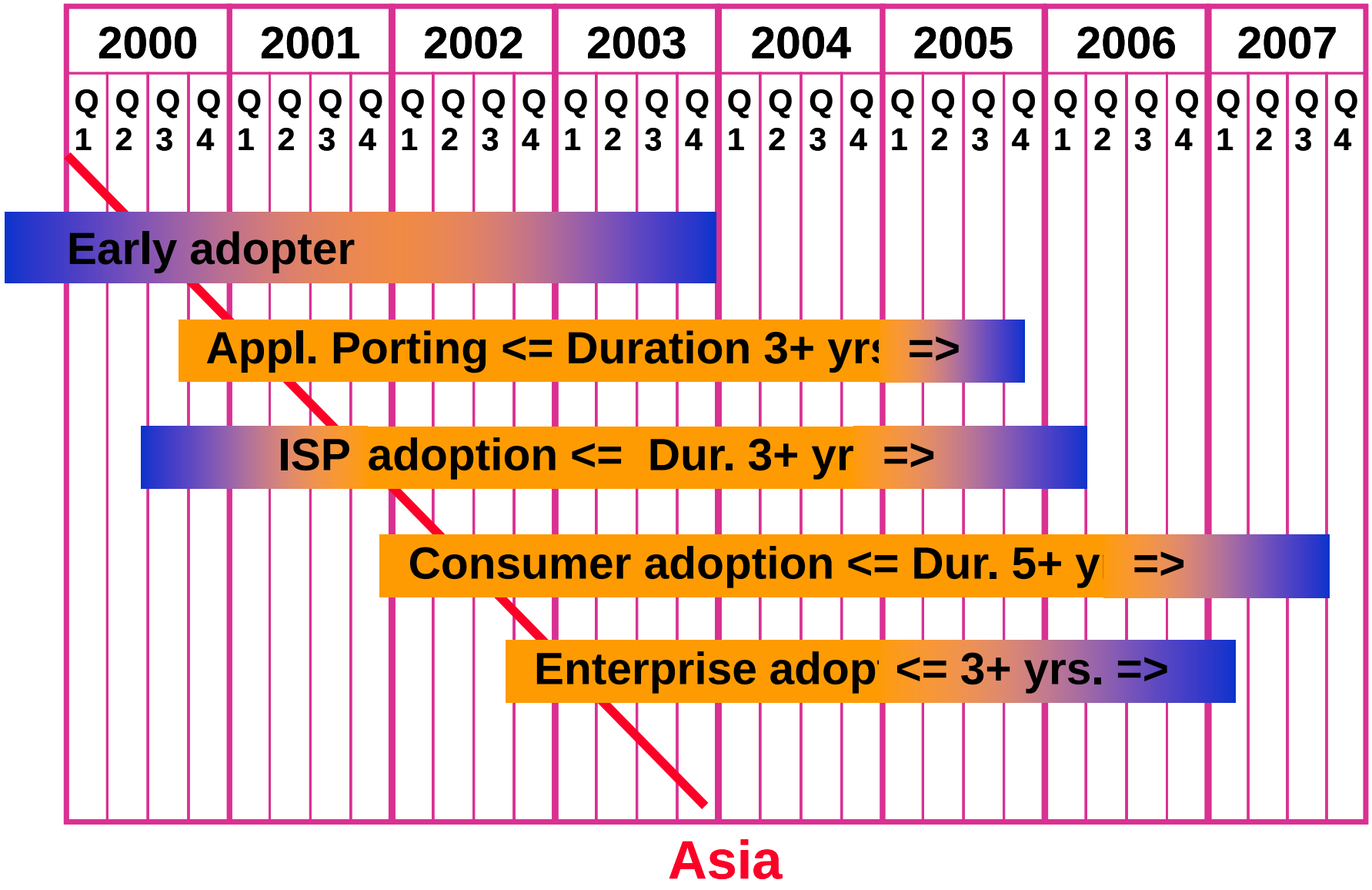
Deployment (cont.)

- IPv6 address allocation
  - 6bone procedure for test address space, phased out in June 2006
  - regional IP address registries (APNIC, ARIN, RIPE-NCC) for production address space
- deployment advocacy (a.k.a. marketing)
  - IPv6 Forum: www.ipv6forum.com

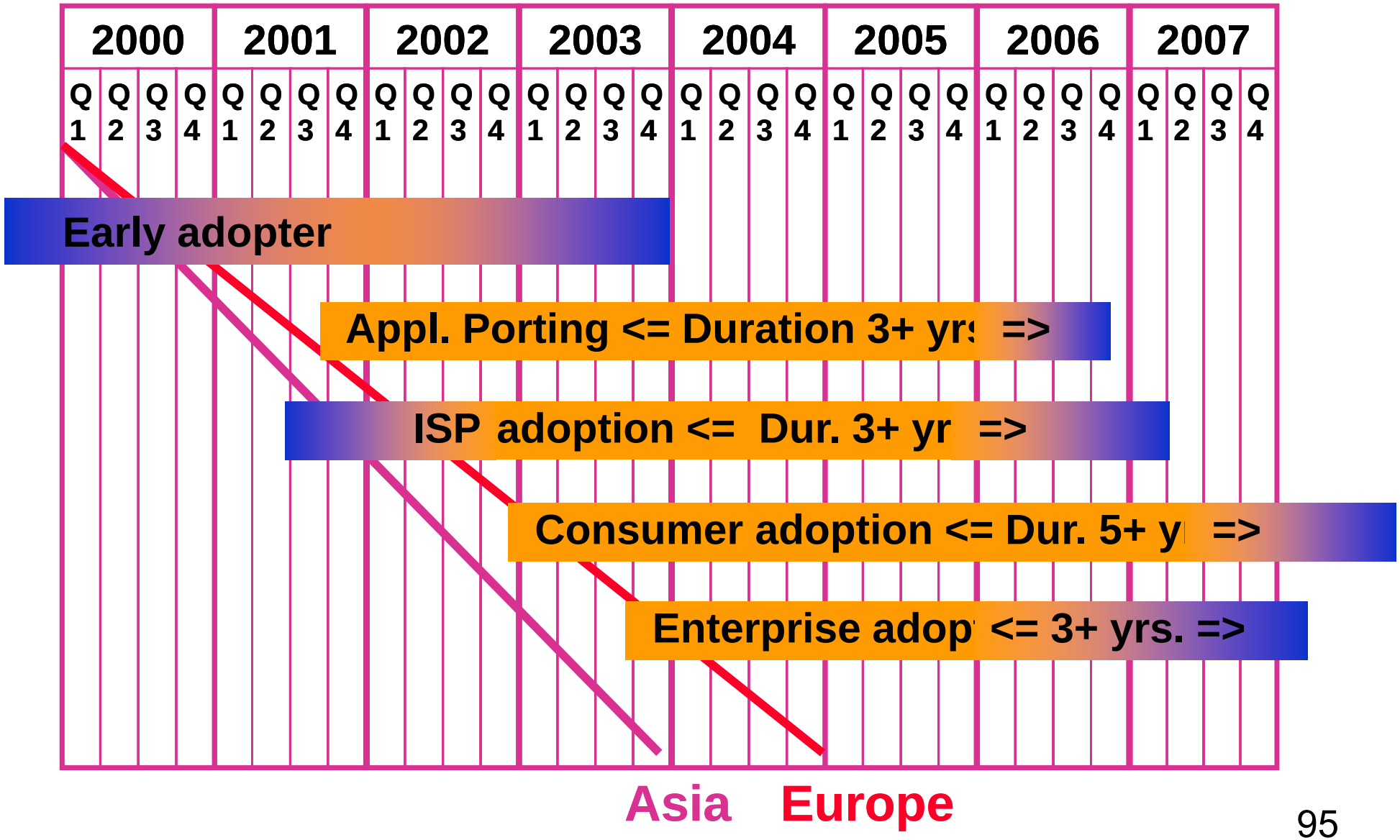
IPv6 Timeline
(A pragmatic projection)



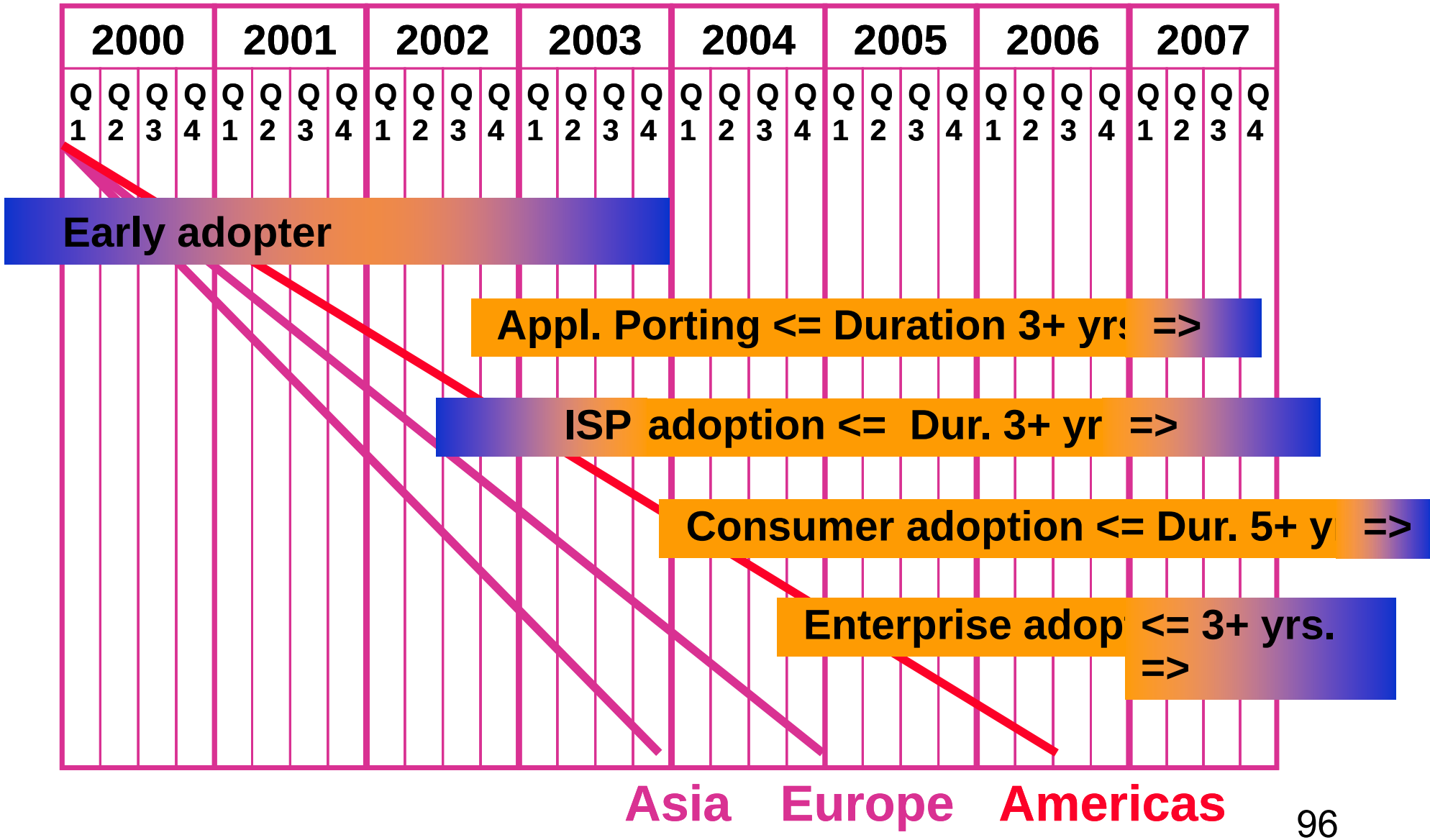
IPv6 Timeline
(A pragmatic projection)



IPv6 Timeline
(A pragmatic projection)



IPv6 Timeline
(A pragmatic projection)



Much Still To Do

though IPv6 today has all the functional capability of IPv4,

- implementations are not as advanced
  (e.g., with respect to performance, multicast support, compactness, instrumentation, etc.)
- deployment has only just begun
- much work to be done moving application, middleware, and management software to IPv6
- much training work to be done
  (application developers, network administrators, sales staff,…)
- many of the advanced features of IPv6 still need specification, implementation, and deployment work

Recent IPv6 “Hot Topics” in the IETF

- multihoming
- address selection
- address allocation
- DNS discovery
- 3GPP usage of IPv6
- anycast addressing
- scoped address architecture
- flow-label semantics
- API issues

  (flow label, traffic class, PMTU discovery, scoping,…)

- enhanced router-to-host info
- site renumbering procedures
- inter-domain multicast routing
- address propagation and AAA issues of different access scenarios
- end-to-end security vs. firewalls
- and, of course, transition / co-existence / interoperability with IPv4
  (a bewildering array of transition tools and techniques)

Note: this indicates vitality, not incompleteness, of IPv6!