2. IPv6 – advanced functionalities

PA159: Net-Centric Computing I.

Eva Hladká

Slides by: Tomáš Rebok

Faculty of Informatics Masaryk University

Autumn 2016

Lecture Overview I

Lecture Overview II

6. IPv6 QoS Support in Detail
   - Integrated Services
   - Differentiated Services
   - QoS and IPv6
7. IPv6 Transition
   - Porting Applications
   - IPv6 and IPv4 Worlds' Interoperability
8. IPv6: Literature
9. IPv6: Where we are?

Lecture overview I

Lecture overview II

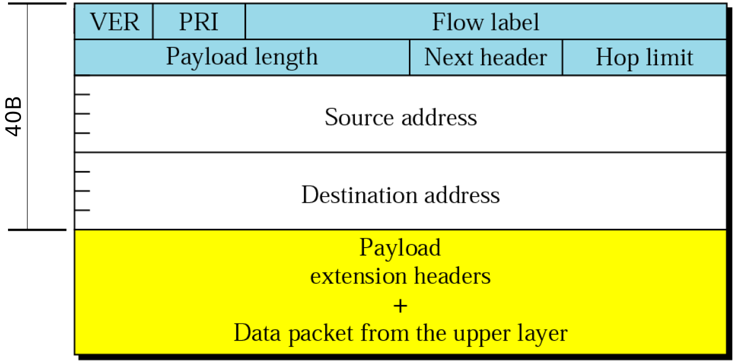
IP Protocol version 6 (IPv6) – Why a new protocol?

- *the master pulse for a new protocol proposal:* relatively fast exhaustion of IPv4 address space
- further reasons: the issues, that arose during IPv4 usage, especially:
  - weak support of real-time applications
  - no support of communication security
  - no devices' autoconfiguration support
  - no mobility support
  - etc.
- (many features retroactively implemented into IPv4)

IP Protocol version 6 (IPv6) – Basic Features

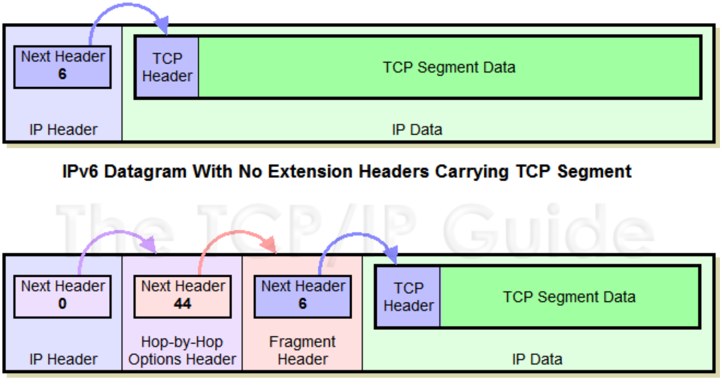
- *bigger address space* – 128-bit IPv6 address, theoretically 2^{128} of unique addresses
- *simpler header format* – basic 40B header containing just the most necessary information
- *possibilities of further extensions* – through so-called *extension headers*
- *support for real-time transmissions* – streams' tagging and priorities
- *support for secure communication* – authentication, encryption and integrity verification support
- *mobility support* – using so-called *home agents*
- *devices' autoconfiguration support* – stateful and stateless autoconfiguration

IPv6 Datagram – Basic Header



- fixed basic header size (40 B)
- checksum, options, and fragmentation information not included in the basic header any more
  - options and fragmentation information has to be ensured via extension headers
  - checksum was removed at all (it's ensured on L2 and L4)

IPv6 Datagram – Extension Headers



IPv6 Datagram With No Extension Headers Carrying TCP Segment

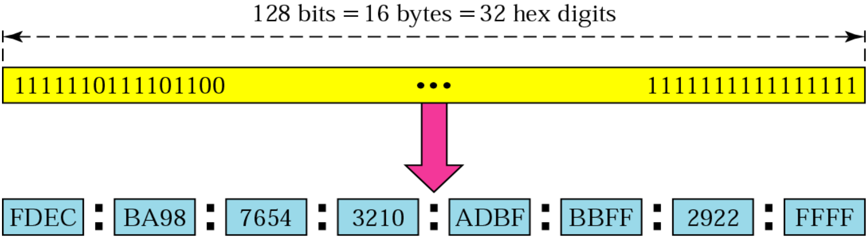
IPv6 Datagram With Two Extension Headers Carrying TCP Segment

Several extension headers have been defined

- e.g., Hop-By-Hop Options, Routing, Fragment, Encapsulating Security Payload, Authentication Header, etc.

IPv6 Addresses

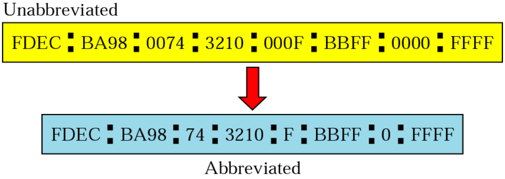
- (currently) final solution to address space shortage
- IPv6 address has 128 bits (= 16 bytes):
  - 2^{128} of unique addresses ($\approx 3 \times 10^{38}$ addresses $\Rightarrow \approx 5 \times 10^{28}$ addresses for every human on the Earth)
  - written in a hexadecimal form instead of decadic (pairs of bytes divided by ":" character)



IPv6 addresses – Address Abbreviation

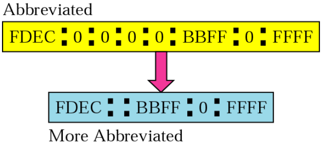
The leading 0s might be omitted in each address group:

- 0074 might be written as 74, 000F as F, . . .
- 3210 **cannot** be abbreviated!



Consecutive groups of zeros might be replaced by “: :” character)

- just **a single group** might be replaced!



IPv6 addresses – Hierarchy

- the goal is to simplify the routing
- the structure of unicast IPv6 addresses is defined by RFC 3587
- basic structure:

n bits	64-n bits	64 bits
global routing prefix	subnet address	interface address

- global routing prefix ≈ network address
- subnetwork address is usually 16-bits long ⇒ the global routing prefix thus has 48 bits
  - first 16 bits contain the value 2001 (hexadecimal form)
  - next 16 bits are assigned by *Regional Internet Registry (RIR)*
  - next 16 bits are assigned by *Local Internet Registry (LIR)*

16 bits	16 bits	16 bits	16 bits	64 bits
2001	assigned by RIR	assigned by LIR	subnet address	interface address

IPv6 Addresses & CIDR

- IPv6 addresses are just *classless* (classes do not exist)
- IPv6 networks are defined using CIDR notation (similarly as in the IPv4 case)
- e.g., *FDEC:0:0:0:0:BBFF:0:FFFF/60*

IPv6 addresses – address types

- *unicast addresses* – same as in IPv4 (a single network interface identification)
- *multicast addresses* – same as in IPv4, used for addressing a group of devices/hosts
  - data is delivered to all the group members
  - prefix `ff00::/8`
- *anycast addresses* – a "newbie" in IPv6
  - identify a group of devices/hosts as well
  - but data is delivered just to a single member of the group (the "closest" one)

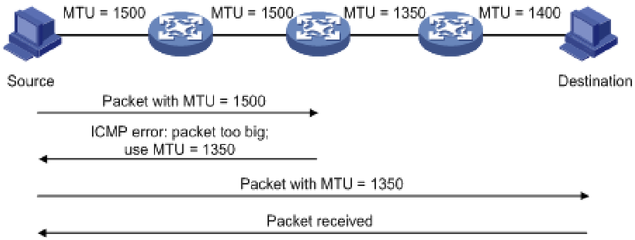
- IPv4 broadcast addresses are not used in IPv6
  - replaced by special multicast addresses (i.e., `FF02::1` – all the nodes on the particular LAN)

IPv6 Path MTU discovery

- just source devices must decide the correct size of fragments
  - *routers are not allowed to fragment datagrams, just the end nodes may!*
  - if a datagram is too large for a router, it must drop it
    - and send back a feedback about this incident to the source (in the form of an ICMPv6 *Packet Too Big* message)
- **Path MTU Discovery**
  - a special technique used for determining what size of fragments should be used
  - uses a feedback mechanism performed by ICMPv6's *Packet Too Big* messages
    - the source node sends a datagram (to the intended receiver) that has the MTU of its local physical link (it represents an upper bound on the MTU)
    - if this goes through without any errors, that value can be used for future datagrams destined to the particular destination
    - if it gets back any *Packet Too Big* messages, it tries again using a smaller datagram size (indicated in the Packet Too Big message)

IPv6 Path MTU discovery

The Schema



Lecture overview I

Lecture overview II

6. IPv6 QoS Support in Detail
   - Integrated Services
   - Differentiated Services
   - QoS and IPv6

7. IPv6 Transition
   - Porting Applications
   - IPv6 and IPv4 Worlds' Interoperability

8. IPv6: Literature

9. IPv6: Where we are?

Neighbor Discovery Protocol I.

- How can we obtain a link (e.g., Ethernet) address of a node (having its IP address)?
    - IPv4: ARP protocol
    - IPv6: a new mechanism called *Neighbor Discovery Protocol*
- *Neighbor Discovery for IP version 6* (RFC 2461)
    - a part of ICMPv6
    - in comparison with the IPv4's ARP, new functionalities have been added
    - IPv6 nodes use Neighbor Discovery for/to:
        - autoconfiguration of IPv6 address (stateful/stateless autoconfiguration)
        - determine network prefixes, routers and other configuration information
        - duplicate IP address detection (DAD)
        - determine layer two addresses of nodes on the same link
        - find neighboring routers that can forward their packets
        - keep track of which neighbors are reachable and which are not (NUD)
        - detect changed link-layer addresses

Neighbor Discovery Protocol II.

- consists of five ICMP messages:
  - *Router Solicitation (RS)*
  - *Router Advertisement (RA)*
  - *Neighbor Solicitation (NS)*
  - *Neighbor Advertisement (NA)*
  - *ICMP Redirect*
- *Inverse Neighbor Discovery* also possible
  - see the literature for details

Neighbor Discovery – L2 address resolution I.

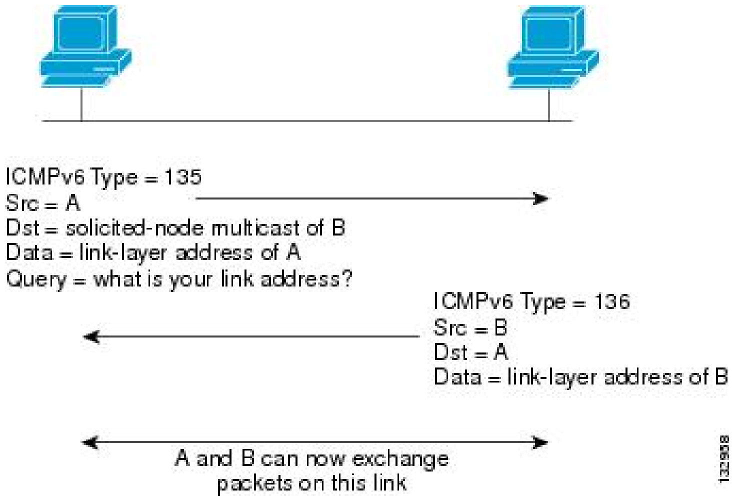
- very similar to ARP in IPv4
- based on *Neighbor Solicitation* and *Neighbor Advertisement* messages
  - a common multicast prefix is defined (`FF02:0:0:0:0:1:FF00::/104`)
  - the node looking for an L2-layer address takes last 24 bits of the IP address, whose L2-address it is looking for, and concatenates it with the prefix
    - e.g., looking for L2-address of `2AC0:56:A319:15:022A:FFF:FE32:5ED1`, it gets `FF02:0:0:0:0:1:FF32:5ED1`
    - (the destination address is a multicast address)
    - the last 24 bits ensure that the multicast group will contain just a few nodes (typically 1 or 0)
  - a **Neighbor Solicitation message** is sent to such a multicast address
    - the message contains the IPv6 address being resolved and the L2 address of the sending node
    - the neighbor has to listen for such messages in his multicast group(s) (based on his IPv6 address(es))

Neighbor Discovery – L2 address resolution II.

- once a node belonging to the particular multicast group receives a NS message, it answers with a **Neighbor Advertisement message**
  - *note:* there might be several nodes in the particular multicast group – just the one having the IPv6 address being resolved answers
- the answer contains:
  - all the IPv6 and L2 addresses the node has
  - an attribute:
    - *R (Router)* – the sender is a router
    - *S (Solicited)* – indicates whether the NA has been solicited or not (unsolicited NAs are possible)
    - *O (Override)* – indicates whether the new information should override the old information previously saved on the node(s)
- *unsolicited Neighbor Advertisement*
  - used in situations, when the node knows that his L2-address has changed
  - these messages are sent to multicast address containing all the nodes (`FF02::1`)

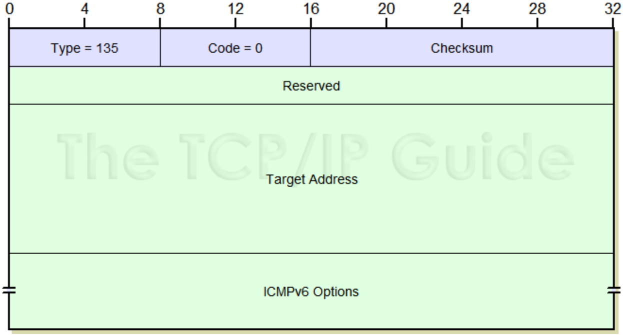
Neighbor Discovery – L2 address resolution II.

The mechanism



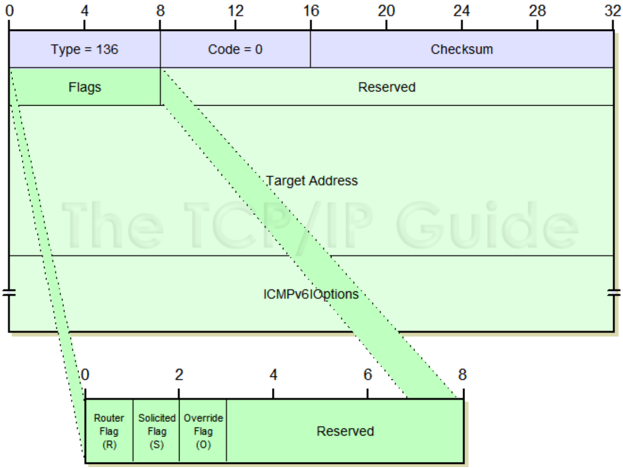
Neighbor Discovery – L2 address resolution II.

The mechanism – Neighbor Solicitation message format



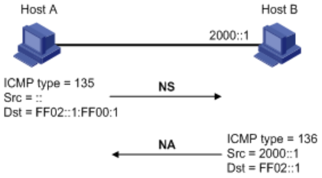
Neighbor Discovery – L2 address resolution II.

The mechanism – Neighbor Advertisement message format



Neighbor Discovery – Duplicate Address Detection (DAD)

- Duplicate Address Detection (DAD)
  - used during autoconfiguration process (see later)
  - the host sends NS message with its own address as the target address
    - destination address in the IPv6 header is set to the solicited-node multicast address
    - the source address is set to the unspecified address (::, i.e. all zeros)
  - if there is another node on the link that is using the same address as the hosts's address, it will reply with a NA message (sent to the all-nodes multicast address), thus exposing the duplicated address to the host

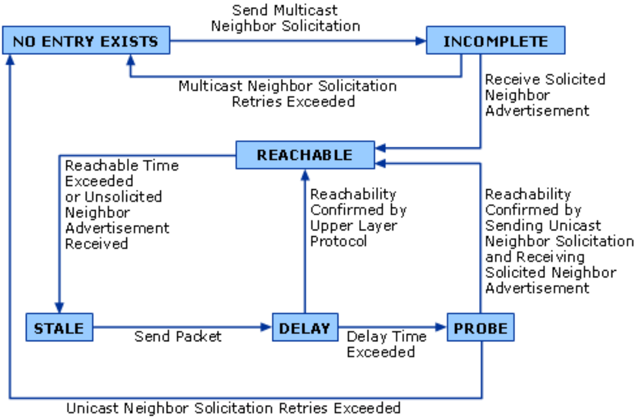


Neighbor Discovery – Neighbor Unreachability Detection (NUD)

- a node periodically controls the reachability of its neighbors (just the ones it is communicating with)
- can be achieved by two ways:
  - a higher-level protocol (e.g., the TCP) informs IPv6 that the communication proceeds and thus the host is alive
  - otherwise, the IPv6 has to perform such a detection on its own
- a cached address might be in one of the following states:
  - *incomplete* – address resolution is currently being performed and awaiting either a response or timeout (a NS has been sent, but the corresponding NA has not been received yet)
  - *reachable* – this neighbor is currently reachable (positive confirmation within the last *ReachableTime* has been received)
  - *stale* – more than *ReachableTime* milliseconds have elapsed since the last positive confirmation has been received
  - *delay* – the neighbor's reachable time has expired; an upper layer protocol might confirm the reachability within a specific time
  - *probe* – a reachability confirmation is being actively attempted

Neighbor Discovery – Neighbor Unreachability Detection (NUD)

The schema



Neighbor Discovery – Autoconfiguration

- designed to overcome a manual hosts configuration before connecting them to the network
  - even larger sites should not need a DHCP server to configure their hosts
  - a key feature required when all sorts of devices (TVs, refrigerators, DVD players, etc.) will use IP addresses
- IPv6 supports two types of autoconfiguration:
  - *Stateful autoconfiguration* – like DHCP in IPv4 world (here called DHCPv6)
  - *Stateless autoconfiguration* – a new type of autoconfiguration
  - (both might be combined)
    - stateless autoconfiguration may be used to generate IPv6 address, and
    - stateful autoconfiguration to obtain additional parameters (like DNS servers, etc.)

Neighbor Discovery – Stateless autoconfiguration

- RFC 2462
- assumes that there are clever wisemen (routers) in the network, who know everything necessary
  - from time to time, they inform all the nodes about current configuration (**Router Advertisements**)
  - a new node just waits for an RA or unsolicitedly asks for it (**Router Solicitation**)
- router advertisements:
  - periodically sent by every router
    - in random intervals to all the connected networks (via multicast destined to all connected hosts), or
    - as an answer to Router Solicitation message (via unicast to the host that has sent the RS)
  - contains specific information about the router
    - MTU
    - network prefixes
    - L2-address of the router's interface through which the RA has been sent
    - etc.

Neighbor Discovery – Stateless autoconfiguration

The mechanism I.

- to generate its IP address, a host uses a combination of local information (such as its MAC address or a randomly chosen ID), and information received from routers
- steps, which a device takes when using stateless autoconfiguration:
  - **Link-Local Address Generation –** the device generates a link-local address (so-called *tentative address*)
    - link-local addresses have `1111 1110 10` as first 10 bits (prefix `FE80`)
    - followed by **54 zeroes**
    - followed by **64 bit interface identifier** (the MAC address or a randomly chosen ID)
  - **Link-Local Address Uniqueness Test –** performed to ensure that the address the node had generated isn't already used in the local network
    - very unlikely an issue if the link-local address came from a MAC address, but more likely if it was based on a generated ID
    - the node sends a NS message and listens for a NA response (see *Duplicate Address Detection* mentioned before)

Neighbor Discovery – Stateless autoconfiguration

The mechanism II.

- cont'd:
  - **Link-Local Address Assignment** – assuming the uniqueness test passes, the device assigns the link-local address to its IP interface
    - this address may be used for communication on the local network, but not on the wider Internet (since link-local addresses are not routed)
  - **Router Contact** – next, the node attempts to contact a local router to obtain more information on continuing the autoconfiguration
    - performed either by listening for RA messages sent periodically by routers, or by sending a specific RS message to ask a router for information on what to do next (to the all-routers multicast group, i.e. `FF02::2`)
  - **Router Direction** – the router provides direction to the node on how to proceed with the autoconfiguration
    - may instruct the node that the "stateful" autoconfiguration is in use, and provide it with the IP address of a DHCP server
    - alternatively, it may instruct the host how to determine its global Internet address

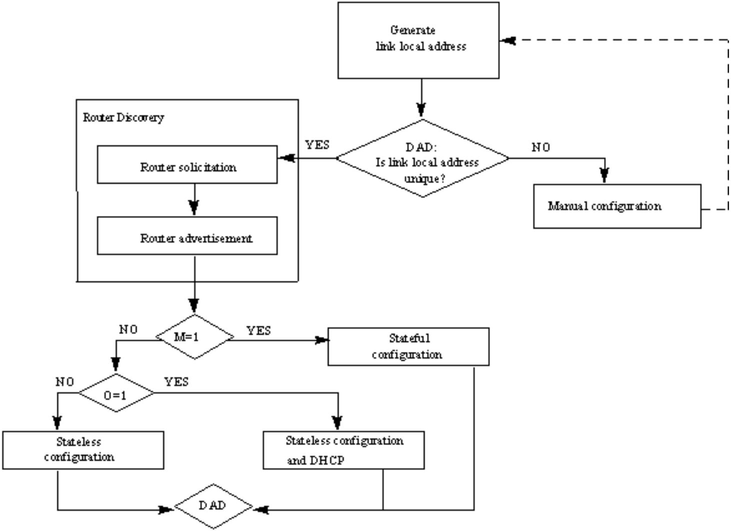
Neighbor Discovery – Stateless autoconfiguration

The mechanism III.

- cont'd:
  - **Global Address Configuration** – assuming that stateless autoconfiguration is in use, the host configures itself with its globally-unique Internet address
    - this address is generally formed from a network prefix provided to the host by the router, combined with the device's identifier generated in the first step

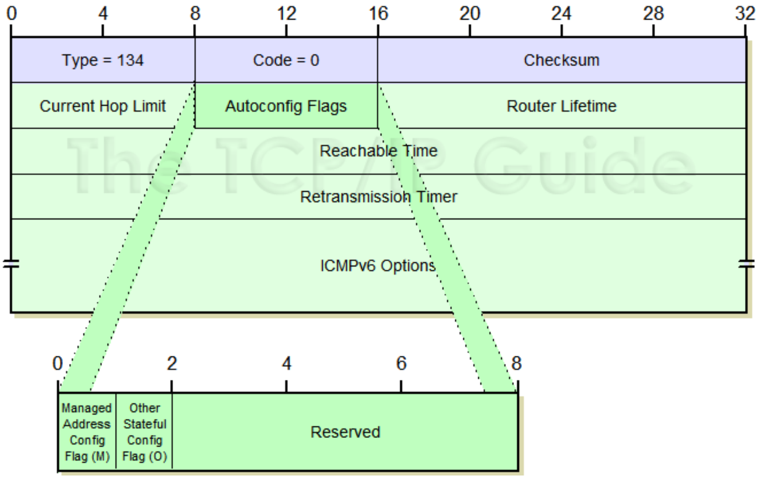
Neighbor Discovery – Stateless autoconfiguration

The schema



Neighbor Discovery – Stateless autoconfiguration

Router Advertisement I.



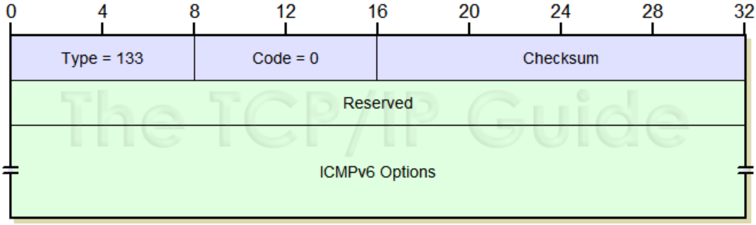
Neighbor Discovery – Stateless autoconfiguration

Router Advertisement II.

- *autoconfiguration flags*:
  - **M** *(Managed Address Configuration Flag)* – informs the hosts to use stateful method for address configuration (e.g., the DHCPv6)
  - **O** *(Other Stateful Configuration Flag)* – informs the hosts to use stateful method for information other than addresses
- *router lifetime* – informs the hosts how long this router should be used as a default router;
  - if set to 0, the router should not be used as a default router
- *reachable time* – informs the hosts how long they should consider a neighbor to be reachable after they have received reachability confirmation
- *retransmission timer* – the amount of time, in milliseconds, that a host should wait before retransmitting
- *ICMPv6 options* – RA messages may contain three possible options:
  - **source L2 Address** – included when the router sending the RA knows its L2 address
  - **MTU** – used to tell local hosts the MTU of the local network
  - **prefix information** – informs what prefix(es) to use for the local network

Neighbor Discovery – Stateless autoconfiguration

Router Solicitation I.



ICMPv6 options: if the device sending the RS knows its L2 address, it should be included

Neighbor Discovery Protocol

Summary

- *Neighbor solicitation (NS) message*
  - used to acquire the link-layer address of a neighbor
  - used to verify whether the neighbor is reachable
  - used to perform a duplicate address detection
- *Neighbor advertisement (NA) message*
  - used to respond to a neighbor solicitation message
  - when the link layer address changes, the local node initiates a neighbor advertisement message to notify neighbor nodes about the change
- *Router solicitation (RS) message*
  - once started, a host sends a router solicitation message to request the router for an address prefix and other configuration information (autoconfiguration)
- *Router advertisement (RA) message*
  - used to respond to a router solicitation message
  - a router periodically sends a router advertisement message containing information such as address prefix and flag bits
- *Redirect message*
  - the default gateway might send a redirect message to the source host so that the host can reselect a better/correct next hop router to forward its packets

Lecture overview I

Lecture overview II

IPv6 – Mobility Support I.

- **main idea:** even mobile devices are somewhere “at home”
  - i.e., their *home network* exists
- used addresses:
  - *Home Address* – a global unicast persistent address, through which a mobile node is always accessible (even though not being in its home network)
  - *Care-of Address* – a global unicast address for the mobile node while it is in a foreign network (the address is based on the network where the host is currently located)
- *Correspondent Node (CN)* – a peer node with which a mobile node is communicating
- *Home Agent (HA)* – a router in the home network, through which the mobile node is always accessible
  - receives datagram destined to the mobile node and forwards them (via a tunnel) to it
- *route optimization* – direct communication of the mobile and corresponding nodes
  - in order to optimize the communication
  - not necessary (the communication might proceed through the home agent all the time)

IPv6 – Mobility Support II.

How things work

- as long as the mobile node is at home, it receives packets through regular IP routing mechanism and behaves like any other host
- when the mobile node is away from the home network, it has an additional care-of address (received via a mechanism available in the foreign network)
  - the association of home address and care-of address is called *binding*
- the mobile node registers its care-of address with a router on its home link (its *Home Agent (HA)*)
  - being either statically or dynamically (discovery requests sent to HAs anycast address) configured
- there are two ways to communicate for a correspondent node and a mobile node:
  - *bidirectional tunneling* – packets from the correspondent node are sent to the HA, which encapsulates them and sends them to the mobile node's care-of address (and vice versa)
  - *route optimization* – the communication between the mobile node and correspondent node can be direct without the usage of the HA
    - the mobile node has to register its care-of address with the correspondent node, and
    - the binding has to be authorized through the *Return Routability Procedure*

IPv6 – Mobility Support II.

The schema

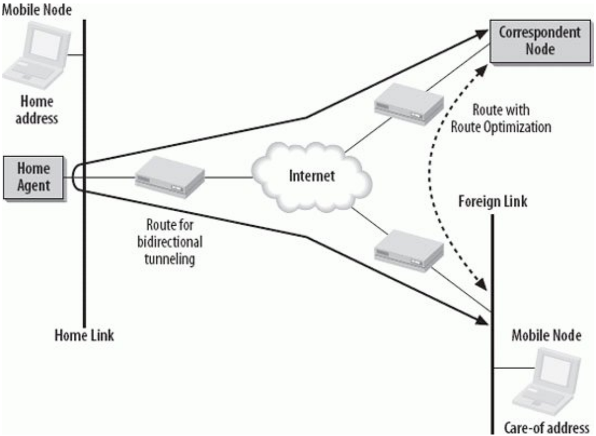


Figure: An illustration of home agent's functionality in IPv6.

IPv6 – Mobility Support II.

Return Routability Procedure

- mobile node has to prove to correspondent node that it owns both home address and care-of address
  - but mobile node does not share any secret with the correspondent node
  - initially (in older times) performed using *IPsec*
    - however, there is no world-wide Public Key Infrastructure (PKI) available for the nodes
- *Return Routability (RR) Procedure*
  - RFC 3775
  - enables the correspondent node to obtain some reasonable assurance that the mobile node is addressable through its claimed care-of address as well as through its home address
    - only when successfully proven, the route optimization might take place
    - reduces the risk of a security attack (a harmful node working off the mobile node)

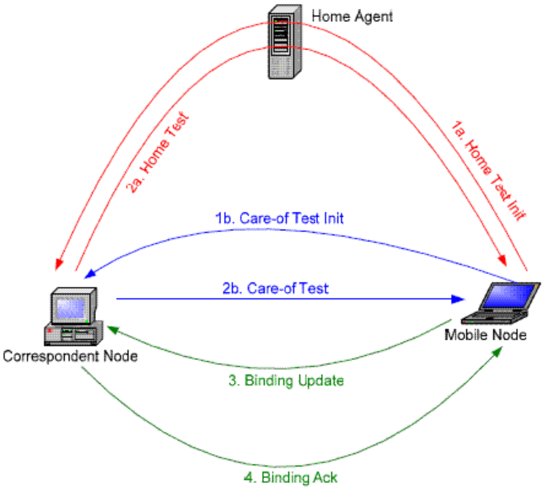
IPv6 – Mobility Support II.

Return Routability Procedure – the steps

1. MN sends a **Home Test Init (HoTI)** message via HA to the CN (this message carries a *Home Init Cookie*)
    - this way the CN learns the home address of the MN
2. MN sends a **Care-of Test Init (CoTI)** message to the CN (this message carries a *Care-of Init Cookie*) – this is sent to the CN directly (not through the HA)
    - this way the CN learns the care-of address of the MN
3. CN replies to the Home Test Init message with a **Home Test (HoT)** message sent via HA (this message carries the *Home Init Cookie* and the *Home Nonce Index*)
    - the MN can now generate a *Home Keygen Token*
4. CN replies to the Care-of Test Init message with a **Care-of Test (CoT)** message sent to the MN's care-of address (this message carries the *Care-of Init Cookie* and the *Care-of Nonce Index*)
    - the MN can now generate a *Care-of Keygen Token*
5. both the MN and the CN compute a 20-byte *Management Key*, which is used to secure the Binding Update messages
    - having the correct *Management Key* the MN has proven that it is reachable both via its home and care-of addresses

IPv6 – Mobility Support II.

Return Routability Procedure – the schema



IPv6 – Mobility Support III.

Home Agent Functionality

Home Agent:

- maintains binding cache and a list of home agents
  - every router, that sits on the same link and provides home agent services, must be listed
- processes bindings
  - indicates primary care-of address
  - processes care-of addresses' changes/removals
- tunnels received packets to care-of address
  - performs Neighbor Advertisements by the name of mobile node
- supports *Home Agent Address Discovery*
  - normally, mobile nodes are configured statically with a home agent's address
  - once a home agent is renumbered (or goes down being replaced by another HA with a different IP), dynamic discovery of the HA's address takes place
    - *Home Agent Address Discovery Request* (sent using home agents' anycast address) and *Home Agent Address Discovery Reply* messages
    - see details in the literature

Lecture overview I

Lecture overview II

General Security Practices I.

- standard network security practices involve two “triads” of thought, **CIA** and **AAA**

- **Confidentiality:**
  - stored or transmitted information cannot be read or altered by an unauthorized party
- **Integrity:**
  - any alteration of transmitted or stored information can be detected
- **Availability:**
  - the information in question is readily accessible to authorized users at all times

General Security Practices II.

- **Authentication:**
  - ensuring that an individual or group are really the one who they say they are (the act of clarifying a claimed identity)
- **Authorization:**
  - ensuring that the authenticated user or group has the proper rights to access information they are attempting to access
- **Accounting:**
  - the act of collecting information on resource usage (e.g., a log)

- **Nonrepudiation:**
  - not included in the CIA/AAA Triads
  - means that a specific action (such as sending, receiving, or deleting of information) cannot be denied by any of the parties involved

General Security Practices III.

- the security requirements need to be provided by two basic security elements:
  - *encryption* – to provide confidentiality
  - *secure checksums* – to provide integrity
  - suitable combinations of both may be used to provide more complex services like authenticity and nonrepudiation
- there are two forms of encryption commonly used:
  - *Secret Key Cryptography (Symmetric Cryptography)* – sender and recipient have to agree on a shared secret
  - *Public Key Cryptography (Asymmetric Cryptography)* – encryption algorithm uses a key pair consisting of a public and private keys
- *message digest (hash)* – a function which takes input of an arbitrary length and outputs fixed-length (unique) code

IPv6 – Security Support

- a general security mechanisms are described by *IPSec* (RFC 2401, updated by RFC 4301)
  - both for IPv4 and IPv6
    - IPv4: IPSec **may** be installed separately
    - IPv6: IPSec is a **mandatory** and integral part of the IPv6 stack
    - ⇒ IPv6 is *not* more secure than IPv4
- elements of IPSec framework:
  - a protocol for authentication – *AH (Authentication Header)*
  - a protocol for encryption – *ESP (Encapsulating Security Payload)* header
  - a definition for the use of cryptographic algorithms for encryption and authentication
  - a definition of security policies and security associations between communicating peers
  - key management
- see the animation: `http://www.sitola.cz/~jeronimo/vyuka/IPSec`

IPv6 – Security Support

Security Associations

- **Security Associations (SA)**:
  - a set of security information that describes a particular kind of secure connection between one device and another
  - three elements:
    - a key
    - an encryption or authentication mechanism
    - additional parameters for the algorithm (counters, duplicity protection, etc.)
  - one-way agreements ⇒ to provide encrypted and authenticated duplex communication, 4 SAs are necessary
- an SA is defined by a set of three parameters:
  - *Security Parameter Index (SPI)* – a 32-bit number chosen to uniquely identify a particular SA
  - *IP Destination Address* – the address of the device for whom the SA is established
  - *Security Protocol Identifier* – specifies whether this association is for AH or ESP

IPv6 – Security Support

Key Management

In order to establish an SA, the peers have to agree on a cryptographic algorithm and negotiate keys

- the negotiation often happens over insecure paths
- several protocols proposed for an automated negotiation:
    - old approach: *Internet Security Association and Key Management Protocol (ISAKMP)* (RFC 2407 and 2408) and *Internet Key Exchange version 1 (IKEv1)* (RFC 2409)
    - current approach: *Internet Key Exchange version 2 (IKEv2)* (RFC 4306)
        - simplifies IKEv1 (consolidates RFCs 2407, 2408, and 2409 into a single document)
        - fixes bugs and ambiguities
        - tries to remain as close to IKEv1 as possible

IPv6 – Security Support

Key Management – Internet Key Exchange version 2 (IKEv2)

Internet Key Exchange version 2 (IKEv2)

- automatically establishes SAs and creates/deletes cryptographic material
- authenticates communicating peers
- works in 2 phases:
  1. establishes a secure channel to negotiate the data protection cryptographic material
     - results in a single ISAKMP/IKE SA
  2. establishes the secure channel for the transmission of data
     - results in a pair of IPsec SAs

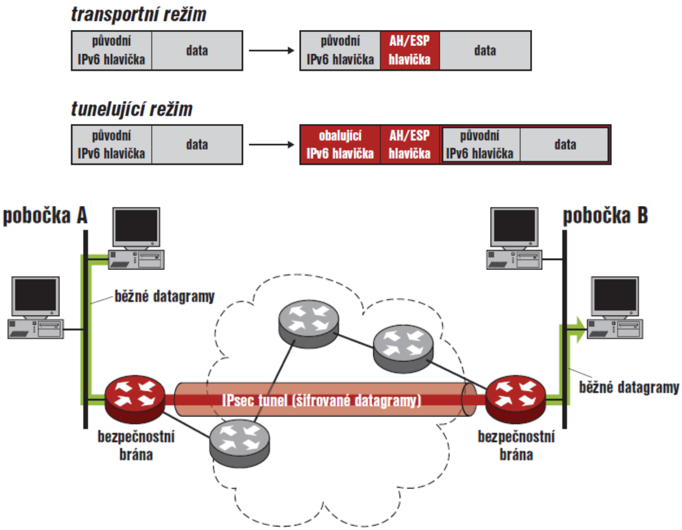
IPv6 – Security Support

IPSec Modes I.

IPSec differentiates two modes of transport:

- *Transport mode*
  - the protocol protects the messages passed down to IP from the transport layer
  - the message is processed by AH/ESP and the appropriate header(s) are added in front of the transport (UDP or TCP) header
  - the IP header is then added in front of that by IP
- *Tunnel mode*
  - IPSec is used to protect a **complete encapsulated IP datagram** after the IP header has already been applied to it
  - the IPSec headers appear in front of the original IP header, and a new IP header is added in front of the IPSec header
  - i.e., entire original IP datagram is secured and encapsulated within another IP datagram

IPv6 – Security Support
IPSec Modes II.



IPv6 – Security Support

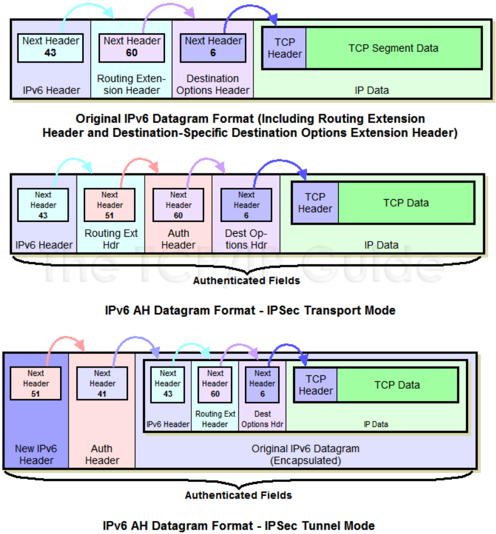
Authentication Header (AH) I.

AH (Authentication Header)

- a protocol that provides **authentication** of either all or part of the contents of a datagram
- performed by an addition of a header calculated based on the values in the datagram
- protocol steps:
  1. a SA has to be set up between the two communicating devices
     - just the source and destination know how to perform the computation but nobody else can
  2. on the source device, AH performs the computation and puts the result (called the *Integrity Check Value (ICV)*) into a special header with other fields for transmission
  3. the destination device does the same calculation using the key the two devices share, which enables it to see immediately if any of the fields in the original datagram were modified
- the presence of the AH header allows to verify the integrity of the message, but doesn't encrypt it
  - the AH provides just **authentication**, not **privacy**!

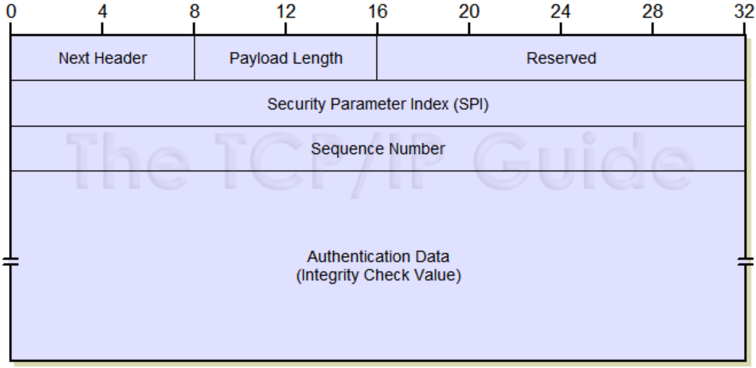
IPv6 – Security Support

Authentication Header (AH) II. – header placement



IPv6 – Security Support

Authentication Header (AH) II. – header format



IPv6 – Security Support

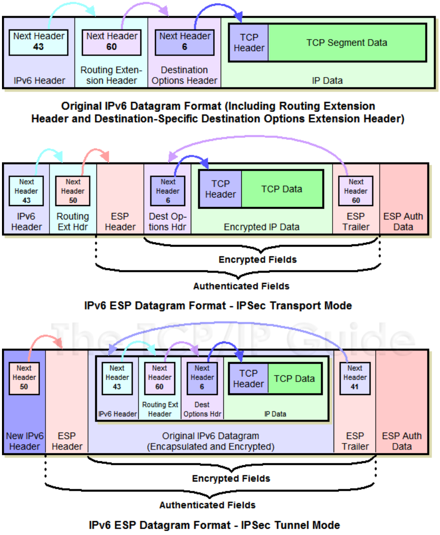
Encapsulating Security Payload (ESP) Header I.

ESP (Encapsulating Security Payload)

- a protocol that protects data against being examined by a non-authorized party
  - performed by data encryption
  - provides **privacy**
- ESP also supports its own **authentication scheme** like that used in AH
- instead of having just a header, ESP divides its fields into three components:
  - *ESP Header* – contains two fields (the SPI and Sequence Number) and comes before the encrypted data
  - *ESP Trailer* – placed after the encrypted data (contains padding that is used to align the encrypted data)
  - *ESP Authentication Data* – when ESP's optional authentication feature is used, this contains an *Integrity Check Value (ICV)*, computed in a similar way like in AH case

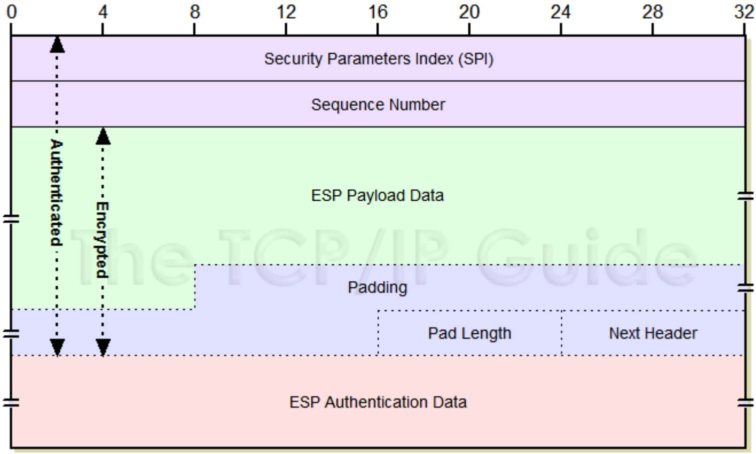
IPv6 – Security Support

Encapsulating Security Payload (ESP) Header II. – header placement



IPv6 – Security Support

Encapsulating Security Payload (ESP) Header II. – header format



IPv6 – Security Support

Why AH?

Since ESP provides authentication, is AH necessary?

- *yes*
- an authentication is often enough
- AH does not require as computational power as the ESP does
  - ESP uses stronger encryption algorithms
- AH authenticates the whole datagram
  - ESP does not authenticate the outer IP header

Lecture overview I

1. Lecture Overview
2. Brief IPv6 Introduction
   - IPv6 Addresses
   - Path MTU discovery
3. IPv6 Neighbor Discovery Protocol in Detail
   - L2 address resolution
   - Duplicate Address Detection
   - Neighbor Unreachability Detection
   - Autoconfiguration
   - Summary
4. IPv6 Mobility Support in Detail
   - Return Routability Procedure
5. IPv6 Security in Detail
   - General Security Practices
   - IPv6 – Security Support

Lecture overview II

6. IPv6 QoS Support in Detail
   - Integrated Services
   - Differentiated Services
   - QoS and IPv6

7. IPv6 Transition
   - Porting Applications
   - IPv6 and IPv4 Worlds' Interoperability

8. IPv6: Literature

9. IPv6: Where we are?

QoS in the Internet

- IPv4 is based on a simple packet forwarding model
  - all packets are treated alike – they are forwarded with *best effort* treatment according to "first-come, first-served" principle
  - there are no options to control flow parameters like delay, jitter, or bandwidth allocations
- two main architectures for providing data streams with priorities and quality guarantees were proposed:
  - *Integrated Services*
    - based on the paradigm that bandwidth and all related resources are reserved per flow on an end-to-end basis (routers store information about flows and analyze each passing packet)
  - *Differentiated Services*
    - based on packets' markup (assigning the packets a certain priority class and their serving in the inner network nodes based on that priority)

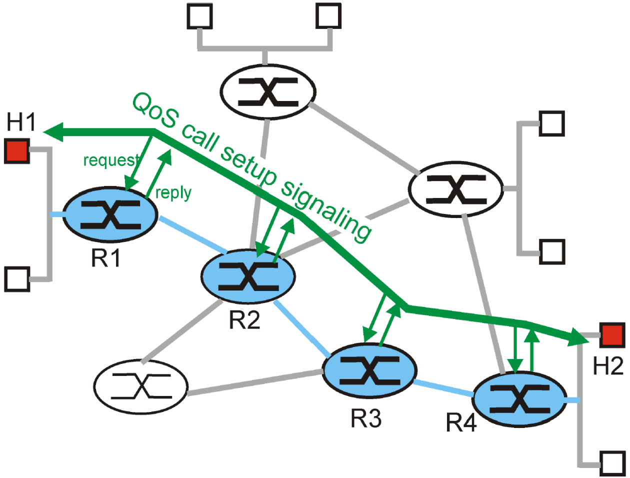
Integrated Services

Intergrated Services:

- an application announces the network with its qualitative requirements
- the network checks, whether the required resources are available, and decides, whether the request will be satisfied (so-called *Admission Control* phase)
- if it's not possible to satisfy the requirements, the connection is refused
  - the application might decide about reducing its requirements
- if the requirements could be satisfied, the network informs all the components on the path to the receiver about necessary resources' reservations (queues size, their priority, etc.)
  - a reservation protocol has to be used
    - e.g., the *Resource reSerVation Protocol (RSVP)* (RFC 2205) or the *YESSIR (YEt another Sender Session Internet Reservations)*
- *main drawback:*
  - it's necessary to maintain a state in the inner network nodes (⇒ scalability problems)

Integrated Services

Resource Reservation Illustration

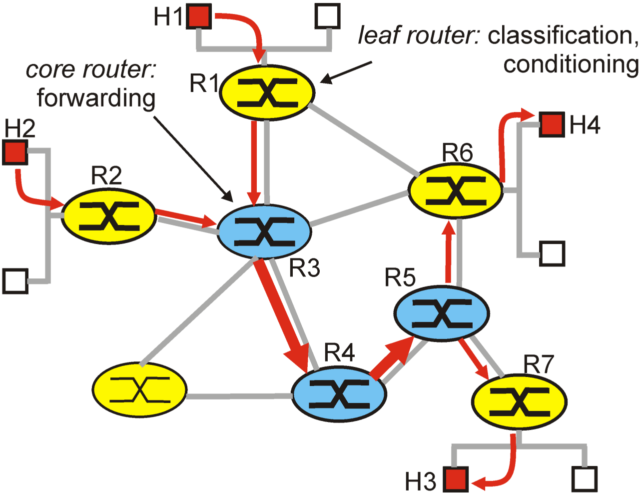


Differentiated Services

- a precise definition of required QoS parameters is not always necessary
  - usually, a guarantee that the transmission quality will not become worse when the network becomes (over)loaded is sufficient
- ⇒ **Differentiated Services**
  - no necessity to inform the network with transmission quality requirements
    - resource reservation protocols are not necessary
  - each packet is marked by a tag indicating a priority class before being sent to the network
    - packets are marked when entering the network only
    - a tag is put into *Type of Service (IPv4)* or *Traffic Class (IPv6)* fields
    - the packet is processed on the inner network nodes based on its priority class (the inner network nodes just read the tag and handle the packet based on it)
  - *main advantage:*
    - simple (for implementation in applications as well as inner network nodes)
    - no state information in the inner network nodes (⇒ good scalability)
    - no initial delay required by the necessity to perform resource reservations

Differentiated Services

Packet Classification Illustration



QoS and IPv6

Traffic Class field I.

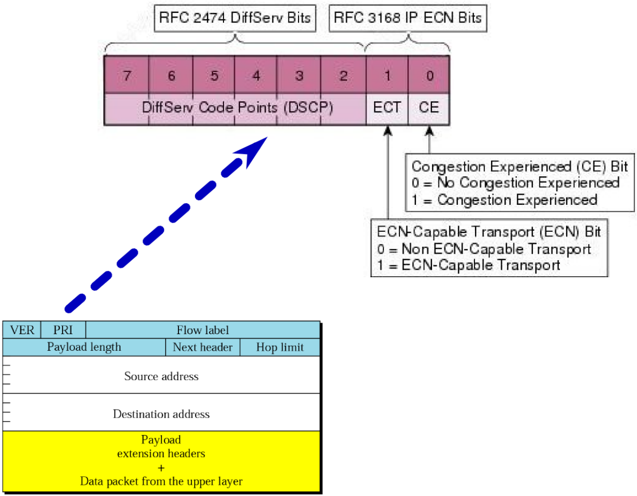
Two IPv6 header fields can be used for QoS:

Traffic Class field:

- sometimes referenced as *Packet Priority* (PRI) field
- 1-byte field
- its use specified in RFC 2474
  - introduces the term "DS field" (DiffServ field) for the Traffic Class field
  - DiffServ routers have a known set of DS routines which are determined by the 6-bit value (*DiffServ CodePoints – DSCP*) in the DS field
    - 64 different codepoints can be specified
    - coding rules are specified in RFC 3140 (assigned by IANA)
    - these DSCP values specify how packets should be forwarded
    - (a default behavior denominated by an all-zeros DSCP must be provided by any DS router (*best-effort service*))
  - last 2 bits (*Explicit Congestion Notification – ECN value*) are specified in RFC 3168
    - four possible codepoints used for Congestion Notification
    - using them, a router can signal overload before a packet loss occurs

QoS and IPv6

Traffic Class field II.



QoS and IPv6

Flow Label field I.

A *flow* is a sequence of related packets sent from a source to a unicast, anycast, or multicast destination.

Flow Label field:

- a 20-bit field which enables classification of packets belonging to a specific flow
  - IPv6 routers must handle all the packets belonging to the same flow in a similar fashion
    - when routers receive the first packet of a new flow, they can process the information carried by the headers (IPv6 header, Routing header, and Hop-by-Hop extension headers) and store the result in a cache memory
    - this information can be used to route all other packets belonging to the same flow (when having the same source address and the same Flow Label, it does not require to examine all those headers)
- how to use this field efficiently is still an open issue

QoS and IPv6

Flow Label field II.

- traditionally, flow classifiers have been based on the 5-tuple: source and destination addresses, ports, and the transport protocol type
  - the classifier must use transport next header value and port numbers ($\Rightarrow$ less efficient)
  - some of these fields may be even unavailable due to either fragmentation or encryption
- IPv6 uses just the triple of the Flow Label and the Source and Destination Address fields
  - it enables efficient IPv6 flow classification
  - only IPv6 main header fields in fixed positions are processed
- IPv6 source nodes supporting the flow labeling must be able to label known flows (e.g., TCP connections, application streams)
  - even if they do not require any flow-specific treatment
  - a Flow Label of zero is used to indicate packets not being part of any flow

Lecture overview I

Lecture overview II

IPv6 Transition – Porting Applications

Applications using IPv4 network sockets need to be converted to IPv6:

- this conversion can be very simple (simple programs) or a challenging effort (complex network applications)
- the programmer has to make a decision, whether the program will be IPv4-only, separated IPv4 and IPv6, or IP protocol version independent (recommended for most cases)
    - IP protocol version independent code makes the code agnostic to the IP protocol version

IPv6 Transition – Porting Applications

Issues I.

Application porting issues under IPv6:

- *Address parsing*
  - IPv4 dotted decimal addresses are trivial to parse
  - IPv6 hex-colon addresses require a library support for input or output
    - a complete input parser can be a few hundred lines of code
    - rendering an address in canonical form involves complex analysis of the address
- *Address memory space*
  - legacy code often stores IPv4 addresses in 32-bit `unsigned integer` variables
    - native data type
    - makes masking operations easy
    - fewer details to remember
  - few machines have a native 128-bit data type
    - all the code has to be changed in order to use the appropriate structure (see later)

IPv6 Transition – Porting Applications

Issues II.

- *URL and text representation of IP addresses*
  - original standards for URLs and URIs do not allow IPv6 addresses in URLs
    - the problem is the colons in hex-colon notation (used for port specification in IPv4)
  - RFC 3986 modifies standard to allow IPv6 addresses in brackets
    - `http://[fe80::219:d1ff:fe06:e908]:8080/`
    - a lot of legacy code does not accept this
- *Multiple addresses*
  - in the IPv4 world, the vast majority of systems have one address per interface
  - IPv6-enabled stack handles multiple IP addresses on a single interface (e.g., one IPv4 address, one global IPv6 address and one link-local IPv6 address)
  - the code must take this into account (e.g., when querying the DNS for server addresses, the client code should loop through all the received IP addresses until one is answering)

IPv6 Transition – Porting Applications

Useful structures and functions I.

Structures:

- `struct addrinfo`
  - a replacement of the `hostent` structure
  - holds connection information used in handling name to IP address resolution
  - it's used by `getaddrinfo` function
- `struct sockaddr_in6`
  - IPv6 version of `sockaddr_in` structure
  - holds IP address and port number of a connection, IPv6 flow label and scope of the address
  - it's used in socket calls as place holder for IPv6 addresses
    - it is specific to IPv6 – it is not IP version independent and thus should be avoided
- `struct sockaddr_storage`
  - a struct defined for casting either `sockaddr_in` or `struct sockaddr_in6`
  - this should be used when making a program IP version independent

IPv6 Transition – Porting Applications

Useful structures and functions II.

Functions:

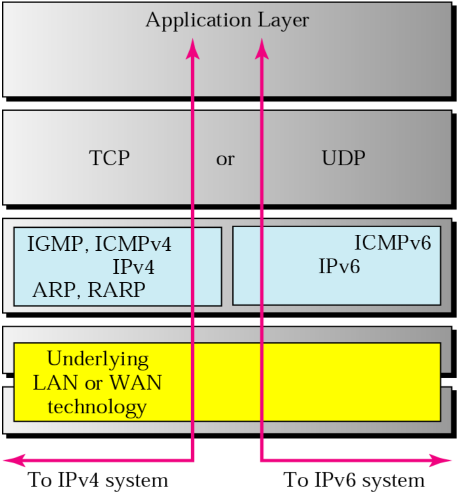
- `getaddrinfo`
  - a replacement of the `gethostbyname` function
  - it queries the DNS for the IP addresses of a hostname
  - the result is linked list of addresses, which should be traversed by the calling program
- `getnameinfo`
  - replacement of the `gethostbyaddr`, `inet_addr` and `inet_ntoa` functions
  - it queries the DNS for the hostname of an IP address
  - the result is the hostname string

IPv4 & IPv6 Interoperability

- during the IPv6 proposal, a gradual transition from IPv4 has been taken into account
    - $\Rightarrow$ a mechanism for IPv4 and IPv6 co-existence is necessary
- 3 main categories:
    - *Dual stack*
        - a device supports both IPv4 and IPv6 simultaneously
        - allows IPv4 and IPv6 to coexist in the same devices and networks
        - RFC 4213
    - *Tunneling*
        - IPv6 datagrams are encapsulated into IPv4 datagram's data
        - allows the transport of IPv6 traffic over the existing IPv4 infrastructure
        - 6to4 (RFC 3056), Teredo (RFC 4380), ISATAP (RFC 5214)
    - *Translators (NAT-PT)*
        - a device translates IPv6 datagrams into IPv4 datagrams (client $\rightarrow$ server direction) and vice versa
        - allows IPv6-only nodes to communicate with IPv4-only nodes

IPv4 & IPv6 Interoperability

Dual Stack



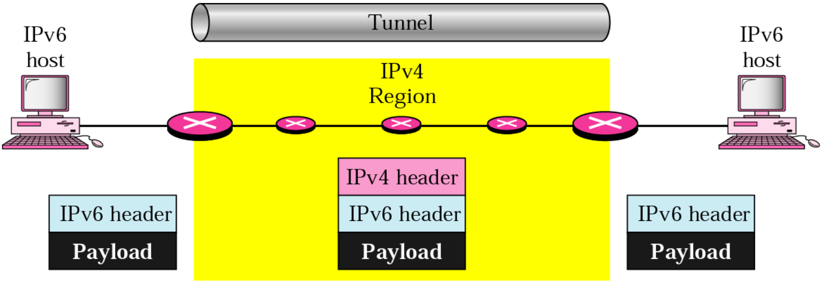
IPv4 & IPv6 Interoperability

Dual Stack – features & drawbacks

- *main advantage:* easy to use and flexible
  - host can communicate with IPv4 hosts using IPv4 or communicate with IPv6 hosts using IPv6
  - once everything becomes upgraded to IPv6, IPv4 stack can be simply disabled/removed
  - offers greatest flexibility in dealing with islands of IPv4-only applications, equipment and networks
- *disadvantages:*
  - two separate protocol stacks have to be running (resource consumption)
  - all applications must be capable of determining whether the host is communicating with an IPv4 or IPv6 peer
  - DNS resolver must be capable of resolving both IPv4 and IPv6 address types
  - routing protocol must deal with both protocols (or separate protocols for IPv4 routing and IPv6 routing have to be used)
  - etc.

IPv4 & IPv6 Interoperability

Tunneling



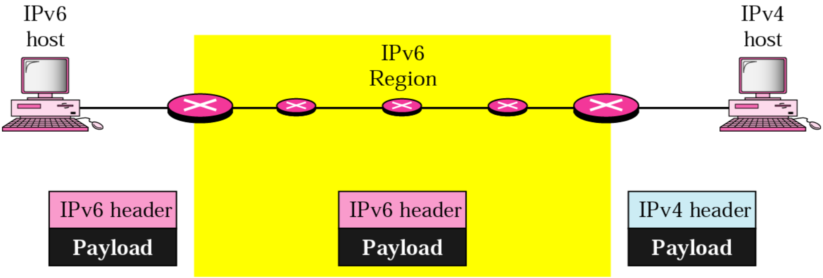
IPv4 & IPv6 Interoperability

Tunneling – features & drawbacks

- *main advantage:* allows to migrate to IPv6 just the way one likes
  - there is no specific upgrade order that has to be followed (separate clouds could be interconnected via tunnels)
  - once everything becomes upgraded to IPv6, no changes are necessary (the tunneling points are just discarded)
- *disadvantages:*
  - additional load is put on the routers
  - tunnel end-points represent single points of failure
  - troubleshooting gets more complex
    - for example, one might run into hop count or MTU size issues, as well as fragmentation problems
  - management of encapsulated traffic (e.g., per-protocol accounting) is also more difficult due to encapsulation
  - tunnels also offer points for security attacks
  - etc.

IPv4 & IPv6 Interoperability

Translators (NAT-PT)



IPv4 & IPv6 Interoperability

Translators (NAT-PT) – features & drawbacks

- should be used only if no other technique is possible
  - just as a temporary solution until one of the other techniques can be implemented
- *advantage:* IPv6 hosts can directly communicate with IPv4 hosts (and vice versa)
- *disadvantages:*
  - does not support the advanced features of IPv6 (such as end-to-end security)
  - poses limitations on the design topology
    - replies have to come through the same NAT router through which requests have been sent
  - NAT router is a single point of failure
  - all applications having IP address in the payload of the packets will stumble

IPv6: Literature

- relevant RFCs
- Satrapa P.: *IPv6.* CZ.NIC association, 2008.
  Available online: `http://knihy.nic.cz/files/nic/edice/pavel_satrapa_ipv6_2008.pdf`
- Hagen S.: *IPv6 Essentials.* O'Reilly Media, Inc., 2006.
- Blanchet M.: *Migrating to IPv6.* John Wiley & Sons, Ltd., 2005.
- `http://www.tcpipguide.com`
- `http://www.ipv6.cz`

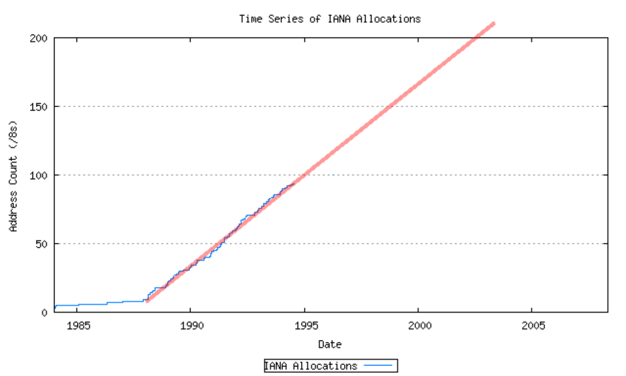
Pracovní skupina pro IPv6

Pavel Satrapa
Pavel.Satrapa@tul.cz

Špetka historie

Adresy – polovina 90. let

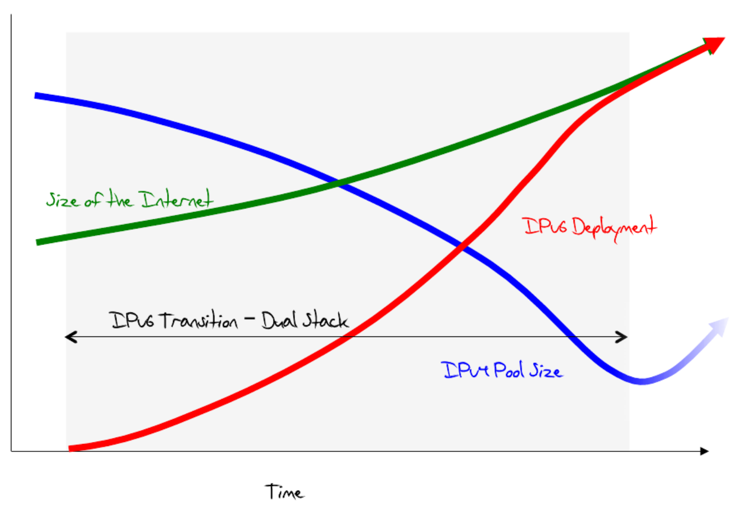
- IPv4 adresy dojdou kolem roku 2003
  - některé kategorie (třída B) mnohem dříve



Návrh nové verze IP

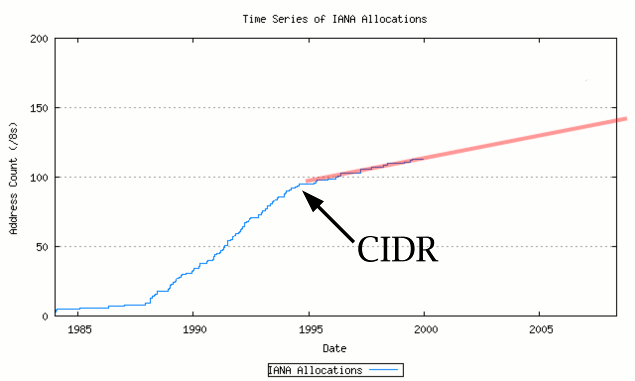
- času je dost – neomezí se jen na prodloužení adresy
- lze začlenit **nové vlastnosti**
  - obrovský adresní prostor (128b adresa)
  - lepší směrování – vysoké rychlosti, QoS
  - automatická konfigurace
  - vestavěné bezpečnostní mechanismy (IPsec)
  - podpora mobility
  - umožnit postupný přechod z IPv4 na IPv6

Přechod k IPv6 – idea



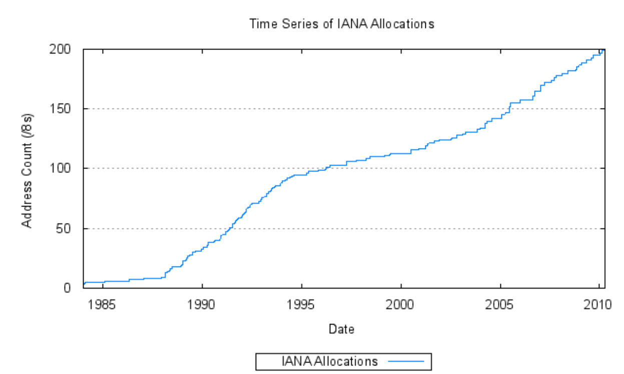
Adresy kolem roku 2000

- IPv6 nebylo samo – CIDR, NAT
  - IPv4 adresy vydrží do roku 2030, pohoda, klídek, tabáček



Adresy dnes

- **IPv4 adresy dojdou na podzim 2012**
  - Houstone máme tu problém



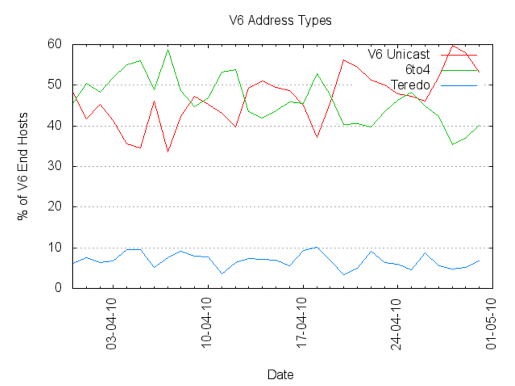
zdroj:
Geoff Huston,
ipv4.potaroo.net

IPv6 je stále raritou

- průměrné **datové toky v AMS-IX**, duben 2010
  - IPv4: 570 Gb/s, stoupá
  - IPv6: 1 Gb/s, stagnuje ... **0,2 %**
- statistiky web serverů:
  cca **1 % návštěvníků** přichází po IPv6

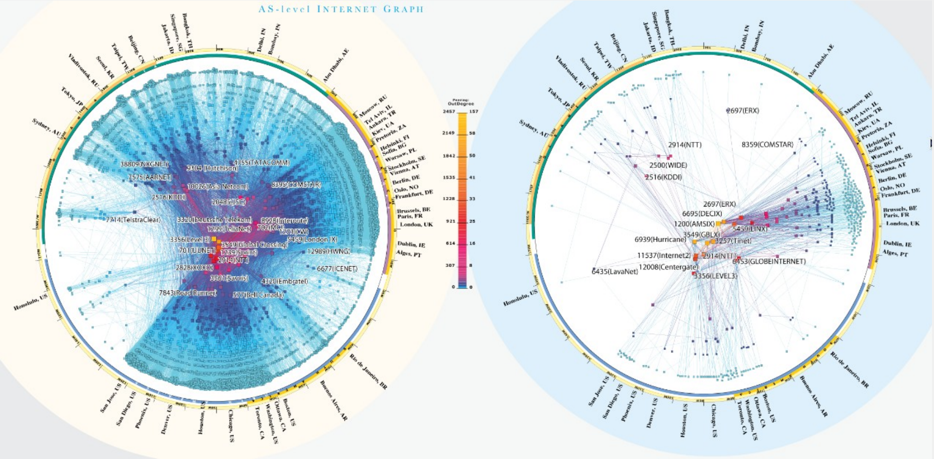
Průzkum konektivity

- aktuální studie ukazují, že v optimistickém případě
  - **5 % podporuje** IPv6
  - **2 % preferuje** IPv6
  - **0,27 % pouze** IPv6
  - nativních je zhruba stejně jako 6to4



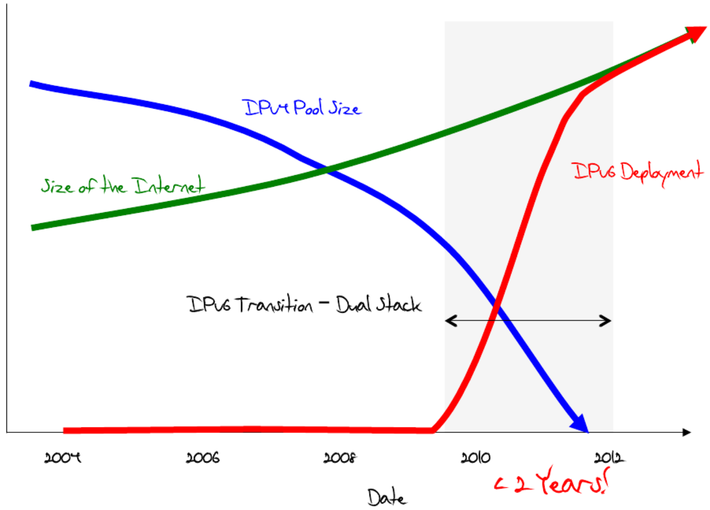
  - http://www.potaroo.net/ispcol/2010-04/ipv6-measure.html

Topologická mapa

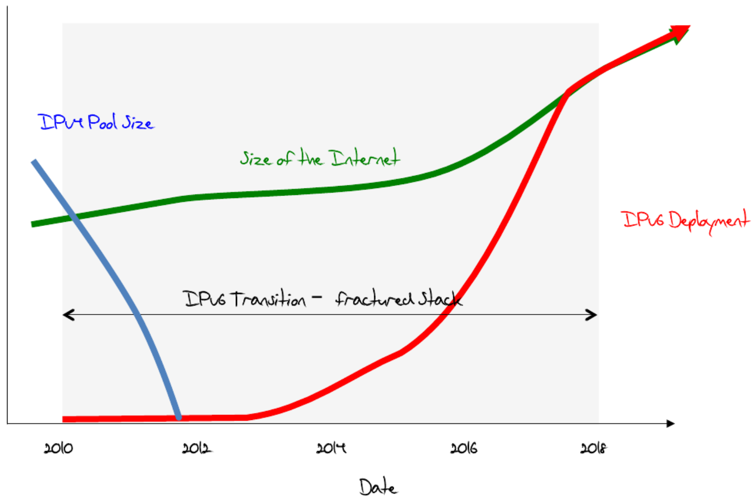


http://www.caida.org/research/topology/as_core_network/

Málo času na přechod ...



... asi dopadneme takto



IPv6 v CESNETu

- experimentálně od 1999
- produkčně od 2004
- zcela rovnocenná podpora IPv4 i IPv6
- zrušeno jako samostatná aktivita, převedeno do provozu – máme hotovo

jenže...

IPv6 v připojených sítích

- **28 sítí členů sdružení**
  - **11 podporuje IPv6**
  - 6 plánuje nasazení během 6 až 12 měsíců
  - 10 zatím ani nepožádalo o adresy
- **cca 300 zákaznických sítí**
  - **1 podporuje IPv6**

Pracovní skupina CESNETu

Cíle

- podpořit nasazení IPv6 v sítích organizací připojených k CESNET2
  - výměna zkušeností
  - diskusní platforma
  - návody, doporučené postupy
  - odborné akce
  - obecná propagace

Účast ve skupině

- neformální – čistě na základě zájmu
- weby:
  - www.ipv6.cz
    obecný web o IPv6, spravován členy skupiny
  - http://www.cesnet.cz/ipv6/wg/
    vlastní stránky skupiny, orientovány na CESNET, prezentace se schůzek apod.
- mailing list ipv6-wg@cesnet.cz
- pracovní schůzka čtvrtek 19.00 salónek 1

Děkuji za pozornost

CESNET