

NSWI090: Computer Networks

<http://www.ksi.mff.cuni.cz/~svoboda/courses/212-NSWI090/>

Lecture 9

Protocols

Martin Svoboda

martin.svoboda@matfyz.cuni.cz

2022

Charles University, Faculty of Mathematics and Physics

Lecture Outline

IPv4 protocol

- **Datagram structure**
 - Meaning and usage of individual header fields
- **Fragmentation** of datagrams
 - Motivation
 - Strategies
 - Process

Lecture Outline

ICMPv4

- Message structure and basic message types

ARP

- Translation of IP addresses to hardware addresses

RARP

- Translation of hardware addresses to IP addresses

DHCP

- Configuration of nodes including assignment of IP addresses

IPv6

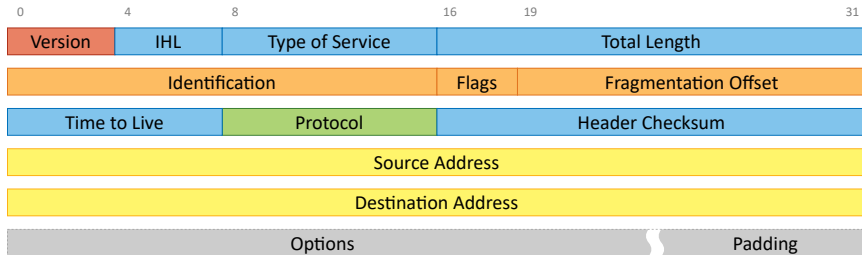
- Packet structure, header fields, and fragmentation

ICMPv6

IPv4 Datagrams

Datagram structure

- **Header**
 - **Required fields** as well as **optional fields** \Rightarrow **variable length**
 - Must be aligned to integral multiples of 4 bytes
- **Body** (payload)
 - **TCP segment, UDP datagram, ...**



Header Fields

Version (4 bits)

- Allows to mutually **distinguish** individual L3 **protocols**
 - **Fixed to value 4** (for IPv4)
 - Analogously, IPv6 has value 6 at the same position

Type of Service (ToS) (8 bits)

- Kind of a *forgotten byte*
 - Its exact originally intended meaning is no longer known
- **Various purposes over the years**
 - Redefined for several times and never actually used widely
 - Always related to various **Quality of Service** aspects
 - Nowadays ignored
 - Or exploited within **DiffServ** (**D**ifferentiated **S**ervices)

Header Fields

Internet Header Length (IHL) (4 bits)

- Overall header length
 - Expressed in integral multiples of 4 bytes
- **Only compulsory header fields are usually present**
 - And so the minimal header length is also the usual one
 - I.e., 20 bytes (IHL = 5)
 - 4 bits are available \Rightarrow maximal length is 60 bytes (IHL = 15)

Total Length (16 bits)

- Overall datagram length
 - I.e., header and body (payload) together
- 16 bits are available \Rightarrow **maximal IP datagram size is 64 kB**
 - Much smaller datagram sizes occur in practice, though
 - Because of MTUs introduced by real-world L2 technologies

Header Fields: TTL

Time to Live (TTL) (8 bits)

- **Limits a time for which a given datagram is supposed to exist**
 - Originally intended as a real-world time in seconds
 - Nowadays used as a **Hop Count**
 - Works as a **decreasing counter**
 - Protects from indefinite dissemination caused by **loops**
- Sender sets TTL to a certain **initial value**
 - Maximal value is 255, recommended initial is 64
- **Each router on the way...**
 - Current TTL value is **decremented by 1**
 - Datagram is / should be **discarded when 0 is reached**
 - In such a case, original sender is notified
 - Via an **ICMP Time Exceeded** message

TraceRoute Tool

`tracert` (`tracert`)

- **Diagnostic tool** allowing for retrieval of **routing paths**
 - I.e., **sequence of routers** on the way to a given target node
 - Including individual measured transit delay times

Basic principle

- **TTLs are intentionally set to very low values**
 - Starting with 1, then gradually increasing, always by 1
- So that **routers on the way are hence pushed to discarding**
 - Causing such routers to reveal their existence
 - As well as providing their IP addresses in particular

TraceRoute Tool

Overall process

- IP datagrams with **ICMP Echo Request** payloads are iteratively sent in a loop, step by step
 - Each time a higher TTL value is used
- When **ICMP Echo Reply** response is received
 - Whole process ends
 - Since the destination node was already reached
- When **ICMP Time Exceeded** response is received
 - Another router on the way was detected
 - And the whole process continues...
- When **no response is received within a given timeout**
 - Another router was also detected
 - But no information is available

Header Fields: Header Checksum

Header Checksum (16 bits)

- Aims at **ensuring header integrity**
 - I.e., allows for detection of potential changes in header fields
- **Does not involve payload content**
 - Its integrity must be treated by L4 if need be

Checksum calculation

- Header is interpreted as a **sequence of 16-bit words**
- **Ordinary checksum** (not CRC) is calculated
 - Checksum field as such is skipped
 - Potential overflow area is summed as well
 - **One's complement** is in fact used as the final check value
 - I.e., individual bits are inverted

Header Fields: Header Checksum

Verification

- Checksum is calculated over absolutely all header fields
 - I.e., including the checksum field itself
- **When 0 is obtained**, no damage was detected
- Otherwise whole **datagram** can be / is **discarded**
 - In which case the **sender is not notified!**
 - I.e., **no ICMP message is sent**
 - Since even the source address could have been damaged
 - And so there is no guarantee the real sender would be notified

Observation: **checksum must be recalculated...**

- Each time **TTL** is decremented
 - Which is quite often = whenever passing through any router
- As well as whenever **NAT** is applied / **fragmentation** occurs

Header Fields: Protocol

Protocol (8 bits)

- **Allows to distinguish different types of data in the payload**
 - I.e., individual L4 transport protocols (TCP, UDP, ...)
 - Including L4 control protocols (RSVP, ...)
 - As well as **internal L3 control protocols** (ICMP, IGMP, ...)
 - Since they also encapsulate their messages into IP datagrams
- Maintained by **IANA**
 - <https://www.iana.org/assignments/protocol-numbers/>
 - Almost 150 values out of 256 are currently assigned
- Examples
 - **UDP** (17), DCCP (33), SCTP (132), **TCP** (6)
 - **ICMP** (1), IGMP (2), RSVP (46)
 - **IPv6** (41) – encapsulation of IPv6 packets in IPv4 datagrams
 - ...

Header Fields: Options

Options

- **Allow to specify additional optional information**
 - So that standard handling of IP datagrams could be adjusted
 - Not used frequently nowadays, though
- **Arbitrary number** of options can be specified (0 or more)
 - Each may have a different size (both **fixed** or **variable**)
 - Overall size of all options must aligned to multiples of 4 bytes
 - If not, extra **padding** must be added at the end

Generic **internal structure**

- **Option Type** (1 byte)
- **Option Length** (1 byte) – omitted in fixed-length options
- **Option Data** (0 or more bytes) – omitted in simple options

Header Fields: Options

Option types

- Maintained by **IANA**
 - <https://www.iana.org/assignments/ip-parameters/>
 - Altogether ≈ 25 options are currently defined
- Have their **internal structure**, too
 - **Copied Flag** (1 bit)
 - Related to the process of **fragmentation of IP datagrams**
 - Indicates whether an option should be copied into fragments
 - **Option Class** (2 bits)
 - Describes the intended usage (control, debugging, ...)
 - **Option Number** (5 bits)
 - Specifies a particular option type

Header Fields: Options

Option examples

- **End of Option List (EOOL, 0, not copied)**
 - Used for **padding** purposes
- **Time Stamp (TS, 68, not copied)**
 - Allows to record time delays between individual routers
- **Options used by Source Routing at L3**
 - **Record Route (RR, 7, not copied)**
 - Allows to record IP addresses of individual routers on the way
 - Used for probe datagrams during the first phase
 - **Strict Source Route (SSR, 137, copied)**
 - Sequence of routers prescribing the intended datagram routing
 - **Loose Source Route (LSR, 131, copied)**
 - Analogous idea, only additional previously unspecified routers might be visited between the compulsory specified ones

Header Fields

Source Address and **Destination Address** (32 bits each)

- Standard IPv4 sender / recipient addresses

Fragmentation

Motivation: **block transmissions**

- There is always a certain **limitation on acceptable block sizes**
 - Regardless of a particular layer or protocol
- Expressed via **Maximum Transmission Unit (MTU)**
 - Defines **maximal payload size** a protocol is willing to accept
 - And so guaranteeing it is capable to transmit
 - Of course, using the services of the lower layer
- ⇒ **it may happen that MTU of the lower layer is insufficient**
 - In terms of the whole prepared PDU we want to transmit
 - I.e., including our header / footer
 - In such a case, **transmission would need to be rejected**
- **Solution: oversized block is split into smaller fragments**
 - Each of which has size which already is acceptable

Fragmentation

Ultimate **objective**

- Need for **fragmentation should be avoided** whenever possible

Avoidance strategies

- Providing **illusion of a byte stream**
 - So that the higher layer does not need to be aware of anything
 - But, of course, that only moves the problem elsewhere...
 - Example: **TCP**
- Announcing **non-fragmenting MTUs**
 - I.e., **maximal size ensuring no fragmentation will be needed**
 - This recommendation is provided to the higher layer
 - In the expectation that this layer will simply respect it
 - I.e., that it will only create blocks of suitable sizes
 - Examples: **IP** → **TCP** or also **IP** → **UDP** → L7

IPv4 Fragmentation

Observation

- **Fragmentation avoidance is not always achievable**
 - Because the announced MTUs may not be **respected**
 - Or MTUs as such might not have been correctly **resolved**
- And so fragmentation has to inevitably be somehow supported

Deployment at L3 in IPv4

- Fragmentation of IP datagrams is supported
 - And so must be the subsequent **defragmentation...**
- **Range of permitted IP datagram sizes**
 - **Theoretically up to 64 kB**, lower in practice...
 - Since it depends on **MTUs of real-world L2 technologies**
 - E.g.: Ethernet II (1500 B), Ethernet 802.3 with 802.2 LLC and SNAP (1492 B), Wi-Fi (2304 B), ...

MTU Detection

Question: **How non-fragmenting MTU should be resolved?**

- Four strategies are basically possible for a given sender...

(1) **No Restrictions** (kind of optimistic approach)

- **Recommended size of IP datagrams is not limited in any way**
 - And so the maximal theoretical size is preserved
 - I.e., 64 kB minus IP headers
- **Suitable only when nothing better is achievable**
 - Since this approach will most likely always cause fragmentation

(2) **Guaranteed Minimums** (kind of pessimistic approach)

- It is guaranteed that certain **minimal IP datagram sizes** must be possible to transmit without fragmentation
 - Theoretically 68 B, in practice 576 B
 - Including IP headers in both cases, though

MTU Detection

(3) Detection of Local MTU

- **L3 MTU is derived from L2 MTU of a given network interface**
 - I.e., particular technology used by such an interface
- This approach is especially **appropriate for routers**
 - Since their interfaces are likely to use different technologies
 - As well as they should not be expected of anything else than **fulfilling their primary tasks only**
 - I.e., they should focus on **routing and forwarding**
 - Not advanced means of MTU discovery
- Unfortunately, even a single network can be heterogeneous
 - I.e., its **individual segments may use different technologies**
 - E.g., combination of Ethernet and Wi-Fi in not just home LANs
 - And so the interface MTU may not be valid within all segments

MTU Detection

(4) Detection of Path MTU

- Even when a datagram leaves our network unfragmented
 - It may still be subjected to fragmentation later on
 - Since **different networks can use different technologies**
- Therefore the **minimal permitted MTU on the way** could help
 - Such MTU can be detected using **Path MTU Discovery** process
- Unfortunately...
 - **Non-trivial overhead is required**
 - Because the detection process itself is not straightforward
 - **May not always work as expected**
 - Because of the **connectionless** nature of the IP protocol
 - I.e., individual datagrams may be routed differently
 - And so the detected path MTU may not actually be relevant

IPv4 Fragmentation

Fragmentation

- **Process of dividing IP datagrams into smaller fragments**
 - Each of which is then **routed and forwarded independently**
 - Without being reassembled sooner than at the destination
- Fragmentation can be performed by both...
 - End nodes acting as **senders** and **routers** on the way

Defragmentation

- **Process of IP datagram reassembling from its fragments**
 - There must exist a way...
 - How it is recognized that fragments **belong to each other** at all
 - And in which **mutual order** they are supposed to be combined
- Defragmentation can only be performed by...
 - End nodes acting as the final intended **recipients**

Fragmentation Process

Fragmentation principle

- **Datagram payload** is taken and **divided into smaller parts**
 - Each of which must have a suitable size
- **New IP datagram is constructed for each of these parts**
 - Its header is created as a **copy of the original header**
 - Where **certain fields** are then affected accordingly
- In particular...
 - **Fragmentation fields**
 - Generated, modified, or preserved as needed...
 - **Options**
 - Only the first fragment will take over all the original options
 - All the remaining fragments will contain **copied options** only
 - **IHL, Total Length** and **Header Checksum** fields are updated

Header Fields

Identification (16 bits)

- **Unique identification of a given group of fragments**
 - Unique means...
 - Unique value for a given **source and destination pair**
 - Within the **scope** of a node which generated this identifier
 - For the time the datagram will be active in the system
 - Undefined if not yet fragmented
- **Identifier life cycle**
 - **Generated during the very first fragmentation**
 - I.e., when fragmenting a not yet fragmented datagram
 - Preserved untouched in subsequent fragmentations

Header Fields

Fragmentation Flags (3 bits)

- Fixed 0 bit
- **Don't Fragment Flag**
 - **Requirement to prohibit fragmentation even if need be**
 - Possible values
 - 0 = fragmentation is permitted / 1 = prohibited
 - If prohibited but unavoidable nevertheless...
 - Such a datagram will need to be **discarded**
 - Sender is notified via **ICMP Destination Unreachable** message
- **More Fragments Flag**
 - **Flag indicating the very last fragment in a given group**
 - Possible values
 - 0 = the last fragment / 1 = more fragments follow

Header Fields

Fragmentation Offset (13 bits)

- **Expresses offset of the beginning of a given fragment**
 - I.e., its relative position with respect to the original whole
- Expressed in **integral multiples of 8 bytes**
 - And so fragment sizes must also be rounded to such multiples
 - Of course, with the exception of the very last fragment
- Observation
 - **It must be possible to further fragment datagrams that have already been fragmented!**
 - And so labeling of fragments with ordinal numbers instead of offset positions would not work for this purpose

Path MTU Discovery

Path MTU Discovery

- **Process allowing for detection of path MTU**
 - I.e., minimal MTU on a path across all involved networks
- Originally intended for routers
 - Nowadays **used by all modern end node operating systems**

Principle

- **Datagrams are iteratively sent in a loop**, step by step
 - Each time a certain particular **datagram size** is chosen
 - **Starting with the local MTU**
 - And **gradually decreasing** in subsequent iterations
 - **Don't Fragment Flag** is intentionally activated
 - I.e., set to value 1

Path MTU Discovery

Principle (cont'd)

- When **ICMP Destination Unreachable** response is received
 - We continue with **another attempt**
 - Where decreased datagram size will be used
 - The problem is that we were notified...
 - But **we were not provided with any particular suggestion**
 - I.e., **particular MTU that caused the problem**
 - And so we have to guess...
- Whole process ends when the intended destination is reached

Defragmentation Process

Defragmentation principle

- Individual fragments may not be delivered in **correct order**
 - And they actually do not need to be delivered at all
 - Any of them, independently on each other
- **Incoming fragments are therefore put into the buffer**
 - Only when we have all of them...
 - Because we know we received the very last of them
 - As well as there are no gaps in offsets and lengths
 - ... **the original datagram is reassembled**
 - For which the fragments are ordered using their offsets
- **When any of the fragments is not delivered within a timeout**
 - Everything is lost
 - Since such fragments will simply not be delivered again
 - Sender is notified via an **ICMP Time Exceeded** message

Fragmentation Issues

Negative impact of fragmentation

- Whole concept **must be supported** by all involved nodes
 - Which in fact is, but...
- There is always a **non-trivial overhead**
 - Even if fragmentation actually did not occur at all
 - Because fragmentation headers are present nevertheless
- **Everything gets complicated**
 - Especially **defragmentation is complex and time demanding**
 - As well as more difficult to implement
- Impact of **reliability issues** is increased
 - **Loss or damage to any of the fragments** makes the entire original block unusable

Fragmentation Issues

Negative impact of fragmentation (cont'd)

- **Changes stateless behavior to stateful**
 - Since **waiting** is necessary until all fragments are received
 - As well as **timeouts** are introduced to handle non-deliveries
 - This is in conflict with design principles of the entire IP

⇒ **fragmentation should really be avoided whenever possible**

ICMPv4 Protocol

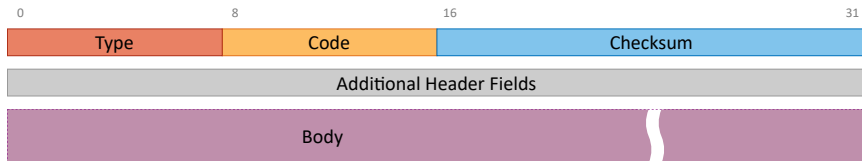
Internet Control Message Protocol (ICMP)

- **Auxiliary L3 protocol providing support to IPv4**
 - Allows to deal with **errors and non-standard situations**
 - Since IP alone is not capable of doing so
- **Types of messages**
 - **Error messages**
 - **Informational messages**: various queries, requests, replies, ...
- **Scope of validity is not limited to just a single network**
 - And so ICMP messages need to be routed across networks
 - ICMP could thus work as yet another full-fledged L3 protocol
 - So that its messages would be inserted directly into L2 frames
 - But that would mean ICMP itself would need to be routable
 - And so instead, **ICMP messages are inserted into IP datagrams**
 - Which kind of (again) contradicts principles of layered models

ICMPv4 Messages

Message structure

- **Header** (64 bits)
 - Always fixed length, though **partially variable header fields**
- **Body**
 - May entirely be missing
 - Fixed or variable length
 - This length does not need to be explicitly remembered
 - Since it is derivable from the length of the entire IP datagram



ICMPv4 Messages

Header fields

- **Type** (8 bits)
 - **Main type** of a given ICMP message
- **Code** (8 bits)
 - Code describing a **particular subtype** of a given message type
- **Checksum** (16 bits)
 - **Checksum of the whole message**, not just its header
 - The same calculation mechanism as in case of IP itself is used
- **Additional header fields** (32 bits)
 - Depend on a particular message type
 - **Often unused** (but always present)

ICMPv4 Messages

Message **types and codes**

- Maintained by **IANA**
 - <https://www.iana.org/assignments/icmp-parameters/>
 - Current state (as of May 2021)
 - Almost ≈ 45 out of 256 types are used or reserved
 - Many of which are deprecated, though

Message **body**

- **Depends on a particular message type**
 - May entirely be missing as already outlined
- Often contains **beginning of the original IP datagram**
 - It means IP datagram which caused a given error message
 - In particular, its **full header and the first 64 bits of its body**
 - So that **source and destination L4 ports** are available, too
 - Which helps in correct recognition of the original datagram

Message Examples

Destination Unreachable (Type 3)

- Error messages sent when datagrams needed to be discarded
 - Since their further processing was not possible
 - Because of various particular reasons...
- **Network Unreachable** (Code 0)
 - Sent by routers on the way when...
 - **Network of the intended destination node is not reachable**
 - I.e., its distance in the routing table equals to infinity
 - Or the destination network is unknown at all
- **Host Unreachable** (Code 1)
 - Sent by inbound routers in the destination network when...
 - **Intended destination node is not reachable**
 - In case such a detection is possible at all

Message Examples

Destination Unreachable (Type 3) (cont'd)

- **Protocol Unreachable** (Code 2)
 - Sent by recipient nodes when...
 - **Designated transport protocol is not supported**
 - More precisely, payload type in **protocol field** is not supported
- **Port Unreachable** (Code 3)
 - Sent by recipient nodes when...
 - **Specified destination L4 port is invalid or not available**
 - I.e., it is not possible to perform datagram demultiplexing
- **Fragmentation Needed** (Code 4)
 - Sent by routers on the way when...
 - **Fragmentation is needed but was explicitly prohibited**
 - I.e., when the **Don't Fragment Flag** was enabled
- ...

Message Examples

Time Exceeded (Type 11)

- **Time to Live Exceeded in Transit** (Code 0)
 - Sent by routers on the way when...
 - **Datagram TTL was exceeded**
 - TTL value dropped to zero
- **Fragment Reassembly Time Exceeded** (Code 1)
 - Sent by recipient nodes when...
 - **Reassembling of a fragmented datagram was not possible**
 - Since not all fragments were received within a given timeout

Message Examples

Echo Request (Type 8, Code 0) and **Echo Reply** (Type 0, Code 0)

- **Allow for testing of reachability of nodes**
 - Sender sends an **Echo Request** message
 - Recipient responds with an **Echo Reply** message
 - At least should respond (it is compulsory)
 - But nowadays often does not because of security reasons
- Additional **header fields** are needed
 - **Allow to match corresponding pairs of requests and replies**
 - Since multiple requests may be sent in a row
 - **Identifier** and **Sequence Number**
 - **Unique identification and serial number given by a sender**
 - Particular implementation varies across individual systems
 - Both fields are preserved in replies, including message data
- Used by **traceroute** or **ping** utilities

Ping Tool

ping (backronym **P**acket **I**nter**N**et **G**roper)

- **Diagnostic tool** allowing for testing of **reachability of nodes**
 - Together with **measured round-trip delivery times**
 - As well as basic **datagram loss statistics**

Basic principle

- **ICMP Echo Request** is sent to the intended destination
 - Actually a whole **series of requests** is sent, one by one
 - In order to make the detection and measurement more precise
- When **ICMP Echo Reply** response is received
 - Intended destination is reachable
- When **no response is received within a timeout**
 - Destination is considered as unreachable
 - Though it might just be unwilling to respond

Message Examples

Source Quench (Type 4, Code 0) (deprecated)

- **Feedback technique for Congestion Control and Flow Control**
 - When **routers or end nodes** reached their **capacity limits**
 - In terms of available buffer sizes, transmission capacity, ...
 - Or even better, when they are just **approaching such limits**
- Principle
 - **Source nodes potentially causing the issues are informed**
 - I.e., respective sender or senders
 - So that **discarding of datagrams** is attempted to be avoided
- Observations
 - It is not possible to specify the **extent of congestion** problems
 - And so it is also not clear how the source nodes should react
 - Nor it is possible to revoke the original announcement

Message Types

Other interesting message types

- **Redirect** (Type 5)
 - **Signals incorrect or not optimal routing**
 - At the level of particular nodes or whole networks
- **Router Advertisement** (Type 9)
 - **Allows routers to reveal their existence** to end nodes
 - Using multicast to 224.0.0.1 (or broadcast when unavailable)
- **Router Solicitation** (Type 10)
 - **Allows end nodes to search for available routers**
 - Using multicast to 224.0.0.2 (or broadcast when unavailable)
- **Parameter Problem** (Type 12)
 - Messages representing **generic so far not covered problems**
- ...

ARP

Address Resolution Protocol (ARP)

- Allows for **translation of IP addresses to hardware addresses**
 - For the purpose of **direct delivery** at L3
 - I.e., delivery within the context of a given network
- In particular...
 - We are about to **locally send an L3 IP datagram**
 - To the final recipient or just the first / next router on the way
 - **Intended recipient is expressed in terms of its L3 IP address**
 - I.e., in both the cases, we are provided with this address
 - Delivery of the datagram is, however, **executed using L2**
 - And L2 is only capable of working with L2 hardware addresses
 - Unfortunately, we only have the intended L3 IP address...
 - And so the corresponding **L2 address needs to be discovered**

ARP

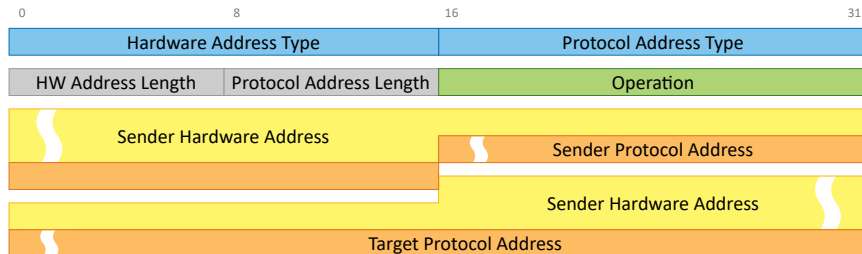
Address Resolution Protocol (cont'd)

- Basic idea
 - **Sender creates an ARP Request message**
 - Includes the queried IP address in this message
 - And **sends it to the whole network using L2 broadcast**
 - **Corresponding target node captures this request**
 - By matching its IP address with the queried one
 - **Creates an ARP Reply message with its hardware address**
 - And **sends it back via an ordinary L2 unicast**
- Observations
 - **Different technologies and addresses are used at L2**
 - As well as **different protocols and addresses are used at L3**
- ⇒ it would be nice to handle them all in a unified way
 - ARP really is capable of such **universal applicability**

ARP Messages

Message structure

- All fields are compulsory, overall length is variable
 - Since individual **address fields have variable lengths**
- **ARP messages are encapsulated to L2 frames**
 - And so ARP as such belongs to the L3 network layer



Message Structure

Hardware Address Type (16 bits) and Length (8 bits)

- Describe the **type and length of L2 addresses**
 - I.e., identify a particular L2 technology
 - As well as length of its addresses in a number of bytes
- Types are maintained by **IANA**
 - <https://www.iana.org/assignments/arp-parameters/>
- Examples
 - **Ethernet** 10 Mb/s (type 1, length 6 bytes)
 - IEEE 802 Networks with **EUI-48** (type 6, length 6 bytes)
 - **EUI-64** (type 27, length 8 bytes)
 - **HDLC** (type 17, length 1 byte or more)
 - ...

Message Structure

Protocol Address Type (16 bits) and **Length** (8 bits)

- Describe the **type and length of L3 addresses**
 - I.e., identify a particular L3 protocol
 - And similarly length of its addresses
- Types are (primarily) maintained by **IEEE RA**
 - **EtherTypes** were recycled for this purpose
 - <http://standards-oui.ieee.org/ethertype/eth.txt>
- Examples
 - **IPv4** (type 0x0800, length 4 bytes)
 - **IPv6** (type 0x86DD, length 16 bytes)

Message Structure

Operation (16 bits)

- Allows to distinguish individual **ARP operations**
- Codes are maintained by **IANA**
 - <https://www.iana.org/assignments/arp-parameters/>
 - **Request (1), Reply (2), ...**

Sender Hardware Address and Sender Protocol Address

- L2 and L3 addresses of the sender
 - I.e., node sending a given request or reply

Target Hardware Address and Target Protocol Address

- L2 and L3 addresses of the indented recipient

Resolution Process

ARP **Request** message (operation code 1)

- **Type and length fields** are set according to the situation
- **Address fields** are filled in as follows...
 - Sender hardware and protocol addresses
 - Both are set according to the sender
 - **Target hardware address** is left undefined
 - **Target protocol address** is set to the **queried IP address**

ARP **Reply** message (operation code 2)

- Reply can be constructed directly from the received request
 - **Operation code** is changed from request to reply
 - **Source and target addresses are mutually swapped**
 - **Source HW address** is then set to the **resolved HW address**

Resolution Process

Resolution process has **significant overhead**

- Not just because **broadcast is required**
- It would also be inefficient to repeat requests over and over
 - And so **discovered mappings are cached**

ARP Cache

- **Table** with resolved **IP and hardware address bindings**
- **Static records** (e.g.: 192.168.1.255 → FF-FF-FF-FF-FF-FF)
- **Dynamic records**
 - Must be **periodically forgotten**
 - So that changes within the network can be reflected
 - Timeout can be 1 minute for end nodes, hours for routers
 - As well as **refreshed** to restrict new unnecessary queries
 - With the aim of optimizing the whole process even more

Resolution Process

Resolution steps

- **ARP Cache table is first consulted**
 - When the required **binding already exists**
 - It is simply fetched and the whole process ends
- **Otherwise an ARP Request message is constructed**
 - And sent to the whole network using L2 broadcast
- **Each and every node captures the request message**
 - And exploits the received information to update its cache
 - I.e., **adds a new binding or refreshes an already existing one**
- **Intended target node (if any) furthermore...**
 - Creates an **ARP Reply message**
 - And sends it to the original node using L2 unicast

Reverse ARP

Reverse Address Resolution Protocol (RARP)

- Allows for **translation of hardware addresses to IP addresses**
 - In terms of **assignment of an IP address to a given node**

Assignment process

- **RARP Request message** is constructed (**operation** code 3)
 - Message format remains the same as in the traditional ARP
 - Both **hardware address fields** are set to the known HW address
 - Both protocol address fields are left unused
 - **Request is then sent to the whole network via L2 broadcast**
- **RARP Server** captures this request
 - I.e., special host configured to serve such requests
 - **RARP Reply message** is constructed (**operation** code 4)
 - Reply is then sent back to the original node via L2 unicast

Reverse ARP

Drawbacks

- Very old and simple solution
- **RARP operates at L3**
 - Encapsulates its messages directly into L2 frames
 - And so **RARP server must be available in each network**
 - Since RARP messages cannot cross network boundaries
- Whole approach **cannot work without L2 broadcast**
- **Only fixed manually defined bindings are supported**
 - This is not sufficient from today's perspective
- **Additional information cannot be passed at all**
 - I.e., only IP address itself can be assigned
 - And not other (nowadays) essential information
 - E.g., netmask / CIDR prefix, router IP address, ...

DHCP

Dynamic Host Configuration Protocol (DHCP)

- Newer solution dealing with the identified drawbacks
 - Based on **BOOTP (Bootstrap Protocol)**
 - Allow for IP address assignment, but its primary motivation was related to providing boot images to diskless workstations
- Advantages
 - One DHCP server can serve **multiple networks**
 - **Dynamic assignments** of addresses is possible
 - **Operates at L7**, uses UDP datagrams at ports 67 and 68
 - Allows for interchange of **additional information**
 - Netmask, routers, DNS servers, time zone, time servers, ...
- Three **allocation strategies** are provided
 - **Manual**, **Automatic** and **Dynamic**

Allocation Strategies

Manual Allocation (also Static Allocation)

- Requesting client acquires a **predefined IP address**
 - Based on fixed **HW address** → **IP address bindings**
 - Created manually by the **network administrator** in advance
- As a consequence, **allocated address is always the same**
 - Which is suitable for network printers or similar devices

Automatic Allocation

- Requesting client acquires an **arbitrary IP address**
 - Chosen by the DHCP server itself from a given **address pool**
- Allocation is understood as **permanent**
 - It means the **binding is remembered the first time it is created**
 - So that the next time **the same address** can be granted again

Allocation Strategies

Dynamic Allocation

- Requesting client acquires an **arbitrary IP address**
 - Again chosen by the DHCP server from a given **address pool**
 - Of course, only currently unused addresses can be considered
- However, allocation is **temporary** only in this case...
 - Based on a **concept of lease**
 - I.e., only for a **limited period of time**
 - Which is specified at the moment of the allocation
- As a consequence...
 - **Different address may be provided each time!**
 - As well as one address can gradually be used by different nodes
- This has a **fundamental impact on IP address management**
 - Though it may not be apparent at first sight...

Lease Concept

Traditional approach

- Once assigned, **nodes have their addresses permanently**
 - In a sense of being their **owners or holders**
 - And so they can use them as long as they want to
 - I.e., for any length of time without any limitation
- This approach is **simple** on one hand
- But, unfortunately, **not flexible enough** on the other
 - Since **end nodes often roam from one network to another**
 - And so the whole traditional concept is no longer suitable for contemporary networks

Lease Concept

Newly introduced **concept of lease**

- Nodes act as **DHCP clients** in a sense of temporary lessees
 - **They must proactively take care of their IP addresses**
 - They are expected to perform **various tasks**
 - And so make transitions through **various states**

Period of lease

- Appropriate length depends on a particular situation
 - **Shorter periods**
 - **Higher flexibility, lower stability, higher overhead**
 - **Longer periods...** on the contrary
- In practice...
 - Hours, days, weeks, months, ...

Client Actions

Allocation

- Client does not yet have an IP address and asks for a lease

Reallocation

- Client does have an IP address and asks for its confirmation
 - I.e., lease of the current address is still valid
 - And so there is actually no reason for such a request
- However, it is **voluntarily willing to accept a new address**
 - Since change at this moment would not cause any obstacles
 - E.g., because this node...
 - Has **just rebooted** or was **turned off for some time**

Client Actions

Renewal

- **Client has an IP address but its lease is approaching its end**
 - And so **extension of lease period is requested**
 - If granted, a new lease with the same address is in fact started
- First renewal attempt is initiated in 50% of lease time

Rebinding

- **Client asks a different server for the currently leased address**
 - Suitable when the original server became unavailable
 - I.e., when standard renewal could not be finished successfully
- First rebinding attempt is initiated in 87.5% of lease time

Release

- **Client returns its IP address before its lease expired**

IPv6 Protocol

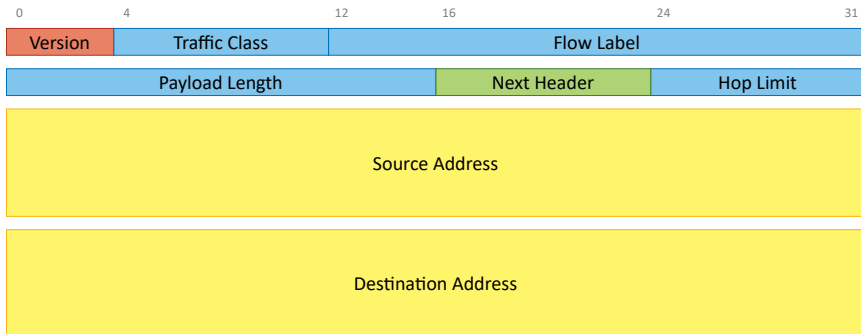
Internet Protocol version 6 (IPv6)

- **Differences** with respect to IPv4
 - **Larger IPv6 addresses**
 - 128 bits instead of just 32 bits
 - Together with 3 levels of routing (site / subnet / interface)
 - **Simpler packet format**
 - **Lower number of header fields**
 - Meaning and / or names of certain fields were changed
 - Some were removed entirely, e.g., **header checksum**
 - **Concept of extension headers**
 - Instead of IPv4 options
 - **Different approach to fragmentation**
 - **Integrated QoS support**
 - ...

IPv6 Packets

Packet structure

- **Header chain**
 - **Main header** (40 bytes) and optional **extension headers**
- Optional **body**



Main Header Fields

Version (4 bits)

- Fixed value 6

Payload Length (16 bits)

- Overall size of **payload and extension headers** (if any)
 - Main header is not included

Hop Limit (8 bits)

- Analogy to IPv4 **Time to Live** field

Traffic Class (8 bits)

- Analogy to IPv4 **Type of Service** field
 - I.e., used for the purpose of **Differentiated Services**

Main Header Fields

Flow Label (20 bits)

- Allows to identify a particular **flow** = **group of related packets**
 - With the aim of treating them all in a similar way
 - E.g., with respect to QoS or other purposes
- In fact, **(Source Address, Flow Label)** forms the full identifier
 - Which allows to recognize such flows even at L3
- In IPv4, **transport connection** identification would be needed
 - I.e., tuple (sender $IP_1:port_1$, protocol, recipient $IP_2:port_2$)
 - Which is less convenient when compared to IPv6
 - Since L4 fields would need to be accessed in the payload

Extension Headers

Packet structure

- Whole packet is composed from a **chain** of...
 - **Compulsory main header**
 - **Arbitrary number of extension headers** (0 or more)
 - Each should **only be used at most once** (exception exists)
 - They should be used in a specific **recommended order**
 - So that processing of IPv6 packets by routers is simplified
 - **Optional body**
- Enumerated blocks are put into the packet one after another
 - **Each header contains the Next Header field**
 - Which allows to mutually chain the individual blocks
 - I.e., describe what the next block is supposed to contain

Extension Headers

Next Header (8 bits)

- Determines the **type of the next block** in a chain
 - I.e., type of the next **extension header** or **body payload**
 - Assuming that body (if any) must be placed at the very end
- Types (**Protocol Numbers**) are maintained by **IANA**
 - <https://www.iana.org/assignments/protocol-numbers/>
 - For simplicity, codes correspond to IPv4 analogies
- **Examples**
 - Extension headers: **IPv6 Fragmentation** (44), ...
 - Payload protocols: **UDP** (17), **TCP** (6), **ICMPv6** (58), ...
 - Special type: **IPv6 No Next Header** (59)
 - Suggests that nothing follows
 - And even if something does follow, it must be ignored

IPv6 Fragmentation

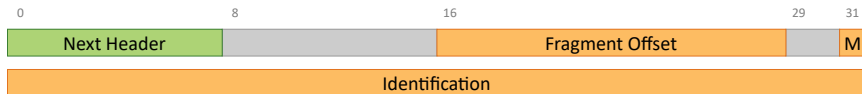
Fragmentation in IPv6

- Differences with respect to IPv4
 - **Only source nodes can perform fragmentation** (never routers)
 - So that they can focus on their primary objective
 - Excessive packets are then automatically discarded
 - All information is stored within the **Fragmentation Header**
 - I.e., it is only used when fragmentation really took place
 - **Guaranteed minimal non-fragmenting MTU** is 1280 B
 - Compared to just 68 B / 576 B in case of IPv4
 - **IPv6 Path MTU Discovery**
 - Basically the same idea as in IPv4, though differences exist...
 - **ICMPv6 Packet Too Big** message is received instead
 - Includes value of the particular **MTU that caused the problem**

IPv6 Fragmentation

Fragmentation header (type 44)

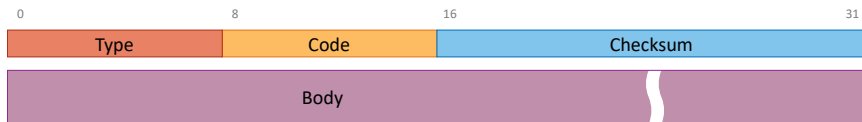
- **Next Header** (8 bits)
- **Fragment Offset** (13 bits)
 - **Relative to the end of the last non-fragmented header**
 - I.e., certain extension headers are fragmented, others not
 - Fragmentation header is then put in between these two groups
- **More Fragments Flag** (1 bit)
- **Identification** (32 bits)
 - The same principle as in IPv4, only larger range



ICMPv6

Internet Control Message Protocol version 6 (ICMPv6) (type 58)

- **Analogy to ICMPv4 for IPv4**
 - Basic principles are the same, though differences exist...
- **Longer part of the original IP packet** is preserved in body
 - As much as can be included not to exceed packet size 1280 B
 - So that fragmentation is avoided
- **Checksum calculation** also involves ICMP **pseudo-header**
 - With IPv6 source / destination addresses and other fields
- Slightly different generic **message structure**
- Different **types of particular ICMP messages**



Lecture Conclusion

IPv4 datagrams

- Header fields
 - Time to Live
 - Header Checksum
 - Protocol
 - ...

IPv4 fragmentation

- Basic principles
- Avoidance strategies
- MTU detection approaches
 - Path MTU Discovery
- Issues

Lecture Conclusion

ICMPv4

- Destination Unreachable, Time Exceeded, ...

ARP and RARP

- Translation of IP addresses to hardware addresses or vice versa

DHCP

- Manual, automatic, and dynamic **allocation strategies**
- Concept of **lease**, client actions

IPv6

- Main header, extension headers, body
- Fragmentation

ICMPv6