NSWI090: **Computer Networks**

http://www.ksi.mff.cuni.cz/~svoboda/courses/202-NSWI090/

Lecture 11

TCP/IP Protocol Suite I

Martin Svoboda
svoboda@ksi.mff.cuni.cz

17. 5. 2021

Charles University, Faculty of Mathematics and Physics

Lecture Outline

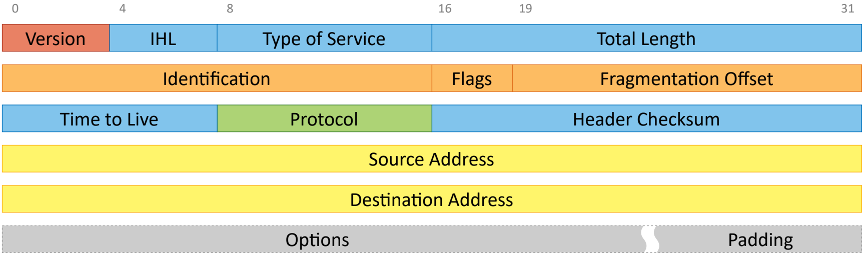
IPv4 protocol

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

IPv4 Datagrams

Datagram structure

- **Header**
  - **Required fields** as well as **optional fields** ⇒ **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** (**Differentiated Services**)

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 ⇒ 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 ⇒ **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

`traceroute` (`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 $\approx$ 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
- $\Rightarrow$ **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** $\rightarrow$ **TCP** or also **IP** $\rightarrow$ **UDP** $\rightarrow$ $\mathrm{L7}$

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 then at the destination
- Fragmentation can be performed by both…
  - End nodes acting as **senders** and **<u>routers</u>** 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 <u>payload</u>** 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 **<u>copied</u> 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

$\Rightarrow$ **fragmentation should really be avoided whenever possible**

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