Introduction to Computer Networks and the Internet COSC 264

IP and Related Protocols

Slides prepared by [1]Dr. Andreas Willig
Updated and presented by [2]Dr. Barry Wu

[1]Department of Computer Science & Software Engineering
[2]Wireless Research Centre
College of Engineering

Semester 2, 2021

Outline

About This Module

- Goals of this Module:
  - Get a first idea of the Internet
  - Get to know the IP protocol and important support protocols
- Useful references:
  - The “bible” on TCP/IP: [9] (old, but still great!)
  - Other references: [4], [8, Part V]
  - Internet protocols are published as **requests-for-comment** (RFC) by the Internet Engineering Task Force (IETF), you can access them via: `http://www.ietf.org/rfc.html`
- Most of these slides are based on [9]

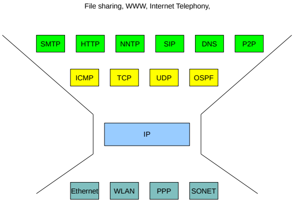
The Internet

- The Internet is a packet-switched network
- It is a **network of networks**:
  - It consists of many different networks, connected by routers
  - The networks or links can be of any technology:
    - Ethernet
    - Optical point-to-point links
    - Wireless LAN
    - ...
    - Carrier pigeons (RFC 1149)
- It is really large:
  - The Internet Systems Consortium estimates ≈ 1.033 billion stations (called hosts) as of July 2015
  - See `https://www.isc.org/network/survey`
- It has a fairly complex topology [1]

The Internet (2)

- The end-to-end principle [7]:
  - Perform intelligent functions in hosts, not in routers
  - For example:
    - Routers know how to deliver packets
    - All functions making this delivery **reliable** are performed in the end host, e.g. by the TCP protocol
    - There is no network-layer mechanism for reliable delivery
    - **Keep the routers simple!**
- Internet is standardized by the IETF, standards are called RFCs
  - IETF = Internet Engineering Task Force (`www.ietf.org`)
  - RFC = Request For Comment
- For the design philosophy see [3]

The Hourglass Model for the Internet Protocol Stack



- "Everything over IP, IP over everything"

Outline

Introduction

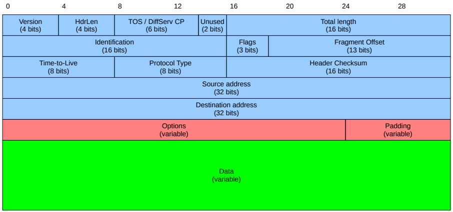
- IP is specified in RFC 791 and many followup RFCs
- It is the network layer protocol of the Internet
- Some terminology:
  - IP packets are called **datagrams**, except if they result from fragmentation and reassembly, then they are called **fragments**
  - End stations are called **hosts**
  - IP routers are called **routers**
- IP addresses are assigned to **network interfaces**:
  - When a host has three Ethernet adapters, it has three IP addresses, one for each adapter
  - Since most hosts have only one adapter, we may speak of the IP address of that host

IP Service – Best Effort

- Basic IP service is **datagram delivery**
- This service is:
  - Connectionless: no connection or shared state is set up before datagram delivery starts
  - Unacknowledged: IP does not use acknowledgements
  - Unreliable: on IP level no retransmissions are carried out
  - Unordered: IP does not guarantee in-sequence delivery [2]
- This kind of guarantee-nothing service is called **best effort**

Outline

Packet Format



Packet Format (2)

- Where applicable (e.g. addresses, total length), header is using **big endian** byte ordering (also called **network byte order**)
- The `Version` field specifies the version of the IP protocol running, always 4 for IPv4
- The `HdrLen` field:
  - specifies the length of IP header as number of 32-bit words
  - If the `Options` field does not use a multiple of 32 bits, a `Padding` field is used to fill up to 32 bits
  - When `HdrLen` > 5, then an `Options` field is present
- The `TOS/DSCP` field:
  - TOS = Type Of Service, DSCP = DiffServ Code Point
  - Allows to mark packets for differentiated treatment to achieve Quality-Of-Service (QoS), e.g. express priorities
  - DiffServ [5] is framework for Internet QoS, another is IntServ [10]
  - Many routers ignore the `TOS/DSCP` field

Packet Format (3)

- The `TotalLength` field:
  - Gives the total length of datagram in bytes (i.e. up to 65,535)
  - Can be modified during **fragmentation and reassembly**
  - The `TotalLength` field is part of IP header, since some technologies (Ethernet!) pad up frames to achieve minimum frame size and do not reverse or mark this
- The `Identification` field:
  - Uniquely identifies each IP payload unit accepted from higher layers for a given interface
  - Incremented by source host for each new IP payload
  - Particularly important in the fragmentation+reassembly mechanism
  - In other words: it is a sequence number
  - Routers do not touch this field
- The `Flags` field:
  - Contains two flags relevant for fragmentation and reassembly (`DF`, Don't Fragment, and `MF`, More Fragments)

Packet Format (4)

- The `FragmentOffset` field:
  - Is used for fragmentation and reassembly
  - Gives the offset of the current fragment within entire datagram, in multiples of eight bytes
- The `HeaderChecksum` field:
  - Is calculated over IP header only, not the data (TCP, UDP etc. have their own checksums to cover their data)
- The `Time-To-Live` field:
  - Gives upper limit to number of routers a packet can traverse
  - Decremented by each router, forces re-computation of checksum
  - When `TTL=0` and packet cannot be directly delivered to destination, datagram is discarded, sender is notified (ICMP message)
  - Typical initial values: 32 or 64
  - Intended usage: eliminate packets caught in a routing loop

Packet Format (5)

`Protocol` value	Encapsulated Protocol
0x01	ICMP
0x02	IGMP
0x04	IP-in-IP Encapsulation
0x06	TCP
0x11	UDP

- `Protocol` field indicates the higher-layer protocol that generated the payload
- This field provides **protocol multiplexing**
- In other words: it provides different SAPs
- Some values shown in table

Packet Format (6)

- The `SourceAddress/DestinationAddress` fields:
  - `SrcAddr` indicates the initial sender of datagram
  - `DstAddr` indicates intended final receiver of datagram
  - Are of 32 bits width
- The `Options` field:
  - Contains header field for optional IP features
  - One example option: source routing
  - Options are rarely used, we will not consider this anymore

Outline

1. IPv4
   - Packet Format
   - IP Addressing
   - IP Forwarding and Routing
   - Fragmentation and Reassembly
2. IP Helper Protocols

IP Address Representation

- IP addresses have a width of 32 bits
- They are supposed to be worldwide unique
  - This is not really true anymore with NAT ...
- IP addresses are written in **dotted-decimal notation**, e.g.:

  `130.149.49.77`

  where decimal (!) numbers are separated by dots
- They have an internal structure:

  `<network-id> <host-id>`

  where:
  - `<network-id>` denotes a network (e.g. an Ethernet)
  - `<host-id>` refers to a host **within** this network
- The `<host-id>` must only be unique w.r.t. its network

Interlude: Routing / Forwarding Tables

- IP routers have several network interfaces or ports (different from TCP/UDP port numbers) where they receive/transmit datagrams
- In IP networks a router getting a packet on some input port looks at the `DestinationAddress` field to determine the output port
- The router consults a **forwarding table**:
  - The forwarding table lists all networks the router knows with their `<network-id>` and the output port to send the packet to in order to reach that network
  - The router performs a **table lookup** for an incoming packet, it searches the forwarding table for a matching network entry
  - Time required for table lookup depends on number of table entries
  - How this table is filled is determined by a separate routing protocol
- This is simplified, more details later!

Important Points

Important Point

A host address is tied to its location in the network, i.e. it is coupled to network topology. When a host switches to another network, it obtains another address and ongoing connections (TCP!) break – IP therefore has no direct support for mobility!!

Important Point

IP Routing is mostly concerned with networks, i.e. forwarding tables in routers mostly store `<network-id>`'s – it is the responsibility of last router on a path to deliver an IP datagram to directly connected host.

Classless Inter-Domain Routing

- Question: how many bits to allocate to `<network-id>`?
- In the early days, this number was fixed to three different values: 8, 16 and 24 (classful addressing)
- This proved inflexible, something better was needed
- CIDR = Classless Inter-Domain Routing
- Introduced 1993, specified in RFCs 1518, 1519, mandatory
- Modern routing protocols (OSPF, RIPv2, BGP) use CIDR
- In CIDR a network is specified by two values:
  - A 32 bit network address
  - A 32 bit network mask (**netmask**)

CIDR – Netmask

- For a given 32-bit IP address the netmask specifies which bits belong to network-id and which bits belong to host-id
- The netmask consists of 32 bits, the leftmost k bits are ones, the remaining $32 - k$ bits are zeros
- Examples:

Netmask	Shorthand
`11111111.11110000.00000000.00000000`	/12
`11111111.11111111.00000000.00000000`	/16
`11111111.11111111.11100000.00000000`	/19
`11111111.11111111.11111110.00000000`	/23

- To fully specify network, give network address and netmask, e.g.:

  `192.168.40.0/21`

- The rightmost $32 - k$ bits of network address `a.b.c.d/k` are zero

CIDR – Netmask (2)

- Example: given host address `192.168.40.3` and netmask /24, the hosts network address is computed as:

```
     11000000.10101000.00101000.00000011   192.168.40.3
AND  11111111.11111111.11111111.00000000   /24
     -----------------------------------------------
     11000000.10101000.00101000.00000000   192.168.40.0
```

- The same example, now with netmask /21:

```
     11000000.10101000.00101000.00000011   192.168.40.3
AND  11111111.11111111.11111000.00000000   /21
     -----------------------------------------------
     11000000.10101000.00101000.00000000   192.168.40.0
```

- In both examples the network addresses are the same, but the networks are of different size
- To distinguish both networks you need to specify both network address and bitmask, network address alone is insufficient

CIDR – Netmask (3)

- In a network `a.b.c.d/k` there are two "special host addresses":
  - The host address `000..00` (with $32 - k$ zeros in total) is part of the network id, signifying that we refer to the network as a whole
  - The host address `111..11` (with $32 - k$ ones in total) is the broadcast address of this network

  All the other host addresses can be assigned to individual hosts
- Example: In the network `192.168.40.64/28` there are 14 addresses available:
  - The netmask leaves four bits for the host-id, i.e. 16 values
  - The value `0000` is part of the network-id
  - The value `1111` is the broadcast address for this network

Reserved IP address blocks

Address Block	Current Usage
10.0.0.0/8	Private-use IP networks
127.0.0.0/8	Host loopback network
169.254.0.0/16	Link-local for point-to-point links (e.g. dialup)
172.16.0.0/12	Private-use IP networks
192.168.0.0/16	Private-use IP networks

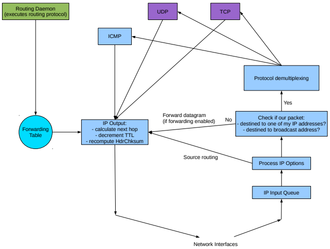
(from: [6], there are more than shown here)

- Private-use IP addresses are often used for broadband clients or by NAT boxes
- The "traditional" loopback address of a host is `127.0.0.1`, but any address from `127.0.0.0/8` network serves same purpose
- Packets with private addresses are not routed in the public internet, only within the provider network

Outline

1. IPv4
   - Packet Format
   - IP Addressing
   - IP Forwarding and Routing
   - Fragmentation and Reassembly
2. IP Helper Protocols

Simplified Packet Processing



Simplified Packet Processing (2)

- Packet processing chain is followed in routers and hosts
- Incoming packets are checked for correctness and stored in IP input queue – correctness includes:
  - right value in IP version field
  - correct IP header checksum
- Next, packet options are processed (rarely used)
- Next, it is checked if packet is destined to **this** host / router or to the broadcast address of any network this host / router is directly attached to
- If so, protocol demultiplexing is carried out
  - The `Protocol` field in IP header is checked for its value
  - Packet payload is delivered to the software entity implementing the indicated higher-layer protocol
  - Packet is not processed any further!

Simplified Packet Processing (3)

- If packet is not destined to this host/router or broadcast address:
  - If packet forwarding is not enabled, the packet is dropped
  - Otherwise:
    - Check if packet is destined to a directly reachable station (e.g. on a directly attached Ethernet) – if so, deliver packet directly
    - If packet is not destined to directly reachable station, consult forwarding table to determine next hop / outgoing interface
    - Decrement TTL value, drop packet when it reaches zero
    - Recompute packet header checksum (why?)
    - Hand packet over to outgoing interface
- Forwarding table is maintained by a **routing daemon**, i.e. a process executing a routing protocol
- Note that datagrams to be routed can come from local applications or from other hosts via IP input queue
- Linux commands to inspect / modify forwarding table:
  - `netstat`
  - `route`

Forwarding Table Contents (First Approximation)

- Each entry in the forwarding table contains:
  - Destination IP address, which can be either:
    - a full host address (i.e. non-zero host-id)
    - a network address, with netmask

    depending on the value of a flag
  - Information about next hop, either:
    - IP address of next-hop router (must be directly reachable)
    - IP address of directly-connected network (network address/netmask)
  - Flags:
    - A flag telling whether destination IP is host or network
    - A flag telling whether next hop is a router or directly attached network
  - Specification of outgoing interface

Forwarding (First Approximation)

- From forwarding table structure it is clear that a host / router does not know the full path, but only next hop
- Forwarding table lookup for a packet with destination IP address `dst` proceeds in three stages (Caveat: reality is different):
  - First look for an entry that is a full-host address matching `dst` – if found, send packet to indicated next hop / outgoing interface and stop processing
    - This is not used very often
  - Next look for an entry that is a network address matching `dst` – if found, send packet to indicated next hop / outgoing interface and stop processing
  - Finally look for special **default** entry – if found, send packet to indicated next hop (the **default router**) and stop processing
  - Otherwise drop packet, possibly send ICMP message back to original sender of datagram

Forwarding – Address Matching

- **Question**: how to check whether a destination address `dst` matches a forwarding table entry for network `a.b.c.d/k`?

Forwarding – Address Matching (2)

- **Answer**: They match when

$$(\mathtt{dst}\ \text{AND} < /\mathtt{k} - \mathtt{netmask} >) \quad == \quad (\mathtt{a.b.c.d}\ \text{AND} < /\mathtt{k} - \mathtt{netmask} >)$$

- **Example**: We are given the following forwarding table:

Destination Network/Netmask	Outgoing interface
`130.1.0.0 / 16`	`eth0`
`141.5.6.0 / 24`	`eth1`

- **Question**: We are given two packets with destination addresses `130.1.9.5`, and `166.42.17.12`, respectively. Which decisions does the router make?
- **Question**: And what happens if a default route is added to the forwarding table?

Forwarding Tables in Hosts

- Most end hosts leverage the **default route** mechanism:
  - An end host can differentiate between packets to local destinations and to all other destinations
    - **Question**: suppose an end host has address `130.149.49.77` and is part of a `/24` network – how does it check whether a destination address `a.b.c.d` belongs to another host in the same network?
  - Packets to local destinations are delivered directly (see discussion of ARP for how to do this in an Ethernet)
  - Packets to all other destinations are sent to default router
- Therefore, forwarding tables in end hosts can be made out of very few entries:
  - One entry for each network it is directly attached to (local networks)
  - The default route
- The default route must be configured (typically done by DHCP)

Forwarding Tables in Routers

- Most routers at the "border" of the Internet only have forwarding table entries for a subset of all networks attached to the Internet (likely other networks belonging to the same owner), for all other networks they rely on default routers
- Some routers in the core:
  - do not have a default router
  - are the default routers of other routers
  - must know (almost) all the Internet networks

Outline

On the Choice of Packet Size

- The link-layer technologies underlying IP offer many different maximally allowed packet sizes, e.g.:
  - Ethernet: 1500 bytes
  - Gigabit Ethernet: 9000 bytes
  - IEEE 802.11 WLAN: 2312 bytes
  - ISDN: 576 bytes
- This max size also known as **maximum transmission unit** (MTU)
- Higher-layer protocols (TCP, UDP) and applications should not be required to know these maximal sizes:
  - One reason: “software hygiene”, separation of concerns
  - Another reason: it is not well defined:
    - Different packets of the same flow can take different routes
    - A packet can use different technologies while in transit
    - Even if all packets use the same route, this route can change due to link failures / restores

Fragmentation and Reassembly

- IP **hides** this from upper layers, offers own maximum message length of 65,515 bytes to higher layers
  - 65,515 = 65,535 - 20, 20 bytes is minimum size of IP header
- To cope with smaller MTUs:
  - Sender IP instance partitions message into **fragments**
  - Each fragment is transmitted individually as a full IP packet, with header information specifying that this is a fragment and giving the position of fragment in whole message
  - Each fragment has a size no larger than MTU of outgoing link
  - IP instance at destination buffers received fragments, re-assembles message and delivers it to higher layers

Question

Would it be useful to have intermediate IP routers perform reassembly?

Fragmentation and Reassembly (2)

- In addition, every intermediate router can:
  - fragment a full message
  - further fragment a fragment

  when necessary for transmission on next hop
- When the destination receives the first fragment, it:
  - Allocates buffer large enough for whole message
  - Starts a timer
- When all fragments arrive before timer expiration:
  - Timer is canceled
  - Re-assembled packet is handed over to higher layers
  - Buffer is de-allocated
- When timer expires before all fragments have arrived:
  - The already received fragments are dropped, buffer is freed
  - ICMP message (type 11, code 1) is sent to source host

Some Details

- Every message handed over from higher layers has own identifier
  - See `Identification` field in IP header
- All fragment datagrams belonging to same message have:
  - A full IP header
  - The same value in the `Identification` field
  - A `TotalLength` field reflecting the fragment size
  - Different values for `FragmentOffset` field (reflecting the start of the present fragment within the whole message):
    - `FragmentOffset` specifies offset in multiples of 8 bytes
  - The `MF` (more-fragments) bit set, except for the last fragment, which has non-zero `FragmentOffset`

Question

With this setup: how much buffer space shall the receiver allocate when it gets the first fragment?

Some Details: The DF bit

- By setting the DF (don’t fragment) bit in the IP header a source node **forbids** fragmentation by intermediate routers
- When a router receives a datagram with DF set, it:
  - Checks whether outgoing link for this packet has an MTU large enough to transmit the packet
  - If so, the packet is transmitted onto next hop
  - If not, the router drops the datagram and returns an ICMP message to original IP source
    - ICMP with type 2 (“destination unreachable”) and code 4 (“fragmentation required, but DF set”)

Some Details: The DF bit (2)

Question

How could you use this for the sender to determine the **path MTU**, defined as the smallest MTU of all links along a path between source and destination?

Fragmentation and Reassembly – Discussion

- Fragmentation/Reassembly creates significant overhead:
  - Several datagrams per message, each having full IP header
  - Reassembly adds significant complexity to receiver
  - Upon loss of single fragment the whole message is possibly re-transmitted by higher layers (TCP!)
- The designers of IPv6 got rid of fragmentation+reassembly, end points need to perform path MTU discovery

Outline

Outline

1. IPv4
2. IP Helper Protocols
   - ARP
   - ICMP

Address Resolution Protocol – ARP

- IP addresses only have a meaning to IP and higher layers
- In an Ethernet, stations have own 48-bit MAC addresses
- Recall that IP datagrams are encapsulated into Ethernet frames
- An Ethernet station picks up a packet only if the destination MAC address matches its own MAC address (ignoring broadcast / multicast), IP addresses and other packet contents are ignored
- An IP address is assigned to an Ethernet adapter

Important Question

How do other stations know to which MAC address a given IP address refers?

Address Resolution Protocol – ARP (2)

- ARP determines MAC address for given IP address
- ARP is specified in RFC 826
- ARP is not restricted to Ethernet, but in general is geared towards LANs with broadcast capabilities
- ARP is **dynamic**:
  - The MAC address for a given IP address does not need to be statically configured, ARP allows to determine this on-the-fly
  - Advantage: nodes can be moved or equipped with new network adapters without any re-configuration
  - Disadvantage: a separate protocol is needed, bringing additional complexity and requiring a little bandwidth
- There is also a protocol that lets stations find an IP address for a given MAC address, this is called RARP (Reverse ARP)

Basic Operation of ARP

- Suppose that:
  - We have two stations *A* and *B* attached to the same Ethernet, having the following addresses:

	Station *A*	Station *B*
MAC	`11:11:11:11:11:11`	`22:22:22:22:22:22`
IP	`130.149.49.11`	`130.149.49.22`

  - Both *A* and *B* are in the same IP network `130.149.49.00/24`, which is an Ethernet network
  - Station *A* wishes to send an IP packet to address `130.149.49.22` and does not yet have any information about the corresponding MAC address
- Each station maintains an **ARP Cache**, which stores the mappings from IP to MAC addresses that the station currently knows about

Basic Operation of ARP (2)

- Station *A* **broadcasts** an **ARP-request** message (displayed in `wireshark` as `arp who-has`), indicating:
  - *A*'s own IP and MAC address
  - *B*'s IP address

  Broadcasting means: packet is sent to **Ethernet broadcast address**, Ethernet frame has value `0x0806` in the length/type field
- Any host *C* having an IP address other than `130.149.49.22` simply drops the ARP-request packet
- Upon receiving the ARP request, host *B* (with IP address `130.149.49.22`) performs the following actions:
  - It stores a binding between between *A*'s IP and MAC address in its own ARP cache
  - It responds with an **ARP-reply packet** that includes:
    - *B*'s MAC and IP address
    - *A*'s MAC and IP address

    ARP reply is unicast to *A*'s MAC addr. (Why no broadcast?)

Basic Operation of ARP (3)

- Upon receiving ARP response from *B*, station *A* stores a binding between *B*'s IP and MAC address in its ARP cache
- This procedure is called **address resolution**
- ARP makes no retransmissions when ARP request not answered
- If a station wants to send an IP packet to a local destination with address `a.b.c.d`:
  - It first checks the ARP cache whether a binding for `a.b.c.d` exists
  - If so, the packet is encapsulated in an Ethernet frame and directed to the MAC address found in the ARP cache entry for `a.b.c.d`
  - Otherwise, the address resolution procedure is started and the packet is sent when the result is available

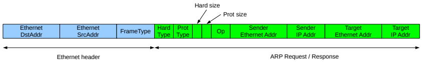
The ARP Cache

- The entries in an ARP cache are soft-state, entries are typically removed 20 minutes after their creation
  - Why?
  - To implement this, for each cache entry a timer is started
  - Some implementations restart timer after referencing a cache entry
- Under Linux you can inspect your ARP cache with the command:

  ```
  /usr/sbin/arp -a
  ```

  The path to the `arp` command can vary between systems

The ARP Frame Format



(See [9, Sect. 4])

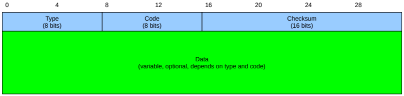
- `HardType` determines the type of MAC addresses used, `0x0001` for Ethernet 48-bit addresses
- `ProtType` determines the higher-layer protocol for which address resolution needs to be done, value `0x0800` for IP
- `HardSize` and `ProtSize` specify the size (in bytes) of the hardware and and protocol addresses – they are 6 and 4 for Ethernet and IP
- `Op` distinguishes between ARP-request and ARP-reply, and some other types (RARP is covered as well)
- The remaining four fields are the mentioned address fields

Outline

Introduction

- ICMP = Internet Control Message Protocol
- Specified in RFC 792
- This protocol:
  - Accompanies the IP protocol by allowing routers or destination hosts to inform sender about “unusual” situations, including:
    - There is no route to the destination
    - Destination host is not reachable
    - Fragmentation required but `DF` set
  - Operates “on top” of IP, i.e. ICMP messages are encapsulated into regular IP datagrams
  - Does not add additional mechanisms (like error control) to IP
  - Does not force any host/router to generate ICMP messages
  - IP sending host **must not rely** on ICMP messages
- These days ICMP messages are often filtered out by firewalls

Message Format



- `type` and `code` specify actual ICMP message type and sub-type
- `checksum` covers ICMP header and data, with `checksum` assumed as zero
- `Data` depends on `type/code` combination, but often includes the first few bytes of the offending IP datagram (including its header)

Some `type/code` Combinations

`type`	`code`	**Meaning**
0	0	Echo reply
3	0	Destination network unreachable
3	1	Destination host unreachable
3	2	Destination protocol unreachable
3	3	Destination port unreachable
3	4	Fragmentation required, but `DF` bit set
3	6	Destination network unknown
3	7	Destination host unknown
4	0	Source quench (Congestion control)
8	0	Echo request
11	0	TTL expired in transit
11	1	Fragment reassembly time exceeded

- There are many more, e.g. for router advertisements, information about malformed IP packets, etc.
- It is implementation-dependent, which ICMP messages are generated
- ICMP messages are often suppressed by firewalls, otherwise too much information about internal network structures could be revealed

Some type/code Combinations (2)

- Source-quench (`type`=4, `code`=0):
  - Generated by IP router when it drops a packet due to congestion
  - Intention is to let source host throttle its rate
- TTL expiration (`type`=11, `code`=0):
  - Generated by IP router when it drops a packet because its TTL value reached zero
  - **Question**: this is used by `traceroute`. How?
- Fragment reassembly timeout (`type`=11, `code`=1):
  - Generated by destination when not all fragments of a message have been received within timeout
  - Used to invite higher-layer protocol at sending host to re-transmit the message (with all the fragments)
  - IP itself does not perform any retransmission!

Some type/code Combinations (3)

- The "destination-unreachable" messages (`type`=3):
  - `code`=0 (destination network unreachable) and `code`=1 (destination host unreachable): generated when:
    - `code`=0: Router doesn't matching entry (and no default entry) for non-directly connected destination address in its forwarding table
    - `code`=1: Router could not deliver datagram to directly connected host (e.g. no ARP response)
  - `code`=2 (protocol unreachable): IP datagram refers to non-existent higher-layer protocol in destination (cf. IP `Protocol` field)
  - `code`=3 (port unreachable): Used with TCP / UDP when no socket is bound to a port number
  - In these messages first 32 bits of the variable ICMP message part are 0, following bytes contain IP header and first few bytes of offending IP datagram

[1] David Alderson, Lun Li, Walter Willinger, and John C. Doyle.
Understanding Internet Topology: Principles, Models and Validation.
IEEE/ACM Transactions on Networking, 13(6):1205–1218, December 2005.

[2] Jon C. R. Bennett, Craig Partridge, and Nicholas Shectman.
Packet Reordering is Not Pathological Network Behaviour.
IEEE/ACM Transactions on Networking, 7(6):789–798, December 1999.

[3] David D. Clark.
The design philosophy of the darpa internet protocols.
ACM Computer Communication Review, 18(4):106–114, August 1988.

[4] Douglas E. Comer.
Internetworking with TCP/IP – Principles, Protocols and Architecture, volume 1.
Prentice Hall, Englewood Cliffs, New Jersey, third edition, 1995.

[5] Kalevi Kilkki.
Differentiated Services for the Internet.
Macmillan Technical Publishing, Indianapolis, 1999.

[6] Deepankar Medhi and Karthikeyan Ramasamy.
Network Routing – Algorithms, Protocols, and Architectures.
Morgan Kaufmann, San Francicso, California, 2007.

[7] Jerome H. Saltzer, David P. Reed, and David D. Clark.
End-to-end arguments in system design.
ACM Transactions on Computer Systems, 2(4):277–288, November 1984.

[8] William Stallings.
Data and Computer Communications.
Prentice Hall, Englewood Cliffs, New Jersey, fourth edition, 2006.

[9] W. Richard Stevens.
TCP/IP Illustrated Volume 1 - The Protocols.
Addison-Wesley, Boston, Massachusetts, 1995.

[10] Paul P. White and Jon Crowcroft.
The integrated services in the internet: State of the art.
Proceedings of the IEEE, 85(12):1934–1946, December 1997.