Lesson 4.2: **Network Layer**

CSC450 – COMPUTER NETWORKS | WINTER 2019-20

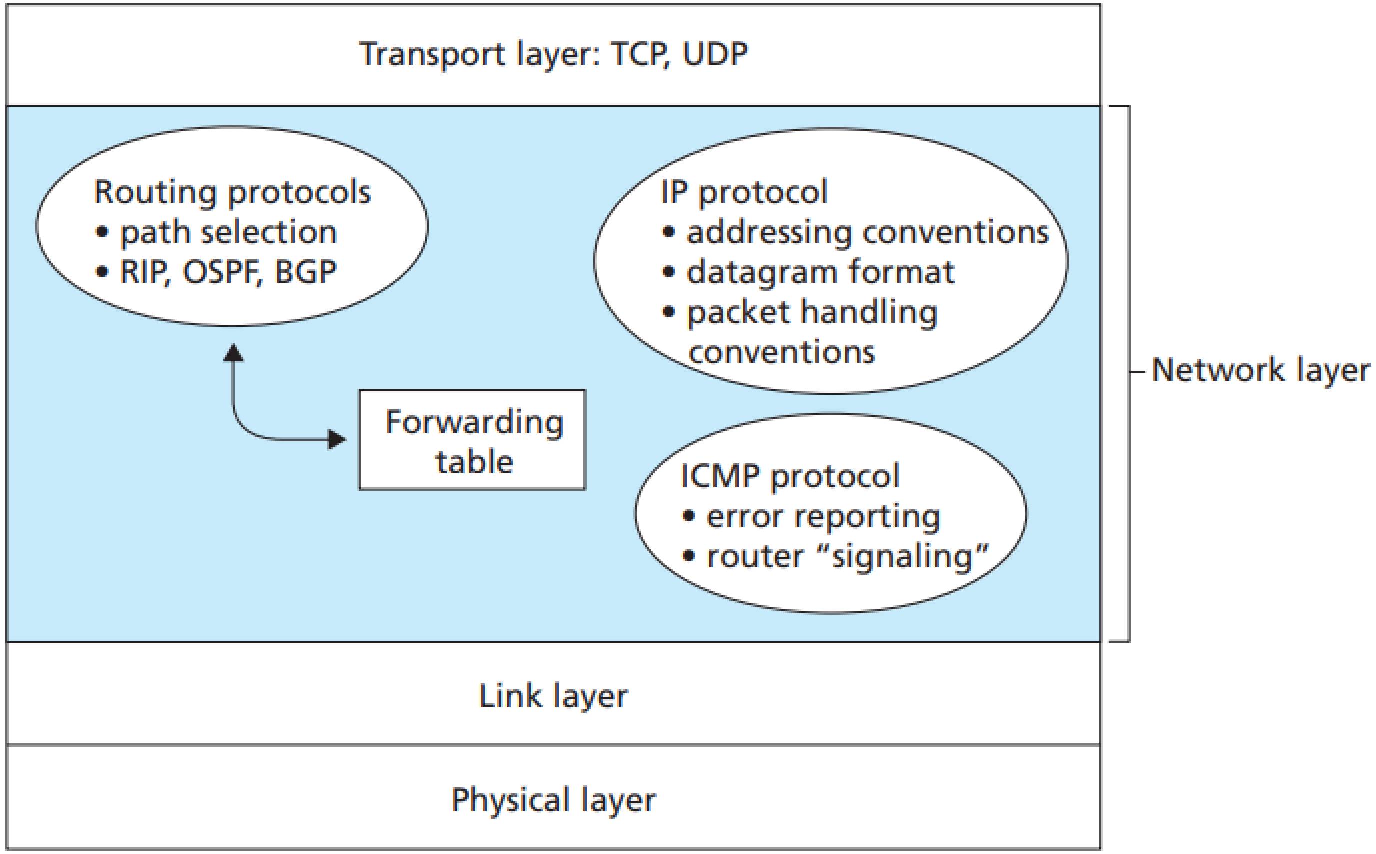
DR. ANDREY TIMOFEYEV

OUTLINE

- Internet Protocol (IP).
  - Introduction.
  - IPv4 datagram format.
  - Datagram fragmentation.
  - IPv4 addressing (revisited).
  - Dynamic Host Configuration Protocol (DHCP).
  - Network Address Translation (NAT).
  - Internet Control Message Protocol (ICMP).
  - IPv6 datagram format.
  - Transitioning between IPv4 & IPv6.
- Generalized forwarding.
  - Software-defined networking.
  - OpenFlow protocol.

INTRODUCTION

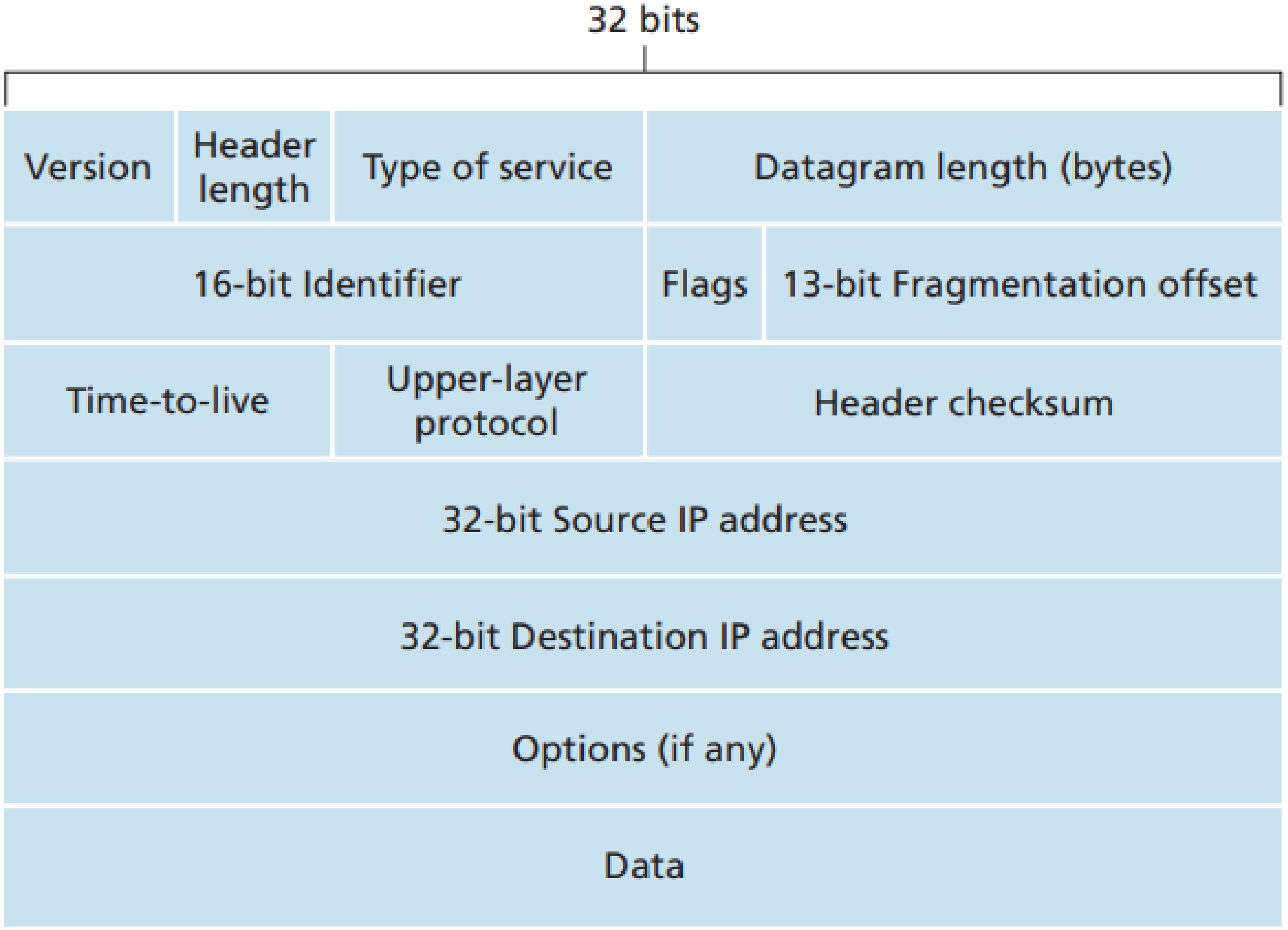
- **Components** of **Network Layer**:
  - **IP** protocol.
  - **Routing** protocols.
  - **Internet Control Message Protocol** (ICMP).



Network Layer components

IPV4 DATAGRAM FORMAT (1)

- **Datagram** – network-layer **packet**.
- Key **fields** of **IPv4 datagram**:
  - Version number.
  - Header length.
  - Type of service.
  - Datagram length.
  - Identifier; flags; fragmentation offset.
  - Time-to-live (TTL).
  - Upper-layer protocol.
  - Header checksum.
  - Source & destination IP addresses.
  - Options.
  - Data (payload).



IPv4 datagram format

IPV4 DATAGRAM FORMAT (2)

- **Description** of **IPv4 datagram** key **fields**:
  - **Version number**.
    - 4 bits specify IP version of datagram (IPv4 / IPv6).
  - **Header length**.
    - 4 bits to indicate the beginning of payload.
  - **Type of service**.
    - 8 bits to specify the type of service datagram provides.
  - **Datagram length**.
    - Total length of datagram = header + data.
  - **Identifier**; **flags**; **fragmentation offset**.
    - 32 bits (in total) responsible for IP fragmentation.
  - **Time-to-live** (TTL).
    - Ensures datagram do not circulate forever.
  - **Upper-layer protocol**.
    - Indicates specific transport-layer protocol: TCP (6) or UDP (17).
  - **Header checksum**.
    - Detects bit errors (only header bytes are summed).
  - **Source** & **destination IP addresses**.
  - **Options**.
    - Allows header to be extended.
  - **Data** (**payload**).
    - Transport layer segment to be delivered to destination.

DATAGRAM FRAGMENTATION (1)

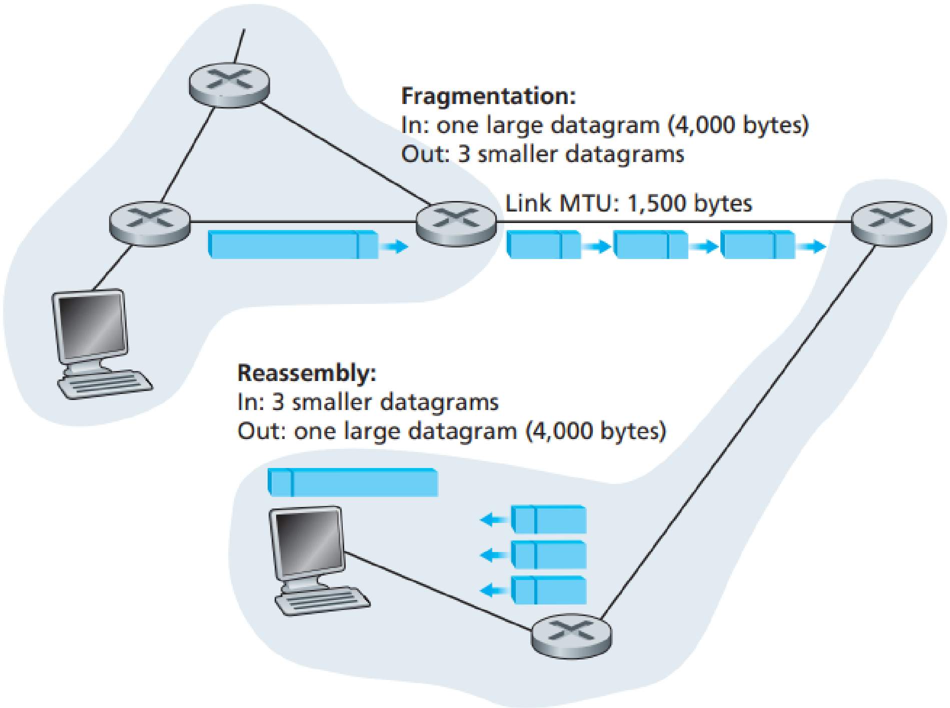
- **Link-layer protocols** are constrained by different **maximum transmission unit** (**MTU**) sizes.
  - **MTU** - size of the **largest** possible link-level frame.
- **Fragmentation** – process of **dividing** (“*fragmenting*”) **large IP datagrams** that are to be forwarded through **smaller MTU links**.
  - **One datagram** becomes **several datagrams**.
  - **Reassembled** only at **final** destination.
  - **Identifier**, **flags**, and **fragmentation offset** header fields are used for **reassembly**.

DATAGRAM FRAGMENTATION (2)

- **Fragmentation** / **reassembly** process:
  - <u>**Sender**</u>:
    - **Stamps** datagrams with **identifier**, **source**, and **destination addresses**.
    - If datagram is **fragmented**:
      - Same **identifier** used for each fragment.
      - Last fragment is specified by **flag field = 0**.
        - All other fragments have **flag = 1**.
      - **Offset field** is set to specify where the fragment **fits** in datagram.
        - Helps to identify **missing** fragments.
  - <u>**Receiver**</u>:
    - Checks **identifier** & **source address fields** to determine fragments of the **same datagram**.
    - Checks **flag field** to determine if the **last fragment** received.
    - Checks **offset field** to determine if **all** fragments are **received** and how to assemble them.

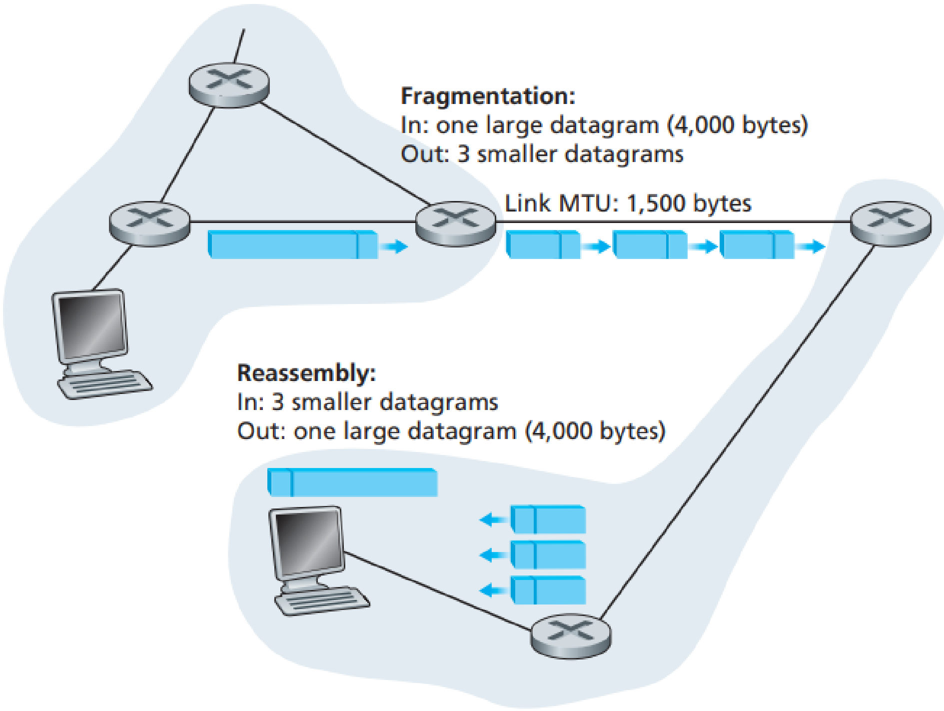
DATAGRAM FRAGMENTATION (3)

- **Example**:
  - Datagram of *4000* bytes (*20* header / *3980* data) forwarded to link with MTU = *1500* bytes.
  - *3980* data bytes allocated to three separate fragments.
  - Identification = *777*. Payload data must be multiple of *8* bytes.



DATAGRAM FRAGMENTATION (4)

- **Example**:
  - Datagram of *4000* bytes (*20* header / *3980* data) forwarded to link with MTU = *1500* bytes.
  - *3980* data bytes allocated to three separate fragments.
  - Identification = *777*. Payload data must be multiple of *8* bytes.



Fragment	Bytes	ID	Offset	Flag
1st fragment	1,480 bytes in the data field of the IP datagram	identification = 777	offset = 0 (meaning the data should be inserted beginning at byte 0)	flag = 1 (meaning there is more)
2nd fragment	1,480 bytes of data	identification = 777	offset = 185 (meaning the data should be inserted beginning at byte 1,480. Note that 185 · 8 = 1,480)	flag = 1 (meaning there is more)
3rd fragment	1,020 bytes (= 3,980–1,480–1,480) of data	identification = 777	offset = 370 (meaning the data should be inserted beginning at byte 2,960. Note that 370 · 8 = 2,960)	flag = 0 (meaning this is the last fragment)

IP fragments

IP fragmentation and reassembly

Fragmentation animation

IPV4 ADDRESSING (1)

- Review of **Classless Inter-Domain Routing** (**CIDR**) **addressing**:
  - **IP address** 172.217.9.142/25
    - **Address in binary form:**
      - *172.217.9.142 → 10101100.11011001.00001001.10001110*
    - **Subnet index:**
      - */25*
    - **Subnet mask:**
      - *11111111.11111111.11111111.10000000 → 255.255.255.128*
    - **Network prefix:**
      - *10101100.11011001.00001001.10001110* (IP address)

AND *11111111.11111111.11111111.10000000* (Subnet mask)

10101100.11011001.00001001.10000000 → ***172.217.9.128*** network prefix

    - **Host identifier:**
      - *0001110 → 14*

Subnets animation

IPV4 ADDRESSING (2)

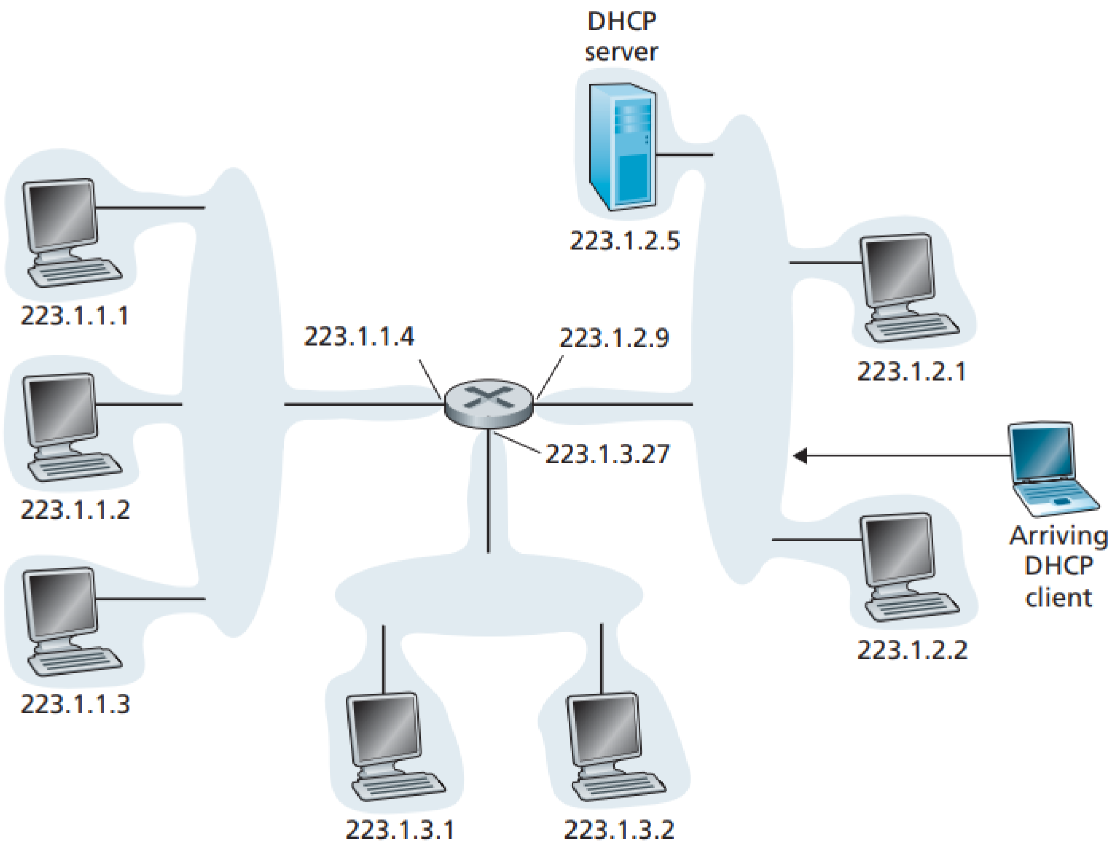
- **ISPs** give out a **block of addresses** to the **organizations**.
  - ISP normally has a **larger block** of addresses that it **divides** into subnets for organizations.

ISP's block	200.23.16.0/20	11001000 00010111 00010000 00000000
Organization 0	200.23.16.0/23	**11001000 00010111 0001000**0 00000000
Organization 1	200.23.18.0/23	**11001000 00010111 0001001**0 00000000
Organization 2	200.23.20.0/23	**11001000 00010111 0001010**0 00000000
…	…	…
Organization 7	200.23.30.0/23	**11001000 00010111 0001111**0 00000000

- Once **organization** obtained a **block of addresses**, individual **addresses** can be **assigned** to **hosts** and **routers**.
  - **Dynamic Host Configuration Protocol** (**DHCP**) is responsible for host **address assignment**.

DHCP (1)

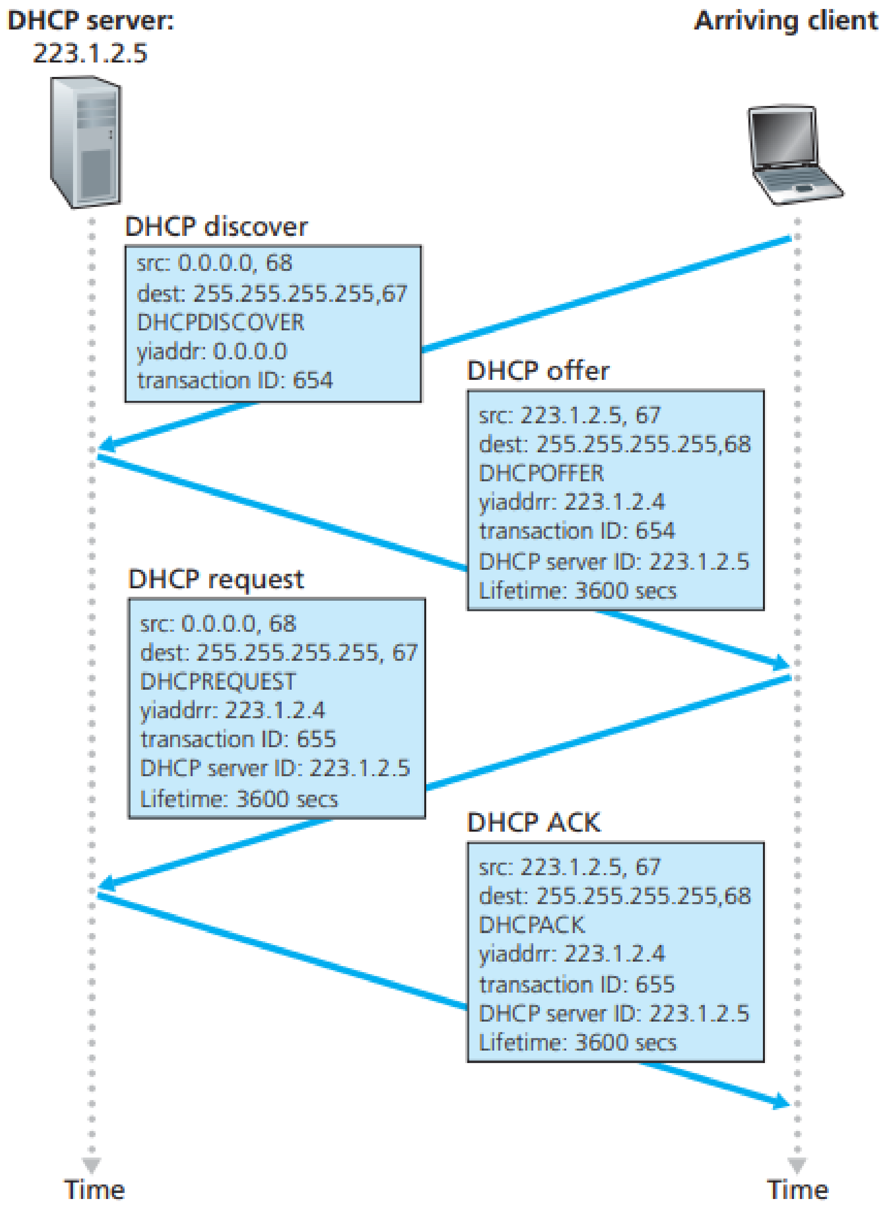
- **Dynamic Host Configuration Protocol (DHCP)** allows a host to **dynamically** obtain an **IP address** from **network server** when it **joins** the network.
  - In addition: **subnet mask**, **address** of **default gateway** (first hop), and **address** of **local DNS server**.
- **DHCP** is a **client-server** protocol.
  - **Client** – newly arriving host, needing to obtain an IP address.
  - **Server** – designated DHCP host in each subnet.
    - No DHCP server in the subnet – **relay agent** (router) that **stores** the **address** of DHCP server for the **network**.



DHCP client-server scenario

DHCP (2)

- **DHCP** protocol follows **four-step process**:
  - DHCP **server discovery**.
  - DHCP **server offer(s)**.
  - DHCP **request**.
  - DHCP **acknowledgment**.

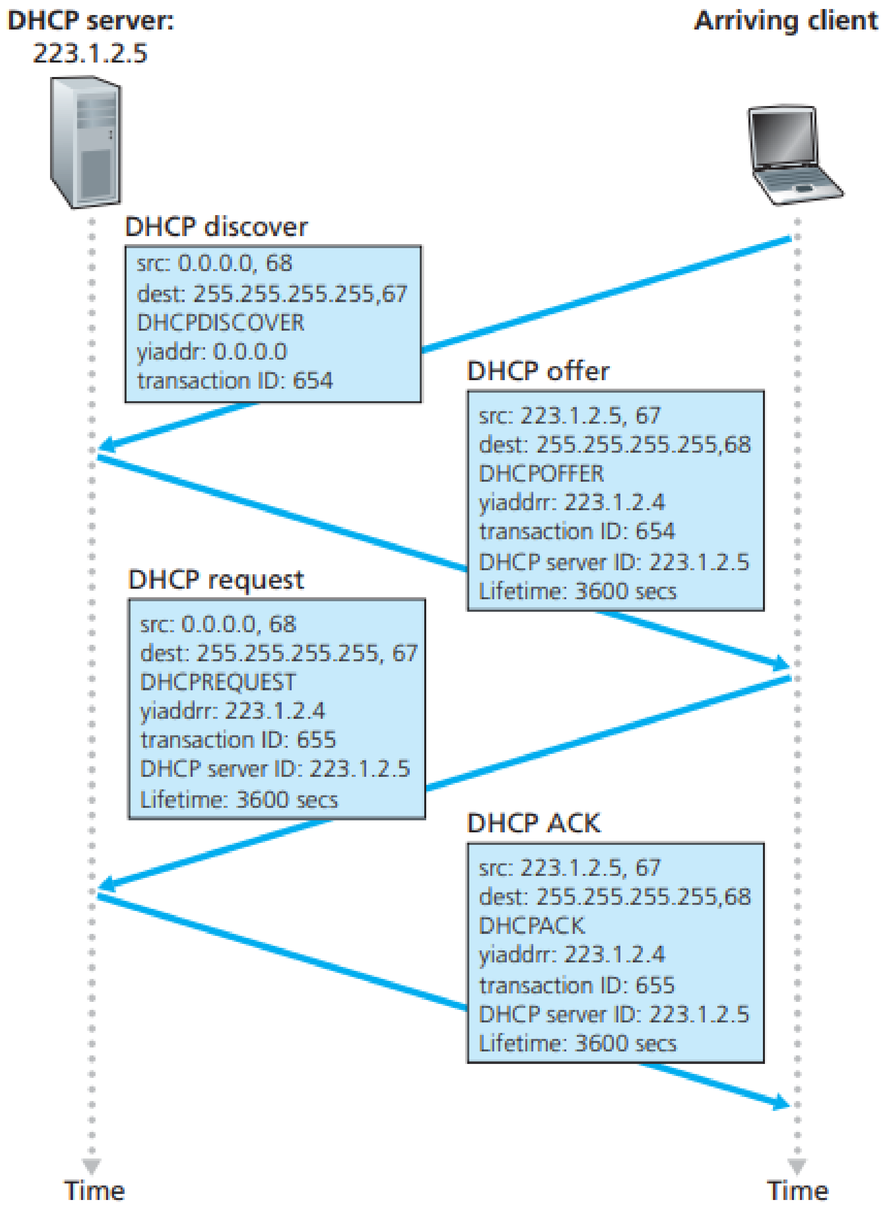


DHCP animation

DHCP client-server interaction

DHCP (3)

- **DHCP** protocol follows **four-step process**:
  - DHCP **server discovery**.
    - Client sends a DHCP discover message.
    - UDP segment to port 67.
    - IP datagram to IP address 255.255.255.255 (broadcast address).
  - DHCP **server offer(s)**.
    - Server responds with DHCP offer message.
    - IP datagram to IP address 255.255.255.255.
    - Contains transaction ID, proposed IP address, network mask, address lease time.
  - DHCP **request**.
    - Client sends DHCP request message.
    - Choses IP address and echoes back configuration parameters.
  - DHCP **acknowledgement**.
    - Server responds with DHCP ACK message.
    - Confirms requested parameters.



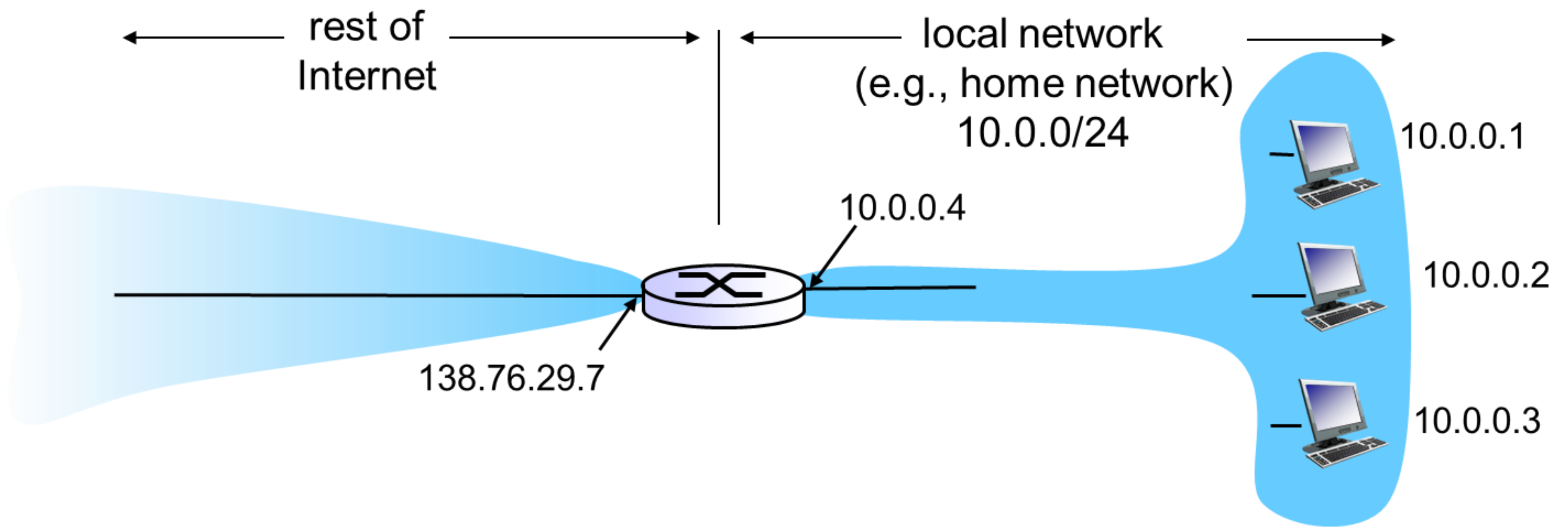
DHCP client-server interaction

NAT (1)

- **Network address translation** (**NAT**) – process of **mapping multiple private** hosts to a **single publicly** exposed **IP address**.
- **Motivation**:
  - **Local network** uses just **one IP address** as far as **outside** world is concerned.
  - Hosts inside a network (**realm**) use **private IP address** (not globally **unique**).
- **Benefits**:
  - **Range of addresses** not needed from ISP.
    - **One public IP address** for all devices.
  - **Change** devices' **addresses** in local network without **notifying outside** world.
  - **Devices** inside local network not **explicitly addressable** (*visible*) by outside world.

NAT (2)

- **NAT process** is governed by **NAT-enabled** router.
  - **Datagrams** with **destination addresses inside** network are forwarded **locally** using **private** addresses.
  - **Datagrams leaving local network** have same single **NAT IP address** and different **source port numbers**.



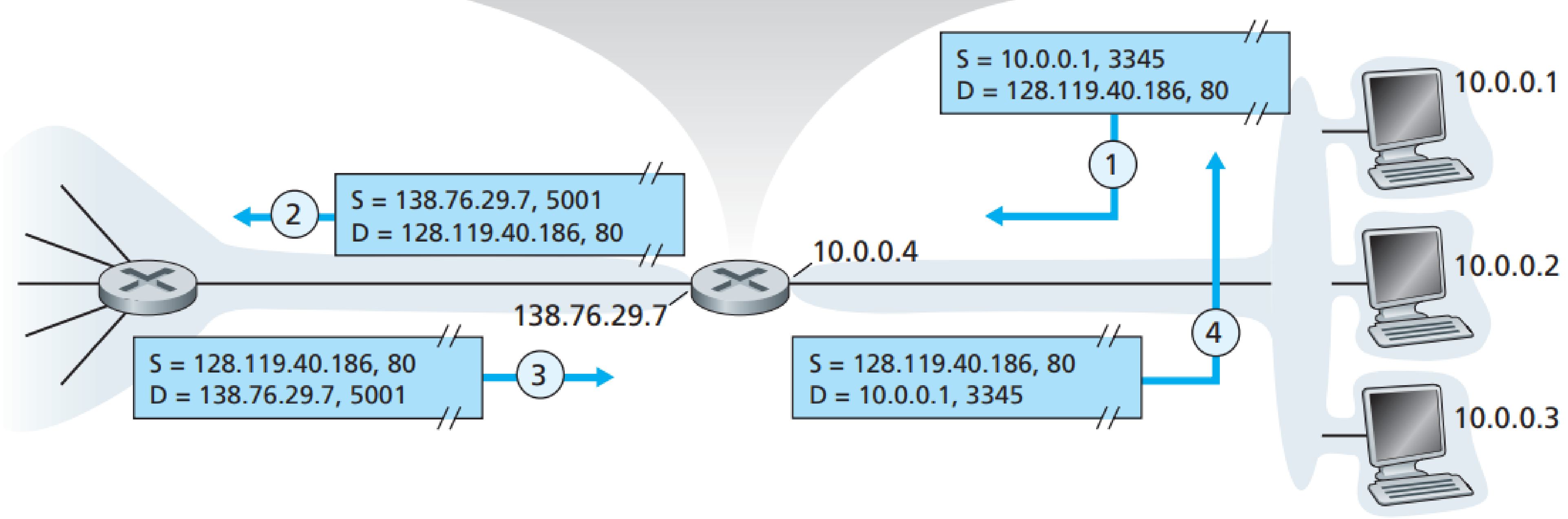
Network address translation

NAT (3)

- **NAT-enabled routers** map **private IP addresses** and **port numbers** to **public IP address** and **port numbers** using **NAT translation table**.

NAT translation table

WAN side	LAN side
138.76.29.7, 5001	10.0.0.1, 3345
. . .	. . .



NAT (4)

- **NAT controversies**:
  - **Port numbers** should be used to address **processes** rather than hosts.
  - **Routers** should not be handling **port numbers** (*why?*).
  - **Address shortage** should be solved with **IPv6 addressing**.

INTERNET CONTROL MESSAGE PROTOCOL (ICMP)

- **Internet Control Message Protocol** (ICMP) is used by **hosts** and **routers** to communicate **network** level **information**.
- **ICMP** is a **network-layer protocol** above the IP protocol.
  - **ICMP messages** are carried inside the IP datagrams **payload**.
  - Indicated in the "**upper-layer protocol**" IP header field.
- **ICMP message contains**:
  - ICMP **type** field;
  - **Code** field;
  - **First 8 bytes** of IP datagram causing **error**.

ICMP Type	Code	Description
0	0	echo reply (to ping)
3	0	destination network unreachable
3	1	destination host unreachable
3	2	destination protocol unreachable
3	3	destination port unreachable
3	6	destination network unknown
3	7	destination host unknown
4	0	source quench (congestion control)
8	0	echo request
9	0	router advertisement
10	0	router discovery
11	0	TTL expired
12	0	IP header bad

ICMP message types

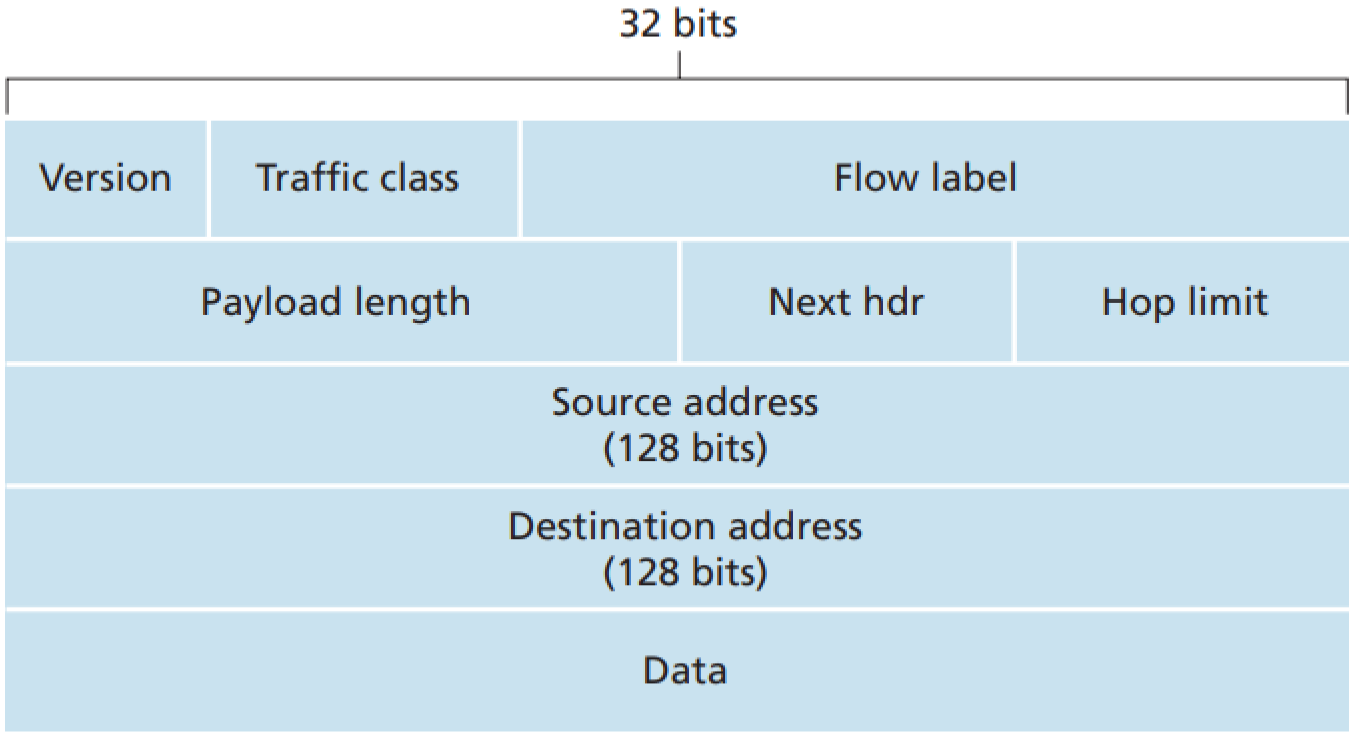
IPV6: MOTIVATION

- **Motivation** behind **IPv6 protocol** development:
  - **32-bit address** space will be **completely allocated**.
  - <u>**In addition:**</u>
    - **Improved header** format helps **speed up processing** & **forwarding**
    - **Additional header changes** to facilitate traffic **quality of service**.
- **Major IPv6** datagram format **changes**:
  - **Expanded addressing capabilities**:
    - 32-bit $\rightarrow$ 128-bit.
    - 2^{128} ~ 340,282,366,920,938,000,000,000,000,000,000,000,000
      - *Three hundred forty undecillion, two hundred eighty two decillion, three hundred sixty six nonillion, nine hundred twenty octillion, nine hundred thirty eight septillion.*
  - **Fixed-length 40-byte** header.
  - **No fragmentation / no checksum**.

[IPv6 adoption statistics]()

IPV6 DATAGRAM FORMAT (1)

- **Key fields** of **IPv6 datagram**:
  - Version number.
  - Traffic class.
  - Flow label.
  - Payload length.
  - Next header.
  - Hop limit.
  - Source & destination address.
  - Data (payload).



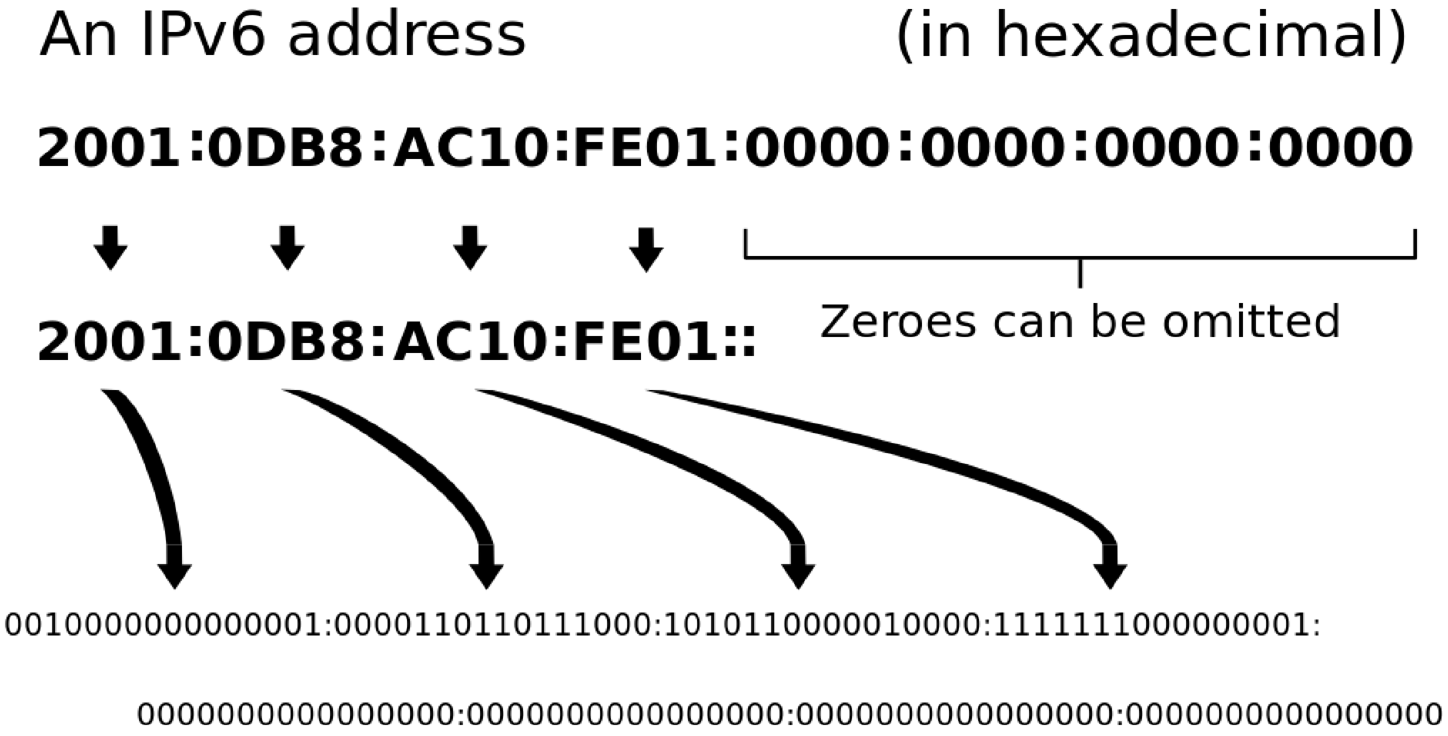
IPv6 datagram format

IPV6 DATAGRAM FORMAT (2)

- **Description** of **IPv6 datagram** key **fields**:
  - **Version number**.
    - 4 bits specify IP version of datagram (IPv4 / IPv6).
  - **Traffic class.**
    - Identifies priority of datagrams in flow.
  - **Flow label**.
    - Identifies datagrams of the same flow.
  - **Payload length**.
    - Number of bytes following fixed-length 40-byte header.
  - **Next header**.
    - Identifies upper level protocol.
  - **Hop limit.**
    - Ensures datagrams do not circulate forever. Number is decremented by each router. When == 0 $\rightarrow$ drop datagram.

IPV6: ADDRESSING

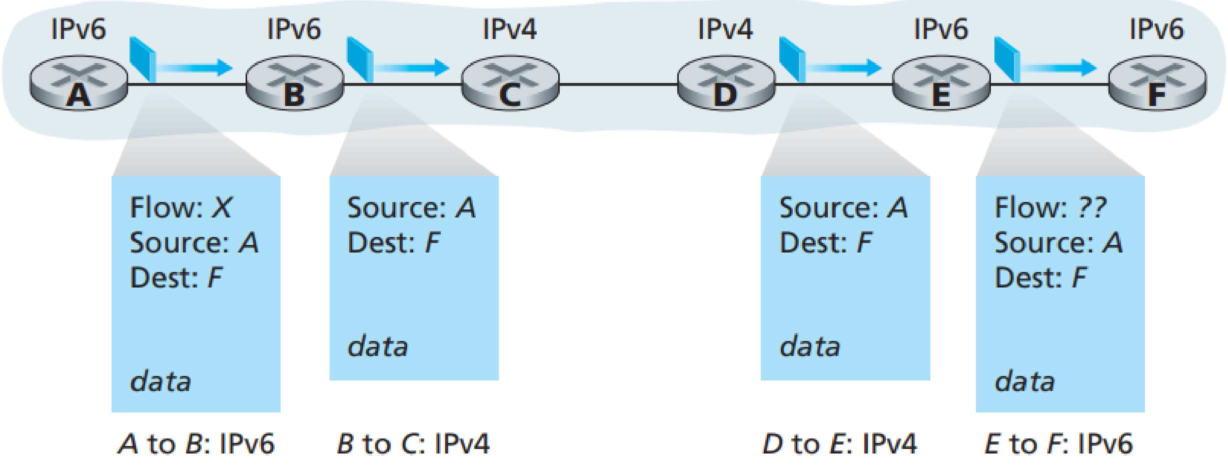
- **Format** of **IPv6 address**:
  - **128 bits** long.
  - **8** groups – **16 bits** each.
  - Each group is expressed by **four hexadecimal digit**.
  - Groups are **separated** by **colons**.



IPv6 address hexadecimal to binary

IPV4 ↔ IPV6

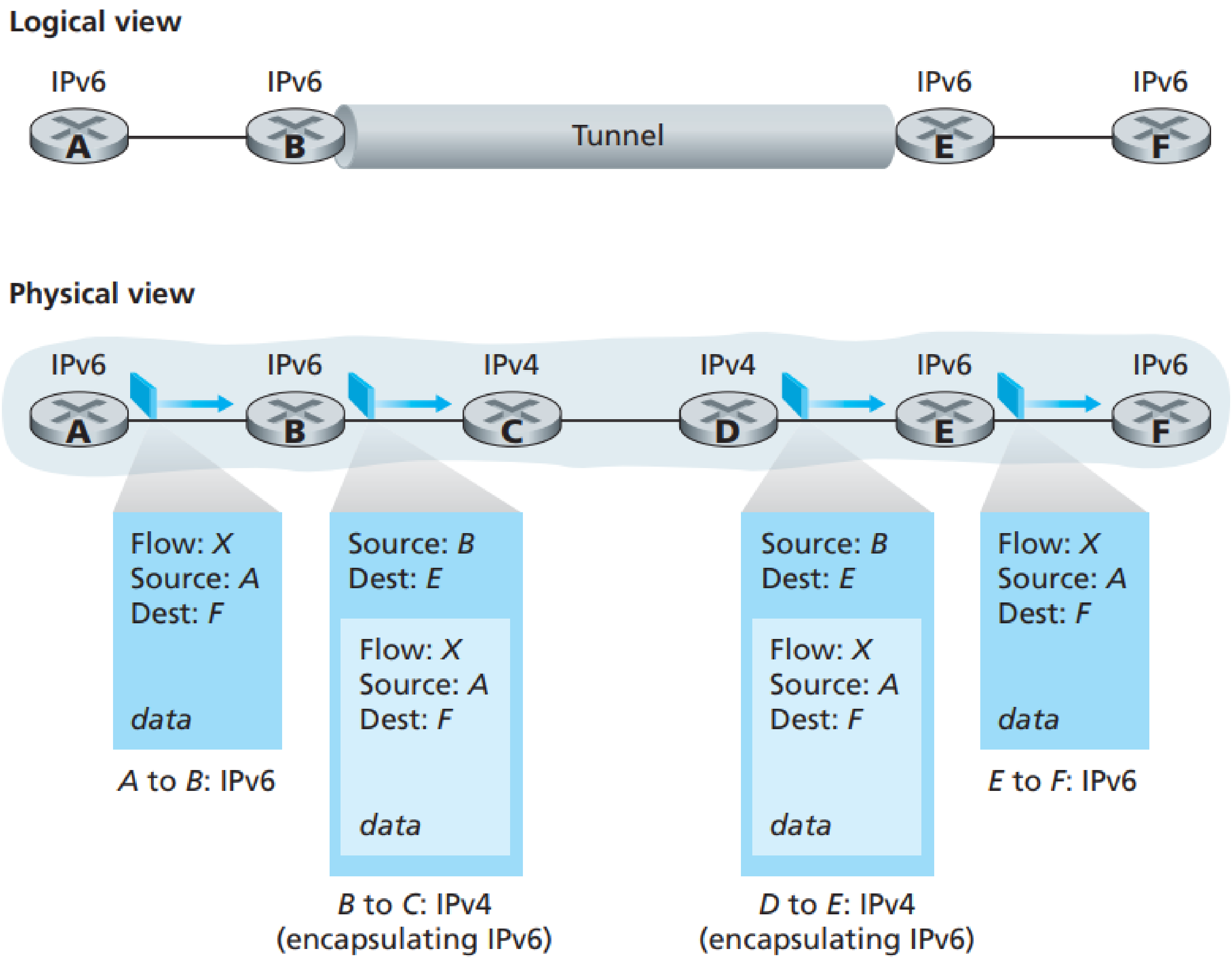
- **IPv4** to **IPv6 transition** options:
  - **“Flag day”** – turn off the Internet and upgrade to IPv6.
    - Not realistic!
  - **Dual-stack** approach.
    - IPv6/IPv4 nodes – able to send and receive both IPv4 and IPv6 datagrams.
    - Will eventually end-up sending IPv4 datagrams only if communicating through IPv4-only nodes.
  - **<u>Solution</u>: tunneling**.



Dual-stack approach

IPV4 ↔ IPV6: TUNNELING

- **Tunneling** - **IPv6 datagram** carried as **payload** in **IPv4 datagram** among **IPv4** routers.

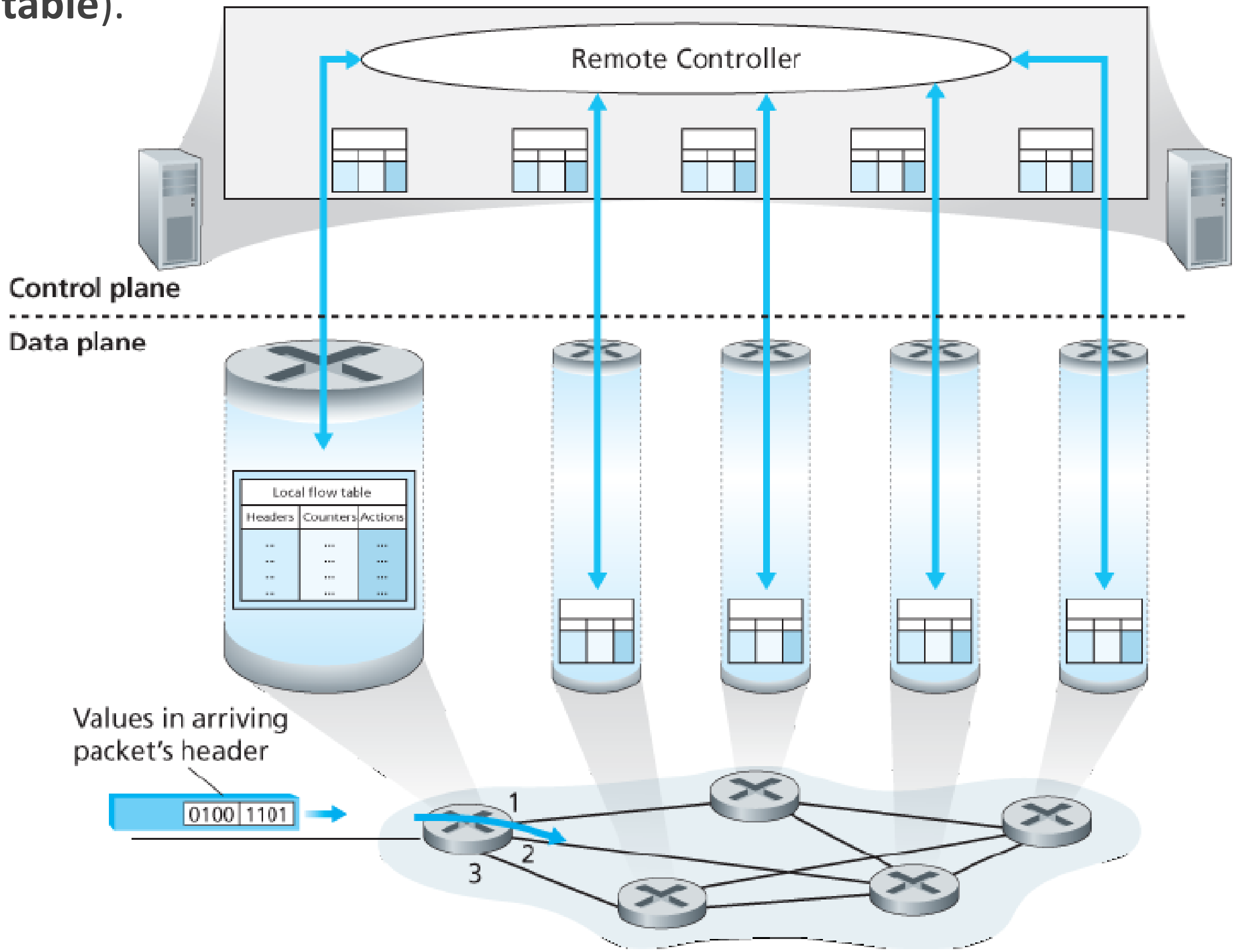


GENERALIZED FORWARDING

- Network layer **forwarding function** provides following **services**:
  - **Rewriting** header IP addresses & port numbers (**NATs**).
  - **Blocking** traffic based on header-field values (**firewalls**).
  - **Redirecting** packets for additional processing (**DPIs**).
  - **Forwarding** packets to servers that provide specific service (**load-balancers**).
- **Destination-based** forwarding is **generalized** into "**match-plus-action**" paradigm.
  - "**Match**" is made over **multiple header fields** of **multiple protocols**.
  - "**Action**" is expanded beyond simple **forwarding**.
- **Routers** are generalized into **packet switches**.
  - Operate on both: network-layer & link-layer source/destination addresses.
- Network is characterized by **software-defined networking** (SDN).

GENERALIZED FORWARDING: OPENFLOW

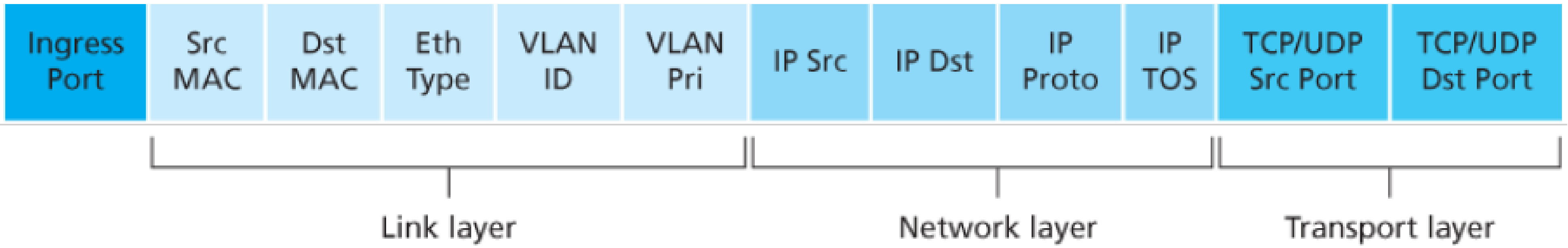
- **SDN** concept was pioneered by **OpenFlow protocol**.
  - Each **packet switch** contains match-plus-action table (**flow table**).
  - Flow tables are **computed**, **installed** & **updated** by **remote controller**.
- **Flow table** contains:
  - Set of **header field values**.
    - **Matched** against header fields in incoming packet.
  - Set of **counters**.
    - **Updated** as packets matched to table entries.
  - Set of actions.
    - **Taken** as packet matched to table entries.



OPENFLOW: MATCH

- OpenFlow "match" abstraction allows **matching** on **fields** from:
  - **Link layer**.
    - Source/destination **MAC addresses**.
  - **Network layer**.
    - Source/destination **IP addresses**.
  - **Transport layer**.
    - Source/destination **port numbers**.

- **Flow table** entries may contain **wildcards**.
  - *128.119.*.** → any datagram that has 128.119 in address field will be matched.



Packet matching fields

OPENFLOW: ACTION

- Each **flow table entry** has zero or more **actions** – processing that is applied if packet is **matched**.
- Possible **actions**:
  - **Forwarding**.
  - **Modifying** field(s).
    - Rewrite header values before forwarding.
  - **Dropping**.

OPENFLOW: EXAMPLES (1)

- IP datagrams destined to **IP address 51.6.0.8** should be forwarded to router **output port 6**.

Ingress Port	Src MAC	Dst MAC	Eth Type	VLAN ID	VLAN Pri	IP Src	IP Dst	IP Proto	IP TOS	TCP/UDP Src Port	TCP/UDP Dst Port	Action
*	*	*	*	*	*	*	51.6.0.8	*	*	*	*	port 6

- Do not forward (block) all datagrams destined to **TCP port 22**.

Ingress Port	Src MAC	Dst MAC	Eth Type	VLAN ID	VLAN Pri	IP Src	IP Dst	IP Proto	IP TOS	TCP/UDP Src Port	TCP/UDP Dst Port	Action
*	*	*	*	*	*	*	*	*	*	*	22	drop

OPENFLOW: EXAMPLES (2)

- Do not forward (block) all datagrams sent by **host 128.119.1.1**

Ingress Port	Src MAC	Dst MAC	Eth Type	VLAN ID	VLAN Pri	IP Src	IP Dst	IP Proto	IP TOS	TCP/UDP Src Port	TCP/UDP Dst Port	Action
*	*	*	*	*	*	128.119.1.1	*	*	*	*	*	drop

- Frames from **MAC address 22:A7:23:11:E1:02** should be forwarded to **output port 6.**

Ingress Port	Src MAC	Dst MAC	Eth Type	VLAN ID	VLAN Pri	IP Src	IP Dst	IP Proto	IP TOS	TCP/UDP Src Port	TCP/UDP Dst Port	Action
*	22:A7:23: 11:E1:02	*	*	*	*	*	*	*	*	*	*	port 3

SUMMARY

- IPv4 datagram format.
- Datagram fragmentation.
- DHCP.
- NAT.
- ICMP.
- IPv4 vs IPv6.
- IPv6 datagram format.
- IPv6 addressing.
- Tunneling.
- SDN & OpenFlow.