

NSWI090: Computer Networks

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

Lecture 8

Addressing

Martin Svoboda

martin.svoboda@matfyz.cuni.cz

2022

Charles University, Faculty of Mathematics and Physics

Lecture Outline

Addresses and addressing

- L2: **MAC** addresses
- L3: **IP** addresses
 - **IPv4 addresses**
 - Subnetting and supernetting
 - CIDR
 - Registries
 - Private addresses and NAT
 - **IPv6 addresses**
- L4: **port** numbers
- L7: **URI** identifiers

Addressing at L2

MAC addresses (**hardware** addresses)

- Used at L2
 - More precisely at the **MAC sublayer** (**Media Access Control**)
- Assigned to **network interface controllers**
 - I.e., individual L2 interfaces of end nodes as well as routers
 - Bridges / switches only have one MAC address, if any
 - In case they should explicitly be accessible as ordinary devices
 - Not because of fulfilling their L2 tasks (filtering / forwarding)
- Types of addresses
 - Standard **unicast**, but also **multicast** and **broadcast**
- **Different technologies** may have different mechanisms
 - However, **sharing of address spaces** may be desirable
 - So that **Wi-Fi and Ethernet** can coexist in one network

Addressing at L2

MAC addresses (cont'd)

- Must be **locally unique** within a given network
 - So that senders can **identify** the intended recipients
 - And these recipients are able to **recognize** their frames
 - Since everything is / may be delivered to everyone
 - Because all nodes are mutually visible and reachable
- **Assignment strategies**
 - **Locally** administered addresses would suffice
 - However, **globally unique addresses** simplify everything
 - Otherwise newly connected devices would need to be treated
 - So that they do not potentially use conflicting addresses
- ⇒ **globally** unique **burned-in** addresses **assigned by device manufacturers** are primarily used in practice
 - Exceptions exist, though

EUI Addresses

EUI numbering systems (**Extended Unique Identifier**)

- **EUI-48** (48 bits, 6 bytes, $\approx$ 281 trillion addresses)
 - Formerly denoted as **MAC-48**
 - Which is inappropriate since MAC denotes the entire sublayer
 - And so this label should no longer be used since it is obsolete
 - **Notation**
 - Six hexadecimal numbers separated by hyphens or colons
 - E.g.: FC-77-74-19-41-1E or FC:77:74:19:41:1E
 - Deployment: **Ethernet**, **Wi-Fi**, **Bluetooth**, **ATM**, ...
- **EUI-64** (64 bits, 8 bytes)
 - Newer version with larger address space
 - **Conversion** of EUI-48 possible by adding FF-FE in the middle
 - E.g.: FC-77-74-FF-FE-19-41-1E
 - Deployment: **FireWire** (IEEE 1394), ...

EUI Addresses

Internal structure

- **Organizationally Unique Identifier (OUI)**
 - **Higher 3 bytes** (24 bits)
 - Describes a particular **vendor or manufacturer**
 - E.g.: FC-77-74 (Intel), 90-F6-52 (TP-Link), ...
- **Interface number**
 - Lower 3 bytes (24 bits) in case of EUI-48
 - Lower 5 bytes (40 bits) in case of EUI-64
 - **Serial number** of a given **network interface controller**

OUI Identifiers

OUI component details

- **M bit (I/G bit)** = the least significant bit of the first byte
 - **0** = **individual address** (unicast)
 - **1** = **group address** (multicast, broadcast)
- **X bit (U/L bit)** = the second least significant bit of the first byte
 - **0** = **Universally Administered Addresses** (UAA)
 - Intended for globally unique addresses
 - **1** = **Locally Administered Addresses** (LAA)
- Examples
 - **Individual** (M = **0**) and **universal** (X = **0**)
 - FC-77-74-19-41-1E: in binary 111111**00**-...
 - **Group** (M = **1**) and **local** (X = **1**)
 - FF-FF-FF-FF-FF-FF (L2 broadcast): in binary 111111**11**-...

OUI Identifiers

Observations

- Why M and X bits are the **two least significant bits**?
 - **Ethernet** uses **Big Endian** for the individual **bytes**
 - But **Little Endian** for individual **bits** within these bytes
 - Therefore the first two transmitted bits become M and X
 - And so various address modes can be distinguished easily
- Both M and X bits are **set to zeros** in **OUI identifiers** as such
 - Only when they are exploited in EUI addresses...
 - ... they can then be changed as required and so their special meaning activated and utilized

OUI Identifiers

Organizationally Unique Identifier (OUI) (cont'd)

- Unique **vendor, manufacturer, or other organization** identifier
 - **Purchased** from the **IEEE Registration Authority**
 - One organization may actually have multiple OUIs at a time
- Various usages
 - Our **EUI-48 and EUI-64 addresses**
 - But also **SNAP Protocol Identifiers** (IEEE 802.2 LLC extension)
 - ...
- **Online list:** <http://standards-oui.ieee.org/oui/oui.txt>
 - Current status (April 2021)
 - Almost 30 thousand OUIs are assigned
 - Huawei $\approx$ 1100, Cisco $\approx$ 1060, Apple $\approx$ 890, Samsung $\approx$ 690, ...
 - TP-Link $\approx$ 160, Technicolor $\approx$ 80, D-Link $\approx$ 60, Zyxel $\approx$ 40, ...
 - More than 17 thousand organizations only have a single OUI

Addressing at L3

IP addresses

- Primary objective is **addressing of nodes** as a whole
- In spite of that...
 - IP addresses are actually assigned to their **network interfaces**
 - I.e., individual L3 interfaces of end nodes as well as routers
 - Note that end nodes used to usually have only one IP address
 - But nowadays, **multi-homed** hosts with more IPs are common
- Another observation...
 - **Routing algorithms perceive networks as a whole**
 - $\Rightarrow$ **networks themselves** must also be **uniquely identified**
 - Though two separate network / node identifiers could work
 - **Internally structured atomic IP addresses** are more practical
 - Otherwise hop-to-hop **routing** and **forwarding** would not work

IP Addresses

Requirements on IP addresses

- **Globally unique** within the whole system of networks
- And internally structured...

Internal components

- **Network part (Network ID)**
 - **Prefix** of the whole address
 - I.e., certain part from the beginning
 - **Uniquely identifies a given network** as a whole
 - Determines affiliation of nodes to a particular network
- **Relative part (Host ID)**
 - Remaining part of the whole address
 - **Uniquely identifies a given node** within a given network

Assignment Principles

Observations = rules that must be followed

- **Two nodes in the same network...**
 - Must have the same network parts
 - And different relative parts
 - So that they can be mutually distinguished
- **Two nodes in different networks...**
 - Must have different network parts
 - So that we can detect they belong to different networks
 - And as for relative parts, they do not matter
 - They may be the same as well as not

Assignment Principles

Block principle

- (1) **network as a whole** must first be assigned with a whole contiguous **block of addresses**
 - I.e., set of IP addresses belonging to a specific range
 - All with the same prefix (network part)
 - **Assignment process** must be **globally coordinated**
 - **IANA** and regional providers
- (2) only than **individual nodes** can be given their addresses
 - Of course, from this range
 - Manually or using **DHCP** or similar protocols

Consequence

- **Unused addresses cannot be used by anyone else!**
 - This may lead / actually led to unacceptable wasting

IPv4 Addresses

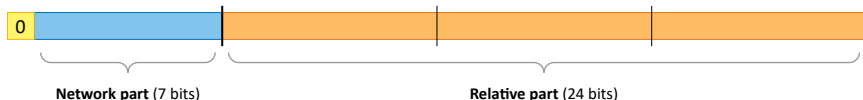
IPv4 addresses

- **4 bytes** (32 bits) – potentially ≈ 4 billion values
 - Which is certainly not much from today's perspective
 - But was initially thought of as generously sufficient
- **Notation**
 - Four decimal numbers, one for each byte, separated by dots
 - E.g.: 195.113.19.170
- **Internal structure: network and relative parts**
 - Where the divide between the parts should be located?
 - 3 possible **divide placements** are possible
 - This gives us **Classes A, B and C** of IP addresses
 - If more options exist, though, how to recognize them?
 - Since it must be possible just from the IP address itself

Classes of IPv4 Addresses

Class A

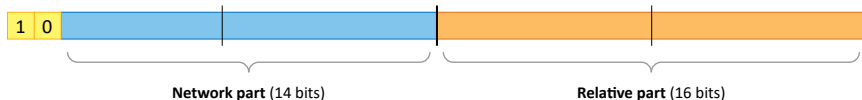
- Divide is positioned **after the first byte**
 - ≈ 128 networks
 - Each with ≈ 17 million addresses
- Overall **range**: **0.0.0.0** – **127.255.255.255**
 - Covers 1/2 of the entire space
- **Highest bit** is **0**
- Suitable for **very large networks**



Classes of IPv4 Addresses

Class B

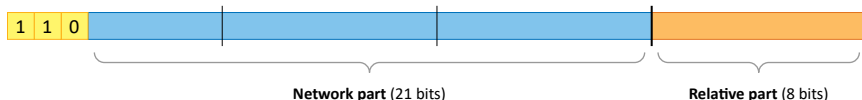
- Divide is positioned **after the second byte**
 - $\approx$ 16 thousand networks
 - Each with $\approx$ 66 thousand addresses
- Overall **range**: **128.0.0.0** – **191.255.255.255**
 - Covers 1/4 of the entire space
- **Highest bits** are **10**
- Suitable for **medium-sized networks**



Classes of IPv4 Addresses

Class C

- Divide is positioned **after the third byte**
 - $\approx$ 2 million networks
 - Each with $\approx$ 256 addresses only
- Overall **range**: **192.0.0.0** – **223.255.255.255**
 - Covers 1/8 of the entire space
- **Highest bits** are **110**
- Suitable for **very small networks**



Classes of IPv4 Addresses

Apparently the **available space** is not yet fully utilized...

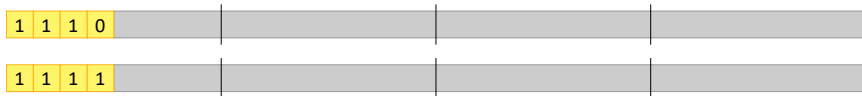
- The rest is covered by **Classes D and E**
 - They have **specific usage** and they are **internally unstructured**

Class D: multicast addresses

- 224.0.0.0 – 239.255.255.255 (1/16 of the entire space)
- Highest bits are **1110**

Class E: reserved for **future extensions**, but never used

- 240.0.0.0 – 255.255.255.255 (1/16 of the entire space)
- Highest bits are **1111**



Special Addresses

Special addresses

- Certain addresses (even whole ranges) have **dedicated usage**
 - Basic principle: bit 0 = **this**, bit 1 = **all**

Nodes

- Self node: **0.0.0.0**
 - Useful when standard unicast address is not yet known
- Node in a local network: **0.X.Y.Z**, **0.0.X.Y**, or **0.0.0.X**

Networks

- Network as a whole: **X.0.0.0**, **X.Y.0.0**, or **X.Y.Z.0**

Broadcasts

- **Targeted** broadcast: **X.255.255.255**, **X.Y.255.255**, or **X.Y.Z.255**
- **Limited** broadcast: **255.255.255.255**

Special Addresses

Loopback

- 1 A-block: **127.X.Y.Z**
 - **127.0.0.1** is in particular usually used
 - But the whole block is in fact available

Link local addresses

- 1 B-block: **169.254.X.Y**
 - Auto-configuration when standard address cannot be obtained

Private addresses

- 1 A-block: **10.X.Y.Z**
- 16 B-blocks: **172.16.X.Y** – **172.31.X.Y**
- 256 C-blocks: **192.168.0.X** – **192.168.255.X**

Multicast Addresses

Multicast transmissions

- Intended recipients are all nodes in a given group
 - This group is predefined or created dynamically on demand
 - **IGMP (Internet Group Management Protocol)**
- Certain blocks are assigned and approved by **IANA**
 - <https://www.iana.org/assignments/multicast-addresses/>
 - Current status (April 2021)
 - ≈ 60 static multicast addresses (beside other)
- **Class D addresses** are used
 - Start with **1110** and they are **internally unstructured**
 - I.e., there is no network / relative part



Multicast Addresses

Static groups (well known groups)

- Node membership is fixed and given in advance
- Range: **224.0.0.0** – **224.0.0.255** (≈ 256 addresses)
 - Reserved for **routing and other low-level protocols**
 - Topology discovery, maintenance, ...
 - **224.0.0.1**: **all hosts** in a given network
 - **224.0.0.2**: **all routers** in a given network
 - ...

Dynamic groups

- **Global**: **224.0.1.0** – **238.255.255.255** (≈ 252 million)
 - Group scope can span multiple different networks
- **Local**: **239.0.0.0** – **239.255.255.255** (≈ 17 million)
 - Group scope is limited to a particular network

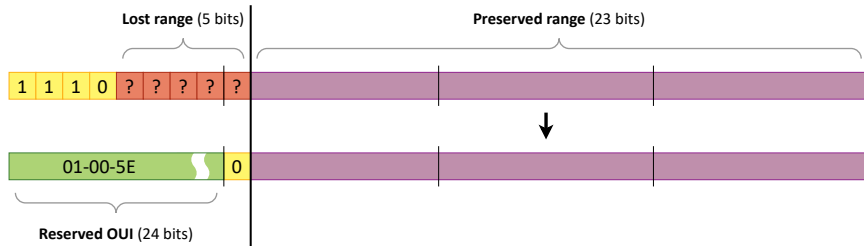
Multicast Addresses

Translation of multicast addresses (IP $\rightarrow$ EUI-48)

- L3 multicast **IP address**
 - 4 bits at the beginning are fixed $\Rightarrow$ only **28 bits are relevant**
 - Unfortunately, **only the last 23 bits** can be considered
 - And so the remaining 5 bits in the middle must be truncated
 - $\Rightarrow$ 32 different addresses are mapped to the same value!
- L2 multicast **EUI-48 address**
 - 00-00-5E reserved **OUI** is used (24 bits)
 - Of course, altered to 01-00-5E so that M bit = 1
 - The 25th bit is set to 0
 - The remaining 23 bits are taken from the original IP address

Multicast Addresses

Translation of multicast addresses (IP $\rightarrow$ EUI-48) (cont'd)



Allocation Strategies

Basic principle

- **Whole blocks** of addresses are / must be assigned to networks
- These blocks correspond to our **classes**
 - One **Class A** block $\approx$ 17 million individual addresses
 - One **Class B** block $\approx$ 66 thousand individual addresses
 - One **Class C** block $\approx$ 256 individual addresses

Strategies

- Formerly: **one closest larger block principle**
 - E.g.: 1 Class B address for 1000 requested individual addresses
 - Extremely wasteful approach...
- Later: **multiple closest smaller blocks principle**
 - E.g.: just 4 Class C addresses for the same requested number
 - Better, but still not enough...

IPv4 Address Exhaustion

Temporary **mitigating solutions**

- 1985: **Subnetting**
 - One larger network is divided into separate sub-networks
- 1988: Allocation mechanism
 - **One larger block** → **more smaller blocks** principle
- 1993: **CIDR (Classless Inter-Domain Routing)**
 - Original concept of IP address classes is entirely dropped
- 1994: **Private addresses**
 - Usage of private IPv4 addresses instead of globally unique ones
 - Requires **NAT (Network Address Translation)**

Permanent solution

- 1995: IPv6 protocol and its **IPv6 addresses**
 - **6 bytes** instead of 4 bytes ⇒ significantly **larger address space**

Subnetting

Motivation

- **One closest larger block allocation principle** is used
- $\Rightarrow$ inner block address space may not be used efficiently
 - Since **unused addresses cannot be used by anyone else**
 - This in fact led to unacceptable wasting
- Objective
 - **Higher utilization of addresses** within allocated blocks

Principle

- **Division of larger blocks into smaller ones**
 - In terms of decomposition of networks into subnetworks

Subnetting

Subnetting

- Standard network is **internally divided into subnetworks**
 - Standard means Class A, B, or C block
- **Divide position is shifted to the right** (toward lower bits)
 - By one or more bits as needed
- $\Rightarrow$ **divide position** must be somehow remembered
 - Since the traditional class boundaries will no longer work
 - And we still must be able to recognize IP address parts
 - Which will now be impossible without extra information
- **Netmask (subnet mask)** was proposed for this purpose
 - Written as an ordinary IP address
 - Contains **bits** 1 in the intended **network part**, **bits** 0 elsewhere
 - E.g.: 255.255.0.0 as an equivalent of Class B network

Subnetting: Example

Assume we have a Class C **network** 195.113.19.0

- Permits ≈ 256 addresses, netmask would be 255.255.255.0

It can be divided into the following **subnetworks**

- Subnetwork 195.113.19.0 with netmask 255.255.255.128
 - I.e., 195.113.19.00000000_B, netmask 255.255.255.10000000_B
 - **Divide shifted by +1**, allows ≈ 128 individual addresses
- Subnetwork 195.113.19.128 with netmask 255.255.255.192
 - I.e., 195.113.19.10000000_B, netmask 255.255.255.11000000_B
 - **Divide shifted by +2**, allows ≈ 64 individual addresses
- Subnetwork 195.113.19.192 with netmask 255.255.255.192
 - I.e., 195.113.19.11000000_B, netmask 255.255.255.11000000_B
 - **Divide shifted by +2**, allows ≈ 64 individual addresses

Subnetting

Observations

- Both **routers and end nodes** must support the whole concept
 - Which is not a big deal since...
 - We are the owners of the **infrastructure** within the network
 - **End nodes** can easily be adapted via software updates
- Subnetting is always **limited to a given standard network** only
 - Its **impact must not be visible from outside**
 - I.e., network as a whole still must act as an atomic unit
 - And so global routing and forwarding are not impacted at all

Supernetting

Motivation

- **More closest smaller blocks allocation principle** is used
- $\Rightarrow$ size of **routing tables** increases
 - Since **each individual network** must have its **own record**
 - Routing tables therefore became unacceptably large
 - As well as routing tables **lookup slowed down**
- Objective
 - **Reducing overall size of routing tables** in backbone routers
 - And so with no impact on depletion of IP addresses themselves

Principle

- **Aggregation of smaller blocks into larger ones**
 - In terms of records in routing tables

Supernetting

Supernetting (Aggregation)

- Several **adjacent aligned** blocks are merged together
 - They must share the same prefix of their network IDs
 - Entire address space defined by this prefix must be covered
 - All the original blocks must have the same routing direction
- **Divide position is shifted to the left** (toward higher bits)
- $\Rightarrow$ once again, **netmasks** are needed

Observations

- Supernetting is **entirely transparent**
 - Contrary to subnetting...
 - And so all **routers** within the system must support the concept

Supernetting: Example

Assume we have the following individual Class C **networks**

- Or just one network with the following address blocks...
 - Target network 195.113.16.0 (i.e., 195.113.00010000_B.0)
 - Target network 195.113.17.0 (i.e., 195.113.00010001_B.0)
 - Target network 195.113.18.0 (i.e., 195.113.00010010_B.0)
 - Target network 195.113.19.0 (i.e., 195.113.00010011_B.0)
- They all have the same routing direction

Their **routing records** can thus be grouped together

- Target network 195.113.16.0 with netmask 255.255.252.0
 - I.e., 195.113.00010000_B.0, netmask 255.255.11111100_B.0
 - **Divide shifted by -2**

Regional Registries

Original arrangement

- **Entire address space** was managed by **IANA**
 - I.e., individual blocks were **directly** assigned to **end users**
 - In terms of **Class A, B, or C blocks**
- Involved **agenda** became far too **extensive and demanding**
 - $\Rightarrow$ individual regional registries were gradually founded
 - And related agenda correspondingly transferred

Regional Internet Registry (RIR)

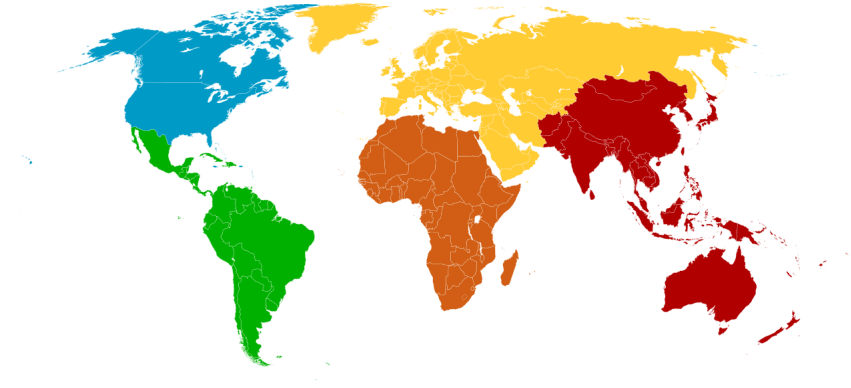
- **Organization** managing **allocation and registration** of Internet resources within a given region
 - **IP addresses** and **Autonomous System Numbers**
- **5 individual RIRs** around the world exist nowadays
 - Each obtains larger blocks of IP addresses from IANA

Regional Registries

Regional registries

- 1992: **RIPE NCC** (Réseaux IP Européens Network Coordination Centre)
 - Europe, Central Asia, Russia, West Asia
- 1993: **APNIC** (Asia Pacific Network Information Centre)
 - South, East, and Southeast Asia, Oceania
- 1997: **ARIN** (American Registry for Internet Numbers)
 - USA, Canada, Antarctica, ...
 - Operating in fact since 1991
- 1999: **LACNIC** (Latin America and Caribbean Network Information Centre)
 - Latin America, Caribbean
- 2004: **AFRINIC** (African Network Information Centre)
 - Africa

Regional Registries



Motivation

- **Allocation of address blocks is still not flexible enough**
 - Because of **coarse granularity** of possible block sizes
 - Especially for networks of **mid-sized organizations**
 - Class C with ≈ 256 addresses is too small
 - Class B with ≈ 66 thousand addresses is too large
- **Subnetting** and **supernetting** both helped...
 - But **transparent** solution is needed so that the **entire address space** can be exploited efficiently enough (not just parts of it)
- Objective
 - **Hierarchical allocation of address blocks with arbitrary sizes**
 - So that block sizes can better match projected needs
- Principle: **fully classless routing mechanism**

CIDR

CIDR (Classless Inter-Domain Routing)

- **Concept of classes** is now definitely abandoned
 - Except for Classes D (multicast addresses) and E (future use)
 - Their meaning and ranges were preserved untouched
 - Including the meaning of other special addresses
- **Divide can now be placed anywhere**
 - I.e., leading bits no longer determine anything
 - And so **divide position** must once again be explicitly declared
 - Though analogous to netmasks...
 - ... different and more convenient notation was introduced
- **CIDR Prefix** (or simply **prefix**)
 - **Number of bits** forming the network part
 - Written as a decimal number after the slash symbol at the end
 - E.g.: 172.217.0.0/16 as an equivalent of former Class B network

CIDR: Example

Assume we have an **allocated block** 195.113.19.0/24

- It can internally be divided into the following networks
 - Network A: 195.113.19.0/25 (i.e., 195.113.19.00000000_B/25)
 - Network B: 195.113.19.128/26 (i.e., 195.113.19.10000000_B/26)
 - Network C: 195.113.19.192/26 (i.e., 195.113.19.11000000_B/26)

Detailed **routing information**...

- Can remain **entirely undisclosed**
 - 195.113.19.0/24 for all our networks
- As well as intentionally **fully or partially exposed** if needed
 - 195.113.19.0/25 for network A
 - 195.113.19.128/25 for aggregated networks B and C

Observations

- CIDR is **deployed globally** and **fully transparent**
 - I.e., its scope is not limited just to a particular internetwork
 - As was the case of subnetting alone
- Meaning of **former classes** can be preserved
 - Class A / B / C blocks correspond to CIDR prefixes 8 / 16 / 24
 - By the way, individual addresses have CIDR prefix 32
- However, they can also be **transparently decomposed...**
 - E.g., 172.217.23.0/24 is a CIDR block with prefix 24
 - I.e., it provides ≈ 256 individual addresses
 - As if it was just an ordinary Class C block
 - But it is not, since it is just a part of a former Class B block
 - Such a thing would not be possible without CIDR
- As a consequence, **entire address space is treated uniformly**

Observations (cont'd)

- Ideas of both **subnetting** and **supernetting** are supported
 - **Larger blocks can be divided into smaller ones**
 - In terms of decomposition of networks
 - So that the **address space** can be utilized more efficiently
 - Because block sizes can be chosen with **finest granularity**
 - **Smaller blocks can be aggregated into larger ones**
 - In terms of grouping of routing records
 - So that size of **routing tables** can hopefully be reduced
 - And so detailed routing information can remain localized
 - Without needing it to be disseminated globally
- Allows for **hierarchical assignment of address blocks**
 - And so the whole hierarchy of registries
 - Which also helps with the growing **agenda**

Hierarchy of Registries

Hierarchy levels

- **CIR (Central Internet Registry) = IANA**
- **RIR (Regional Internet Registry)**
 - RIPE NCC, APNIC, ARIN as well as later on LACNIC and AFRINIC
 - Later on liaised through **NRO (Number Resource Organization)**
 - Informal body coordinating matters of global importance
- Optional **NIR (National Internet Registry)**
 - National allocaters in **larger countries** only
 - **APNIC** region: China, India, Japan, Korea, Indonesia, ...
 - **LACNIC** region: Brazil, Mexico
- **LIR (Local Internet Registry)**
 - **ISPs**, larger enterprises, or **academic institutions**
 - Membership in a given RIR / NIR is required

Allocation Mechanisms

Example: end node **195.113.19.170**

- **IANA**
 - 195.0.0.0/8 ($\approx$ 17 million addresses) → **RIPE NCC**
- **RIR: RIPE NCC**
 - 195.113.0.0/16 ($\approx$ 66 thousand addresses) → **Cesnet**
 - Autonomous System AS2852
 - 195.113.0.0/18 ($\approx$ 16 thousand) → **Charles University**
 - Publicly invisible as for routing records
- **Internal invisible decomposition**
 - ...
 - 195.113.18.0/23 ($\approx$ 512 individual addresses)
- **Target end node**
 - 195.113.19.170/32 → nosql.ms.mff.cuni.cz

Allocation Mechanisms

Allocation process

- **IANA**
 - Delegates **/8 blocks** to individual RIRs
 - Certain blocks are assigned to particular organizations directly
 - E.g.: US Postal Service, US Department of Defense, ...
 - **Online database**
 - <https://www.iana.org/assignments/ipv4-address-space/>
 - Contains only records for **/8 blocks**
- **RIRs / NIRs**
 - Delegate parts of allocated blocks to subordinated LIRs
- **LIRs**
 - Assign smaller blocks to **end users**
 - Often **singleton addresses** only
 - Consequence: **addresses became dependent** on particular LIRs

Allocation Mechanisms

IANA top level **database** for /8 blocks

- **Types** of records
 - **Allocated**
 - Delegated entirely to a specific RIR or other organization
 - **Legacy**
 - Formerly allocated by IANA prior to the foundation of RIRs
 - Later on transferred to and administered by individual RIRs
 - **Reserved**
 - Designated for **specific purposes** (e.g., loopback, ...)
 - **Unallocated**
 - Not yet allocated or reserved and so **available** for assignment

Allocation Mechanisms

Current situation (May 2021)

- **Distribution of /8 blocks** between individual RIRs
 - ARIN (93), APNIC (50), RIPE NCC (40), LACNIC (10), AFRINIC (6)
 - Class D multicast addresses (16), Class E future use (16), ...
- **Overall allocation of /8 blocks**
 - 2011: IANA delegated the very last available blocks
 - One to every individual of all 5 RIRs
 - $\Rightarrow$ there is **no longer any unallocated /8 block**
- Situation in **RIPE NCC**
 - 2019: the very last block from the pool was allocated
 - Only **recovered addresses** via a **waiting list** are now available
 - I.e., blocks returned by former LIR holders
 - Currently $\approx$ 320 thousand individual addresses available

Private Addresses

Motivation

- Each node must have a **globally unique IP address**
 - I.e., address distinct within the whole system of networks
 - Otherwise routing will not work
 - **Number of available addresses is still decreasing**, though
 - Despite all the other already discussed mitigating measures
- Idea
 - **Nodes in a private network** can use **private addresses** instead
 - These **private addresses will then be translated** to public ones
 - In order to ensure they do not leave a given private network
- Two basic translation mechanisms are available
 - **NAT (Network Address Translation)**
 - **L7 Gateways**

Private Addresses

Observations

- **Any range of addresses could theoretically be used**
 - However, it is not desirable and correct
 - In particular, when such addresses would (even accidentally) **leak out from a given private network...**
 - It will not be possible to remedy the situation later on
 - Simply because other routers will not be able to detect them
- Therefore **dedicated addresses** should be used
 - 1 Class A block: **10.X.Y.Z**
 - I.e., **10.0.0.0/8**
 - 16 Class B blocks: **172.16.X.Y – 172.31.X.Y**
 - I.e., **172.16.0.0/12**
 - 256 Class C blocks: **192.168.0.X – 192.168.255.X**
 - I.e., **192.168.0.0/16**

Network Address Translation

Network Address Translation (NAT)

- Generic translation mechanism
 - Allows not just to spare public IP addresses
 - Probably the most successful mitigating solution
- Deployment
 - **Inner private and outer public networks**
 - Separated by a router implementing the NAT mechanism
- Disadvantages
 - **Decreases the overall throughput**
 - **May not always work**
 - Since the translation primarily works at L3 / L4 layers
 - Therefore it is incapable of modifying L7 data which may also contain the same addresses otherwise subjected to translation

NAT Principle

Delivery mechanism

- **Outgoing** transmission
 - (1) **inner node sends an IP datagram** in a standard way
 - Source address is set to private IP_S address of a given node
 - Destination address is IP_T address of the intended recipient
 - (2) this **datagram is captured by the router** and **NAT is applied**
 - Source address is replaced with appropriate public IP_P address
 - (3) **modified datagram is then sent to the public network**
 - As if its sender was actually the router itself
- **Incoming** transmission
 - (4) response from the target recipient is delivered back to IP_P
 - (5) this response is captured by our router and translated
 - Destination address is replaced with the original IP_S
 - (6) modified response is internally sent to our node

Static and Dynamic NAT

Static NAT

- Mapping table is **fixed and given in advance**
 - Each node will always be mapped to the same public address
 - And so individual bindings are predictable in advance
- Inner nodes are always **reachable from outside**
 - In terms of incoming transmissions initiated from outside
 - I.e., not in terms of responses to our outgoing transmissions
 - Since these are always deliverable
- Disadvantage
 - **Sizes of both public and private blocks must be identical**
 - And so there is no saving effect
 - At least unless certain inner nodes shall not communicate at all

Static and Dynamic NAT

Dynamic NAT

- Records in mapping tables are added / removed **dynamically**
 - I.e., individual **bindings are created on demand**
 - Only when they are really needed (new outgoing connection)
 - And will only exist for a limited period of time
 - **Assigned address may always be different** for a given node
 - And so it cannot be determined in advance
- Inner nodes are **not (automatically) reachable from outside**
 - **Unless their bindings already exist**
 - This may be treated as a disadvantage
 - As well as on the contrary...
 - Since NAT can therefore act as a kind of **firewall**
- **Not all inner nodes must necessarily have their bindings**
 - And so we can really save public addresses

Port Address Translation

Network Address and Port Translation (NAPT)

- Also abbreviated just as **Port Address Translation (PAT)**
- **Motivation**
 - Suitable when we do not have as many public addresses as nodes in our private network
 - This often means we may only have just a **single public address**
- **Principle**
 - All inner nodes are mapped to the same public address
 - Individual transmissions are **distinguished via different ports**

PAT Principle

Delivery mechanism

- **Outgoing** transmission
 - (1) **inner node sends an IP datagram** in a standard way
 - Source transport private address is $IP_S:port_S$
 - Destination transport address of recipient is $IP_T:port_T$
 - (2) this **datagram is captured by the router** and **PAT is applied**
 - Source address is replaced with appropriate public $IP_P:port_P$
 - Datagram TCP / UDP payload is also accordingly translated
 - (3) **modified datagram is then sent to the public network**
 - As if its sender was actually the router itself
- **Incoming** transmission
 - Steps (4), (5), and (6) for response delivery are analogous...

PAT Observations

Dynamic character

- Created **bindings** always exist only for a **limited period of time**
 - And so responses can only be delivered during this window
- Depends on a **particular L4 protocol** and its **implementation**
 - **UDP**: 30 – 300 seconds
 - **TCP**: 30 – 60 minutes

PAT alternatives

- How the assigned public $IP_P:port_P$ address is to be resolved?
 - It can depend solely on the source private $IP_S:port_S$ address
 - **Full / IP Restricted / Port Restricted Cone NAT**
 - As well as even on the intended destination $IP_T:port_T$ address
 - **Symmetric NAT**
- Incoming transmissions from which nodes will be accepted?

PAT Alternatives

Full Cone NAT

- Assigned public $IP_p:port_p$ depends...
 - Solely on private $IP_s:port_s$ address
 - I.e., assigned public address remains the same even for further connections to possibly different intended destinations
- **Responses** to $IP_p:port_p$ are accepted from...
 - **All IP addresses and all their ports** without any limitation

IP Restricted Cone NAT (or just **Restricted Cone NAT**)

- Assignment rules are the same
- **Responses** to $IP_p:port_p$ are accepted only from...
 - **Contacted IP addresses and still all their ports**

PAT Alternatives

Port Restricted Cone NAT

- Assignment rules are the same once again
- **Responses** to $IP_p:port_p$ are accepted only from...
 - **Contacted IP addresses and only from contacted ports**

Symmetric NAT

- Assigned public $IP_p:port_p$ depends...
 - Both on private $IP_s:port_s$ and destination $IP_T:port_T$ addresses
 - I.e., assigned public address (in particular its port) changes every time a different destination is requested
- **Responses** to $IP_p:port_p$ are accepted only from a given single...
 - **Contacted IP address and its only contacted port**

IPv6 Addresses

Motivation

- IPv4 exhaustion problem
 - **Temporary mitigating measures** worked
 - And actually worked better than it was perhaps anticipated
 - Since the first exhaustion threat appeared around 1990
 - And IPv4 is still the primary approach even after 30 years
 - Nevertheless, **permanent solution was needed**
 - Formerly intended **Class E** was not found realistically applicable
 - Since IPv4 would need to be entirely redesigned
 - Together with dozens of other related protocols (OSPF, RIP, ...)
 - And so an entirely new solution was introduced
- **IPv6 protocol and addresses**
 - Primarily **larger address space**
 - But also several other **major improvements** and changes

IPv6 Addresses

IPv6 addresses

- Differences to IPv4
 - **More address hierarchy levels**
 - Allows for better aggregation
 - And so smaller and localized routing tables
 - **Easier assignment of addresses**
 - Including autoconfiguration options
 - Network interface can have several unicast addresses at a time
 - **Introduction of anycast addresses**
 - And removal of broadcast addresses
 - **Significantly larger address space**
 - 128 bits (16 bytes) instead of just 32 bits (4 bytes)
 - Theoretically $\approx 3.4 \times 10^{38}$ individual addresses
 - Compared to just $\approx 4.3 \times 10^9$ in case of IPv4

IPv6 Addresses

IPv6 addresses

- **Notation**

- Eight 2-byte-long words, written as 4-digit hexadecimal numbers, mutually separated by colons
- E.g.: 805b:2d9d:dc28:0000:0000:fc57:d4c8:1fff

- **Abbreviations**

- **Suppressed leading zeros**

- Leading zeros in individual words are truncated
- E.g.: 805b:2d9d:dc28:0:0:fc57:d4c8:1fff

- **Zero-compressed**

- One adjacent group of zero words is entirely omitted
- E.g.: 805b:2d9d:dc28::fc57:d4c8:1fff

- **Mixed notation** for **embedded IPv4 addresses**

- E.g.: 0:0:0:0:0:0:212.200.31.255 or ::212.200.31.255

IPv6 Addresses

Types of addresses

- **Unicast addresses**
 - Allow for standard communication with **individual nodes**
- **Multicast addresses**
 - Allow for communication with **all nodes in a given group**
 - Always start with ff/8 (11111111_B...)
 - Can be used to simulate traditional **broadcasts**, too
- **Anycast addresses**
 - Allow for communication with **one node from a given group**

IPv6 Addresses

Types of **unicast** addresses

- **Global Unicast**
 - Globally unique public individual addresses
- **Local Unicast (Unique Local Addresses) (ULA)**
 - Private addresses unique within all subnets of a given site
 - Labeled with such site and subnet identifiers
 - Therefore suppressed chances of accidental leak outs
 - Always start with `fc00/7` (`1111110B...`)
- **Link Local**
 - Private addresses unique within a given individual subnet
 - Allow for **autoconfiguration** based on MAC addresses
 - Always start with `fe80/10` (`1111111010B...`)
- **Site Local**
 - Private addresses unique within all subnets of a given site
 - Should no longer be used

IPv6 Addresses

Global individual addresses

- Three components
 - **Site identifier**
 - Identifies a particular site = group of related networks
 - **Subnet identifier**
 - Identifies a particular network within a given site
 - **Interface identifier**
 - Identifies a particular node within a given subnet
- **Routing mechanisms**
 - **Public topology**
 - Routing algorithms in public Internet only work with sites
 - **Site topology**
 - Internal site topology is concealed to the public Internet

Addressing at L4

Port numbers

- Assigned to **access points** between L4 and L7 in order to allow **end-to-end** communication of individual **application entities**
 - Within a given end node
 - **Separately for each transport protocol (TCP, SCTP, DCCP, UDP)**
 - Yet in practice usually the same for all these protocols
- Necessary for both **incoming and outgoing** directions
- Allow for the identification of **transport connections**
 - Tuple (sender $IP_1:port_1$, protocol, recipient $IP_2:port_2$)
 - **Target** of the **outgoing** transmission ($IP_2:port_2$, protocol)
 - **Source** of the **incoming** transmission ($IP_1:port_1$, protocol)
- **Requirements**
 - Ports must be **unique, abstract, implicit and static**

Ports

Port numbers

- **16-bit** long integer numbers: permitted **values 0 – 65535**
- System and registered ports maintained by **IANA**
 - **Internet Assigned Numbers Authority**
- Online table of **port numbers** and **service names**
 - <https://www.iana.org/assignments/service-names-port-numbers/>

Types of ports

- **System Ports (Well Known Ports)** (0 – 1023)
 - **Assigned to one purpose** (usually system-oriented services)
 - Should not be used for any other purpose
 - Examples
 - FTP (21), SSH (22), SMTP (25), DNS (53), HTTP (80), POP3 (110), NTP (123), IMAP (143), HTTPS (443), ...

Ports

Types of ports (cont'd)

- **User Ports (Registered Ports)** (1024 – 49151)
 - **Assigned to one purpose** again
 - But may freely be used for any other purpose
 - Examples
 - Registered: MySQL (3306), PostgreSQL (5432), Redis (6379), Neo4j (7474), MongoDB (27017), ...
 - Not registered: Riak (unsigned 8087 and 8098), Cassandra (assigned 7000), ...
- **Dynamic Ports (Private Ports)** (49152 – 65535)
 - Not assigned, available for unrestricted usage
 - Usually for **outgoing transmissions**

Addressing at L7

Requirements and expectations

- **Various kinds of objects** need to be identified at L7
 - Web pages, files, e-mail addresses, publications, ...
- Two aspects actually need to be covered
 - **Identification**
 - So that objects of one kind can mutually be distinguished
 - At least **locally** (within a given end node) but also **globally**
 - **Location**
 - In terms of a particular node where such objects can be found
 - It make sense to logically decouple both these aspects
- **Each application** may have its own proprietary **naming system**
 - Yet it makes sense to pursue **unification** and **coordination**
 - As well as to recycle approaches that already exist

URI Framework

Uniform Resource Identifier (URI)

- Generic, federated and extensible **naming system**
 - Allows to identify basically anything
 - Including real-world objects (people, places, concepts, ...)
- Types of identifiers
 - **Uniform Resource Locator (URL)**
 - **Web resource reference** (web address)
 - Specifies particular location as well as retrieval mechanism
 - **Uniform Resource Name (URN)**
 - **Globally unique persistent resource identifier**
 - Does not imply any location, not widely used
 - **Uniform Resource Characteristics (URC)**
 - Description of **meta data** about URLs or URNs (citations, ...)
 - Never standardized, not even implemented

URI Examples

Sample URLs

- **http** scheme (**H**ypertext **T**ransfer **P**rotocol)
 - Addresses of web pages or other resources
 - E.g.: **http**://www.mff.cuni.cz/en/index.php?page=people#5460
- **ftp** scheme (**F**ile **T**ransfer **P**rotocol)
 - Paths to files or directories accessible using the FTP protocol
 - E.g.: **ftp**://svoboda:password@ulita.ms.mff.cuni.cz/
- **file** scheme
 - Host-specific paths on local or remote **file systems**
 - E.g.: **file**:///home/svoboda/NSWI090/Lecture-03-Layers.pdf
- **mailto** scheme
 - **E-mail addresses** including additional parameters
 - E.g.: **mailto**:svoboda@ksi.mff.cuni.cz?subject=NSWI090

URI Examples

Sample URLs

- **tel** scheme
 - **Telephone numbers**
 - E.g.: **tel**:+420-951-554-250
- **sip** scheme (**Session Initiation Protocol**)
 - Participants of multimedia sessions such as **voice calls** (VoIP, ...)
 - E.g.: **sip**:martin.svoboda@mff.cuni.cz
- **jdbc** scheme (**Java Database Connectivity**)
 - Connections to **relational databases** from Java applications
 - E.g.: **jdbc**:postgresql://nosql.ms.mff.cuni.cz:5432/database

URI Examples

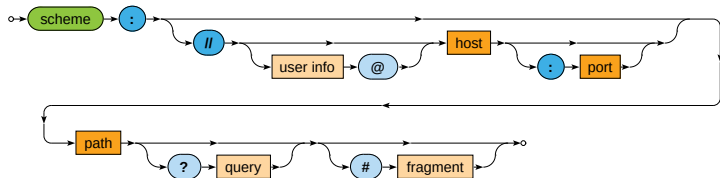
Sample URNs

- **isbn** namespace (**International Standard Book Number**)
 - Printed or electronic **books**
 - E.g.: `urn:isbn:9780132126958`
- **issn** namespace (**International Standard Serial Number**)
 - Printed or electronic **serial publications** (journals, ...)
 - E.g.: `urn:issn:0302-9743`
- **ietf** namespace (**Internet Engineering Task Force**)
 - IETF family of RFC, STD, FYI, and BCP documents
 - E.g.: `urn:ietf:rfc:2648`
- **iso** (**International Organization for Standardization**)
 - **Standards** and other technical specifications
 - E.g.: `urn:iso:std:iso-iec:9075:-1:ed-5:en`

URI Structure

Generic syntax

- May further be restricted by particular schemes
- **Syntax diagram** (for context-free grammars)



- **Green boxes: literals**
 - Expected to be replaced with a particular value
- **Blue boxes: constants** (preserved as they are)
- **Orange boxes: non-terminals**
 - Unfolded to more complicated fragments

URI Schemes

URI components

- **Scheme**: case-insensitive **scheme name** (usually lower-case)
- **Authority**
 - **User info**: authentication tokens such as **name** and **password**
 - Deprecated for security reasons
 - **Host**: **domain name**, **IP address**, or a different registered name
 - **Port**: transport layer **port**
- **Path**: usually **hierarchical path** with individual segments
- **Query**: usually parameters in a form of **attribute / value pairs**
- **Fragment**
 - Reference to a **secondary resource** related to the primary one
 - E.g.: anchors in HTML pages, classes in OWL ontologies, ...

URI Framework

Both **schemes** and **namespaces** are registered with **IANA**

- **URI schemes**
 - <https://www.iana.org/assignments/uri-schemes/>
 - Current status (April 2021)
 - $\approx$ 100 permanent, 230 provisional, and 10 historical schemes
- **URN namespaces**
 - <https://www.iana.org/assignments/urn-namespaces/>
 - Current status (April 2021)
 - $\approx$ 70 namespaces

Internationalized Resource Identifier (IRI)

- Just an extended version of the traditional URIs
- Allows to use most of **Unicode characters**
 - And so Chinese, Japanese, Korean or other national characters

Lecture Conclusion

L2: **MAC** addresses

- **EUI-48** and EUI-64 numbering systems, **OUI**, M and X bits

L3: **IPv4** addresses

- **Network and relative** parts
- **Classes** A, B, C, D, and E
- Special and **multicast** addresses
- **Subnetting** and **supernetting**
- **CIDR** blocks and prefixes
- RIR, NIR, and LIR **registries**
- **Private addresses** and **NAT / PAT** translation

Lecture Conclusion

L3: **IPv6** addresses

- **Site, subnetwork, and interface** parts
- Types of addresses

L4: **port** numbers

- System / User / Dynamic ports

L7: **URI** identifiers

- URI (IRI) framework: **URL** locators, **URN** names