RCOMP - Redes de Computadores (Computer Networks)

2024/2025

Lecture 07

- IPv6 and ICMPv6.
- Names resolution.
- DNS.

Internet Protocol version 6 (IPv6)

In the 90's, the internet reached an initially unexpected expansion leading to the IPv4 address space exhaustion. Meanwhile, several steps were taken to make better use of the rather limited 32-bits IPv4 address space:

- The use of 8-, 16-, and 24-bits network prefix-length (classful) as a first step in adjusting the number of addresses assigned to a network to its real needs.
- Classless addressing. The free network prefix-length definition allows a much better adjustment to real needs (e.g., 30 bits prefix for a two nodes dedicated link).
- Use of private addresses together with address translation (NAT) capable devices. A big number of private node addresses, not globally unique, all sharing a single globally unique public address.

With special focus on definitely solving the address space issue, a successor for IPv4 was developed, initially known as IP-NG (IP *Next Generation*) it was later given the version number six and is now currently known as IPv6.

Hopefully, one day IPv6 is going to be the only IP protocol used on the internet, but for now, both IPv4 and IPv6 coexist.

IPv6 – 128-bits node addresses

Undoubtedly one major IPv6 improvement over IPv4 it´s simply about the address space size. An IPv6 node address has four times more bits than an IPv4 node address.
Of course, the address space size (possible addresses) grows exponentially, so it’s 2^{128} addresses in IPv6 against 2^{32} addresses in IPv4. The IPv6 address space is so huge, it would almost be sufficient to assign the equivalent to the entire IPv4 address space to each earth inhabitant.

And yet, changes from IPv4 to IPv6 are not restricted to the address space:

- Fragmentation in intermediate nodes was abandoned, only Path MTU Discovery**.**
- Layer two addresses handled by ICMPv6 (IPv4 requires ARP).
- Integrated multicast groups support in ICMPv6 (IPv4 requires IGMP).
- Nodes auto-configuration avoiding the use of DHCP.
- Datagrams up to 4 Gb (on IPv4 only up to 64 Kb is supported).
- Integrated authentication and confidentiality support (IPv4 requires IPSEC).

IPv6 addresses representation

IPv6 addresses are represented as eight groups of sixteen bits separated by colons, each in hexadecimal notation (lowercase hexadecimal symbols are preferred).

`xxxx:xxxx:xxxx:xxxx:xxxx:xxxx:xxxx:xxxx`

The representation should be condensed by removing leading zeros within each sixteen bits group. Also, the longest sequence of groups with value zero should be replaced by a double colon. For instance:

`0000:3278:0a04:0005:0000:0000:0000:0034`

Should be presented as:

`0:3278:a04:5::34`

IPv4 addresses can be represented within IPv6 address space by using the `::ffff:0:0/96` addresses block, the IPv4 address fills the 32 less significant bits and is usually represented in dot-decimal notation, this is called **IPv4-mapped**:

The IPv4 address **`X1.X2.X3.X4`** becomes **`::ffff:X1.X2.X3.X4`**

IPv4-mapped IPv6 addresses are rather convenient because most current nodes are dual-stack (IPv4 + IPv6), so, applications must handle both with IPv4 and IPv6 addresses. With IPv4-mapped, both IPv4 and IPv6 are handled the same way.

IPv6 – address types

As with IPv4, the address space is divided into networks, the initial part of the address (prefix) identifies the network, the rest will identify a node within that network. For instance:

`2001:0db8:2b00::/40`

Represents a 40 bits mask network (40 bits prefix-length).

The 1st node is: <u>2001:0db8:2b</u>00::1 (<u>2001:0db8:2b</u>00:0000:0000:0000:0000:0001 **)**

The last node is: <u>2001:0db8:2b</u>ff:ffff:ffff:ffff:ffff:ffff

Unlike with IPv4, in IPv6 the network's last address is not reserved for broadcast, in fact, there's no broadcast in IPv6, multicast is used instead.

IPv6 supports three address types

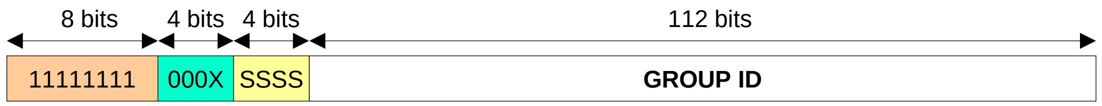
UNICAST – represents a single node, must be unique, data is delivered to that node only.

MULTICAST – represents a set of nodes, a data copy is delivered to each one of them.

ANYCAST – represents a set of nodes, data is delivered to only one of them.

IPv6 – multicast addresses

IPv6 multicast addresses start by a sequence of eight one bits: ff00::/8



For well-known multicast addresses, the X bit will be zero, one for others. The next four bits settle the scope, meaning how far is the address valid, and thus, how far traffic will reach, some important values are:

0x1 – For node-local or interface-local – within the sender node

0x2 – For link-local – the local network (layer two network)

0xe – For global – all the internet

Each well-known multicast address supports only some scopes, these addresses are used to identify and locate services, for instance:

ff02::1	all nodes in the local networks (somewhat equivalent to IPv4 broadcast)
ff02::2	all routers on the local network.
ff02::1:2	all DHCP agents on the local network.
ff02::43	all NTP servers on the local network (scope 0x2)
ff0e::43	all NTP servers on the Internet (scope 0xe).

IPv6 – multicast addresses and Ethernet

Ultimately, IPv6 datagrams sent to a multicast address must be transported by layer two frames, in local networks most often Ethernet. IPv6 multicast addresses are mapped to Ethernet multicast addresses **33:33:xx:xx:xx:xx**.
For Ethernet, the less significant bit of the first byte is the **multicast bit**.
Ethernet switches retransmit to all ports, frames with destination addresses having value one in the multicast bit.

IPv6 multicast addresses are mapped to these Ethernet multicast addresses by coping the less significant 32 bits, thus for each IPv6 multicast address there’s a corresponding Ethernet multicast address, for instance:

The IPv6 multicast address:	**`FF00::2AB1:20A3`**
, is mapped to the Ethernet multicast address:	**`33:33:2A:B1:20:A3`**

Regarding IEEE 802.11 wireless networks, they also handle MAC addresses started by 33:33 as layer two multicast addresses.

Unlike IPv4, where an additional protocol is required (IGMP), in IPv6 multicast groups are managed by ICMPv6 (the ICMP equivalent for IPv6).

IPv6 – link-local unicast addresses

Unicast addresses are unique addresses assigned to nodes, though, in IPv6 all depends on the scope. For link-local scope, they are required to be unique only within the attached layer two network. To be able to start operating, an IPv6 node will assign to itself a link-local unicast address. The **`fe80::/10`** addresses prefix is reserved for this purpose; each node uses this prefix and generates the node bits from its own layer two MAC address.

Because each MAC address is unique, the link-local address will most likely be unique as well. Even so, it will be tested using Neighbour Solicitation and Neighbour Advertisement messages from ICMPv6.

By definition these addresses allow the communication between nodes within the same layer two network but not with nodes on other networks. Additionally, nodes can also register themselves in link-local multicast addresses (**`ff02::`**).

Once the link-local unicast address is settled, the node can use IPv6. It will start by requesting a router, this is achieved by sending an ICMPv6 type 133 message (*Router Solicitation*) to the multicast address **`ff02::2`** (local routers). Routers present on the network will reply with an ICMPv6 type 134 message (*Router Advertisement*) containing a list of network prefixes to use, and the way the node should configure its network interface, one of:

- **Stateless via SLAAC (Stateless Address Auto Configuration)**
- **Stateful via DHCPv6 (Dynamic Host Configuration Protocol for IPv6)**

SLAAC (Stateless Address Auto Configuration)

As instructed by the received **Router Advertisement** message, the node may use SLAAC, in this case the router provided prefix is used and the node bits will be generated in the same fashion as for the link-local unicast address. As well, duplicate node addresses existence will be tested.

The unicast address settled by SLAAC is a global scope address, and thus, it allows communication with IPv6 nodes on other networks, and ultimately on the internet.

SLAAC also informs nodes about the DNS servers they should use and the default DNS domain name (RFC 6106, IPv6 Router Advertisement Options for DNS Configuration), however, not all nodes support this feature. Even though using SLAAC, the **Router Advertisement** message can instruct the node to get additional settings from the DHCPv6 service, after setting the unicast address via SLAAC.

Other alternative is letting DHCPv6 set the unicast node address. If the **Router Advertisement** message demands **stateful configuration** the node will not use SLAAC, instead it will request all the configuration to a DHCPv6 server.

For stateful configuration, there must be a DHCPv6 server managing a pool of IPv6 addresses, pretty much like in DHCP for IPv4 addresses.

IPv6 – anycast and unicast addresses

Anycast addresses are allocated from the unicast addresses space, nevertheless, they are assigned to several nodes at the same time. When a packet is sent to an anycast address, routers will try to find the nearest node having that address and deliver to that node only.

One notorious case is the **subnet-router anycast address**, it's simply the network address (network prefix and node bits with value zero), a packet sent to this address will be delivered to the nearest router for that network.

The IPv6 address space is divided in addresses blocks (prefixes) allocated for different uses, accordingly to the value of the first bits (prefix) :

010 … - Unicast addresses assigned to service providers
100 … - Unicast addresses assigned to geographical areas
1111 1110 … (fd …) – IPv6 private use unicast addresses

Some special purposes reserved addresses are:

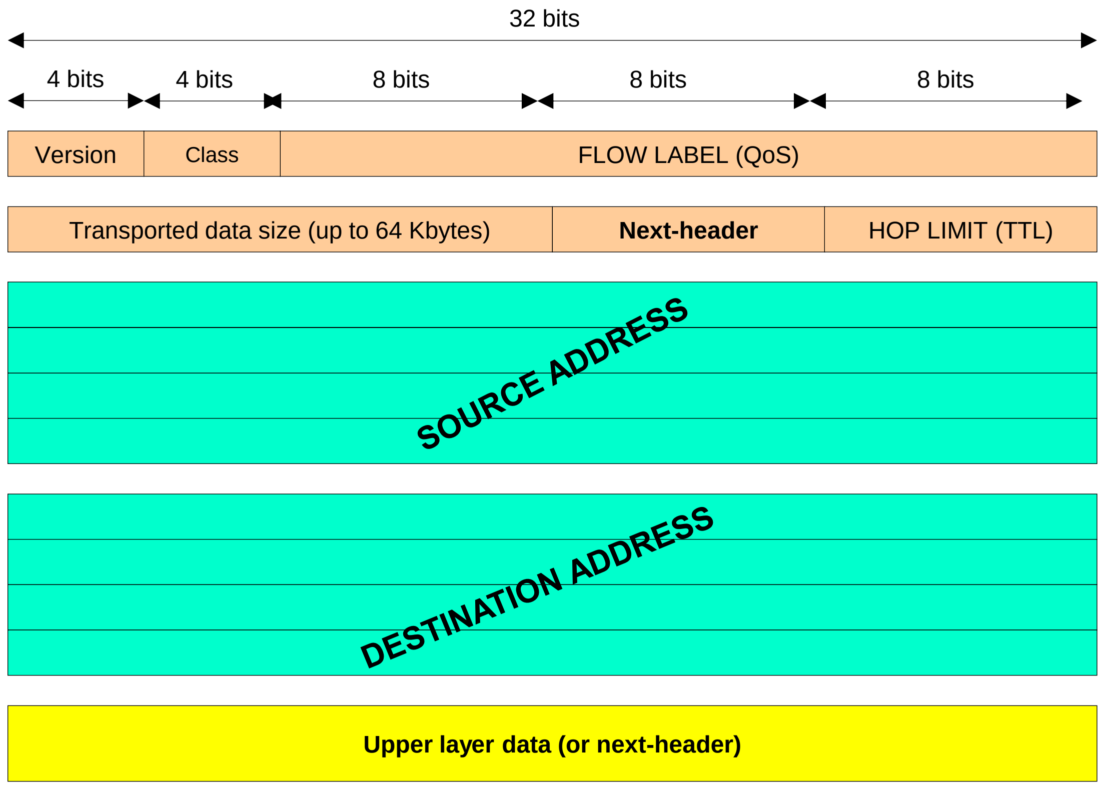
`::` (`0:0:0:0:0:0:0:0`), represents an unknown source address (pretty much like in IPv4).
`::1` (`0:0:0:0:0:0:0:1`), loopback interface address (equivalent to 127.0.0.1 in IPv4).

IPv6 datagrams

Although bigger than the IPv4 header, the IPv6 header has several simplifications, including the absence of the header checksum. The header has a **fixed length** of 40 bytes, this includes 32 bytes used for source and destination addresses. Some fields are equivalent to those on IPv4, but with different names, for instance the **protocol** field is now named **next-header**.

It can be highlighted the lack of fragmentation related fields, nevertheless, IPv6 supports fragmentation, but only between end nodes. That is implemented through an extension header.

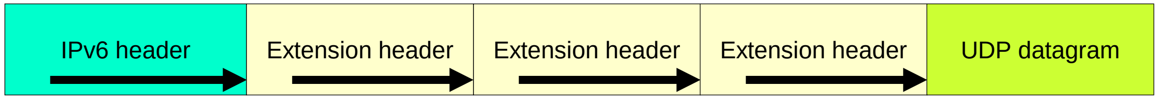
The second field (Class) is used to set different priorities in datagrams handling by routers, for instance, if it's real time critical, or if data recovery is relevant. Traffic class is related to FLOW LABEL that is used to identify a flow of datagrams belonging to the same flow with some settled QoS parameters.



IPv6 datagrams – extension headers

Unlike with IPv4, the IPv6 header has a fixed size, and no options, yet additional features are supported through the extension header concept. The next-header field is used to identify the payload type following the present header, typically ICMP, UDP or TCP. However, in IPv6, the next-header field can also identify an extension header.

All extension headers also have a next-header field, and a header length field. So, it may end up being a sequence of extension headers, usually terminated by data belonging to an upper-level protocol:



In addition to the standard upper-level protocols identifiers, such as 6 for TCP and 17 for UDP, the next-header can also hold other values used to identify extension headers. Examples:

0	HOP-BY-HOP OPTIONS – option to be handled by routers (e.g., JUMBO PAYLOAD)
43	ROUTING – routing options (e.g., SOURCE-ROUTING)
44	FRAGMENTATION – IPv6 fragmentation on intermediate nodes is not supported, only between end nodes
50/51	IPSEC – tunneling, authentication, confidentiality and integrity
60	DESTINATION OPTIONS – options to be handled only on the final node

ICMPv6 (ICMP for IPv6)

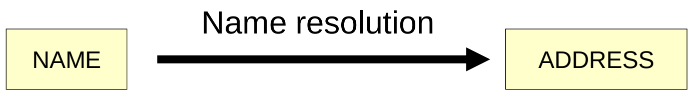
Although with similar purposes to ICMP used with IPv4 (ICMPv4), some differences exist, ICMPv6 is identified by protocol number 58 in the next-header field. The message format is very similar to ICMPv4:

TYPE (8 bits) + CODE (8 bits) + CHECKSUM (16 bits) + DATA (variable length)

Still, message type numbers are different, some most relevant are:

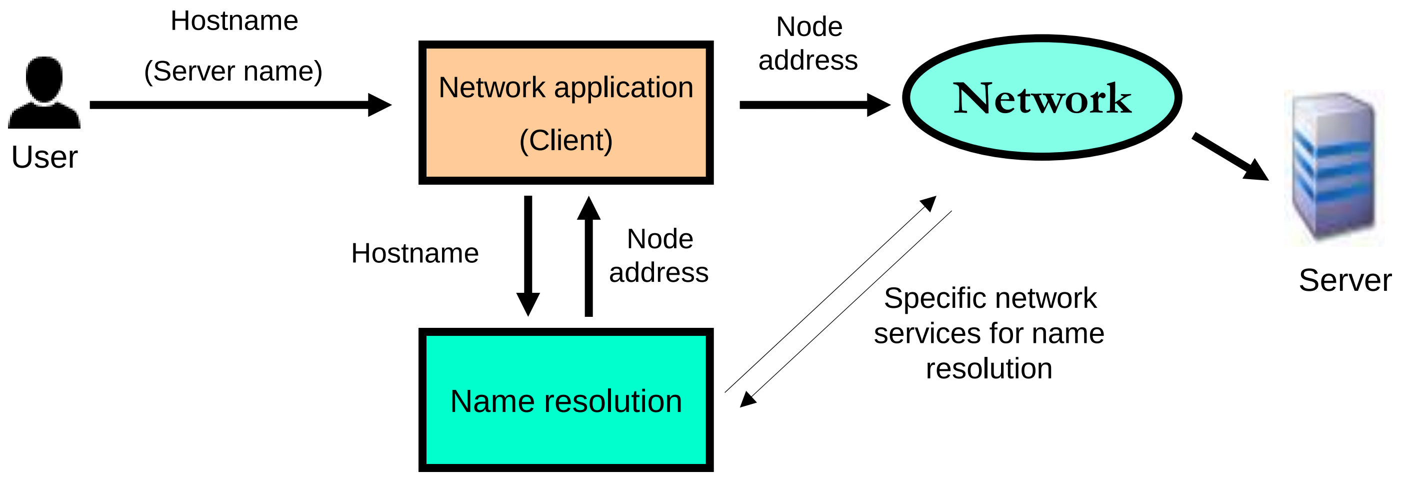
Type	Description
1	Destination unreachable – a datagram failed to be delivered, the code will represent the reason
2	Packet too big – the datagram won't fit on the next MTU, this message is used by PATH MTU DISCOVERY
3	Time exceeded (TTL) – like ICMPv4 (Code 0 = TTL exhausted; Code 1 = reassembly timeout)
128/129	Correspondingly ECHO request and ECHO reply. Like ICMPv4.
130	GROUP MEMBERSHIP QUERY – sent by routers to detect multicast groups members.
131	GROUP MEMBERSHIP REPORT – reply to the previous message sent by multicast group members
132	GROUP MEMBERSHIP REDUCTION – sent by nodes that wish to be removed from a multicast group.
133	ROUTER solicitation (RS) – sent to the all-routers multicast address to get a list of available routers.
134	ROUTER advertisement (RA) – reply to the previous message
135	Neighbor solicitation (NS) – used to identify a neighbor and get its MAC address (ARP request equivalent)
136	Neighbor advertisement (NA) – reply to the previous request (ARP reply equivalent).
…	…

Names resolution



Handling node addresses is neither suitable nor pleasant for end-users, but ultimately applications need them to identify other applications within the network infrastructure. Of course, this issue becomes even worst in IPv6 with 128 bits addresses.

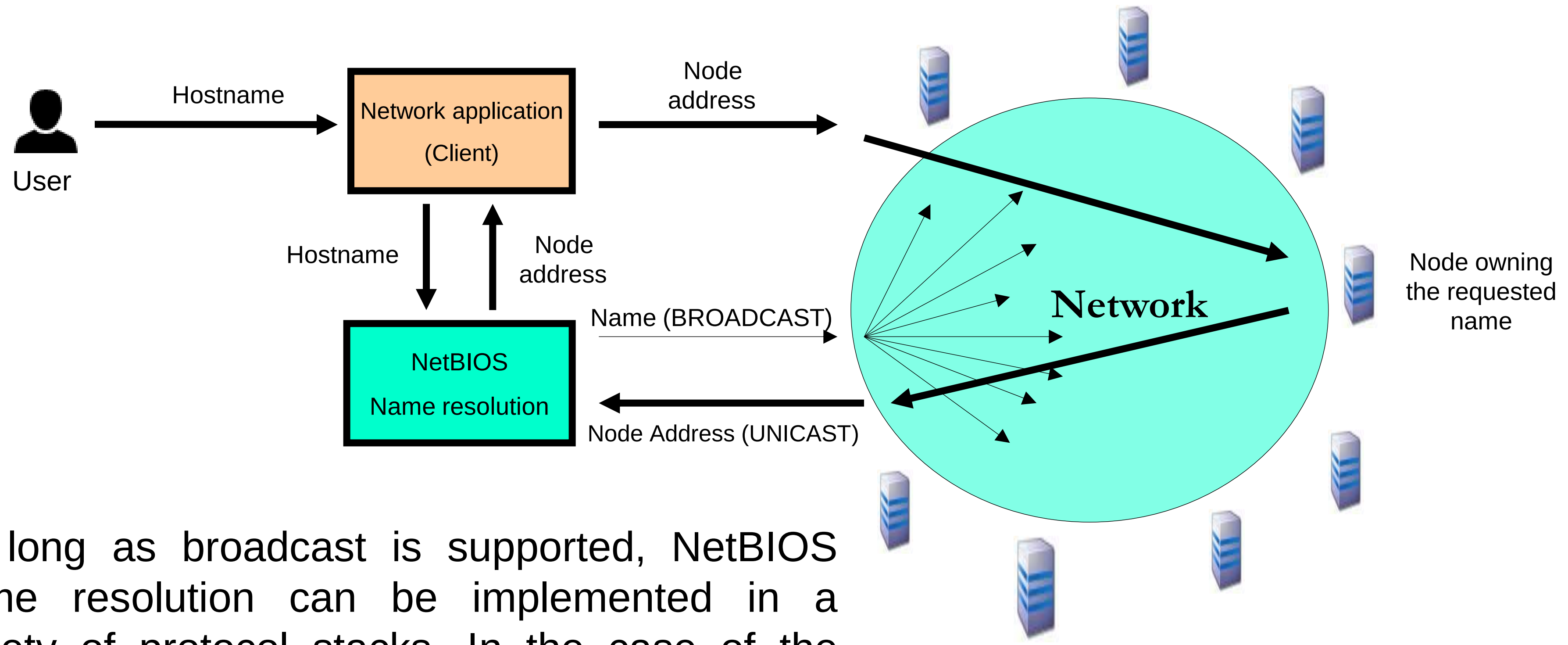
The name resolution aims at protecting users from node addresses handling and provides a hosts names interface. Instead of node addresses, nodes can then be referred to by rather more friendly host names.



In simple terms, the names resolution system receives a node name and returns the corresponding node address.

NetBIOS name resolution

NetBIOS was a very successful name resolution system. It was introduced when many local networks were using a peer-to-peer model, those networks had only workstations and no dedicated servers. NetBIOS operates by sending a broadcast request (NAME QUERY) with the name to be resolved, the owner of the requested name should then reply to the requester providing its own address.



As long as broadcast is supported, NetBIOS name resolution can be implemented in a variety of protocol stacks. In the case of the TCP/IP stack, UDP on port number 137 is used.

NetBIOS name registration

NetBIOS name resolution was designed for peer-to-peer networks, with no servers, thus, each node has the obligation to reply on queries about its own name.

Nevertheless, nodes names must be unique, to ensure that, when a node starts up, it must send name registration requests to the broadcast address. During this registration procedure, any node using the same name should report a duplicate name error, otherwise, the name is assumed to be available and will be used.

The absence of duplicate name errors during the name registration is a poor proceeding: if a node, temporarily, losses the network connection during other node's registration, it has no opportunity to say it's already using the name.

As well, before powering off, a NetBIOS node should announce in broadcast its used name is being released (NAME RELEASE).

Broadcast-based NetBIOS has severe security flaws because it's based on an expected good behaviour from all network nodes.

NetBIOS name types (Microsoft)

NetBIOS names can be unique or group names. Unlike with unique names, the same group name may be register by several nodes without conflict.
Group names are used to create logical nodes groups, in Microsoft networks, group names are used to implement the Microsoft workgroup and domain concepts.
In the original NetBIOS specification, names could be up to 16 characters long, in Microsoft networks the 16th character is used to identify the name type, so names can be only up to 15 characters long.
Usually the name type is represented in hexadecimal, separated from the name by a hash. For instance, SERVP1#20 represents the name SERVP1 of type 20 (hexadecimal). Some important name types in Microsoft windows networks are:

00	General use unique name
01	Associated to the special name "__MSBROWSE__" identifies the local browse list collector.
03	Logged in username (old clients) and also node name.
1B	Associated to a domain name identifies the PDC (Primary Domain Controller).
1C	Associated to a domain name identifies a login server for the domain.
1D	Associated to a domain name identifies the local representative for the domain.
1E	Associated to a domain or workgroup name, all members will have this name.
20	File Server

Each NetBIOS node will have several names, for instance:

```
MYSERVER#00
MYSERVER#03
MYSERVER#20
__MSBROWSE__#01
MYDOMAIN#1B
MYDOMAIN#1C
MYDOMAIN#1D
MYDOMAIN#1E
HST222#00
HST222#03
HST222#20
```

WINS – NetBIOS name server

Several issues impact on broadcast-based NetBIOS:

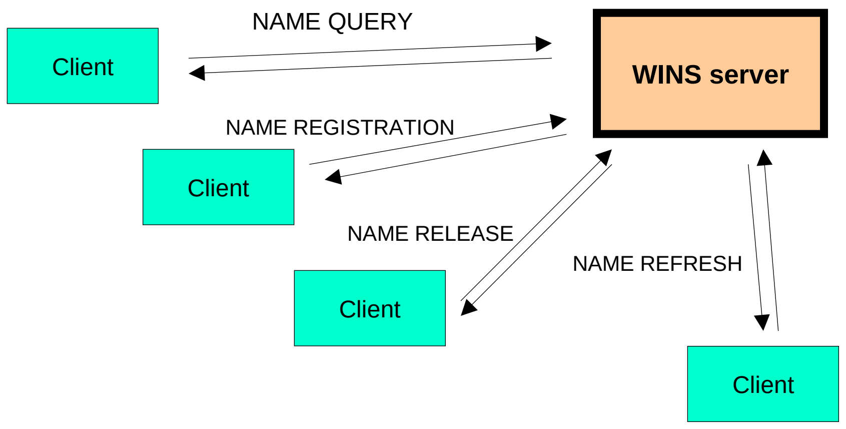
- It's unsafe and unreliable (name registration/query trusts on every node's good behavior).
- Broadcast propagation is limited to the layer two network broadcast domain (all nodes must be within the same layer two network).
- Broadcast traffic, is bad for network performance because it bypasses layer two switches segmentation.

To solve these issues Microsoft has introduced the WINS (**Windows Internet Name Service**) concept, in fact, there are very few changes to the original broadcast method. The main difference is dropping the use of broadcast and replacing it by a dedicated service.

All requests are now sent to the **unicast address of the WINS server**; thus, the WINS server can be on a remote network and serve clients from several different networks.

Beyond that, the transition to a client/server model adds security and reliability. Now, all requests will have an explicit reply.

In WINS, name registration also has a associated time to live, when the lease time expires records are removed if not refreshed.



NetBIOS name resolution future perspectives

NetBIOS name resolution was dragged to current days by Microsoft operating systems, starting with Windows for Workgroups. WINS servers replaced broadcast-based NetBIOS and solved most issues. However, one design issue remains, the name structure is shallow, without any hierarchy.

Although efficient, NetBIOS is limited to local environments. It can’t be widely escalated, because names would have to be globally unique, and that would require a single administration point. This makes it impossible to handle a huge number of nodes spread around the world.

The Domain Names System (DNS) solves this scalability problem by setting domain names and giving administrative autonomy to each domain.

Another lesson learned from computer networks history is that two systems that do same thing will not coexist for a long time. The same way the TCP/IP protocol stack made all others disappear; we can expect the DNS system used over the internet to totally replace NetBIOS.

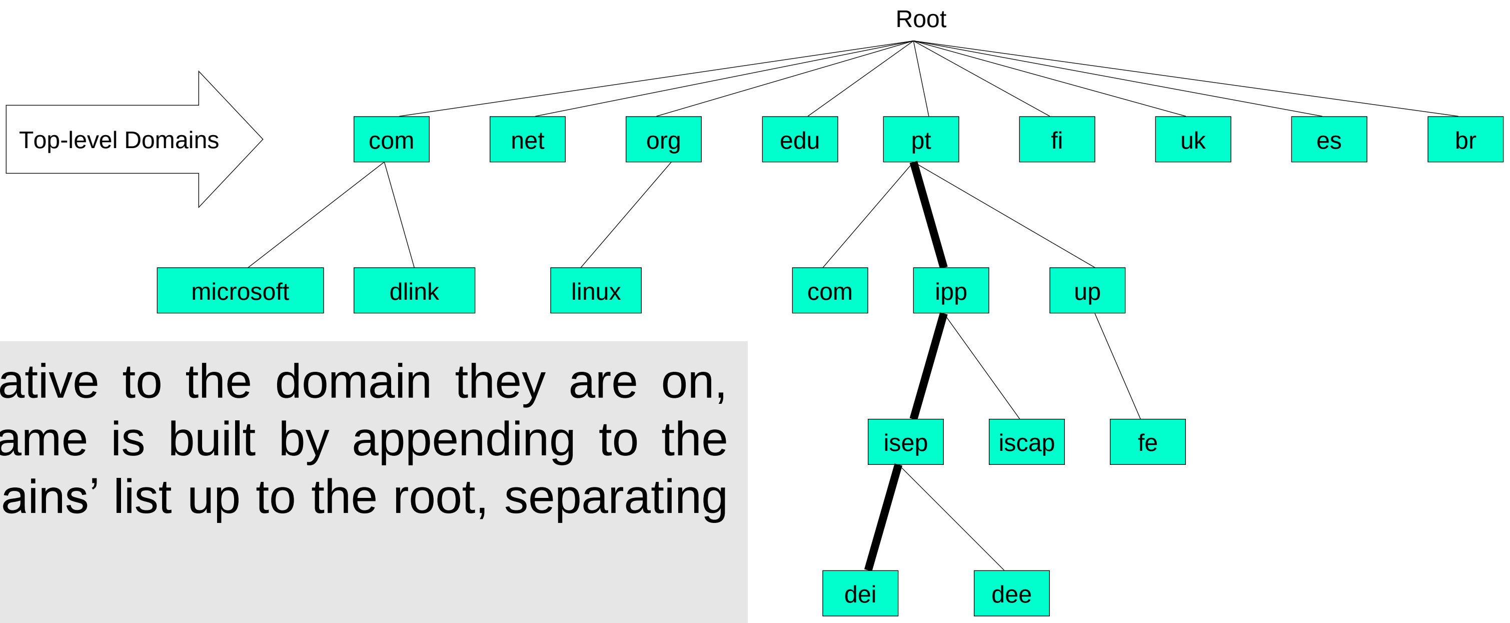
Recent MS-Windows clients, with Active Directory support, are already capable of operating without NetBIOS, and use DNS only. For now, WINS and DNS still coexist on some MS-Windows clients, some WINS servers can even be configured to use DNS when they can’t find a name on NetBIOS, acting as application gateways.

Windows clients may combine several query techniques: broadcast NetBIOS, WINS, and DNS, sometimes creating some confusion.

DNS – Domain Name System

DNS has the major advantage of being based on a hierarchal tree structure where each branch is administratively independent of other branches. Each branch is called a domain and has a name.

Domains are independent because a DNS name is significant only within a domain. To globally identify a host's name, the domain must also be specified, this is the qualified host name. Domains themselves are identified by names, but again to globally identify a domain name, the domain it belongs to must also be specified.

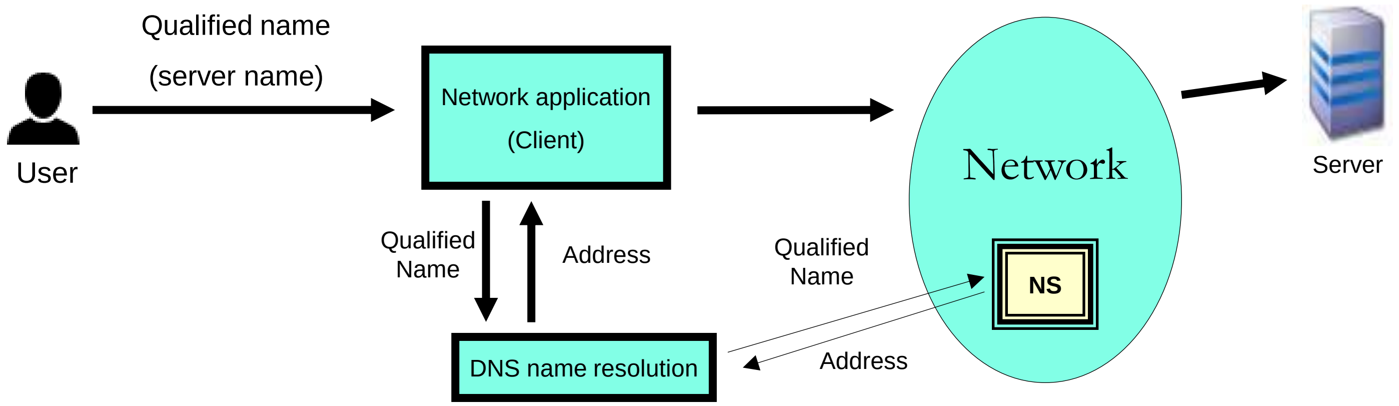


Names are relative to the domain they are on, the qualified name is built by appending to the name, the domains' list up to the root, separating each by a dot.

For instance: **www.dei.isep.ipp.pt**

DNS servers

The logical DNS tree structure is sustained by name servers (NS). Name servers communicate with each others, in such a way, any of them will be able to resolve any qualified name in whatever domain.
So, for a client, knowing one name server (NS) address is enough to be able to resolve any name.



Each name server maintains a local database with records of all names for its own domain. In fact, a single name server can maintain databases for several domains.

If a name server has a local copy of the names database for a domain, it's an authoritative DNS server for that domain. This means it does not depend on other name servers to resolve than domain's names.

To resolve other domains' names, a DNS server must query other name servers.

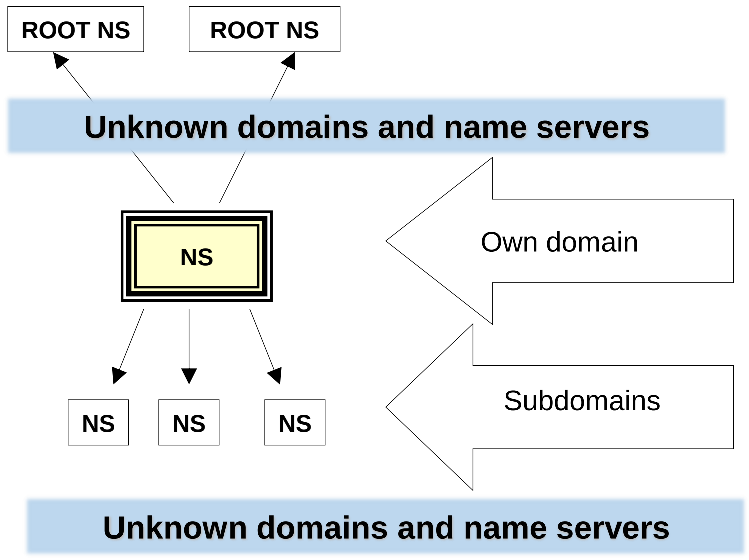
DNS – Name servers' logical infrastructure

For the DNS system to work as a whole, name servers must be linked together by mutual awareness. Though, each name server is not required to be directly aware of all other name servers. **Each only needs to know:**

- all records (names) of its own domain (may be a copy from the master server).
- the addresses of the root name servers (above the top-level domains).
- the addresses of the name servers of all its subdomains (immediately below).

By enforcing these rules every domain will be reachable for any name server.

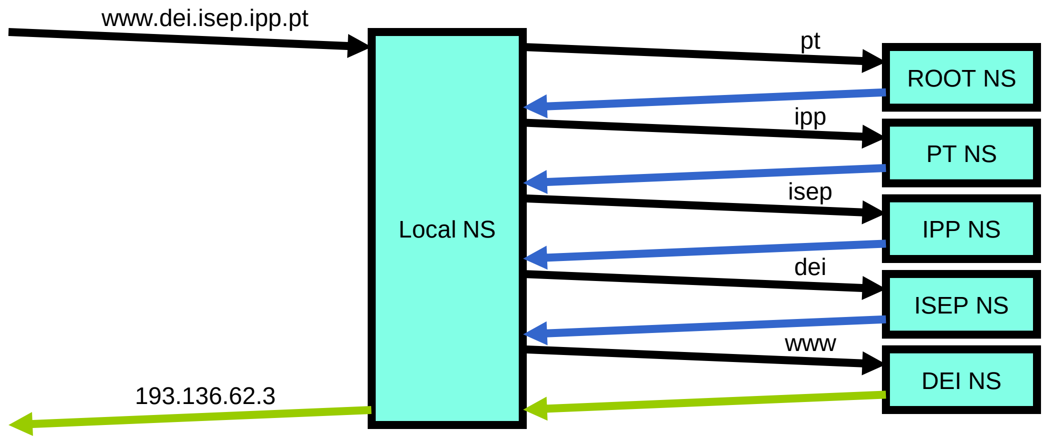
Root name servers are a guaranteed starting point to resolve any name, this is because they, as well, must know all their subdomains' name servers. For the case of root name servers, their subdomains are also known as top-level domains.


ROOT NS
ROOT NS
Unknown domains and name servers
NS
Own domain
Subdomains
NS
NS
NS
Unknown domains and name servers

DNS name resolution

Name resolution works step by step, starting from the root. For instance, if somewhere on the internet an application queries for the **www.dei.isep.ipp.pt** host address, it will contact a local name server (whose address it must know). The local NS will then query a root NS for a pt NS, which the root name server must know because it's a top-level domain. The process will be repeated until the last domain NS is reached (dei.isep.ipp.pt):

NS records contain host names, not host's addresses, so a query for a domain's NS returns a host name.
If that host name belongs to the domain itself, then there's a deadlock due to a circular dependency.
To solve this issue, a record with the NS address is added on the parent domain. These records are known as **glue records**.



The image above represents a recursive query. Recursive means the local name server is doing all the work for the client. However, a name server may refuse the recursive query and require the client to do the job, providing to it the required information, namely addresses of name servers. For the sake of efficiency, name servers hold in cache query replies they get, for that purpose each record has an attached time to live (TTL). The TTL value is settled for each record by the domain's administrator. DNS caching significantly reduces the number of direct queries to name servers, specially for top-level domain and root name servers. The bigger the TTL value, the less is the number of queries we could expect the name server will have to handle.

DNS records (Resource Records)

A DNS database holds several record types with different purposes, they are called Resource Records. Every resource record (RR) has the same general format:

NAME (up to 255 characters)	- The qualified name (record owner), a null terminated string.
TYPE	- 16 bits number identifying the record type (RR TYPE)
CLASS	- 16 bits number identifying the address family class (RR CLASS), for IP the class is 1 (IN class)
TTL	- 32 bits number representing the record's TTL in seconds
RDLENGTH	- 16 bits number defining RDATA length in bytes
RDATA	- The record's content (RDATA), the content depends on RR TYPE

DNS records are stored at name servers and are delivered to clients or other name servers when requested (DNS QUERIES). Name servers receive query requests in DNS protocol format on port 53, usually transported by UDP datagrams.
Each query will have one or more requests in the form:

NAME (up to 255 characters)	**TYPE**	**CLASS**

The TYPE field in the request may be an RR TYPE value or special values. Value 255 (*) represents **any type**, value 252 (AXFR) represents all RR records in the domain (the whole database).

AXFR queries are mostly used for database synchronization between masters and slaves. Due to the high volume of data involved, in this case, a TCP connection may be used instead, though the service port number is still 53 as for the UDP service.

Major resource record types

Type (RR TYPE)	Type mnemonic – Purpose	NAME content	RDATA content
1	A - An IPv4 node address	A qualified node name	An IPv4 address
2	NS – A name server	A qualified subdomain name	A qualified host name (of a name server)
5	CNAME – An alternative name (alias)	A qualified alternative node name (alias)	A qualified node name (A or AAAA)
6	SOA (Start Of Authority) – domain settings	A qualified domain name	Several data, including the serial number (see below)
12	PTR – Name matching an IPv4 address	Address name in IN-ADDR.ARPA	A qualified node name
15	MX – Domain mail hubs	A qualified domain name	Priority and qualified node name
16	TXT – A comment	A qualified domain or node name	Free text
28	AAAA - An IPv6 node address	A qualified node name	An IPv6 address
29	LOC – A geographical localization	A qualified domain name	Geographical coordinates and altitude
33	SRV – A network service	_service._protocol.qualified-domain-name	Priority, weight, port number, a qualified node name

Sample SOA record in BIND 9 zone configuration file on Linux

```
@ 99999999 SOA picasso.dei.isep.ipp.pt. root.picasso.dei.isep.ipp.pt. (
                              2008042402      ; serial
                              28800           ; refresh (8 hours)
                              7200            ; retry (2 hours)
                              604800          ; expire (7 days)
                              86400           ; minimum (1 day)
             )
```

@ represents the zone domain name this record refers to, in this case “dei.isep.ipp.pt”.

(“$ORIGIN” directive in BIND)

TTL – numerical value in seconds

If absent, $TTL directive value is used

The class is absent

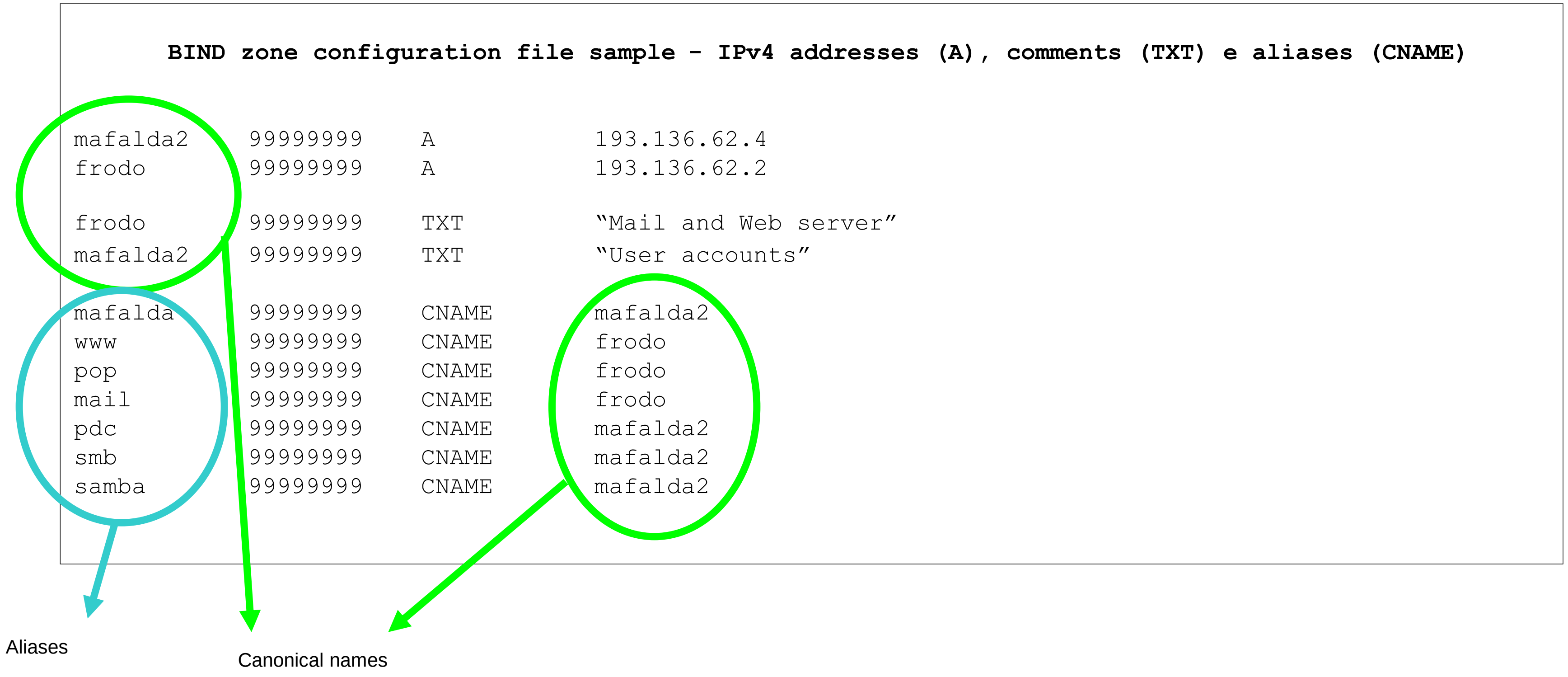
By default is “IN”.

Primary (master) name server for the domain

Serial number for slave synchronization.

(holds the date and a serial number within the day)

DNS RR – names and aliases



```
      BIND zone configuration file sample - IPv4 addresses (A), comments (TXT) e aliases (CNAME)

mafalda2    99999999    A           193.136.62.4
frodo       99999999    A           193.136.62.2

frodo       99999999    TXT         "Mail and Web server"
mafalda2    99999999    TXT         "User accounts"

mafalda     99999999    CNAME       mafalda2
www         99999999    CNAME       frodo
pop         99999999    CNAME       frodo
mail        99999999    CNAME       frodo
pdc         99999999    CNAME       mafalda2
smb         99999999    CNAME       mafalda2
samba       99999999    CNAME       mafalda2
```

Because this sample belongs to the **dei.isep.ipp.pt.** zone definition (origin), all unqualified names (that are not dot terminated) are implicitly qualified with **dei.isep.ipp.pt.**

For instance, www.dei.isep.ipp.pt name will end up resolved to the 193.136.62.2 address.

$$\begin{tikzcd}
\boxed{\textbf{www}} \arrow[r, "\text{CNAME}"'] & \boxed{\textbf{frodo}} \arrow[r, "\text{A}"'] & \boxed{193.136.62.2}
\end{tikzcd}$$

DNS RR – NS and glue records

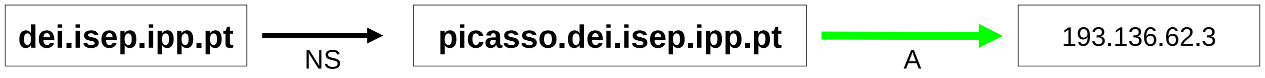
NS records are critical because they make the DNS system work as a whole, they are the ones that link domains to its subdomains. As stated before, each domain must know its subdomains' name servers.

Sample BIND configuration for the dei.isep.ipp.pt subdomain in the isep.ipp.pt domain

```
$ORIGIN isep.ipp.pt
@    IN SOA nsrv1.isep.ipp.pt. admin.isep.ipp.pt. 2007120500 2h 15M 3W12h 2h20M

@            IN          NS          nsrv1.isep.ipp.pt.
@            IN          NS          nsrv2.isep.ipp.pt.

nsrv1        IN          A           193.136.6.40
nsrv2        IN          A           193.136.6.47


dei.isep.ipp.pt.                     IN          NS          picasso.dei.isep.ipp.pt.
dei.isep.ipp.pt.                     IN          NS          slave.dei.isep.ipp.pt.

picasso.dei.isep.ipp.pt.             IN          A           193.136.62.3
slave.dei.isep.ipp.pt.               IN          A           193.136.62.110
```

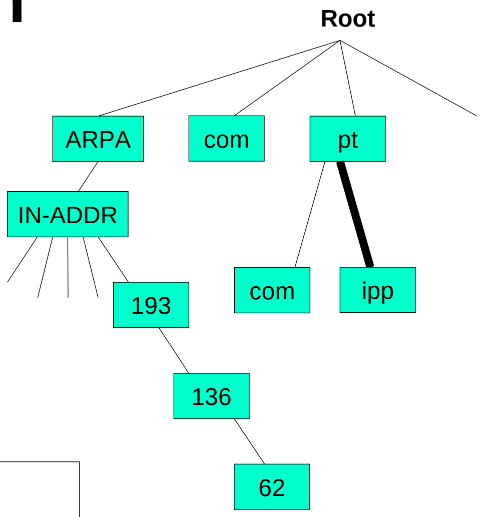
NS records define a qualified host name of a name server

Glue records (highlighted in green) are A (or AAAA) records that do not belong to the domain, but without them, queries about subdomain name servers could never get an address for the name server.


$$\begin{tikzcd}\boxed{\textbf{dei.isep.ipp.pt}} \arrow[r, "NS"'] & \boxed{\textbf{picasso.dei.isep.ipp.pt}} \arrow[r, "A"'] & \boxed{193.136.62.3}\end{tikzcd}$$

Without the glue record, the last resolution would be impossible as picasso.dei.isep.ipp.pt A record would be stored on the dei.isep.ipp.pt subdomain itself, and that's the domain we want to reach in the first place.

DNS RR – PTR and the in-addr.arpa. domain



The special domain **in-addr.arpa.** contains a structure of records for IPv4 addresses. Each subdomain level corresponds to a byte in the IPv4 address, the left byte is the upper-level domain.

Records in **in-addr.arpa.** are used for reverse resolution, this means getting the hostname given an IPv4 address. PTR records define the qualified host names for each address.

Sample "in-addr.arpa." – BIND zone configuration

```
2.62.136.193.in-addr.arpa.          IN          PTR          frodo.dei.isep.ipp.pt.
3.62.136.193.in-addr.arpa.          IN          PTR          picasso.dei.isep.ipp.pt.
4.62.136.193.in-addr.arpa.          IN          PTR          mafalda2.dei.isep.ipp.pt.
```

Sample "in-addr.arpa." with "$ORIGIN" directive – BIND zone configuration

```
$ORIGIN 62.136.193.in-addr.arpa

2    IN      PTR          frodo.dei.isep.ipp.pt.
3    IN      PTR          picasso.dei.isep.ipp.pt.
4    IN      PTR          mafalda2.dei.isep.ipp.pt.
5    IN      PTR          srv1.dei.isep.ipp.pt.
6    IN      PTR          srv2.dei.isep.ipp.pt.
```

PTR records are also used for IPv6, then the domain is **ipv6.arpa.** and each subdomain matches a single hexadecimal digit.
Of course, PTR records should be consistent with A and AAAA records.

DNS and electronic mail

DNS domain names are used by electronic mail (SMTP) to identify mailboxes over the internet in the form **mailbox@domain-name**.

In order to deliver mail messages to a mailbox in a domain, SMTP (Simple Mail Transfer Protocol) needs to identify and contact one mail server for that domain.

One older solution requires the definition of an A or AAAA record in the parent domain associated to the domain name representing the IP address of the mail server:

Sample A record for dei.isep.ipp.pt domain – BIND zone configuration in isep.ipp.pt domain

```
$ORIGIN isep.ipp.pt
@    IN SOA nsrv1.isep.ipp.pt admin.isep.ipp.pt 2007120500 2h 15M 3W12h 2h20M

dei.isep.ipp.pt         IN          A          193.136.62.2
dee.isep.ipp.pt         IN          A          193.136.63.3
```

Subdomains mail servers

(MTA – Mail Transfer Agent ; Mail server; Mail Exchanger)

If the recipient of a mail message is username@dei.isep.ipp.pt, the A record for dei.isep.ipp.pt is queried and 193.136.62.2 is returned. Next, SMTP protocol will be used to send the mail message to node 193.136.62.2, as that's the mail server for the domain.

DNS RR – MX records

MX (Mail eXchanger) records are more appropriate to identify mail servers in a domain. With MX records several mail servers with different priority levels can be established for a domain, also, they can be defined on the domain itself without involving the parent domain.
One domain can have several MX records, each defines a qualified host name and a 16 bits priority level (a lower priority level mean a higher preference).

MX records sample - BIND zone configuration

```
dei.isep.ipp.pt.                  IN          MX          10          frodo.dei.isep.ipp.pt.
dei.isep.ipp.pt.                  IN          MX          20          picasso.dei.isep.ipp.pt.

picasso.dei.isep.ipp.pt.          IN          A           193.136.62.3
frodo.dei.isep.ipp.pt.            IN          A           193.136.62.110
```

By using the MX records in the example, applications sending SMTP mail to mailboxes at dei.isep.ipp.pt will get a list of two servers, frodo.dei.isep.ipp.pt. would be tried first, if not available, then picasso.dei.isep.ipp.pt. would be the next try.

If there are several mail servers with the same priority level, the client will usually pick the first. Therefore, BIND name servers use round-robin to sort the list, and thus, distribute clients.

DNS RR – SRV records

SRV records (RFC 7282) are to some extent a generalisation for MX records. They allow clients to identify which hosts provide given services on a domain. They are therefore suitable for servers to announce their services, is used for instance by Microsoft Active Directory servers.

SRV records are symbolic names representing the type of network service provided (application protocol) in the form :

_SERVICE._PROTOCOL.DOMAIN-NAME

The underline character is supposed to avoid conflicts with other normal domain names.
SERVICE is a standard identifier for the service (application protocol).
PROTOCOL identifies the transport protocol to be used (either udp or tcp).

The SRV record data contains a priority level similar to MX records and a weight level to be used when the priority is the same, next the service port number is specified followed by the canonical qualified host name providing the service.

```
                    SRV records sample - BIND zone configuration

_ldap._tcp.dei.isep.ipp.pt.        IN  SRV     5 4 389     mafalda2.dei.isep.ipp.pt.
_ldap._tcp.dei.isep.ipp.pt.        IN  SRV     5 6 389     frodo.dei.isep.ipp.pt.
_ldap._tcp.dei.isep.ipp.pt.        IN  SRV     20 20 389   picasso.dei.isep.ipp.pt.
```

Looking at the above example, when an LDAP client wants to find a service provider for dei.isep.ipp.pt domain, it will query for the SRV record with name **_ldap._tcp.dei.isep.ipp.pt** and receives three replies. From those the lower priority (in the case 5) ones are selected, among those two, it will consider weights (4 and 6), so it will use the first with a 40% probability and the second with a 60% probability.

DNS RR – SPF – Sender Policy Framework

RFC 4408 (Sender Policy Framework for Authorizing Use of Domains in E-Mail, Version 1) establishes a method to set a domain policies over which nodes are authorized to send mail on the behalf of that domain.
When an SMTP server receives a mail message, it should check the sender's domain (From field) to see if the client node (source address) is authorized.

SPF records contain a formatted string defining the policy, a simple TXT record can also be used for the same purpose:

```
                          SPF records sample – BIND zone configuration

dei.isep.ipp.pt         IN  SPF     “v=spf1 +mx -all”
dei.isep.ipp.pt         IN  TXT     “v=spf1 +mx -all”
```

The more recent RFC 7208 definitely discontinued the SPF records keeping only the use of TXT records.

The SPF string settles which nodes are authorized to send mail on the behalf of the domain, it starts by identify the version (v=spf1), then defines which nodes are authorized (+) and which are not (-). In the above example only the MX servers of the dei.isep.ipp.pt. domain are allowed.

A wide variety of options are supported for the SPF string format (RFC 4408) .

DNS RR – LOC record

LOC record defines a geographical localization, so it’s usually associated to the domain name.

The LOC record contains:

- Latitude in degrees, minutes and seconds
- Longitude in degrees, minutes and seconds
- Altitude above the see level in meters.
- The size in meters (the diameter of a sphere that contains the local)
- Horizontal and vertical precision

LOC sample record – Bind zone configuration

```
dei.isep.ipp.pt         IN  LOC  41 10 39.782 N 8 36 28.578 W 50.00m 100m 10m 10m
```

This record defines the geographical localization for dei.isep.ipp.pt domain as being latitude = 41º 10’ 39,782” North ; longitude = 8º 36’ 28,578” West ; altitude = 50 meters; size = 100 meters; horizontal precision = 10 meters e vertical precision = 10 meters.

Some examples of LOC record queries

```
-bash-3.00$ host -t LOC yahoo.com
yahoo.com location 37 23 30.900 N 121 59 19.000 W 7.00m 100m 100m 2m
-bash-3.00$ host -t LOC ckdhr.com
ckdhr.com location 42 21 43.528 N 71 5 6.284 W -25.00m 1m 3000m 10m
-bash-3.00$ host -t LOC dei.isep.ipp.pt
dei.isep.ipp.pt location 41 10 39.782 N 8 36 28.578 W 50.00m 100m 100m 10m
```

Dynamic DNS (DNS)

DNS was originally designed to manage fairly static data. Changes to DNS records were supposed to be manual operations performed by systems administrators. In fact, the original DNS protocol (RFC 1035) defines no update operation.

RFC 2136 adds the update request to the DNS protocol, allowing dynamic updates. In addition, RFC 3645 defines the Generic Security Service Algorithm for Secret Key Transaction Authentication for DNS (GSS-TSIG).

With dynamic DNS, servers, and generally speaking any node, may have a variable IP node address, and yet, a fixed DNS name. Whenever their IP address changes, they request a DNS update. Dynamic DNS is also used to announce available services, for instance, Microsoft Active Directory servers use SRV records to announce service to the domain members.

One thing to bear in mind when deploying dynamic DNS, is caching and the TTL value. DNS servers and clients will store local copies of received records and keep regarding them as up to date during the corresponding TTL period. Thus, records that change frequently should have low TTL values.

Less standard techniques for DNS update include the use of HTTP services and others, they have nothing to do with the DNS protocol, it’s just a matter of changing records and reloading the DNS server. One well-known client is **ddclient**.