RIPE NCC
RIPE NETWORK COORDINATION CENTRE

IPv4-mapped IPv6 addresses

Ondřej Caletka | 23 May 2024 | RIPE 88

What is an IPv4-mapped IPv6 address?

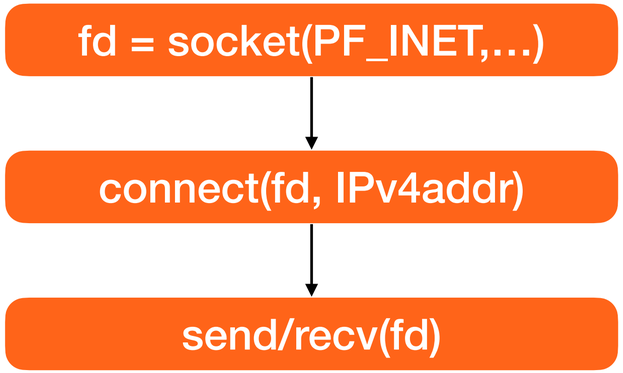
`::ffff:192.0.2.1`

- IPv6 address **like any other**
- Constant prefix `::ffff:0:0/96` + IPv4 address
- Used for **IPv4 compatibility** in IPv6 socket API

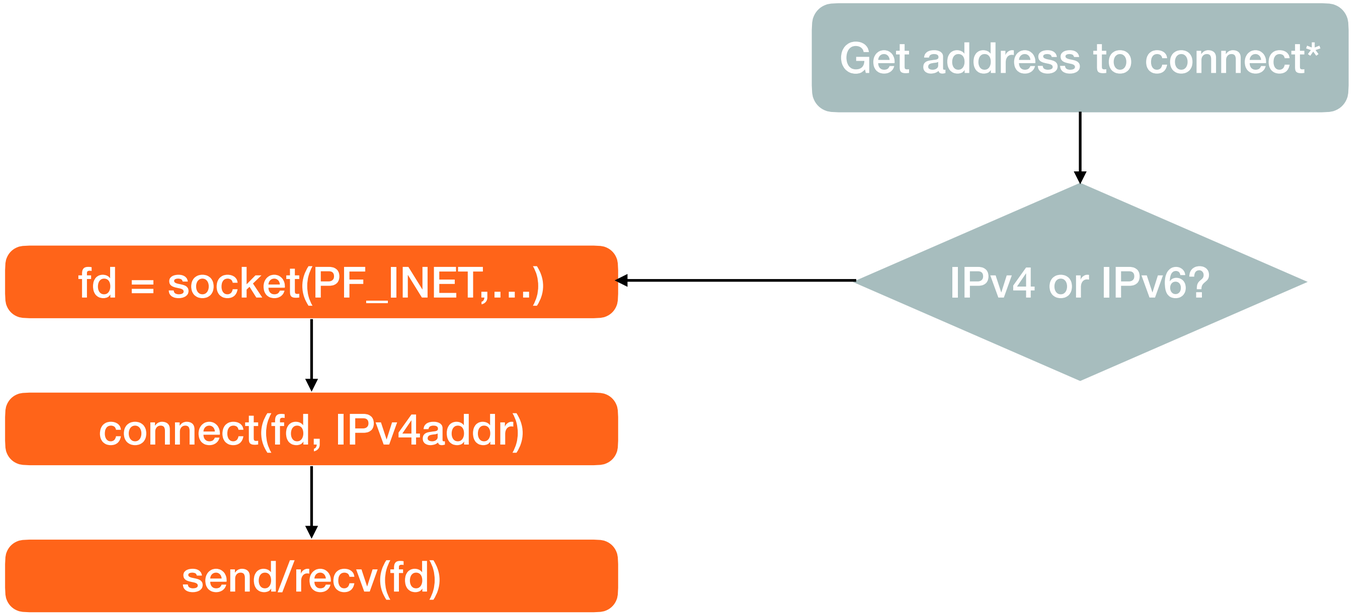
Socket API

How to program for both
IPv4 and IPv6

Client

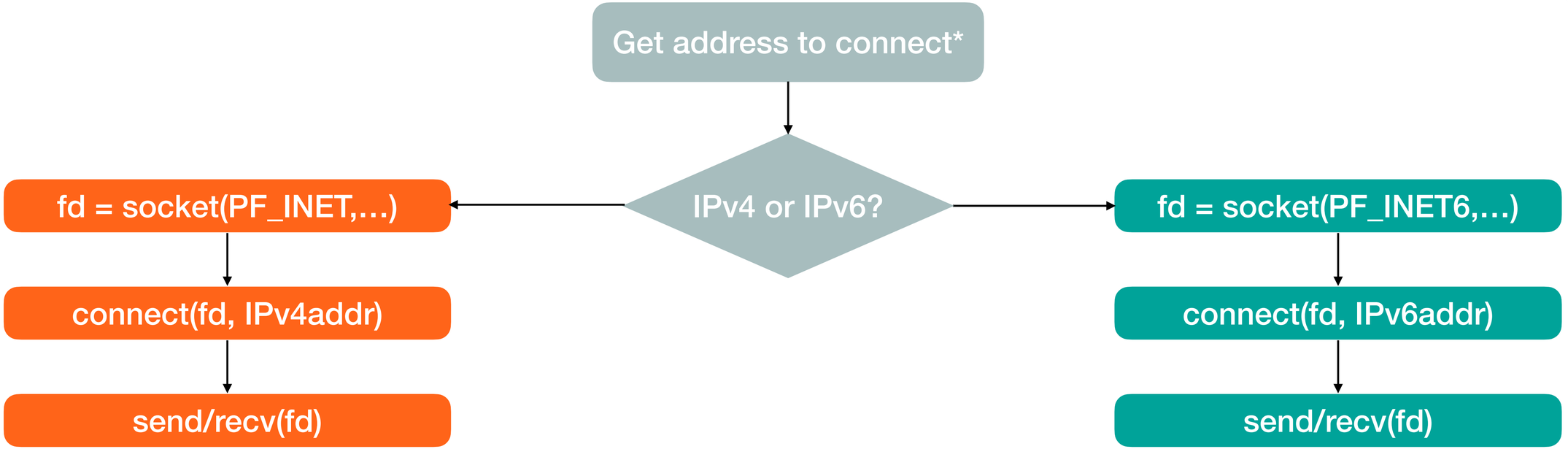


Client



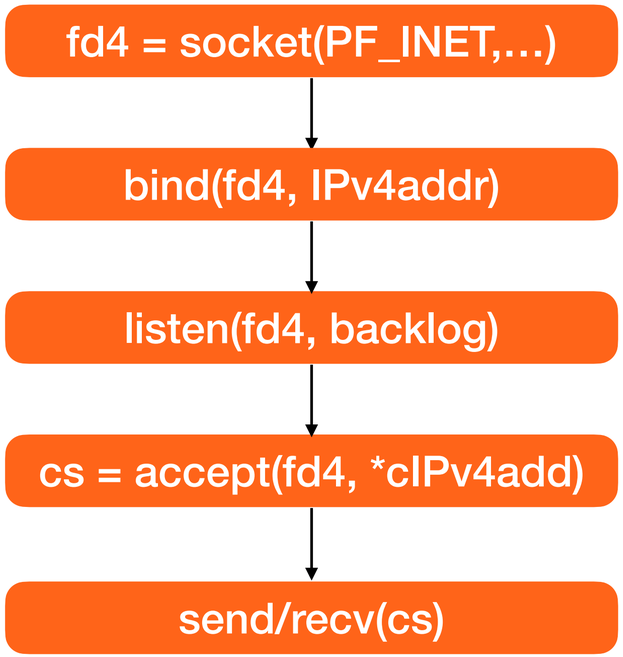
*for instance, by means of `getaddrinfo(3)`

Client

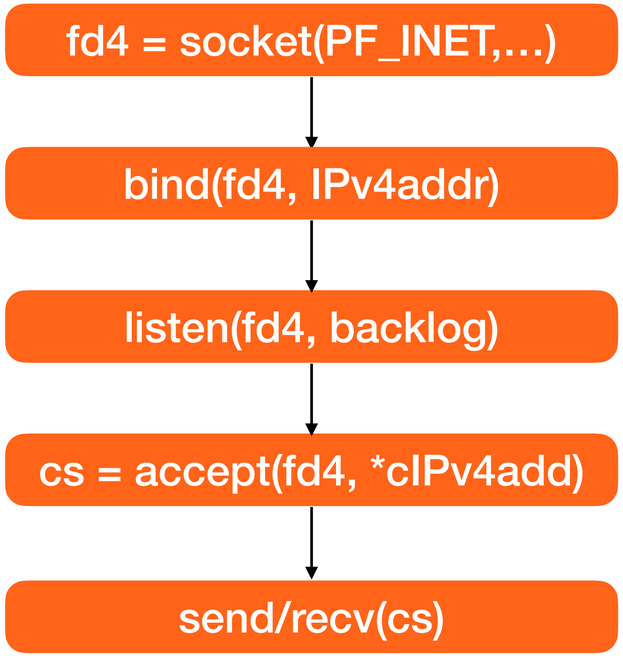


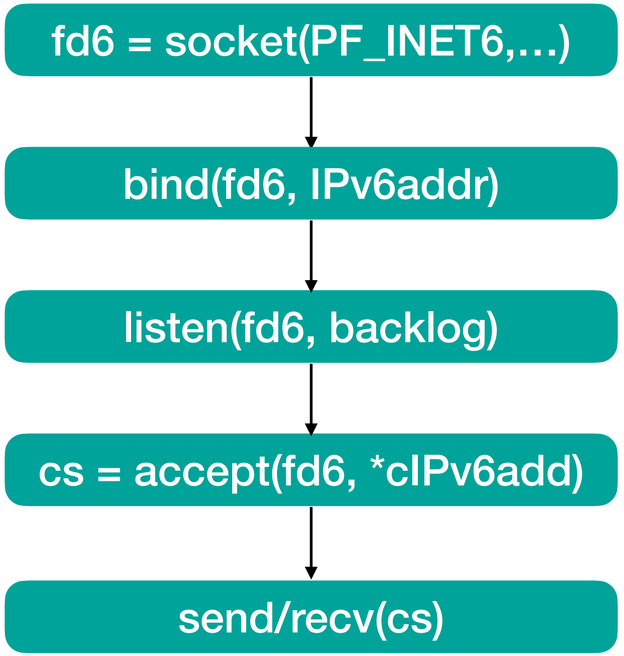
***for instance, by means of `getaddrinfo(3)`**

Server

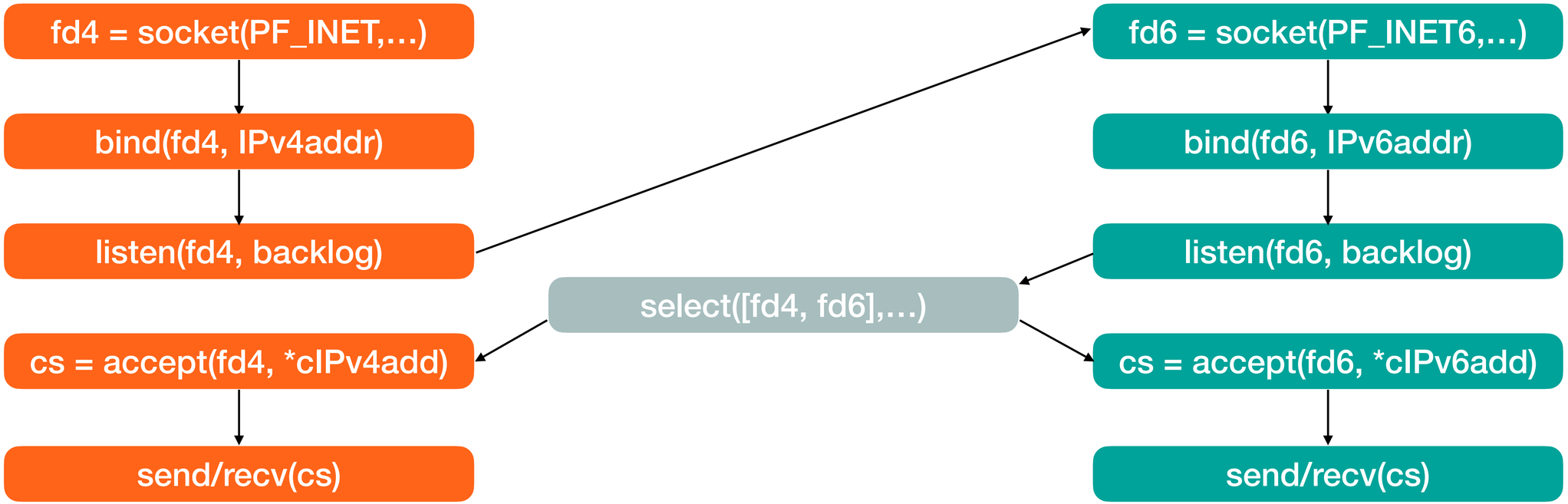

fd4 = socket(PF_INET,…)
bind(fd4, IPv4addr)
listen(fd4, backlog)
cs = accept(fd4, *cIPv4add)
send/recv(cs)

Server


fd4 = socket(PF_INET,…)
bind(fd4, IPv4addr)
listen(fd4, backlog)
cs = accept(fd4, *cIPv4add)
send/recv(cs)


fd6 = socket(PF_INET6,…)
bind(fd6, IPv6addr)
listen(fd6, backlog)
cs = accept(fd6, *cIPv6add)
send/recv(cs)

Server


fd4 = socket(PF_INET,…)
bind(fd4, IPv4addr)
listen(fd4, backlog)
cs = accept(fd4, *cIPv4add)
send/recv(cs)
fd6 = socket(PF_INET6,…)
bind(fd6, IPv6addr)
listen(fd6, backlog)
select([fd4, fd6],…)
cs = accept(fd6, *cIPv6add)
send/recv(cs)

IPv4 compatibility of IPv6 sockets

- IPv6 sockets **might be able** to receive IPv4 traffic as well

IPv4 compatibility of IPv6 sockets

- IPv6 sockets **might be able** to receive IPv4 traffic as well
- This is supposed to make the **server applications simpler**

IPv4 compatibility of IPv6 sockets

- IPv6 sockets **might be able** to receive IPv4 traffic as well
- This is supposed to make the **server applications simpler**
- Server uses **only IPv6 sockets** = sees **only IPv6 addresses**

IPv4 compatibility of IPv6 sockets

- IPv6 sockets **might be able** to receive IPv4 traffic as well
- This is supposed to make the **server applications simpler**
- Server uses **only IPv6 sockets** = sees **only IPv6 addresses**
- IPv4 traffic is mapped to **IPv4-mapped IPv6 addresses**

IPv4 compatibility of IPv6 sockets

- IPv6 sockets **might be able** to receive IPv4 traffic as well
- This is supposed to make the **server applications simpler**
- Server uses **only IPv6 sockets** = sees **only IPv6 addresses**
- IPv4 traffic is mapped to **IPv4-mapped IPv6 addresses**
- IPv4 packets are **not internally translated**

IPv4 compatibility of IPv6 sockets

- IPv6 sockets **might be able** to receive IPv4 traffic as well
- This is supposed to make the **server applications simpler**
- Server uses **only IPv6 sockets** = sees **only IPv6 addresses**
- IPv4 traffic is mapped to **IPv4-mapped IPv6 addresses**
- IPv4 packets are **not internally translated**
- Using of IPv4 compatibility is **not recommended**

IPv4 compatibility of IPv6 sockets

- IPv6 sockets **might be able** to receive IPv4 traffic as well
- This is supposed to make the **server applications simpler**
- Server uses **only IPv6 sockets** = sees **only IPv6 addresses**
- IPv4 traffic is mapped to **IPv4-mapped IPv6 addresses**
- IPv4 packets are **not internally translated**
- Using of IPv4 compatibility is **not recommended**
  - properly written applications should **rather open two separate sockets**

IPv4 compatibility of IPv6 sockets

- IPv6 sockets **might be able** to receive IPv4 traffic as well
- This is supposed to make the **server applications simpler**
- Server uses **only IPv6 sockets** = sees **only IPv6 addresses**
- IPv4 traffic is mapped to **IPv4-mapped IPv6 addresses**
- IPv4 packets are **not internally translated**
- Using of IPv4 compatibility is **not recommended**
  - properly written applications should **rather open two separate sockets**
  - one cannot trust that **IPv6 stack is present** anyway

IPv4 compatibility of IPv6 sockets

- IPv6 sockets **might be able** to receive IPv4 traffic as well
- This is supposed to make the **server applications simpler**
- Server uses **only IPv6 sockets** = sees **only IPv6 addresses**
- IPv4 traffic is mapped to **IPv4-mapped IPv6 addresses**
- IPv4 packets are **not internally translated**
- Using of IPv4 compatibility is **not recommended**
  - properly written applications should **rather open two separate sockets**
  - one cannot trust that **IPv6 stack is present** anyway
  - OpenBSD deliberately **does not support it**

Socket option IPV6_V6ONLY

Socket option IPV6_V6ONLY

IPV6_V6ONLY=0
compatibility enabled

default for Linux and macOS

Socket option IPV6_V6ONLY

IPV6_V6ONLY=0
compatibility enabled

default for Linux and macOS

IPV6_V6ONLY=1
compatibility disabled

default for Windows and BSD
enforced on OpenBSD

Socket option IPV6_V6ONLY

IPV6_V6ONLY=0
compatibility enabled

default for Linux and macOS

IPV6_V6ONLY=1
compatibility disabled

default for Windows and BSD
enforced on OpenBSD

- Portable applications should **always set the option** properly

Socket option IPV6_V6ONLY

IPV6_V6ONLY=0
compatibility enabled

default for Linux and macOS

IPV6_V6ONLY=1
compatibility disabled

default for Windows and BSD
enforced on OpenBSD

- Portable applications should **always set the option** properly
- Enabled compatibility will block opening similar IPv4 socket

IPv4-mapped IPv6 addresses in the wild

IPv4-mapped IPv6 addresses

- Represent IPv4 addresses in **IPv6-only socket API**
- Should **never leave the host**
- Should never appear in **any IPv6 packet anywhere**
- It **would be silly** to try to put them **into the DNS**

IPv4-mapped IPv6 addresses

- Represent IPv4 addresses in **IPv6-only socket API**
- Should **never leave the host**
- Should never appear in **any IPv6 packet anywhere**
- It **would be silly** to try to put them **into the DNS**
- **Yet people are doing it**

```
$ host bam.nr-data.net
bam.nr-data.net is an alias for bam.cell.nr-data.net.
bam.cell.nr-data.net is an alias for fastly-tls12-bam.nr-data.net.
fastly-tls12-bam.nr-data.net has address 162.247.243.29
fastly-tls12-bam.nr-data.net has IPv6 address ::ffff:162.247.243.29
```

Why would somebody do that?

We did this to **drive down the cost** with our DNS provider. Queries for AAAA records that didn't exist, followed by queries for A records, was **costing us significantly** and we needed to alleviate that.

Our AAAA answers **follow the standards**, and our local dual-stack testing has shown no issues. The IPv4 addresses embedded in the IPv6 answers should be accurate, and should match the A record requests, and should all be routable in the IPv4 space.

Source: New Relic support forum, shared by Thomas Schäfer

This is silly. Or is it?

- I have set up two **test websites:**

This is silly. Or is it?

- I have set up two **test websites:**

https://ipv4-mapped.0skar.cz

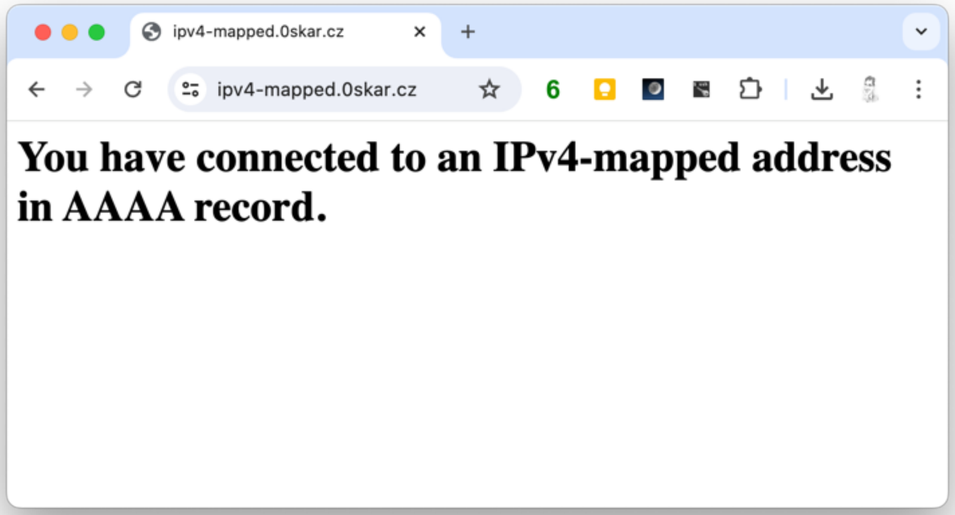
  - only AAAA record pointing to an IPv4-mapped IPv6 address
  - should be **universally unreachable**

This is silly. Or is it?

- I have set up two **test websites:**

https://ipv4-mapped.0skar.cz

- only AAAA record pointing to an IPv4-mapped IPv6 address
- should be **universally unreachable**



This is silly. Or is it?

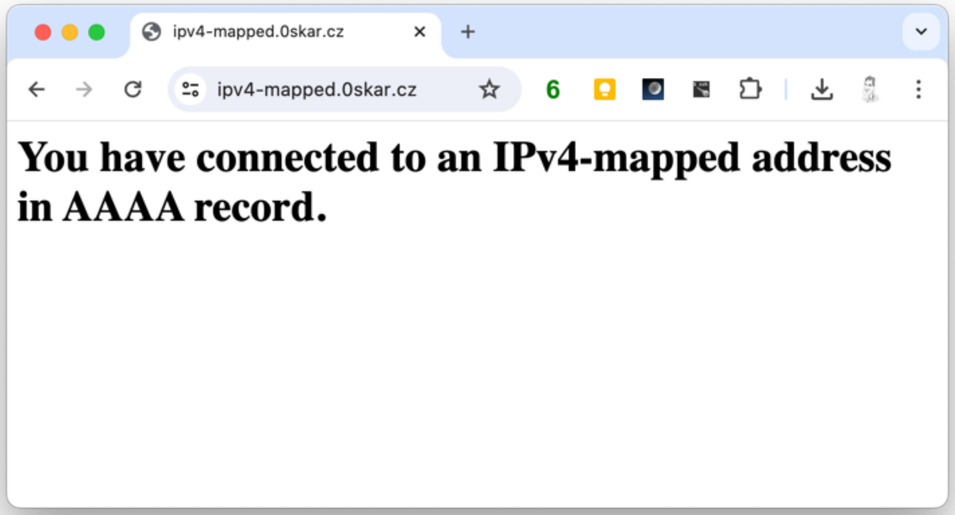
- I have set up two **test websites:**

https://ipv4-mapped.0skar.cz

- only AAAA record pointing to an IPv4-mapped IPv6 address
- should be **universally unreachable**

https://ipv4-mapped-pref.0skar.cz

- dual stack with AAAA record pointing to an IPv4-mapped IPv6 address
- should **always reach the A-record target**

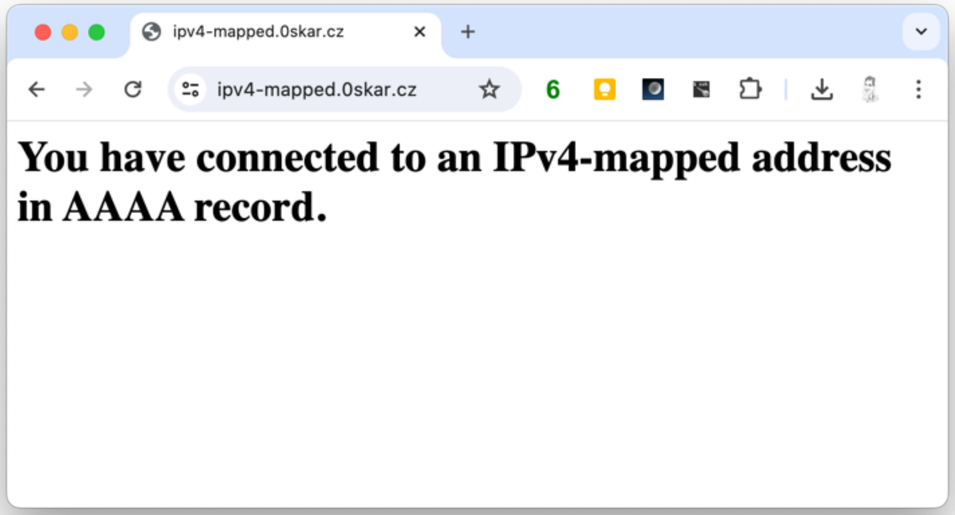


This is silly. Or is it?

- I have set up two **test websites:**

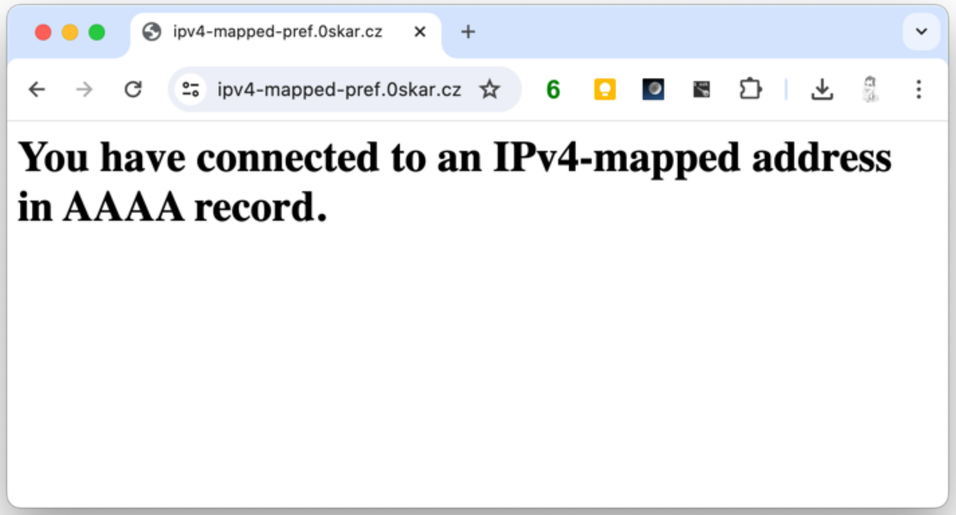
https://ipv4-mapped.0skar.cz

- only AAAA record pointing to an IPv4-mapped IPv6 address
- should be **universally unreachable**



https://ipv4-mapped-pref.0skar.cz

- dual stack with AAAA record pointing to an IPv4-mapped IPv6 address
- should **always reach the A-record target**



Looks like we have a problem

- The results depend on:
  - operating system
  - browser
  - network (dual-stack vs. IPv6-only)
- But in any case, all hosts issued **both AAAA and A queries**
  - so you **cannot save money** by putting IPv4-mapped addresses in AAAA records

Looks like we have a problem

- The results depend on:
  - operating system
  - browser
  - network (dual-stack vs. IPv6-only)
- But in any case, all hosts issued **both AAAA and A queries**
  - so you **cannot save money** by putting IPv4-mapped addresses in AAAA records

```
$ curl https://ipv4-mapped.0skar.cz
curl: (6) Could not resolve host: ipv4-mapped.0skar.cz
$ curl https://ipv4-mapped-pref.0skar.cz
<h1>You have connected to an IPv4 address in A record.</h1>
```

macOS on a dual-stack network

Looks like we have a problem

- The results depend on:
  - operating system
  - browser
  - network (dual-stack vs. IPv6-only)
- But in any case, all hosts issued **both AAAA and A queries**
  - so you **cannot save money** by putting IPv4-mapped addresses in AAAA records

```
$ curl https://ipv4-mapped.0skar.cz
curl: (6) Could not resolve host: ipv4-mapped.0skar.cz
$ curl https://ipv4-mapped-pref.0skar.cz
<h1>You have connected to an IPv4 address in A record.</h1>
```

macOS on a dual-stack network

```
$ curl https://ipv4-mapped.0skar.cz
<h1>You have connected to an IPv4-mapped address in AAAA record.</h1>
$ curl https://ipv4-mapped-pref.0skar.cz
<h1>You have connected to an IPv4-mapped address in AAAA record.</h1>
$ curl https://ipv4-mapped-pref.0skar.cz
<h1>You have connected to an IPv4 address in A record.</h1>
```

macOS on an IPv6-only network

Is this really a problem?

- Happy Eyeballs successfully **hide similar problems**
- Having a broken AAAA record **will break DNS64**
  - this can be avoided by setting up DNS64 to **ignore AAAA records** with addresses **outside the global unicast range**

What can we do about this?

- **DNS64 operators**:
  - **ignore addresses** outside 2000::/3 as valid AAAA-records
- **Operating system and/or browser vendors**:
  - *maybe* filter IPv4-mapped IPv6 addresses in the resolver?
- **DNS hosters**:
  - **don’t charge** your customers more **for empty responses**
- **Anyone**:
  - bring this to the IETF and clarify *unacceptable usage* of IPv4-mapped addresses

Questions

Ondrej.Caletka@ripe.net
@oskar456@mastodon.social

RIPE NCC
ACADEMY

IPv6 Fundamentals E-learning Course

- Free online course
- Study at your own pace
- Interactive learning

academy.ripe.net