

Single-stack IPv6-only data center deployments

Tore Anderson
CG Security and Networking
Redpill Linpro

IPv6 Forum Norway 2, Stavanger, November 2011

Our plan, in a nutshell

- Provision servers exclusively with IPv6 addresses/connectivity
- Let the network translate traffic from IPv4 clients to IPv6

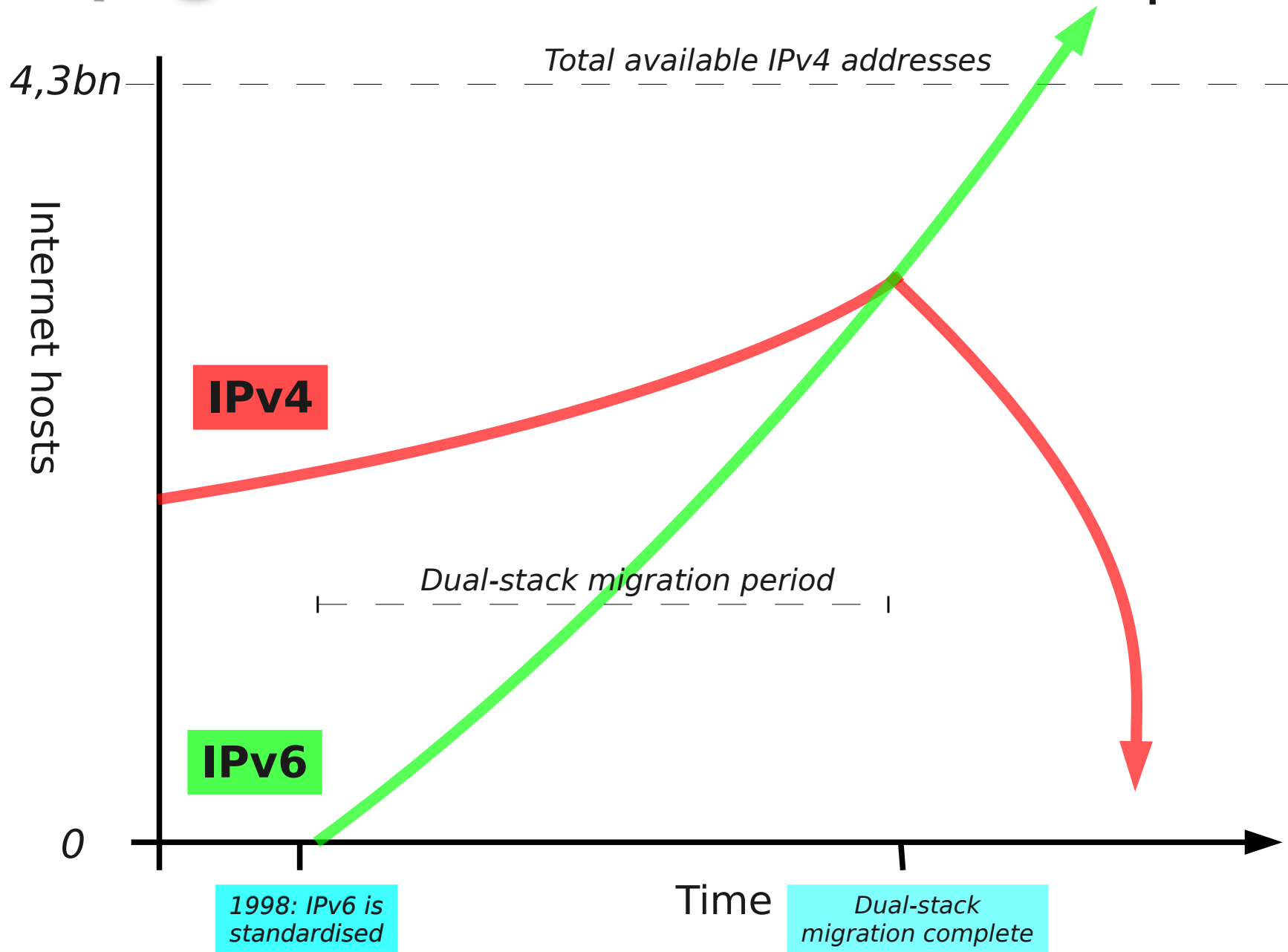
But what about dual-stack?

The simplest deployment model is dual stack: one turns on IPv6 throughout one's existing IPv4 network and allows applications using the two protocols to operate as ships in the night. This model is applicable to most networks -- home, enterprise, service provider, or content provider network.

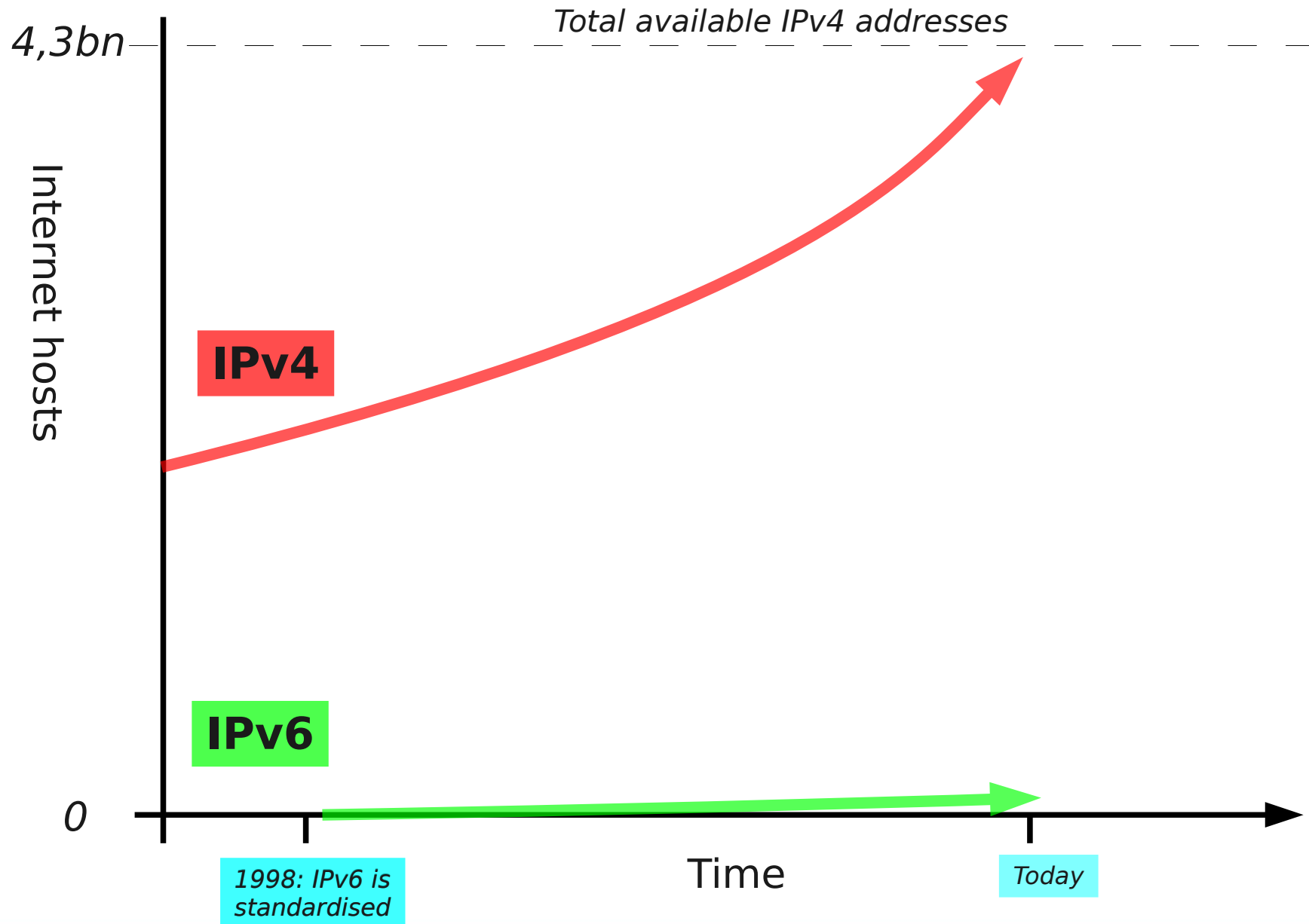
[...] the native dual-stack connectivity model remains the recommended approach.

-- RFC 6180: "Guidelines for Using IPv6 Transition Mechanisms during IPv6 Deployment"

Reminder: the dual-stack plan



Dual-stack progress report



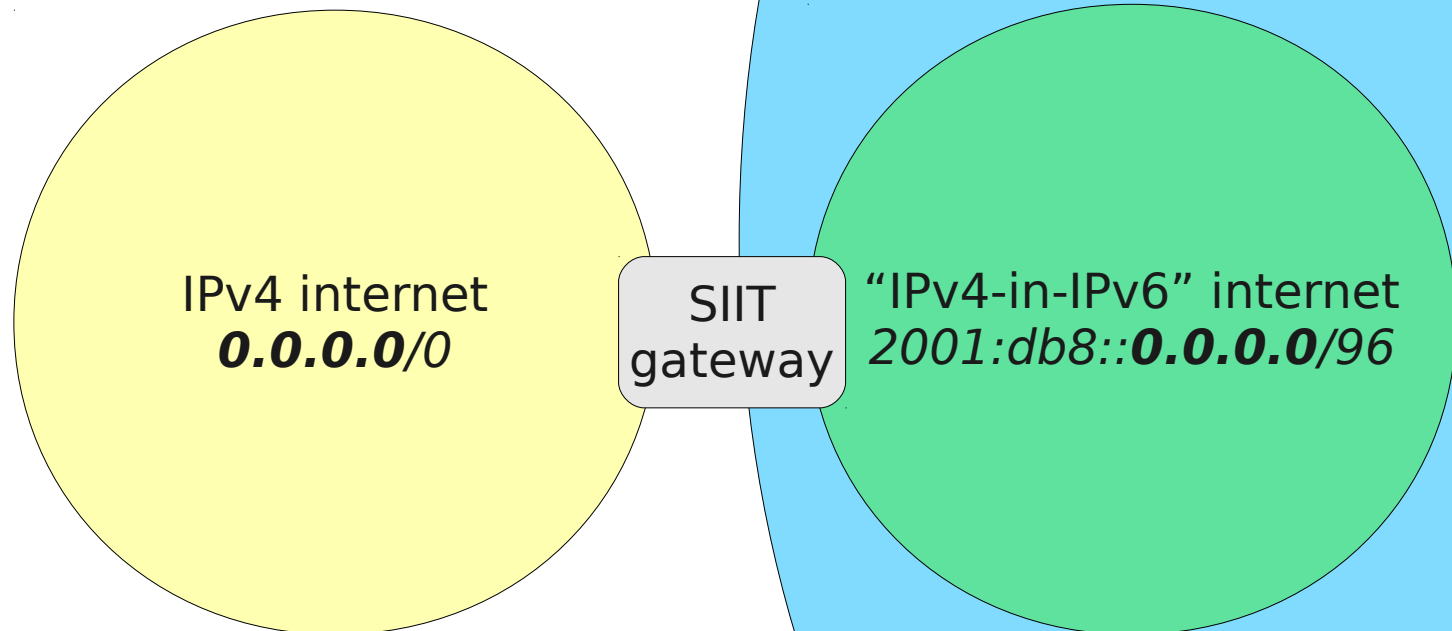
Disillusioned dual-stack

- A decade of being the recommended approach, yet **NO** deployment
- Does **not** in **any way** help with IPv4 depletion
 - ...the only real reason interest in IPv6 is picking up nowadays
 - We see increasing address consumption due to virtualization
- Adds complexity and operational overhead
 - More ACLs, more monitoring, more address management, more possible failure scenarios, more setup, more things to test....
 - Sysadmins resist complexity: *simple=stable*

IPv6-only viability

- Most of the server operating systems and applications we commonly use support IPv6(-only) very well
 - Apache, Bacula, Exim, HAProxy, Icinga, Linux, MySQL, Nginx, OpenSSH, OpenSolaris, Postfix, PostgreSQL, Puppet, Tomcat, Varnish, Zimbra....just to name a few
- Proprietary applications may or may not support IPv6
- A few server vendors support IPv6 ILO/OOB management
- Network boot/provisioning (PXE) does **not** support IPv6
- ***It all depends on the applications used***
 - I estimate that the majority of our customers' servers and applications could have run IPv6-only today with no problems

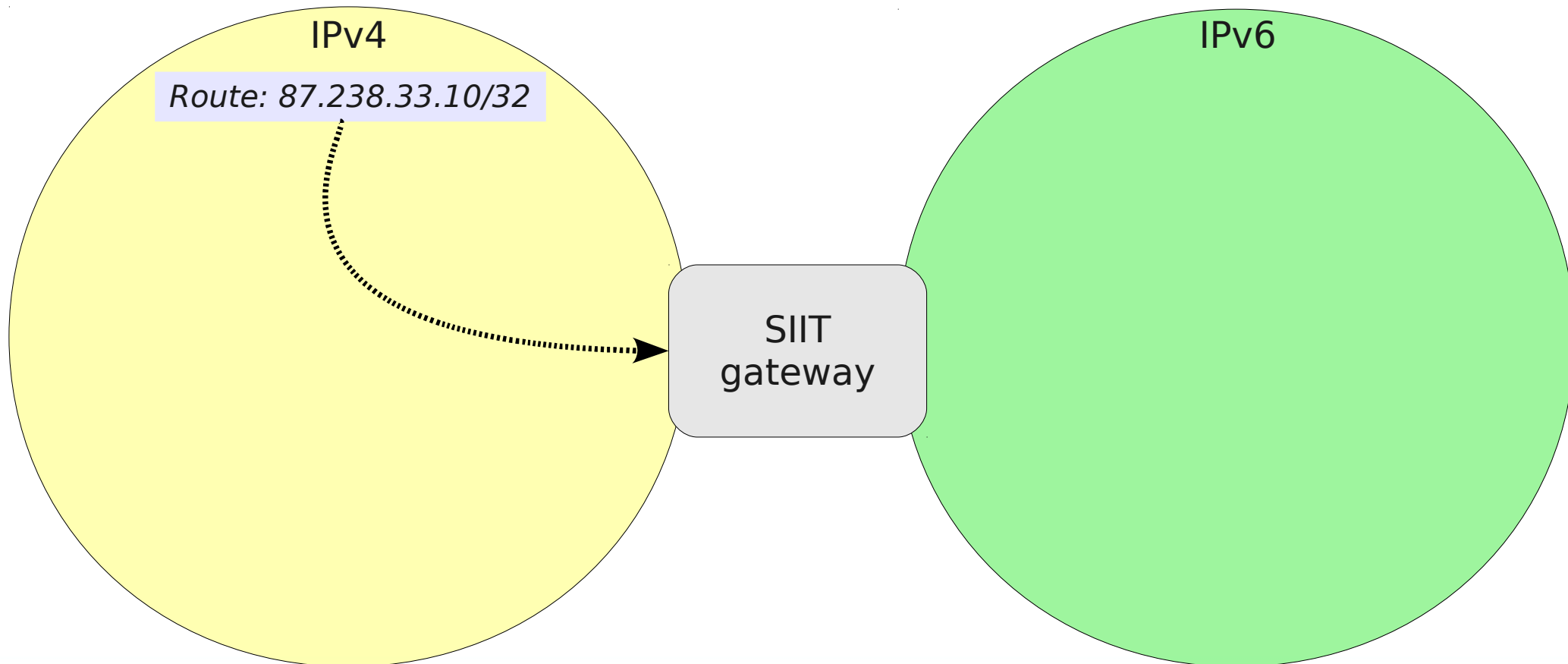
- **Stateless IP/ICMP Translation - SIIT**
 - Specified in RFC 6145 + RFC 6052
 - Also known as **IVI** and **NAT46**
- Maps the IPv4 internet into an arbitrary IPv6 prefix – e.g. **2001:db8::0.0.0.0/96**:



SIIT 101, part 1

One (or more) of the provider's public IPv4 addresses are routed to the IPv4 interface of the SIIT gateway, using standard IPv4 routing protocols.

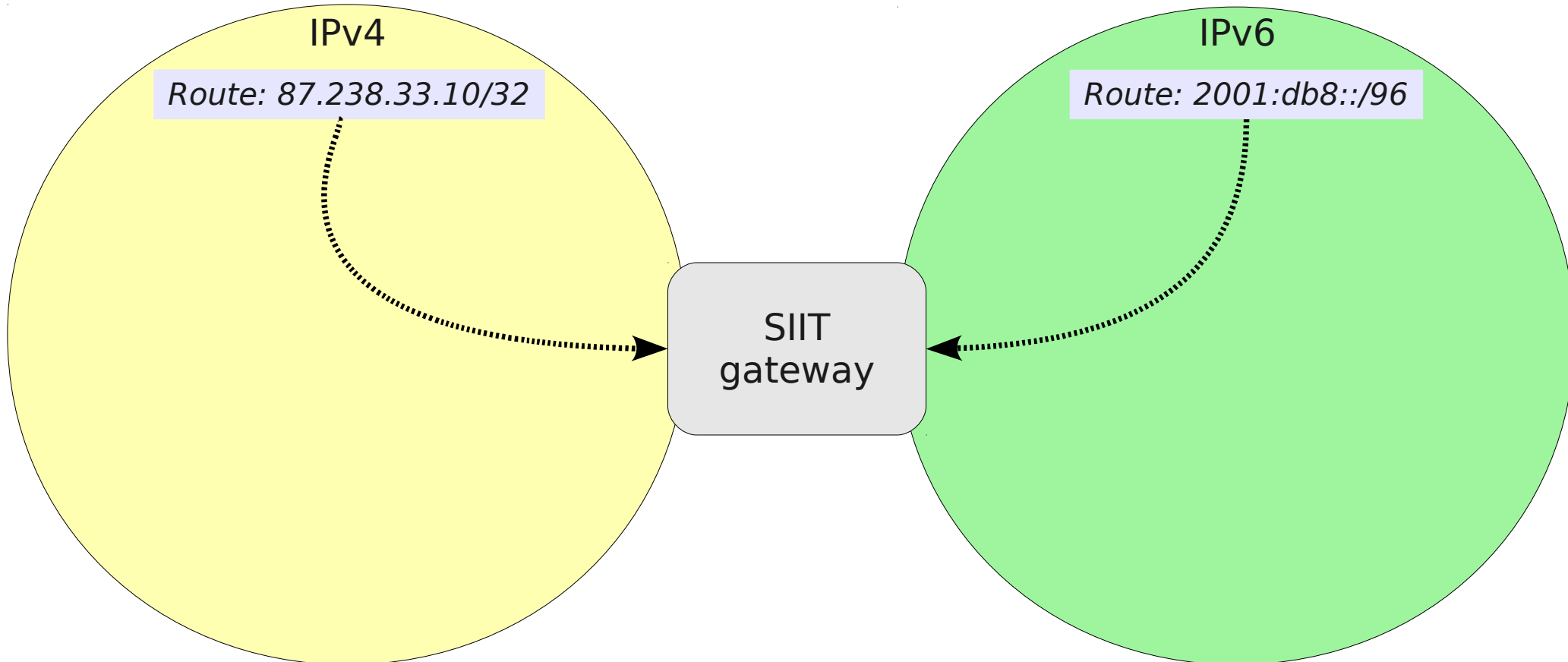
This address represents a single node inside the IPv6 domain.



SIIT 101, part 2

An IPv6 prefix is routed to the IPv6 interface of the SIIT gateway, again using standard routing protocols.

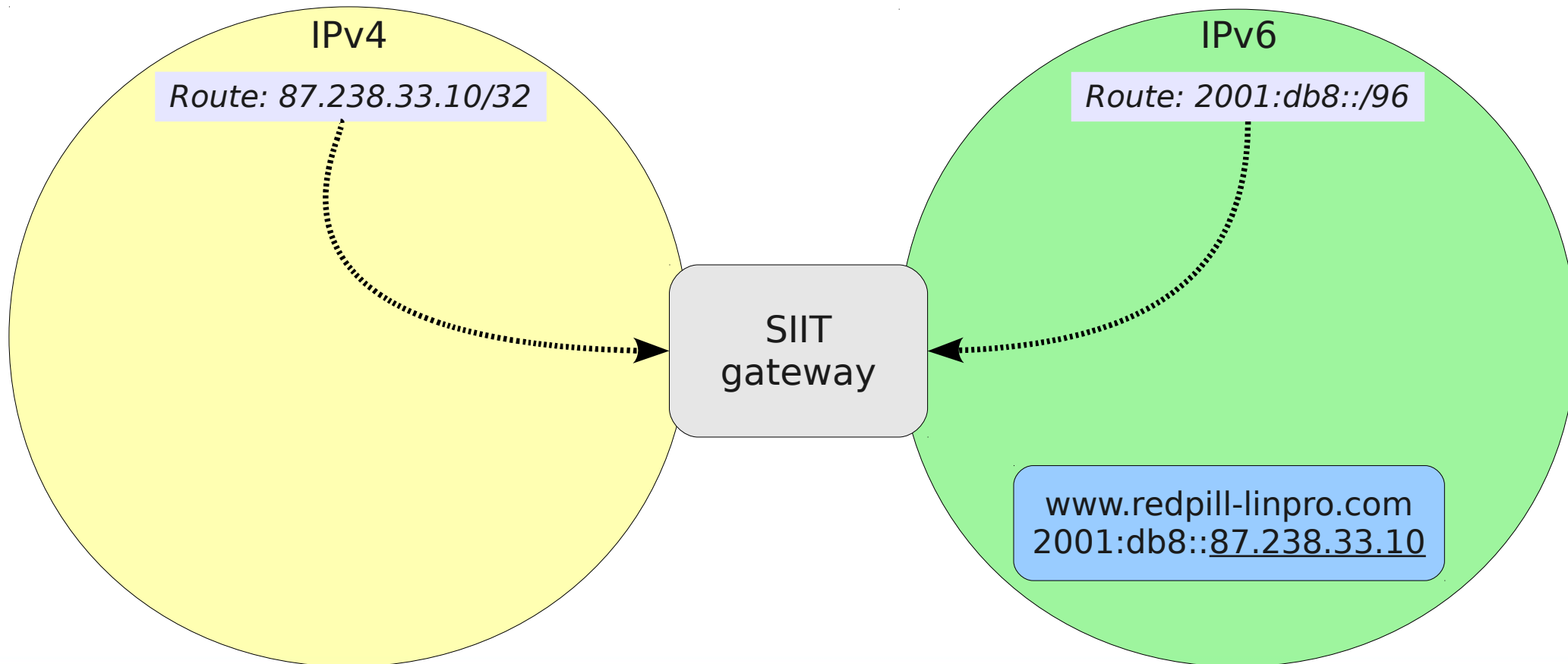
This prefix represents the entire IPv4 internet mapped into IPv6. It must be statically configured on the SIIT gateway.



SIIT 101, part 3

The server is configured with an IPv6 address that embeds the entire IPv4 address it will be reachable at from IPv4 clients protocols.

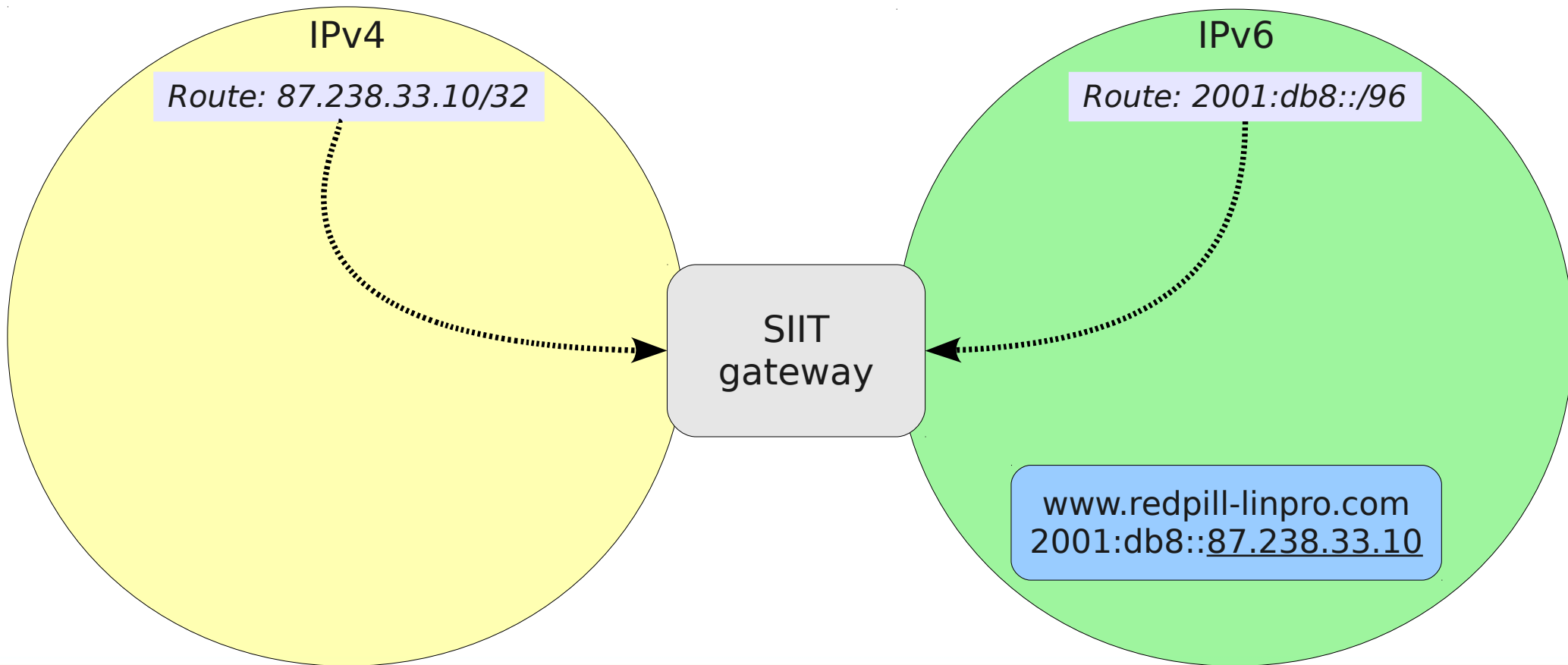
This address is routed to the server using standard IPv6 routing protocols.



SIIT 101, part 4

The server's translated IPv4 address is published in DNS alongside its native IPv6 address:

```
www.redpill-linpro.com.    IN A      87.238.33.10
www.redpill-linpro.com.    IN AAAA   2001:db8::87.238.33.10
```



The IP source/destination addresses are converted as follows (RFC 6052):

- When translating from IPv4 to IPv6: Prepend the IPv6 translation prefix
- When translating from IPv6 to IPv4: Strip the IPv6 translation prefix

The remainder of the IP header fields are converted according to a set of rules specified in RFC 6145, for example:

- IPv4 Time To Live <-> IPv6 Hop Limit;
- IPv4 Protocol <-> IPv6 Next Header; and so forth.

IPv4 packet

Source:

1.2.3.4

Destination:

87.238.33.10

Time To Live:

64



IPv6 packet

Source:

2001:db8::1.2.3.4

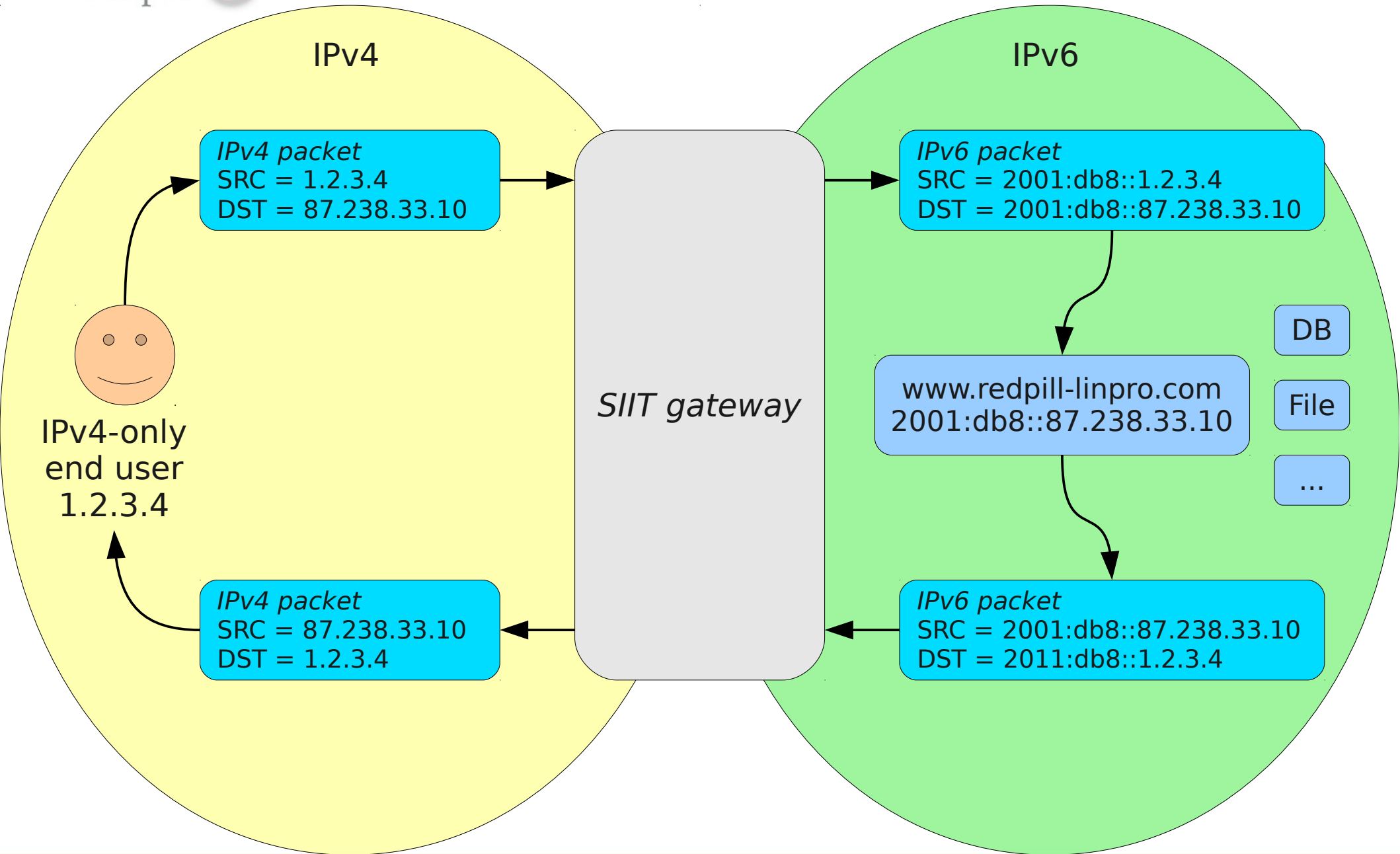
Destination:

2001:db8::87.238.33.10

Hop Limit:

64

Summary of a translated flow



Why we went with SIIT

- Avoids the complexity and operational overhead we get with dual-stack
- Excellent IPv4 address conservation
 - Only the servers that are running public services gets IPv4 addresses
 - No waste due to aggregation, infrastructure, oversized LAN prefixes
- Stateless operation has some **very** advantageous properties:
 - Performance: no need for flow tracking - wirespeed throughput
 - Availability: works fine with anycast and equal-cost multipathing
 - Flexibility: translators does not need to be placed on-path, flows does not have pass bi-directionally across a single translator
- Users' IPv4 addresses remain known to the application, e.g., for geo-loc
- Several production quality implementations exist, e.g., Cisco ASR
- We want to move towards the eventual sunseting of IPv4

Some possible pitfalls

- Applications that in general do not support NAT (e.g. FTP)
 - ALGs may solve the problem in some cases
- “Layer 4 MTU” mismatch due to the larger IPv6 header
 - Means full-sized IPv4 packets must be fragmented on the IPv6 side, if the MTU is the same on both the IPv4 and IPv6 interfaces
 - Not a problem for TCP as MSS will be negotiated separately
- Services that need to initiate connections (e.g. outbound mail servers)
 - Stateful NAT64+DNS64 or proxies may be used for sporadic outbound communication (such as OS patch retrieval)
- We intend to start with HTTP, which will avoid all of the above

Questions?

Thank you!

Further reading:

- RFC 6052 - IPv6 Addressing of IPv4/IPv6 Translators
- RFC 6145 – IP/ICMP Translation Algorithm
- RFC 6219 - The CERNET IVI Translation Design and Deployment for the IPv4/IPv6 Coexistence and Transition