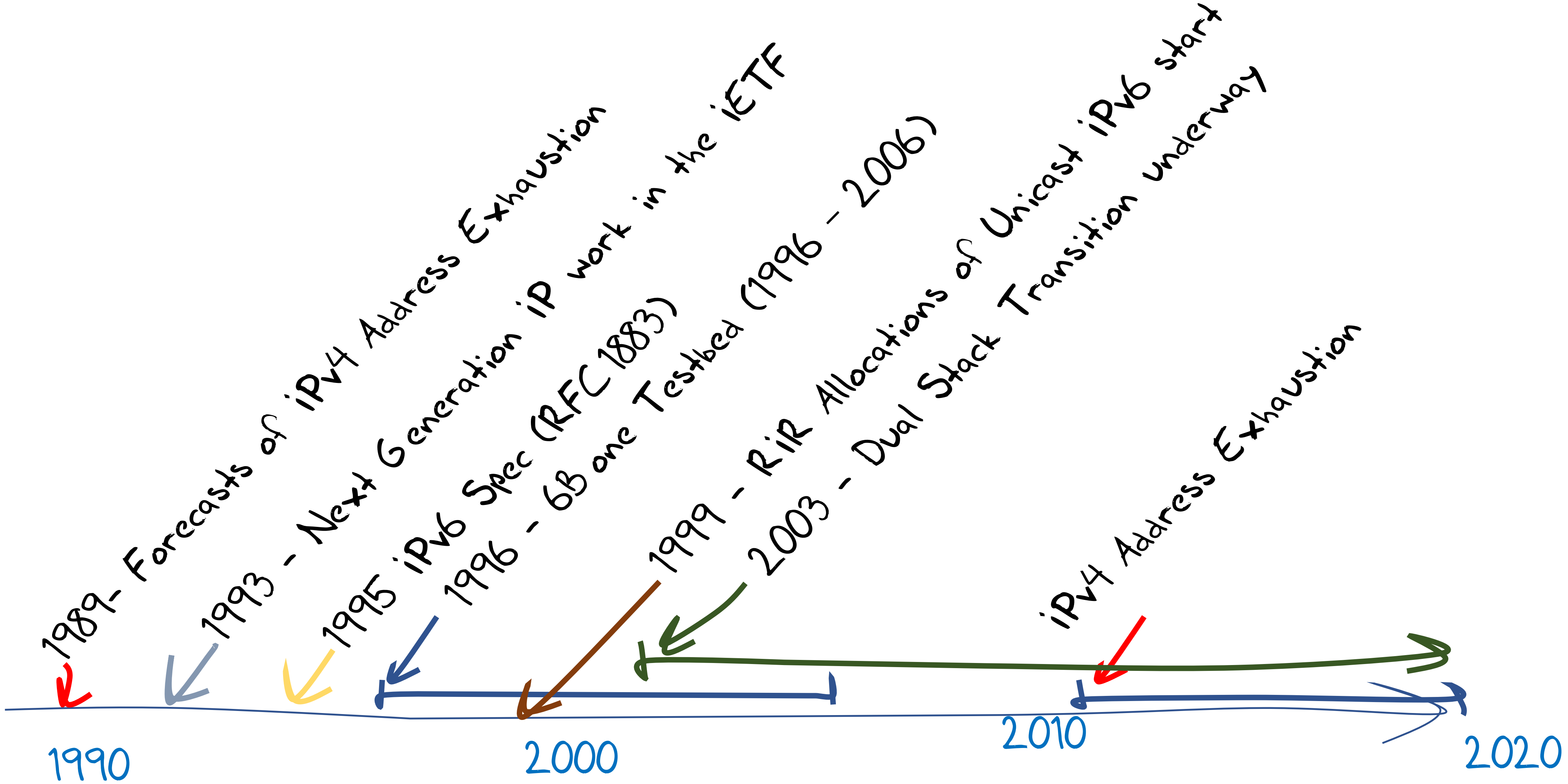
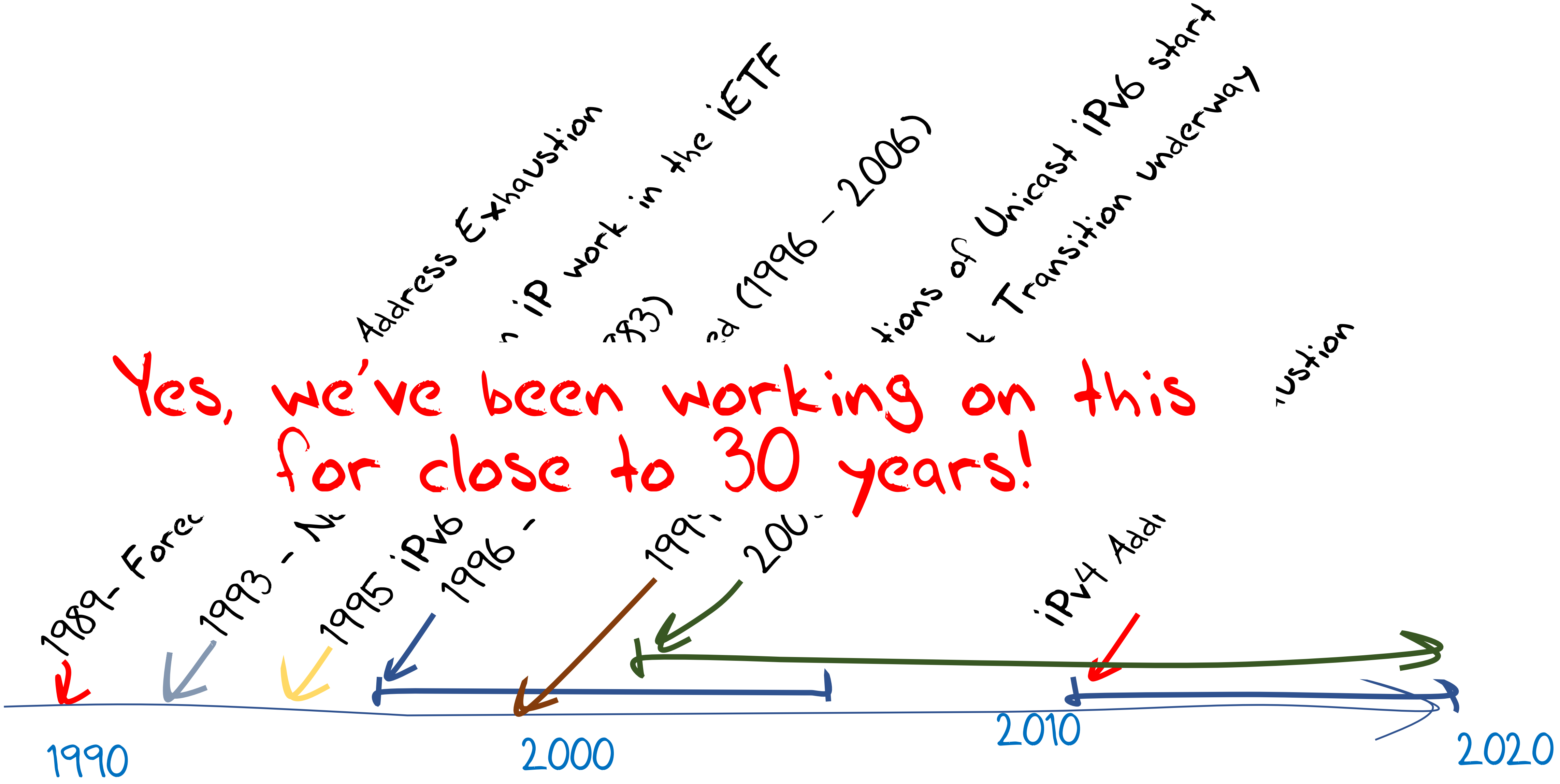
IPv6, the DNS and Big Packets

Geoff Huston, APNIC

The IPv6 Timeline…

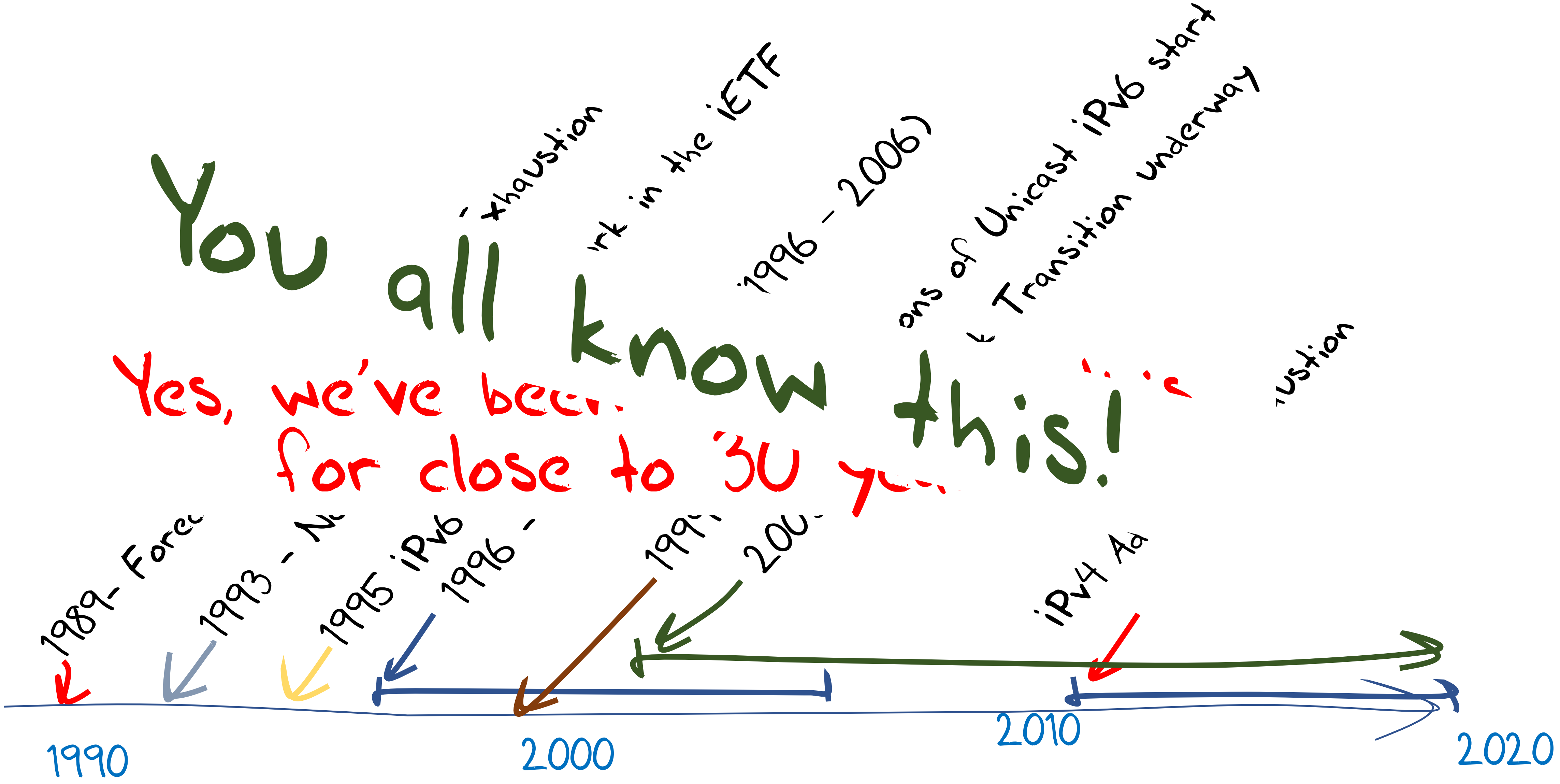

1989- Forecasts of IPv4 Address Exhaustion
1993 - Next Generation IP work in the IETF
1995 IPv6 Spec (RFC 1883)
1996 - 6Bone Testbed (1996 - 2006)
1999 - RIR Allocations of Unicast IPv6 start
2003 - Dual Stack Transition underway
IPv4 Address Exhaustion
1990
2000
2010
2020

The IPv6 Timeline…



Yes, we've been working on this for close to 30 years!

The IPv6 Timeline…



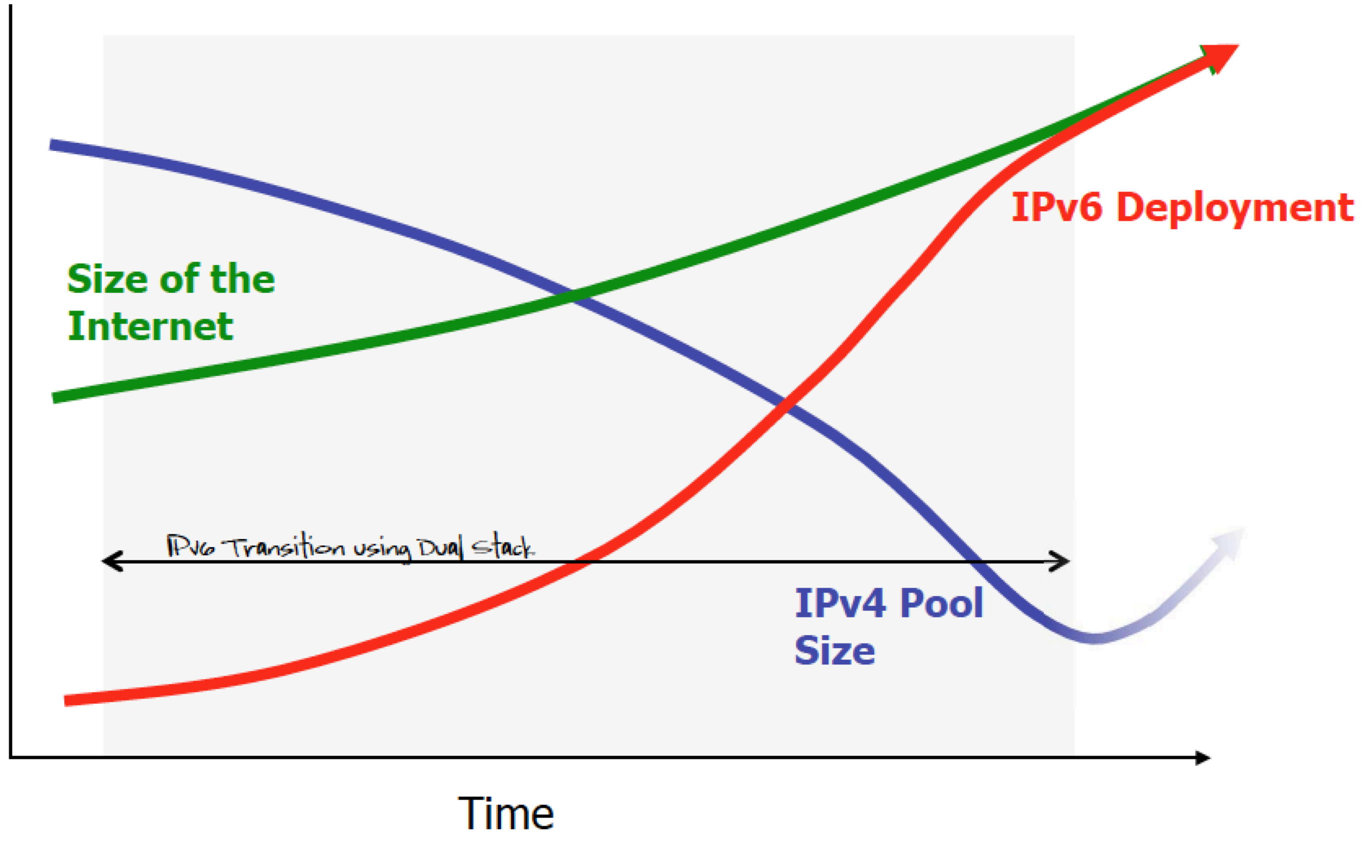
You all know this!

Yes, we've bee

for close to 30 ye

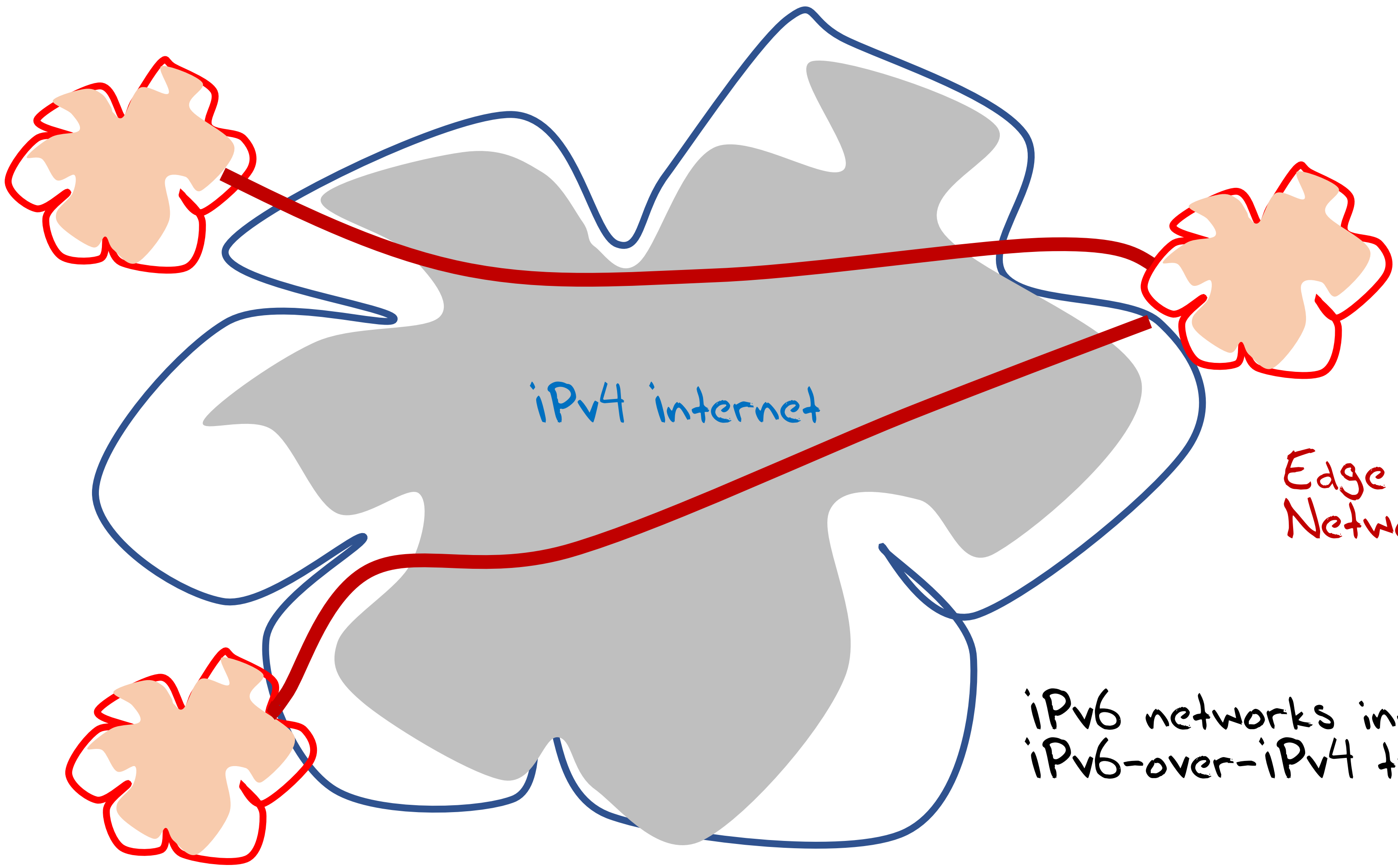
In-situ transition…

We had this plan …



In-situ transition…

Phase 1 - Early Deployment

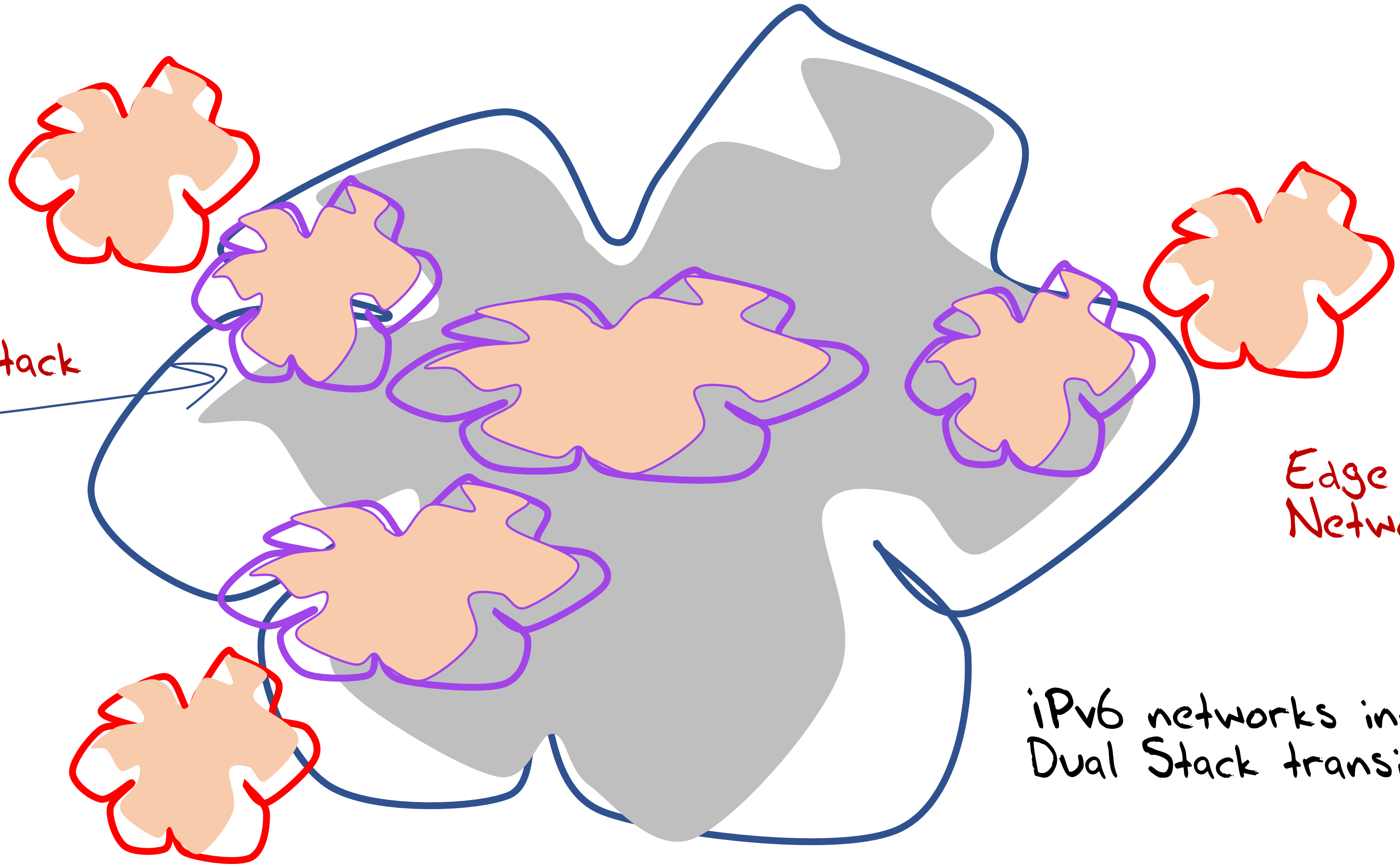


Edge Dual -Stack Networks

iPv6 networks interconnect by iPv6-over-iPv4 tunnels

In-situ transition…

Phase 2 - Dual Stack Deployment



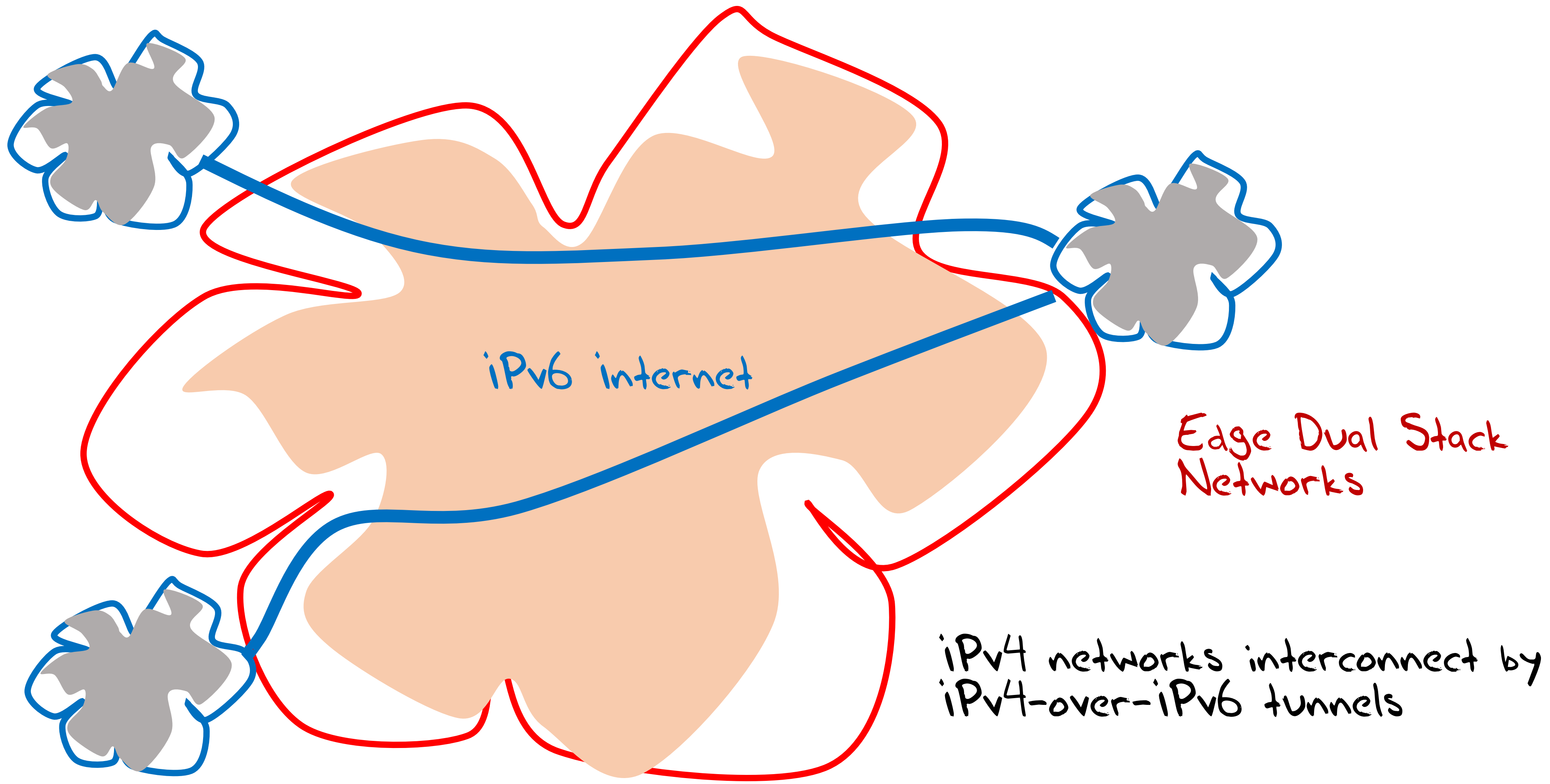
Transit Dual-Stack Networks

Edge Dual-Stack Networks

iPv6 networks interconnect by Dual Stack transit paths

In-situ transition…

Phase 3 - IPv4 Sunset

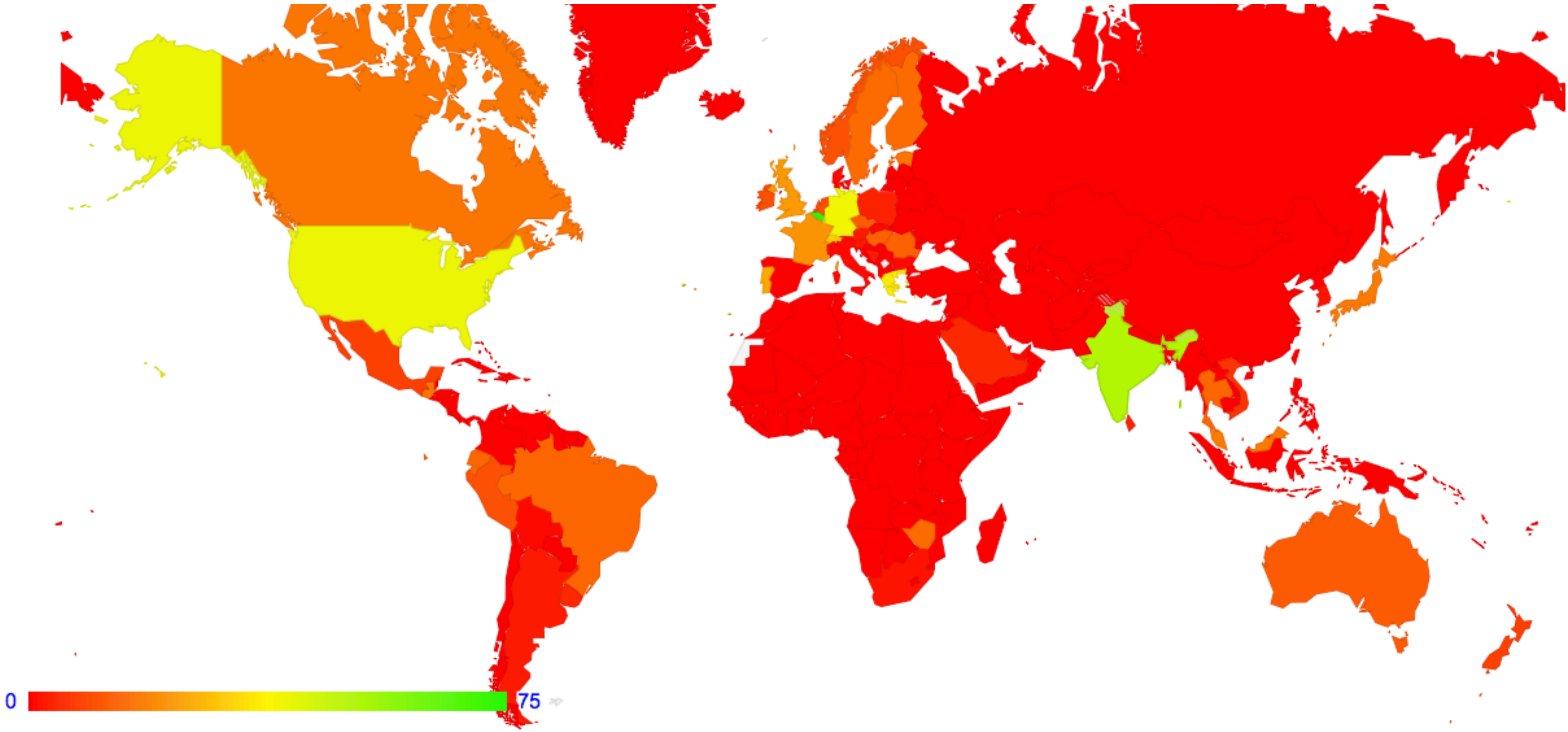


We are currently in Phase 2 of this transition

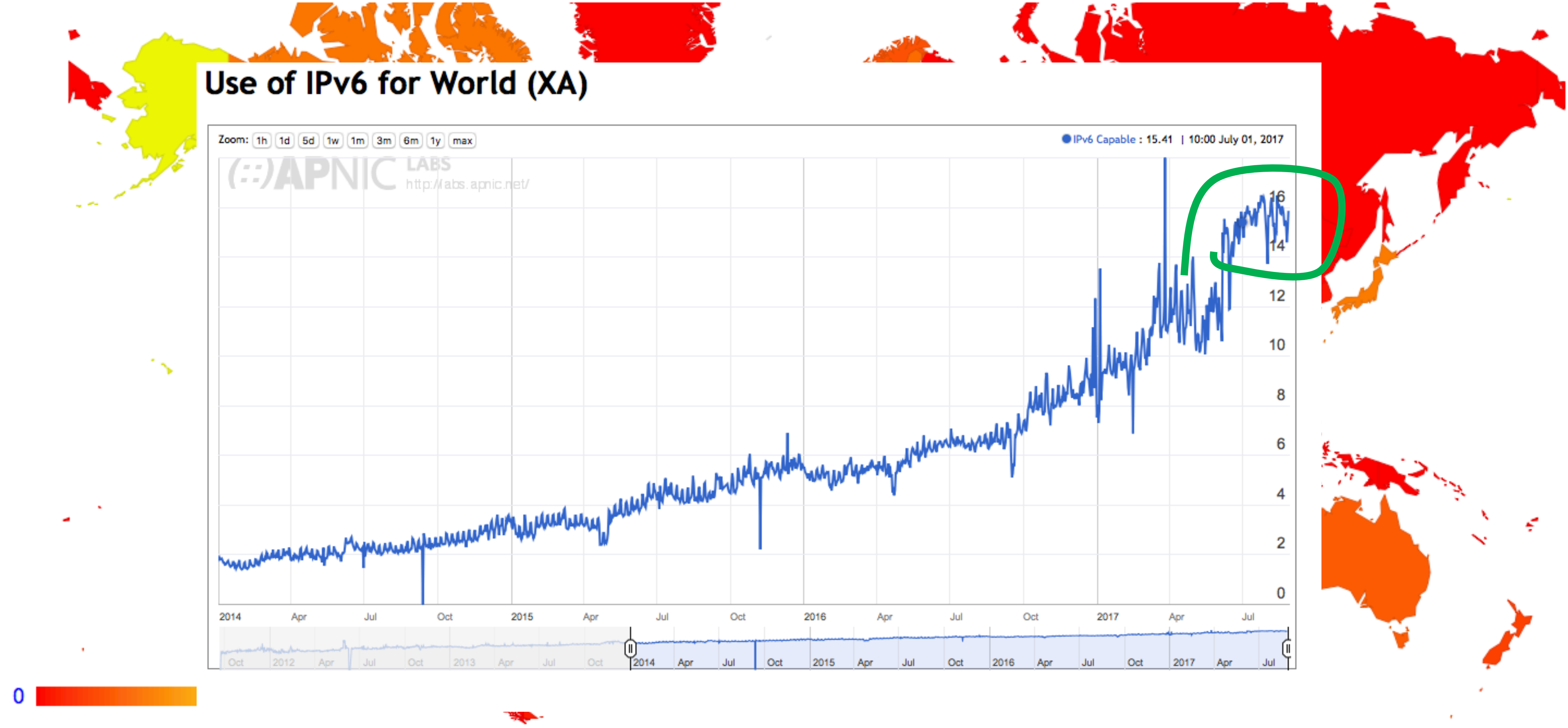
Some 15÷ - 20÷ of internet users have iPv6 capability

Most new iP deployments use iPv6+ (NATTED) iPv4

iPv4-only Legacy networks are being (gradually) migrated to dual stack



The Map of iPv6 penetration – August 2017



The Map of iPv6 penetration - August 2017

We are currently in Phase 2 of this transition

Some 15% of internet users have i

Most new IP deployme

IPv4-onl being (gradually) migrated to dual stack

The current situation is that the universal "glue" of the internet remains IPv4, while IPv6 is still an optional feature!

Today

We appear to be in the middle of the transition!

Dual Stack networks use apps that prefer to use a iPv6 connection over an iPv4 connection when both are available (*)

This implies that the higher the iPv6 deployment numbers the less the level of use of V4 connection, and the lower the pressure on the NAT binding clients

* Couple of problems with this:

This preference is often relative, and in the quest for ever faster connections the ante keeps rising – Apple is now pressing for a 50ms differential. This means that there is strong pressure for the iPv4 and iPv6 routing systems to be pretty much identical – and this is just not the case today!

Secondly, it's a client/server internet, rather than a client/client network, and the number of end clients running iPv6 has to be matched against the server population

Today

We appear to be in the middle of the transition!

Dual Stack networks cannot drop support for iPv4 as long as significant services and user populations do not support iPv6 – and we can't tell when that may change

Nobody is really in a position to deploy a robust at-scale ipv6-only network service today, even if they wanted to!

And we are not even sure if we can!

Today

We appear to be in the middle of the transition!

Dual Stack networks cannot drop support for IPv4 as long as significant services and user populations do not support IPv6 – and we can't tell when that may change

Nobody is really in a position to deploy a robust at-scale ipv6-only network service today, even if they wanted to!

And we are not even sure if we can!

The Issue

We cannot run Dual-Stack services indefinitely

At some point we need to support networks that only have iPv6

is that viable today?

In other words…

What do we rely on today in iPv4 that does not appear to have a clear working counterpart in iPv6?

In other words…

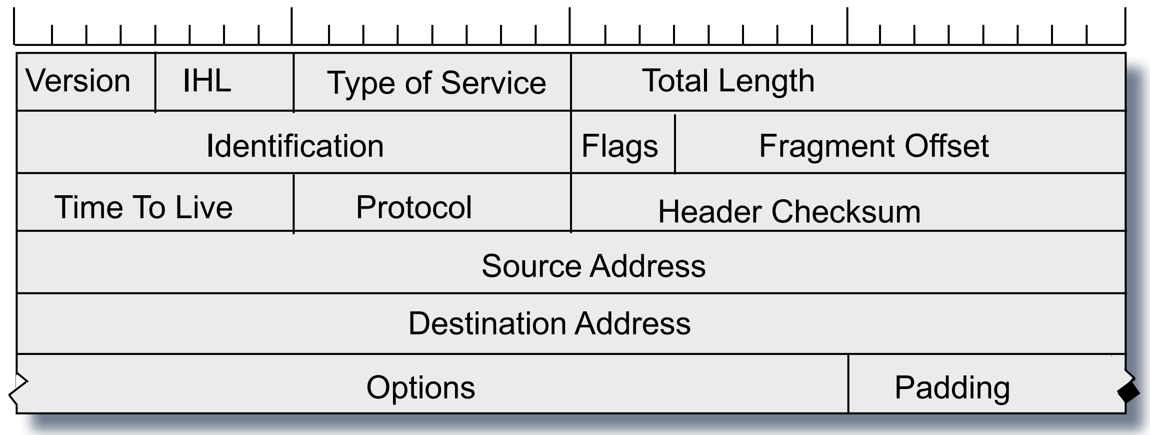
What do we rely on today in IPv4 that does not appear to have a clear working counterpart in IPv6?

if the answer is "nothing" then we are done!

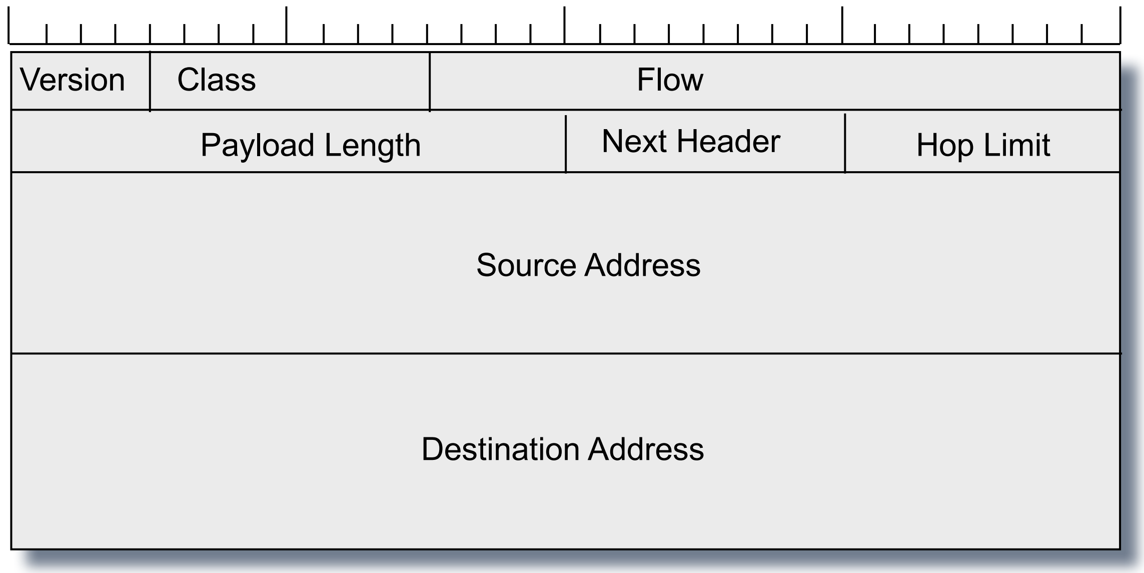
But if there is an issue here, then we should be working on it!

IPv6: What changed?

iPv4 Header



iPv6 Header



IPv6: What changed?

Type of Service is changed to Traffic Class

Flow Label Added

Options and Protocol fields replaced by Extension Headers

32 bit Fragmentation Control were pushed into an Extension Header

Checksum becomes a media layer function

IPv6: What changed?

Type of Service is changed to Traffic Class

Flow Label Added

Options

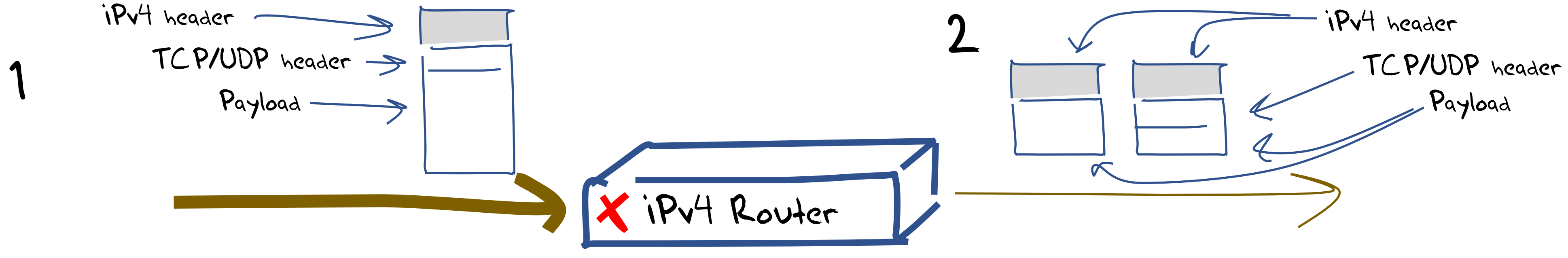
The substantive change with IPv6 is the handling of Fragmentation and the use of Extension Headers

into an Extension Header

becomes a media layer function

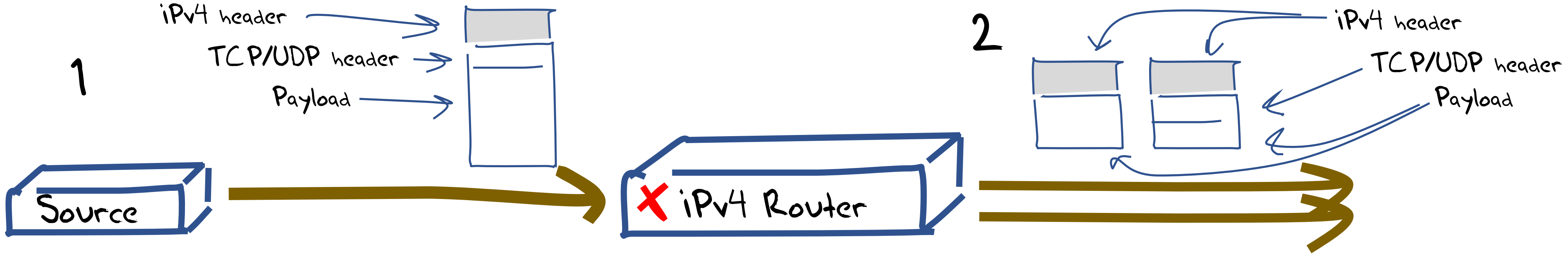
IPv6: What changed?

IPv4 "Forward Fragmentation"

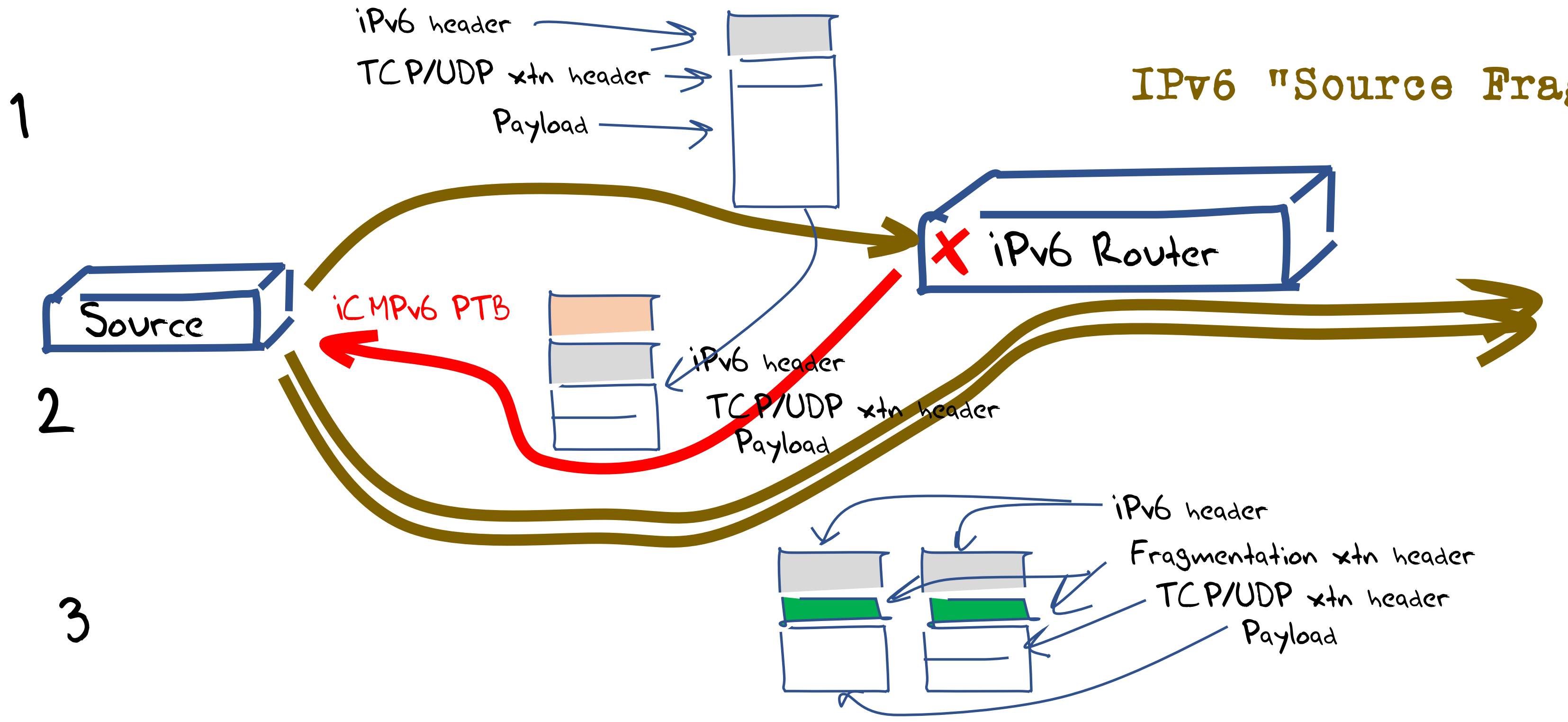


IPv6: What changed?

IPv4 "Forward Fragmentation"



IPv6 "Source Fragmentation"



New Dependencies

For IP fragmentation to work in IPv6 then:

- all ICMPv6 messages have to be passed backwards from the interior of the network to the sender
- IPv6 packets containing a IPv6 Fragmentation Extension header should not be dropped

ICMPv6

Only the sending host now has control of fragmentation – this is a new twist

A received ICMPv6 message needs to alter the sender's state to that destination

For TCP, if the ICMP payload contains the TCP header, then you can pass this to the TCP control block. TCP can alter the session MSS and resend the dropped data, or you can just alter the local per-destination MSS and hope that TCP will be prompted to resend

For UDP – um, err, um well

ICMPv6

Only the sending host now has control of fragmentation – this is a new twist

A received ICMPv6 message needs to alter the sender's state to that destination

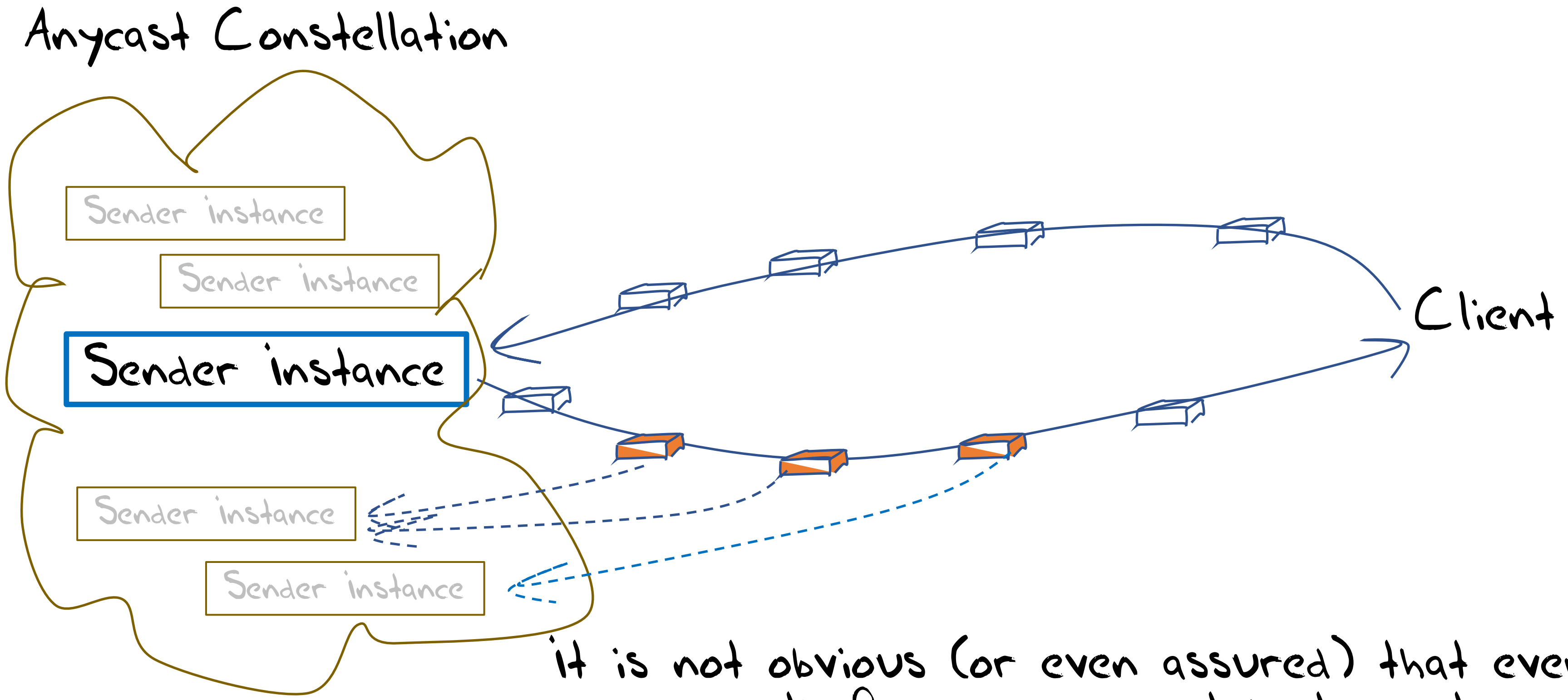
For TCP, if the ICMP payload contains the TCP header, then you can pass this to the TCP control block. TCP can alter the session MSS and resend the dropped data, or you can just alter the local per-destination MSS and hope that TCP will be prompted to resend

For UDP – um, err, um well

Maybe you should store the revised path MTU in a host forwarding table cache for a while

if you ever need to send another UDP packet to this host you can use this cache entry to guide your fragmentation behaviour

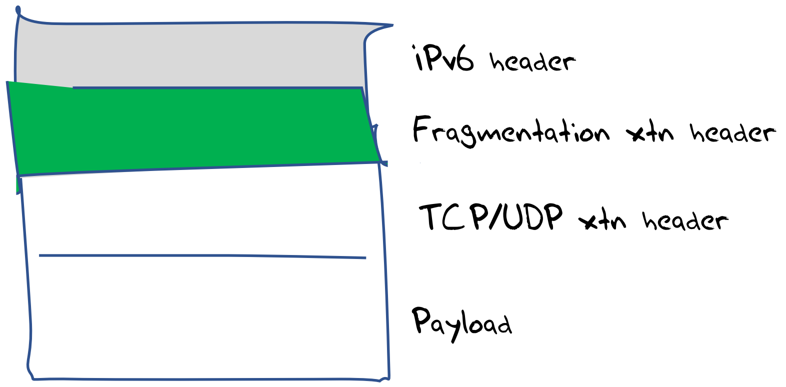
ICMPv6 and Anycast



It is not obvious (or even assured) that every router on the reverse path from an anycast instance to a client host will necessarily be part of the same anycast instance "cloud"

The implication is that in anycast, the reverse ICMPv6 PTB messages will not necessarily head back to the original sender!

IPv6 Fragmentation Extension Header Handling



The extension header sits between the iPv6 packet header and the upper level protocol header for the leading fragged packet, and sits between the header and the trailing payload frags for the trailing packets

Practically, this means that transport-protocol aware packet processors/switches need to decode the extension header chain, if its present, which can consume additional cycles to process/switch a packet – and the additional time is not predictable. For trailing frags there is no transport header!

Or the unit can simply discard all ipv6 packets that contain extension headers!

Which is what a lot of transport protocol sensitive iPv6 deployed switching equipment actually does (e.g. load balancers!)

IPv6 Fragmentation Extension Header Handling

There is a lot of "drop" behaviour in the ipv6 internet for Fragmentation Extension headers

RFC 7872 – recorded drop rates of 30÷ - 40÷

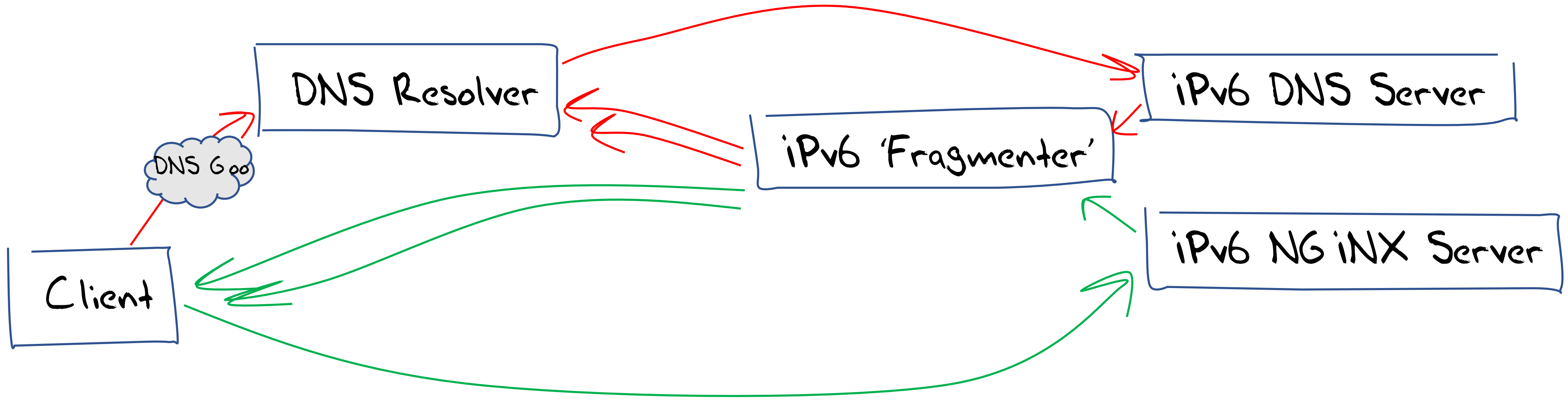
This experiment sent fragmented packets towards well-known servers and observed whether the server received and reconstructed the fragmented packet

B ut sending fragmented queries to servers is not all that common – the reverse situation of big responses is more common

So what about sending fragmented packets B ACK from servers – what's the drop rate of the reverse case?

IPv6 Fragmentation Extension Header Handling

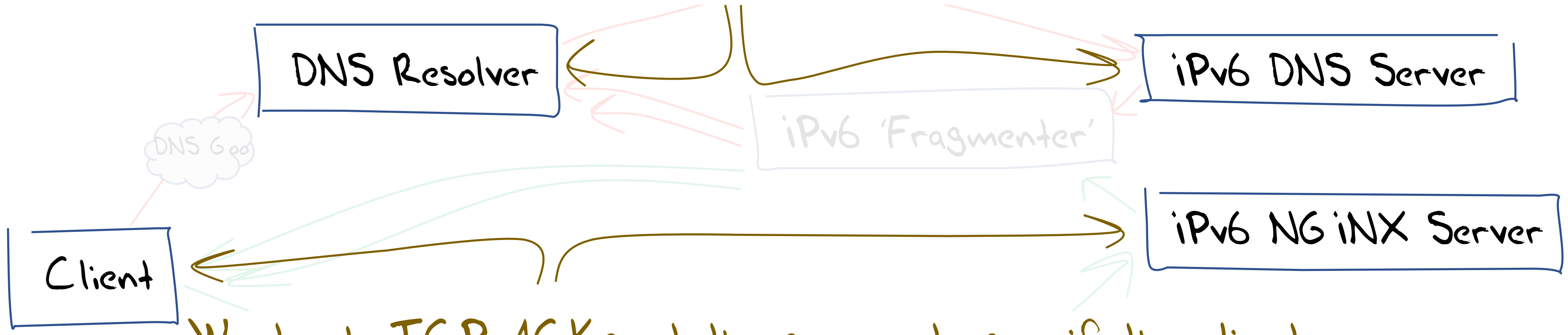
We used an ad-based measurement system, using a custom packet fragmentation wrangler as a front end to a DNS and Web server to test iPv6 fragmentation behaviour



IPv6 Fragmentation Extension Header Handling

We used an ad-based measurement system, using a custom packet fragmentation wrangler as a front end to a DNS and Web server to test iPv6 fragmentation behaviour

We use a technique of "glueless" delegation and fragmentation of the NS query response to allow us to detect if the DNS resolver received the fragmented response



We track TCP ACKs at the server to see if the client received the fragmented TCP response

IPv6 Fragmentation Extension Header Handling

Our Experiments across some 40M invidiaul sample points:

37÷ of end users who used iPv6-capable DNS resolvers could not receive a fragmented iPv6 response

20÷ of iPv6-capable end users could not receive a fragmented iPv6 packet

IPv6 Fragmentation is very unreliable

Why don't we see this unreliability in today's IPv6 networks affecting user transactions?

IPv6 Fragmentation is very unreliable

Why don't we see this unreliability in today's IPv6 networks affecting user transactions?

Because IPv4 papers over the problem!

IPv6 Fragmentation is very unreliable

Why don't we see this unreliability in today's iPv6 networks affecting user transactions?

Because iPv4 papers over the problem!

in a Dual-Stack environment there is always the option to flip to use iPv4 if you are stuck with ipv6.

The DNS does this, and Happy Eyeballs does this

So there is no user-visible problem in a dual stack environment

This means that there is no urgent imperative to correct these underlying problems in deployed iPv6 networks

There is little in the way of practical incentives to fix this today!

Living without IPv6 Fragmentation

if we apparently don't want to fix this, can we live with it?

We are living with it in a Dual Stack world, because ipv4 just makes it all better!

B ut what happens when there is no ipv4 left?

Living without IPv6 Fragmentation

But what happens when there is no ipv4 left?

We have to avoid iPv6 Fragmentation!

The balance is between iCMPv6 PTB signal loss and iPv6 Frag EH packet loss

Maybe we need to use protocol-sensitive MTU settings for iPv6:

- TCP should go "low" – 1,350 bytes is a decent point – to avoid iMCPV6 PTB Black holes
- UDP (except QUiC) should go "high" – 1,500 bytes avoids unnecessary fragmentation and avoids unnecessary iPv6 Frag EH loss
- QUiC should be especially careful to avoid iCMPv6 PTB completely, so "low" is better

Living without IPv6 Fragmentation

B ut what happens when there is no ipv4 left?

We have to avoid iPv6 Fragmentation!

TCP can work as long as iPv6 sessions use conservative MSS sizes

UDP can work as long as UDP packet sizes are capped so as to avoid fragmentation

Living without IPv6 Fragmentation

B ut what happens when there is no ipv4 left?

We have to avoid iPv6 Fragmentation!

TCP can work as long as iPv6 sessions use conservative MSS sizes

UDP can work as long as UDP packet sizes are capped so as to avoid fragmentation

Who needs to use large UDP packets anyway?

Living without IPv6 Fragmentation

B ut what happens when there is no ipv4 left?

We have to avoid iPv6 Fragmentation!

TCP can work as long as iPv6 sessions use conservative MSS sizes

UDP can work as long as UDP packet sizes are capped so as to avoid fragmentation

Who needs to use large UDP packets anyway?

DNSSEC!

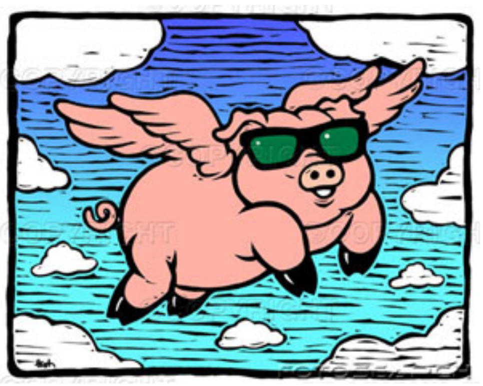
Where are we?

in terms of protocol support and reliability, it seems that we are mostly ready for an iPv6-only environment, with the one exception of iPv6 packet fragmentation handling.

The consequence is that today's environment cannot support an ipv6-only environment for the DNS, and cannot support iPv6-only DNSSEC

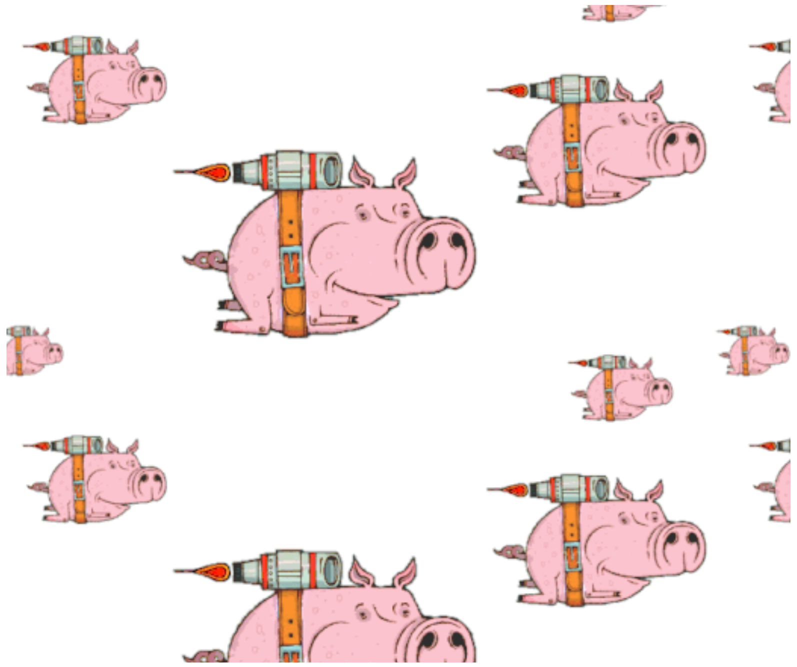
What can we do about it?

A. Get all the deployed routers and switches to accept packets with IPv6 Fragmentation Headers



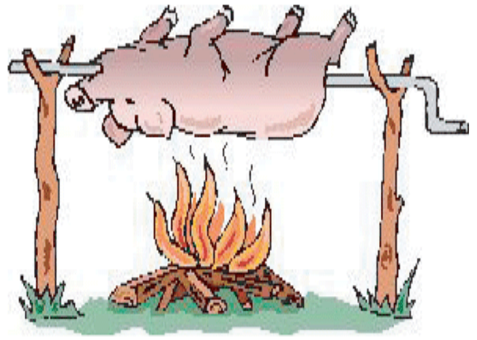
What can we do about it?

B. Alter the way iPv6 describes packet fragmentation



What can we do about it?

C. Move the DNS off UDP



An IPv6-only Internet?

We're close!

An IPv6-only Internet?

We're close!

The issue of the unreliability of iPv6 fragmentation is a significant issue.

We can largely avoid this in TCP (and QUIC) with conservative MSS settings that avoid path MTU mismatch and fragmentation.

This issue is most prevalent in the public internet in DNSSEC
if we can change the behaviours of the DNS away from reliance of large UDP packets then we can be more confident that we can operate the net as iPv6-only without an iPv4 backstop!

For iPv6-only environments, its time to take DNS over something other than only UDP a little more seriously than we do today

Acknowledgements and Thanks!

For the Ad-based measurement platform:

- Google has supported our efforts to conduct this measurement since its inception, and continues to support us.
- ICANN provide support for this measurement activity, particularly relating to our measurement of DNS and DNSSEC capabilities
- ISC for a conveniently located server at PAIX
- And thanks to Ray Bellis for development of the original dynamic DNS server code

Thanks!