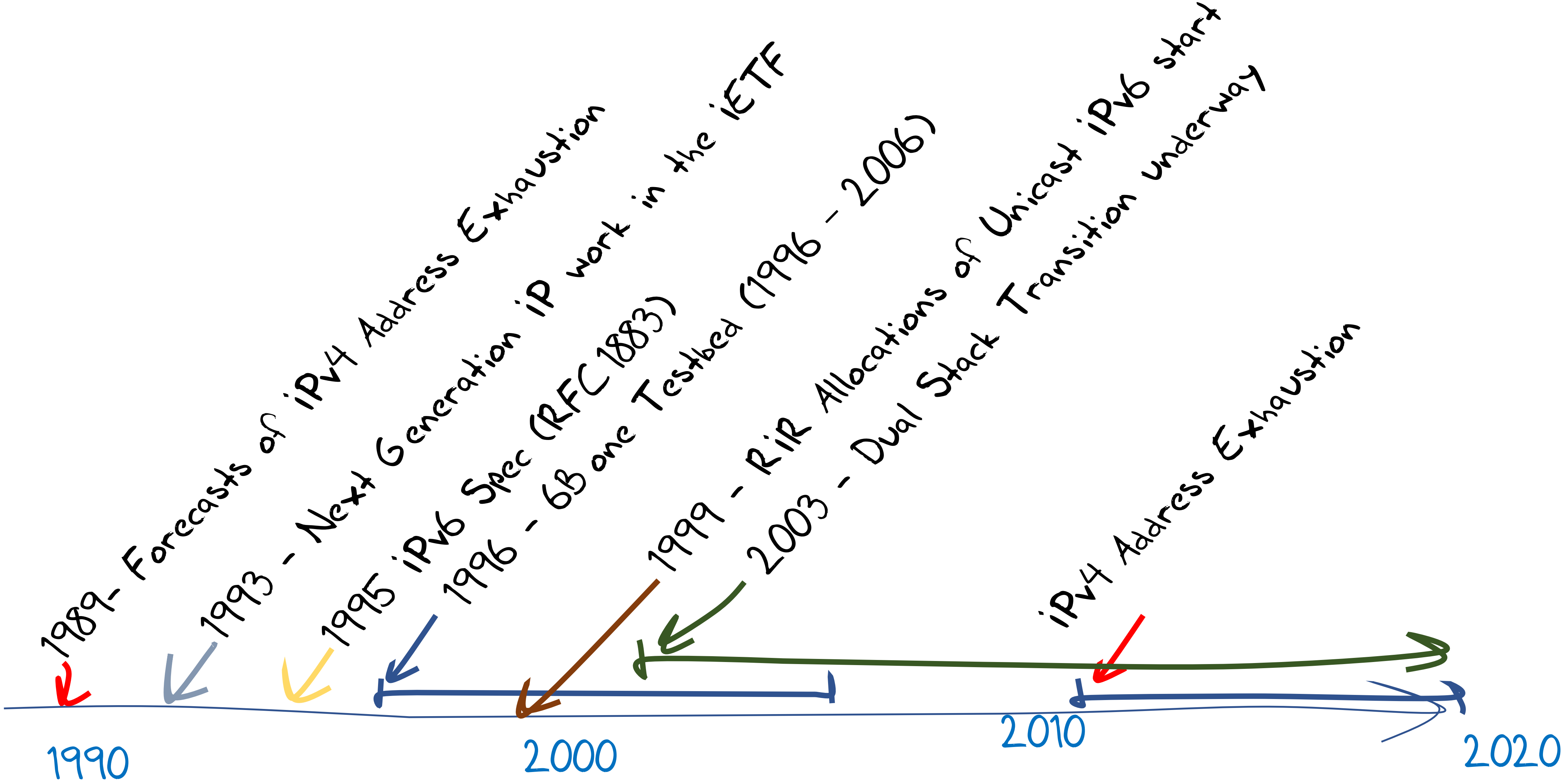
IPv6: Are we really ready to turn off IPv4?

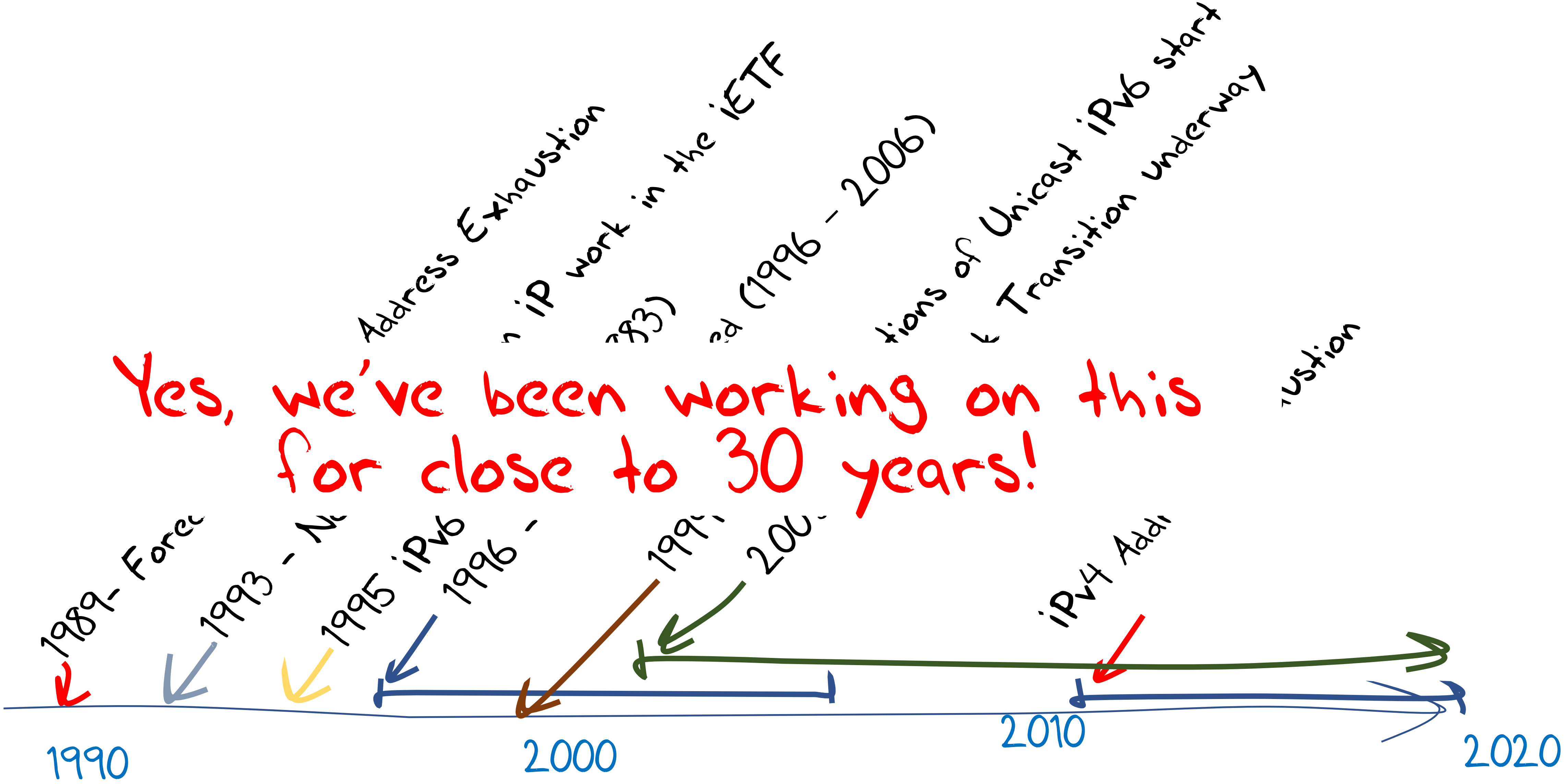
Geoff Huston

APNIC

The IPv6 Timeline…

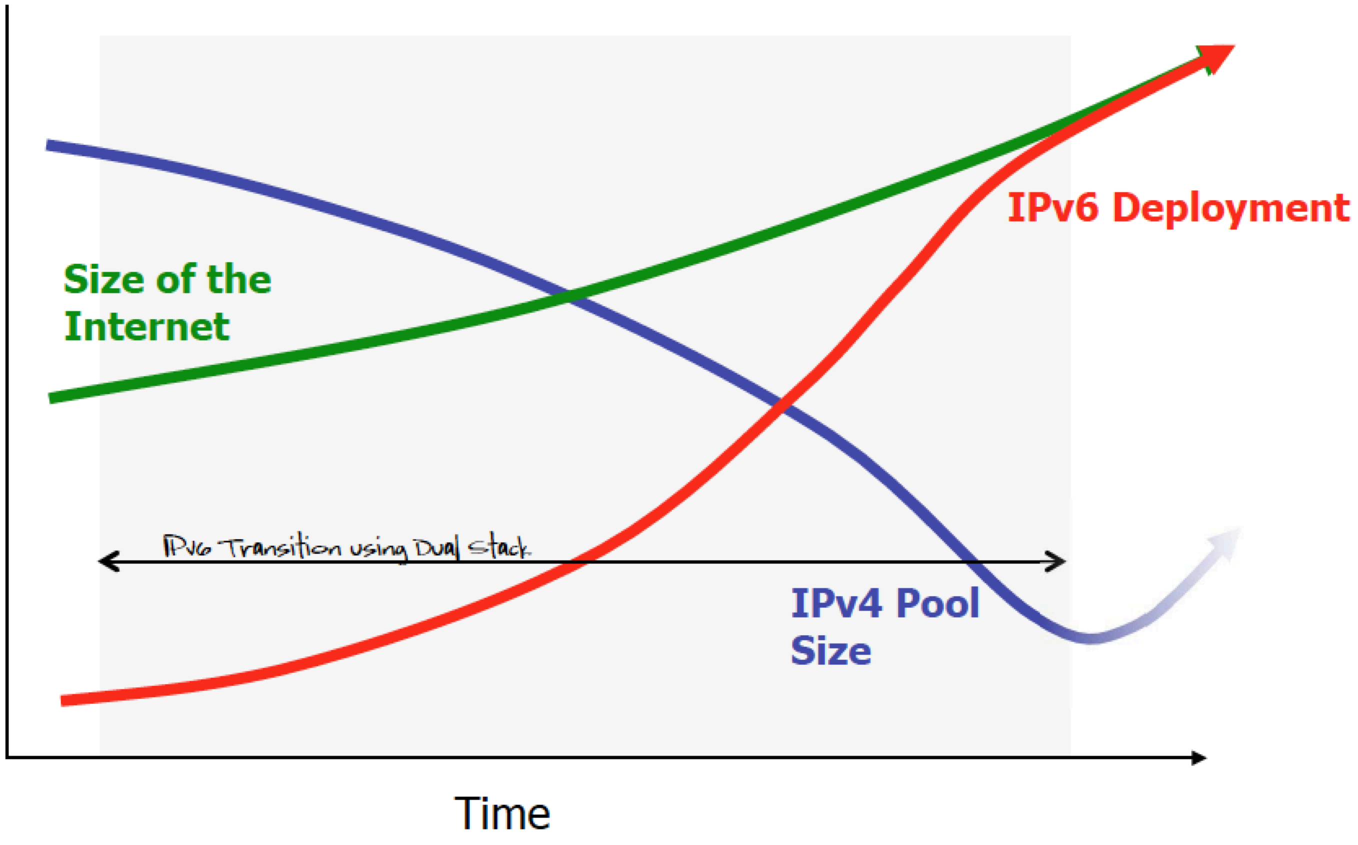


The IPv6 Timeline…



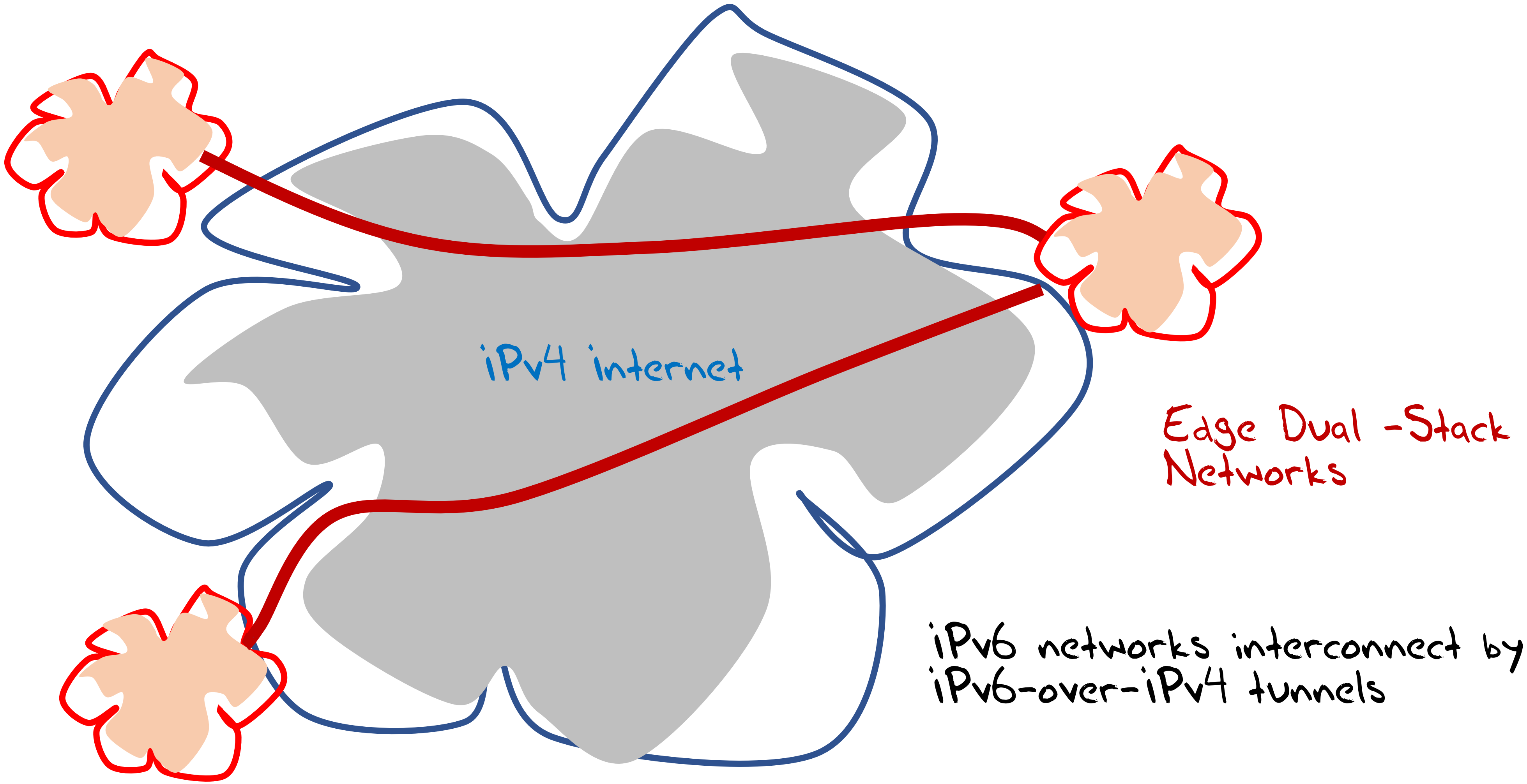
In-situ transition…

We had this plan …



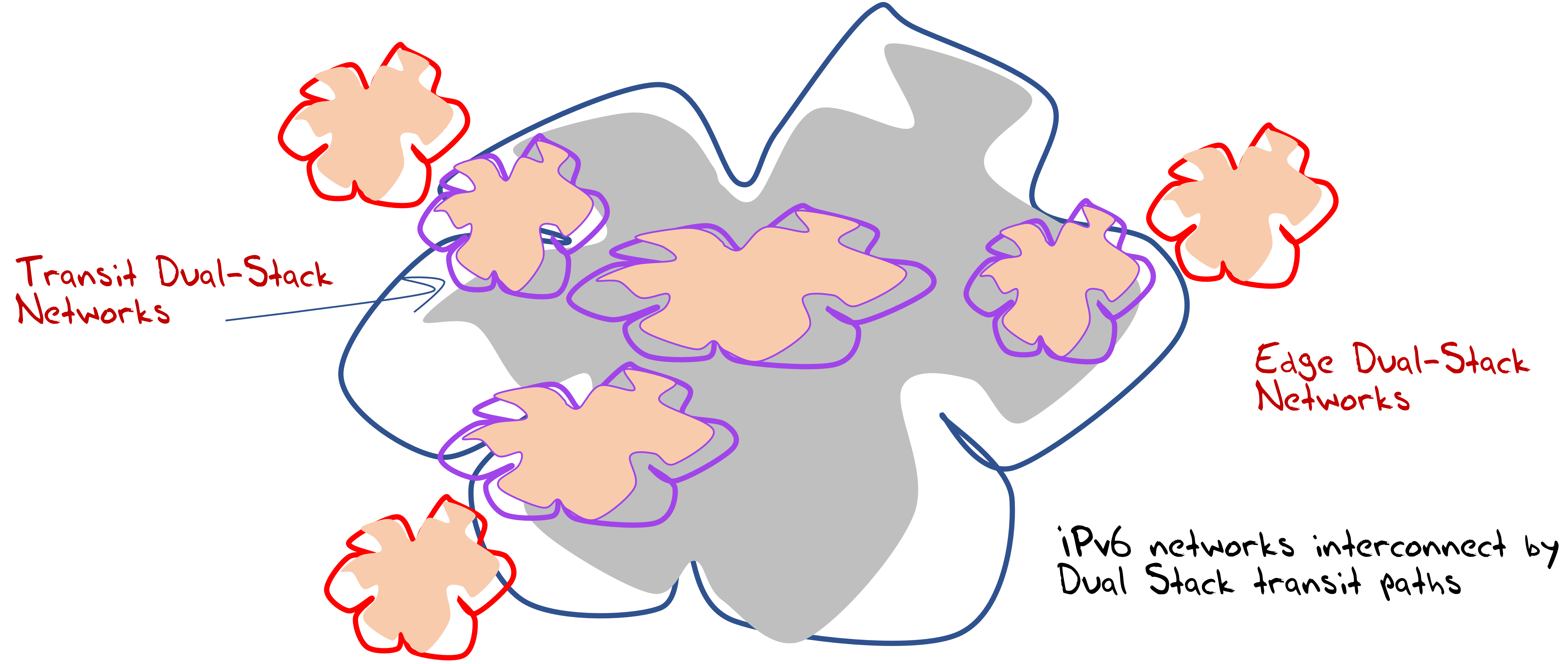
In-situ transition…

Phase 1 - Early Deployment



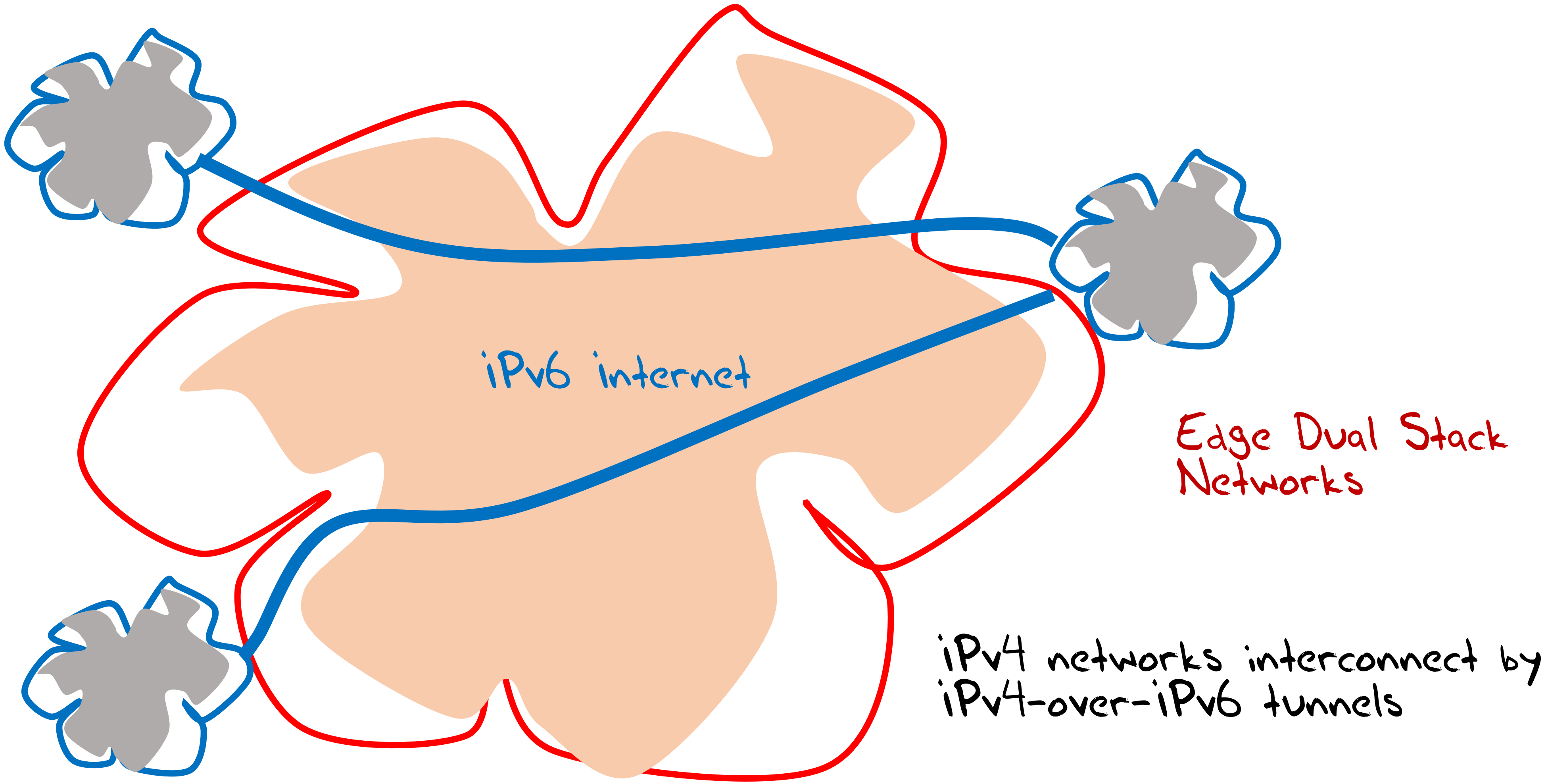
In-situ transition…

Phase 2 - Dual Stack Deployment



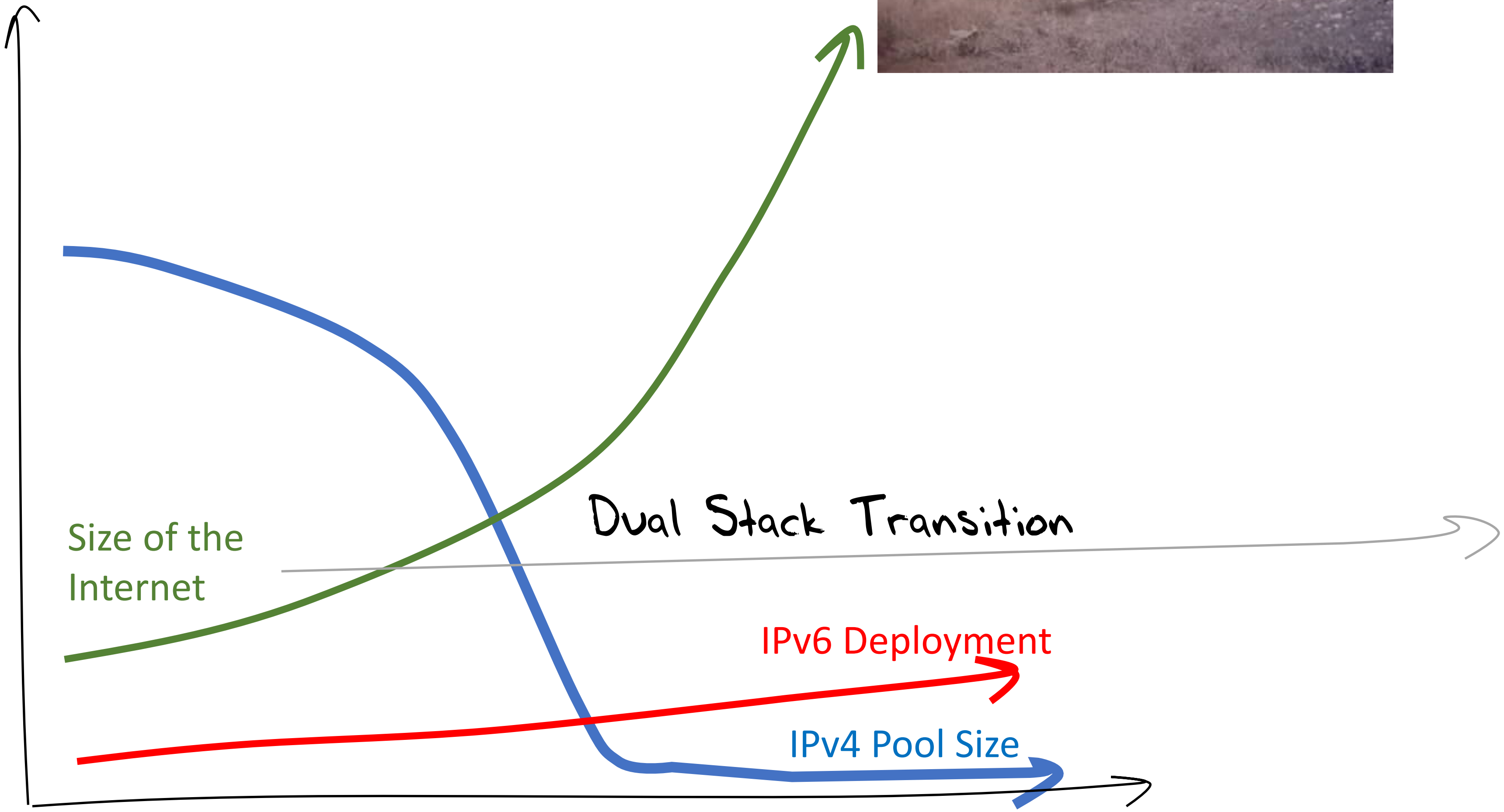
In-situ transition...

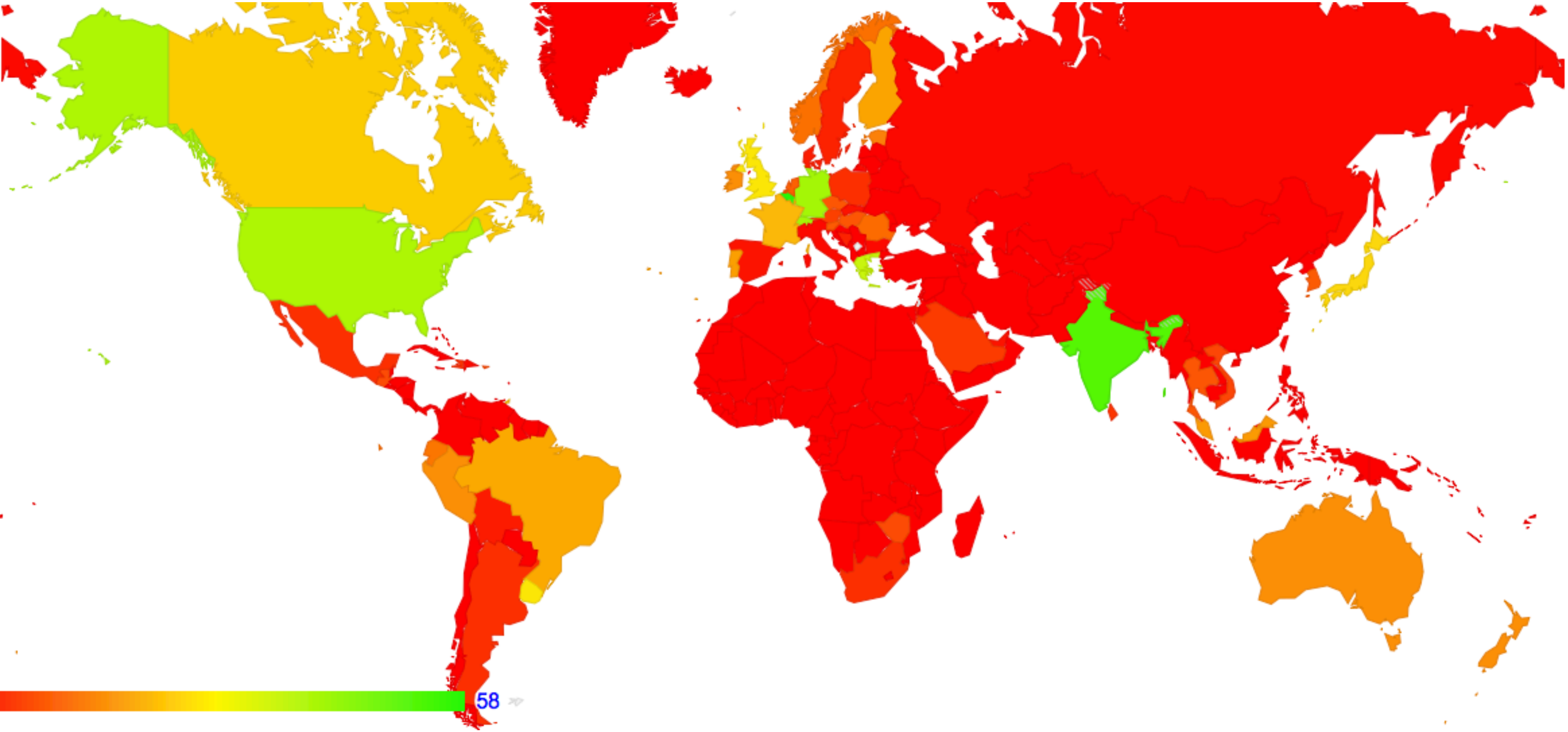
Phase 3 - IPv4 Sunset



In-situ transition…

We're pretty lousy at following plans!





The Map of iPv6 penetration – November 2017

Use of IPv6 for World (XA)

The Map of iPv6 penetration - November 2017

We're stuck in Phase 2

Some 15% - 20% of Internet users have IPv6 capability

In the IoT world the IPv6 numbers appear to be far lower than this

Most new IP deployments use IPv6+ (NATTED) IPv4

IPv4-only Legacy networks are being (gradually) migrated to dual stack

We're stuck in Phase 2

Some 15% - 20% of Int… …ability

In the … …s appear to be far lower

…w IP deployments use IPv6+ (NATTED) IPv4

IPv4-only Legacy networks are 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 cannot drop support for IPv4 as long as significant services and user populations do not support IPv6

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?

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…

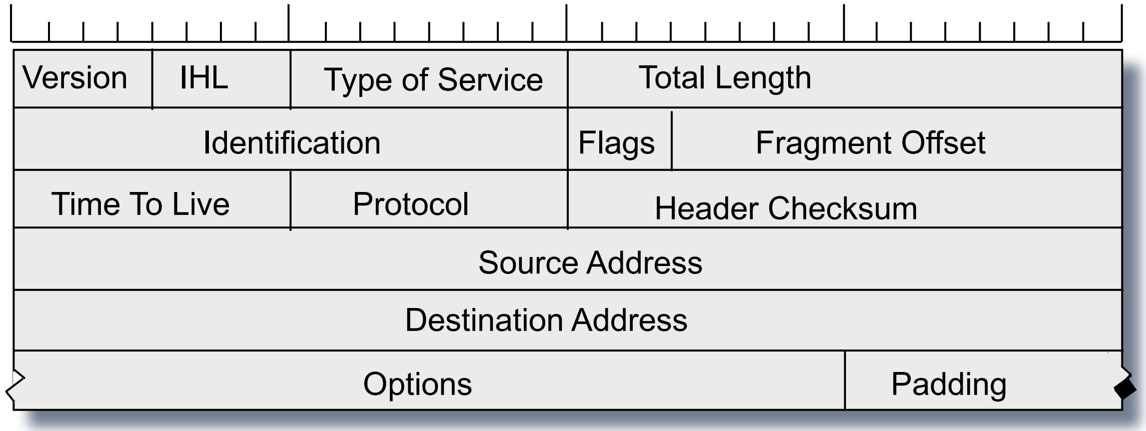
What do we rely on today in IPv4 that does not appear to have a clear working counterpart in IPv6?

If the answer is “nothing” then we are done!

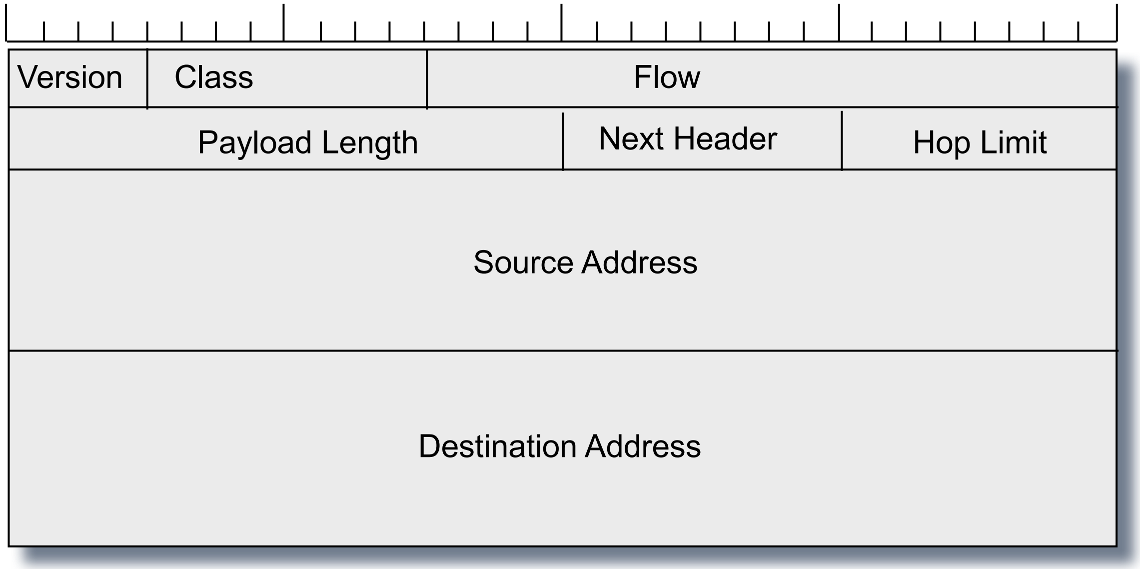
But if there is an issue here, then we should be working on it!

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 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

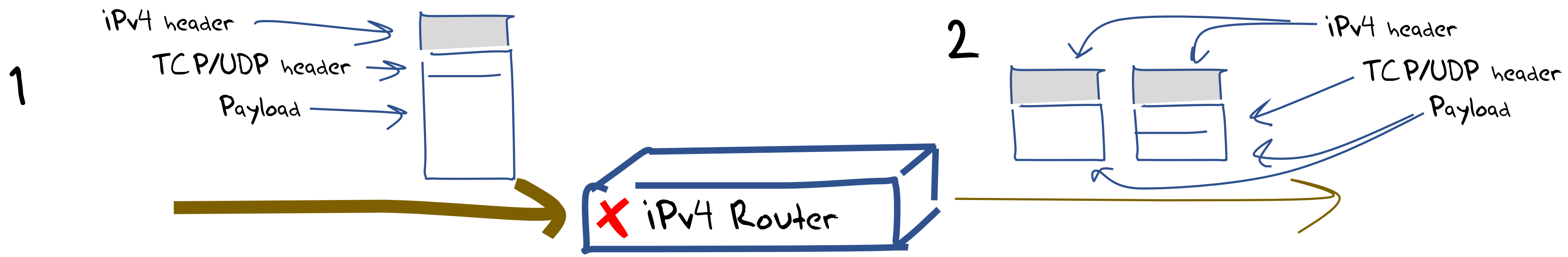
The only substantive changes with iPv6 are:

- the handling of fragmentation via Extension Headers
- cementing the Don't Fragment bit to ON for routers

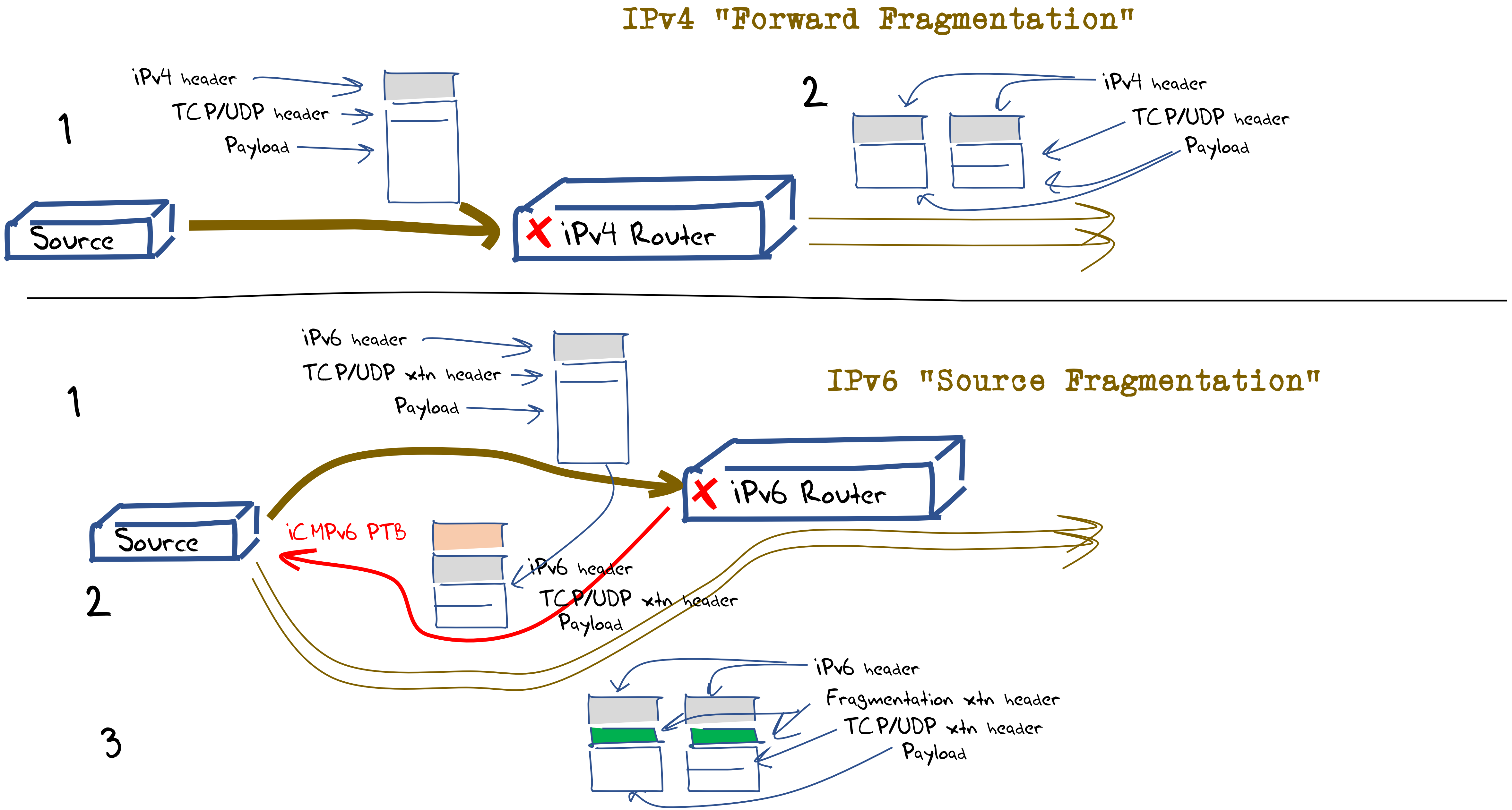
...layer function

IPv6: What changed?

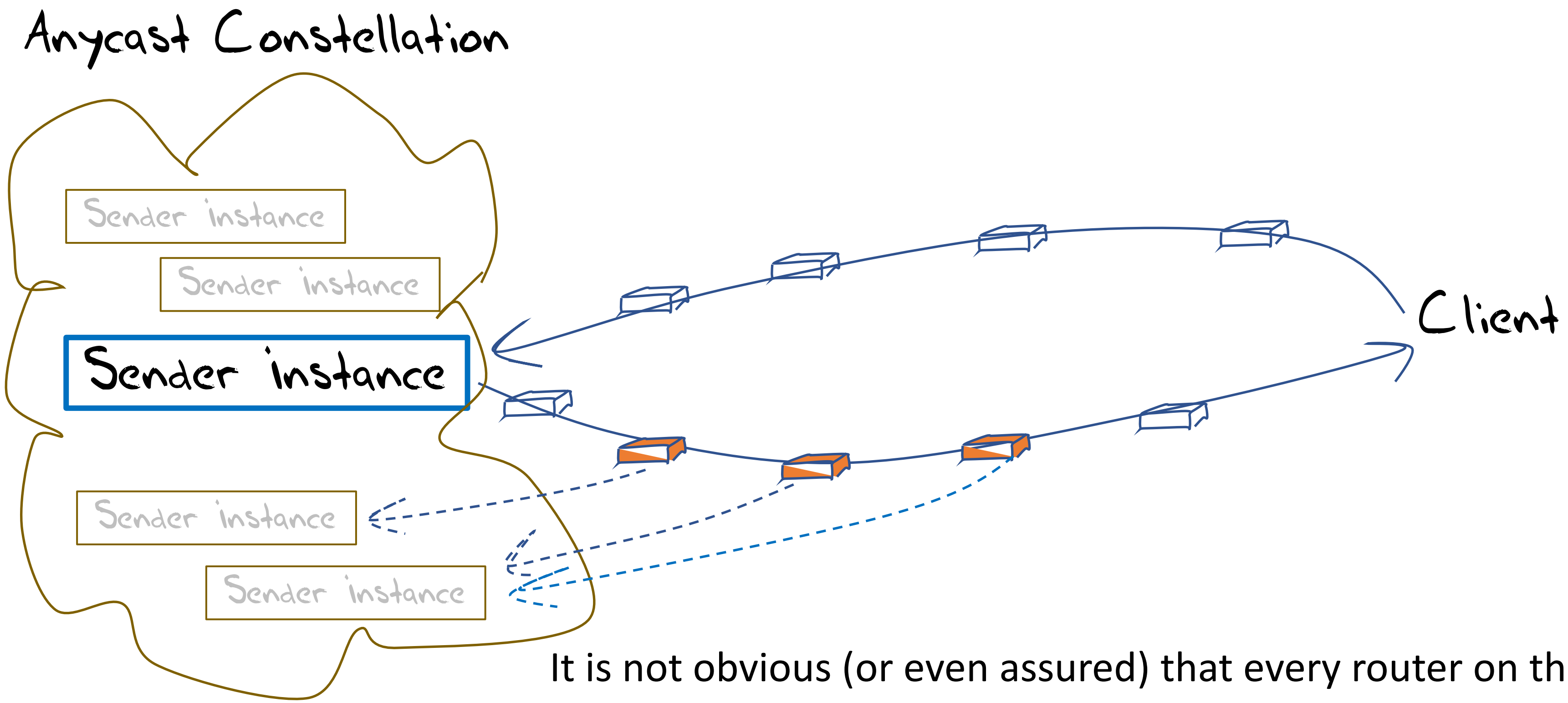
IPv4 "Forward Fragmentation"



IPv6: What changed?



ICMPv6 and Anycast



It is not obvious (or even assured) that every router on the 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!

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

Processing incoming ICMPv6 messages

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

For **UDP** – um, err, um well

Processing incoming ICMPv6 messages

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

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

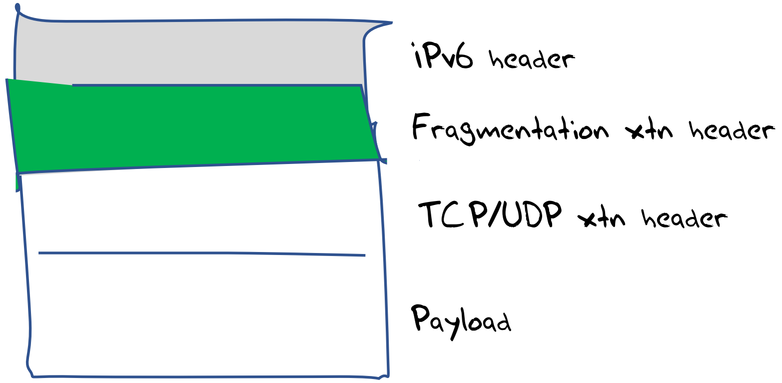
IPv6 and Fragmentation

The theory is that TCP in IPv6 should never send a fragmented packet - TCP should use the session MSS as a guide to packet sizes and segment the stream according to the session MSS

However, UDP cannot avoid fragmentation - large payloads in UDP simply need to be fragmented to fit within the path MTU

> Fragmentation in IPv6 uses the same control fields as IPv4 – a packet identifier, a fragmentation offset and a More Frags flag
>
> BUT they are located in an inserted “shim” that sits between the IPv6 packet header and the UDP transport header - this is an instance of the IPv6 “Extension Header”

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

RFC7872 – recorded drop rates of 30% - 40%

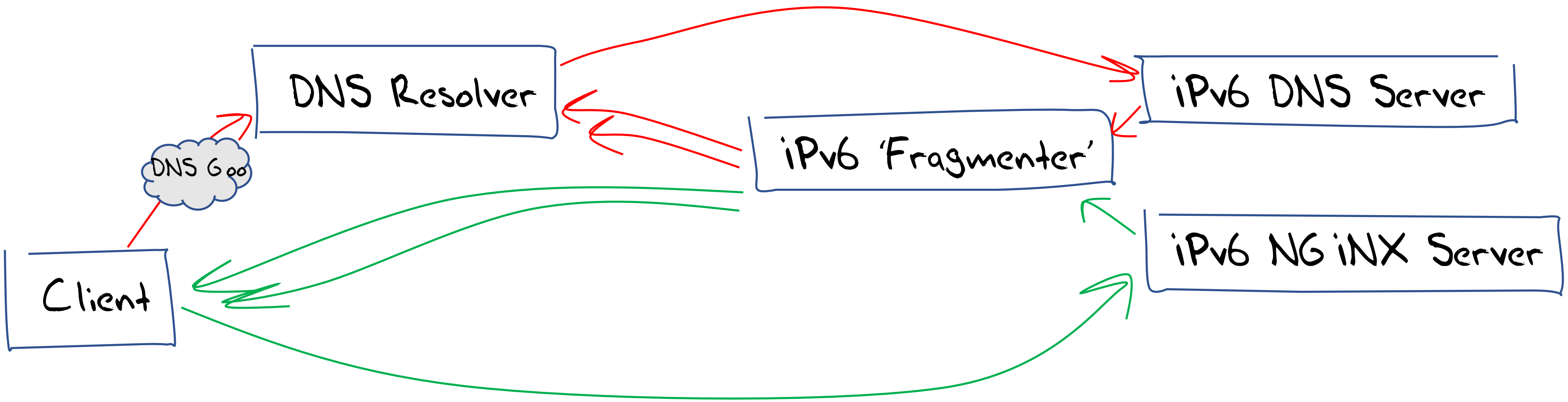
This experiment sent fragmented packets towards well-known servers and observed whether the server received and reconstructed the fragmented packet

But 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 **BACK** 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 te

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 were run across some 40M individual sample points in August 2017:

37% of end users who used IPv6-capable DNS resolvers could not receive a fragmented IPv6 DNS response

IPv6 Fragmentation Extension Header Handling

Our Experiments were run across some 40M individual sample points in August 2017:

37% of end users who used IPv6-capable DNS resolvers could not receive a fragmented IPv6 DNS 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 are few user-visible problems in a dual stack environment

This means that there is no urgent imperative to correct these underlying problems in deployed IPv6 networks

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 probl

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

option to flip to

IPv6.

The DNS does this, and Happy Eyeballs does this

So there are few user-visible problems in a dual stack environment

This means that there is no urgent imperative to correct these underlying problems in deployed IPv6 networks

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!

But what happens when there is no IPv4 left?

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!

But 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

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

We have to avoid IPv6 Fragmentation!

TCP can work as long as IPv6 sessions use conservative MSS sizes

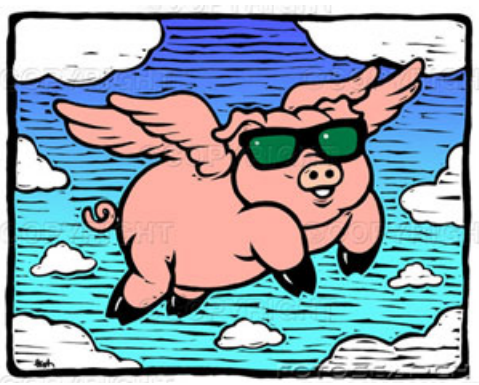
UDP can work as long as UDP packet sizes are capped so as to avoid fragmentation

Who needs to use large UDP packets anyway?

DNSSEC!

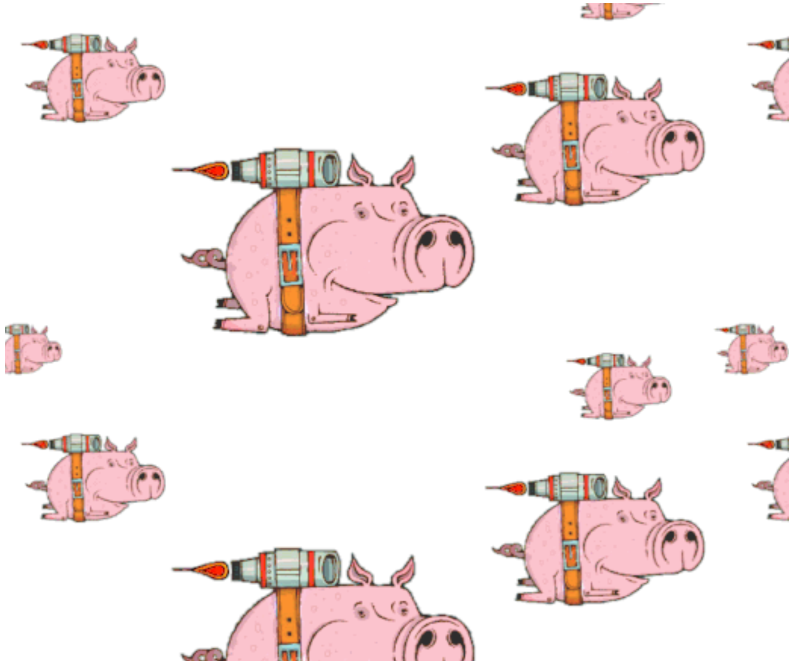
What can we do about it?

A. Get all the deployed routers and switches to deliver ICMPv6 packets and accept packets with IPv6 Fragmentation Headers



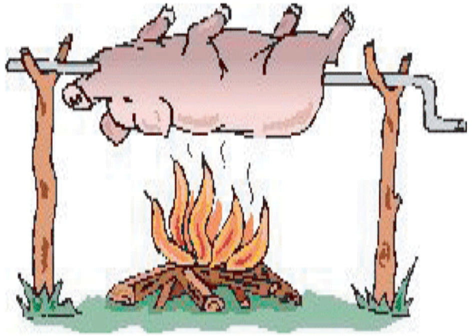
What can we do about it?

B. Get all the deployed routers and switches to alter the way iPv6 manages packet fragmentation



What can we do about it?

C. Move the DNS off UDP



Pick one?

All of these options have a certain level of pain, cost and potential inconvenience

Its hard to work out what is the best course of action, but it seems like a lot of extra effort if we take on all three at once!

For TCP …

Working around this issue in TCP can be as simple as a very careful selection of a default IPv6 TCP MSS

- Large enough enough to offer a tolerable data carriage efficiency
- Small enough to avoid Path MTU issues

And perhaps you might want to to support TCP path MTU discovery (RFC 4281)

For TCP …

But you have to take into account the observation that Path MTU discovery without reliable ICMPv6 signaling takes a number of Round Trip Times (delay)

And time is something no application designer has enough of to waste on probing path characteristics

So choose your TCP MSS very carefully

Hint: Smaller TCP MSS sizes are Better in IPv6!

For UDP …

- Working around this issue can be challenging with UDP
  - ICMPv6 Packet Too Big filtering causes silence
  - Fragment drop is silent drop
    - Which means that protocols need to understand timeouts

- An effort to work around this necessarily involves application-level adaptation to pass large responses without relying on UDP packet fragmentation

If we can't fix IPv6

- And we can’t fix end-to-end transport

- Then all that’s left is to look at the application protocol and see if we can re-define the protocol behaviour in a way eliminates fragmentation behaviour

"Old Style" DNS

- The original DNS protocol had this behaviour
  - If the DNS payload was <= 512 bytes send the answer over UDP
  - Otherwise send as much as will fit in 512 bytes set the truncate bit
  - The receiver is meant to re-query using TCP upon receipt of a truncated response
- Why did we change this behaviour?
- Because we thought that fragmentation was “safe” and TCP was too costly
- So we added Extension options for DNS to signal it was OK to send large fragmented UDP responses
- But its not OK

Large DNS Responses and IPv6

Change the protocol behaviour?

- Shift Additional Records into additional explicit UDP query/response transactions rather than bloating the original DNS response
- Perform UDP MTU discovery using EDNS(0) UDP Buffer Size variations as a probe
- Add a truncated minimal UDP response to trail a fragmented response (ATR)

Change the transport?

- DNS over TCP by default
- DNS over TLS over TCP by default
- DNS over QUIC
- Devise some new DNS framing protocol that uses multiple packets instead of fragmentation

Where now?

- We have a decent idea of the problem space we need to resolve
- We'd prefer a plan that allows each of us to work independently rather than a large scale orchestrated common change
- We're not sure we can clean up all the ICMPv6 filters and EH packet droppers in the IPv6 network
- And it sure seems a bit late in the day to contemplate IPv6 protocol changes
- Which means that we are probably looking at working around the problem by changing the behaviour of applications

What do the RFC's say?

What do the RFC's say?

Internet Engineering Task Force (IETF)
Request for Comments: 8085
BCP: 145
Obsoletes: 5405
Category: Best Current Practice
ISSN: 2070-1721

L. Eggert
NetApp
G. Fairhurst
University of Aberdeen
G. Shepherd
Cisco Systems
March 2017

UDP Usage Guidelines

Abstract

The User Datagram Protocol (UDP) provides a minimal message-passing transport that has no inherent congestion control mechanisms. This document provides guidelines on the use of UDP for the designers of applications, tunnels, and other protocols that use UDP. Congestion control guidelines are a primary focus, but the document also provides guidance on other topics, including message sizes, reliability, checksums, middlebox traversal, the use of Explicit Congestion Notification (ECN), Differentiated Services Code Points

What do the RFC's say?

Internet Engineering Task Force (IETF) L. Eggert
Request for Comments: 8085 NetApp
BCP: 145
Obsoletes
Category:
ISSN: 207

Abstract

The Us
transp
docume
applic
contro
provid
reliab
Conges

Due to these issues, an application SHOULD NOT send UDP datagrams that result in IP packets that exceed the Maximum Transmission Unit (MTU) along the path to the destination. Consequently, an application SHOULD either use the path MTU information provided by the IP layer or implement Path MTU Discovery (PMTUD) itself [RFC1191] [RFC1981] [RFC4821] to determine whether the path to a destination will support its desired message size without fragmentation.

However, the ICMP messages that enable path MTU discovery are being increasingly filtered by middleboxes (including Firewalls) [RFC4890]. When the path includes a tunnel, some devices acting as a tunnel ingress discard ICMP messages that originate from network devices over which the tunnel passes, preventing these from reaching the UDP endpoint.

What do the RFC's say?

Internet Engineering Task Force (IETF) L. Eggert
Request for Comments: 8085 NetApp
BCP: 145
Obsoletes
Category:
ISSN: 207

Abstract

The Us
transp
docume
applic
contro
provid
reliab
Conges

Due to these issues, an application SHOULD NOT send UDP datagrams that result in IP packets that exceed the Maximum Transmission Unit (MTU) along the path to the destination. Consequently, an application SHOULD either use the path MTU information provided by the IP layer or implement Path MTU Discovery (PMTUD) itself [RFC1191]

Applications that do not follow the recommendation to do PMTU/PLPMTUD discovery SHOULD still avoid sending UDP datagrams that would result in IP packets that exceed the path MTU. Because the actual path MTU is unknown, such applications SHOULD fall back to sending messages that are shorter than the default effective MTU for sending (EMTU_S in [RFC1122]). For IPv4, EMTU_S is the smaller of 576 bytes and the first-hop MTU [RFC1122]. For IPv6, EMTU_S is 1280 bytes [RFC2460].

endpoint.

What do the RFC's say?

Internet Engineering Task Force (IETF) L. Eggert
Request for Comments: 8085 NetApp
BCP: 145
Obsoletes
Category:
ISSN: 207

Abst

Due to these issues, an application SHOULD NOT send UDP datagrams that result in IP packets that exceed the Maximum Transmiss t (MTU) along the path to the destination. Conseq application SHOULD either use the path

Applications that d PMTUD
discover sult
in MTU
is s
tha S
in ytes and the
firs 1280 bytes [RFC2460].

This BCP is saying that using EDNS(0) in the DNS to signal the capability of accepting large fragmented DNS responses was unwise, and if a host/application does know the path MTU, it should truncate at UDP at 1280 octets (and iPv4 should truncate at 512 octets!)

But would that be enough?

- Is the root cause problem with the way our IPv6 networks handle Fragmented IPv6 packets?
- Or with the way our IPv6 networks handle IPv6 packets with Extension Headers?

- The data presented here suggests that EH drop could be the underlying significant issue here
- Perhaps we might want to think about advice to host stacks and applications to avoid EH altogether!
  - Including fragmentation!

What was the question again?

Oh yes, that’s right:

“Are we ready for an IPv6-only Internet?”

It appears that the answer is “no, not if we want the DNS to work!”

What was the question … ain?

Oh yes, that's right:

"Are w…

… is "no, not if we want the DNS to work!"

IPv6 Grade: B-

It's close, but more effort is needed if you want to properly finish this work!

Thanks!