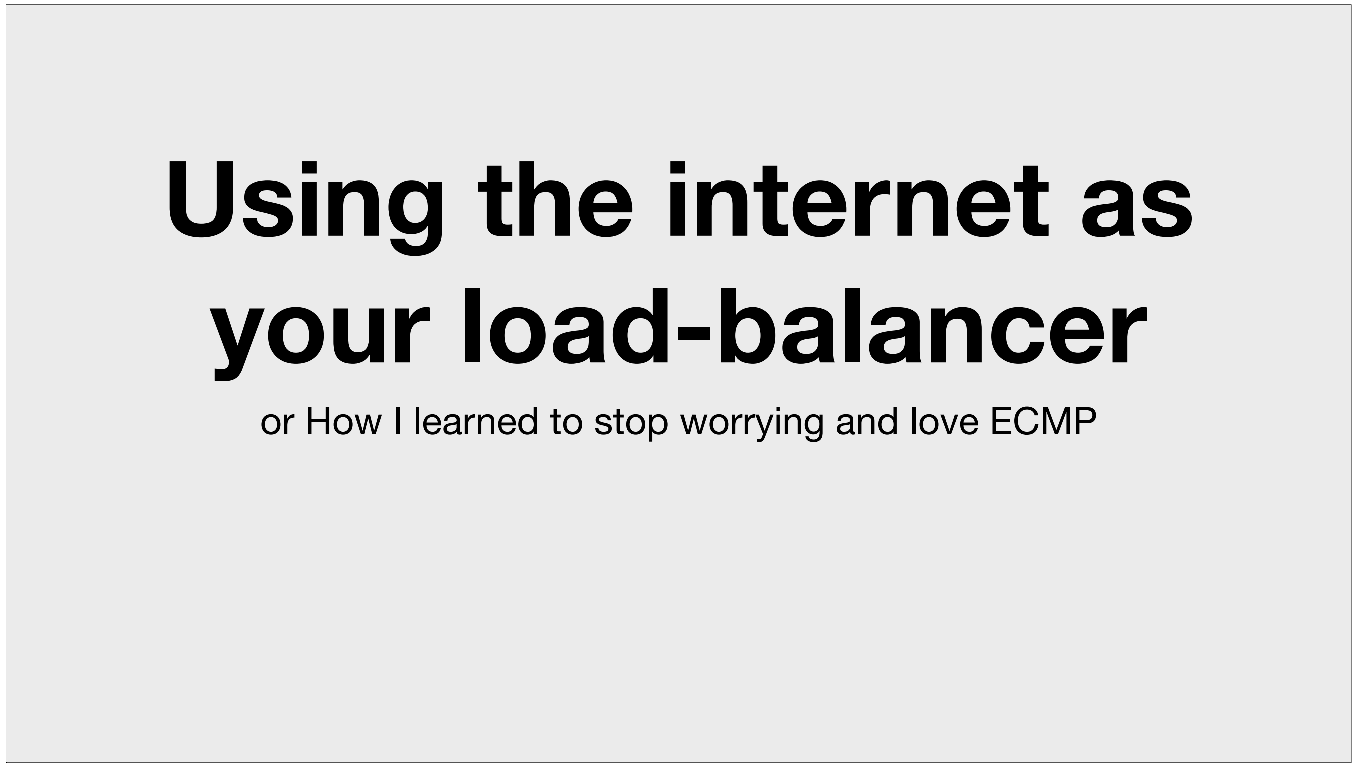


Using the internet as your load-balancer

or How I learned to stop worrying and love ECMP

Good afternoon SREcon.

We're almost at the end of an interesting three days but I hope you all have enough energy to stick with me as I try to shed some light on what it takes to use the internet as your load-balancer.

Martin Barry

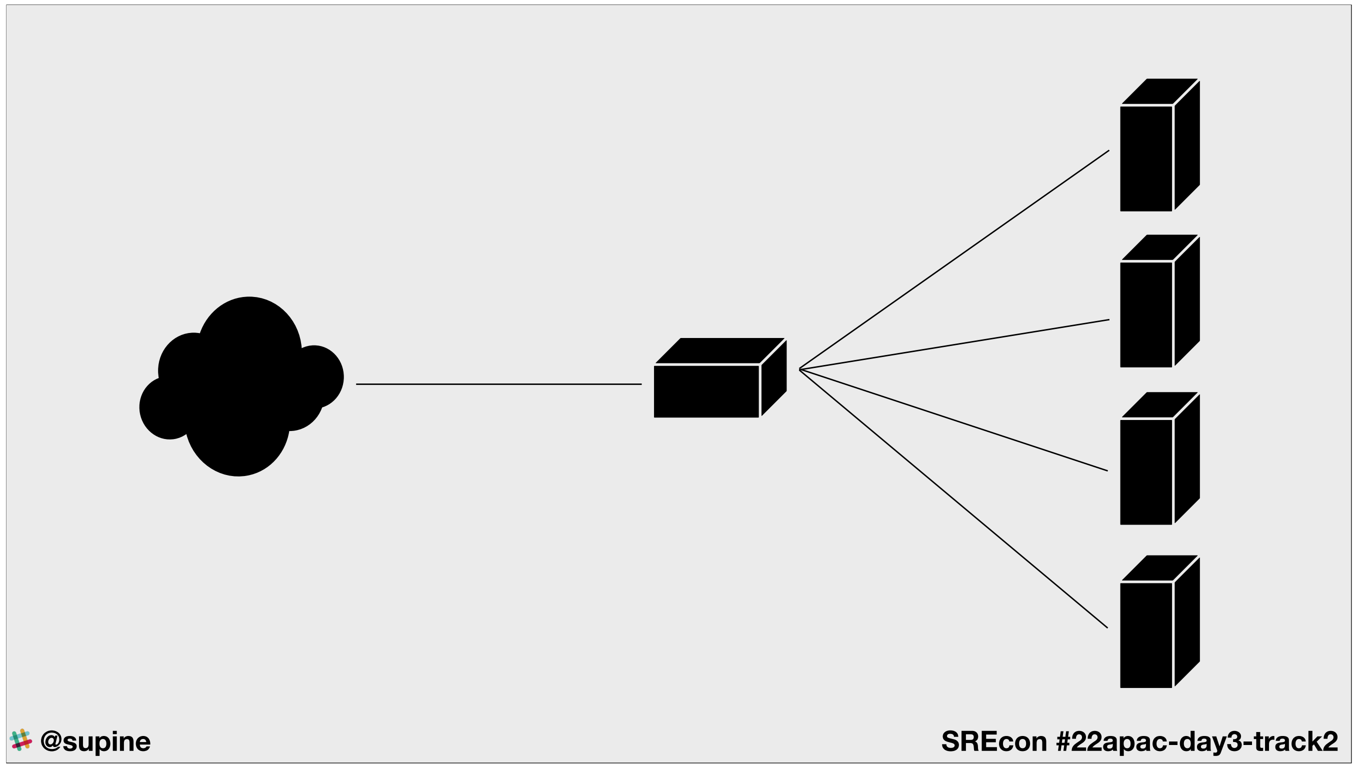
marty@supine.com

@supine

But first a little introduction…

I’m Martin, an Australian who moved to Germany 13 years ago.

I’ve worked in network and system administration for more than two decades across a variety of companies and industries.



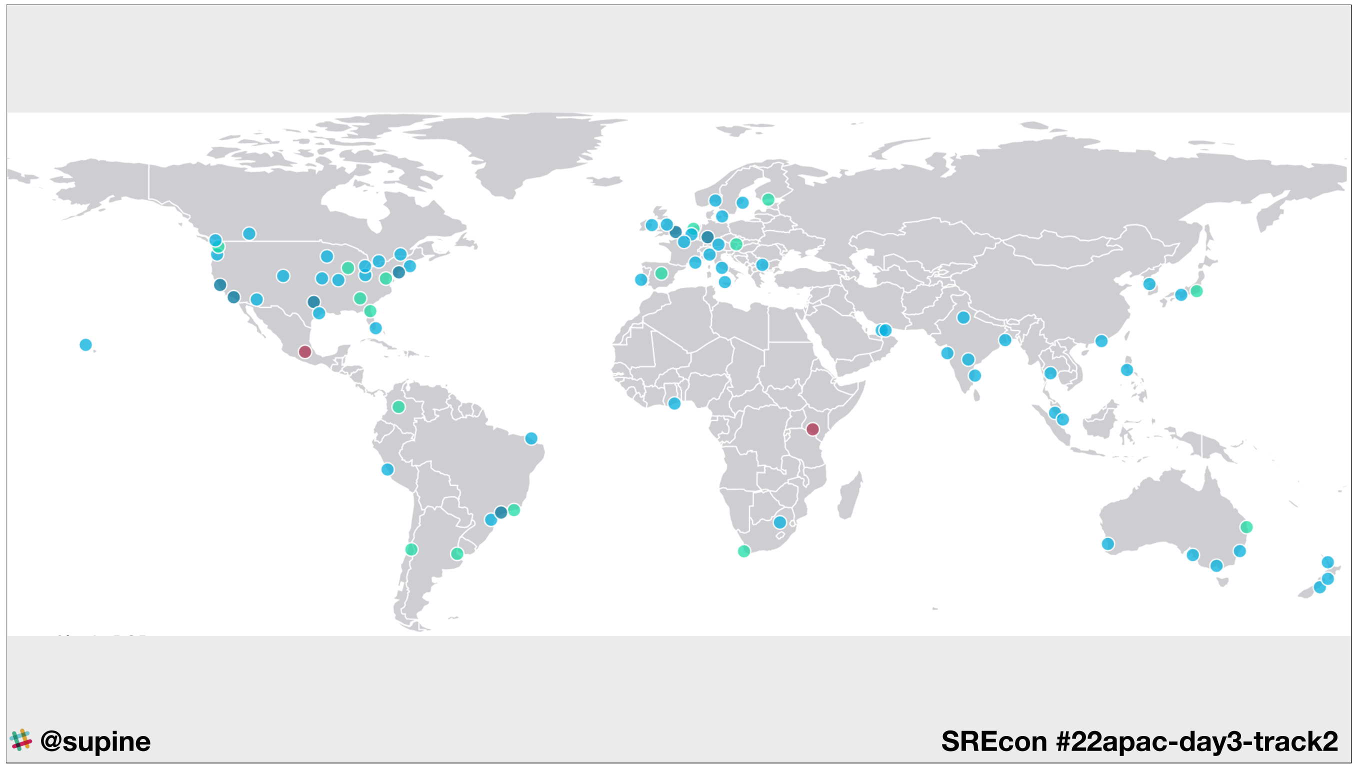
When someone talks about “load-balancing” this is usually what most people have in mind.

Work arrives at a self-contained system and is divided on to a set of resources.

The “load-balancer” might be a dedicated appliance or the function might be built in to networking equipment.

Back when I started in the industry the targets were usually individual physical machines.

These days they are more likely to be virtual machines or containers.



@supine

SREcon #22apac-day3-track2

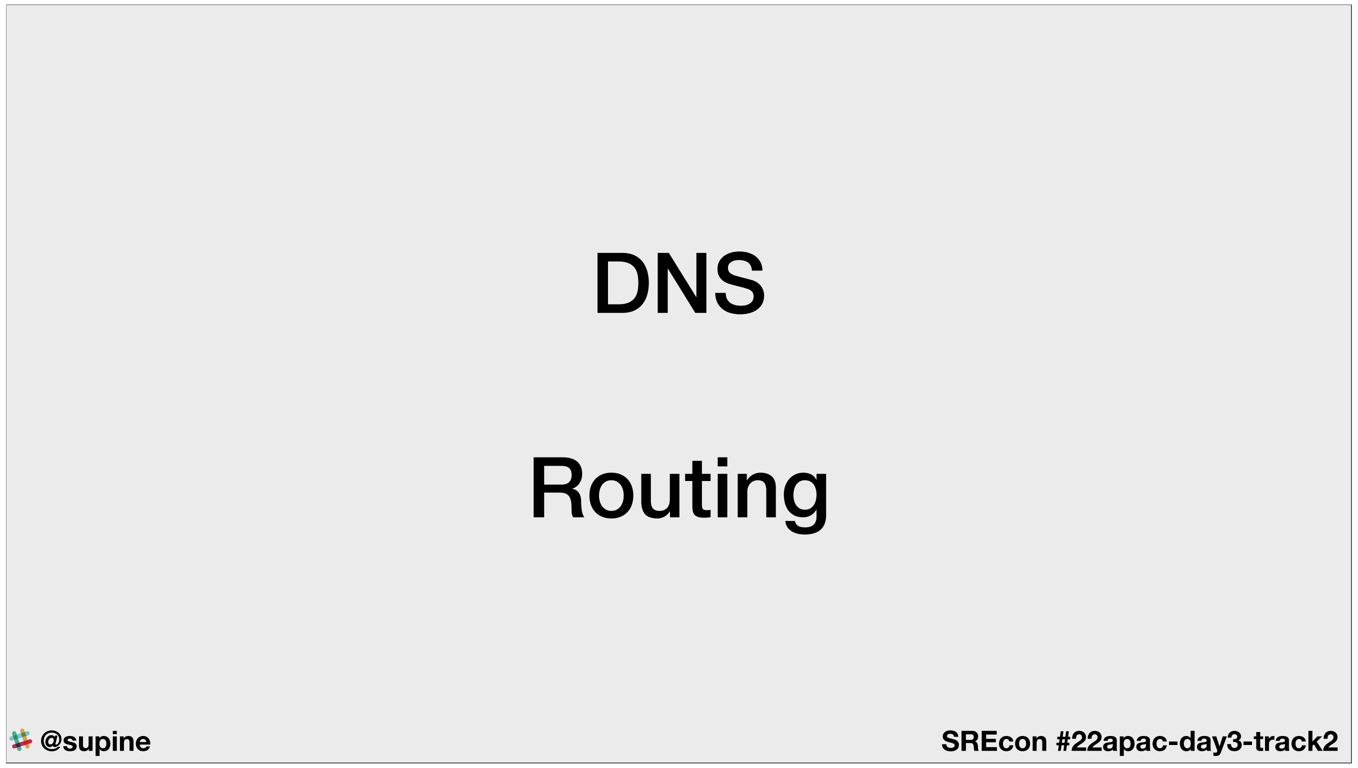
But what if your targets are not in one place…

What if the targets are spread around the globe, across continents and countries…

How do you control the work arriving on your service at internet scale?

For big companies this is the life of their edge team.

For others this is the kind of problem you outsource to a CDN.



DNS

Routing

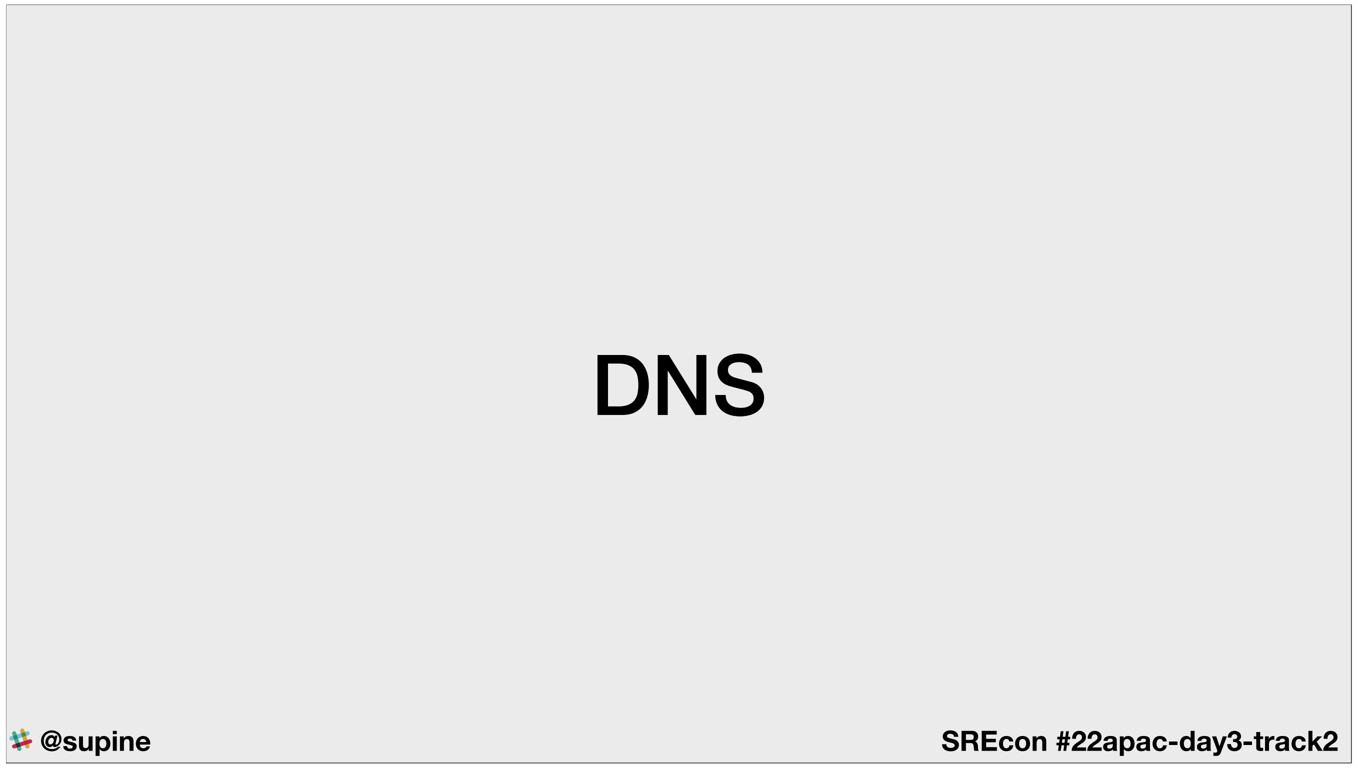
@supine

SREcon #22apac-day3-track2

So how do you load-balance at internet scale?

The most common solutions use a combination of DNS and routing.

If you have full control over the clients connecting to your service, like your own mobile application or single-page website, you can perhaps take advantage of other solutions, but if you don't control the client in that way then these two tools are how most folks solve this problem.

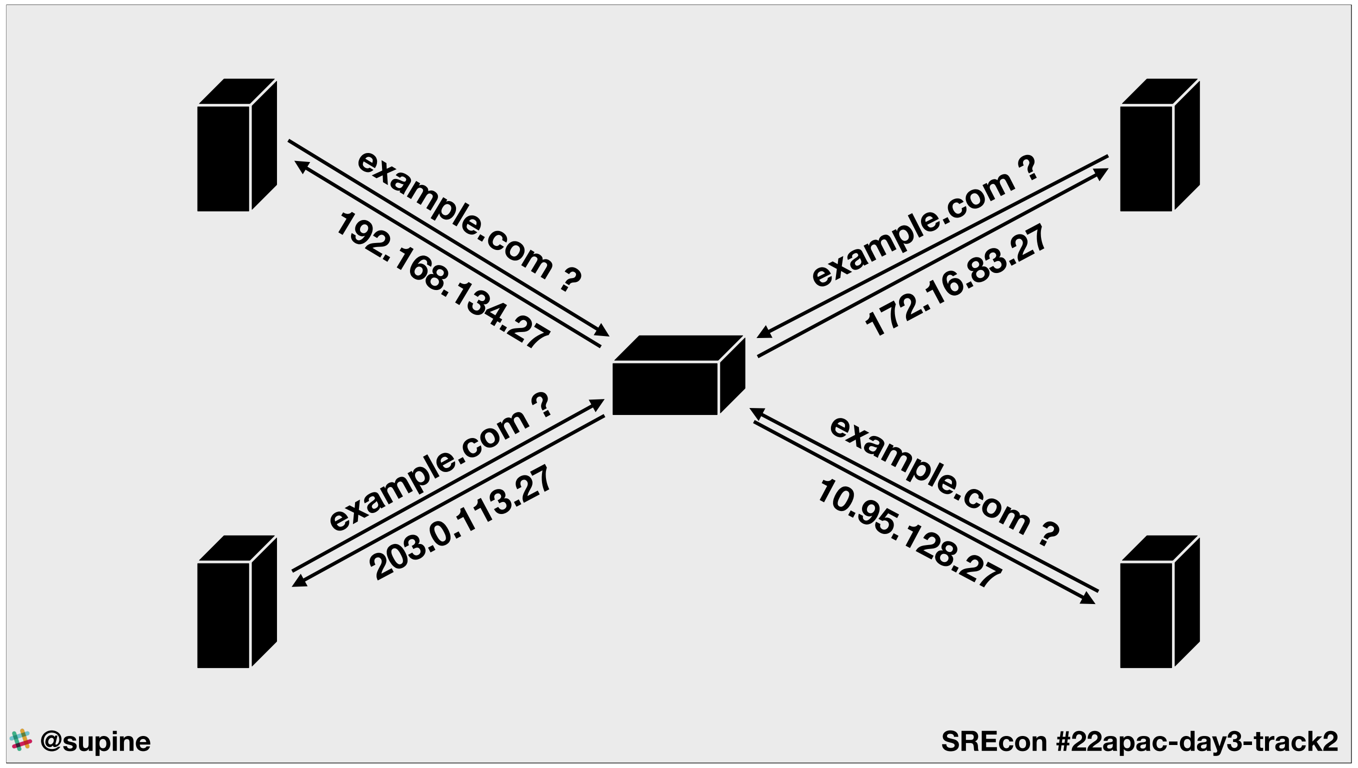


DNS

@supine SREcon #22apac-day3-track2

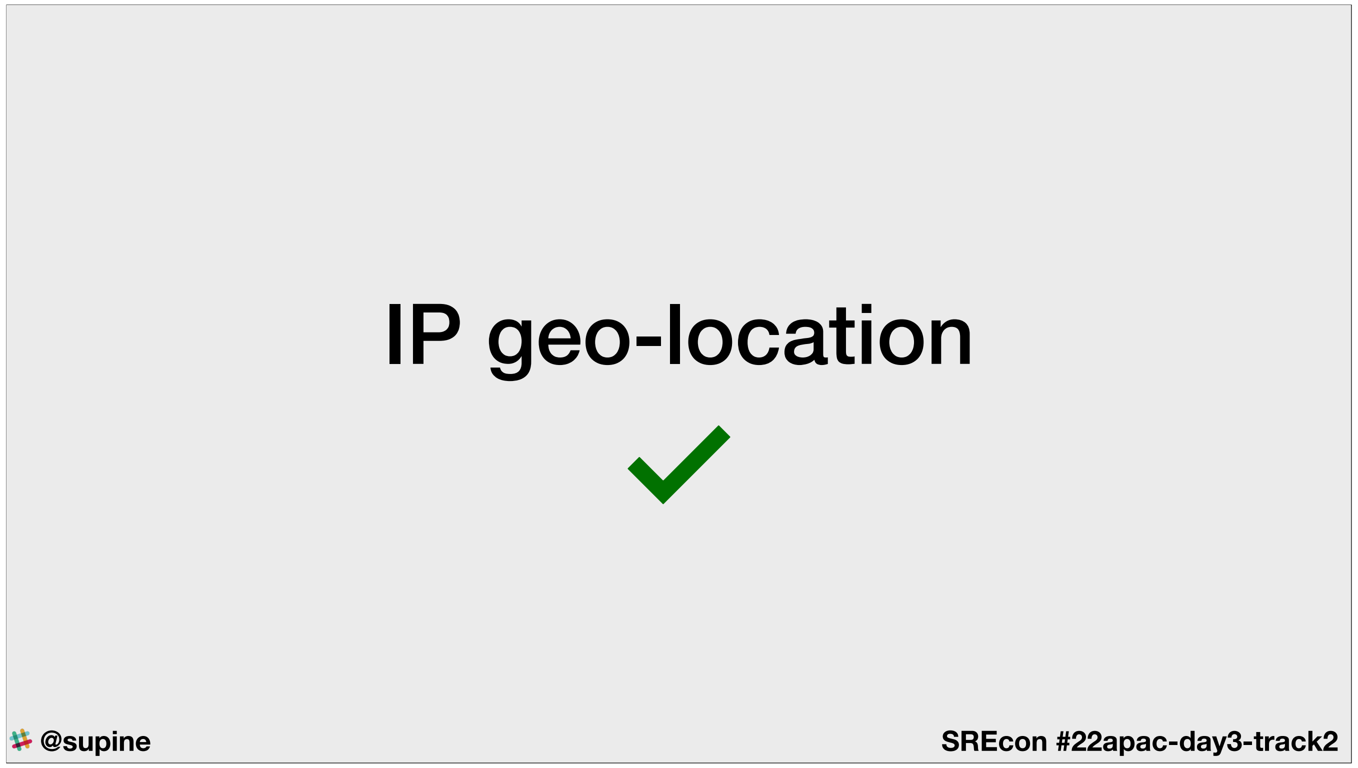
So let us start with DNS.

The concept of having a domain name where the IP varies when resolved is a fairly straightforward idea.



Whatever service is acting as your authoritative DNS responds with a different IP depending on who is asking.

The complexity is in how do you determine which IP to return and the tradeoffs you’ve made arriving at that decision.



IP geo-location

✓

@supine

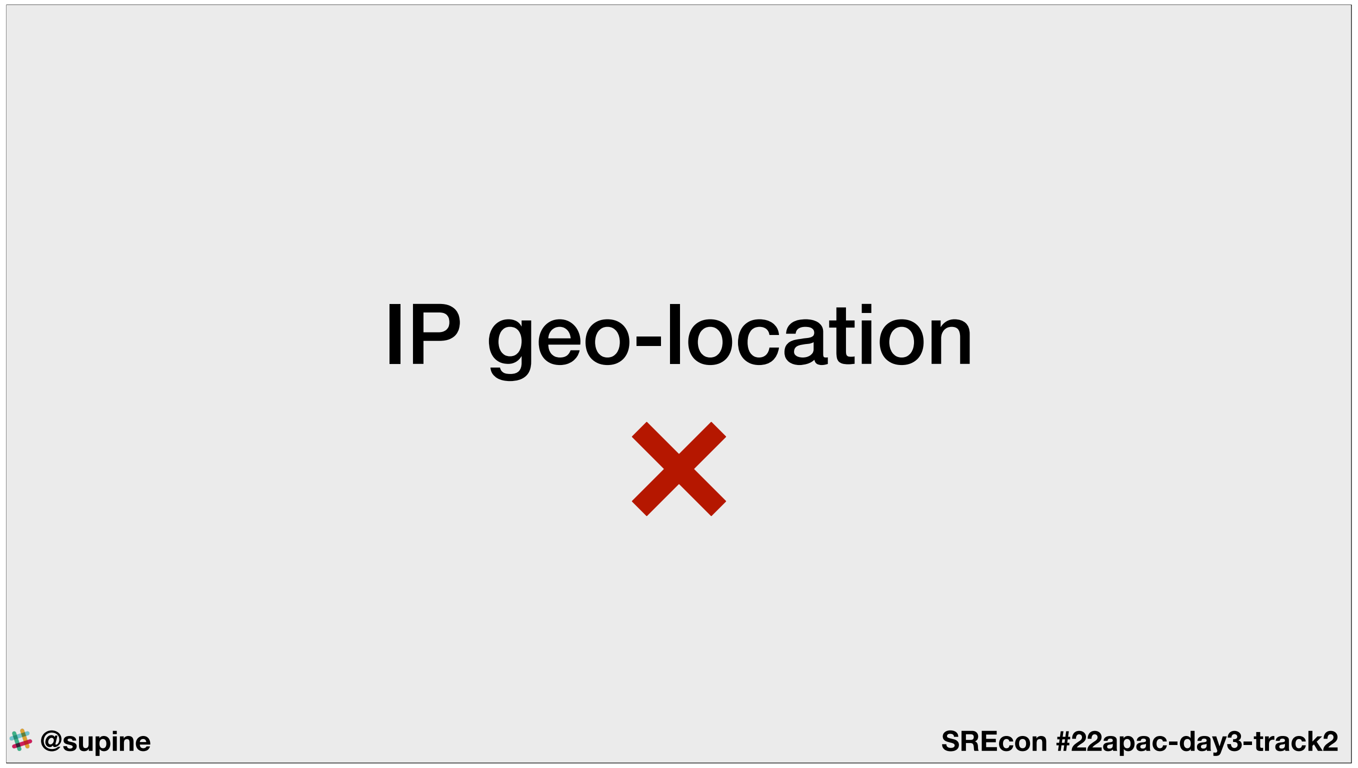
SREcon #22apac-day3-track2

One input to the decision can be IP geo-location.

There are databases available which supposedly map an IP to a location, down to the level of country, state, city or even postcode.

This data can allow you to make incredibly nuanced decisions, choosing an edge location based on physical and topological information.

However those decisions are only as good as the accuracy of the data.



IP geo-location

❌

@supine

SREcon #22apac-day3-track2

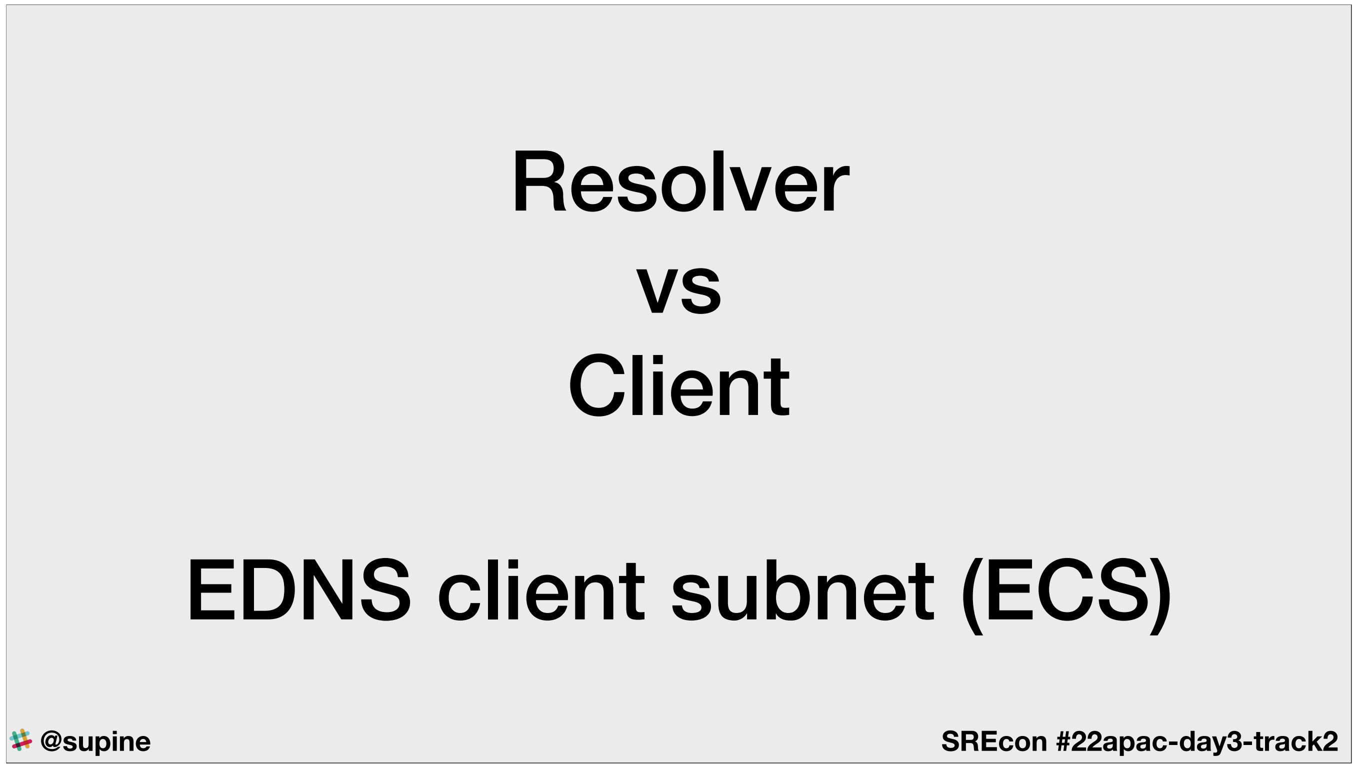
IPs can be reallocated to a different network

They can be repurposed by their current network

They can be dynamically assigned

Or the location starts off inaccurate, based on false assumptions or conclusions.

A common error is a large IP range being located at the address of the network's head office instead of where the IPs are being used inside their network.



Resolver
vs
Client

EDNS client subnet (ECS)

@supine SREcon #22apac-day3-track2

Another problem is you often only have the IP of the DNS resolver, not the client.

This is a problem if the resolver location is not representative of the client location, either because of how the ISP has deployed their resolver infrastructure or because the client is using a third party resolver like OpenDNS or Google's Public DNS.

If the resolver supports EDNS client subnet (ECS), something that gives you a hint of the client IP, the response can perhaps still be optimised.

But not all resolvers support ECS and not all clients want it due to privacy concerns.

Authoritative server location

@supine

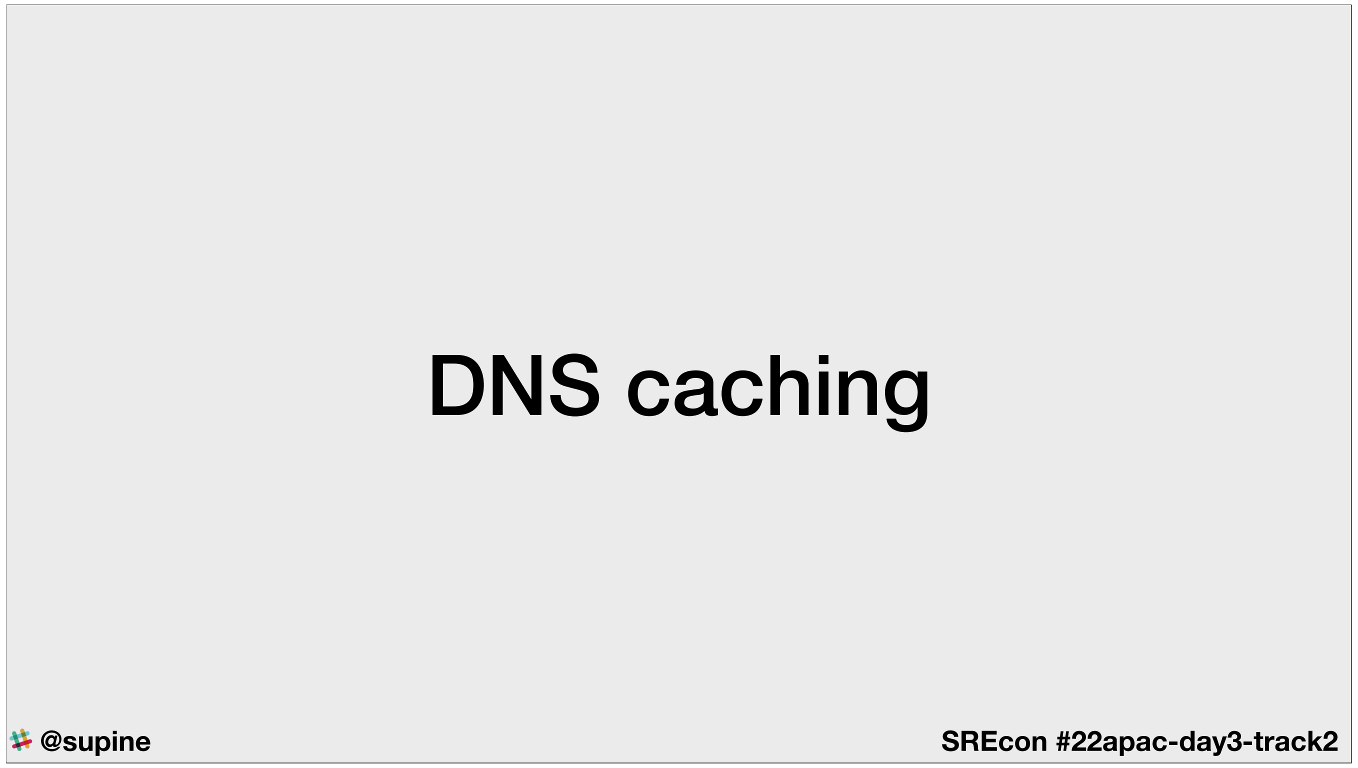
SREcon #22apac-day3-track2

An alternative input to the decision can be where the DNS request is processed.

Presuming that the subsequent connection will come over a similar path to the DNS request, the IP you reply with can be based on the location of the authoritative DNS server.

However this requires that your authoritative server locations are aligned with your edge locations. Not so hard if you are running both yourself but tricky if either of them, or both, are outsourced.

This usually only tells you something about the location of the DNS resolver, not the client. Again, this is a problem if the resolver location is not representative of the client location


DNS caching

@supine

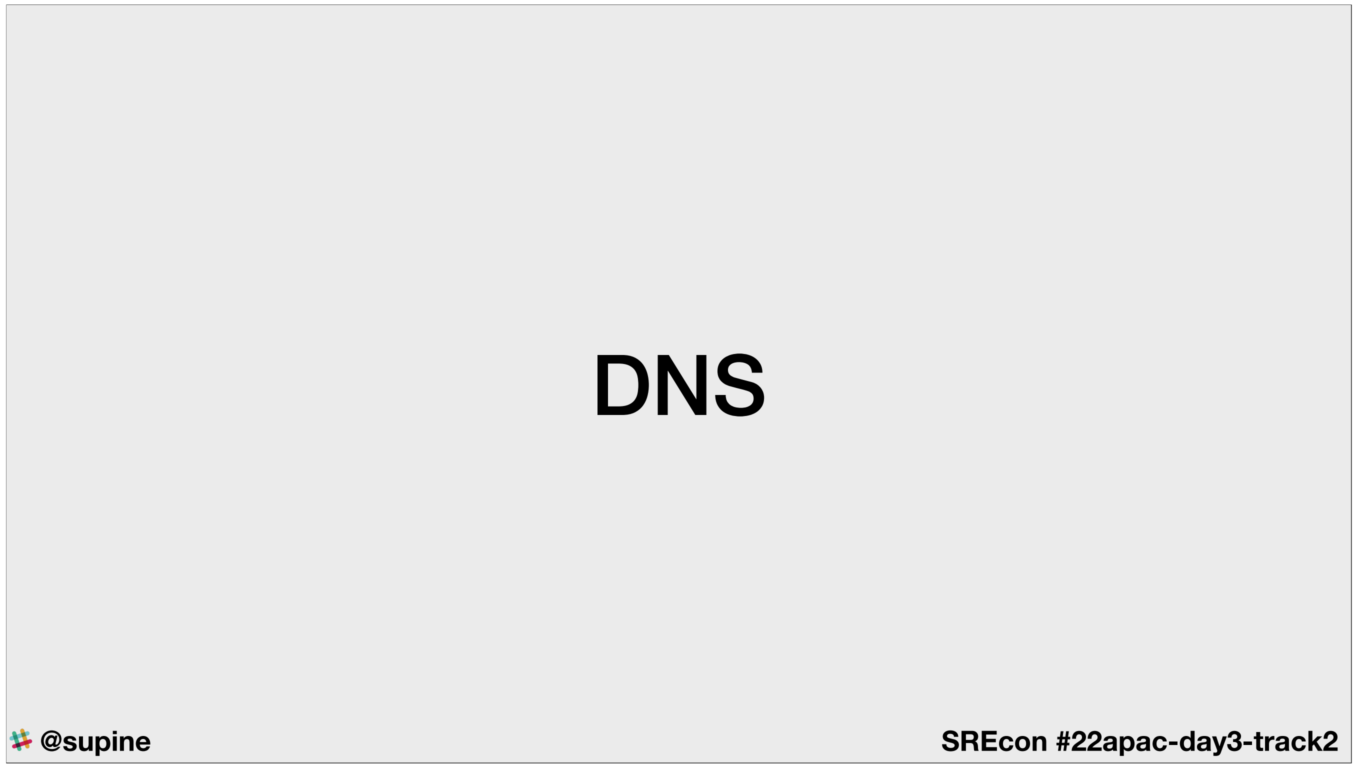
SREcon #22apac-day3-track2

There are also some concerns in the basic idea of a DNS solution.

DNS responses are generally cached at the client and resolver to reduce query load.

Even though you can suggest how long they should cache the response, you can’t stop them enforcing a minimum time or ignoring your suggestion completely.

If you want to change the IP you would respond with, perhaps due to something failing, the caching can delay the propagation of the new IP to clients.



DNS

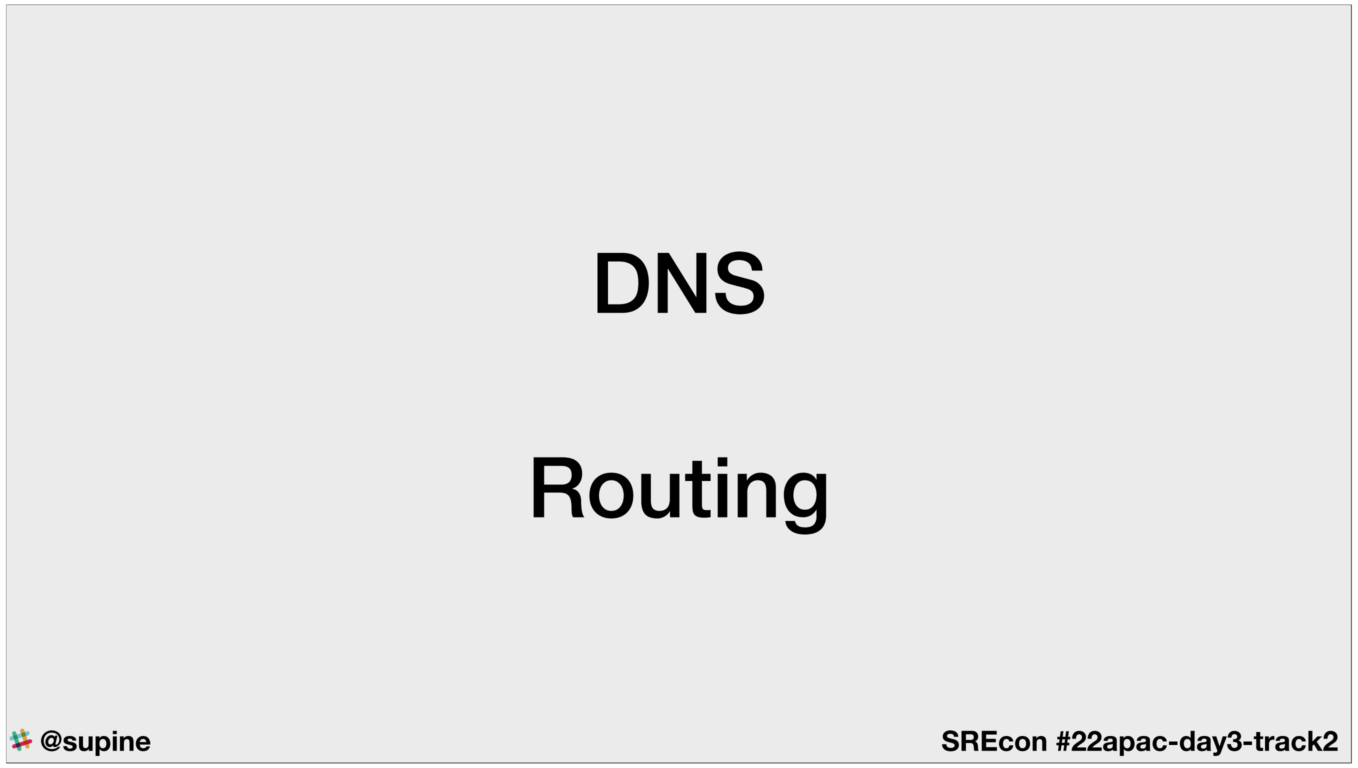
@supine

SREcon #22apac-day3-track2

It’s not all bad news for DNS.

This kind of solution is generally quite stable, the client will connect to the same edge location until you change the DNS response.

It has the potential to give you good control, to direct requests with great nuance, particularly if you have ECS hints to work with.



DNS

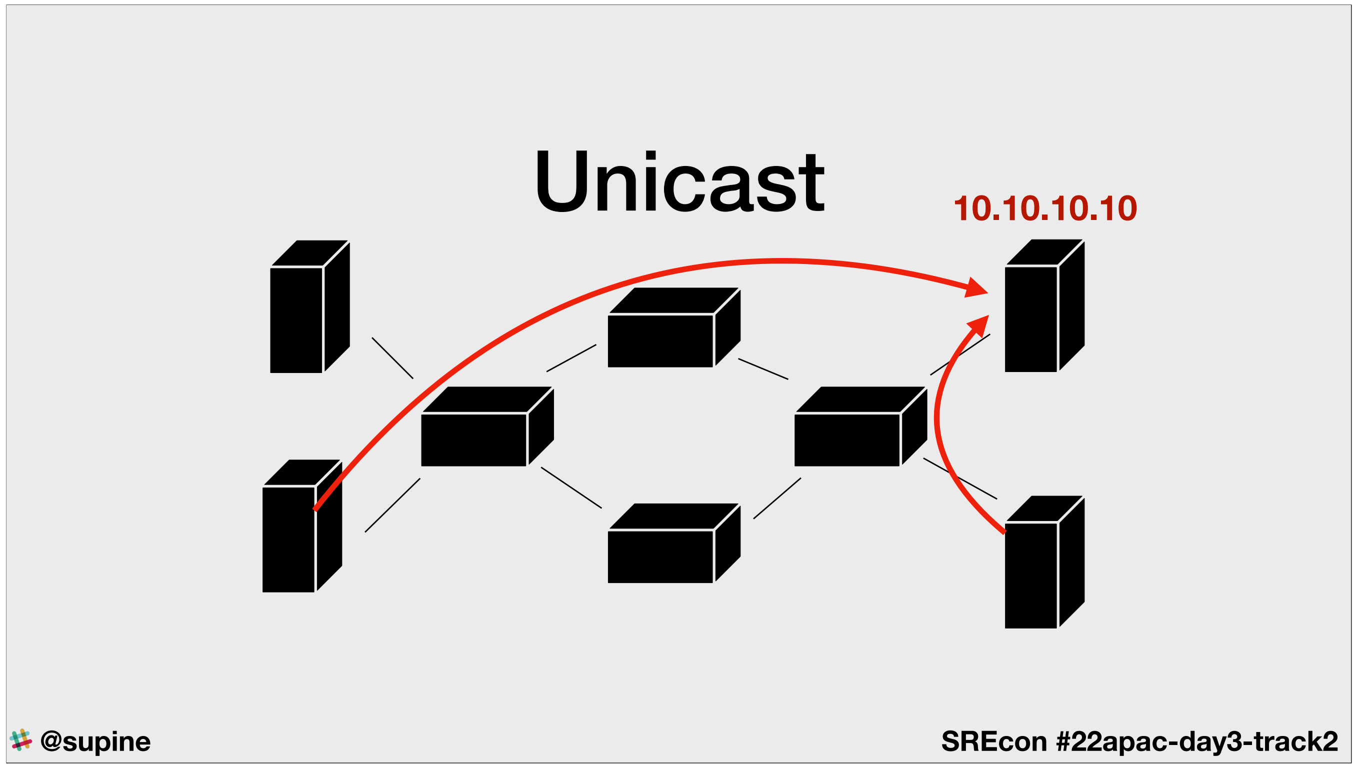
Routing

@supine

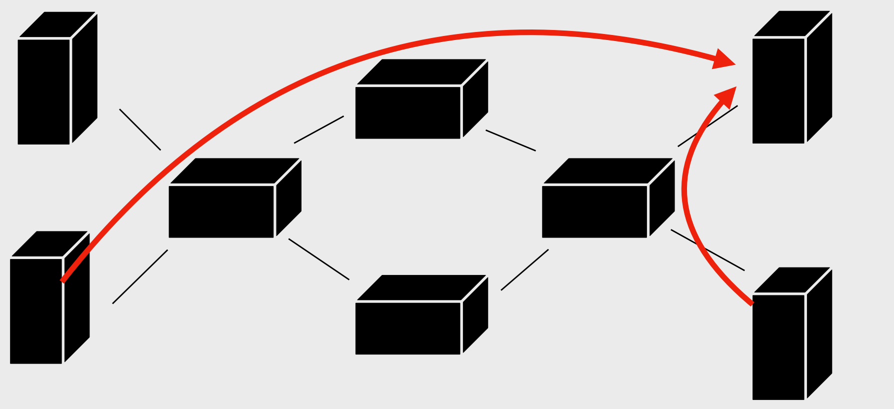
SREcon #22apac-day3-track2

So to quickly recap, varying DNS responses is one method to direct requests but, while the concept is simple, the implementation details and operational complexities can cause problems.

An alternative load-balancing solution is using the internet routing system to guide requests to an edge location.



Unicast



@supine

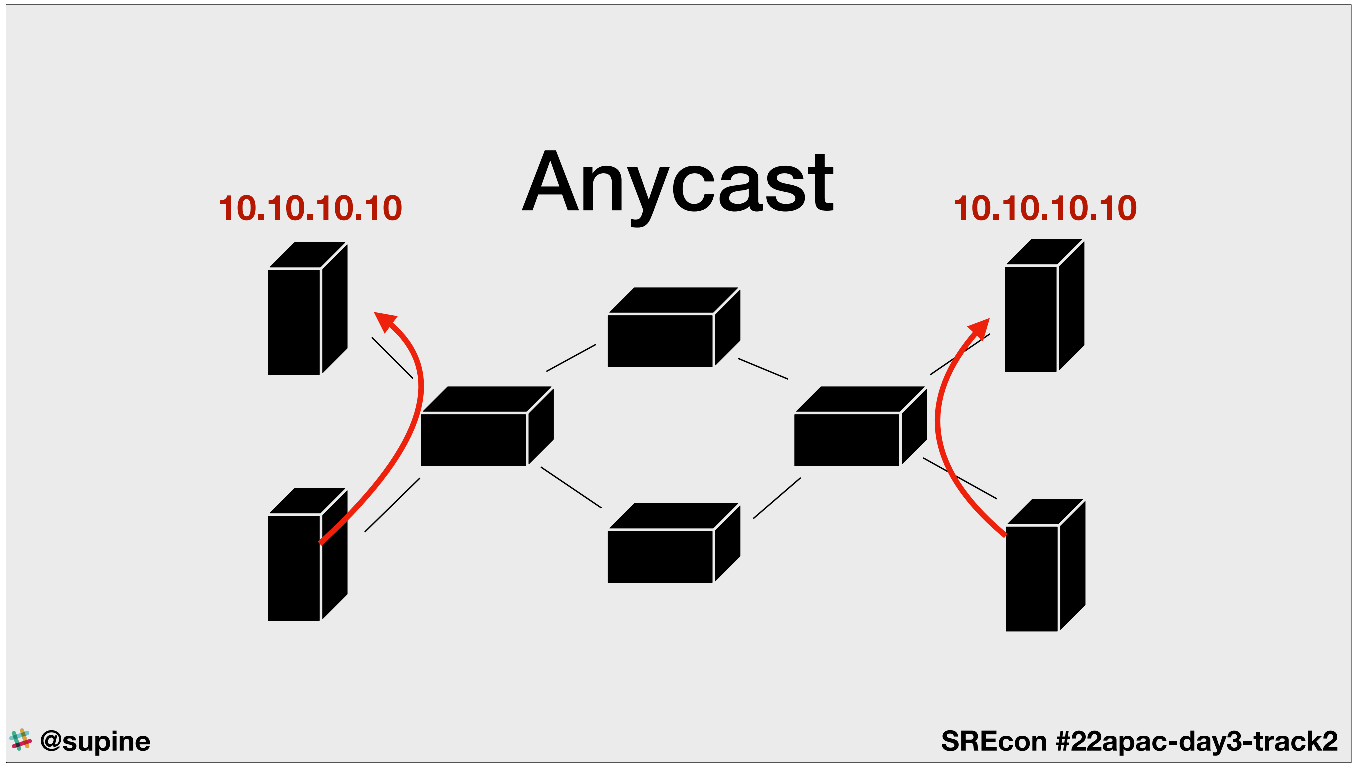
SREcon #22apac-day3-track2

One of the important routing concepts is anycast.

But to understand anycast it's helpful to first talk about the opposite, understand what unicast is.

Most routing occurs to a specific unique destination.

Packets headed towards that IP are all going to the same place.



Anycast

@supine SREcon #22apac-day3-track2

The alternative is anycast. This is when an IP is advertised from multiple locations, any of which can serve the request and routing will select where the packets arrive.

Anycast is used by many UDP based protocols, the best example being DNS.

ISPs will use anycast inside their network so your configured resolver IP will always be a DNS server or cluster not too far away.

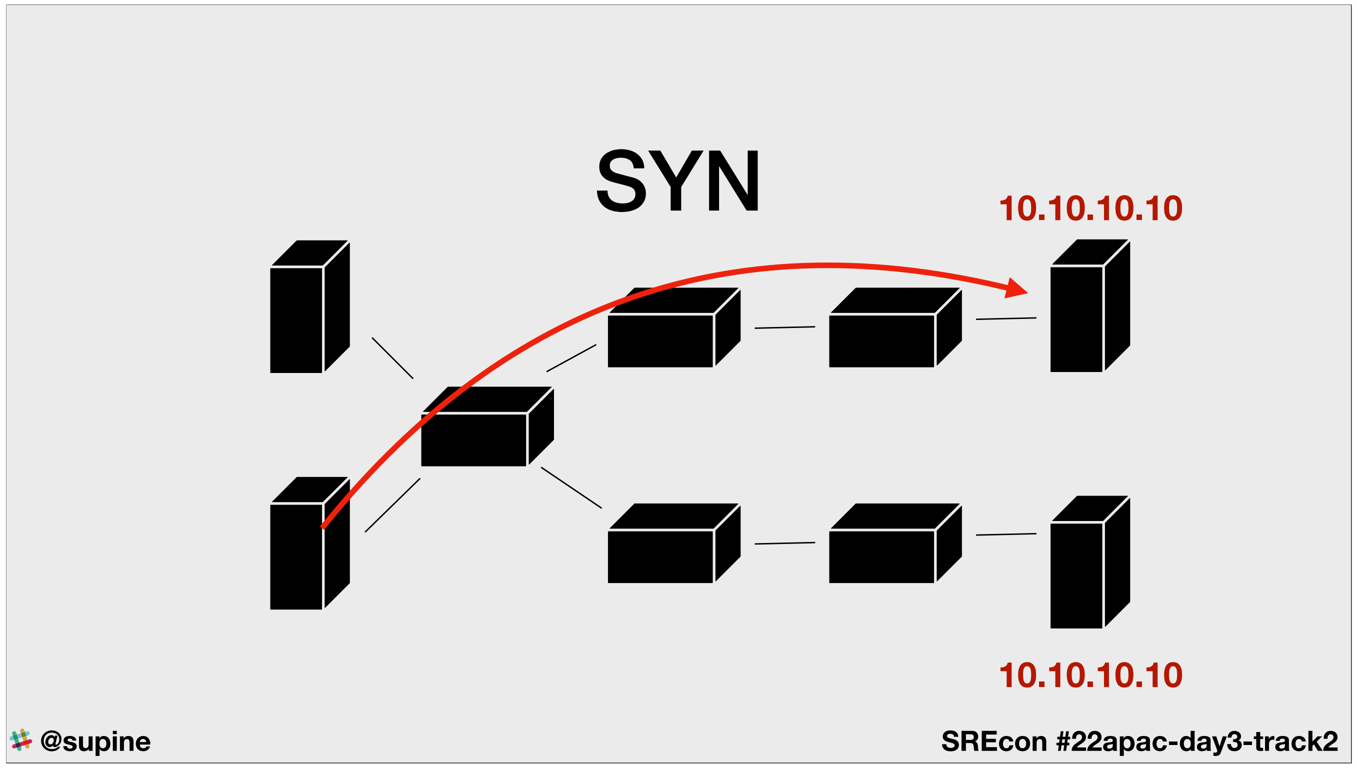
DNS root servers use anycast so that, despite only thirteen configured IPs, the service is provided by hundreds of servers around the world.

TCP Anycast

@supine

SREcon #22apac-day3-track2

But while a connectionless protocol like UDP works just fine with anycast, a connection oriented protocol like TCP requires that the routing decisions are stable enough to send packets to the same anycast location for the whole of the session.



SYN

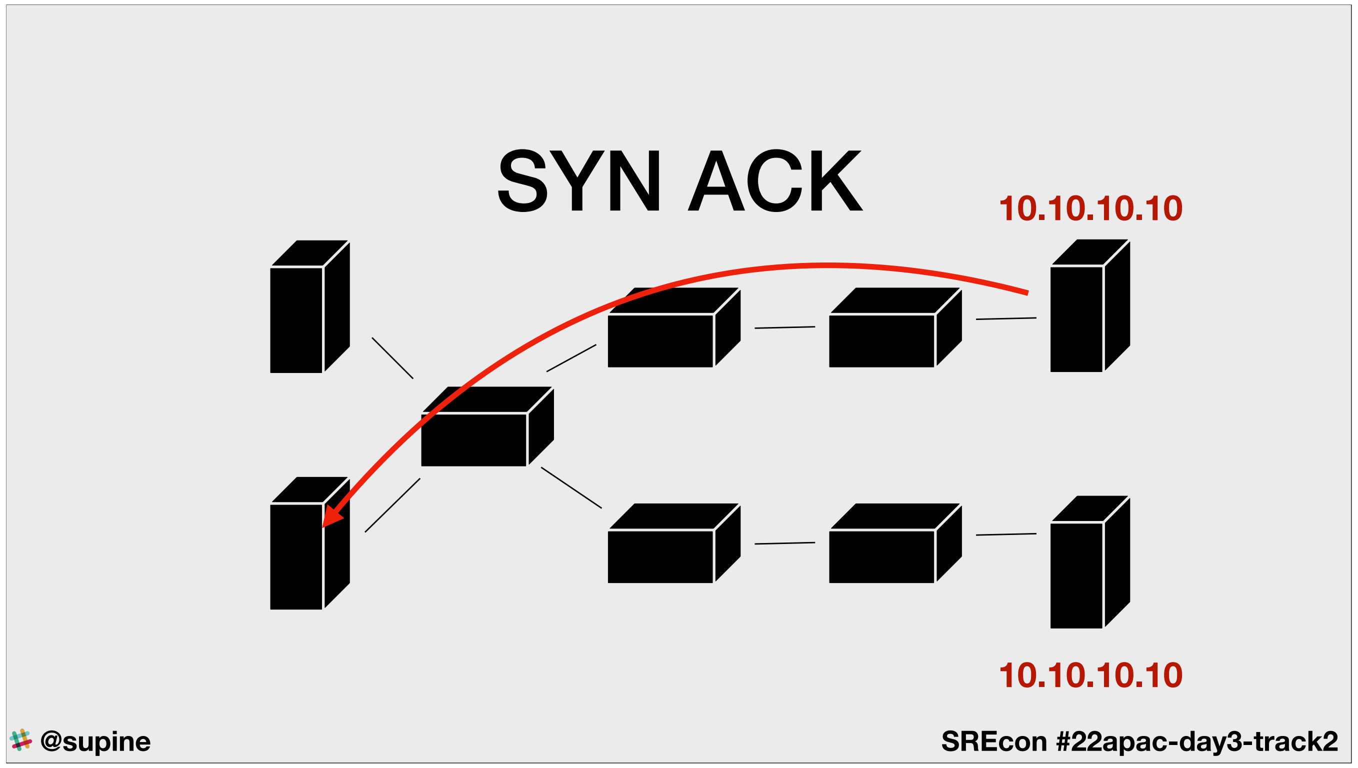
@supine

SREcon #22apac-day3-track2

To illustrate the pathological case, an unstable path between the client and the potential servers, let us walk through a TCP handshake.

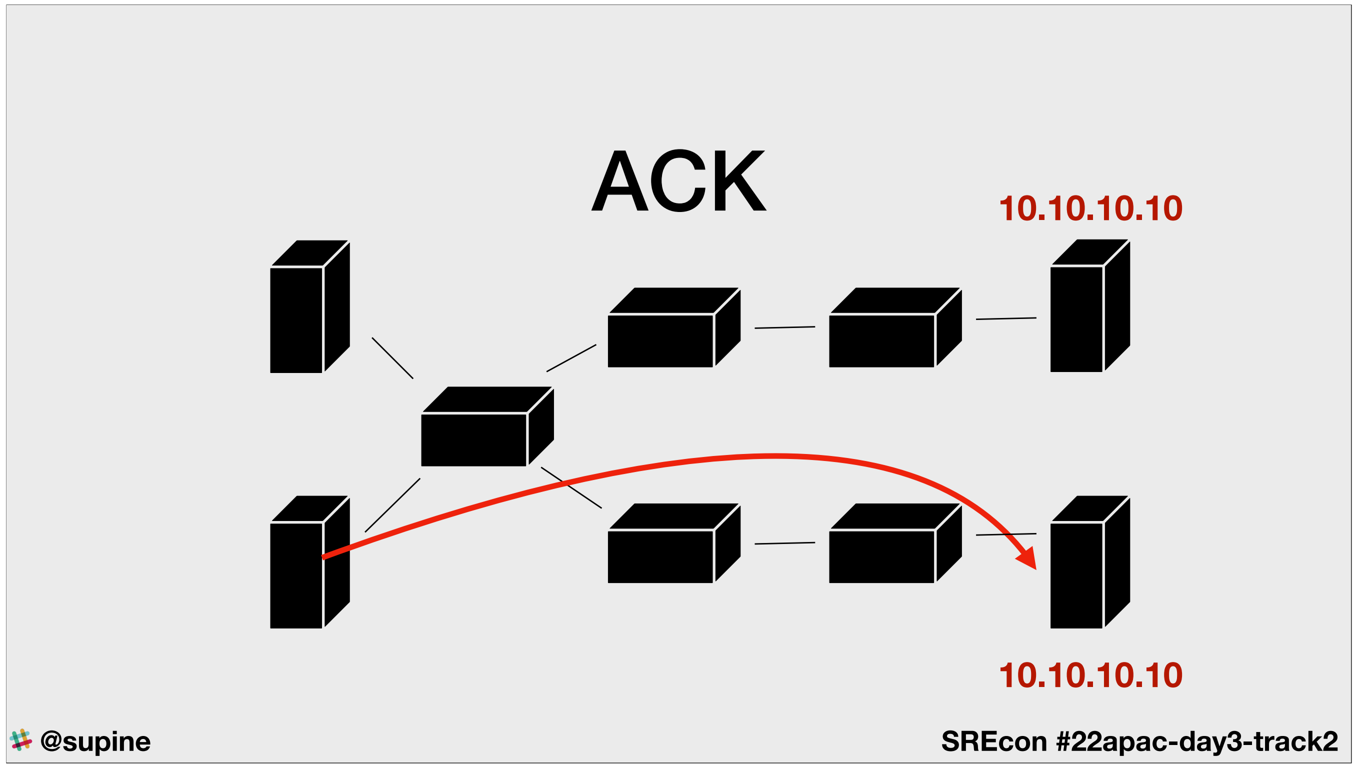
Connection is coming from the client on the bottom left.

SYN packet is routed to the edge location at the top right.



SYN ACK

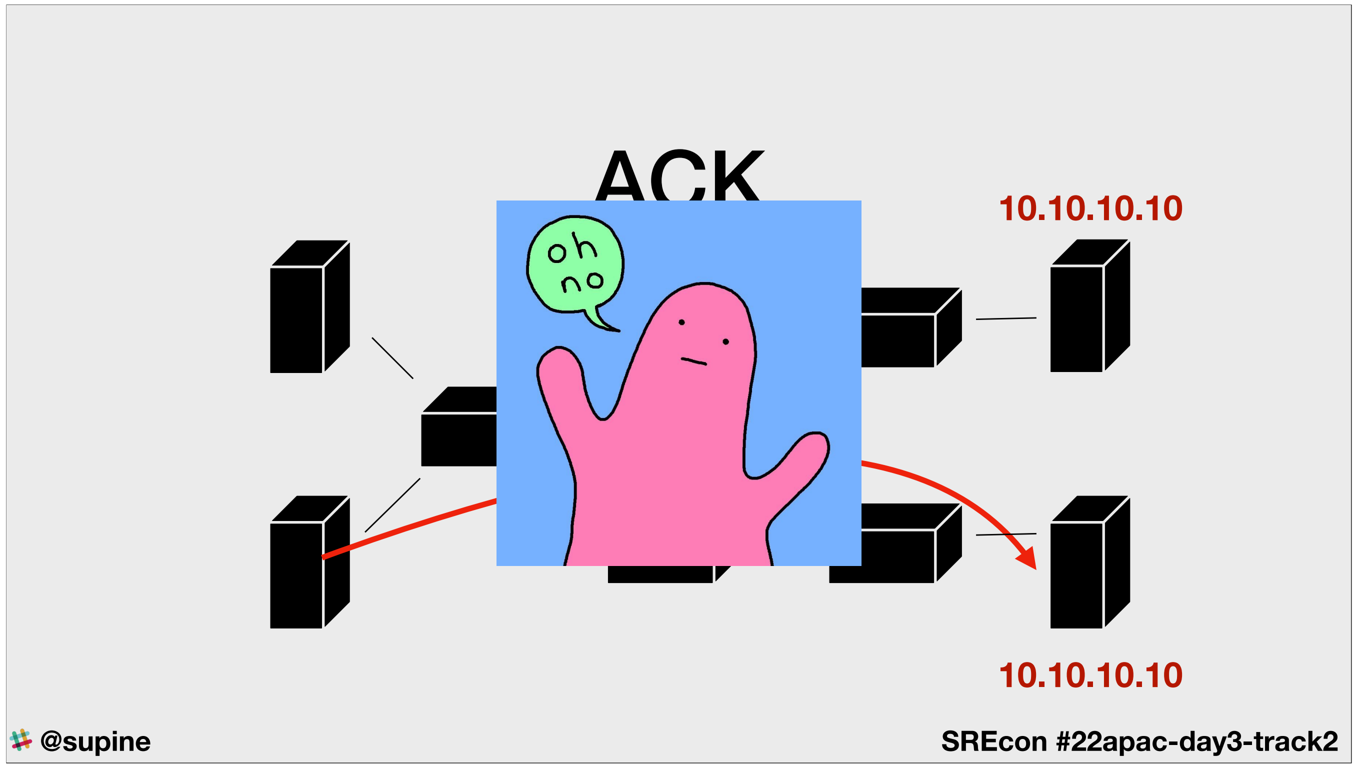
The server returns a SYN ACK packet to the client.



ACK

The client sends an ACK packet but instead of being routed to the same edge location, it is routed to a different one.

The server that receives the ACK packet has no connection state for this client and returns a reset packet.

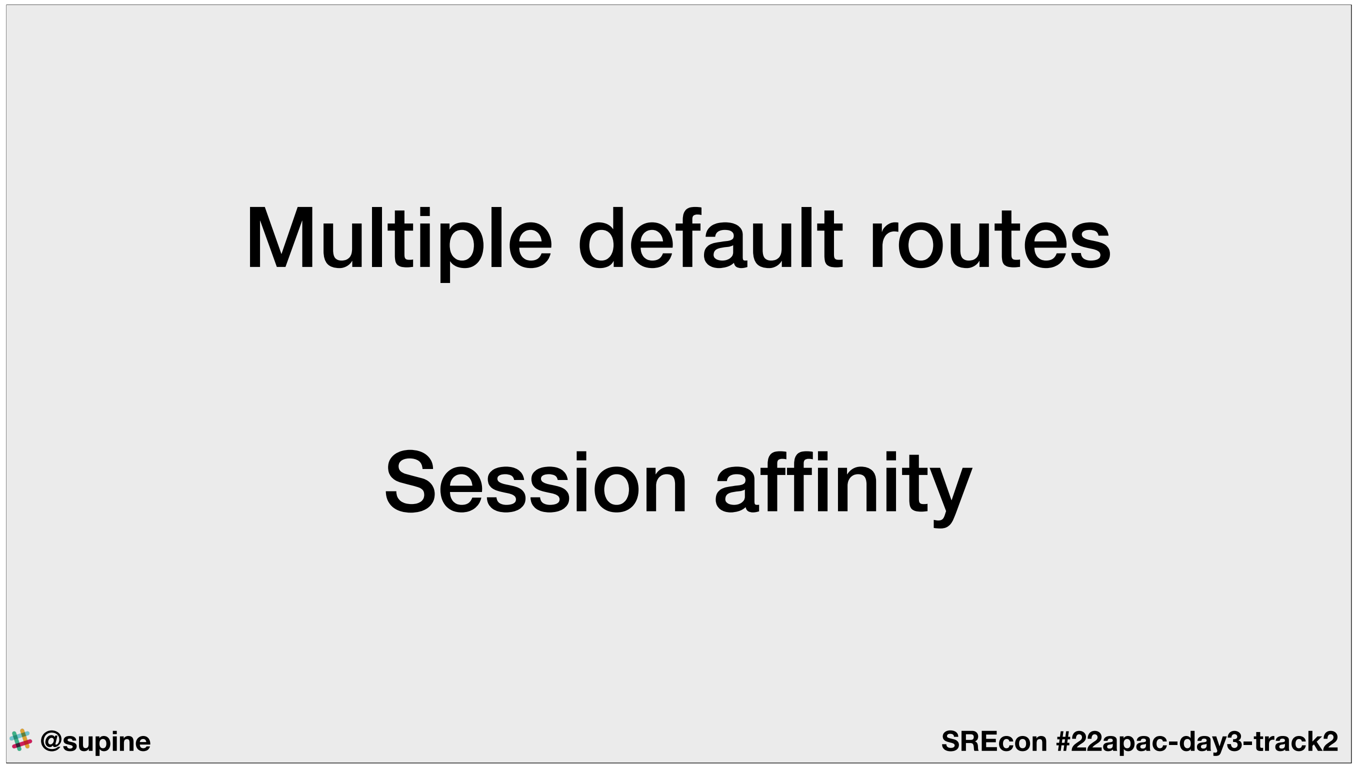


ACK

@supine

SREcon #22apac-day3-track2

We have one very confused client who may or may not retry the connection in a timely manner.



Multiple default routes

Session affinity

@supine SREcon #22apac-day3-track2

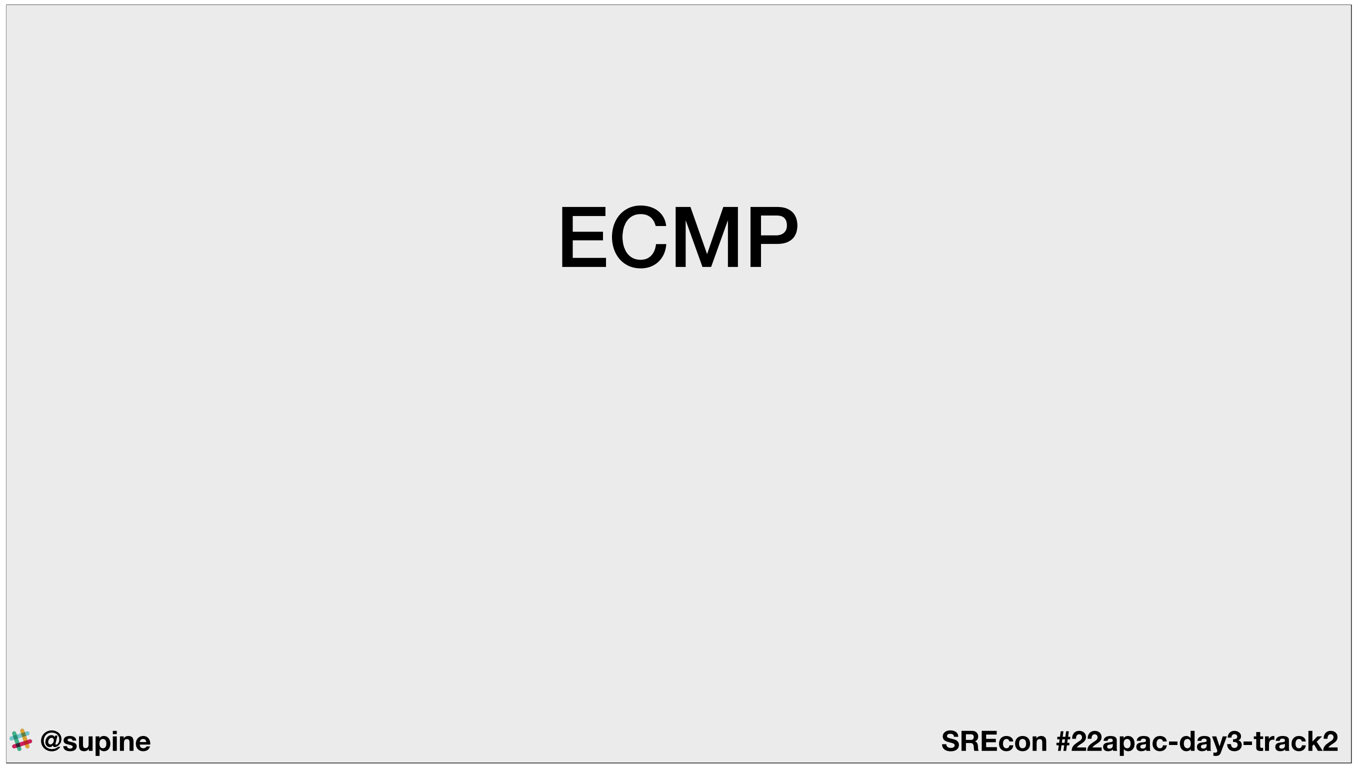
So let us talk about how to get stable paths between clients and edge locations.

It’s possible for internet routers or firewalls to utilise multiple upstream links, each with a simple default route.

The internet connections can operate in a load-sharing or active-backup mode.

This equipment should operate with session affinity, which is when they persistently use the same upstream link for the life of any single TCP session.

Alternatives like per-packet load-balancing can lead to unstable paths and connection resets.



ECMP

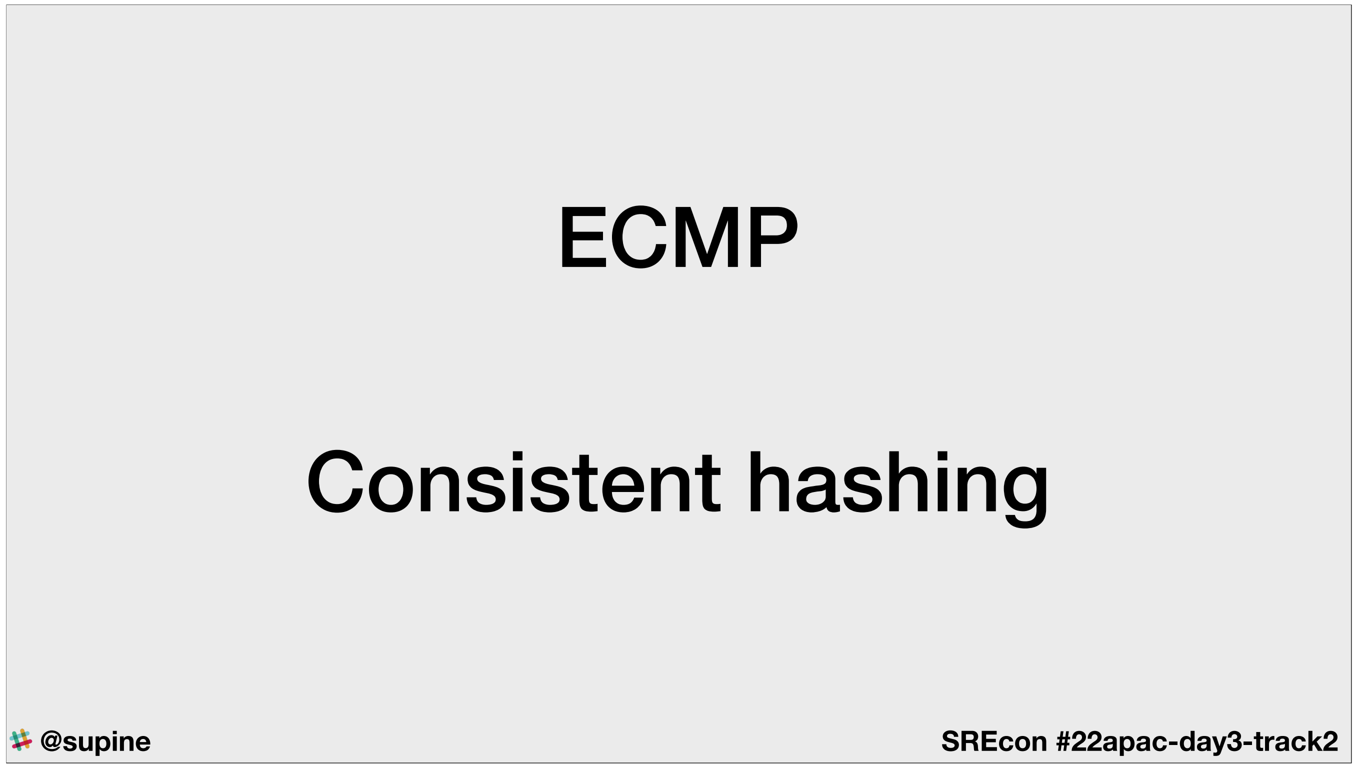
@supine SREcon #22apac-day3-track2

The next case we need to consider is routers participating in internal or external routing systems.

ECMP is equal cost multi-pathing.

This is the routing algorithm installing multiple paths to an IP range because it calculated them as equal.

One of the paths is selected by a hashing algorithm.



ECMP

Consistent hashing

@supine

SREcon #22apac-day3-track2

Persistently routing the packets of the same TCP session over the same path requires consistent hashing.

This means that the inputs to the hash need to be stable for the life of the TCP session.

Typically this means you can only use the source and destination IPs and ports as inputs to the hashing algorithm.

Any other inputs to the hash are likely to make it inconsistent and hence unable to ensure stable path selection.

BGP

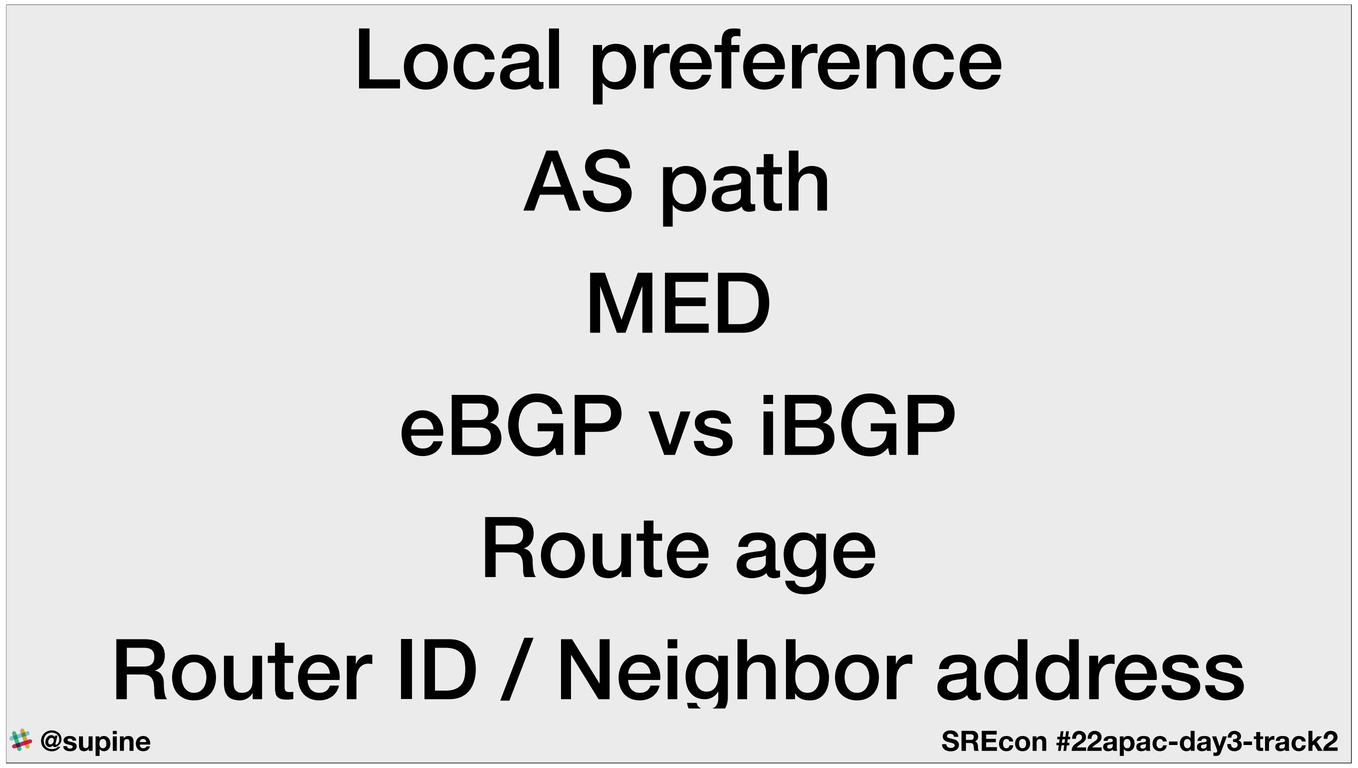
@supine

SREcon #22apac-day3-track2

As we spoke about ECMP there was mention of routing algorithms and it’s important that we dig into that aspect a little deeper.

When networks exchange routing information with other networks they require an external gateway protocol and the defacto one in use on the internet is BGP, or border gateway protocol version 4.

In order to maximise the potential for stable paths, we need the BGP algorithm to be deterministic to minimise or eliminate churn between alternative paths.


Local preference

AS path

MED

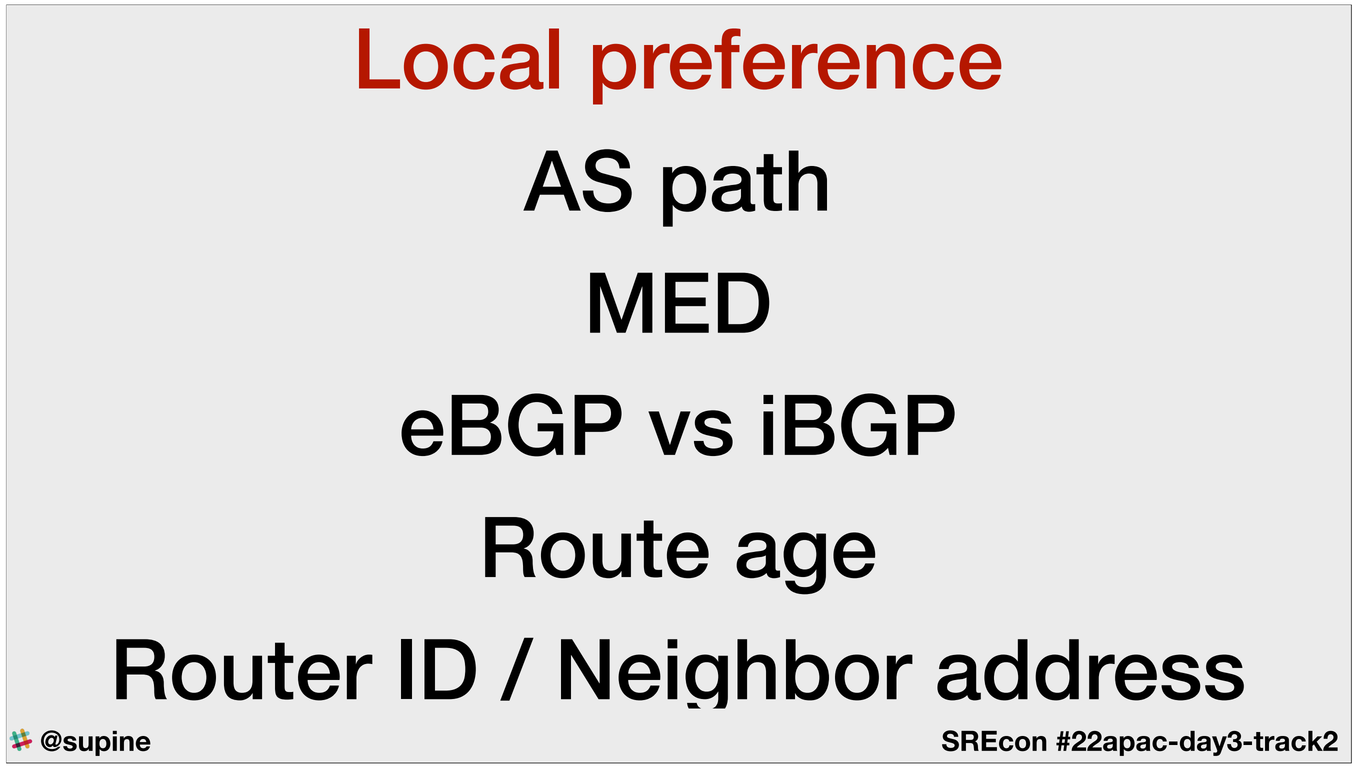
eBGP vs iBGP

Route age

Router ID / Neighbor address

@supine SREcon #22apac-day3-track2

The BGP route selection algorithm has up to a dozen decision factors but I'll focus on just the most important ones.


Local preference

AS path

MED

eBGP vs iBGP

Route age

Router ID / Neighbor address

@supine SREcon #22apac-day3-track2

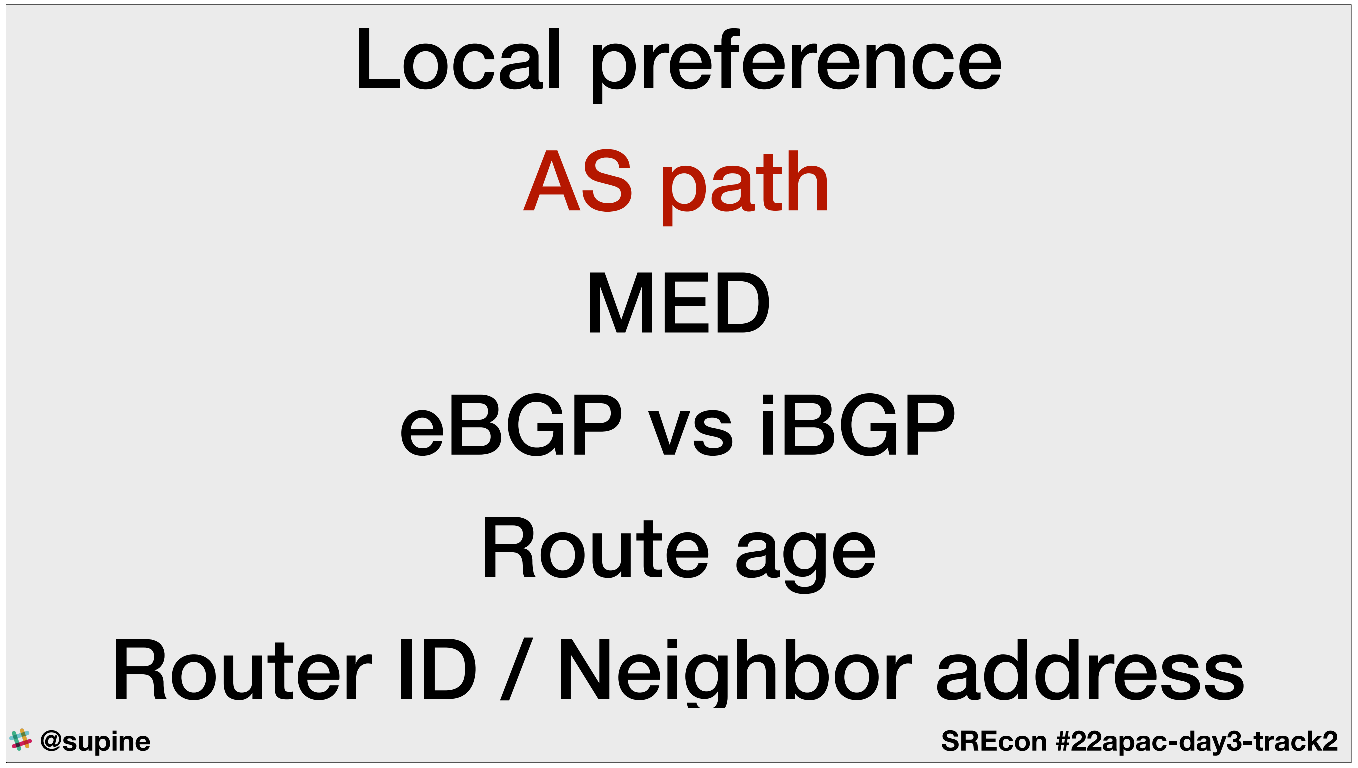
Local preference is a large hammer that networks can use to override all other tiebreakers.

It’s usually statically configured and is essentially an expression of a network’s routing policy.

A typical configuration would prefer downstream routes over upstream ones.

But it can also be an expression of other policies, like preferring cheaper paths over more expensive ones, preferring higher capacity paths over smaller ones.

Because the configuration is typically static, it is also deterministic.


Local preference

AS path

MED

eBGP vs iBGP

Route age

Router ID / Neighbor address

@supine

SREcon #22apac-day3-track2

BGP is a path vector protocol and that surfaces in the AS path attribute.

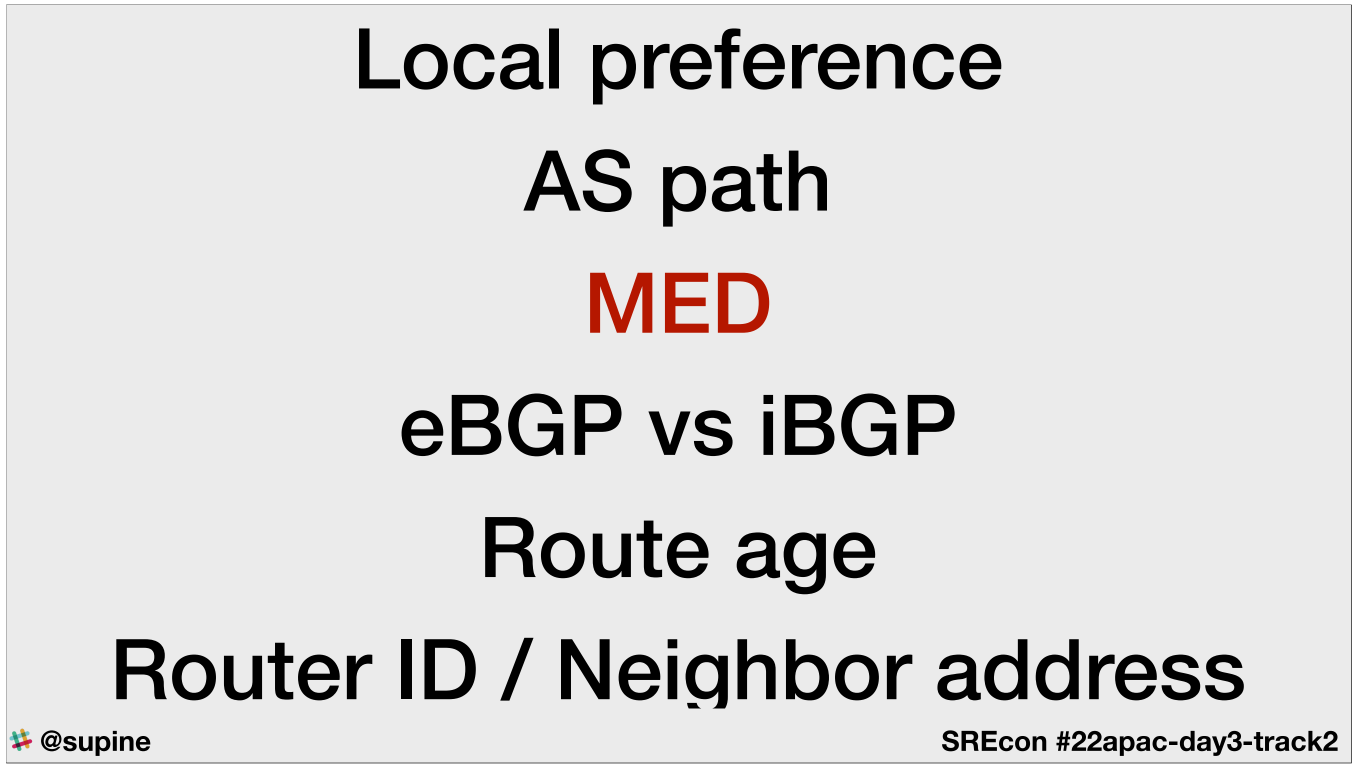
AS stands for autonomous system, essentially a network under the control of one organisation.

Each AS has a designated number, referred to as an ASN.

The network that announces an IP range puts it's ASN in the attribute and each new AS that the route is passed to adds it's own ASN to the front of the list.

The attribute is also used for loop detection but for our discussion today it's just important to know that BGP prefers a shorter AS path.

As long as our source of the route is stable, this tiebreaker will also be deterministic.



Local preference

AS path

MED

eBGP vs iBGP

Route age

Router ID / Neighbor address

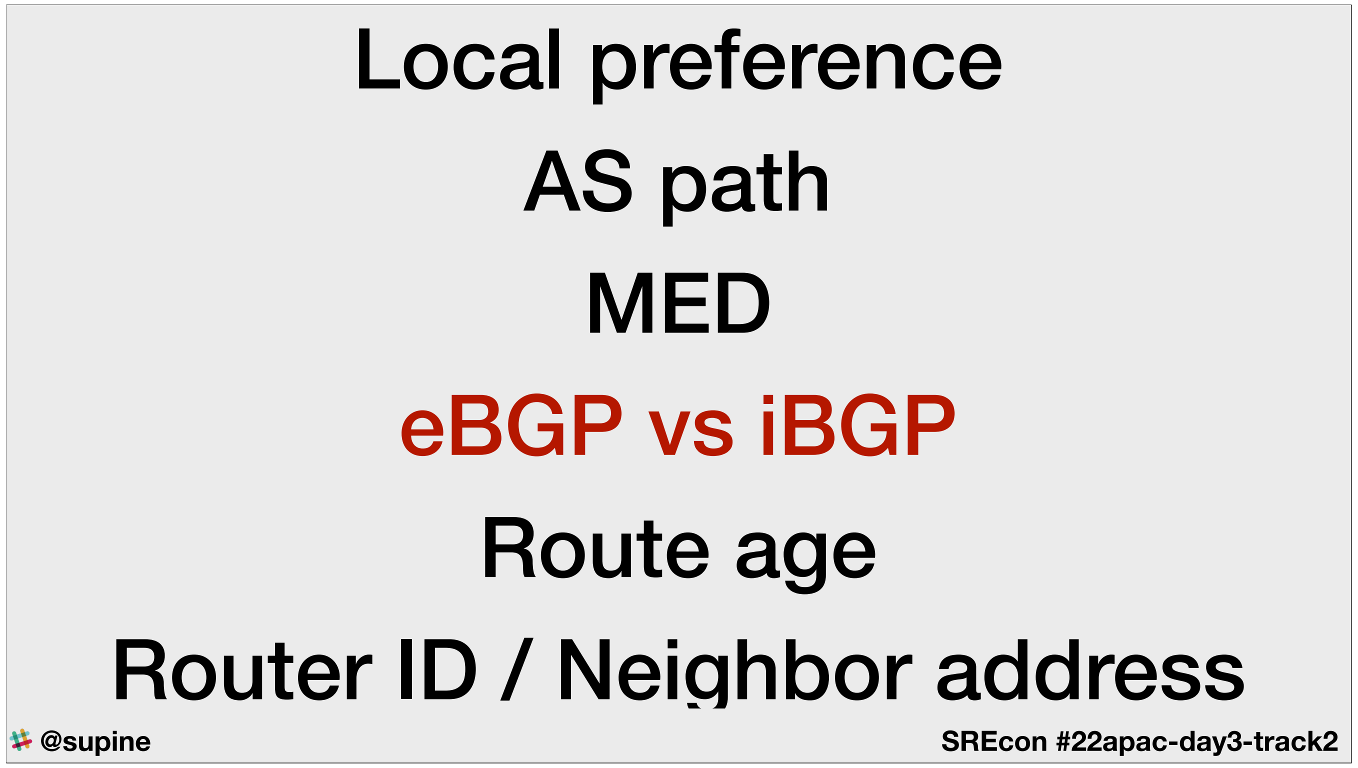
@supine SREcon #22apac-day3-track2

MED is multi-exit discriminator.

If two networks interconnect in multiple places they can apply MED as they export routes to signal their preferred location for receiving traffic for that IP range.

However the network receiving those announcements can choose to honour that expressed preference … or not.

If they want something different to happen they can override the MED attribute itself or they can manipulate earlier attributes so MED is never considered.



Local preference

AS path

MED

eBGP vs iBGP

Route age

Router ID / Neighbor address

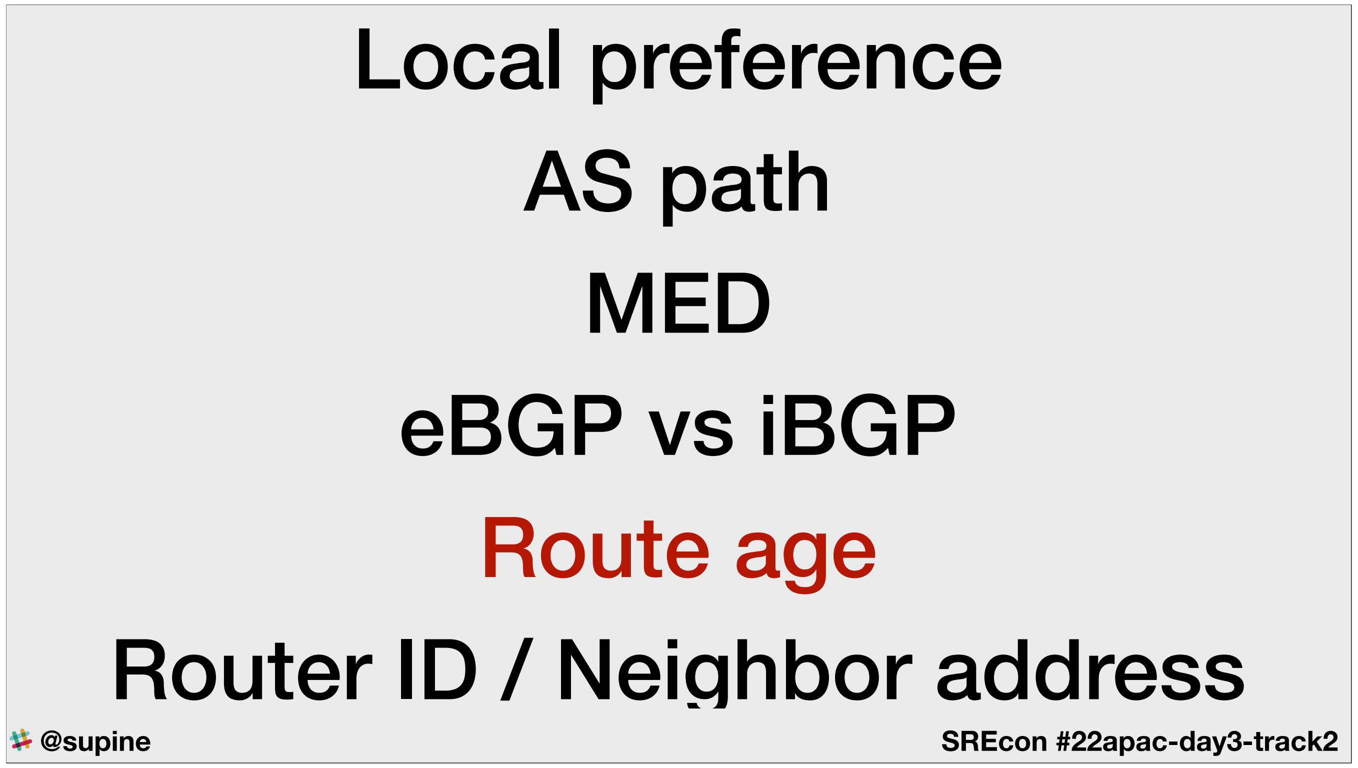
@supine SREcon #22apac-day3-track2

We have been discussing BGP used externally (eBGP) but it can also be used internally (iBGP).

The next tiebreaker is preferring routes received via external BGP over routes received via internal BGP.

This means BGP prefers to send packets out of our network as soon as possible.

This is our last deterministic tiebreaker and so a router configured to do ECMP will have stopped at this point and would install the remaining equal paths.


Local preference

AS path

MED

eBGP vs iBGP

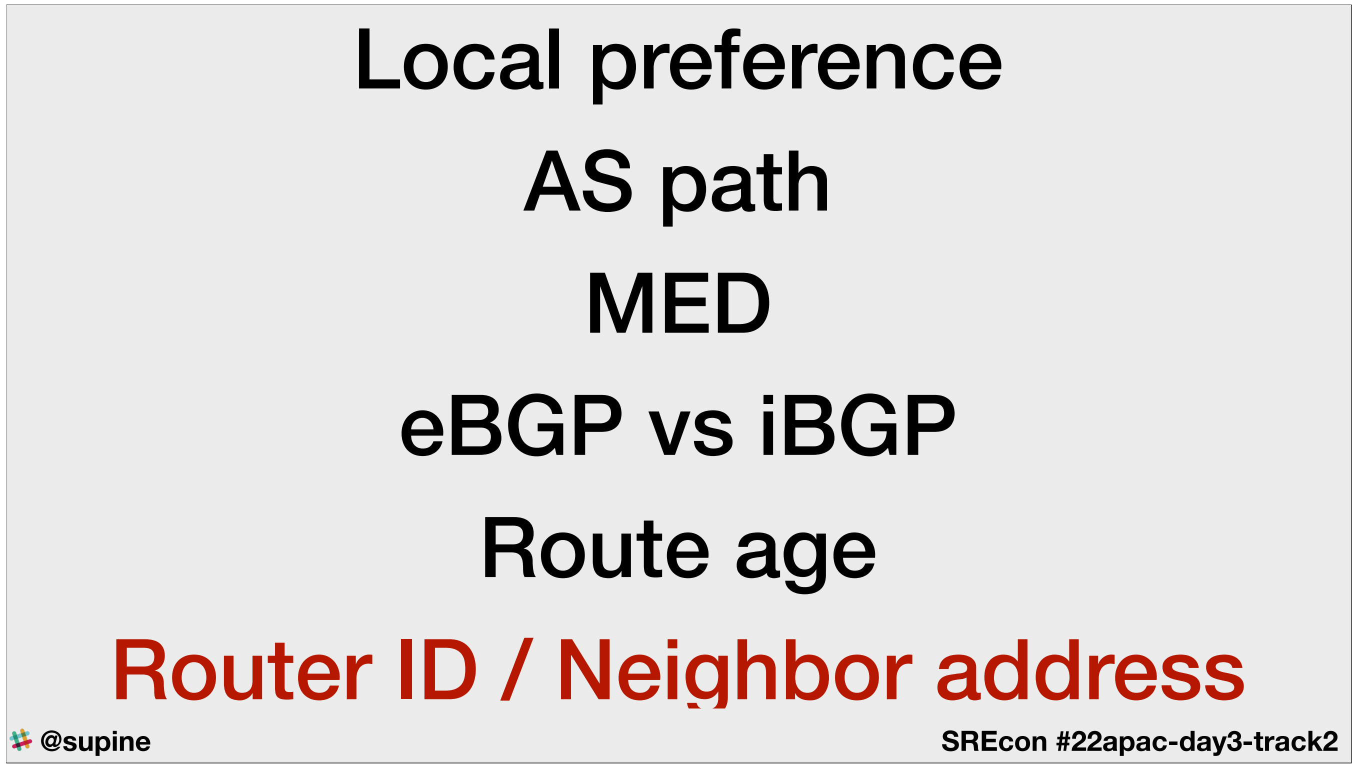
Route age

Router ID / Neighbor address

BGP will prefer a route it received earlier over newer ones.

This is designed to prefer more stable paths that have not changed recently.

However it’s not deterministic and causes route churn when the oldest path is interrupted.


Local preference

AS path

MED

eBGP vs iBGP

Route age

Router ID / Neighbor address

@supine

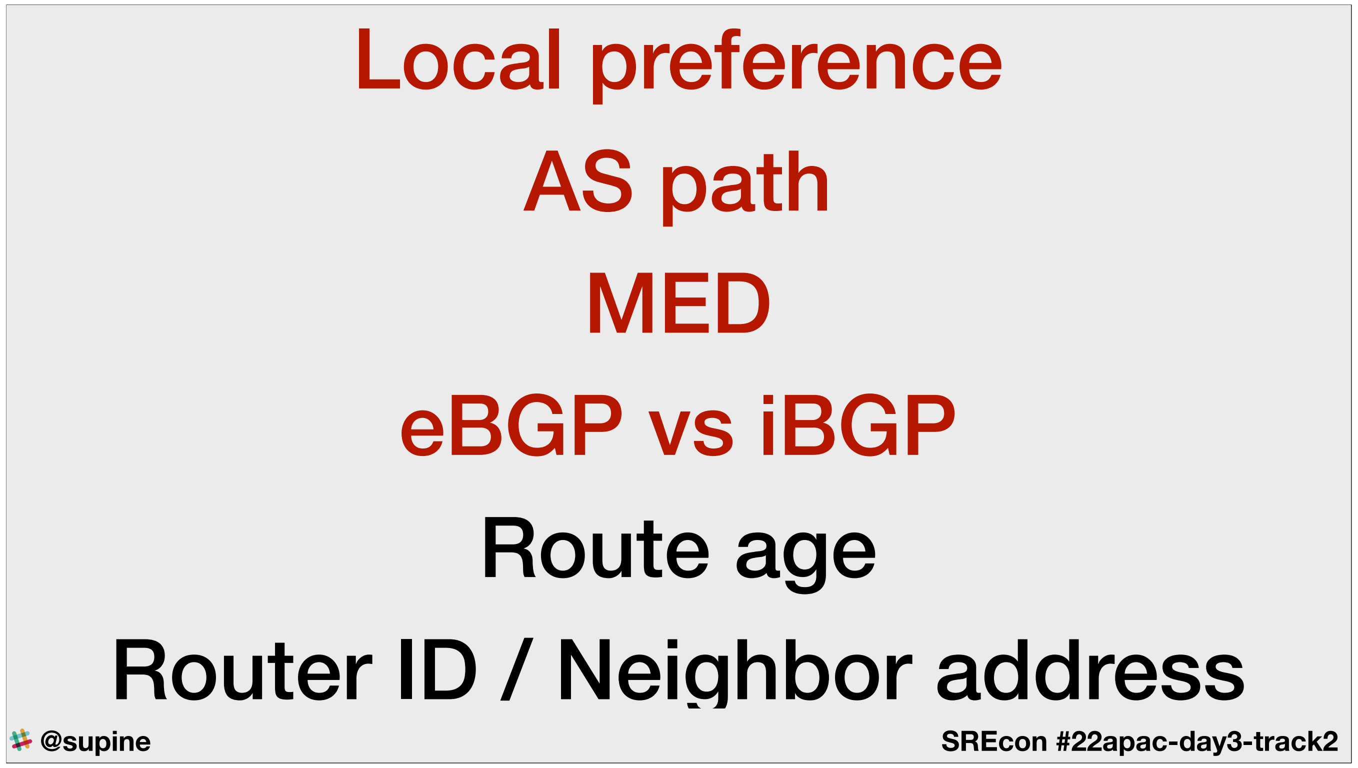
SREcon #22apac-day3-track2

If BGP is still trying to tiebreak at this point it is getting desperate and uses attributes that have little to do with the quality of the path.

Router ID is usually the primary IP address of the neighbouring router and BGP prefers the “lower” router ID.

If two BGP sessions go to the same neighbour router over different circuits, then the algorithm will prefer the routes received over the circuit with the “lower” IP address.

Both of these tiebreakers are deterministic but they are also completely arbitrary, the computing equivalent of deciding something with a coin flip.



Local preference

AS path

MED

eBGP vs iBGP

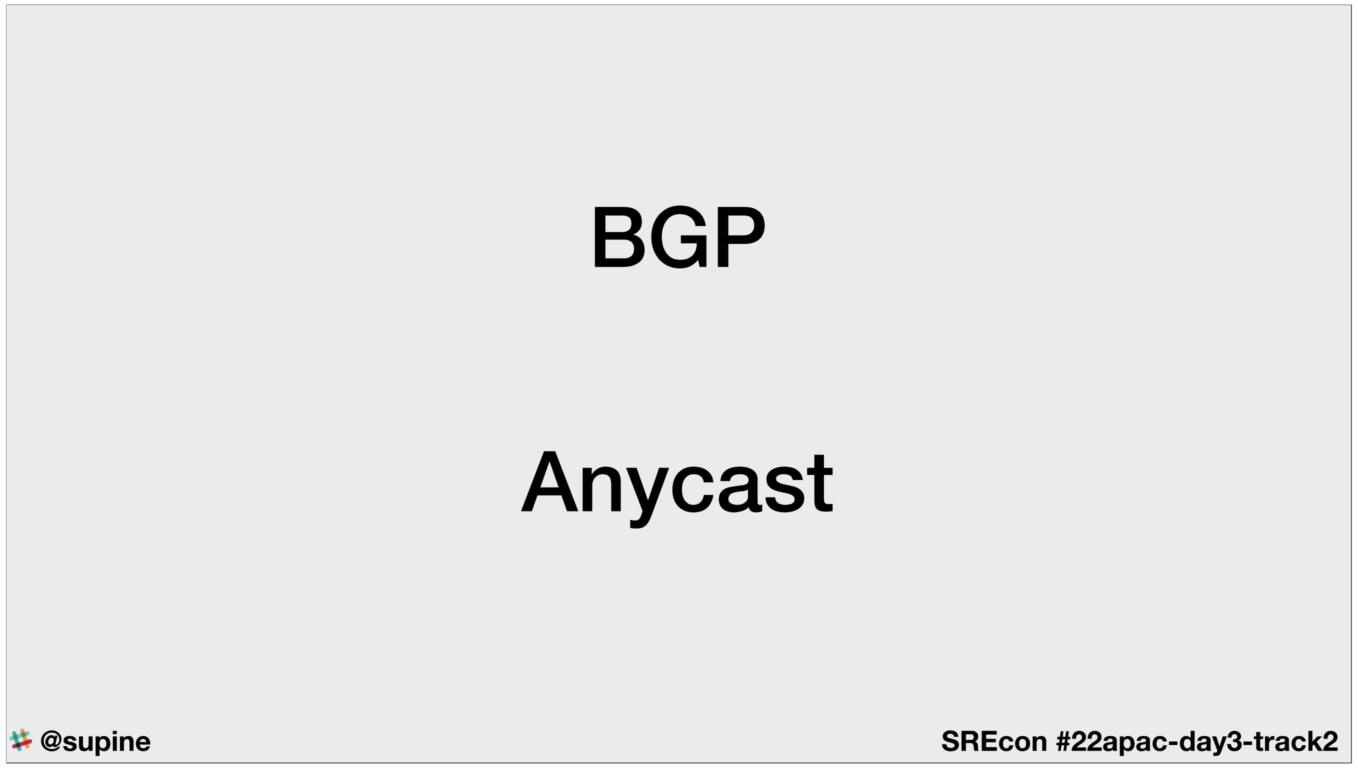
Route age

Router ID / Neighbor address

@supine

SREcon #22apac-day3-track2

So in summary, BGP can choose multiple stable paths as long as it stops tie-breaking early enough.



BGP

Anycast

@supine

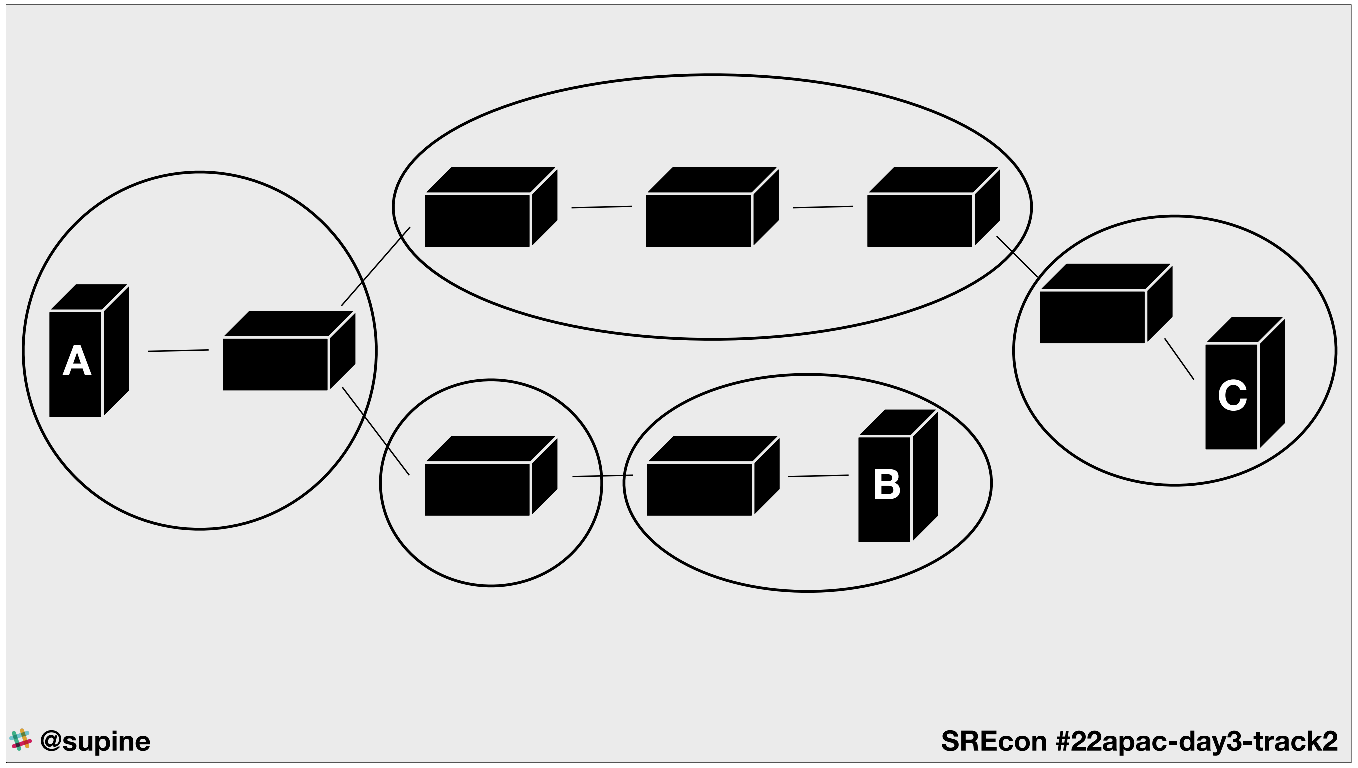
SREcon #22apac-day3-track2

But we are not quite done talking about BGP just yet.

Anycast has a goal of packets being routed to the nearest edge location.

But if you think back to the tiebreakers there wasn’t actually much that was optimising for “nearest”.

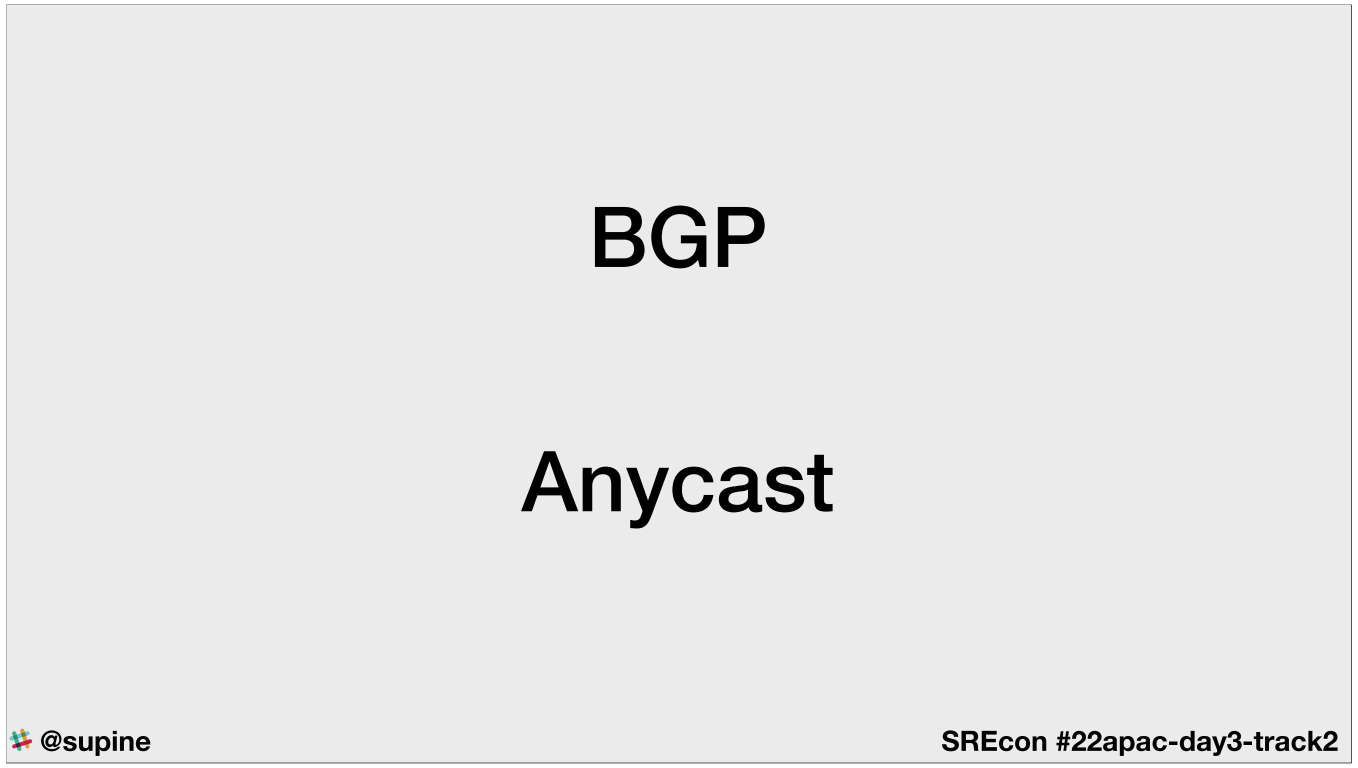
The closest is AS path but it’s explicitly a path vector, not a distance vector.



An ASN is one AS no matter how big or small that network is in reality.

A tiny local network is counted the same as a global one.

If you imagine each of those ovals is a single ASN, then BGP will treat the path between A and B as equal to the path between A and C despite B being topologically closer than C.



BGP

Anycast

@supine

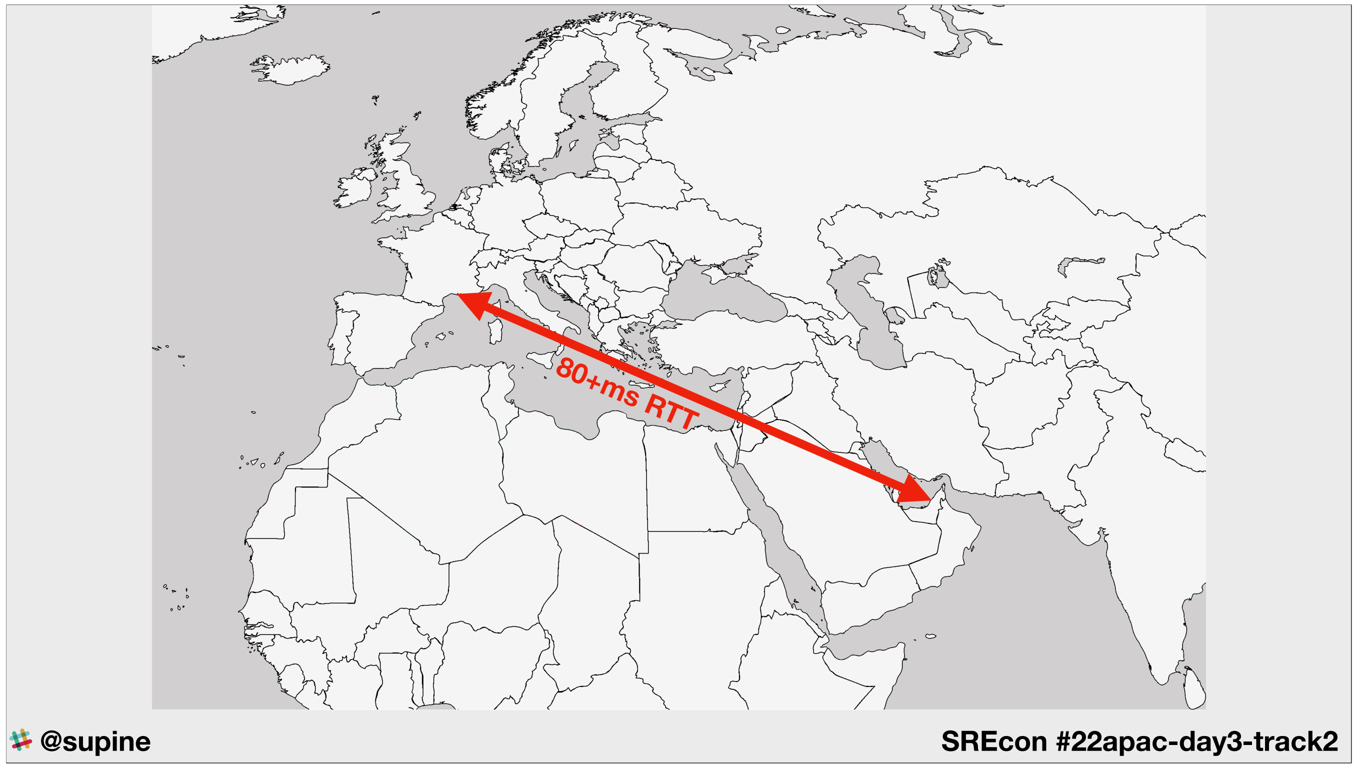
SREcon #22apac-day3-track2

But AS path was overridden by local preference.

That means a network's decisions about how they want to run their network take precedence.

If they decide that latency and performance are not important, they can configure a policy that optimises for other things like cost.

To achieve performant anycast you need to understand which networks will do things like that and come up with an interconnection strategy that plays into how they want to optimise their network.



A good example to illustrate the problem is to consider the United Arab Emirates, a location with expensive connectivity.

Unless you find a way to connect directly to local ISPs, either with direct circuits or via an internet exchange, they will prefer to exchange traffic with you in Europe.

Submarine capacity back to Europe can actually be cheaper than local transit capacity.

ISPs would prefer to exchange traffic in Marseille, adding an 80+ millisecond round-trip, because it will save them money.

You need to find a solution that ensures the traffic stays local if you want your anycast to remain performant.

BGP communities

@supine SREcon #22apac-day3-track2

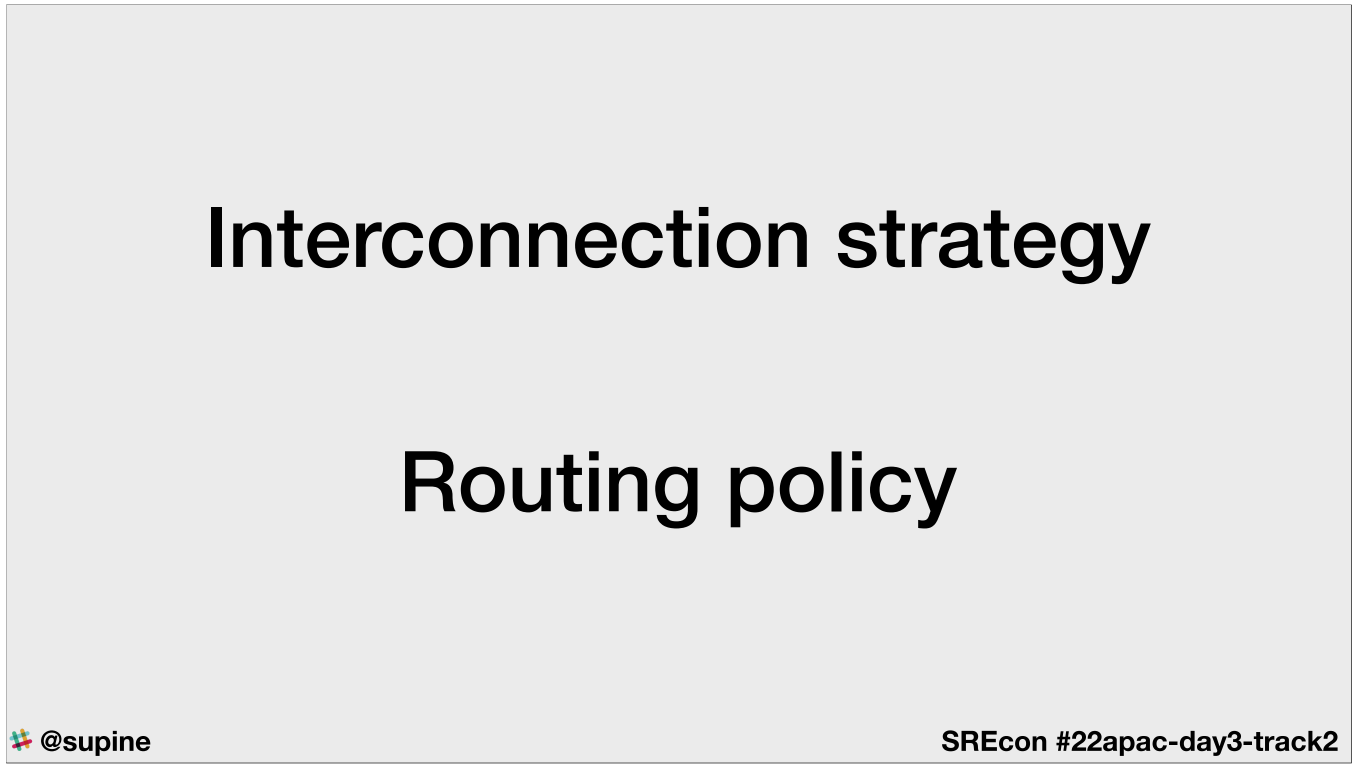
But there will always be a subset of networks you can’t reach directly.

You will exchange traffic with them via your transit providers and shaping how they route towards your systems can be done using BGP communities.

Communities are labels you can tag routes with as you export them to your transit providers.

Each of them has a different meaning, indicating to the provider they should handle that route in a non-default way.

One possibility is asking them to not export that route to a particular AS. Or to not export the route outside a country or region. Or apply a different local preference to that route when propagating it inside their network.



Interconnection strategy

Routing policy

@supine

SREcon #22apac-day3-track2

So in summary, you can turn the internet into your load balancer using anycast and focusing on these two areas.

You pay attention to which networks you connect to and where.

You select which routes to announce in which locations to which networks.

You use communities to tweak how those routes reach the far corners of the internet.

SLOs

@supine SREcon #22apac-day3-track2

This is SREcon so of course I have to talk about SLOs (service level objectives).

Based on historical trends and analysis of the current environment we should be able to determine a number of goals.

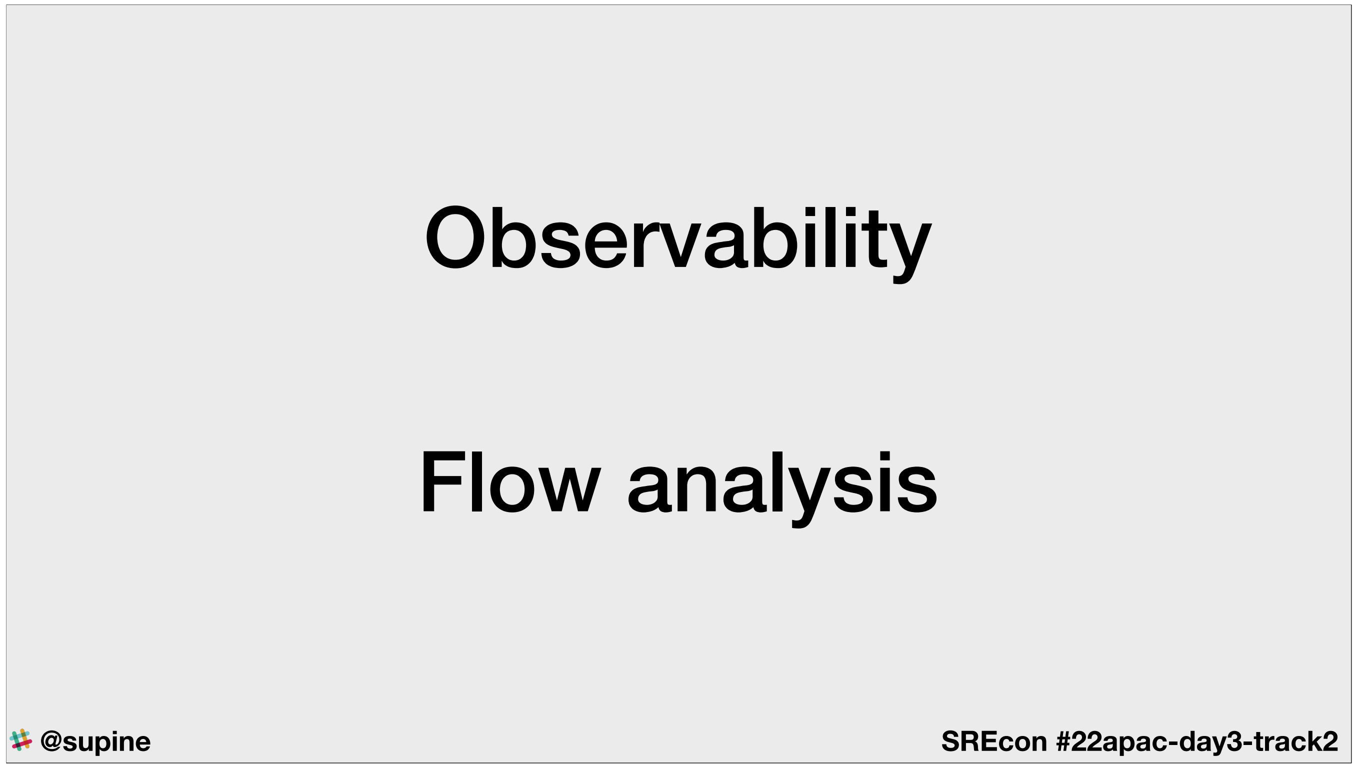
Round-trip latency, throughput, retransmissions due to packet loss are all relevant measurements which we can evaluate our performance with.

What our goals are for each measure will vary by location and network. Some regions have higher quality infrastructure than others. Some networks invest more in capacity and maintenance than others.

And sometimes you can't tell if there is a problem you should be troubleshooting or if that packet loss is just someone browsing social media while on the toilet where their wifi signal is bad.

Observability

So let us discuss a little how to collect measurements at internet scale.



Observability

Flow analysis

@supine

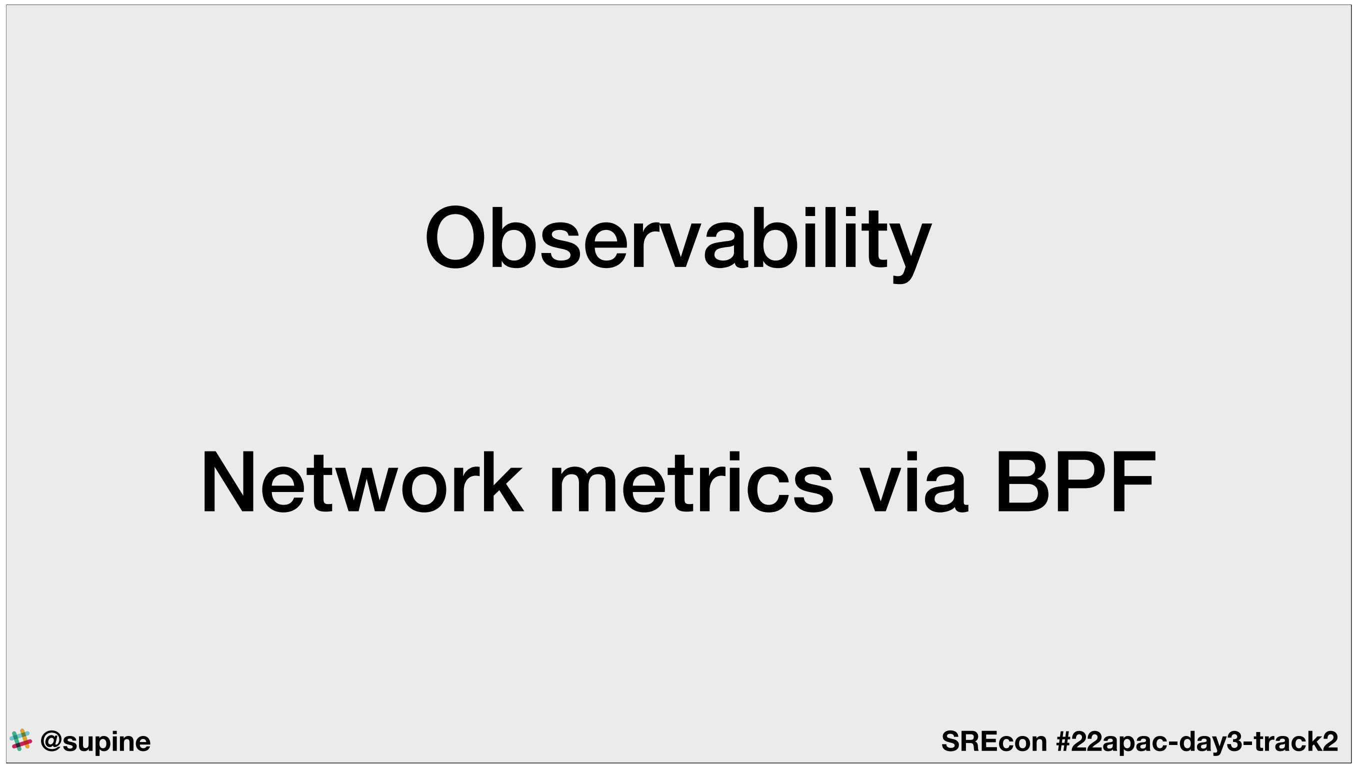
SREcon #22apac-day3-track2

One of our vantage points for measurements is the routing equipment in each of our edge locations.

Most modern network equipment can sample the traffic flowing through it and emit details that can be collected and aggregated.

Typically you learn the duration of the flow, IPs and ports, packet counts and total bytes transferred.

Given sufficiently large traffic volumes you have to reduce your sampling rate and aggressively aggregate the data ingested at your collector infrastructure but you still should have a usable picture of how much traffic is exchanged with particular IP ranges and networks.


Observability

Network metrics via BPF

@supine

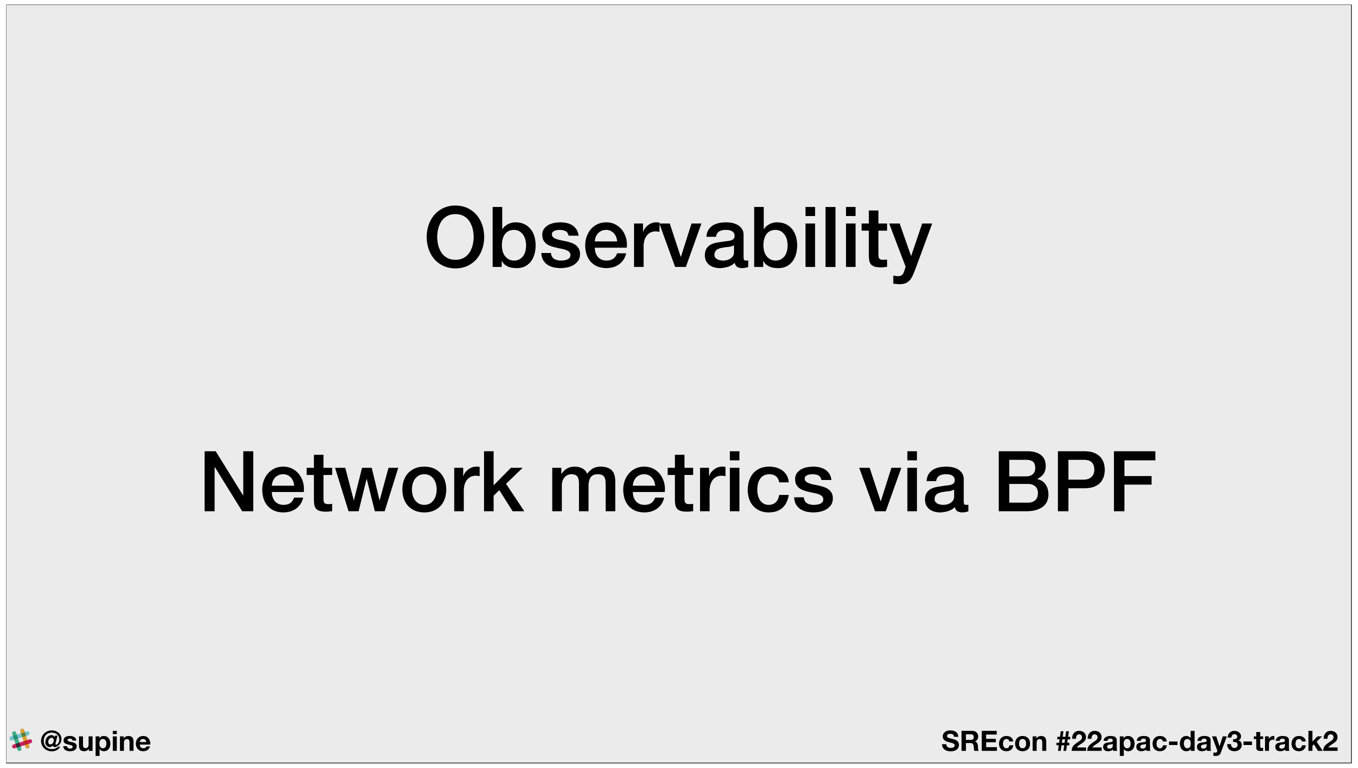
SREcon #22apac-day3-track2

Although flow analysis can be helpful, you are limited by the equipment vendors implementation and how configurable they have made it.

You can collect more detailed and customised network metrics at the server operating system level but that is perhaps not the best vantage point available to us today.

Recent advances in BPF mean it’s now possible to collect detailed, custom network metrics at the driver or even hardware level.

This is particularly useful if we are also handling packets with BPF, to process them in a non-standard way or to protect the operating system from denial of service attacks.



Observability

Network metrics via BPF

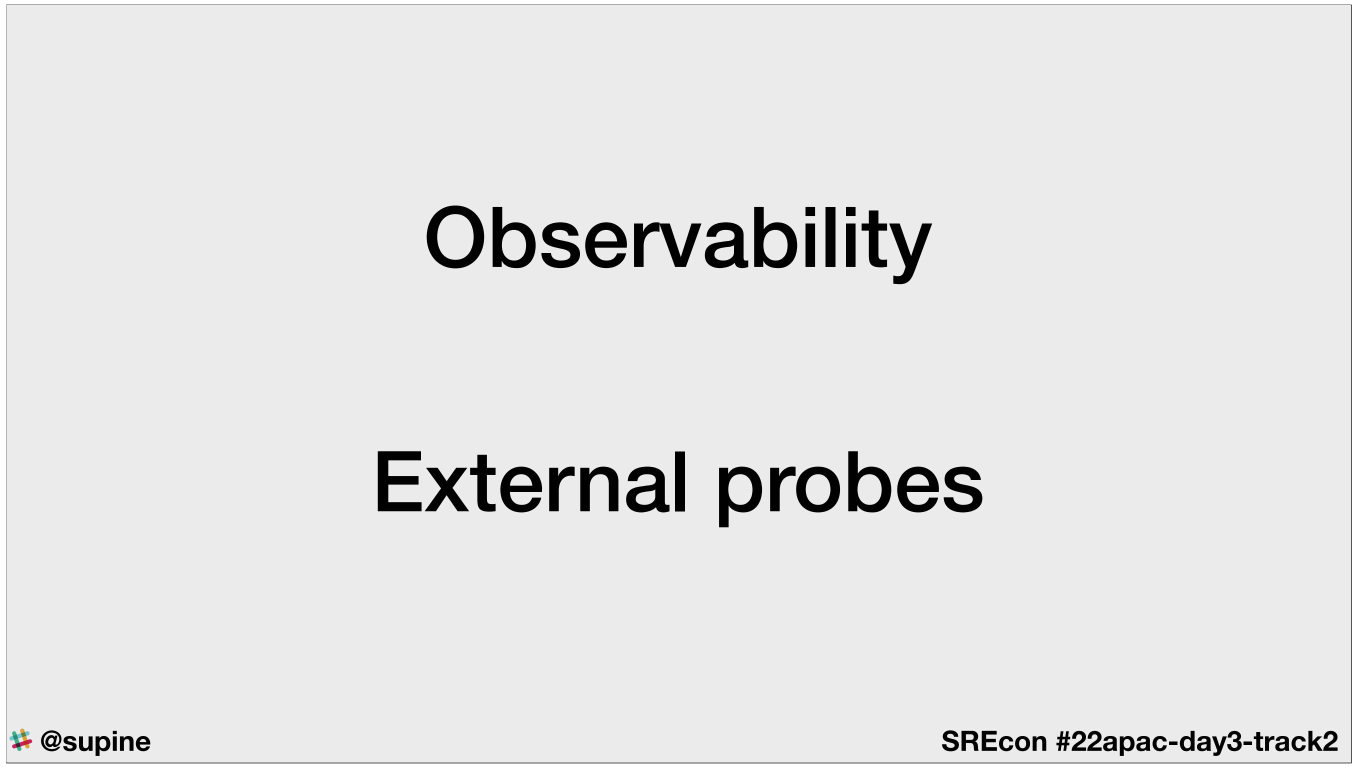
@supine

SREcon #22apac-day3-track2

To help analyse TCP performance we can measure many things on a per-session basis.

Round-trip latency, jitter, retransmissions, maximum segment size, window size, effective throughput, socket re-use.

We can also use weighted sampling to aggregate away uninteresting details and retain more data that helps us with troubleshooting anomalies.



Observability

External probes

@supine

SREcon #22apac-day3-track2

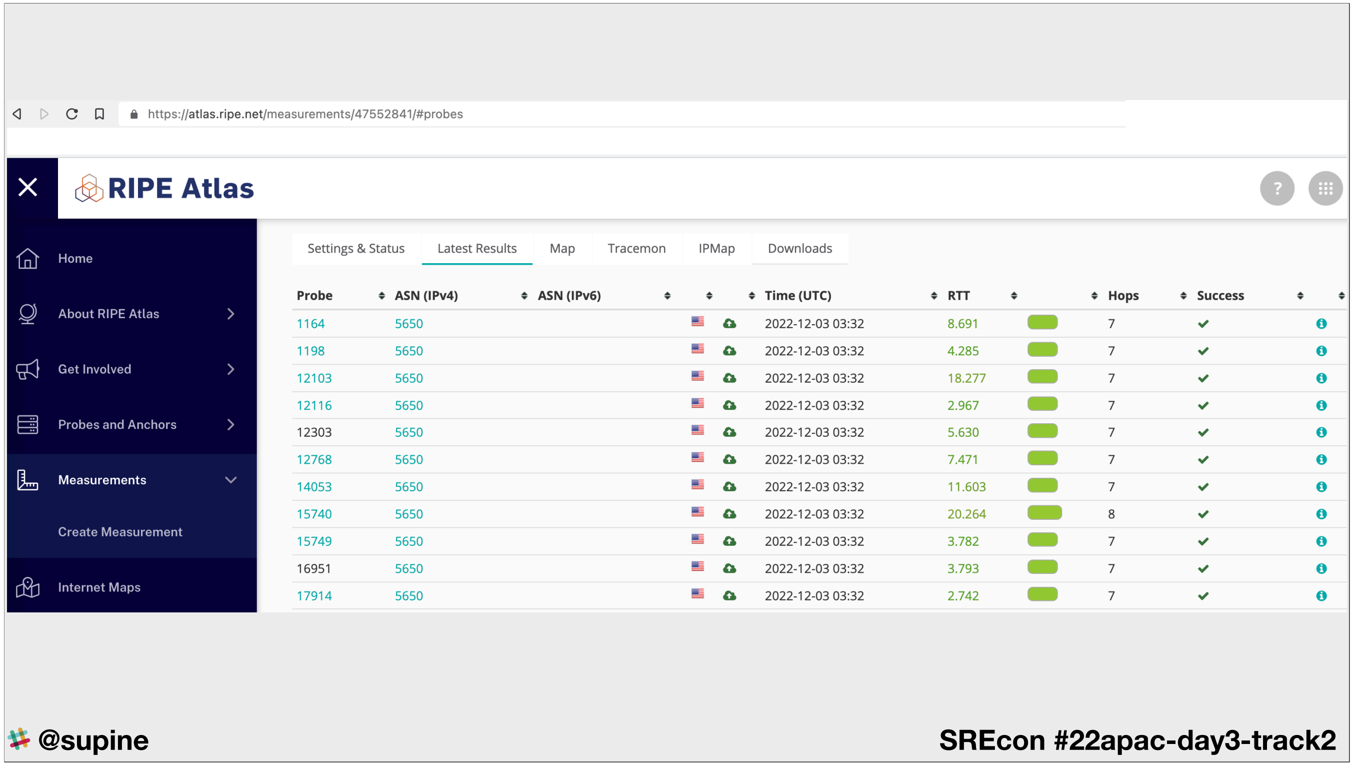
But so far we are only measuring inside our own network.

We are operating at internet scale and we need to also measure at the same scale.

We need to understand the reachability and performance of our system from outside our network.

One way of collecting these measurements is from probes in other networks.

We could run these probes ourselves, purchase access to a vendors probes or utilise community projects like RIPE Atlas.



Probe	ASN (IPv4)	ASN (IPv6)			Time (UTC)	RTT		Hops	Success	
1164	5650		🇺🇸		2022-12-03 03:32	8.691	🟩	7	✓	
1198	5650		🇺🇸		2022-12-03 03:32	4.285	🟩	7	✓	
12103	5650		🇺🇸		2022-12-03 03:32	18.277	🟩	7	✓	
12116	5650		🇺🇸		2022-12-03 03:32	2.967	🟩	7	✓	
12303	5650		🇺🇸		2022-12-03 03:32	5.630	🟩	7	✓	
12768	5650		🇺🇸		2022-12-03 03:32	7.471	🟩	7	✓	
14053	5650		🇺🇸		2022-12-03 03:32	11.603	🟩	7	✓	
15740	5650		🇺🇸		2022-12-03 03:32	20.264	🟩	8	✓	
15749	5650		🇺🇸		2022-12-03 03:32	3.782	🟩	7	✓	
16951	5650		🇺🇸		2022-12-03 03:32	3.793	🟩	7	✓	
17914	5650		🇺🇸		2022-12-03 03:32	2.742	🟩	7	✓	

@supine

SREcon #22apac-day3-track2

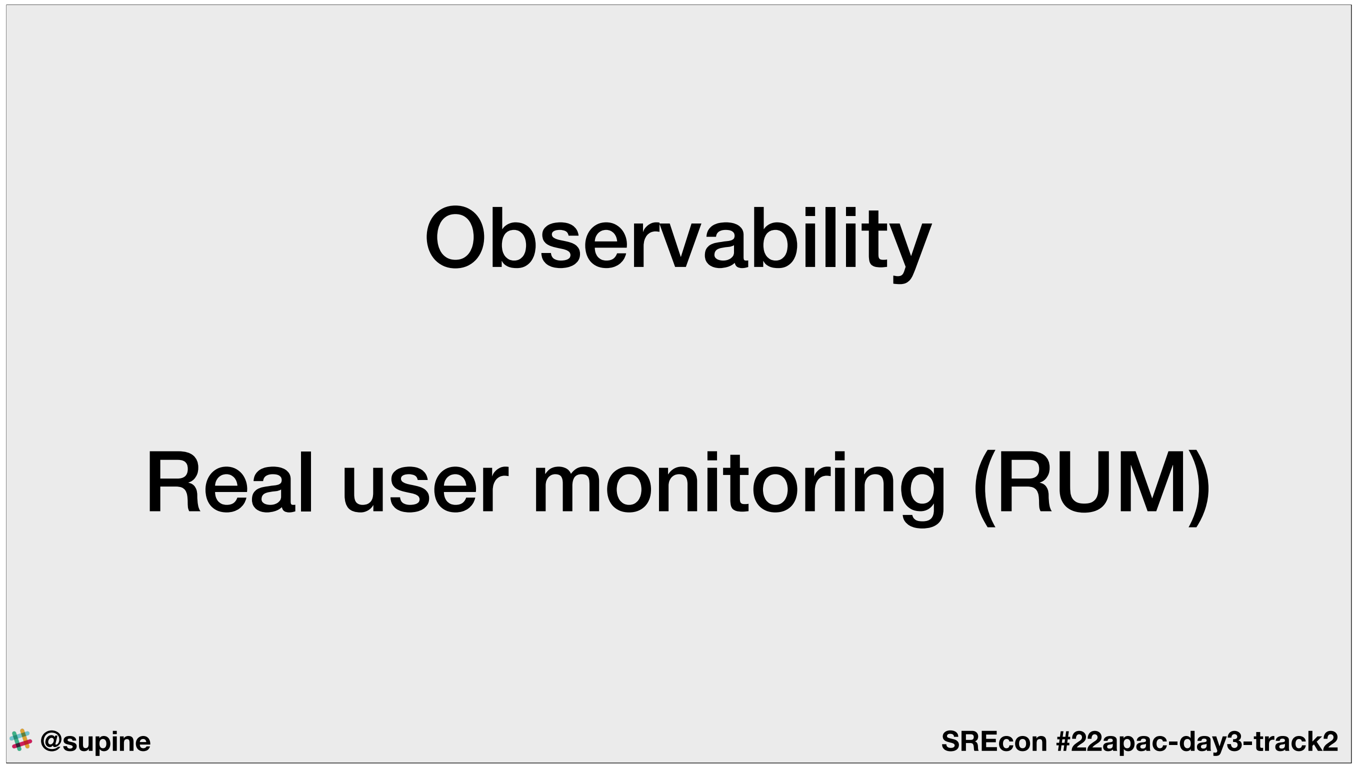
RIPE is the internet resources organisation for Europe, the Middle East and parts of Central Asia.

Their Atlas project has been distributing hardware probes for free to a wide variety of internet industry people.

They tend to be connected to home networks and provide measurement vantage points inside a large majority of ISPs.

You can create adhoc measurements or schedule regular ones.

There is a variety of measurement types including traceroutes, DNS lookups and HTTP tests.



Observability

Real user monitoring (RUM)

@supine

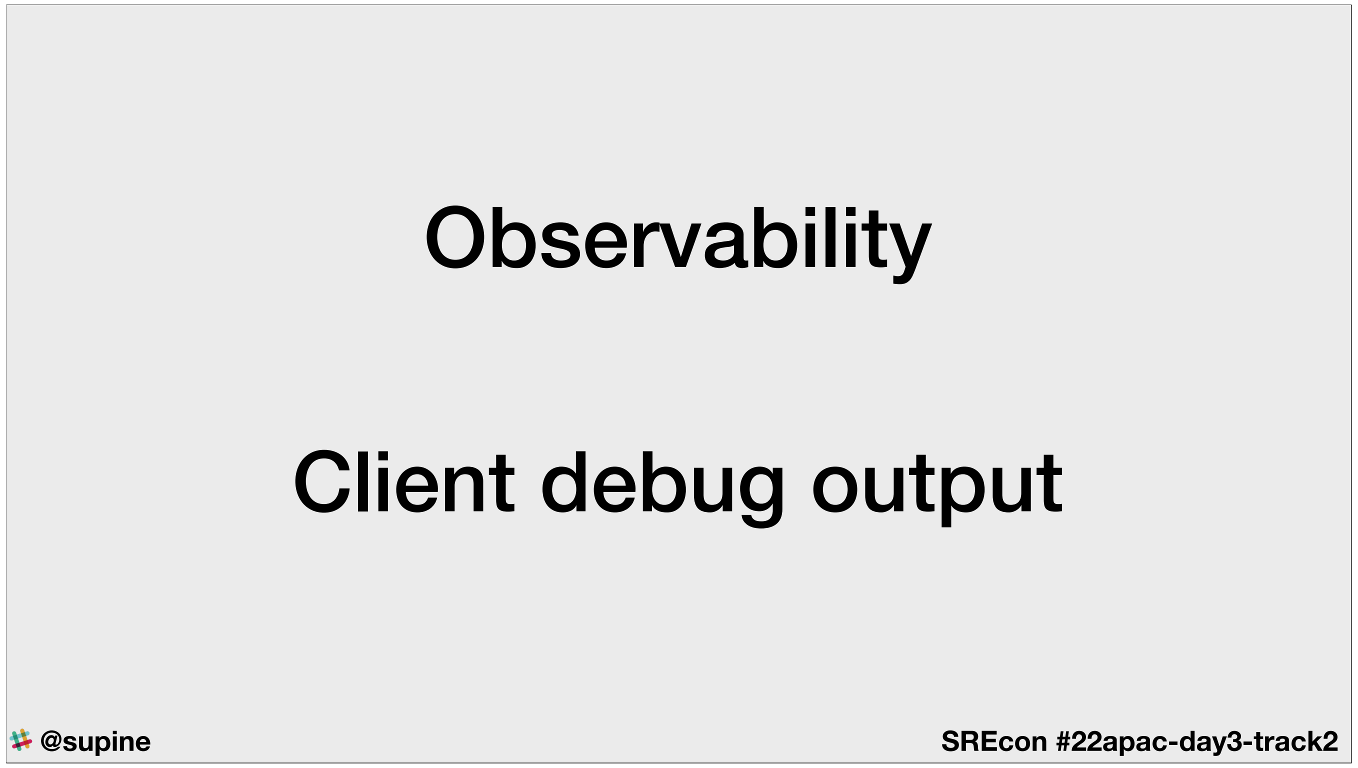
SREcon #22apac-day3-track2

Although the external probes mentioned so far can be very helpful, they still sometimes fail to capture the actual client experience as broadly as we would like.

This is where real user monitoring, or RUM, can come into play.

By instrumenting the client we can report on their side of the TCP session, capturing information we don’t have from the server side.

We can also incorporate experiments, like also testing a TCP session to alternative edge locations which we can then compare to the primary TCP session.



Observability

Client debug output

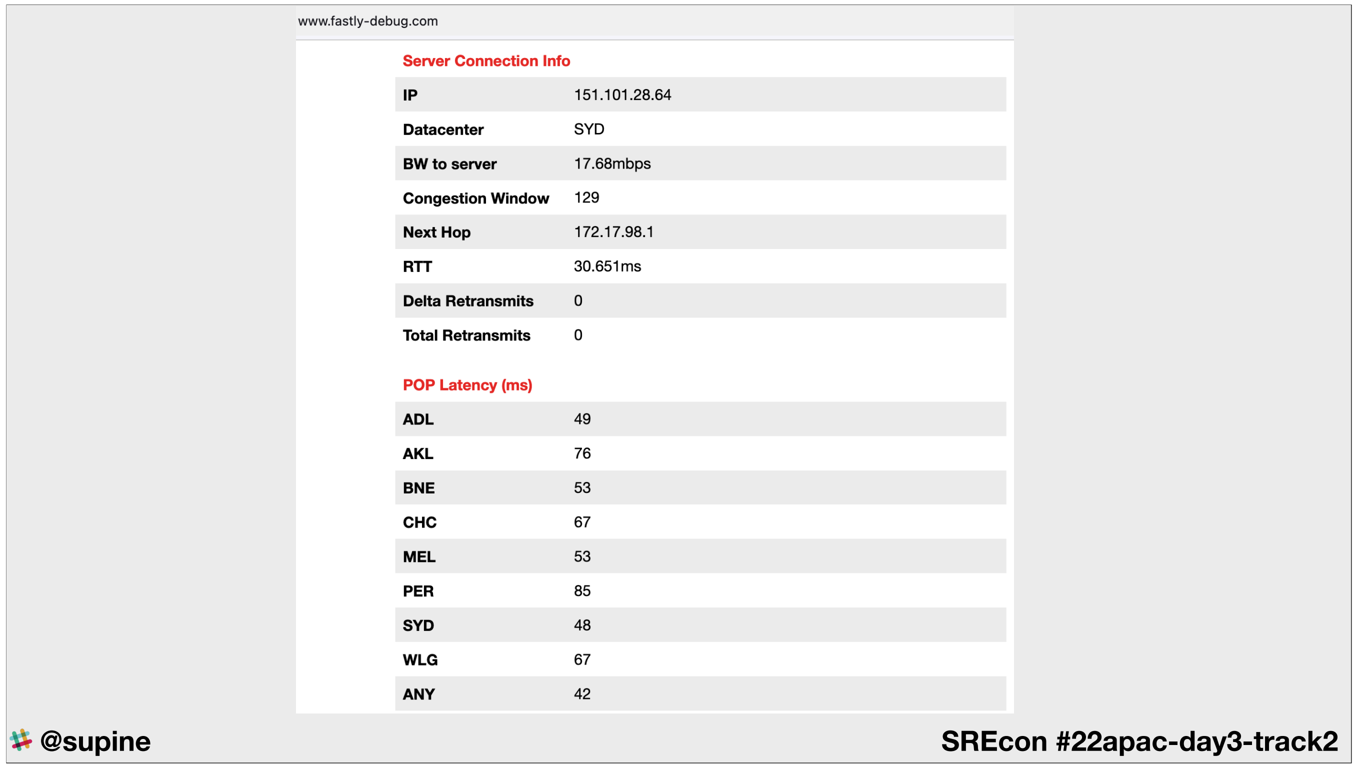
@supine

SREcon #22apac-day3-track2

Although the measurements we collect during production operations can help us analyse the system and troubleshoot problems, sometimes we need data specific to a particular client who is reporting a problem.

We could walk them through using operating system and third-party tools to take helpful measurements, but this can be hard with non-technical users.

Debug websites can do this for us, the client just needs to visit the URL and RUM-like tests obtain the measurements we need.



Server Connection Info

IP	151.101.28.64
Datacenter	SYD
BW to server	17.68mbps
Congestion Window	129
Next Hop	172.17.98.1
RTT	30.651ms
Delta Retransmits	0
Total Retransmits	0

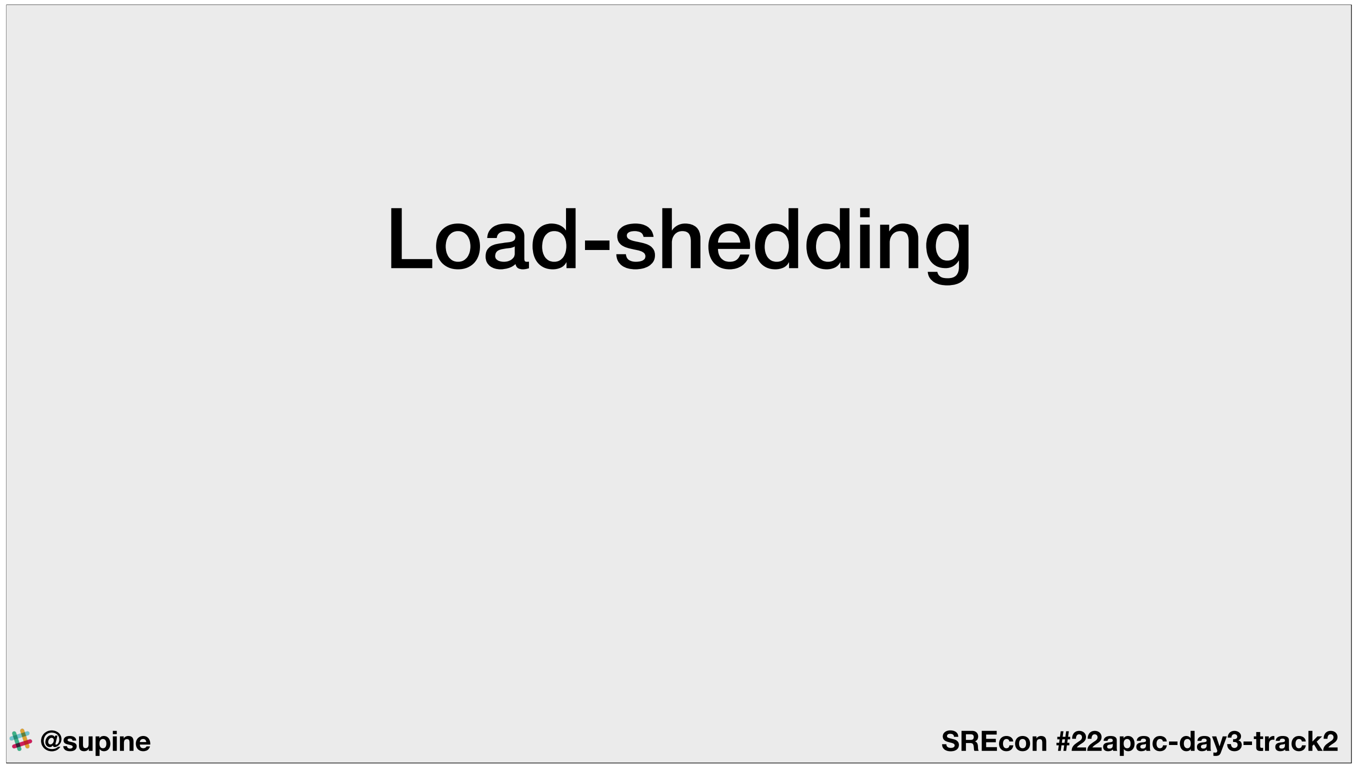
POP Latency (ms)

ADL	49
AKL	76
BNE	53
CHC	67
MEL	53
PER	85
SYD	48
WLG	67
ANY	42

@supine SREcon #22apac-day3-track2

Here is some sample debug output I took while tethered to my phone.

It nicely highlights that 4G networks have decent bandwidth but relatively poor latency.



Load-shedding

@supine

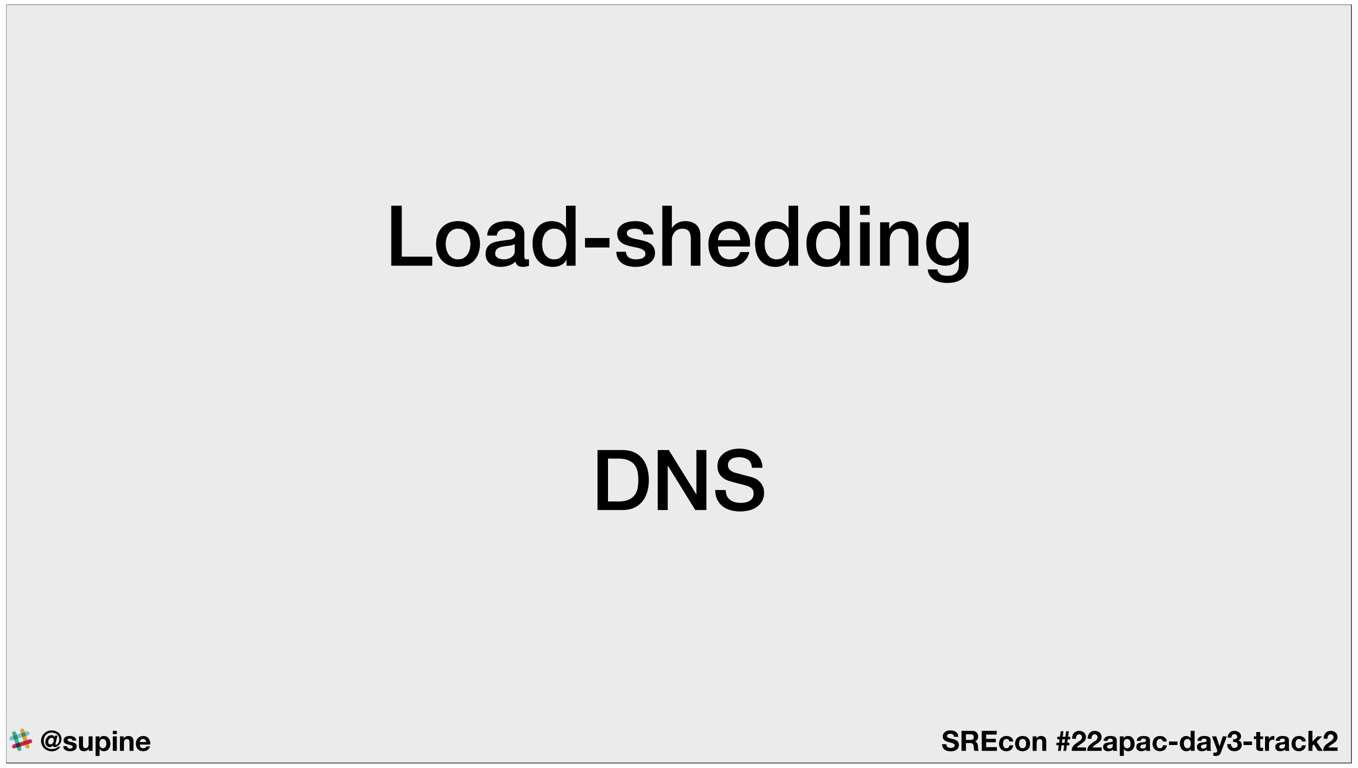
SREcon #22apac-day3-track2

So we did it. All that hard work defining, implementing and monitoring an internet scale load-balancer.

And in a steady state, everything is fine.

But now something has happened and we are seeing too much load arriving in one or more locations.

It’s not organic traffic growth so it doesn’t need a permanent solution but for the next hour or three we need to move traffic elsewhere.



Load-shedding

DNS

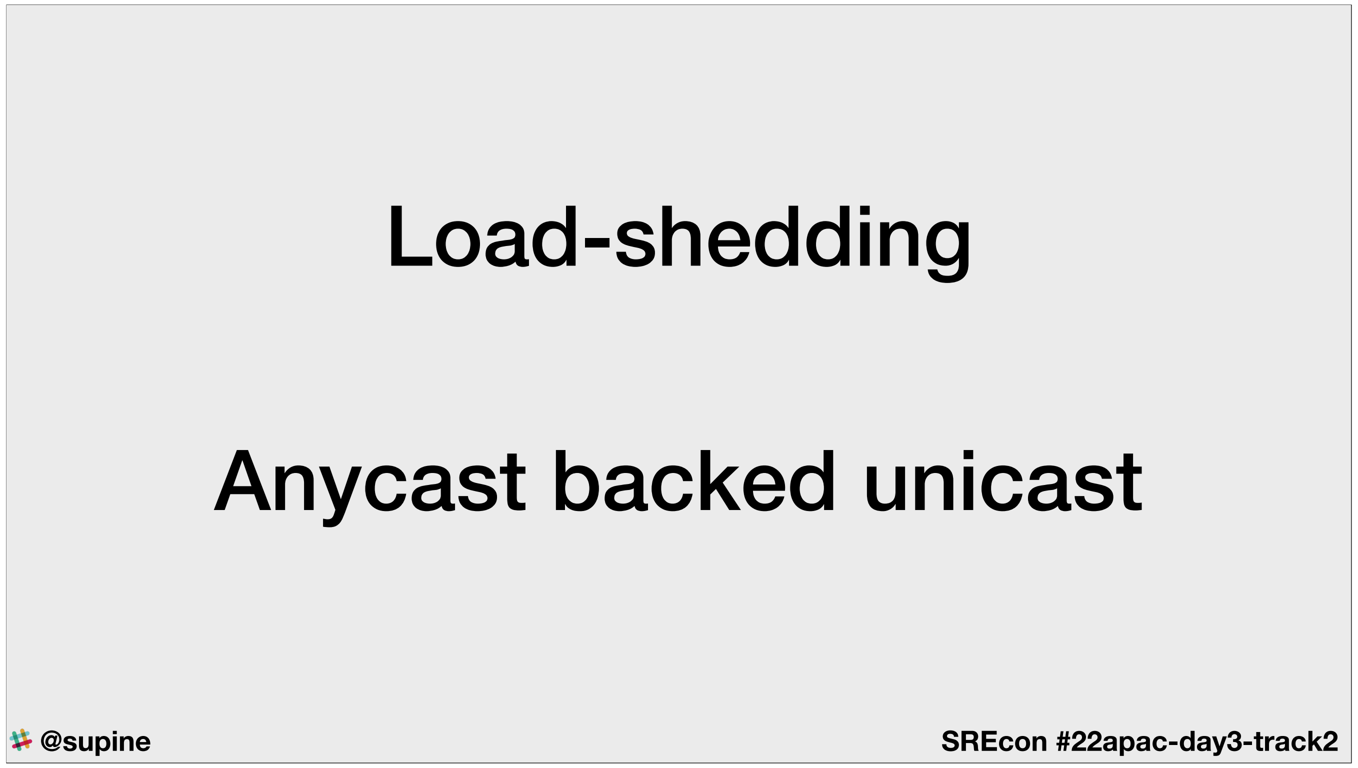
@supine SREcon #22apac-day3-track2

If we went with the DNS solution we can adjust some of the IPs we are returning to relevant resolvers.

Calculating how many responses need to change can be tricky as the ratio of traffic per DNS request varies greatly, from resolvers used by a single person at home through to ISP resolvers used by thousands or even millions of people.

Done early enough, any change will propagate through DNS caching before the traffic levels become problematic.

But what if we don’t have that much time? What if the traffic increases too quickly and we need to respond faster?



Load-shedding

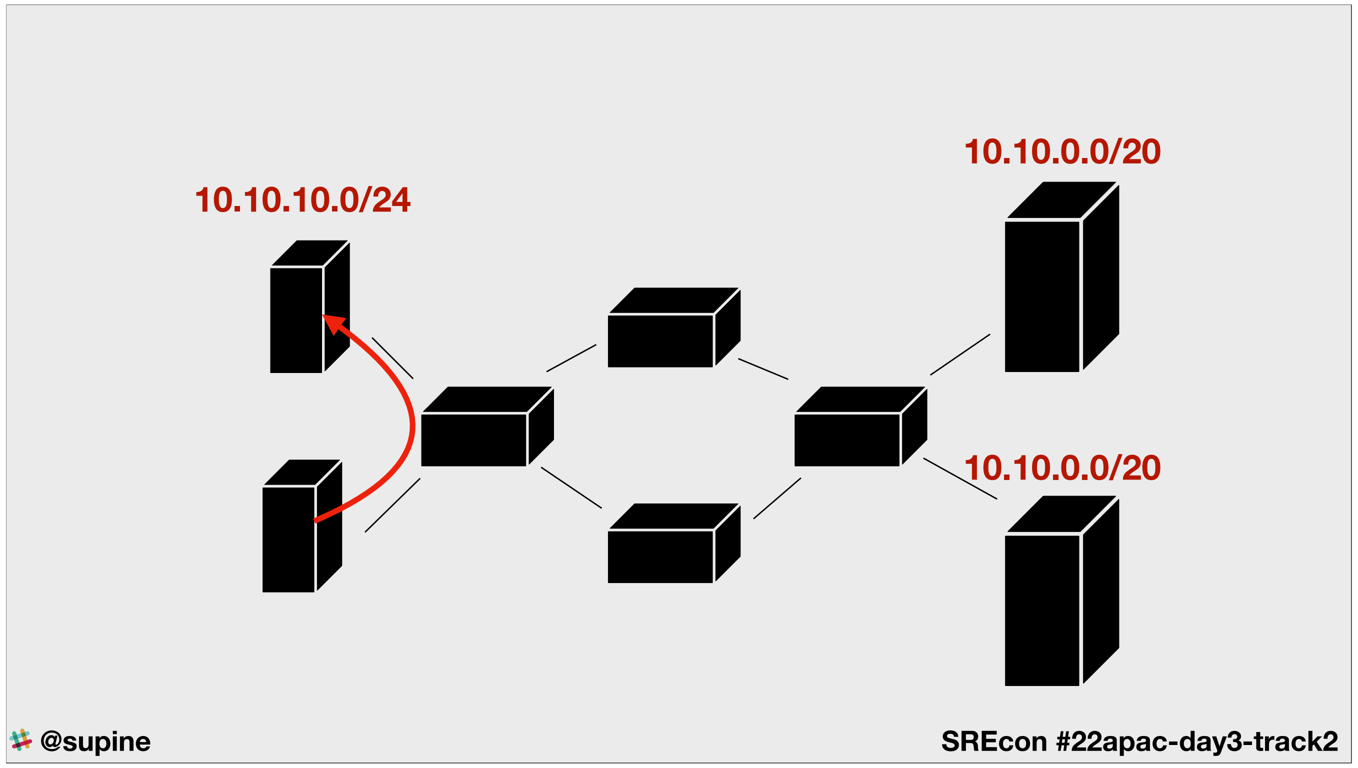
Anycast backed unicast

@supine

SREcon #22apac-day3-track2

This is where we can use anycast to backstop our unicast DNS solution.

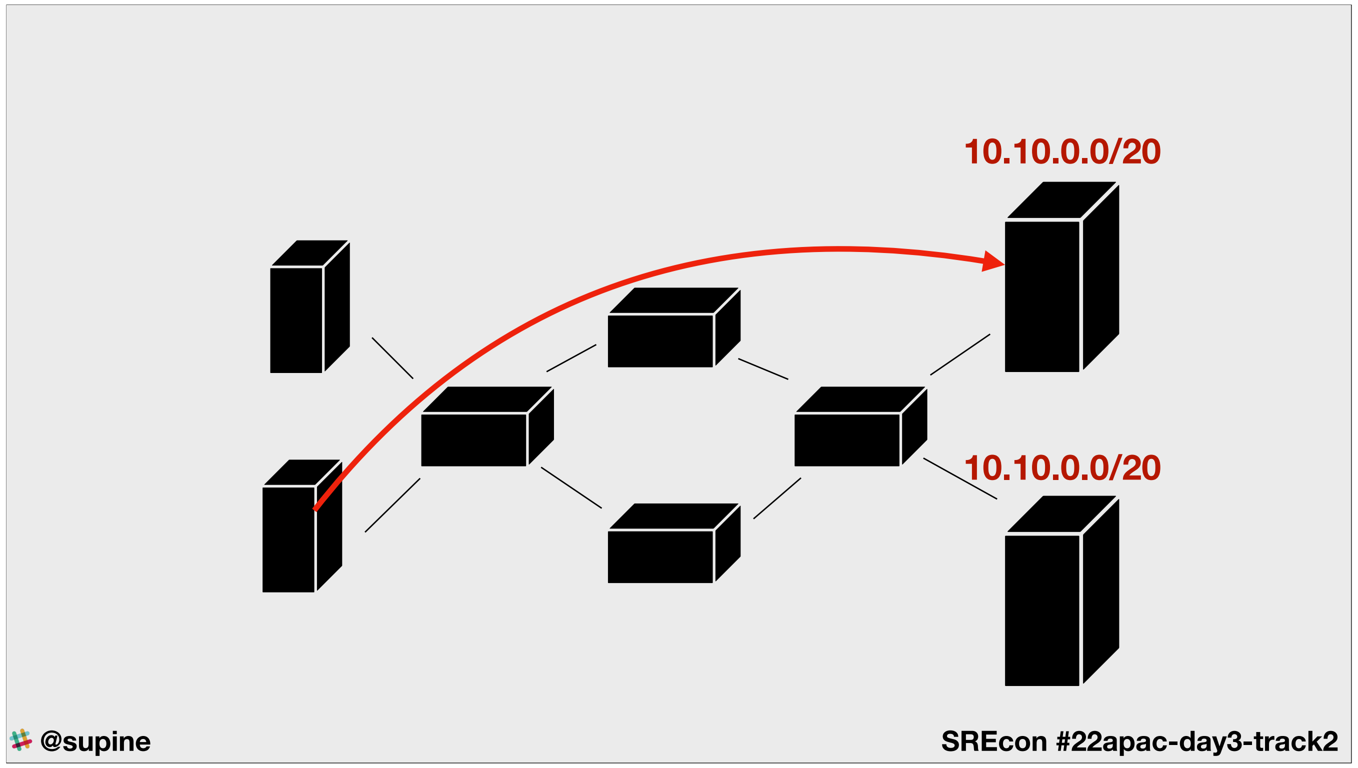
By switching from unicast to anycast during incidents we can stabilise the system quicker.



Presume that one unicast IP range assigned to a particular edge location was a /24.

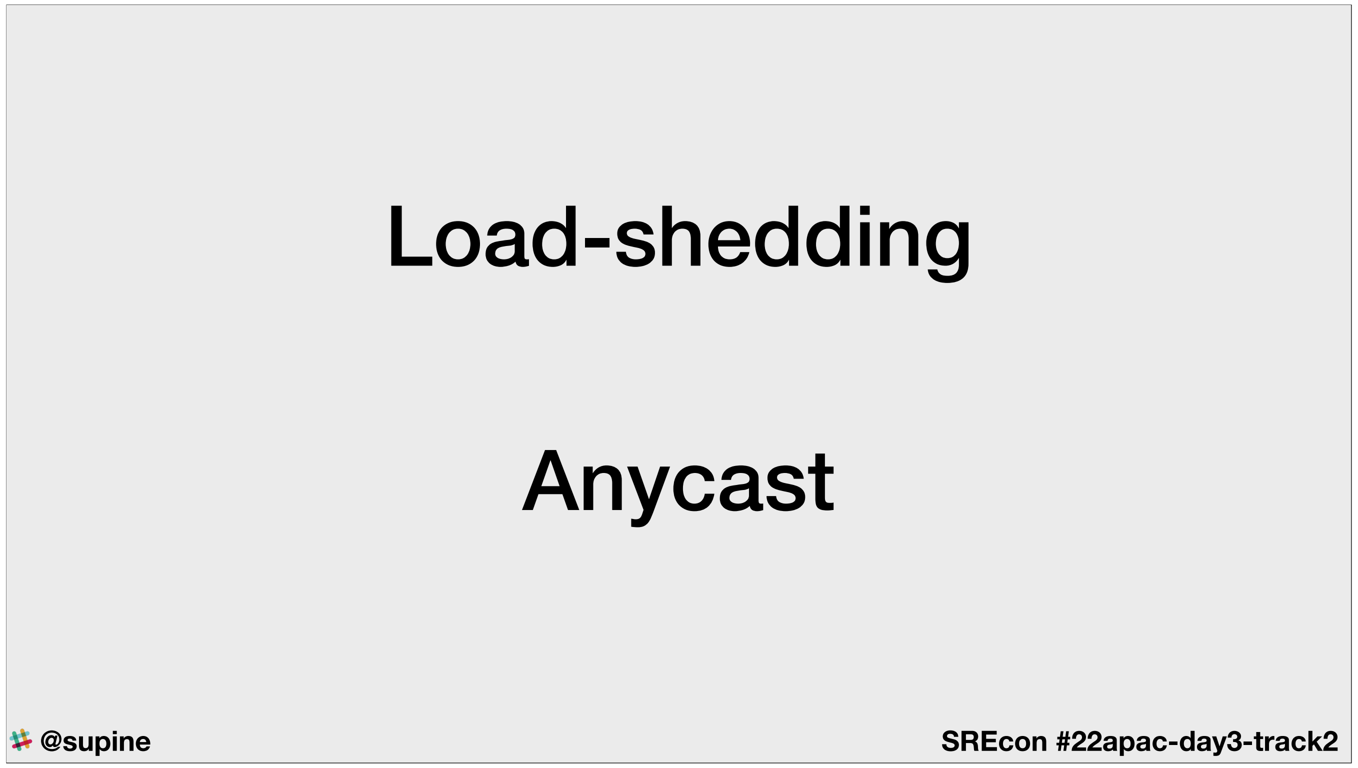
From larger edge locations with excess capacity we can announce the supernet, say a /20, that contains the /24.

If the original edge location becomes overwhelmed with traffic we can stop announcing the unicast /24 …



… and traffic will start routing via anycast to the larger locations that are announcing the /20.

Some TCP sessions will be interrupted but not as many as would be affected if we allowed traffic to continue to overwhelm the original edge location.



Load-shedding

Anycast

@supine

SREcon #22apac-day3-track2

If our original solution was anycast only, we can shed load by withdrawing some or all route announcements from the affected edge location.

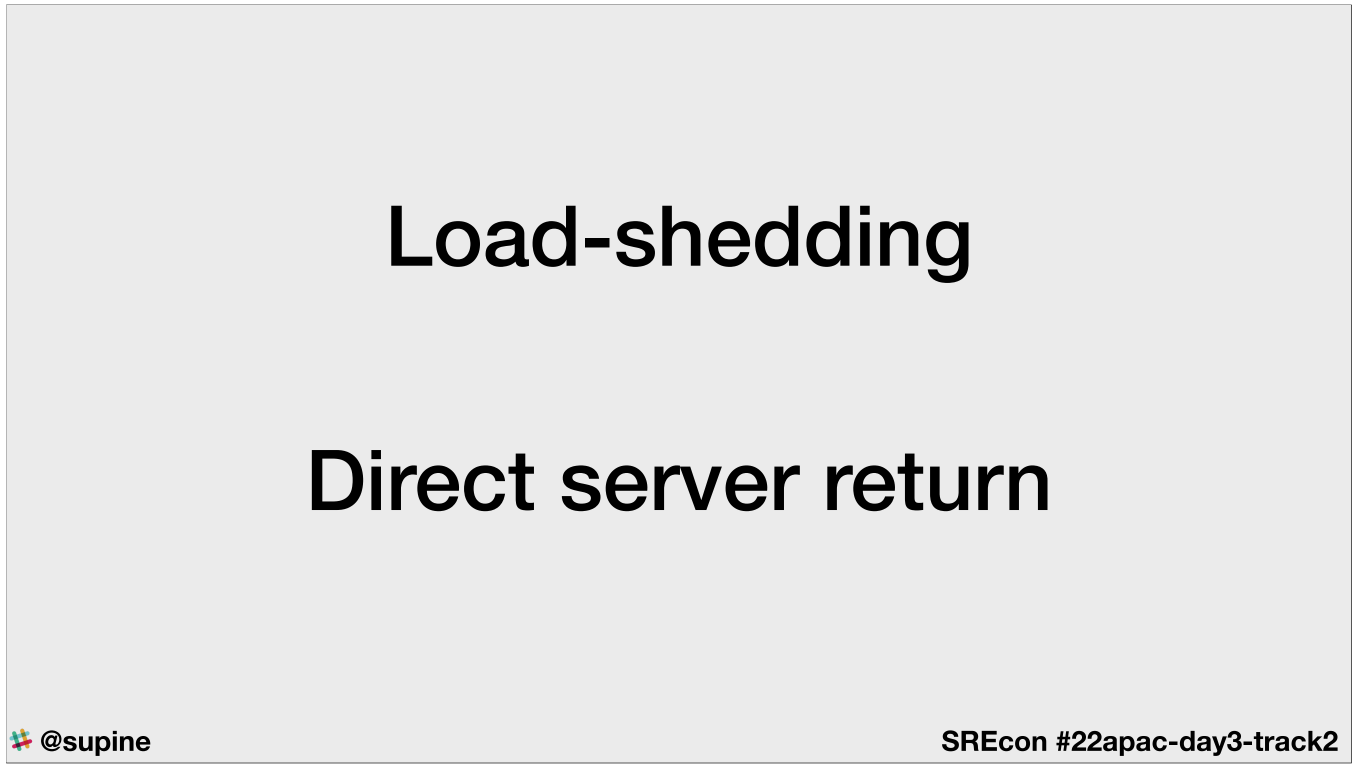
Traffic would ideally be rerouted to higher capacity edge locations or diffuse across enough other edge locations to not cause cascading load issues.

However, rerouting traffic in a nuanced fashion can be tricky.

On the internet a /24 route is the smallest IP range that will generally be accepted by other networks.

This means that you can generally only announce or withdraw routes in a minimum of /24 increments.

You have to make a tradeoff between a /24 being a proportionally large amount of your traffic in that location or you need a lot more /24s to shard your traffic over, wasting precious IPs and polluting the global routing table with more routes.



Load-shedding

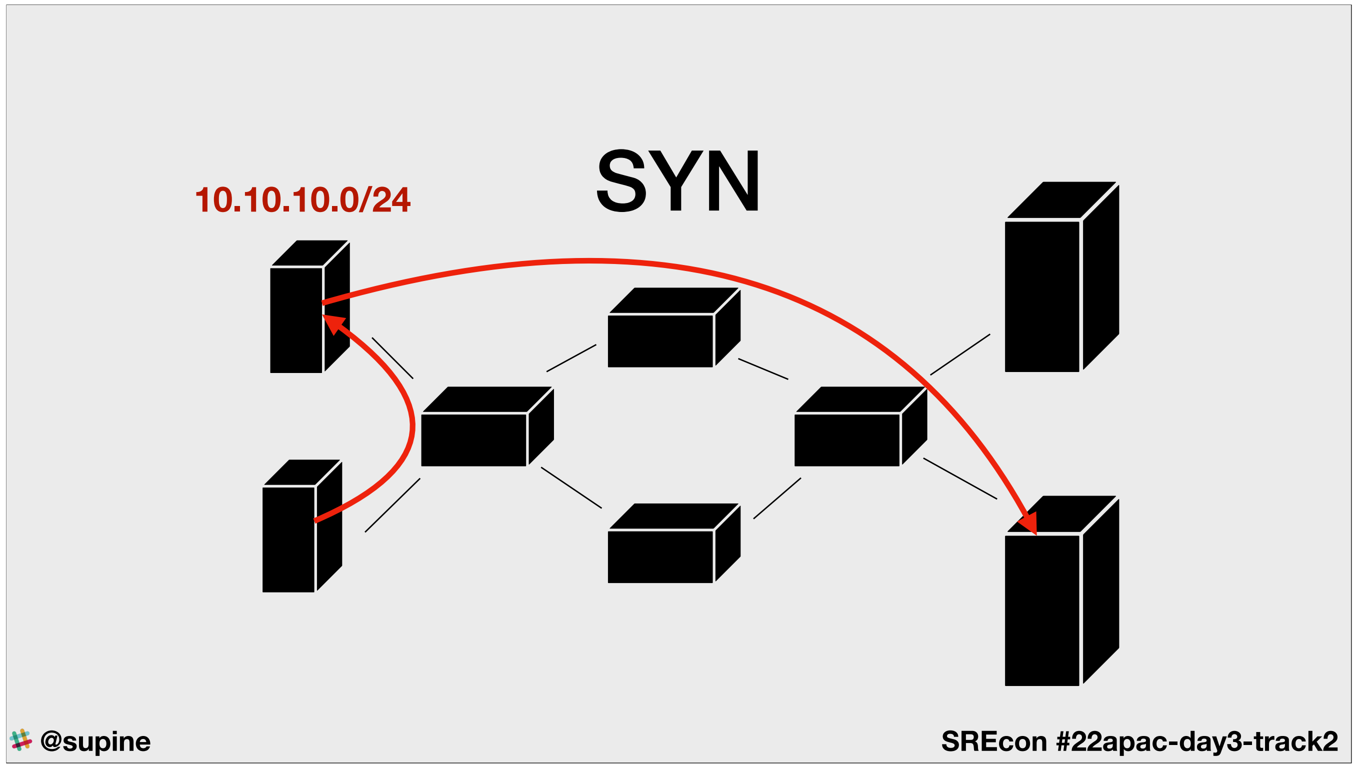
Direct server return

@supine

SREcon #22apac-day3-track2

The load-shedding techniques we just discussed relied on adjusting the ways we guide traffic into our edge locations.

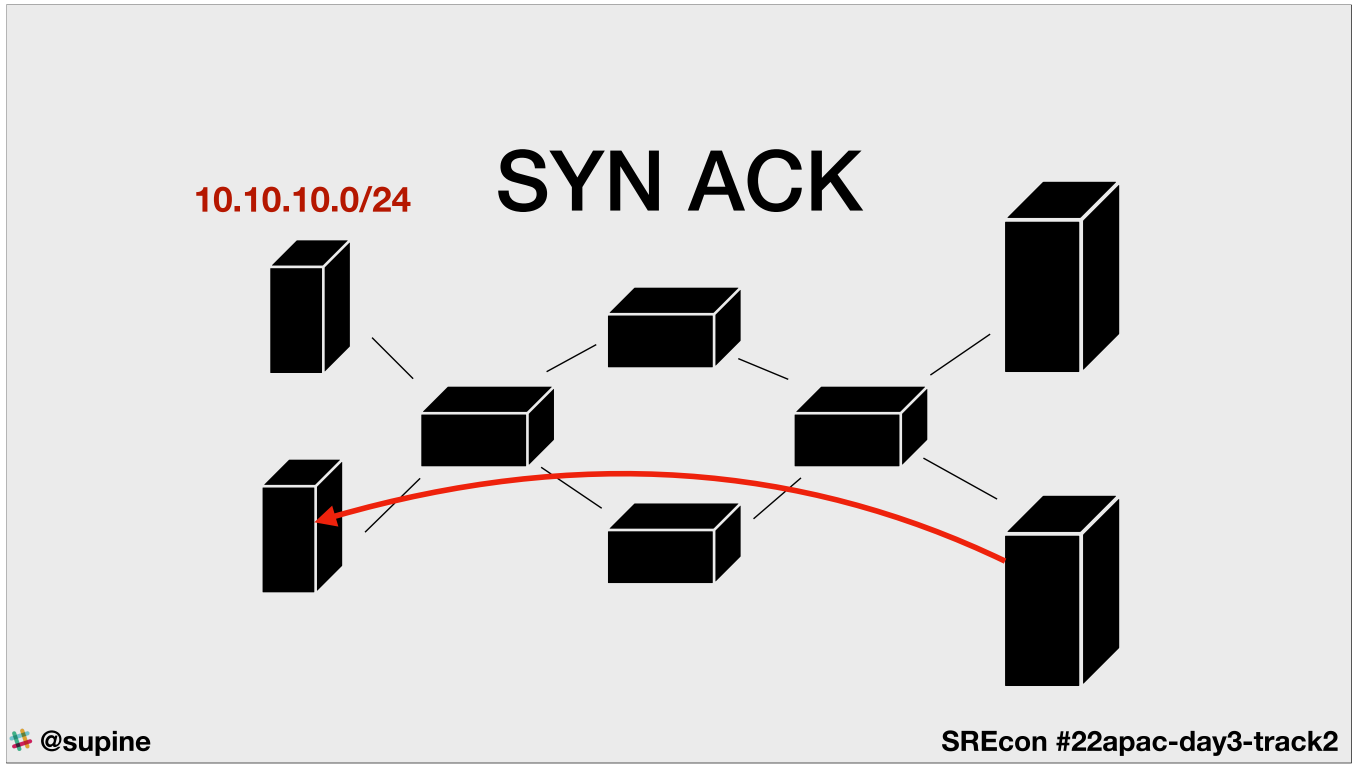
But another alternative is we can redirect some TCP sessions to a different edge location which then replies directly to the client.



SYN

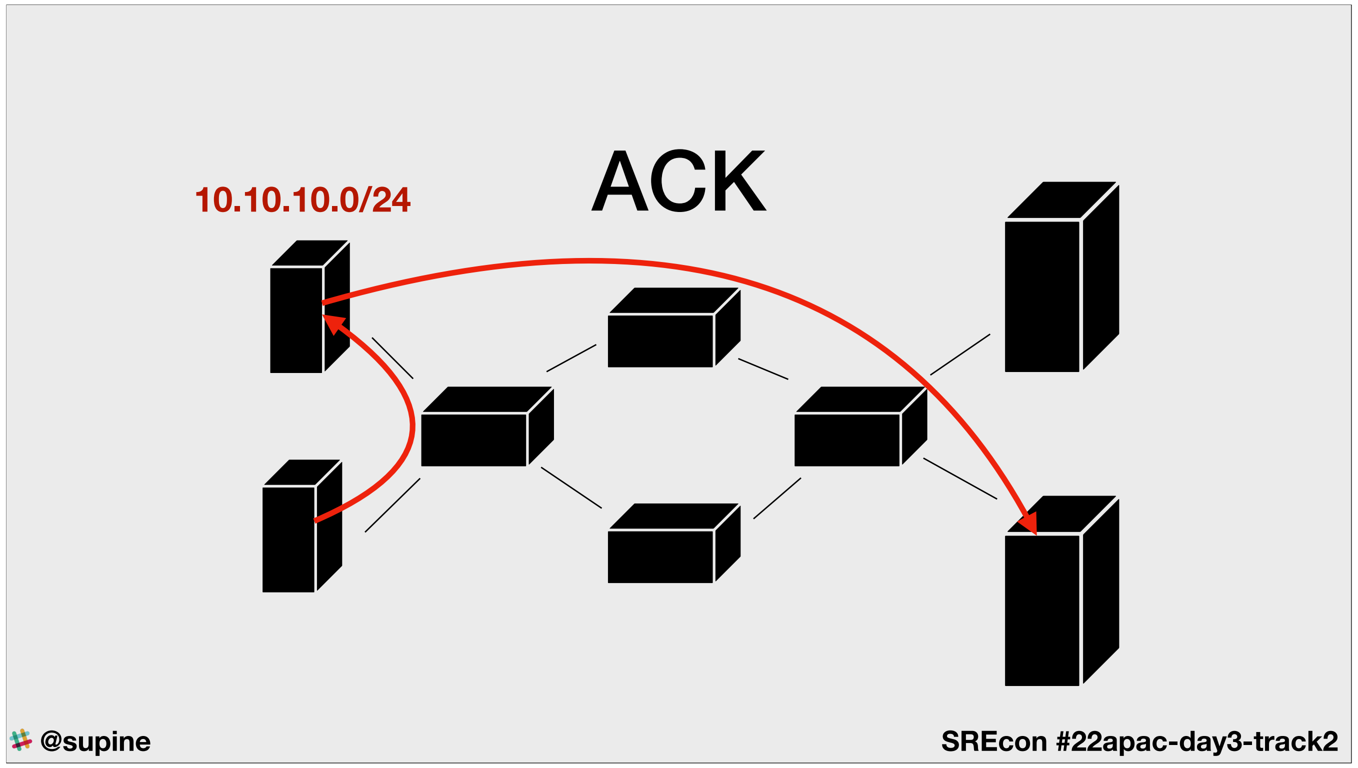
When the SYN packet arrives in the first edge location a BPF program can decide to shed this session.

It then forwards the SYN packet to another edge location with spare capacity.



SYN ACK

A server in the second edge location accepts the connection and sends a SYN ACK packet directly back to the client.



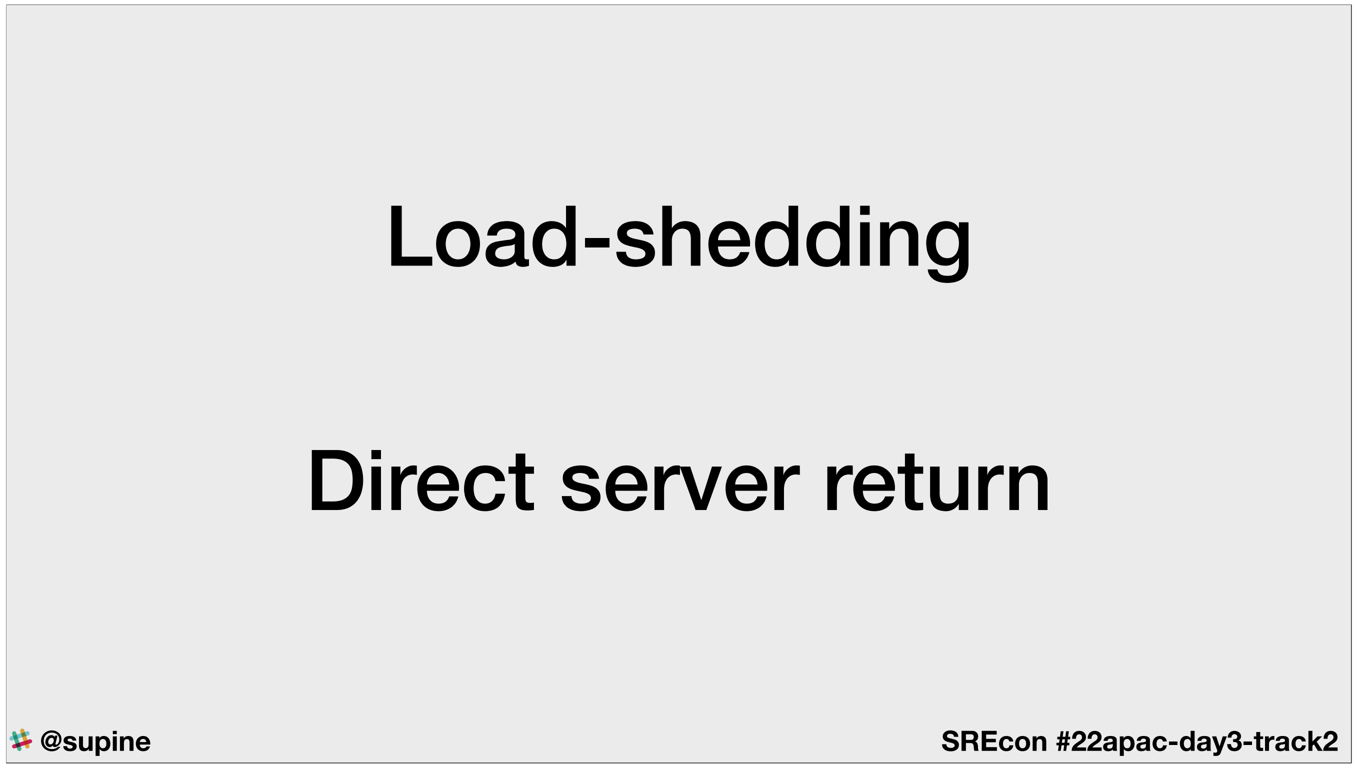
ACK

@supine

SREcon #22apac-day3-track2

The client responds with an ACK packet which the original edge location forwards as well, completing the handshake.

The TCP session then proceeds, with the triangulation ongoing through the life of the connection.



Load-shedding

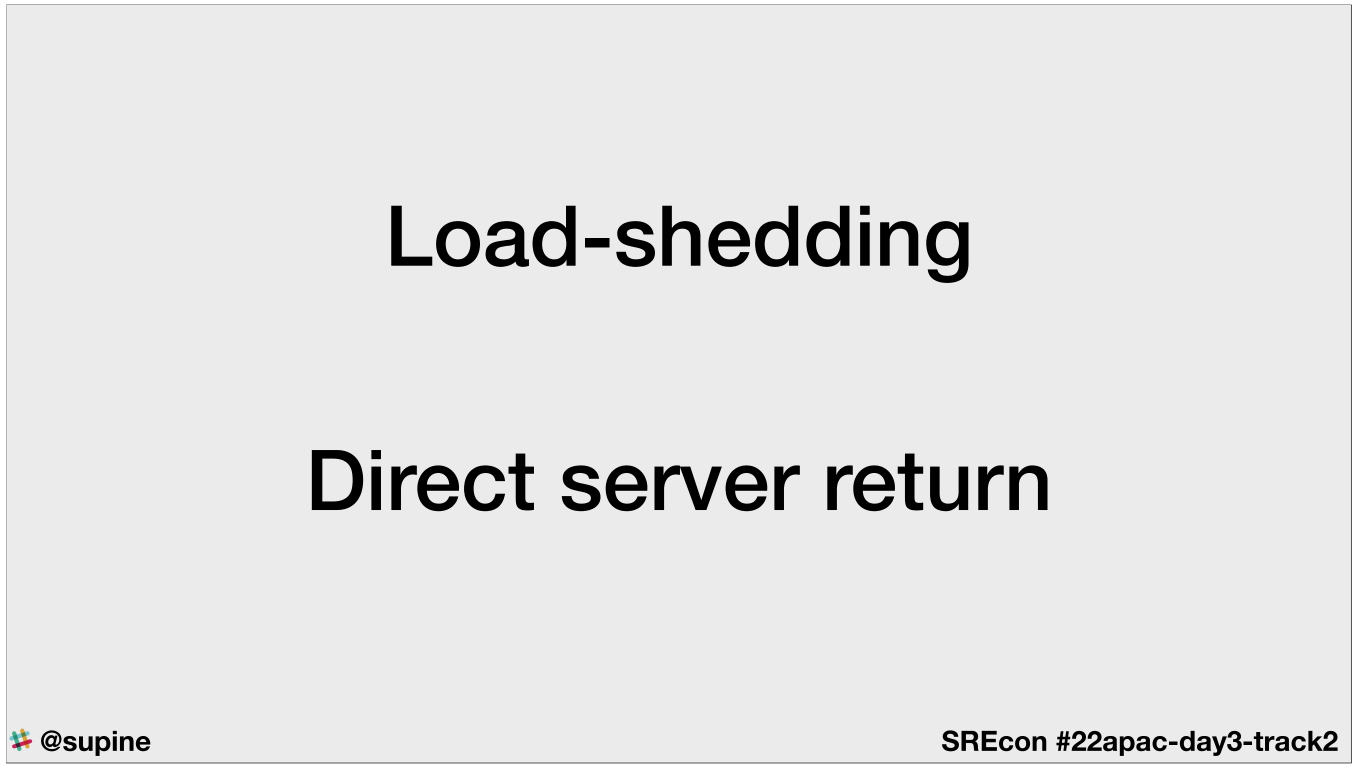
Direct server return

@supine SREcon #22apac-day3-track2

Load-shedding via direct server return allows us to be extremely nuanced about how much traffic is shed.

Because we can differentiate down to individual TCP sessions, what we shed can be based on anything we know about that connection.

Client IP or range, client RPS, traffic type, anticipated traffic volume, anticipated session duration and many more.


Load-shedding

Direct server return

@supine

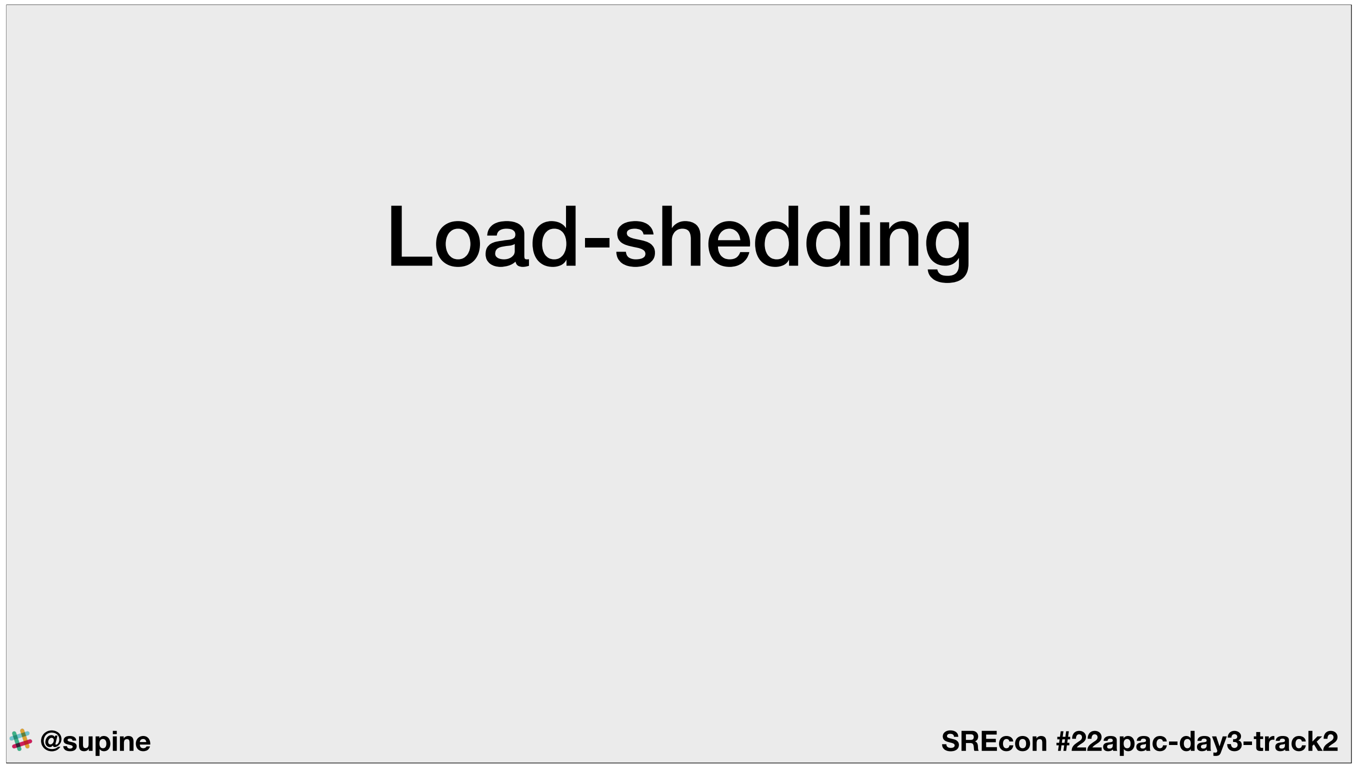
SREcon #22apac-day3-track2

Direct server return has very few downsides but it does require we spend some precious resources at the overloaded edge location forwarding packets to the second location for the life of each session.

It also means both locations are points of failure for the redirected sessions and both need to be healthy enough to participate in the ongoing triangulation.

And we have likely added some latency, sometimes a little due to just the triangulation, sometimes a significant amount due to the second edge location being further away.

The other edge location might also not have the best paths back to the client due to being in a completely different topological location.

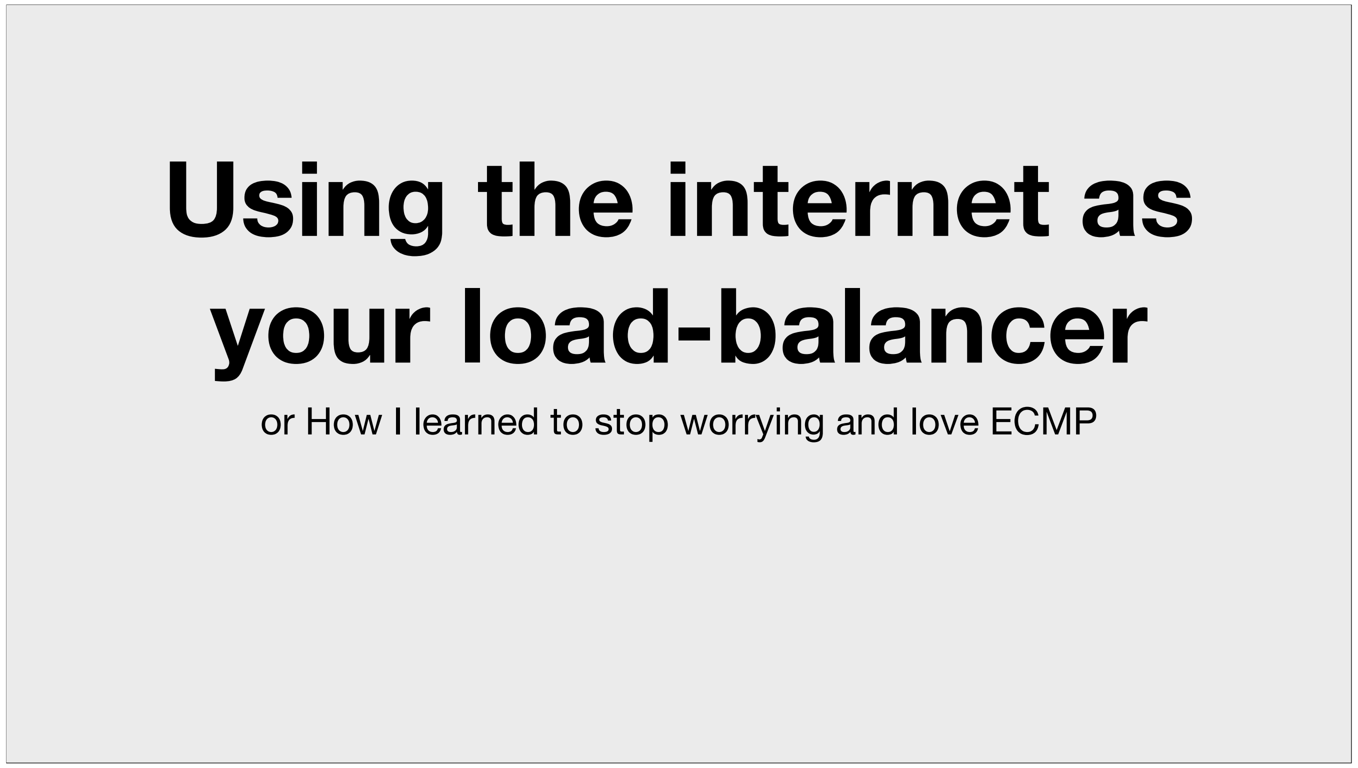


Load-shedding

@supine

SREcon #22apac-day3-track2

Those are the ways we can shed load, either changing how requests arrive at our edge locations or forwarding some requests to a different edge location.



Using the internet as your load-balancer

or How I learned to stop worrying and love ECMP

To wrap up I'd like to revisit the subtitle of my talk that was poking a little fun at ECMP.

ECMP is everywhere in the networks of today.

There is ECMP in ISP networks.

There is ECMP in global backbone networks.

There is ECMP even inside edge locations.

It's ECMP all the way down.

Thank you!

supine.com/srecon22apac

marty@supine.com

@supine

@supine

SREcon #22apac-day3-track2

…and that's all I have for you today.

I hope I've been able to explain a little about what it takes to operate a service at internet scale and given you some things to think about as you return to your own work next week.

You can find my slides at that URL.

Thanks for listening and feel free to chat to me in the hallways or contact me online.