

Download

Download materials to play along

<https://github.com/rohess/sfeu25>

- Slides
- Sample Pcap
- Wireshark Profile



Making WebRTC work in times of locked down networks

Robert Hess

#sf25eu



github.com/robert.hess/sfeu25

On a quest to help customers to fix
their network communication issues

Quality Inquisitor @GoTo – the makers
of GoToMeeting & GoToWebinar

My perspective is one of a
troubleshooter – requirements,
problems, solutions

robert.hess@goto.com

© robert.hess@posteo.de

3

When I started thinking about what to tell today it was originally some generic trouble shooting talk, and I was not overly happy with it. But then, I encountered a rather fascinating problem at the end of the summer, and this made for a more interesting story – so I will tell you today essentially of my work of the last 2 months.



Complex mechanism on WebRTC but also on the network side +
dynamically changing environment = problems

It all began ...

- Ongoing reports of annoying reconnects at in the middle of the meeting
- Increase over the last year
- Happens also in our own company
- Network captures point toward dynamic changes in network connectivity
- Perusal of Zscaler logfiles point to its tunnel handling

it all began ...

- Not so much about boring config for packet filters, but rather about the dynamic aspects of today's networks.
- Disrupts in-sessions audio & video or fully disconnects your session.

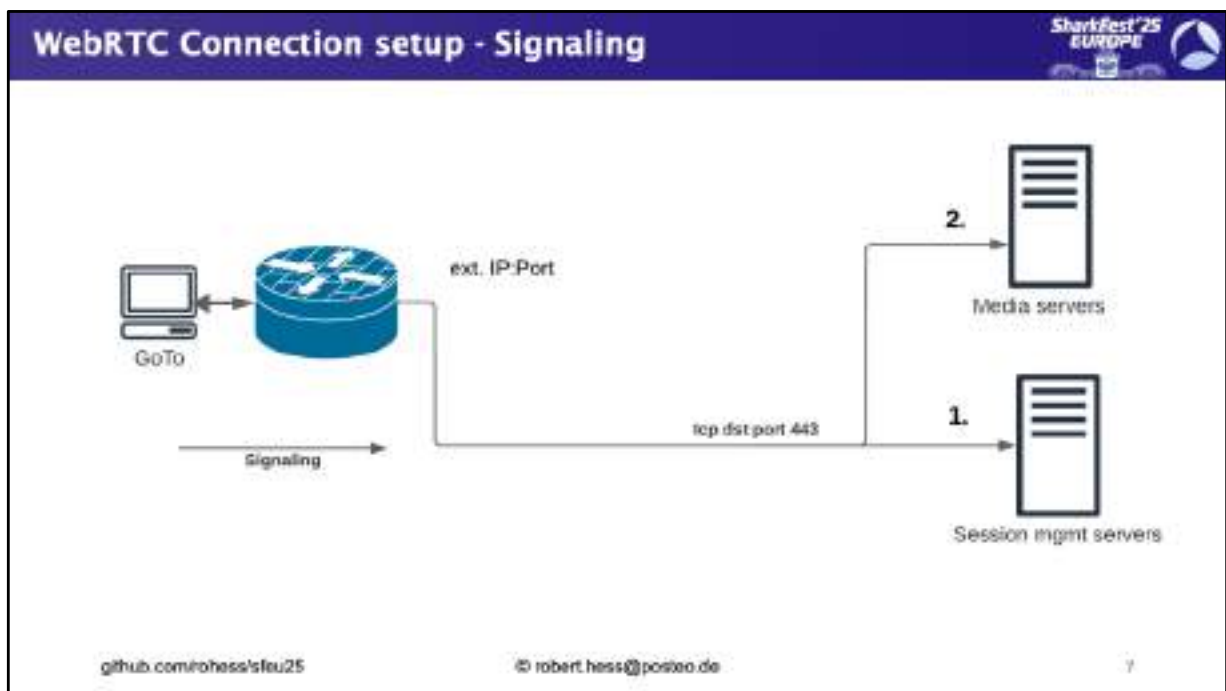
Puzzle: which packet contains the SDP offer from Client to server with an ICE candidate in it?

In a nutshell:

The way our packets take through the network changes over time. This may interrupt online meetings.

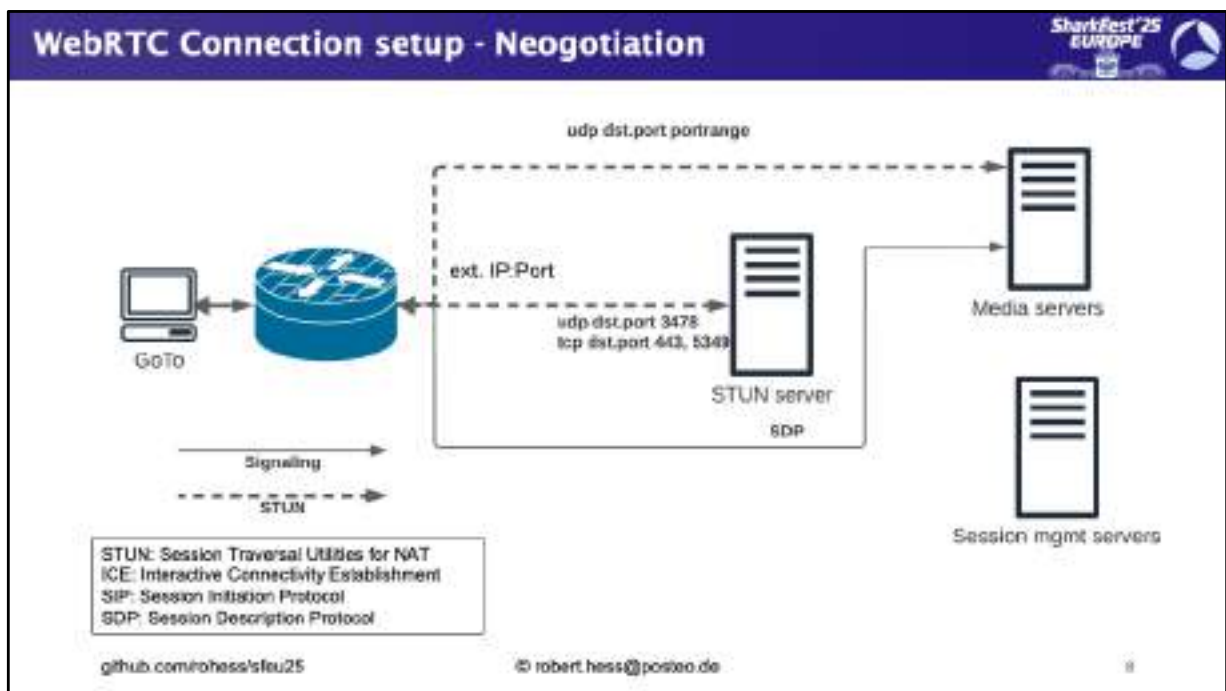
Puzzle for those who know WebRTC already
There are clues in the slides

Lets recap WebRTC connection setup
briefly

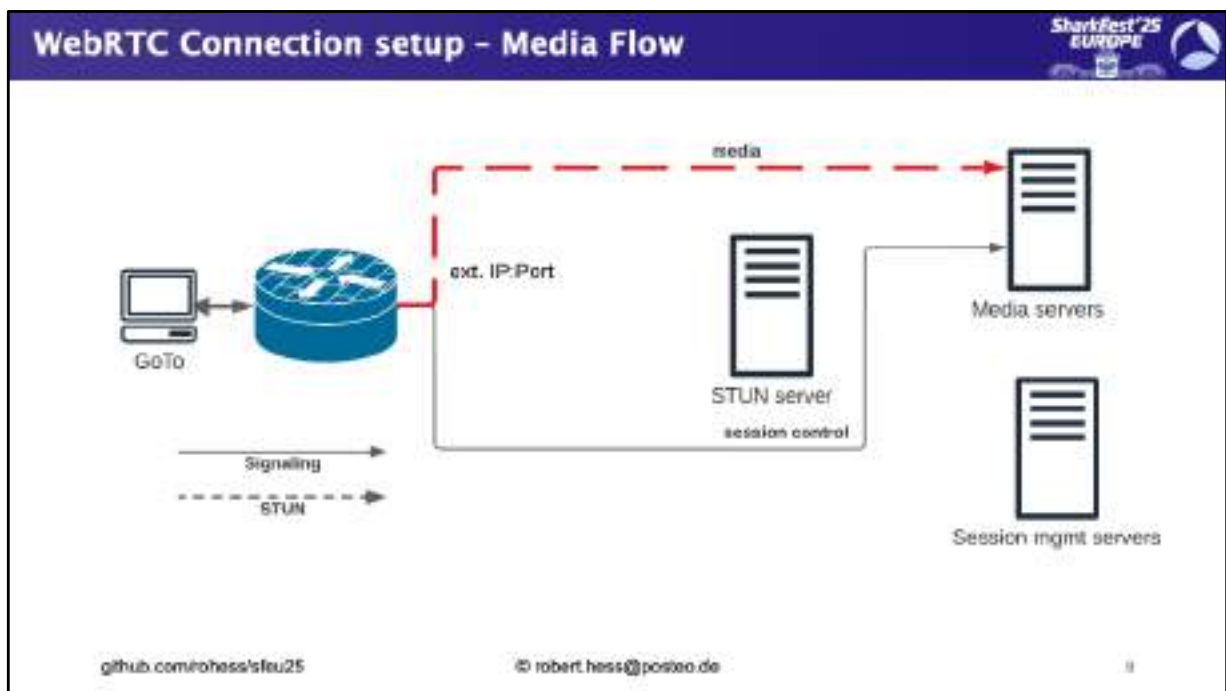


Client is behind a NAT (lets assume IPv4 for now)

- 1) Fetch address of media server
- 2) Media server sends back peer connection config



Happens all in parallel and repeatedly
Keeps going in the background once connection is established



Media will flow

- The red dotted line uses the same address port as the STUN before – its essentially the same UDP stream
- Its all client-server these days, but the machinery is more complex to support p2p
- P2p is no longer a thing if everyone uses Zscaler and traffic is tunneled
- This is the same for audio, video, screensharing

- STUN requests: probing packets to test whether there is a reply via a specific path (filter *stun*)
- Each side (client and server) create local and remote ***candidates*** – which are ***ip:port:protocol*** combinations
- Each side figures out which combination it likes best – one side is making the final decision which is used by both sides
- Ideally this is the same in both direction. Chrome bug currently leads sometimes to asymmetrical connections

```
{ "type": "offer", "sdp": "v=0
...
a=candidate:640760911 1 udp 2122260223 192.168.111.55 57261
typ host generation 0 network-id 1
```

github.com/robert-hess/sfou25

© robert.hess@posteo.de

11

- a=candidate:3974728418 1 udp 2122129151 192.168.66.34 59783 typ host generation 0 network-id 1 network-cost 10
- Asymmetric connections are no problem, as the receiver will still send back STUN pings – this keeps NAT bindings alive
- They are however confusing
- Much simplified - there is a whole RFC about this with priorities etc.



a free interpretation of a comment in the COTURN Turn server
==== Show him the instruments, Practical Frost: ====

Its really worth to read the programmers comments on some snide remark
on their code:
<https://github.com/coturn/coturn/issues/422>

quote from
The Blade Itself
Book one of The First Law
by Joe Abercrombie

” ‘Show him the instruments,’ whispered Glokta.”




you can see everything here

The universal tool to get insights into your WebRTC connection

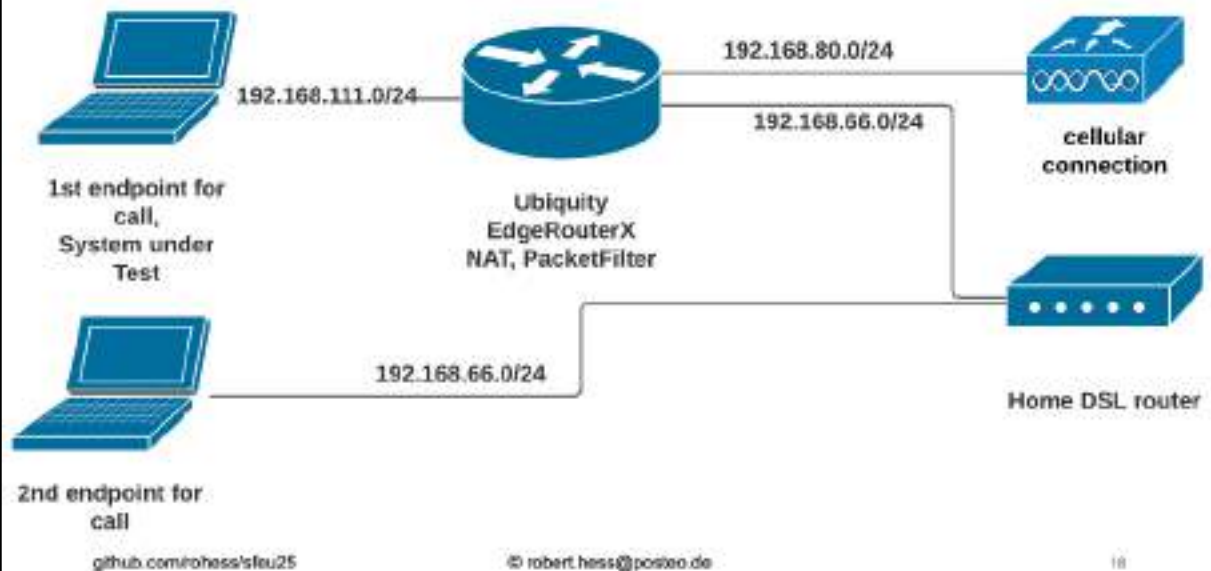
Only available during call

WebRTC internals dump to save stuff

ICE Candidate grid					
					
▼ ICE candidate grid					
Candidate (pair) id	State / Candidate type	Network type / address	Port	Protocol / candidate type	(Pair) Priority
CPQN6TMn8r_lRenkvsy	succeeded	ethernet		udp	0x7
lQN6TMn8r	local-candidate	87.142.197.72	49676	prflx	0x6e7f1eff 110 32542 255
lRenkvsy	remote-candidate	23.239.237.149	45046	srflx	0x647f3eff
CPC50Hkw/E_lRenkvsy	succeeded	wifi		udp	0x7
lC50Hkw/E	local-candidate	87.142.197.72	59783	srflx	0x647d1eff 100 32030 255
lRenkvsy	remote-candidate	23.239.237.149	45046	srflx	0x647f3eff
<ul style="list-style-type: none"> • Two successful connections • Ethernet gets preferred over Wifi • peer-reflexive or server-reflexive depends on RTT to servers 					
github.com/rohess/sfeu25		© robert.hess@posteo.de		14	

Very simple case

How to analyze - you can do this at home



- Consider capture points/tools
- Export TLS keys (Tools->TLS Keylog Launcher / Edit-> Inject TLS Secrets) – you can use Wireshark for launch and still capture elsewhere
- Run clean capture
- Open chrome://webrtc-internals
 - Check ICE Grid
 - Create WebRTC-Internals dump – you can get nice stats from it via <https://rtcstats.github.io/rtcstats/dump-importer/>
- Get application logfiles

Why decrypt? - You want to see the Signaling, SDP etc.

You can use Wireshark's keylog launcher to start your browser in question – saves you the hassle with the CLI ... and still capture with builtin capture, i.e. from Zscaler, open this pcap later in Wireshark and import SSLKEYFILE

How to analyse -3

- Restrict depending on what you want to investigate
 - Block UDP except Turn Server ports/addresses
 - Block all UDP except DNS
 - Block all UDP (you will most likely still need a DNS entry for your TURN server - use local hosts file)
 - Configure a proxy and block all TCP except proxy
 - Configure your proxy with auth
 - Configure your proxy for DPI and install matching certs on your system
- For most of the following experiments, I just needed my work PC in the office, flapping between Frankfurt and Copenhagen DCs
- Sample capture was done on my own setup
- Packet capture done in Zscaler UI

application logfiles if you can get them.
you can also intercept remote logging/telemetry


I did capture on the machine to see the insides of the zscaler tunnel

Capture Packets when running Zscaler

Capture in Zscaler UI

Two files on Windows:

- CaptureAdapters_2025-10-26-07-20-07.XXX.pcapng- you will see the DTLS Tunnel connection
- CaptureLWF_2025-10-26-07-14-29.XXX.pcapng - inside the Tunnel - you will see packets twice
- Check the comments



The screenshot shows the Zscaler Internet Security application window. In the 'Troubleshoot' tab, 'Start Packet Capture' is highlighted with a red box. To the right, the 'Configure Packet Capture Options' dialog is open, showing settings for 'Run Session For' (1 hour), 'Disk Space Limit' (1 GB), 'Frame Size Limit' (1514 bytes), and a 'Packet Capture Filter' field. The 'Start' button is highlighted in blue.

```

TUNNEL2_RESPONSE_PACKET
ORIGINAL_PACKET,Filter Id: -1
TUNNEL2_REQUEST_PACKET,Forward Packet to T2 as per include/exclude
TUNNEL2_RESPONSE_PACKET
TUNNEL2_RESPONSE_PACKET
    
```

github.com/rohess/sfeu25
© robert.hess@posteo.de
19

uses same npcap driver as wireshark
 may lead to version conflicts
 Comments are only in the CaptureLWF



The applicants

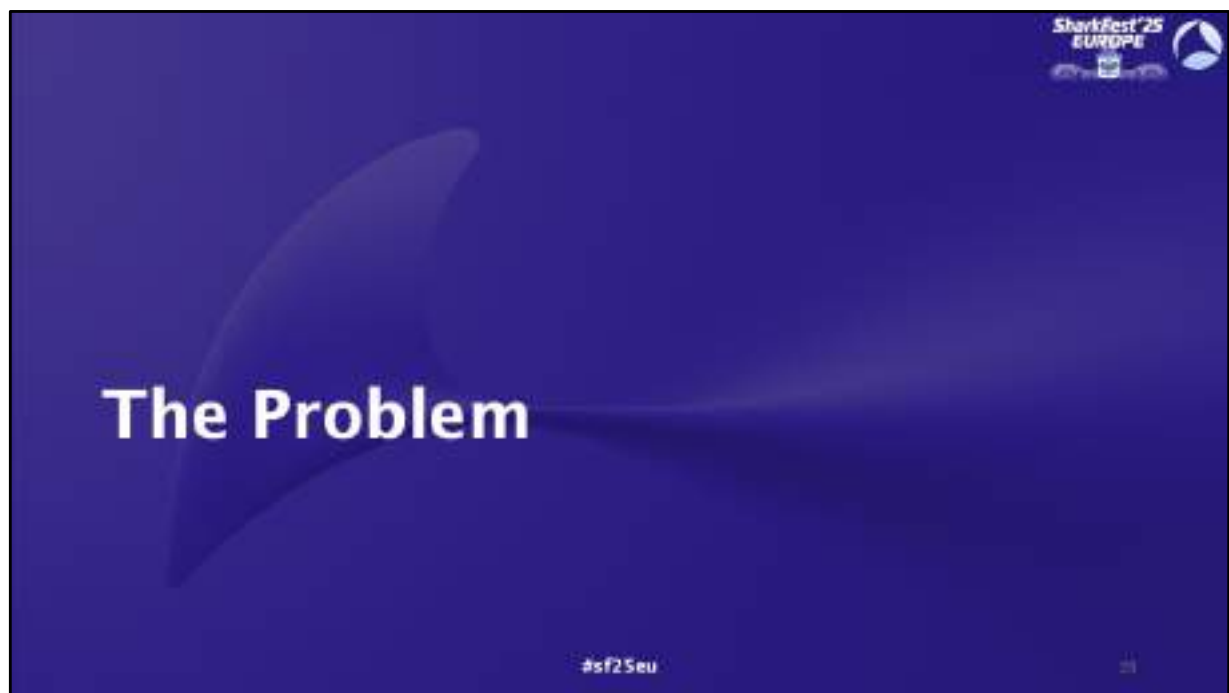
#sf25eu

101

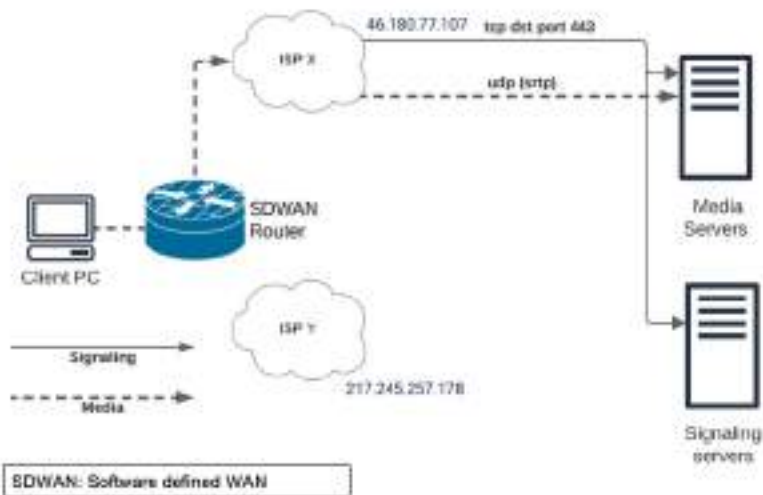
What are we looking at

- Most common WebRTC conferencing solutions
 - MS Teams
 - Zoom
 - Google Meet
 - Webex
 - GoToWebinar
- Running in Chrome / Windows

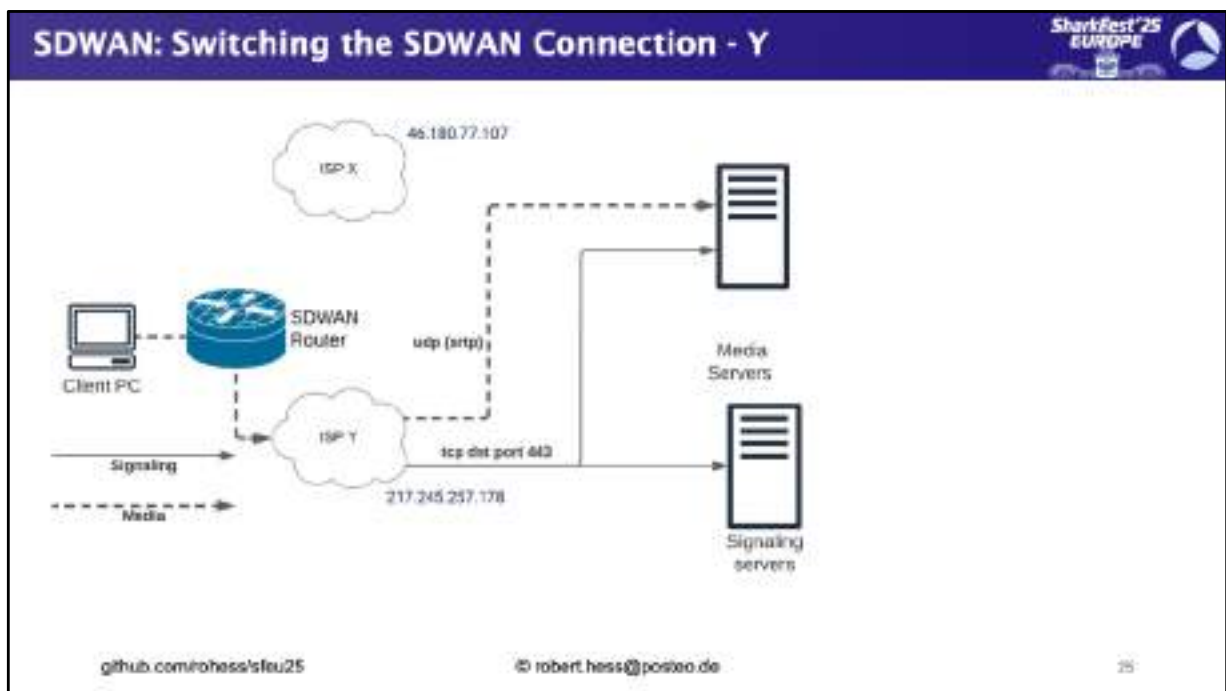
Running this in the browser forces everyone to use WebRTC – Desktops apps might not always do this.



SDWAN: Switching the SDWAN Connection - X



again, we have several steps

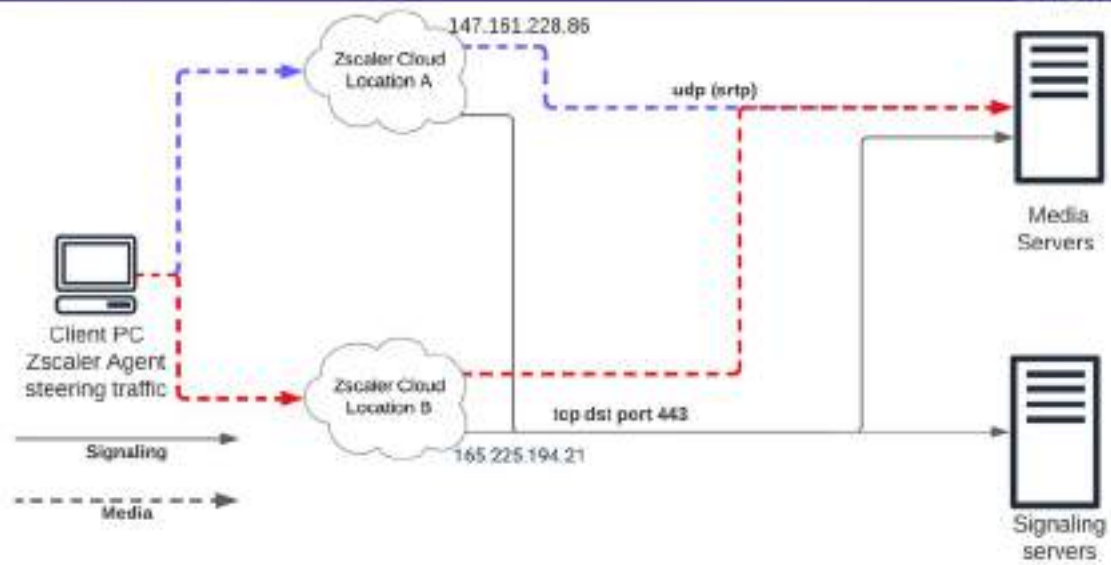


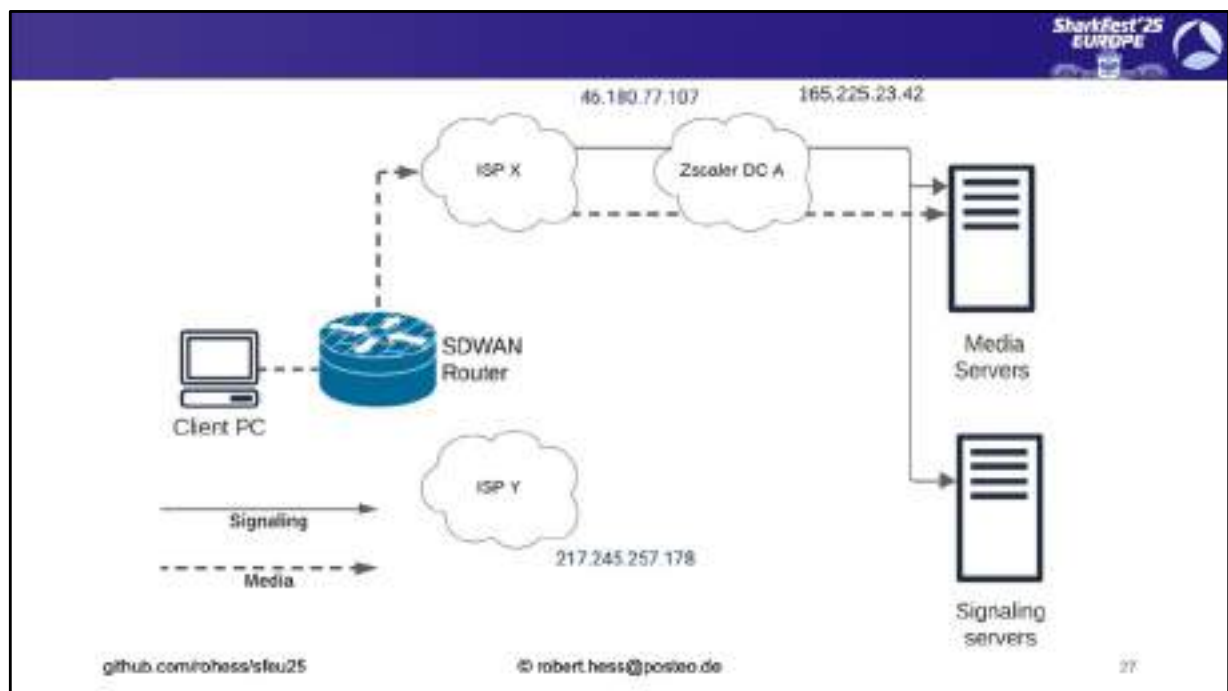
this is active/active

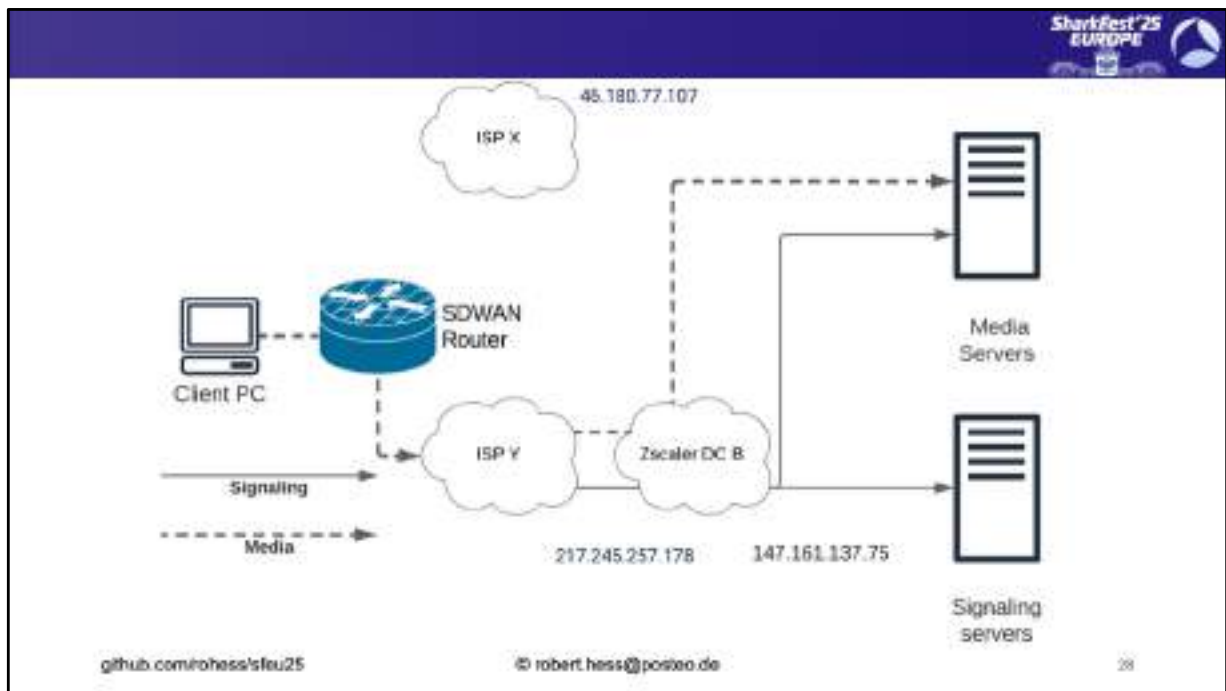
you would expect connections established over one of the links to stay on the link

for a new connection (TCP/UDP) it can change, even for the same target

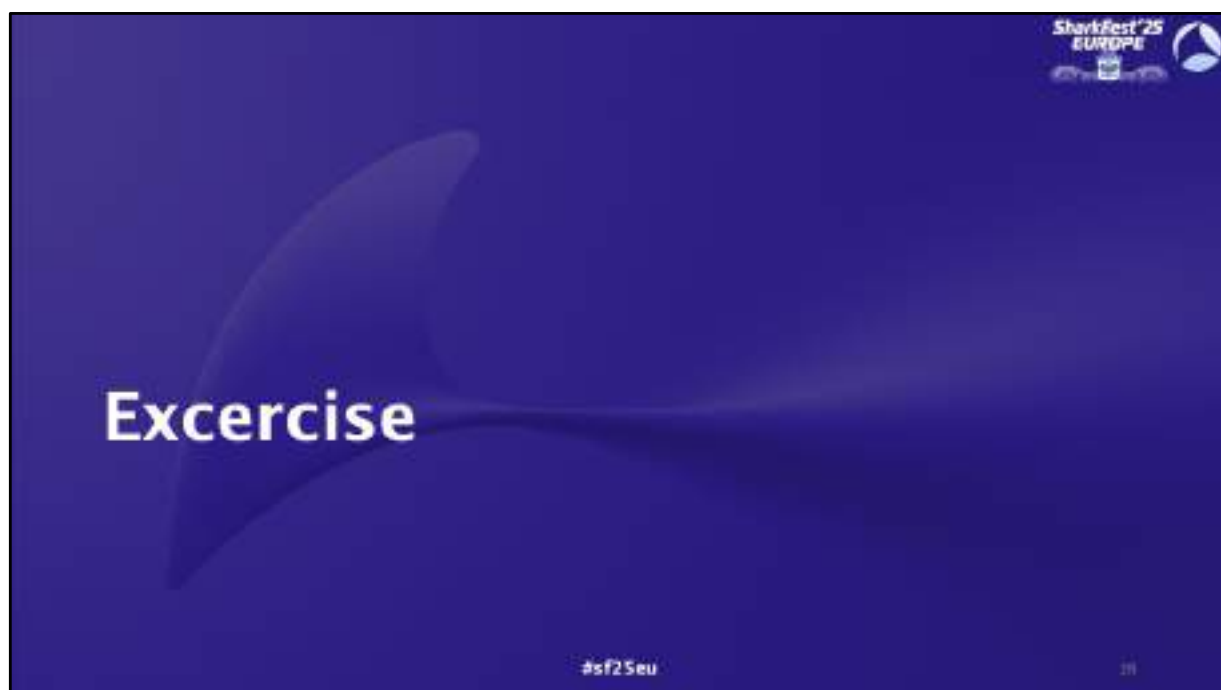
Zscaler: Switching the Zscaler Tunnel







what happens to our media connection when this happens?



Reconnect - aka time without RTP media streams - UDP
(single experiment - just ballpark numbers)



- Teams - 22s - full setup of peer connection including redirect
- Meet - 5s - no ICE negotiation, existing peer connection just finds a new prflx candidate
- Zoom - 14s - setup of peer connection for data channels
- Webex - 2.4 sec - same as Meet- looks like both Zscaler tunnels are overlapping ~600ms
- GoToWebinar - 2.2 sec again only new prflx cand. Different setup

Conclusions

What can the application do – and what can the admin do

What happens with HTTPS/TCP ?



- Zscaler acts as a https proxy. Client connection gets never lost.
- Gap when no packets arrive - Zscaler proxy reconnects to the server somewhat seamlessly.
- Server needs to recognize that this is a reconnect instead of a new connection.
- As this gap can be rather large (90 secs in one case) it's more about any higher-level timeouts on the server side which may create havoc

What can applications do?

It's all about not adding extra delay to these reconnects

- Make sure that the peer connection switches over seamlessly without a full disconnect/reconnect
- If app needs to do full ICE setup, make it fast
- Have long enough timeouts on TCP connections and/or reconnect TCP signaling without tearing down UDP media
- Don't Panic – if there is a gap of a few seconds – just wait until you do drastic things – aka longer timeouts – which comes with its own problems

Not sure we need to restart ICE in such a situation

But we can figure this out with analysing what meet does based on webRTC internals

- Bypass web conferencing traffic from any tunneling (Split tunnel)
 - > Not only media, but also signaling
- Check how often your tunnels reconnect, because that breaks also other things (e.g. RDP)
- Modify PAC file and dynamic checks to minimize these reconnects
- Make sure all SDWAN endpoints default to the same Zscaler DC -> less tunnel switches

What else is there

- TCP connections for Media where UDP is not allowed
- Divers split tunnel setups
- Sometimes also DIRECT traffic seems to be affected from tunnel restart -> Bugs in the Zscaler agent ?
- Browser config to force only public IPs
- Max # of proxy connections from browser – can this become a problem?
- IPv6 end to end ??

I see rare disconnects also for Split tunnel setups - why?

Chrome claims to have max 32 connection to the proxy – is this per Tab or per Browser?? Only relevant for long poll.

Questions & Feedback, please

*Puzzle: which packet contains
the SDP offer from Client to
server with an ICE candidate
in it?*

