RTCP, Racecars, video and 5g

Or: how to make WebRTC work at 200kph

Tim Panton - @steely_glint@chaos.social - tim@pi.pe - @steely_glint:matrix.org

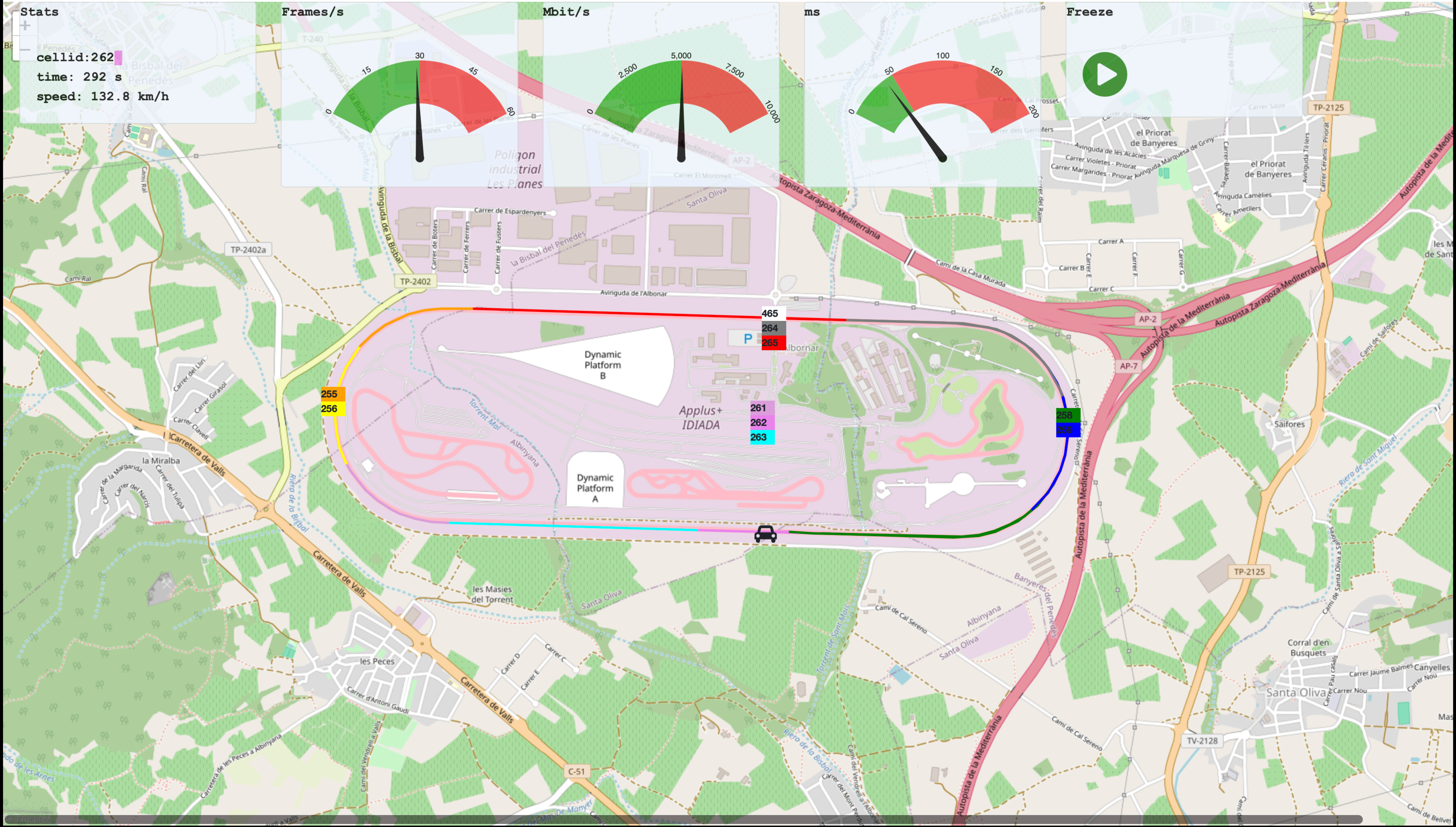
Tim Panton
CTO
pi.pe GmBH

Wrote a WebRTC stack for small cameras (baby monitors etc) Using OSS modules.

Race Car Camera

- Over driver’s shoulder
- Audio and Video to the pit crew
- High quality
- Low latency
- Long range
- High speeds
- Public networks (5G)

camera type: nopipe speed: 130km/h Draw



Details

For those who like numbers

- Funded by the EU
- Thanks to TargetX
- And Idiada

Km/h	Min	freeze %	comp	hops	cells	Rtt ms	Mbit/s
nopipe							
30	5	18.28	635	4	4	37	5.0
50	5	23.37	600	5	5	41	5.0
80	5	26.02	609	9	9	42	5.0
100	5	31.62	572	10	10	39	5.0
130	5	33.60	573	14	10	43	5.0
160	6	32.42	596	18	10	43	5.0
all	32	29.67	578	55	15	41	5.0
|pipe|							
30	4	0.00	1023	4	4	36	1.0
50	5	0.09	916	7	6	40	2.9
80	5	0.14	837	8	8	42	3.9
100	5	2.91	893	9	9	39	3.7
130	5	0.52	916	13	9	41	4.0
160	5	0.63	938	16	10	40	3.6
all	30	0.73	917	53	16	40	3.2

So what is the difference between them?

What is the magic?

- Same hardware
- Same network
- Both using RTP over 5G
- Same codec
- Same target bitrate
- Same endpoint
- Same track

RTCP

RTCP

RTP’s feedback channel

- Stats
  - Sender reports (SR)
  - Receiver reports (RR)
- Control
  - BYE
- Feedback
  - Picture loss (PLI)
  - Packet loss (NACK)
  - Bandwidth estimates (REMB or TWCC)

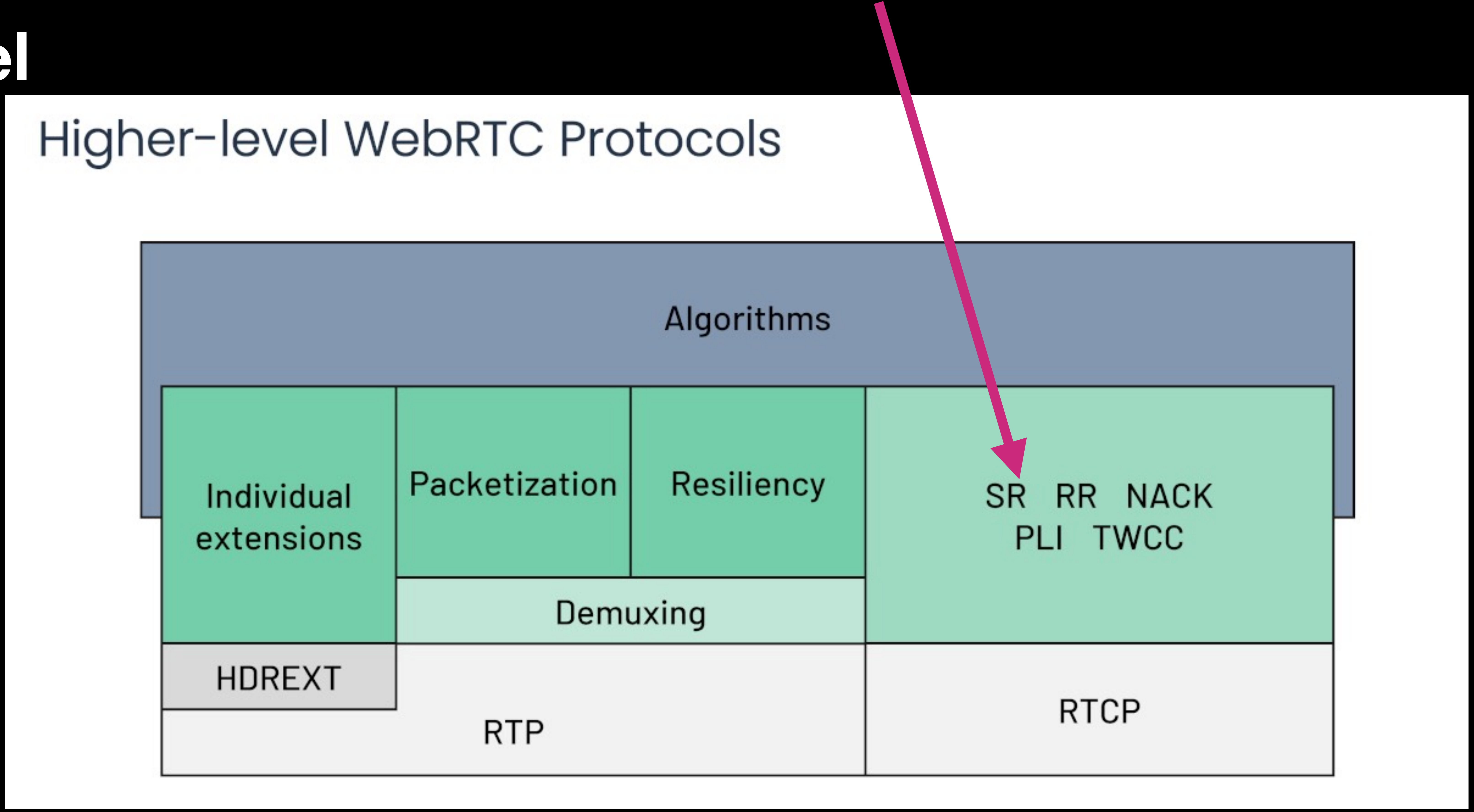


Image credit: webrtccourse.com

The bad news

RTCP is ugly

- A mess of optional extensions
- Used to run on it's own port
- Muxed on same port as RTP mostly
- Every extension has it's own incompatible binary format.
- Come back ASN.1
  all is forgiven…

```
public class REMB {
    public static int FMT = 15;
    /*
    0                   1                   2                   3
    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |V=2|P| FMT=15  |   PT=206      |             length            |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |                  SSRC of packet sender                        |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |                  SSRC of media source                         |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |  Unique identifier 'R' 'E' 'M' 'B'                            |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |  Num SSRC     | BR Exp    |  BR Mantissa                      |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |   SSRC feedback                                               |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |  ...                                                          |
     */
    public static long getBwe(byte[] fci) {
        long bwe = 0;
        int remb = 0x52454d42;
        if (fci.length >= 12) {
            ByteBuffer bb = ByteBuffer.wrap(array: fci);
            int sig = bb.getInt();
            int val = bb.getInt();
            int ssrc = bb.getInt();
            if (sig == remb) {
                Log.verb(string: "got remb data");
                int mant = val & 0x3ffff;
                int exp = (val & 0xfc0000) >> 18;
                int ssrcn = (val & 0xf0000000) >> 24;
                bwe = mant << exp;
                Log.verb("bwe =" + bwe + " mant =" + mant + " exp=" + exp + " srcn=" + ssrcn+ " ssrc = "+ssrc);
            } else {
                Log.warn(string: "not remb");
            }
        }
        return bwe;
    }
```

Worse News

SDP

- What you get is negotiated in the offer-answer
- More arcane nonsense

```
a=rtcp-mux
a=rtpmap:96 H264/90000
a=rtcp-fb:96 goog-remb
a=rtcp-fb:96 nack
a=rtcp-fb:96 nack pli
```

REMB

Bandwidth estimation (use TWCC if you have multiple streams)

- Uses packet arrival interval to infer imminent congestion
- We can reduce bitrate before congestion gets too bad.
- This minimises latency and reduces risk of freezes caused by packet loss.

NACK

Negative ack - packet loss

- Can resend lost packets
- If RTT ~ frame interval this is invisible
- Otherwise Jitter Buffer has to grow, increasing latency
- We limit growth by holding a small cache of old packets for resending

NACK PLI

Worst case scenario

- We lost a packet
- We couldn’t resend it
- The packet contains info that the decoder requires to continue decoding
- So we have to send a Full frame and start afresh
- Key frames are expensive (10x or more a normal frame)
- Anything we sent before a keyframe is useless (so we flush the queue and cache)

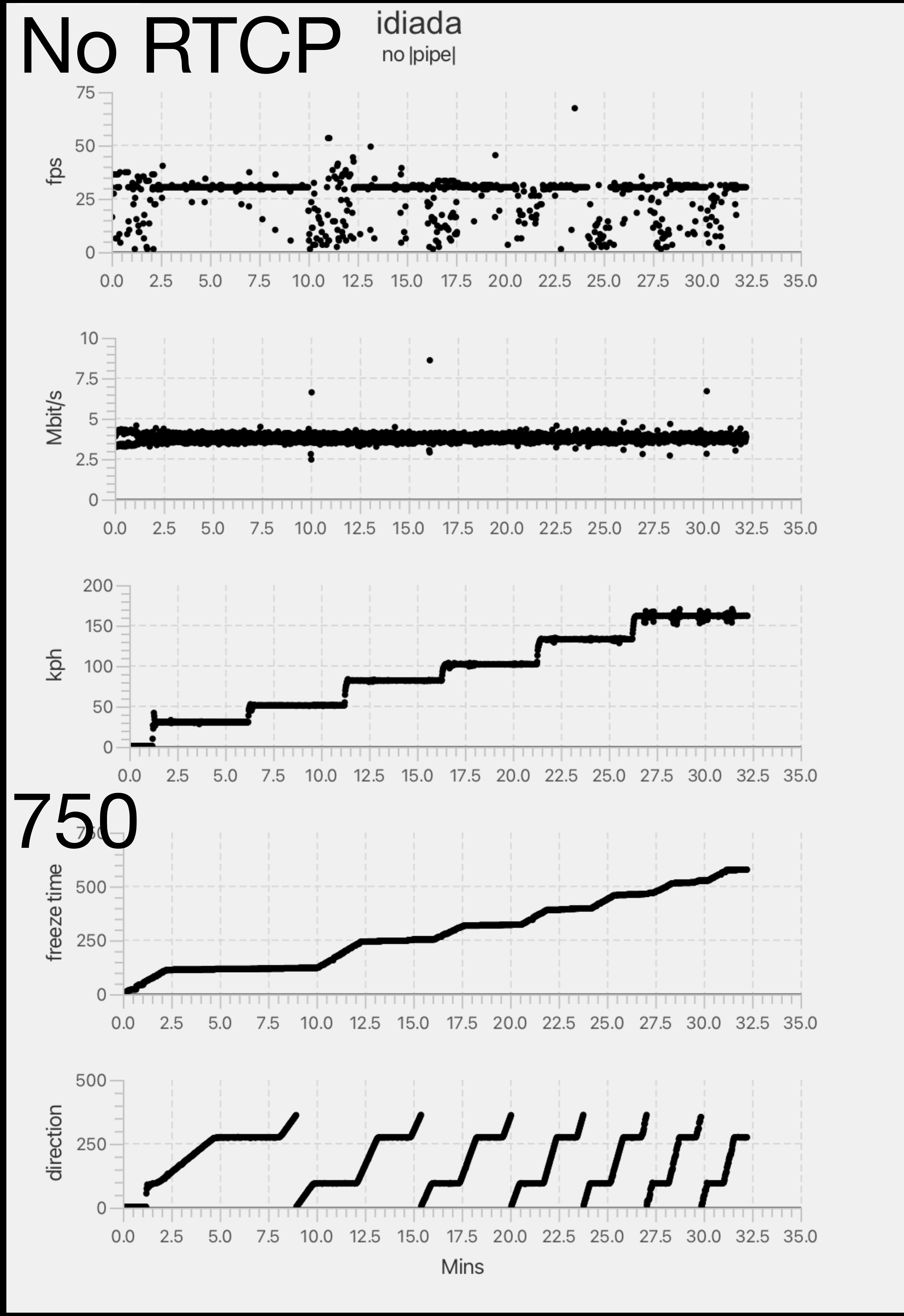
WIP

RFC 8888

- RTCP feedback for L4S
- Network can inform endpoints of congestion using ECN bits
- Hopefully results in better bandwidth estimation

Summary

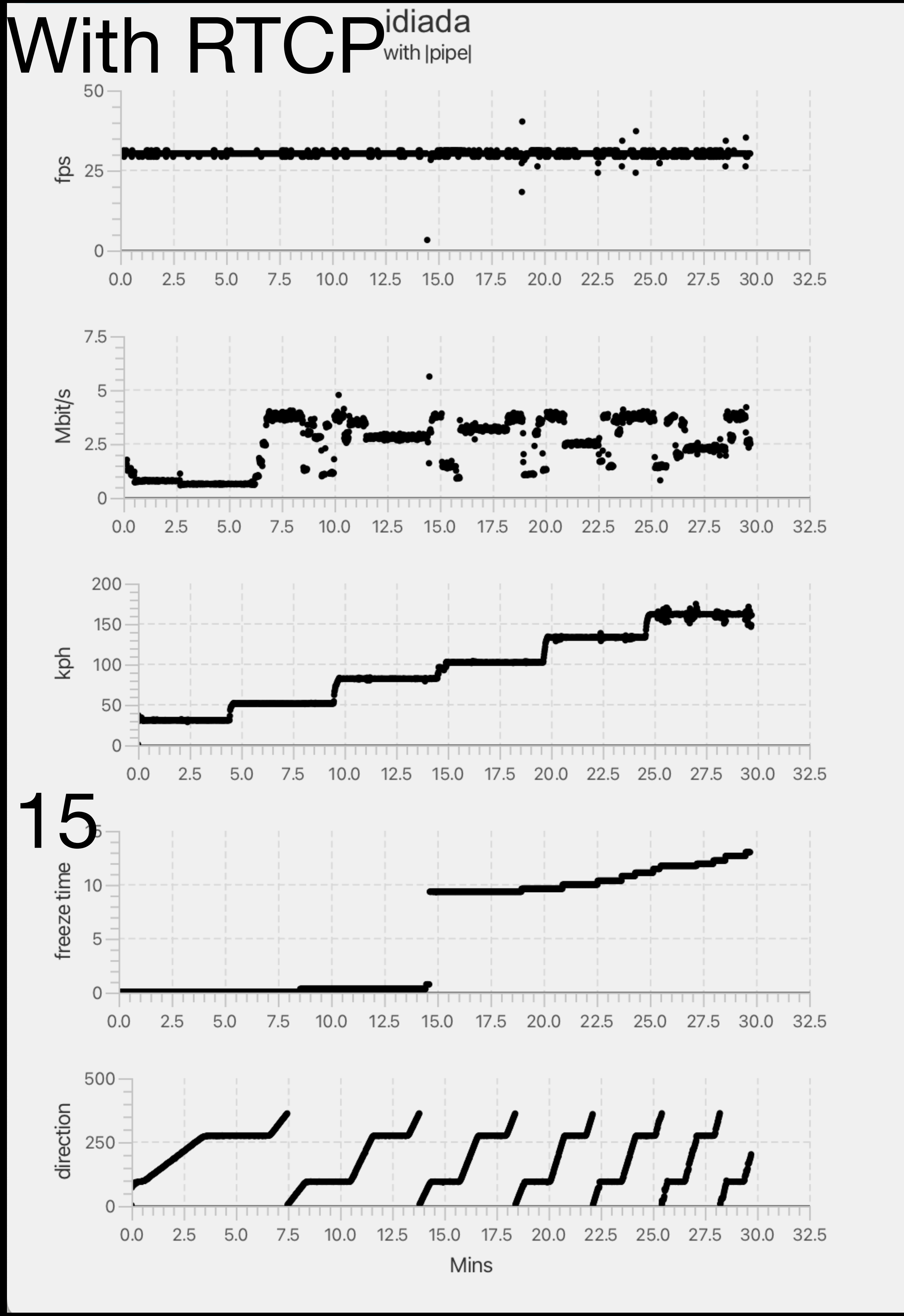
RTCP lets us cope with fluctuating network conditions.



->Stable frame rate->

->Variable bitrate->

->Reduced freeze time->



Open source
Srtplight

- https://github.com/steely-glint/srtplight
- RTP in Java written for Voxeo back in the day
- With RTCP
- Sample app https://github.com/pipe/whipi
- Ask:
    - PRs for RTP extensions , TWCC etc
    - Issues etc…

Thanks! Questions in the matrix room

steely_glint@matrix.org

- Or Contact:
  - tim@pi.pe
  - @steely_glint@chaos.social
- Consulting on open source WebRTC protocols
  - SRTP : https://github.com/steely-glint/srtplight
  - ICE : https://github.com/steely-glint/slice
  - SCTP : https://github.com/pipe/sctp4j
  - WHIP : https://github.com/pipe/whipi
  - Pion/gstreamer etc
- Building things with |pipe|