ECN bleaching detection with Pietrasanta traceroute

RIPE 88

Luca Sani

Senior R&D Software Engineer @Catchpoint

catchpoint

Previously @ RIPE87 MAT… Pietrasanta Traceroute

- Based on Dmitry Butskoy *Linux traceroute*
- Several enhancements
  - Speedup
  - QUIC traceroute
  - ECN bleaching detection
  - Work in Azure environment
  - TCP “In Session”
  - … and many more

Pietrasanta – “A noble town since 1841 and a city of art” (and where our Italian office is located!)

https://github.com/catchpoint/Networking.traceroute/

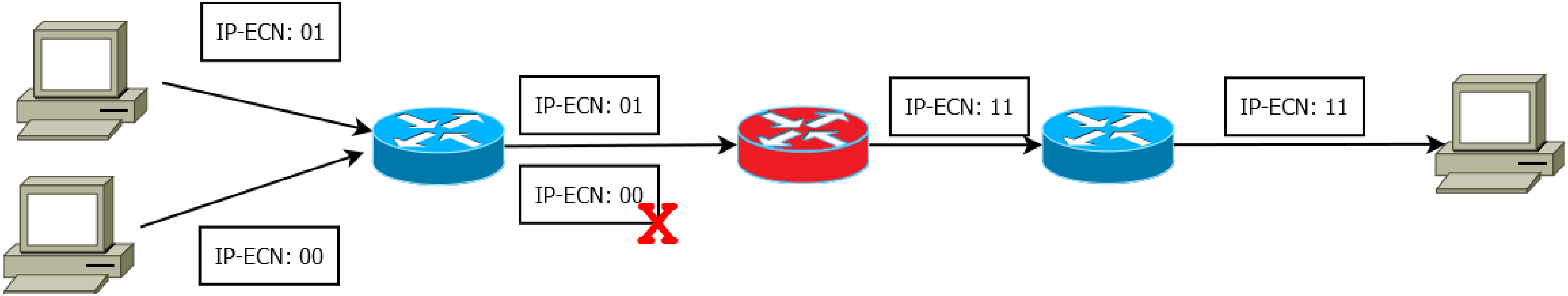
©2024 Catchpoint Systems

ECN bleaching detection

©2024 Catchpoint Systems

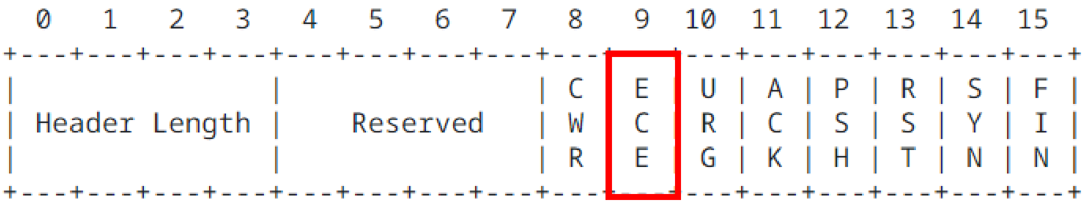
ECN mechanism

- *The Addition of Explicit Congestion Notification to IP*, rfc3168, 2001
    - Two bits in the IP header
- The source declares that a packet should be treated with ECN by setting the IP-ECN fields either to 01 or 10
- When congestion happens, instead of dropping the packet the router sets the IP-ECN fields to 11 (CE - Congestion Experienced)



ECN feedback

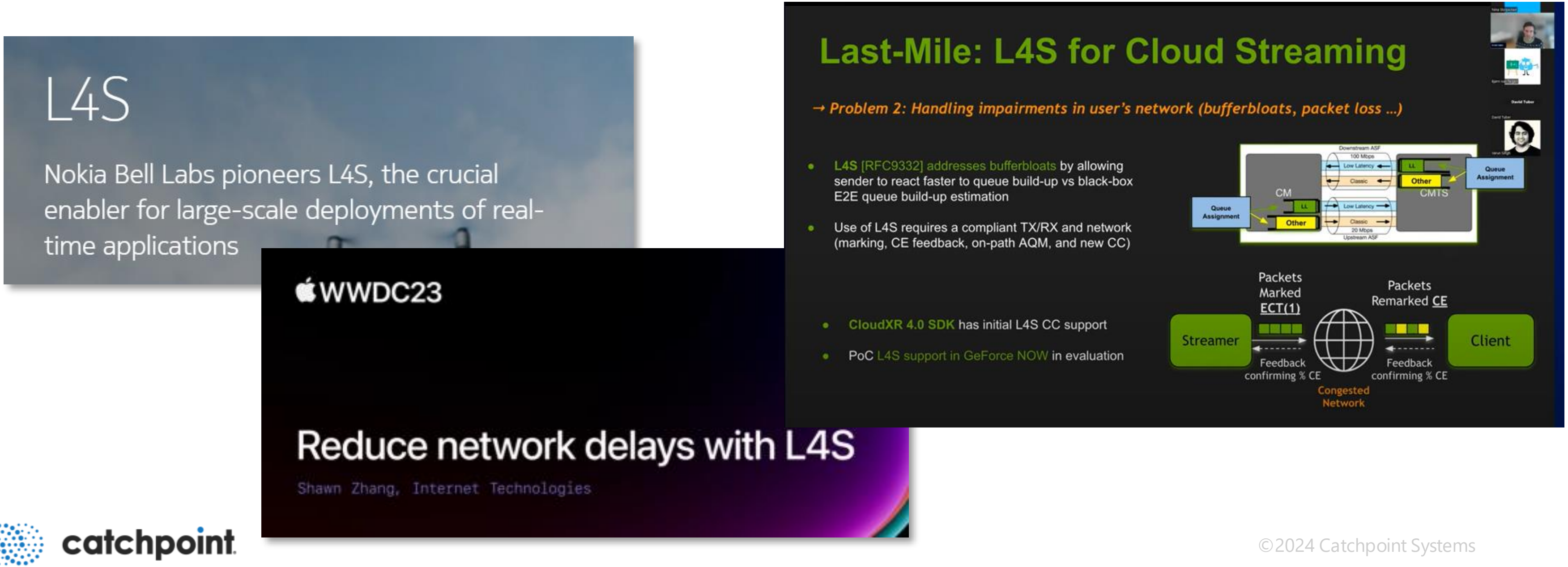
- A destination that receives a packet with IP-ECN = CE should report to the source this event
- The source should then adjust the rate
- The report is done at transport/application layer
    - Example: in TCP, this event can be reported using a dedicated TCP flag (ECE – ECN-Echo)



```
  0   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|               |               | C | E | U | A | P | R | S | F |
| Header Length |    Reserved   | W | C | R | C | S | S | Y | I |
|               |               | R | E | G | K | H | T | N | N |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
```

ECN and L4S

- Recently, ECN mechanism got renewed attention due to L4S (Low Latency, Low Loss, and Scalable Throughput – rfc9330, 2023)
- L4S requires an ECN feedback more accurate wrt the "classic" 2001 version



L4S

Nokia Bell Labs pioneers L4S, the crucial enabler for large-scale deployments of real-time applications

WWDC23

Reduce network delays with L4S

Shawn Zhang, Internet Technologies

Last-Mile: L4S for Cloud Streaming

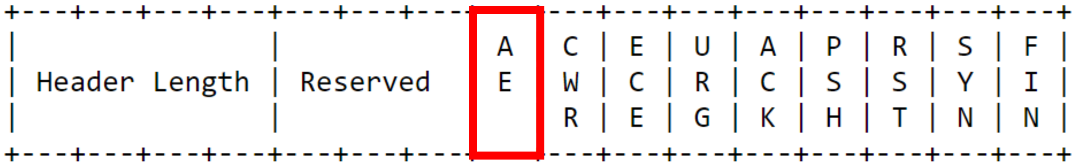
→ *Problem 2: Handling impairments in user's network (bufferbloats, packet loss …)*

- L4S [RFC9332] addresses bufferbloats by allowing sender to react faster to queue build-up vs black-box E2E queue build-up estimation
- Use of L4S requires a compliant TX/RX and network (marking, CE feedback, on-path AQM, and new CC)
- CloudXR 4.0 SDK has initial L4S CC support
- PoC L4S support in GeForce NOW in evaluation

catchpoint

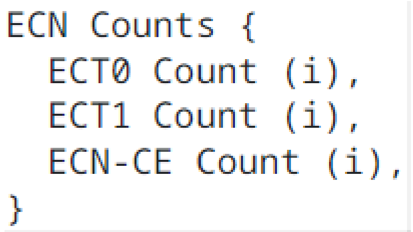
More accurate ECN feedback

- TCP: *More Accurate Explicit Congestion Notification (AccECN) Feedback in TCP* (still a draft)



```
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|               |           | A | C | E | U | A | P | R | S | F |
| Header Length | Reserved  | E | W | C | R | C | S | S | Y | I |
|               |           |   | R | E | G | K | H | T | N | N |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
```

- QUIC: Supported natively via ECN counters in the ACK frame (rfc9000)



```
ECN Counts {
  ECT0 Count (i),
  ECT1 Count (i),
  ECN-CE Count (i),
}
```

ECN bleaching detection

- Intermediate hops can bleach/alter the value of ECN into the IP header (see for example: *The Benefits of Using Explicit Congestion Notification (ECN)* – rfc8087, 2017)

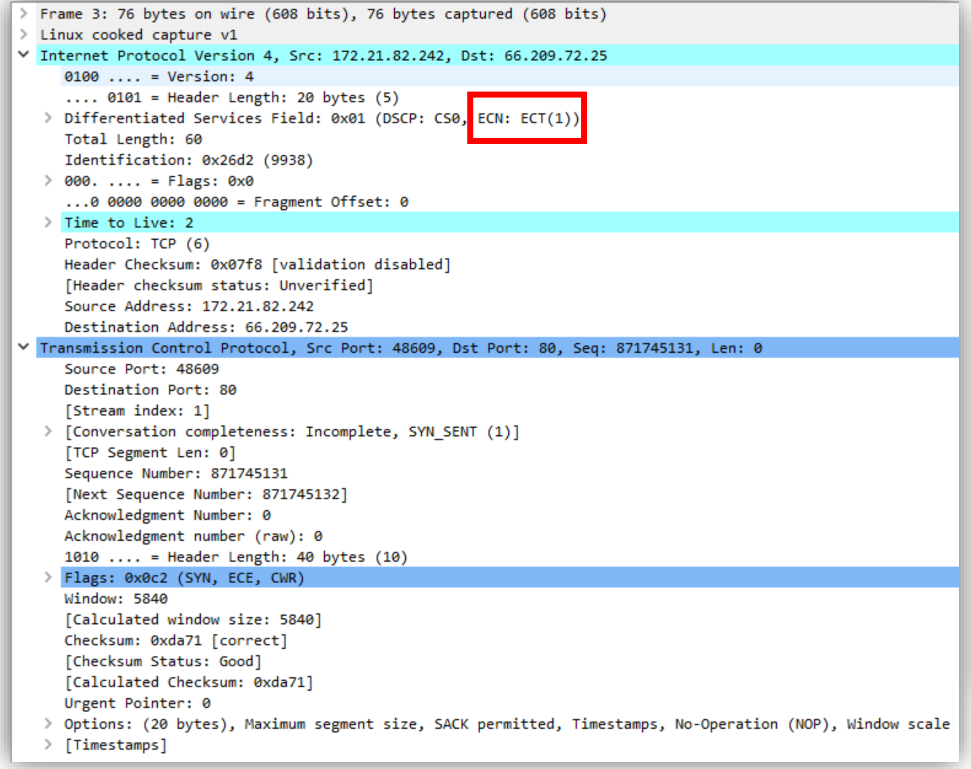
- With Pietrasanta traceroute we can send probes with IP-ECN values different from zero and check hop by hop what was the IP-ECN value of the probe ***when it expired***
  - Detect bleaching, but also congestion and any kind of alteration

- We can also check whether the destination transport layer (either TCP or QUIC) supports more accurate ECN feedbacks, because:
  - TCP stack need to be patched
  - Not all QUIC implementations report ECN counters

©2024 Catchpoint Systems

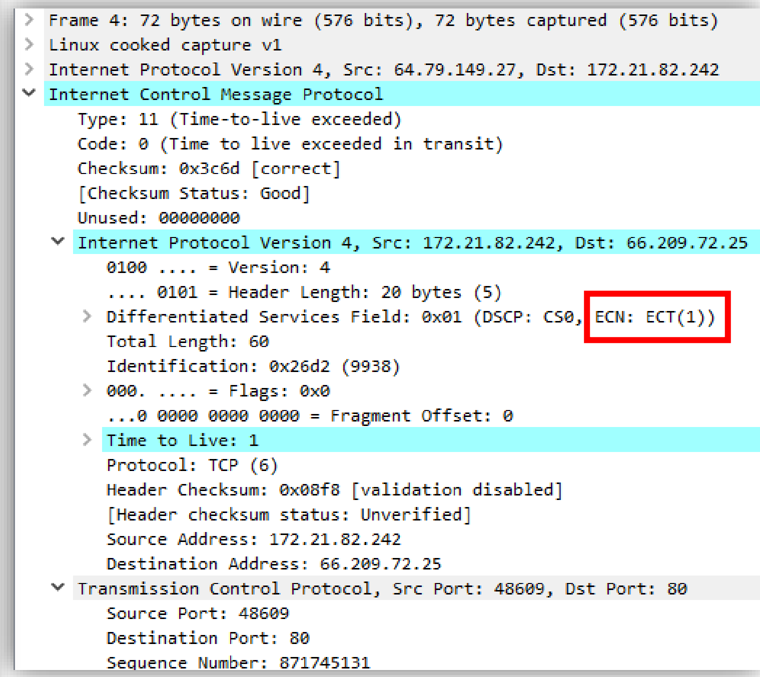
Report ECN hop by hop

Probe sent



```
> Frame 3: 76 bytes on wire (608 bits), 76 bytes captured (608 bits)
> Linux cooked capture v1
v Internet Protocol Version 4, Src: 172.21.82.242, Dst: 66.209.72.25
    0100 .... = Version: 4
    .... 0101 = Header Length: 20 bytes (5)
  > Differentiated Services Field: 0x01 (DSCP: CS0, ECN: ECT(1))
    Total Length: 60
    Identification: 0x26d2 (9938)
  > 000. .... = Flags: 0x0
    ...0 0000 0000 0000 = Fragment Offset: 0
  > Time to Live: 2
    Protocol: TCP (6)
    Header Checksum: 0x07f8 [validation disabled]
    [Header checksum status: Unverified]
    Source Address: 172.21.82.242
    Destination Address: 66.209.72.25
v Transmission Control Protocol, Src Port: 48609, Dst Port: 80, Seq: 871745131, Len: 0
    Source Port: 48609
    Destination Port: 80
    [Stream index: 1]
  > [Conversation completeness: Incomplete, SYN_SENT (1)]
    [TCP Segment Len: 0]
    Sequence Number: 871745131
    [Next Sequence Number: 871745132]
    Acknowledgment Number: 0
    Acknowledgment number (raw): 0
    1010 .... = Header Length: 40 bytes (10)
  > Flags: 0x0c2 (SYN, ECE, CWR)
    Window: 5840
    [Calculated window size: 5840]
    Checksum: 0xda71 [correct]
    [Checksum Status: Good]
    [Calculated Checksum: 0xda71]
    Urgent Pointer: 0
  > Options: (20 bytes), Maximum segment size, SACK permitted, Timestamps, No-Operation (NOP), Window scale
  > [Timestamps]
```

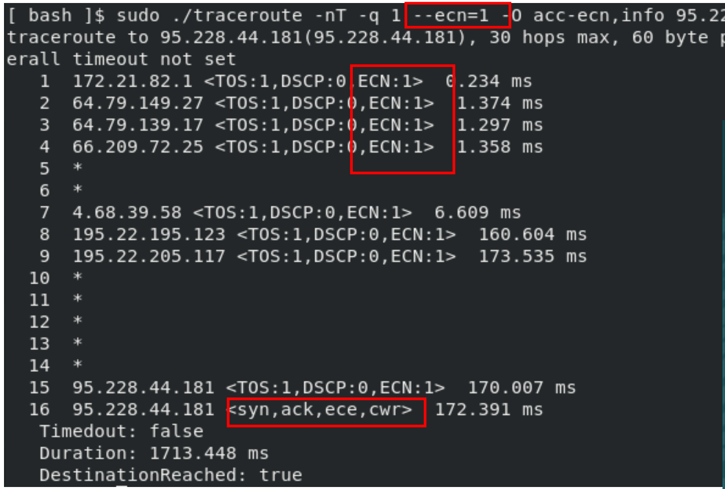
ICMP TTL Exceeded



```
> Frame 4: 72 bytes on wire (576 bits), 72 bytes captured (576 bits)
> Linux cooked capture v1
> Internet Protocol Version 4, Src: 64.79.149.27, Dst: 172.21.82.242
v Internet Control Message Protocol
    Type: 11 (Time-to-live exceeded)
    Code: 0 (Time to live exceeded in transit)
    Checksum: 0x3c6d [correct]
    [Checksum Status: Good]
    Unused: 00000000
  v Internet Protocol Version 4, Src: 172.21.82.242, Dst: 66.209.72.25
      0100 .... = Version: 4
      .... 0101 = Header Length: 20 bytes (5)
    > Differentiated Services Field: 0x01 (DSCP: CS0, ECN: ECT(1))
      Total Length: 60
      Identification: 0x26d2 (9938)
    > 000. .... = Flags: 0x0
      ...0 0000 0000 0000 = Fragment Offset: 0
    > Time to Live: 1
      Protocol: TCP (6)
      Header Checksum: 0x08f8 [validation disabled]
      [Header checksum status: Unverified]
      Source Address: 172.21.82.242
      Destination Address: 66.209.72.25
  v Transmission Control Protocol, Src Port: 48609, Dst Port: 80
      Source Port: 48609
      Destination Port: 80
      Sequence Number: 871745131
```

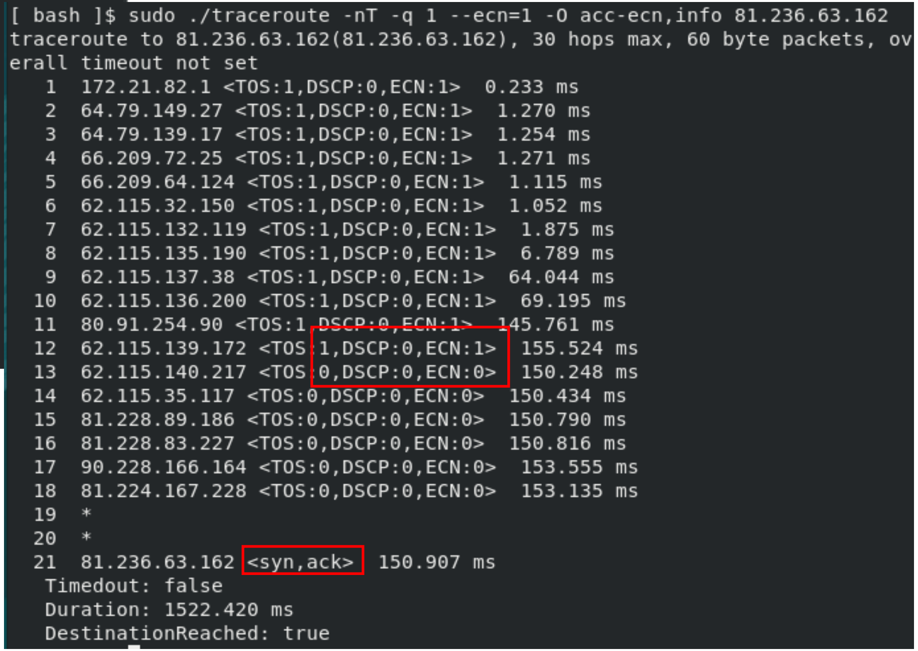
©2024 Catchpoint Systems

ECN detection: Some examples



```
[ bash ]$ sudo ./traceroute -nT -q 1 --ecn=1 -O acc-ecn,info 95.228.44.181
traceroute to 95.228.44.181(95.228.44.181), 30 hops max, 60 byte packets, ov
erall timeout not set
   1  172.21.82.1 <TOS:1,DSCP:0,ECN:1>  0.234 ms
   2  64.79.149.27 <TOS:1,DSCP:0,ECN:1>  1.374 ms
   3  64.79.139.17 <TOS:1,DSCP:0,ECN:1>  1.297 ms
   4  66.209.72.25 <TOS:1,DSCP:0,ECN:1>  1.358 ms
   5  *
   6  *
   7  4.68.39.58 <TOS:1,DSCP:0,ECN:1>  6.609 ms
   8  195.22.195.123 <TOS:1,DSCP:0,ECN:1>  160.604 ms
   9  195.22.205.117 <TOS:1,DSCP:0,ECN:1>  173.535 ms
  10  *
  11  *
  12  *
  13  *
  14  *
  15  95.228.44.181 <TOS:1,DSCP:0,ECN:1>  170.007 ms
  16  95.228.44.181 <syn,ack,ece,cwr>  172.391 ms
   Timedout: false
   Duration: 1713.448 ms
   DestinationReached: true
```

No bleaching, destination supports AccECN over TCP



```
[ bash ]$ sudo ./traceroute -nT -q 1 --ecn=1 -O acc-ecn,info 81.236.63.162
traceroute to 81.236.63.162(81.236.63.162), 30 hops max, 60 byte packets, ov
erall timeout not set
   1  172.21.82.1 <TOS:1,DSCP:0,ECN:1>  0.233 ms
   2  64.79.149.27 <TOS:1,DSCP:0,ECN:1>  1.270 ms
   3  64.79.139.17 <TOS:1,DSCP:0,ECN:1>  1.254 ms
   4  66.209.72.25 <TOS:1,DSCP:0,ECN:1>  1.271 ms
   5  66.209.64.124 <TOS:1,DSCP:0,ECN:1>  1.115 ms
   6  62.115.32.150 <TOS:1,DSCP:0,ECN:1>  1.052 ms
   7  62.115.132.119 <TOS:1,DSCP:0,ECN:1>  1.875 ms
   8  62.115.135.190 <TOS:1,DSCP:0,ECN:1>  6.789 ms
   9  62.115.137.38 <TOS:1,DSCP:0,ECN:1>  64.044 ms
  10  62.115.136.200 <TOS:1,DSCP:0,ECN:1>  69.195 ms
  11  80.91.254.90 <TOS:1,DSCP:0,ECN:1>  145.761 ms
  12  62.115.139.172 <TOS:1,DSCP:0,ECN:1>  155.524 ms
  13  62.115.140.217 <TOS:0,DSCP:0,ECN:0>  150.248 ms
  14  62.115.35.117 <TOS:0,DSCP:0,ECN:0>  150.434 ms
  15  81.228.89.186 <TOS:0,DSCP:0,ECN:0>  150.790 ms
  16  81.228.83.227 <TOS:0,DSCP:0,ECN:0>  150.816 ms
  17  90.228.166.164 <TOS:0,DSCP:0,ECN:0>  153.555 ms
  18  81.224.167.228 <TOS:0,DSCP:0,ECN:0>  153.135 ms
  19  *
  20  *
  21  81.236.63.162 <syn,ack>  150.907 ms
   Timedout: false
   Duration: 1522.420 ms
   DestinationReached: true
```

Bleaching happened

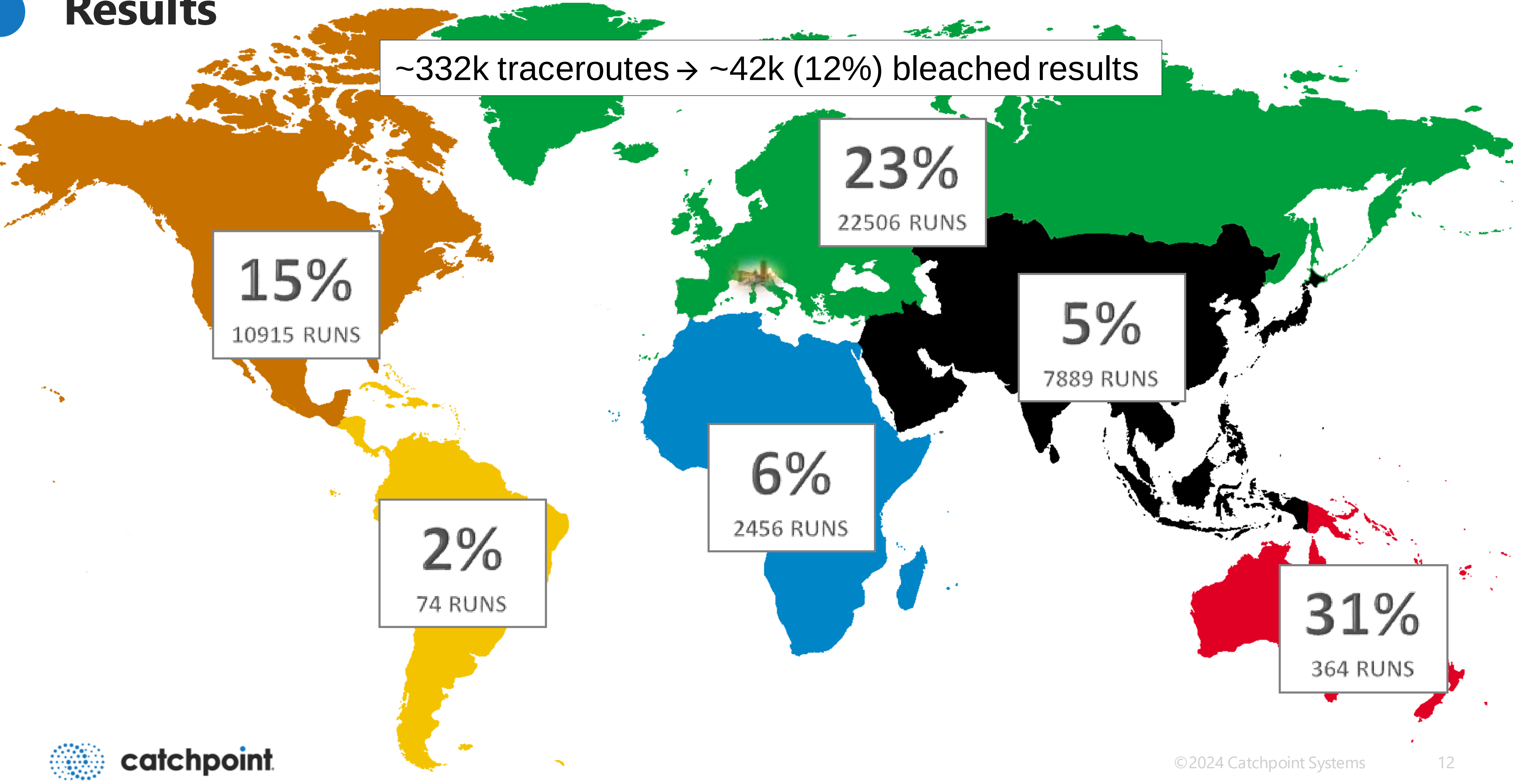
IP-ECN bleaching in the wild

- We run Pietrasanta traceroute from Catchpoint nodes deployed around the world to understand how many traceroutes show the effects of ECN bleaching

- Besides research curiosity, this can be useful to understand how much the network is prepared to accommodate L4S.
  - ECN is an essential requirement for L4S

- This is not intended to be a rigorous research work
  - The results presented are obviously biased by the node selection
  - We tried to be as fair and distributed as possible in selecting sources and destinations

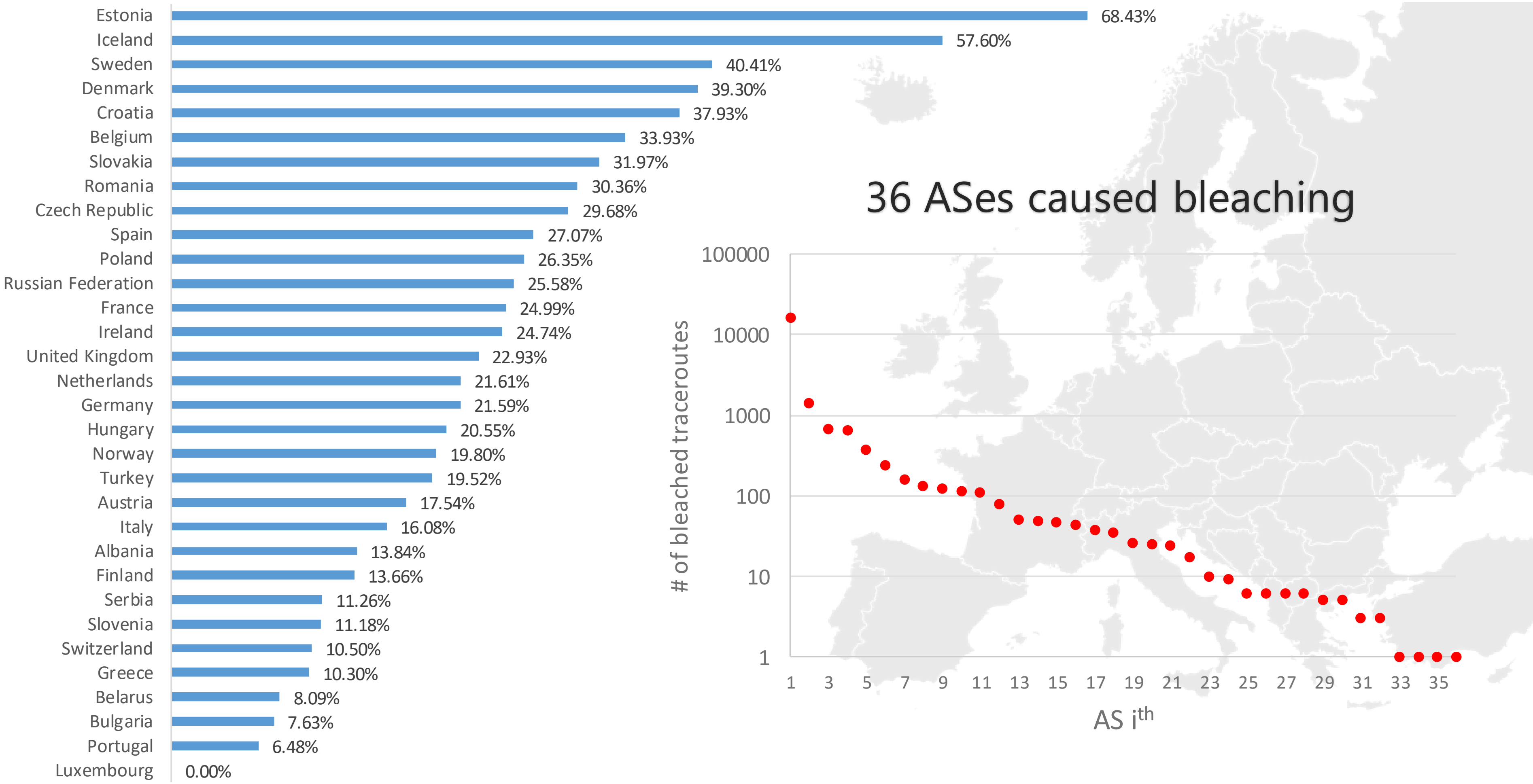
Results



~332k traceroutes → ~42k (12%) bleached results

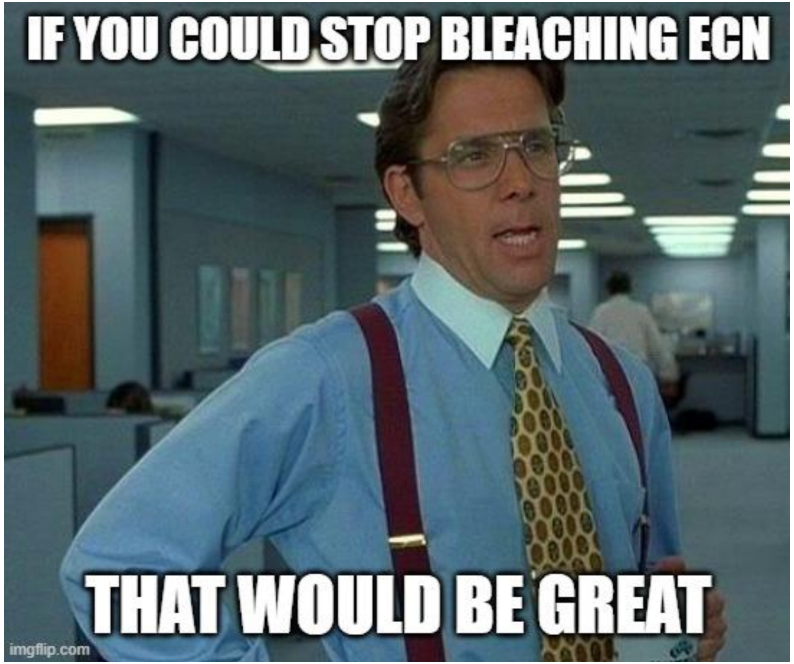
©2024 Catchpoint Systems

Inside-out results



36 ASes caused bleaching

Conclusions and future work



- ECN Bleaching is not a tale and still around
- Pietrasanta traceroute may help in identifying where the bleaching is happening
  - You cannot fix what you cannot see!
- It may be extremely interesting to see what RIPE Atlas could see!

©2024 Catchpoint Systems

Thank you!

- Feel free to check/use/ & contribute! https://github.com/catchpoint/Networking.traceroute/ (GPL!)
- And come by to meet us!
  - Pietrasanta is a nice town on Tuscany seaside …