Arquitectas de Internet

Eva M. Castro
eva.castro@urjc.es

Abril 2026

Universidad
Rey Juan Carlos

Radia Perlman

Doctora en Ciencias de la Computación, MIT (1988)

- **STP** — Spanning Tree Protocol, 1985 · IEEE 802.1D (1990)
- **TRILL** — Transparent Interconnection of Lots of Links · RFC 6325 (2011)
- **Autenticación para IS-IS**
- **SKIP** (precursor de IKE/IPsec)

SIGCOMM Award 2010 · USENIX Lifetime Achievement Award 2006

Retrocedamos a 1986, UC Berkeley...

En 1986...

Internet sufrió su **primer colapso de congestión**

> El throughput entre LBL y UC Berkeley cayó
> de **32 000 bps a 40 bps**

Un descenso de **800 veces en minutos**.
Van Jacobson lo descubrió porque era su propia conexión la que se caía.

Control de congestión · Van Jacobson (1988)

- **TCP Tahoe (1988)**
  - **slow start** — arrancar despacio, crecer exponencialmente
  - **congestion avoidance** — crecer en aditivo, retroceder en multiplicativo
  - **fast retransmit** — retransmitir sin esperar timeout
  - estimación dinámica de RTT

```
cwnd → pérdida detectada → cwnd = 1
```

- **TCP Reno (1990)**
  - fast recovery

```
cwnd → pérdida detectada → cwnd / 2
```

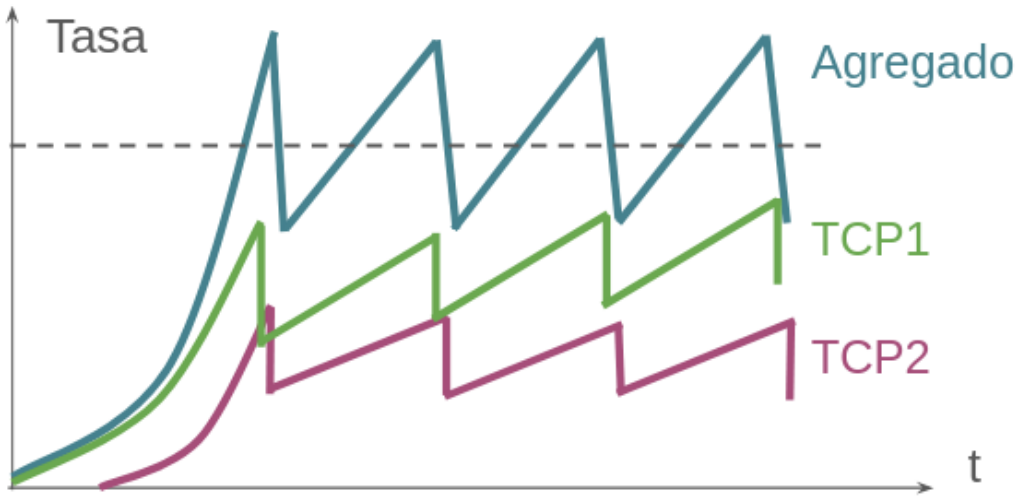
> **La señal de congestión es la pérdida de segmentos TCP**

Sincronización global (1993) · Sally Floyd

Todos los flujos TCP perciben la pérdida **al mismo tiempo**:

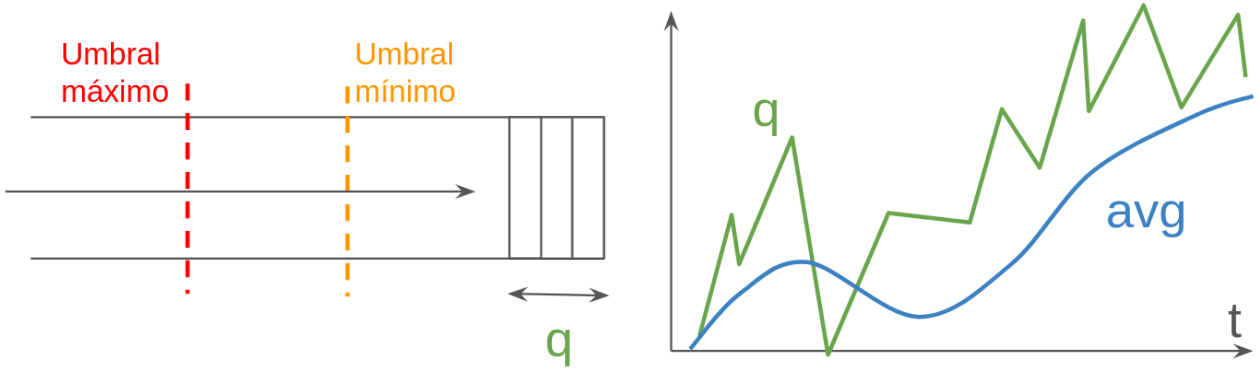
1. La cola se llena → todos pierden un paquete
2. Todos reducen su ventana a la vez → cola se vacía
3. Todos aceleran juntos → cola se llena de nuevo

El router descarta en **ráfagas**.
La ocupación oscila entre llena y vacía — sin que ningún flujo sea el culpable.



Random Early Detection (RED, 1993) · Sally Floyd

RED descarta paquetes **antes** de que la cola se llene:




```
avg = (1 - w) × avg + w × q
```

- `avg < min_th` → no se descarta
- `min_th ≤ avg ≤ max_th` → descarte probabilístico creciente
- `avg > max_th` → P = 1, descarte seguro

> RED es una solución en el **nivel de red** ⟶ AQM: Active Queue Management

Explicit Congestion Notification (ECN, 1994) · Sally Floyd

RFC 3168 (2001)

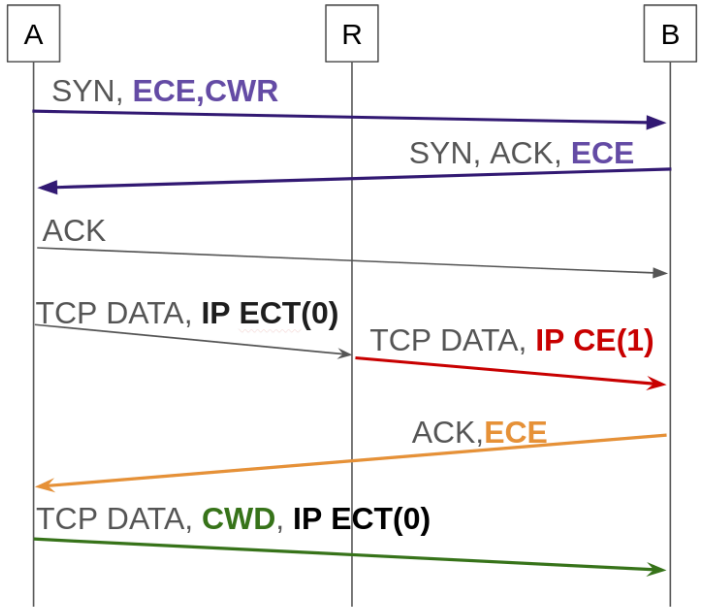
RED descarta. ECN va más lejos: **marcar sin descartar**.

No ECN	0	0
Extremos entienden ECN	0	1
Extremos entienden ECN	1	0
Hay congestión	1	1

Notificación ECN · Sally Floyd

```
 8                           15
┌─┬─┬─┬─┬─┬─┬─┬─┐
│ │ │ │ │ │ │ │ │
│C│E│U│A│P│R│S│F│
│W│C│R│C│S│S│Y│I│
│R│E│G│K│H│T│N│N│
└─┴─┴─┴─┴─┴─┴─┴─┘
```

ECE: ECN-Echo
CWR: Congestion Window Reduced



> ECN requiere cooperación: el router señaliza,
> los extremos reaccionan.

ECN en el cable

No.	Time	Source	Destination	Protocol	Length	Explicit Congestion Notification	Info
43	5.278000	1.1.23.3	1.1.12.1	TCP	60	Not ECN-Capable Transport	[TCP Window Update] 46557 → 80 [ACK] Seq=162 Ack=6924 Win=4128 Len=0
44	6.009000	1.1.12.1	1.1.23.3	TCP	590	ECN-Capable Transport codepoint '10'	80 → 46557 [ACK] Seq=6924 Ack=162 Win=3967 Len=536 [TCP segment of a reassembled PDU]
45	6.088000	1.1.23.3	1.1.12.1	TCP	60	Not ECN-Capable Transport	46557 → 80 [ACK] Seq=162 Ack=7460 Win=3592 Len=0
46	6.101000	1.1.23.3	1.1.12.1	TCP	60	Not ECN-Capable Transport	[TCP Window Update] 46557 → 80 [ACK] Seq=162 Ack=7460 Win=4128 Len=0
47	6.792000	1.1.12.1	1.1.23.3	TCP	590	ECN-Capable Transport codepoint '10'	80 → 46557 [ACK] Seq=7460 Ack=162 Win=3967 Len=536 [TCP segment of a reassembled PDU]
48	6.823000	1.1.12.1	1.1.23.3	TCP	590	Congestion Experienced	80 → 46557 [ACK, CWR] Seq=7996 Ack=162 Win=3967 Len=536 [TCP segment of a reassembled
49	6.870000	1.1.23.3	1.1.12.1	TCP	60	Not ECN-Capable Transport	46557 → 80 [ACK] Seq=162 Ack=7996 Win=3592 Len=0
50	6.871000	1.1.23.3	1.1.12.1	TCP	60	Not ECN-Capable Transport	46557 → 80 [ACK, ECE] Seq=162 Ack=8532 Win=3056 Len=0
51	6.895000	1.1.23.3	1.1.12.1	TCP	60	Not ECN-Capable Transport	[TCP Window Update] 46557 → 80 [ACK, ECE] Seq=162 Ack=8532 Win=3592 Len=0
52	6.896000	1.1.23.3	1.1.12.1	TCP	60	Not ECN-Capable Transport	[TCP Window Update] 46557 → 80 [ACK, ECE] Seq=162 Ack=8532 Win=4128 Len=0
53	7.477000	1.1.12.1	1.1.23.3	TCP	590	Congestion Experienced	80 → 46557 [ACK] Seq=8532 Ack=162 Win=3967 Len=536 [TCP segment of a reassembled PDU]
54	7.545000	1.1.23.3	1.1.12.1	TCP	60	Not ECN-Capable Transport	46557 → 80 [ACK, ECE] Seq=162 Ack=9068 Win=3592 Len=0
55	7.580000	1.1.23.3	1.1.12.1	TCP	60	Not ECN-Capable Transport	[TCP Window Update] 46557 → 80 [ACK, ECE] Seq=162 Ack=9068 Win=4128 Len=0
56	7.663000	1.1.12.1	1.1.23.3	TCP	590	Congestion Experienced	80 → 46557 [ACK] Seq=9068 Ack=162 Win=3967 Len=536 [TCP segment of a reassembled PDU]
57	7.705000	1.1.23.3	1.1.12.1	TCP	60	Not ECN-Capable Transport	46557 → 80 [ACK, ECE] Seq=162 Ack=9604 Win=3592 Len=0
58	7.721000	1.1.23.3	1.1.12.1	TCP	60	Not ECN-Capable Transport	[TCP Window Update] 46557 → 80 [ACK, ECE] Seq=162 Ack=9604 Win=4128 Len=0
59	8.294000	1.1.12.1	1.1.23.3	TCP	590	Congestion Experienced	80 → 46557 [ACK] Seq=9604 Ack=162 Win=3967 Len=536 [TCP segment of a reassembled PDU]
60	8.373000	1.1.23.3	1.1.12.1	TCP	60	Not ECN-Capable Transport	46557 → 80 [ACK, ECE] Seq=162 Ack=10140 Win=3592 Len=0
61	8.392000	1.1.12.1	1.1.23.3	TCP	590	ECN-Capable Transport codepoint '10'	80 → 46557 [ACK, CWR] Seq=10140 Ack=162 Win=3967 Len=536 [TCP segment of a reassembled
62	8.409000	1.1.23.3	1.1.12.1	TCP	60	Not ECN-Capable Transport	[TCP Window Update] 46557 → 80 [ACK, ECE] Seq=162 Ack=10140 Win=4128 Len=0
63	8.445000	1.1.23.3	1.1.12.1	TCP	60	Not ECN-Capable Transport	46557 → 80 [ACK] Seq=162 Ack=10676 Win=3592 Len=0
64	8.479000	1.1.23.3	1.1.12.1	TCP	60	Not ECN-Capable Transport	[TCP Window Update] 46557 → 80 [ACK] Seq=162 Ack=10676 Win=4128 Len=0
65	9.115000	1.1.12.1	1.1.23.3	TCP	590	ECN-Capable Transport codepoint '10'	80 → 46557 [ACK, CWR] Seq=10676 Ack=162 Win=3967 Len=536 [TCP segment of a reassembled
66	9.264000	1.1.23.3	1.1.12.1	TCP	60	Not ECN-Capable Transport	46557 → 80 [ACK] Seq=162 Ack=11212 Win=3592 Len=0
67	9.327000	1.1.23.3	1.1.12.1	TCP	60	Not ECN-Capable Transport	[TCP Window Update] 46557 → 80 [ACK] Seq=162 Ack=11212 Win=4128 Len=0
68	9.903000	1.1.12.1	1.1.23.3	TCP	590	ECN-Capable Transport codepoint '10'	80 → 46557 [ACK, CWR] Seq=11212 Ack=162 Win=3967 Len=536 [TCP segment of a reassembled

```
▸ Frame 48: 590 bytes on wire (4720 bits), 590 bytes captured (4720 bits)
▸ Ethernet II, Src: c0:01:14:7c:00:01 (c0:01:14:7c:00:01), Dst: c0:02:12:68:00:00 (c0:02:12:68:00:00)
▾ Internet Protocol Version 4, Src: 1.1.12.1, Dst: 1.1.23.3
    0100 .... = Version: 4
    .... 0101 = Header Length: 20 bytes (5)
  ▾ Differentiated Services Field: 0x03 (DSCP: CS0, ECN: CE)
      0000 00.. = Differentiated Services Codepoint: Default (0)
      .... ..11 = Explicit Congestion Notification: Congestion Experienced (3)
    Total Length: 576
    Identification: 0x01a0 (416)
  ▸ 000. .... = Flags: 0x0
    ...0 0000 0000 0000 = Fragment Offset: 0
    Time to Live: 254
    Protocol: TCP (6)
    Header Checksum: 0x940f [validation disabled]
```

Y pasan los años ...

los fabricantes de routers no implementan ECN

Nuevo paradigma · Lixia Zhang (1996)

La red y los extremos tienen que cooperar

Quién	Qué hace
Red	Fair Queuing: una cola por flujo
Red	Gestión de cola: descarte temprano (RED)
Red - Extremo	Realimentación: drops (implícita) o ECN (explícita)
Extremo	Libre para elegir algoritmo de congestión

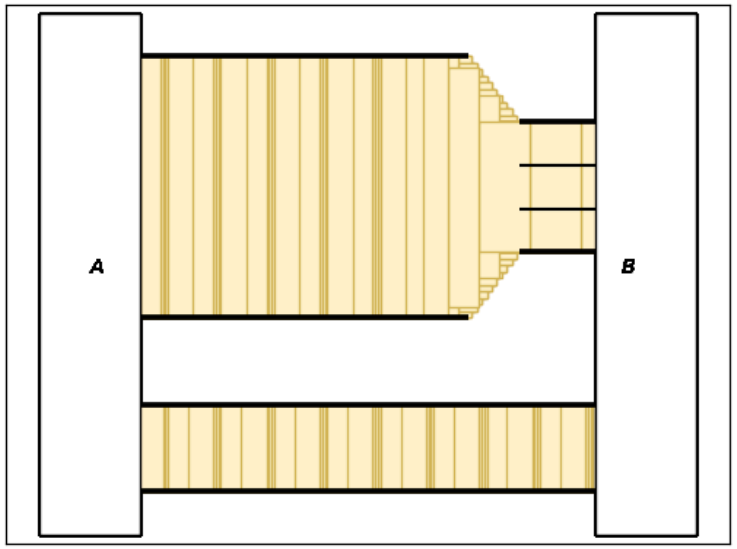
Y pasan los años ...

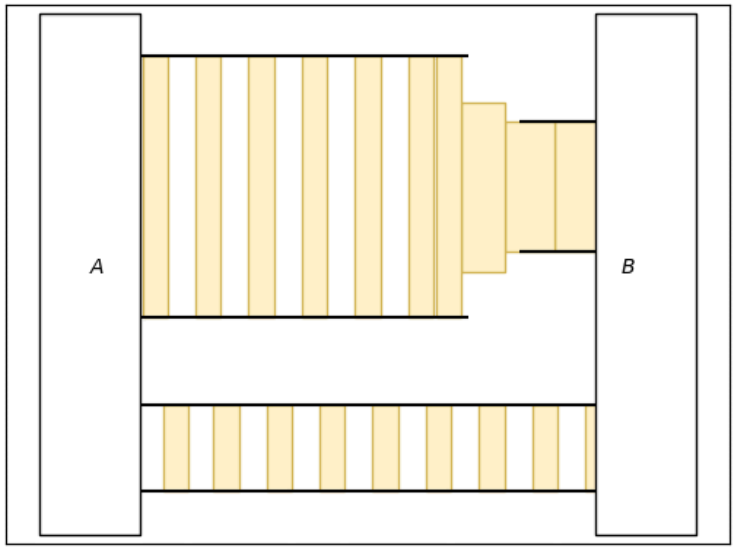
más algoritmos de congestión, más memoria, buffers más grandes

Bufferbloat · Jim Gettys (2011)

Cola persistente — Los datagramas permanecen en cola dando lugar a RTTs enormes que TCP no interpreta como problema mientras no haya pérdida.

> La pérdida tarda demasiado en llegar. Se necesita una señal mejor.





Fuente APNIC, Bjørn Teigean

Controlled Delay (CoDel, 2012) · Kathleen Nichols

RFC 8289

Usa el **sojourn time** (tiempo en cola) para predecir congestión — no el tamaño del buffer.

```
Para cada paquete que sale:
  si t_actual - t_entrada < 5 ms:
      se envía
  si llevamos > 100 ms con sojourn alto:
      marca/descarta
      próxima actuación en 100ms/√n
```

```
29.3ms     13ms    7.7ms   5.3ms
  |          |       |       |
[marca]  [marca] [marca] [marca]
```

- No necesita calibrar thresholds por enlace
- Funciona igual en 1 Mbps que en 10 Gbps

Soporte ECN: evolución en el tiempo

Año	Evento	Autora/es
1988	TCP Tahoe	Van Jacobson
1993	**RED**	**Floyd** & Jacobson
1994	**ECN** (propuesta)	**Floyd**
1996	**Paradigma en la arquitectura** (conceptual)	**Zhang**
1999	ARED / Gentle RED	**Floyd** et al.
2001	**ECN** RFC 3168	**Floyd**
2012	**CoDel**	**Nichols** & Jacobson

iOS (iOS 9, 2015): activa ECN · (iOS 17, 2023): marca baja latencia con ECT(1), resto con ECT(0).
macOS (10.11 El Capitan, 2015): activa ECN · (macOS 14 Sonoma, 2023): ECT(1) para baja latencia.
Linux (kernel 2.4, 2001): ECN configurable (`tcp_ecn=1`) · kernel 2.4.20 (2002): pasivo por defecto (`tcp_ecn=2`).
Windows (Vista/Server 2008): soporte ECN, desactivado por defecto · activo vía netsh.
BSD (FreeBSD 8+, 2009 · NetBSD, OpenBSD 2014): ECN configurable.
Solaris (Solaris 11): ECN configurable.

L4S · Low Latency, Low Loss, Scalable Throughput · 2023

Arquitectura · RFC 9330–9332

Reutiliza ECT(1) para identificar flujos L4S y separar el tráfico en **dos colas** (DualPI2):

Cola	Flujos	AQM
Clásica	ECT(0) / sin ECN	CoDel (retardo 100ms) / RED
L4S	ECT(1)	señal continua y proporcional sobre congestión (retardos 1ms)

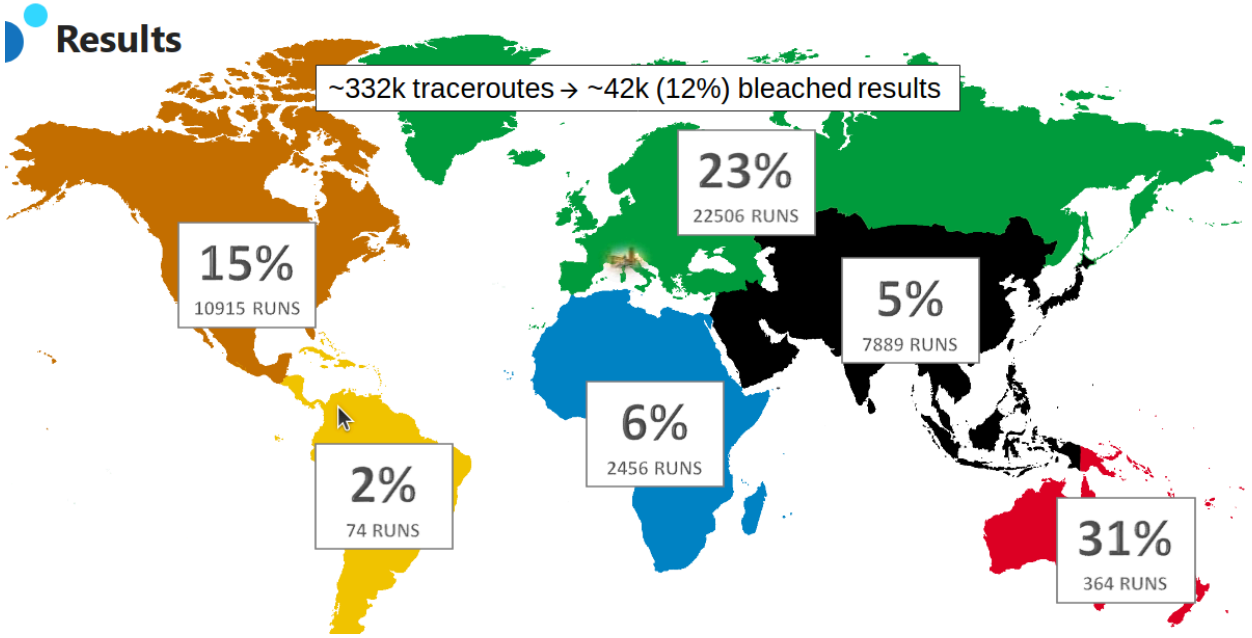
- Los routers gestionan las colas e introducen CE de forma diferente.
- Los extremos usan **AccECN**: incluye en opción de TCP el número de paquetes con CE.
  - **por defecto en Linux 7.0, 13 abril 2026**
- Los extremos usan **TCP Prague** y reducen cwnd proporcionalmente a la información AccECN.
  - **experimental**

> La señal ya no es binaria. La reacción ya no es cwnd/2

¿Estamos preparados para el despliegue de L4S?

Medidas traceroutes, Luca Sani - Catchpoint (RIPE 88, 2024)

- Pruebas desde los nodos de Catchpoint en todo el mundo para saber si se están borrando los bits IP ECN en routers intermedios.
- Envío de segmentos TCP con TTL creciente (TCP SYN+ECE+CWR e IP ECT(1)), respuesta ICMP (¿lleva IP ECT(1)?)
- Los resultados están sesgados por la selección del origen/destino de la prueba pero muestra datos interesantes: en Europa **36 ASs son los responsables del borrado de bits ECN**.



Medidas de ECN · Geoff Huston (oct. 2025)

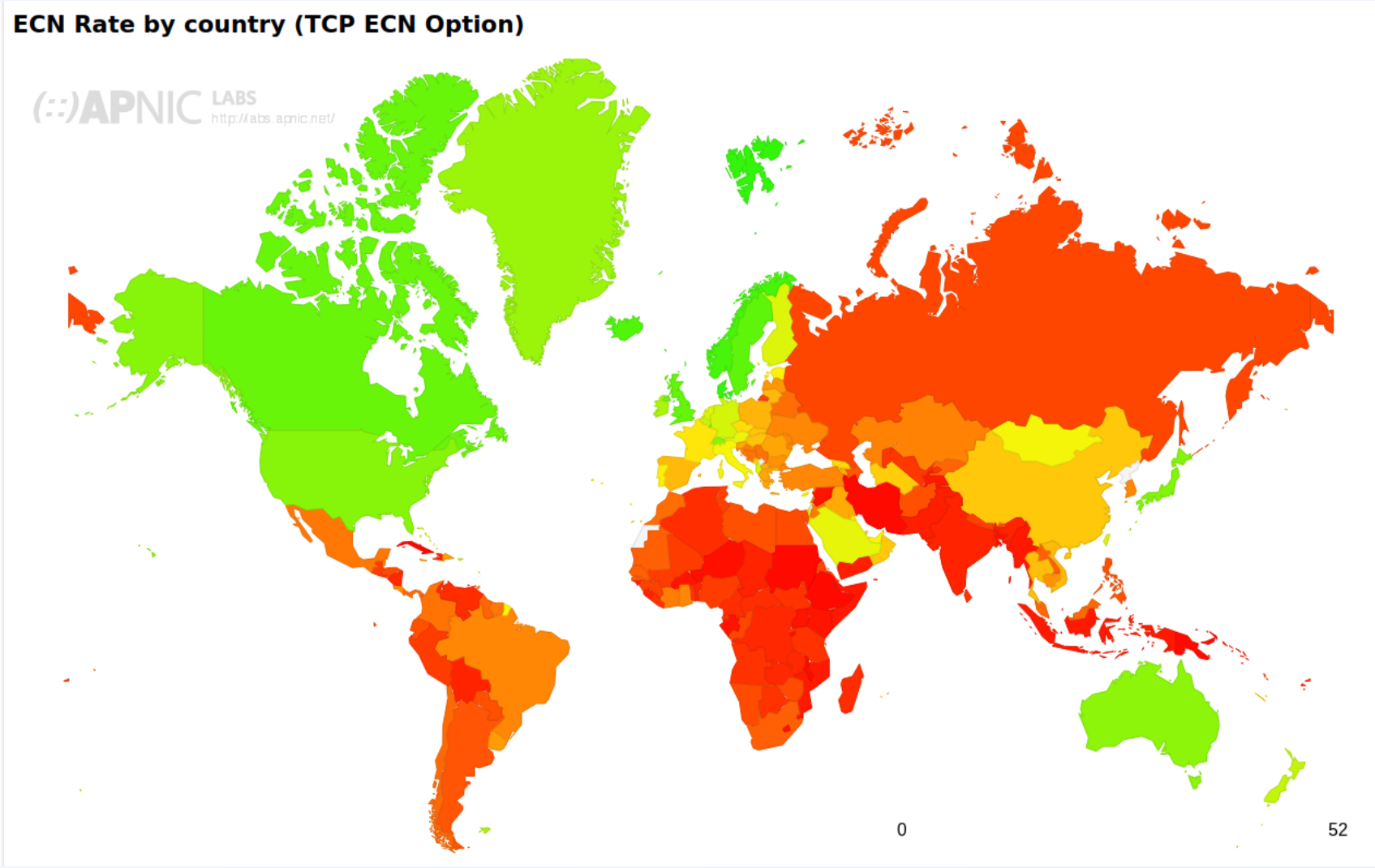
- Medidas de tráfico TCP entre navegador y servidor HTTP (6 meses). El uso de TCP ECN en el cliente era **despreciable (2–3%)**:
  - iOS/macOS con ECN desde 2015 → deberían verse 15–20%
  - Android → debería verse ~60%
  - ¿Alguien en el camino está **borrando los bits ECN de sesiones salientes TCP**?
- Además si una sesión TCP ha negociado el uso de ECN, el resto de segmentos TCP con datos deberían llevar ECT(0) o ECT(1) en la cabecera IP. Sin embargo, 3.57% en promedio borran los bits ECT (pero algunos más que otros: USA 63%, Israel ~49%, etc)
- Algunos ISPs borran los bits IP ECN en **todas** las conexiones TCP:
  - O2BROADBAND (GB), Chubu Telecomunications (JP), GET Norway (NO), ICC Corporation (JP)...

> *"Frankly, I'm not very optimistic about ECN's prospects. Much of today's technology environment has been focused on reducing external dependencies, rather than increasing them. ECN relies on the opposite of this trend."*
> — G. Huston

Datos de APNIC 13 de abril 2026

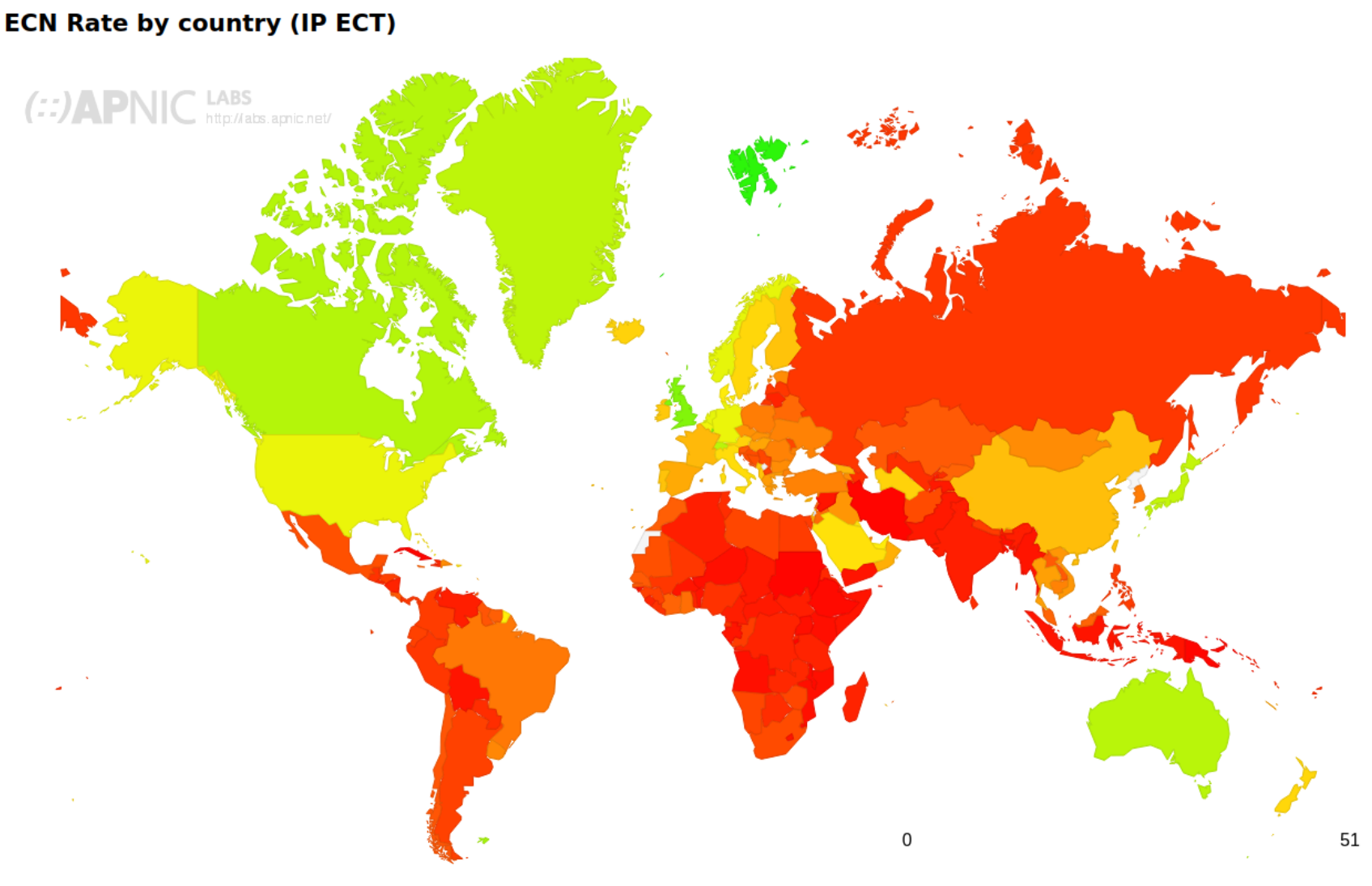
siete meses después...

Clientes con ECN activo en TCP



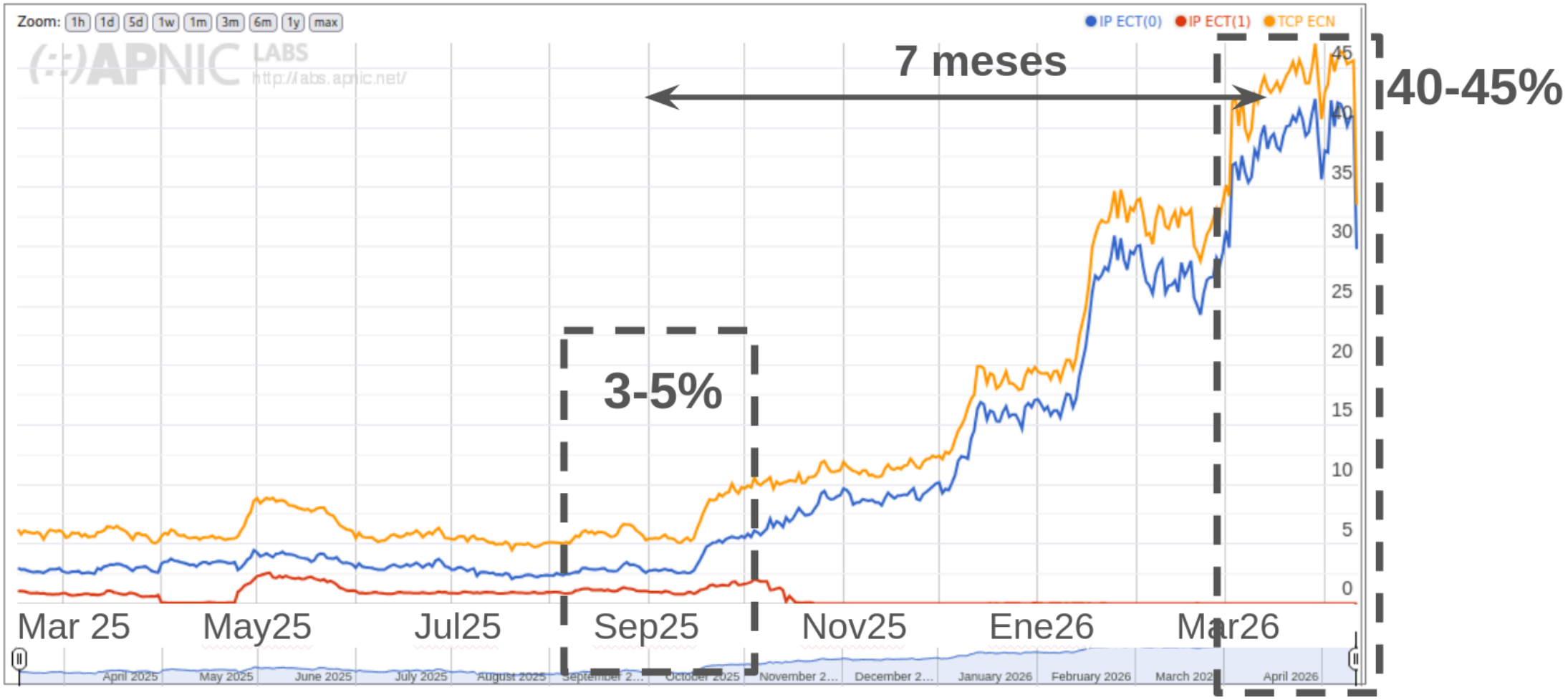
SYN+ECE+CWR · Fuente APNIC · lunes 13 de abril de 2026

Paquetes con IP ECT



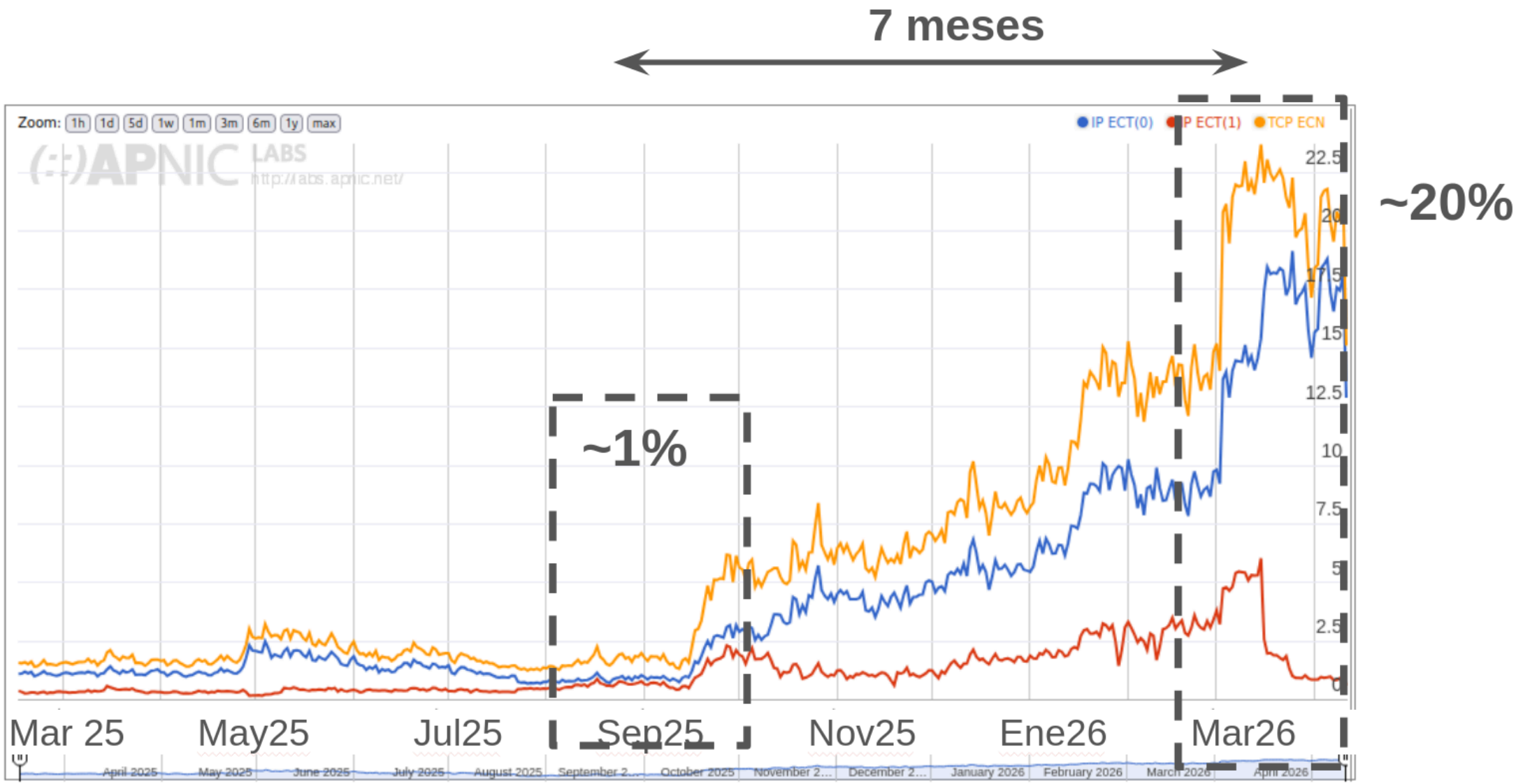
Fuente APNIC

UK: TCP ECN, IP ECT(0), IP ECT(1)



7 meses. De 3–5% a 40–45%.

España: TCP ECN, IP ECT(0), IP ECT(1)



7 meses. De ~1% a ~20%.

¿Qué ha pasado? ¿Cómo va a evolucionar?

Comcast despliega **L4S** en Xfinity (enero 2025)

- 6 ciudades USA · Reducción significativa de la latencia
- Primeras aplicaciones: FaceTime, GeForce Now, Steam

T-Mobile (EEUU) — primer operador inalámbrico con L4S a escala (julio 2025)

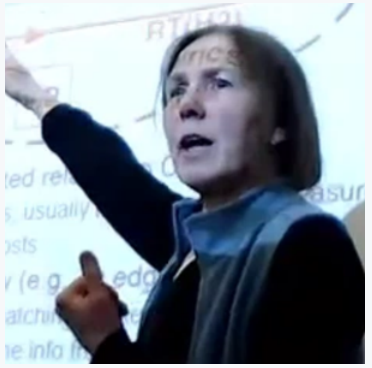
Linux 7.0 (13 abril 2026) activa **AccECN** por defecto.

Sus ideas siguen construyendo Internet

Van Jacobson

Sally Floyd

Lixia Zhang



Kathleen Nichols