Respect the ORIGIN! A Best-case Evaluation of Connection Coalescing

Sudheesh Singanamalla
Muhammad Talha Paracha
Suleman Ahmad
Jonathan Hoyland
Luke Valenta
Yevgen Safronov
Peter Wu
Andrew Galloni
Kurtis Heimerl
Nick Sullivan
Christopher A. Wood
Marwan Fayed
What is connection coalescing?

Same IP addresses but results in multiple **possibly blocking** DNS queries.

1. `example.com` AAAA?
2. `IP_A, IP_B`
3. TCP+TLS, HTTP (IP_A)
4. HTML <200 OK>
Next: What happens for subresources?

1. example.com AAAA?

2. IP_A, IP_B

3. TCP+TLS, HTTP (IP_A)

4. HTML <200 OK>

https://example.com
Chrome’s Approach: IP addresses for different hostnames must match

1. example.com AAAA?
2. $IP_A, IP_B$
3. TCP+TLS, HTTP ($IP_A$)
4. HTML <200 OK>
5. cdnjs.com AAAA?
6. $IP_B, IP_C$
7. TCP+TLS, HTTP ($IP_B$)
8. JS/CSS <200 OK>
9. https://example.com
10. https://cdnjs.com

Two separate TCP + TLS connections to two different IPs ($IP_A, IP_B$)
Firefox’s Approach: Transitivity between sets of IPs

1. example.com AAAA?
2. IP_A, IP_B
3. TCP+TLS, HTTP (IP_A)
4. HTML <200 OK>
5. cdnjs.com AAAA?
6. IP_B, IP_C
7. TLS, HTTP (IP_A)
8. JS/CSS <200 OK>

https://example.com
https://cdnjs.com

Reuses TCP connections to IP_A because of transitivity (IP_A ~ IP_B ~ IP_C)
Key Research Questions

A. How much of the Internet is coalescable?
   a. Where are the subresources located?
   b. How are coalescable sub-resources distributed?

B. What changes are required to enable missed opportunities?

C. Can this be done (and at scale)?
Where are the subresources located?

Insights:

1. 14% of web pages have a dependency on resources from one other AS.

2. More than 50% of webpages need no more than 6 ASes for all subresources.
Where are the most coalescable sub-resources?

<table>
<thead>
<tr>
<th>Rank</th>
<th>AS Number</th>
<th>Org. Name</th>
<th>#Req</th>
<th>%</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>AS 15169</td>
<td>Google</td>
<td>7932198</td>
<td>22.10</td>
</tr>
<tr>
<td>2</td>
<td>AS 13335</td>
<td>Cloudflare</td>
<td>4937395</td>
<td>13.75</td>
</tr>
<tr>
<td>3</td>
<td>AS 16509</td>
<td>Amazon 02</td>
<td>3017176</td>
<td>8.40</td>
</tr>
<tr>
<td>4</td>
<td>AS 14618</td>
<td>Amazon AES</td>
<td>2019308</td>
<td>5.62</td>
</tr>
<tr>
<td>5</td>
<td>AS 54113</td>
<td>Fastly</td>
<td>1281402</td>
<td>3.57</td>
</tr>
<tr>
<td>6</td>
<td>AS 16625</td>
<td>Akamai AS</td>
<td>1087172</td>
<td>3.02</td>
</tr>
<tr>
<td>7</td>
<td>AS 32934</td>
<td>Facebook</td>
<td>998685</td>
<td>2.78</td>
</tr>
<tr>
<td>8</td>
<td>AS 20940</td>
<td>Akamai Intl. B.V.</td>
<td>583700</td>
<td>1.62</td>
</tr>
<tr>
<td>9</td>
<td>AS 16276</td>
<td>OVH SAS</td>
<td>548107</td>
<td>1.52</td>
</tr>
<tr>
<td>10</td>
<td>AS 24940</td>
<td>Hetzner Online GmbH</td>
<td>469293</td>
<td>1.30</td>
</tr>
<tr>
<td>Total</td>
<td></td>
<td></td>
<td>6368</td>
<td></td>
</tr>
</tbody>
</table>

Insights:

1. The top 10 ASes handle more than 60% of all web requests for subresources.

2. Connection re-use potential (Min. number of connections) could be approximated to number of unique ASes contacted.

Note: Coalescing opportunities exist because of CDNs!

(further supported by Fayad et al. SIGCOMM '21)
Challenges with ORIGIN Frames (RFC 8336) *(Standardized in 2018)*

1. Default ORIGIN Frame standard allows any hostname(s) to be sent by the server (*lack of authority*).

2. Clients validate the hostnames in the ORIGIN frame for authenticity
   a. Firefox is the only client which supports ORIGIN Frame
   b. Clients resolve DNS queries and retrieve TLS Certificates
      i. If the IP addresses match IP based coalescing results.
      ii. Else, new TCP+TLS connections are made.

3. Lack of server software support for ORIGIN Frames.

4. Not widely adopted ... yet, despite standardization!
Authoritative ORIGIN Frames (RFC 8336) could preclude DNS

1. example.com AAAA?
2. IP_A, IP_B
3. TCP+TLS, HTTP (IP_A)
4.1 BTW, Ask me for {{cdnjs.com}}
4.2 HTML <200 OK>
5. HTTP
6. JS/CSS <200 OK>

Could Prevent unwanted DNS queries if authority established
Respect the ORIGIN! A Best-case Evaluation of Connection Coalescing

**Authoritative ORIGIN Frames (RFC 8336) could preclude DNS**

1. example.com AAAA?
2. IP₁, IP₂
3. TCP+TLS, HTTP (IP₁)
4.1 BTW, Ask me for [[cdnjs.com]]
4.2 HTML <200 OK>
5. HTTP
6. JS/CSS <200 OK>

Requires changes to certificates and introduction of **ORIGIN Frames**

Could Prevent unwanted DNS queries if authority established
Modelling: > 60% improvement in Number of DNS and TLS connections

Modelled Timeline reconstruction when *.example.com is proxied by the CDN network also serving cdnhost.com
Real World – IP coalescing ties services, hard to coordinate

Always returns the same IP address for all services

Browser reuses connection because of same IP

IP Coalescing Requires Coordination
Real World – ORIGIN...

No changes to DNS or service coordination necessary!

Service sends ORIGIN Frames

Advantage: Does not disrupt existing traffic engineering practices
Real World – ORIGIN... makes coalescing practical.

*(DNS)*

- Service sends ORIGIN Frames
- Advantage: Little difference to ‘wire-line’ activities

No changes to DNS or service coordination necessary!
Takeaway 1: Connection Coalescing works in practice!

- ~50% reduction in number of new connections to the cdnjs hostname we attempted coalescing to.

**Client:** Reduced Number of Cryptographic Certificate Validations.

**Client:** Active measurements show ~65-70% connections coalesced.

**Server:** Reduced number of connections → allow more client connections.
Takeaway 2: No-worse performance, almost immeasurable improvement

(a) Measured and modelled.  
(b) IP and ORIGIN

Wide impact depends on path characteristics, AND bottleneck share, AND numbers of operators that support ORIGIN.
Takeaway 2: No-worse performance, almost immeasurable improvement

Performance cannot be assumed to improve, and **should** be avoided as primary motivation.

(e.g. fewer bottleneck connections, and more)
Open Source: We contribute a public large scale server implementation

ORIGIN Frames are yet to see large scale adoption, no public server implementation exists.

We contribute (to our knowledge) the first public server side implementation.

https://github.com/cloudflare/net-originiframe

https://github.com/cloudflare/go-originiframe
Needs Careful Deployment – Non RFC Compliant network stacks exist

Deployment of ORIGIN Frames resulted in uncovering **compliance issues** in popular antivirus and Internet security software which did not drop unknown HTTP/2 frames and instead resulted in connection tear-down.
Claim: ORIGIN Frame based Coalescing improves privacy

Whose privacy? What does it mean?

Each coalesced connection hides an otherwise exposed plaintext SNI and prevents at least one additional plaintext DNS query-response.
Resource Scheduling Opportunities at the Endpoints!

Extra Connections → non-deterministic path characteristics
Resource Scheduling Opportunities at the Endpoints!

Extra Connections $\rightarrow$ non-deterministic path characteristics

Call to action: Implement ORIGIN Frames!
Thank You!

sudheesh@cs.washington.edu
marwan@cloudflare.com

Link to paper: https://dl.acm.org/doi/10.1145/3517745.3561453

We'd like to thank Larry Archer, Michel Bamps, Petros Gigis, Vasileios Giotsas, Vânia Gonçalves, John Graham-Cumming, Mihir Jham, Lucas Pardue, Kyle Schomp, and Avani Wildani.