The Life of a Request: Past, present and future

Marwan Fayed*
Cloudflare Research

* A view from within Cloudflare; all opinions are my own, but not all artwork.
General Overview

90+ Global Interns
- San Francisco
- New York
- Austin
- Champaign
- London
- Lisbon
- Singapore
- And many more places!

15+ Teams
- Research
- Marketing
- Special Projects
- Engineering
- Product Management
- Security
- Legal
Activities & Presentations

❖ 1:1 Chats with C-suite Team
❖ Mentor & Buddy Program
❖ Intern Week: Games, Challenges,
❖ Bi-Weekly Newsletters
❖ CloudflareTV Episodes
❖ Blog Posts
❖ Company-Wide Internship Presentations in August
Visit https://www.cloudflare.com/en-gb/careers/jobs/

We're helping build a better Internet. **Come join us.**
Let's first establish

A Shared Language
“The life of a request” is a phrase...

- Used to describe where packets go to fulfill a request
  - e.g. all the pieces involved in retrieving example.com

- Consists of
  - DNS
  - content request (e.g. HTTP)
  - all services, requests, or connections, that trigger as a result.
Jargon First!

User or Client
(or “eyeball” if I slip)

Server

Origin server
(source of truth)

Server somewhere on the Internet
Tip 1: In computer networking, the ‘starting’ answer to ‘why’?
Tip 1: In computer networking, the ‘starting’ answer to ‘why’?

9 times out of 10: Scale

Otherwise:
Tip 1: In computer networking, the ‘starting’ answer to ‘why’?

9 times out of 10: Scale

Otherwise: Money
Tip 1: In computer networking, the ‘starting’ answer to ‘why’?

9 times out of 10: Scale

Otherwise: Money

Tip 2: In content distribution and delivery, thinking of ‘what?’ and ‘how?’...
Tip 1: In computer networking, the ‘starting’ answer to ‘why’?

9 times out of 10: Scale

Otherwise: Money

Tip 2: In content distribution and delivery, thinking of ‘what?’ and ‘how?’

...it helps to define things in terms of connection termination.
More to come but...

Question to ponder: Who is the Customer?
Evolution of Content Delivery Networks

“In the beginning...”
1. **One client, one server**

- This is the traditional textbook view
  - The foundation of user-oriented connections, for all delivery models
  - Great for teaching
  - Dominant through late 1990s
1. **One client, one server**

- This is the traditional textbook view
  - The foundation of user-oriented connections, for all delivery models
  - Great for teaching
  - Dominant through late 1990s
1. One client, one server

- What is the problem here?
  - Hint: The ‘Internet’ is growing fast (and faster with time)
  - And what is the natural step to solve it?
IDEA: One client, many servers?

- The natural response is to have multiple servers
It is One client, but *not just* many servers

- What is missing? Consider:
  - *routing*, and by extension...
  - ... *connection termination*
It is One client, but not just many servers

- Physics: A single packet can’t reach multiple recipients, nor should it.
  - (unicast) Routing protocols forward to a single interface, not many;
  - they have to, otherwise connection termination makes no sense
It is One client, but *not just* many servers

- Insight: *something* has to terminate the connection.
2. The server “farm”: One client, one server, many back-ends

- One IP address for routing to, and connection termination
  - with multiple connections at the back-end
  - connected to duplicate servers that mirror content.
2. The server “farm”: One client, one server, many back-ends

- Notice that here, the provider **is** the origin of content, as well.
  - e.g. the IBMs, Microsofts, Yahoos, etc.
  - and so the back-ends might share or process between them.
The “mix-and-match” period

Separating provider service from origin services to share resources (or improving scale, while saving money, too).
Background: Putting “bits on the wire” has a monetary cost

- Remember this?

Unicast is too expensive

“Time (in flight) is money, but there is even more to say...
Background: Putting “bits on the wire” has a monetary cost

- Scaling-up with more servers → costs money
  - hardware
  - storage
  - space
  - person time, and expertise
  - *Energy*

- Scaling-up with more network capacity → costs money
  - network capacity
  - more specialized devices (routers and switches)
  - ...and everything else from above!
What to do if you want to scale *and* save money?

Because as demand grows...

...most are unable to increase supply.
Leads to two shifts in content delivery (late 1990s to late 2000s)

The ‘Classic’ CDN Model

- Content closer to users:
  - Serve content on behalf of the origin
  - But “ground truth” lies with owners
- Relies on DNS (& CNAME) to redirect, with unicast, e.g. example.com → example.akamai.com

Virtual Hosting

- Infrastructure without costs
  - Host services at provider
  - Maintain control (origin)
  - Offers isolation
  - Scale-up as needed
- Rely on provider for servers, up-time, network resources, etc.
Customer has a remote interface to control their origin (e.g. ssh or web)

Virtual Hosting

- The customer is on-site.

...all co-exist.
Customer has no access to provider, only updates content

The ‘Classic’ CDN Model

- The customer is off-site
The Reverse Proxy period + Anycast

Keeping the content,
while shifting the origins edge out into the network
The classic ‘forward proxy’ (for comparison)

- A forward proxy terminates client connection
- A forward proxy ‘acts’ on behalf of the user (or user’s organization)
- Proxy is transparent, and pre-configured
The ‘reverse proxy’

- A reverse proxy terminates client connection
- A reverse proxy acts on behalf of the origin
- Proxy is transparent, and inserted into request path via DNS
Functionally similar, operationally opposite

**Forward Proxy**
- Terminates client connection
- A forward proxy acts on behalf of the user (or user’s org)
- Proxy is transparent
- Pre-configured

**Reverse Proxy**
- Terminates client connection
- A reverse proxy acts on behalf of the origin
- Proxy is transparent
- Inserted into request path via DNS
Important: The reverse proxy acts on behalf of the origin

- Provides “network as a service” (NaaS) functionality
  - e.g. DoS protection, certificate mgmt (HTTPS), firewall, cache, ...
In a Cloudflare context

- DNS (authoritative)
- DoS
- Firewall
- HTTPS
- Caching not content
- ...

![Cloudflare diagram](image-url)
Breaking this down into steps

- **Authoritative DNS** is the key to service management

1. A example.com?
2. a.b.c.d
3. https://example.com
4. <h1>Welcome</h1>
5. (stay tuned for more on step 4.)
Pause for a moment -- there is a strong implication on an Internet ideal

- Seminal work, 1981.

END-TO-END ARGUMENTS IN SYSTEM DESIGN

J.H. Saltzer, D.P. Reed and D.D. Clark*

M.I.T. Laboratory for Computer Science

This paper presents a design principle that helps guide placement of functions among the modules of a distributed computer system. The principle, called the end-to-end argument,
Pause for a moment -- there is a strong implication on an Internet ideal

- The end-to-end principle says, “application-specific features reside in the communicating end nodes of the network, rather than in intermediary nodes, such as gateways and routers, that exist to establish the network. [Wikipedia]”

- What does this mean for content delivery architectures?
  - Ask David Clark, Nov 20!
Implication: From e2e to e2e(dge)

- In the content-oriented Internet, connections are end-to-edge

(A) 1 connection
(B) 2 connections
(C) 3 connections -- 6 IP addresses, 4 of those in the edge infrastructure!
But inside the edge, it’s possible to adapt to Internet events
Globally, within 50ms of 95% of world’s Internet users

Spans more than 250 cities in more than 100 countries

10,000 Interconnects

12M DNS requests per second

25M+ Web requests per second
Cloudflare operates Anycast for all content (not just DNS)

- Every PoP / datacenter
  - advertises same IP ranges

- Closest PoP / datacenter
  - selected by BGP shortest path

- From “One machine, one IP”
  - to “Many machines, many IPs!”
Anycast in Action (comparison):

```
elTio-Cf:~ marwan$ dig +short cloudflare.com
104.17.175.85
104.17.176.85
```
Anycast in Action:

eLTio-Cf:$ marwan$ dig +short cloudflare.com
104.17.175.85
104.17.176.85

eLTio-Cf:$ marwan$ dig +short google.com
172.217.164.206

klovia:$ mf208$ dig +short google.com
216.58.210.238
“Stay close” is another consequence of anycast
Anycast in Action:

eTio-Cf:~ marwan$ dig +short cloudflare.com
104.17.175.85
104.17.176.85

eTio-Cf:~ marwan$ sudo mtr -4 -r cloudflare.com
HOST: C0.
  1.  192.168.1.1  0.0%  10  1.9  12.4  1.3 107.6 33.5
  2.  10.90.0.1   0.0%  10 12.2  22.9  9.6  53.8 18.0
  3.  10.0.68.89  0.0%  10 27.5  33.5 20.8  63.3 16.3
  4.  10.0.18.69  0.0%  10 18.6  31.9 16.8  62.4 17.3
  5.  cloudflare.ip4.toronto.int 0.0%  10 19.2  27.3 16.9  49.4 13.6
  6.  104.17.176.85 0.0%  10 17.0  20.5 16.4  39.9  7.0
Anycast in Action:

```
eltio-cf:~ marwan$ dig +short @klovia 104.17.175.85
104.17.175.85
eltio-cf:~ marwan$ mf208$ mtr -4 -r cloudflare.com
Start: 2020-05-28T00:34:52+0100
HOST: klovia
  1.l -- 138.251.22.253  0.0% 10  0.9  0.7  0.3  3.0  0.8
  2.l -- 138.251.1.128  0.0% 10  0.4  7.5  0.3 39.9 15.1
  3.l -- 138.251.0.65  0.0% 10  0.5  0.5  0.5  0.6  0.0
  4.l -- xe-0-1-0.und-ban1.ja.net  0.0% 10  0.7  0.7  0.7  0.8  0.0
  5.l -- ae1.und-ban3.ja.net  0.0% 10  0.9  1.1  0.8  3.3  0.7
  6.l -- ae24.leedaq-sbr2.ja.net  0.0% 10  6.0  6.0  5.9  6.7  0.2
  7.l -- ae30.manckh-sbr2.ja.net  0.0% 10 11.2 14.3  8.4 34.1  9.4
  8.l -- ae0.manckh-ban2.ja.net  0.0% 10  8.5  8.5  8.4  8.9  0.2
  9.l -- ixmanchester.as13335.net  0.0% 10  9.0 11.6  8.9 34.9  8.2
 10.l -- 104.17.176.85  0.0% 10  8.4  8.4  8.3  8.4  0.0
HOST: C0:
  1.l -- 192.168.1.1
  2.l -- 10.90.0.1
  3.l -- 10.0.68.89
  4.l -- 10.0.18.69
  5.l -- cloudflare.ip4.torontoint  0.0% 10 19.2 27.3 16.9 49.4 13.6
  6.l -- 104.17.176.85  0.0% 10 17.0 20.5 16.4 39.9  7.0
```
“Stay close” is another consequence of anycast
Inside Cloudflare

...and ways this influences thinking.

But first an aside on ‘value’.
Keep things Simple: Every Internet user can be a customer

**Free**

$0/mo

Cloudflare for Individuals is built on our global network. This package is ideal for people with personal or hobby projects that aren’t business-critical.

**Support response:** Community forums available for tips and troubleshooting.

**Free**
- Fast, easy-to-use DNS
- Free automated SSL certificates
- Global content delivery network (CDN)
- Unmetered mitigation of DDoS attacks with up to 67 Tbps capacity
- Up to 100k Workers requests and 30 scripts
- 3 Page Rules
- Enhanced security with Web Application Firewall (WAF)
- Bot report & basic mitigation
- DDoS alerts

**Pro**
- Lossless image optimization
- Accelerated mobile page load speed
- Privacy-first analytics
- 24x7x365 chat support
- 100% uptime SLA
- CNAME set-up compatibility
- PCI DSS 3.2 compliance

**Business**

**Enterprise**

Need help figuring out what products you need? Answer 4 easy questions to get a recommendation >
And remember, money changes things

- Cloudflare’s pricing model changes definitions of ‘value’
And remember, money changes things

- Cloudflare’s pricing model changes definitions of ‘value’

- DNS (authoritative)
- DoS
- Firewall
- HTTPS
- Caching
- ...

$0 \leftrightarrow CLOUDFLARE \leftrightarrow $0
Inside Cloudflare:

Keep the systems simple, or at least uniform.

Step 4., as promised.
The conventional view of the data center

- Large, dedicated facilities
- Require large up-front investments
- Different servers and machines have different services
  - A single request can spawn countless other ‘microservice’ requests
- (worth thinking about other attributes)
“Life of a Request” inside Cloudflare

ECMP + Consistent Hashing → forwards all packets in a flow to a single server!
"Life of a Request"

First thing that happens: Denial of service protections
“Life of a Request”

Next is “layer-4 load balancing -- details on ‘Unimog’ at https://blog.cloudflare.com/unimog-cloudflares-edge-load-balancer/
“Life of a Request”

L4LB to other PoPs and data centers, too
Increasingly, one global edge
“Life of a Request”

Clients connect only after DoS protections and load-balancing!
“Life of a Request”
“Life of a Request”

Every server participates in a shared distributed cache.
Every edge server is identical.
Also, there are core data centers for logs & configs
Sometimes Twitter knows before we do.

This is getting better!
Every edge server is identical.
Questions welcomed.