USING APACHE TRAFFIC CONTROL

TO CACHE ANY WEB OBJECT AT SCALE

Jeff Elsloo

jeffrey_elsloo@cable.comcast.com

September 10, 2019

COMCAST

INTRODUCTION

JEFF ELSLOO, SR. PRINCIPAL ENGINEER @ COMCAST

- 11 years with Comcast, the last 6 with CDN Engineering
- Participated in the initial “IPCDN” deployment at Comcast
- Participated in the effort to open source the project
- Member of the Apache Traffic Control PMC
- Currently serve as lead architect for the CDN at Comcast
- Motorcycle enthusiast – any type will do

COMCAST

OVERVIEW

CDN 101

A quick overview of CDN concepts, and how Apache Traffic Control and Apache Traffic Server fit into the big picture.

CACHING AT SCALE

From routing to caching, we will learn how to use ATC and ATS to effectively cache different types of web objects using several approaches.

RETROSPECTIVE

After learning about different routing and caching strategies, we will examine a few lessons learned and see what is next on the horizon.

COMCAST

CDN 101

CDN BASICS

DELIVER BITS LOCALLY

Content Delivery Networks are comprised of ***geographically distributed*** web servers that cache content and are often configured in a hierarchy.

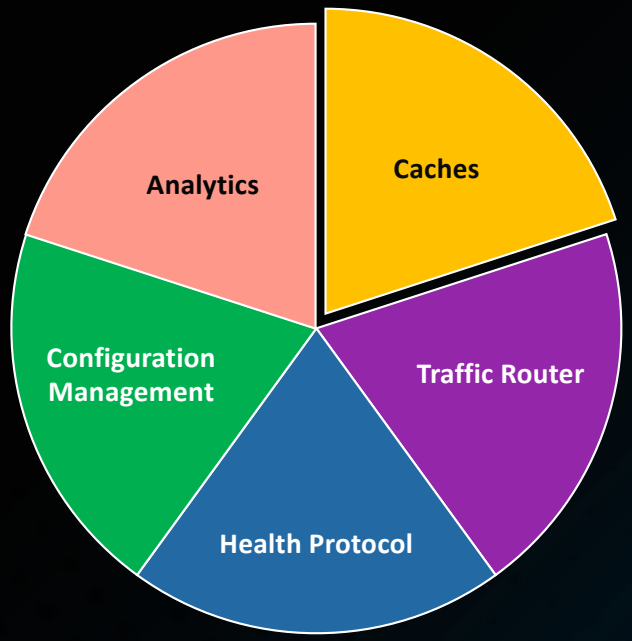
In order to route traffic to the ideal cache, CDNs employ routing technology that operates at some mix of layers 3, 4 and 7 of the OSI model.

- Route traffic to the ideal cache using
  - ***DNS***
  - ***HTTP***
  - Anycast
- The cache that receives the initial request is called the ***edge cache***
  - Checks local cache, services request if present (***cache hit***)
  - Performs upstream request to parent and fills cache (***cache miss***)
  - Parent could be another cache; each upstream layer is called a ***tier***
- Final upstream host is known as the ***origin***

COMCAST

COMPONENTS

HOW APACHE TRAFFIC CONTROL RELATES TO APACHE TRAFFIC SERVER



A COMPLIMENTARY RELATIONSHIP

Apache Traffic Control provides the control plane, health protocol, routing component, and basic statistics and analytic capabilities.

Apache Traffic Server is the main caching platform used with ATC, though ATC does provide a memory-based caching component.

ATC and ATS are both designed to be multi-tenant. ATC allows for multiple CDNs and Delivery Services (properties) to be configured and deployed to ATS (or ATC Grove!) caches.

COMCAST

HEALTH PROTOCOL INFLUENCES ROUTING

TRAFFIC MONITOR MAINTAINS CDN HEALTH STATE

- Polls all caches set to `REPORTED` or `ADMIN_DOWN`
- Problem detection (IP, TCP, application)
- Configurable thresholds
  - System (timeouts, NIC throughput, load average)
  - Application (remap stats)
- 2 second ***system poll***, 6 second ***stat poll*** (configurable)
- Optimistic Health Protocol
  - Raw health state polled from all peers (configurable)
  - All peers must agree on negative state
  - Traffic Router uses result to inform routing decisions

Traffic Control provides a monitoring component called Traffic Monitor that monitors all caches and implements the ***optimistic health protocol*** that is consumed by Traffic Router.

$$\forall m \; \exists c \; \text{Monitor}(m) \wedge \text{Cache}(c) \wedge \neg\text{Healthy}(m, c) \Rightarrow \neg\text{Available}(c)$$

COMCAST

TRAFFIC ROUTING AND DNS

AUTOMATED AUTHORITATIVE DNS SERVER

- Authoritative for CDN top level domain
- Configurable TTLs, RRset size, bypass destinations
- Dynamic responses for edge, static DNS entries for all else
- DNS-based “Federation”
- DNSSEC NSEC support
- EDNS0 client subnet extensions
- Localization (configurable)
  - (Deep) Coverage Zone
  - Geolocation by delivery service
  - Anonymous proxy blocking

Traffic Router provides a completely automated authoritative DNS server that generates all records based on the configuration in Traffic Ops, and the dynamic health state of the CDN. Responses to requests for some edge records are dynamically generated.

COMCAST

TRAFFIC ROUTING AND HTTP

HIGHLY TARGETED CACHING

- Fields initial requests for HTTP Delivery Services
- Uses the URL information to determine ideal cache
- Supports TLS and SNI
- Routing options
  - Default is an HTTP 302 redirect to ideal cache
  - 200 OK with JSON body
  - Client Steering, JSON array with or without 302
- Bypass destinations
- API and metrics

Traffic Router consumes the result of the ***optimistic health protocol*** from Traffic Monitor and uses the health state to influence cache selection once a client has been localized for all routing responses, regardless of Delivery Service type.

COMCAST

CONSISTENT HASHING

THE ENABLER OF EFFICIENCY WITHIN THE CDN

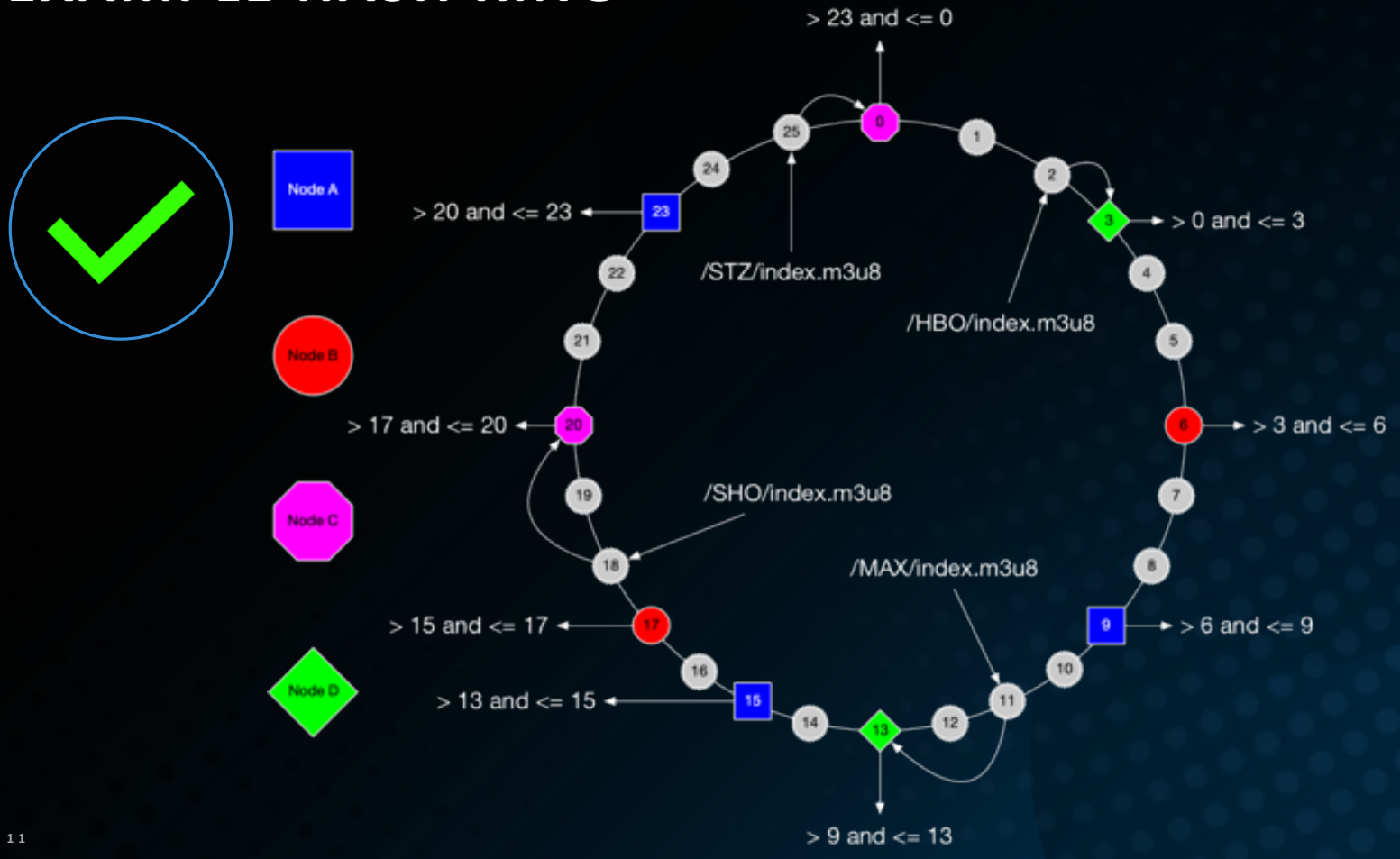
- The mechanism that provides cache efficiency within a CDN
- Allows `K/n` rehashed `Keys` for add/removals of `nodes`
- Avoids costly rehashes, and is perfect for dynamic CDNs
- Used throughout Traffic Router and Traffic Server
  - Consistent hash edge name for predictable dynamic RRsets
  - Consistent hash on path[1] for HTTP-based routing
  - Consistent hash on path for upstream requests to parents

Created by Daniel Lewin and F. Thomson Leighton at MIT[2], founders of Akamai Technologies.

[1]Optionally, selected query parameters too
[2]https://en.wikipedia.org/wiki/Consistent_hashing

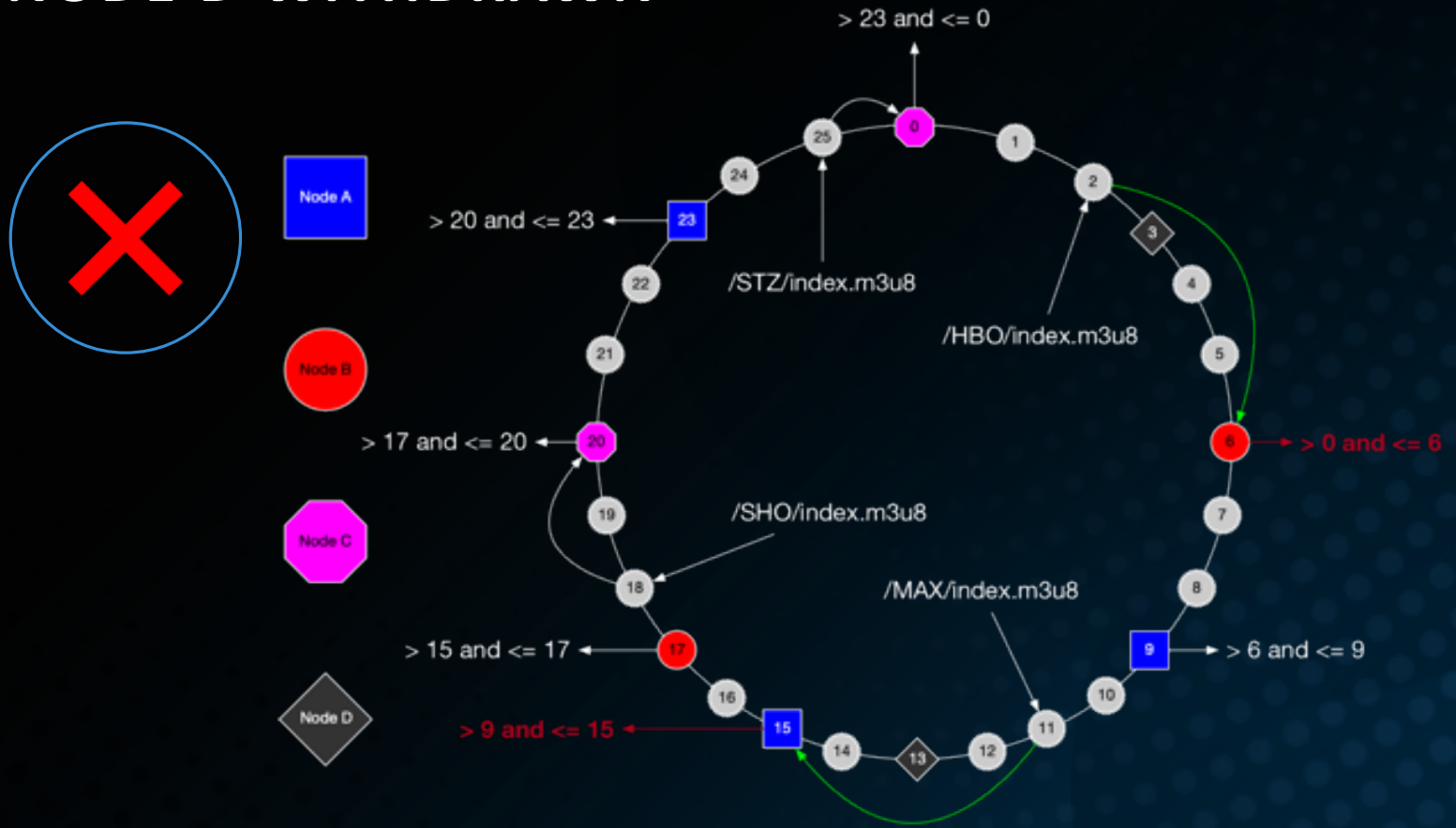
COMCAST

EXAMPLE HASH RING



COMCAST

NODE D WITHDRAWN



COMCAST

CACHING AT SCALE

OBJECT CLASSIFICATION AND CACHING STRATEGY

OBJECT TYPE AND ROUTING STRATEGY ARE LINKED TO EFFICIENCY

Object classification and use case must be analyzed in order to determine the best caching strategy. The Comcast CDN caches myriad object types.

The routing strategy selected influences ***cache efficiency*** and ***cache pollution***.

Cache efficiency at each tier depends on a number of factors:

- Volume size
- For each delivery service sharing a volume
  - ***Cache key*** configuration and how to handle query parameters
  - Upstream request and ***cache control headers*** received in response
  - Origin library size
  - Popularity, long tail, and ***cache promote***

COMCAST

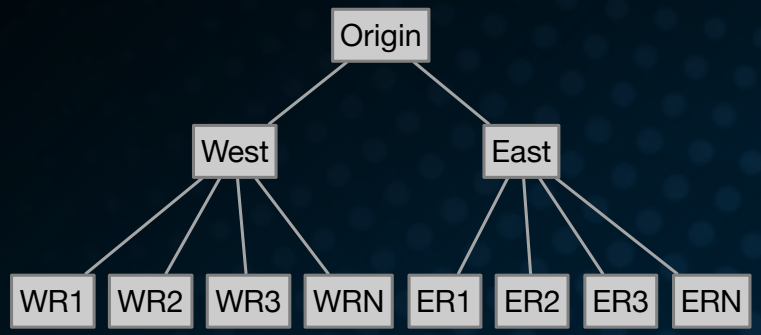
CDN CACHING HIERARCHY

CDNs may or may not choose to employ a tiered cache hierarchy using ***cache groups***. Traffic Control supports several options out of the box and will be enhanced in the future.

DELIVERY SERVICE TYPES MATTER

Today, storage volume type and whether an upstream tier is used is controlled by the delivery service type. In the future, ***Flexible Cache Groups*** will provide robust granularity with respect to storage and cache assignments within the topology.

- HTTP, DNS, ***national*** services use parents
- Live services use ***RAM cache***
- ***No cache*** services do not cache, go direct



COMCAST

TIMEOUT PYRAMID AND NORTHBOUND REQUESTS

TIME TO DETECT FAILURE DECREASES WITH EACH TIER

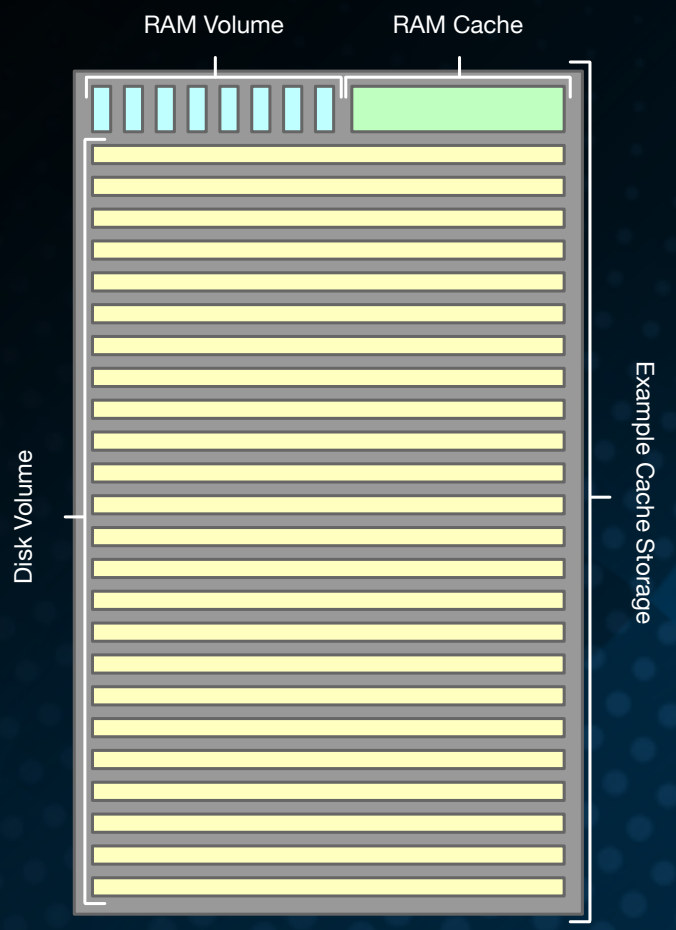
- Timeout and retry logic timing must decrease with tiers
- Total time to fail is a function of retries and timeouts
- Tiers ideally collapse all requests for a single object
- Tiers use persistent connections when possible
- Limits should be placed on max connections in and out
- Jumbo frames improve throughput
- Each tier is an opportunity to add latency on the miss case

COMCAST

CACHE STORAGE

DIFFERENT APPROACHES FOR DIFFERENT OBJECT TYPES

- Spinning drives for bulk storage
- RAM drives for transient content such as live TV
- RAM cache accelerates IO requests
- Solid-state drives such as NVMe now viable for bulk storage
- No caching; request termination for TCP efficiency
- Efficient caching and connection settings are key to protecting upstream resources



COMCAST

DNS DELIVERY SERVICES

DYNAMICALLY GENERATED AND LOCALIZED RRSET

- Locate zone
- Locate static records, or…
- Match Delivery Service
- Localize client
- Select healthy caches
- Consistent hash on FQDN
- Fill dynamic zone
- Serve response

DNS delivery services are a good match for small objects such as images, CSS and JavaScript.

Objects and libraries that have little impact on storage and benefit from low latency would use this type of service. Large VOD libraries are a poor fit.

COMCAST

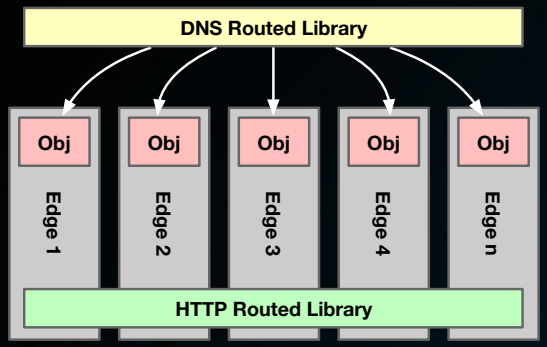
DNS DELIVERY SERVICE EXAMPLE

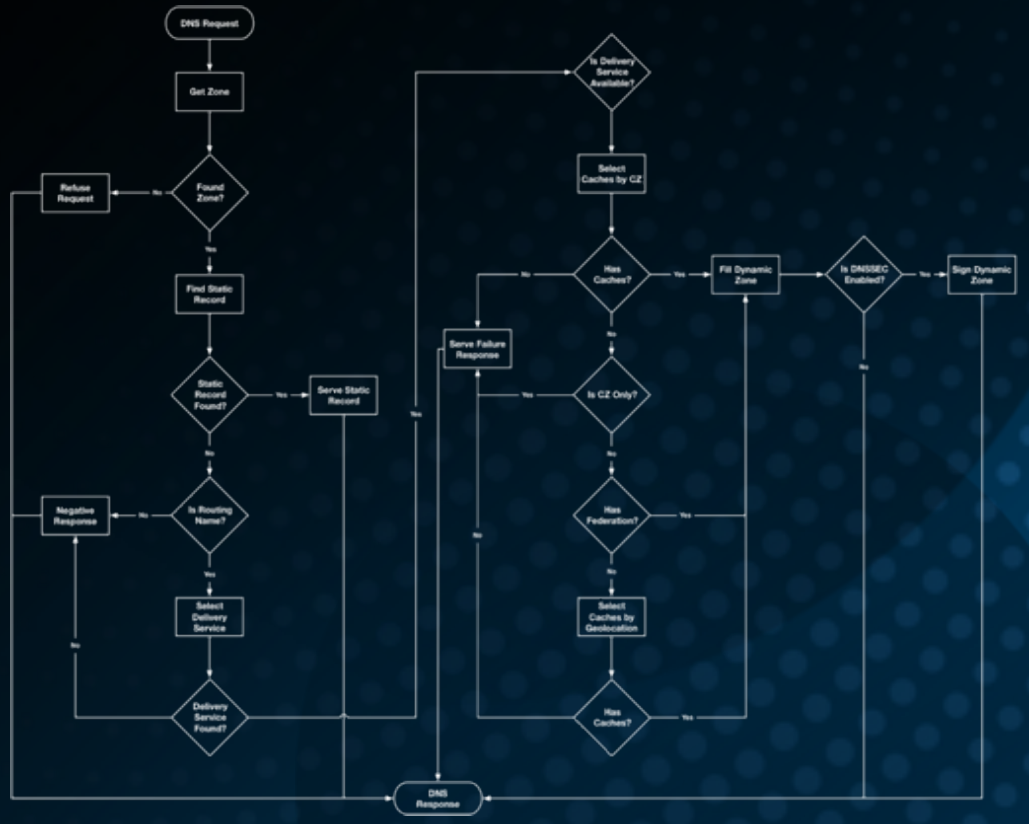
DNS query from **Denver**:

```
$ dig edge.images.cdn.example.com
edge.images.cdn.example.com. 30    IN    A    192.168.12.10
edge.images.cdn.example.com. 30    IN    A    192.168.175.10
edge.images.cdn.example.com. 30    IN    A    192.168.115.31
edge.images.cdn.example.com. 30    IN    A    192.168.10.64
```

…or the same request made from **San Francisco**:

```
$ dig edge.images.cdn.example.com
edge.images.cdn.example.com. 30    IN    A    10.59.132.53
edge.images.cdn.example.com. 30    IN    A    10.18.190.53
edge.images.cdn.example.com. 30    IN    A    10.16.119.92
edge.images.cdn.example.com. 30    IN    A    10.27.117.38
```





COMCAST

HTTP DELIVERY SERVICES

CLIENT IP AND REQUEST PATH ENABLE TARGETED CACHING

- What type of request?
- Match Delivery Service(s)
- Localize Client
- Select healthy caches
- Consistent hash on path
- Select ideal cache(s), disperse as necessary
- Order Subordinates
- Serve response (302 or JSON)

HTTP delivery services are a good match for large objects with large or infinite libraries such as those found with video on demand or live video.

Requires ***well behaved clients*** that ***stick to the edge cache*** after completing the request.

COMCAST

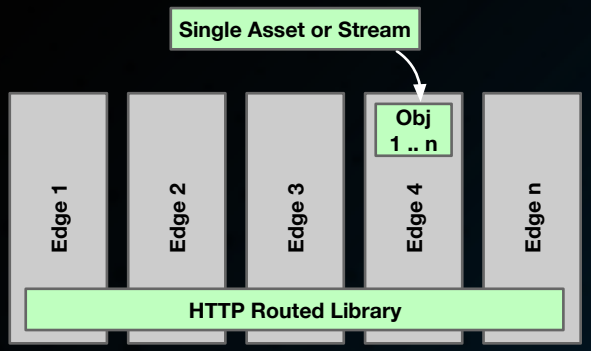
HTTP DELIVERY SERVICE EXAMPLE

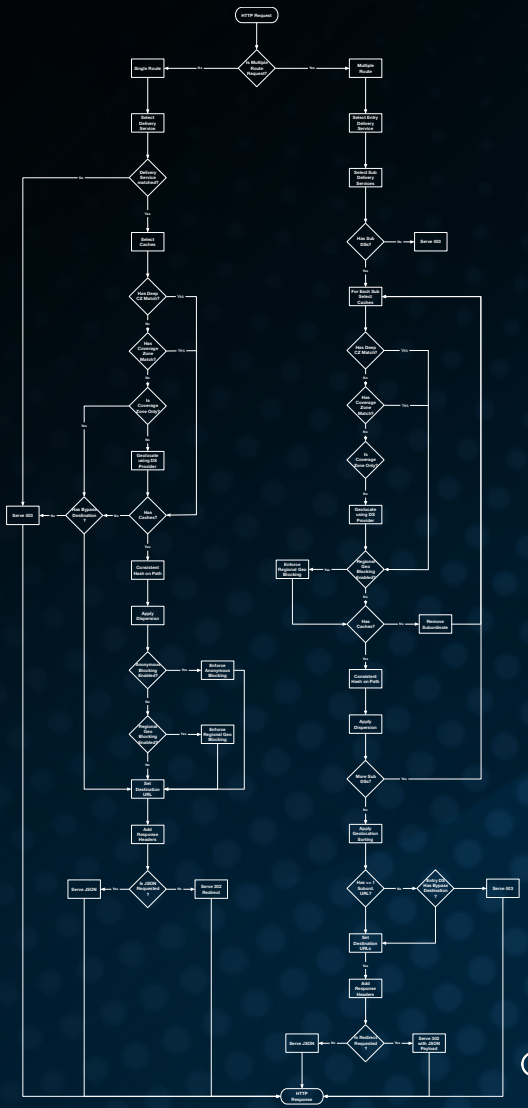
HTTP request from Denver:

```
> GET /foo.m3u8 HTTP/1.1
> Host: tr.linear.cdn.example.com
>
< HTTP/1.1 302 Found
< Location: http://edge-den-02.linear.cdn.example.com/foo.m3u8
< Content-Length: 0
< Date: Wed, 04 Sep 2019 16:38:18 GMT
```

…or the same request made from San Francisco:

```
< HTTP/1.1 302 Found
< Location: http://edge-sfb-10.linear.cdn.example.com/foo.m3u8
< Content-Length: 0
< Date: Wed, 04 Sep 2019 16:38:18 GMT
```



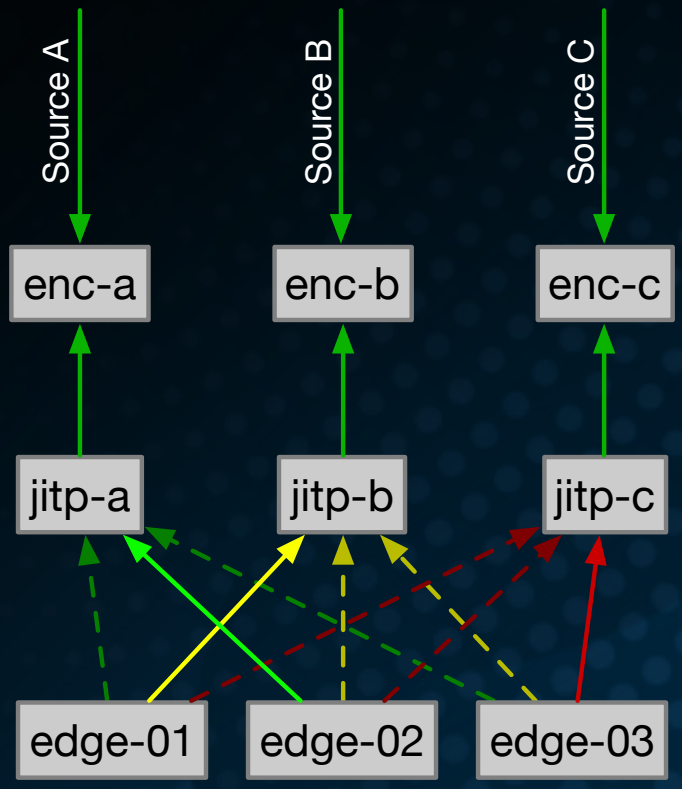


COMCAST

STEERING DELIVERY SERVICE EXAMPLE

In some cases, failover must be client initiated. Client steering provides multiple edge cache and delivery service options in a predictable order in case of any error, in particular HTTP errors that affect only certain paths or objects. The list can be made diverse using dispersion or by force.

```
< HTTP/1.1 302 Found
< Access-Control-Allow-Origin: *
< Location: https://edge-den-02.linear-
a.cdn.example.com/foo.m3u8
< Content-Type: application/json
< Content-Length: 206
< date: Wed, 04 Sep 2019 20:01:33 GMT
<
* Closing connection 0
{
   "locations" : [
      "https://edge-den-02.linear-a.cdn.example.com/foo.m3u8",
      "https://edge-den-01.linear-b.cdn.example.com/foo.m3u8",
      "https://edge-den-03.linear-c.cdn.example.com/foo.m3u8"
   ]
}
```



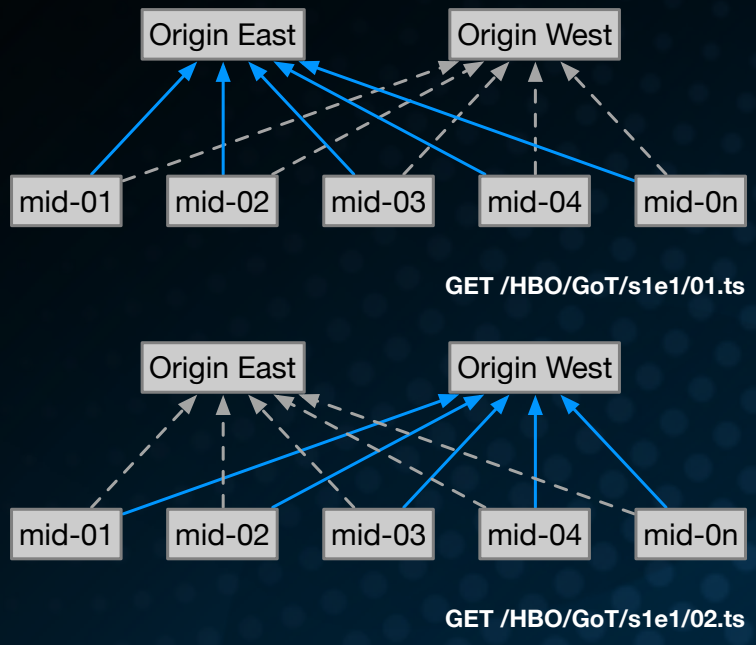
COMCAST

MULTI-SITE ORIGIN

In some cases, failover must be managed by the CDN. In this case, ATS can be configured to use multiple origins. This approach utilizes the same parent selection process used between tiers on the CDN.

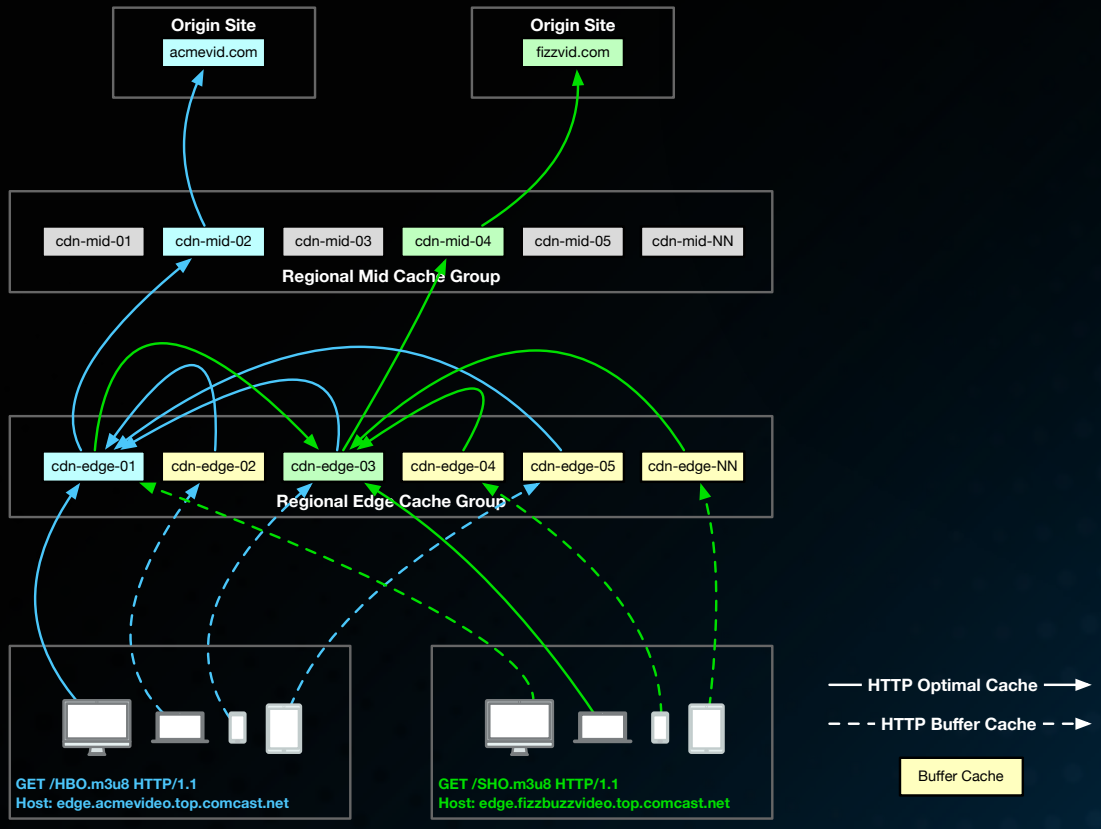
Multiple strategies can be used to select the correct origin, including round robin and consistent hash.

Note that errors detected with a parent may lead to it being marked down for a period of time. These errors are configurable and include HTTP errors that could be generated by any path. This is a key difference in approach from the Client Steering.



COMCAST

WIDE EDGE



The Wide Edge architecture allows an operator to sacrifice memory and network to utilize existing edge storage. This could work well for large libraries routed using DNS. Requires chaining ATS remap rules and disabling loop detection on the first.

COMCAST

RETROSPECTIVE

RETROSPECTIVE

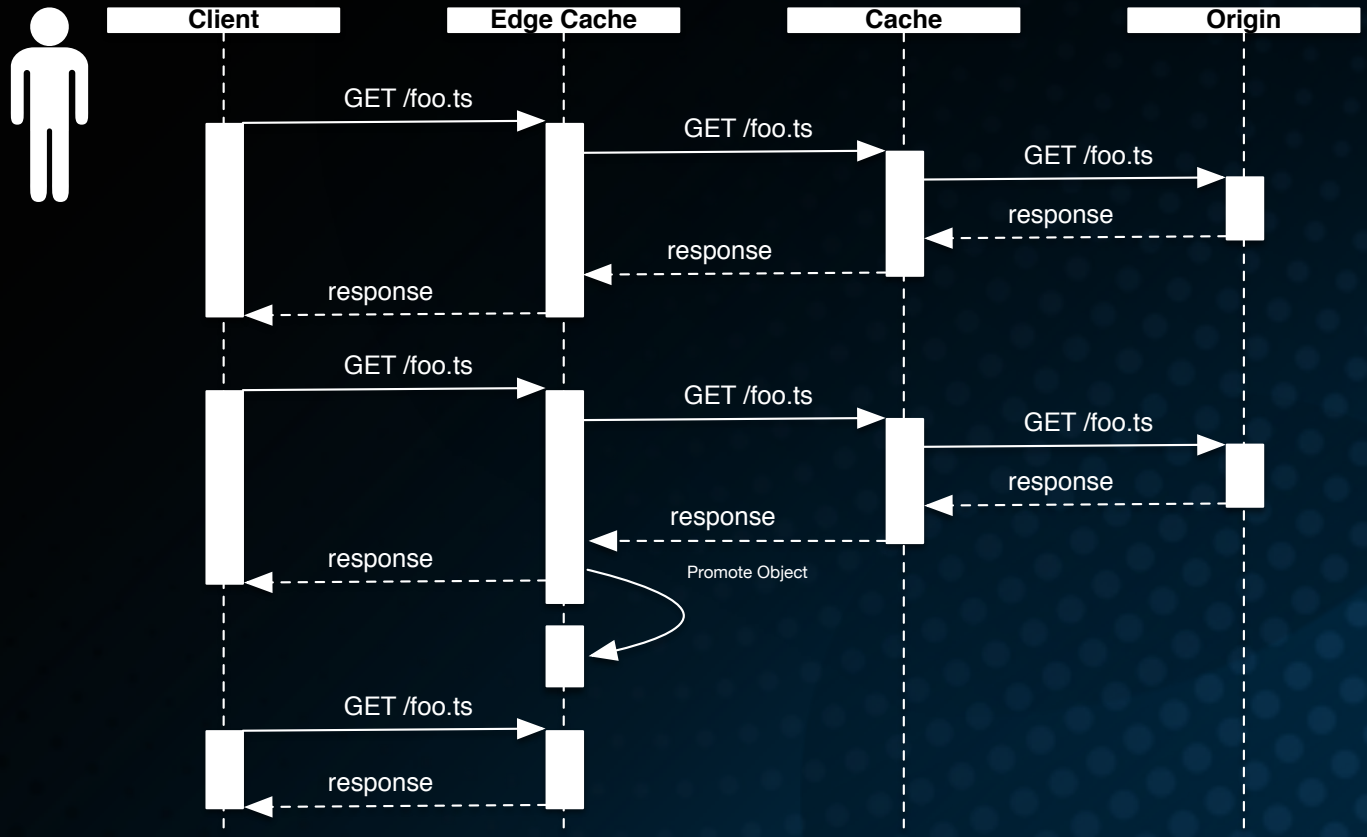
WHAT HAVE WE LEARNED

- DNS delivery services can lead to hot spots
  - Localization is critical, geolocation accuracy mixed
  - Overrides help but not exactly the go-to fix
  - EDNS0 client subnet extensions CDN-wide, not granular
  - Inefficient with large libraries without ***cache promote***

Lack of geolocation accuracy can lead to hot spots within a cache group based on default latitude and longitude settings, or the override settings if configured.

COMCAST

CACHE PROMOTE



COMCAST

RETROSPECTIVE

WHAT HAVE WE LEARNED

- DNS delivery services can lead to hot spots
  - Localization is critical, geolocation accuracy mixed
  - Overrides help but not exactly the go-to fix
  - EDNS0 client subnet extensions CDN-wide, not granular
  - Inefficient with large libraries without ***cache promote***

Lack of geolocation accuracy can lead to hot spots within a cache group based on default latitude and longitude settings, or the override settings if configured.

RETROSPECTIVE

WHAT HAVE WE LEARNED

- Health protocol sometimes too reactive, unsophisticated
- Traffic Monitor, while fast, has limits that must be solved
- Dispersion is static and must be dynamic
- Sophistication of client steering grows with experience
- Service isolation and limits increasingly important

Improving Traffic Monitor and killing the astats bug unlocks sophistication in our routing and caching strategies.

COMCAST

INTEGRATION

WHERE THE RUBBER MEETS THE ROAD

- Service architects find creative ways to break caching
  - Pattern-based consistent hash
  - Consistent hash with query params
  - Client steering forced diversity
  - Cache key and query params between tiers
- More flexibility within the caching hierarchy is needed

Flexibility and harmony between routing and the caching hierarchy is critical.

Integration concerns drive requirements as CDN and service architects solve redundancy and scale problems together.

SOLUTIONS

SOLUTIONS TO OUR PROBLEMS EXIST

- Complete the Perl migration
- Flexible cache groups
- Upstream parent selection diversity
- Distributed Traffic Monitor with fixed stats bug
- More granular health protocol, within TM and on the cache
- Large library DNS, wide-edge architecture wired into ATC
  - Unlocks improved dispersion
  - More options to apply service limits
  - Better accuracy

"Debugging is twice as hard as writing the code in the first place. Therefore, if you write the code as cleverly as possible, you are, by definition, not smart enough to debug it."

Kernighan's Law


COMCAST
APACHE
TRAFFIC CONTROL