# Building a large scale CDN with Apache Trafficserver

Jan van Doorn

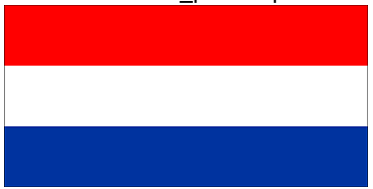
jan_vandoorn@cable.comcast.com

# About me

- Engineer at Comcast Cable
  - National Engineering & Technical Operations
    - NETO-VSS-CDNENG
  - Tech Lead for next generation CDN development
- Long time (Interactive) TV Geek
- Recovering Unix SysAdmin
  - Still can't help wanting to solve everything with Perl
- Colorado based but originally from The Netherlands

COMCAST
NE&TO
VIDEO SYSTEMS SERVICES

vi world_peace.pl

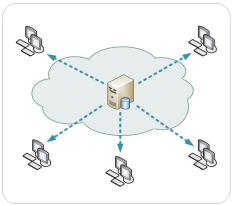


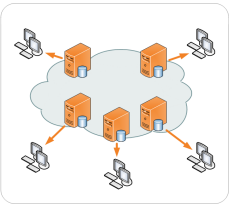
# Disclaimer stuff

Comcast uses CDNs in a number of ways but the CDN I will be discussing in this presentation relates primarily to how Comcast uses a CDN to deliver its IP cable services over its own network and not how Comcast uses a CDN to deliver Internet content.

# What is a CDN?

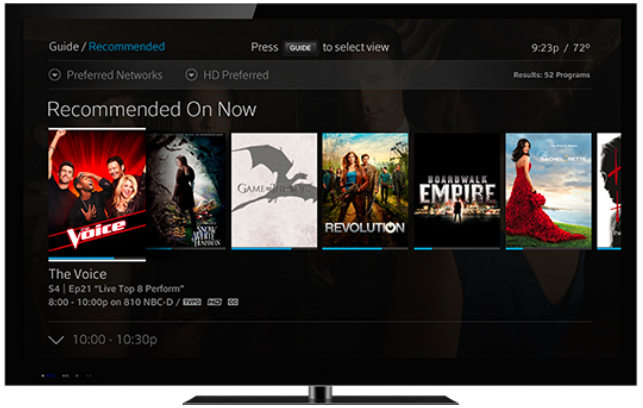
- Lots of Caches
  - The HTTP/1.1 compatible work horses in multiple tiers and edge locations
- Content Router
  - Get customer to best cache for his requested content in his location
- Health Protocol
  - A way to tell CR which caches are able to take work
- Management and Monitoring System
  - A way to manage a geographically disperse set of servers
- Reporting System
  - Log file analysis of edge, mid and CR contacts for (internal) billing and sizing

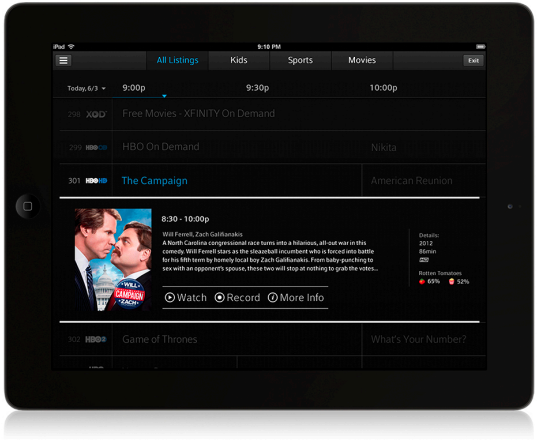




# Why does Comcast need one?

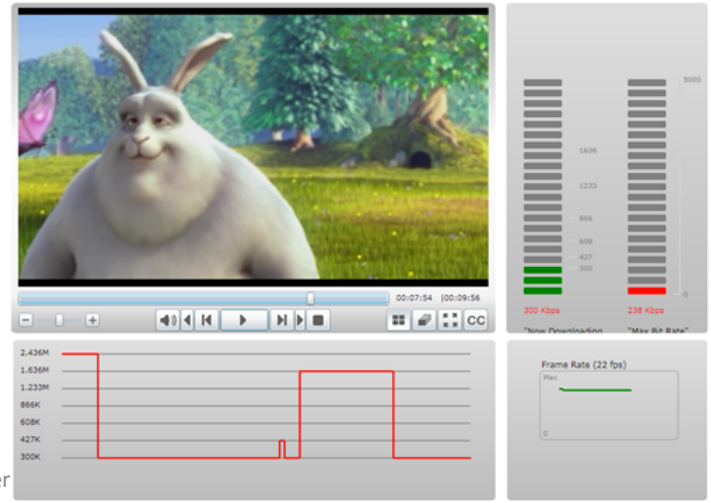
- X1 / X2 Cloud based User Interface
  - Images / UI elements
  - Code Downloads
- Next generation video distribution
  - Video on Demand (VoD)
  - Live Television
  - Cloud DVR
  - Second Screen (tablets, phones, PCs)
  - First Screen (big screen TV in living room)



# About Video and HTTP/1.1

- Adaptive BitRate (ABR) video delivery
  - Video gets “chopped up” in short chunks (2 – 6s)
  - Chunks are retrieved from server using HTTP
  - Adapts quality to resource availability and needs
  - Still video (high bandwidth)
  - Highly cacheable
  - HTTP KeepAlive
  - Live Television
  - VoD / cDVR

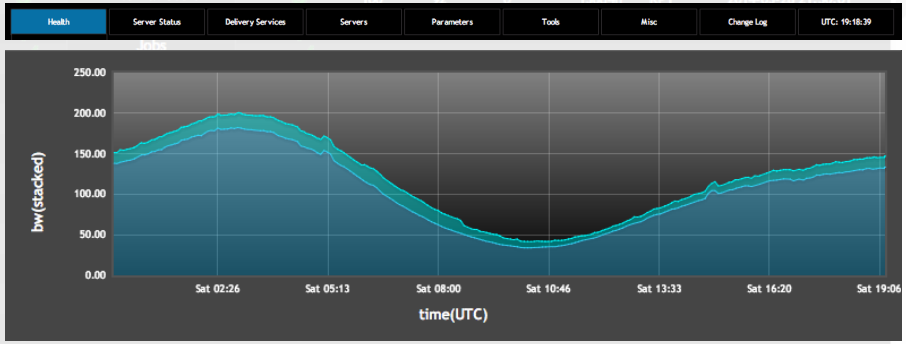


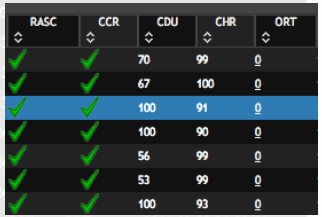
# The Comcast CDN Design Principles

- Open standards based
- No vendor or system lock-in
- Cost effective
- All customer facing parts are IPv6 and IPv4
- Horizontally scalable
- Well suited for ABR video, but not exclusively for video
- Loosely coupled components, stateless
- 100% availability, handle component failure gracefully
- Maintenance should be part of normal life
- Simple

# The Comcast CDN

- The Caches
  - Apache Traffic Server
    - more on that later
- Content Router
  - Built in-house - Apache Tomcat application
    - more on that later
- Health Protocol
  - Built in-house - Apache Tomcat application
    Basically an aggregator of enhanced stats plugin in Trafficserver
- Management and Monitoring System
  - Built in-house
    - Perl / Mojolicious framework against Pg or MySQL db
    - jQuery UI
- Reporting System
  - The only thing we bought (Splunk)



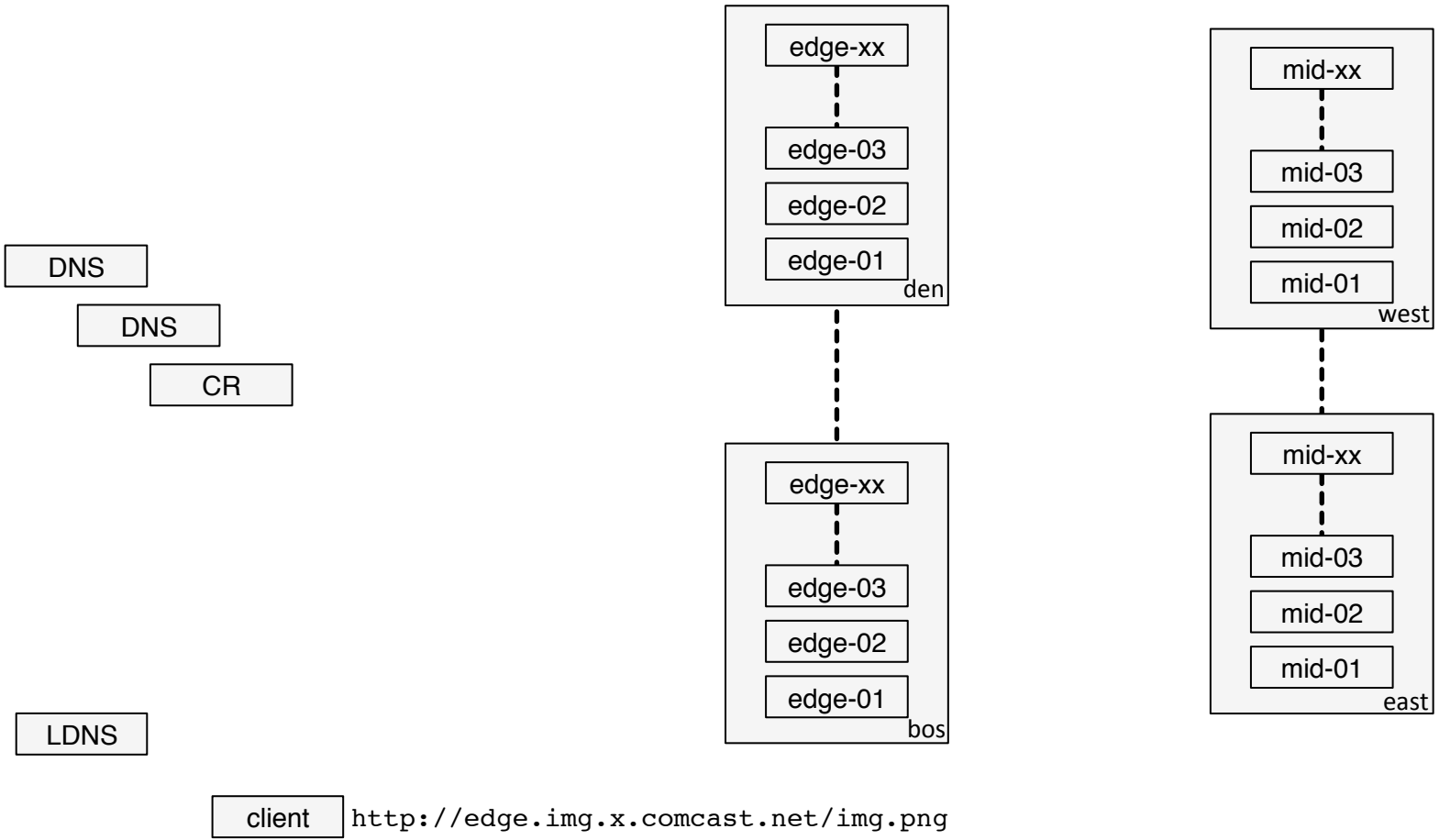



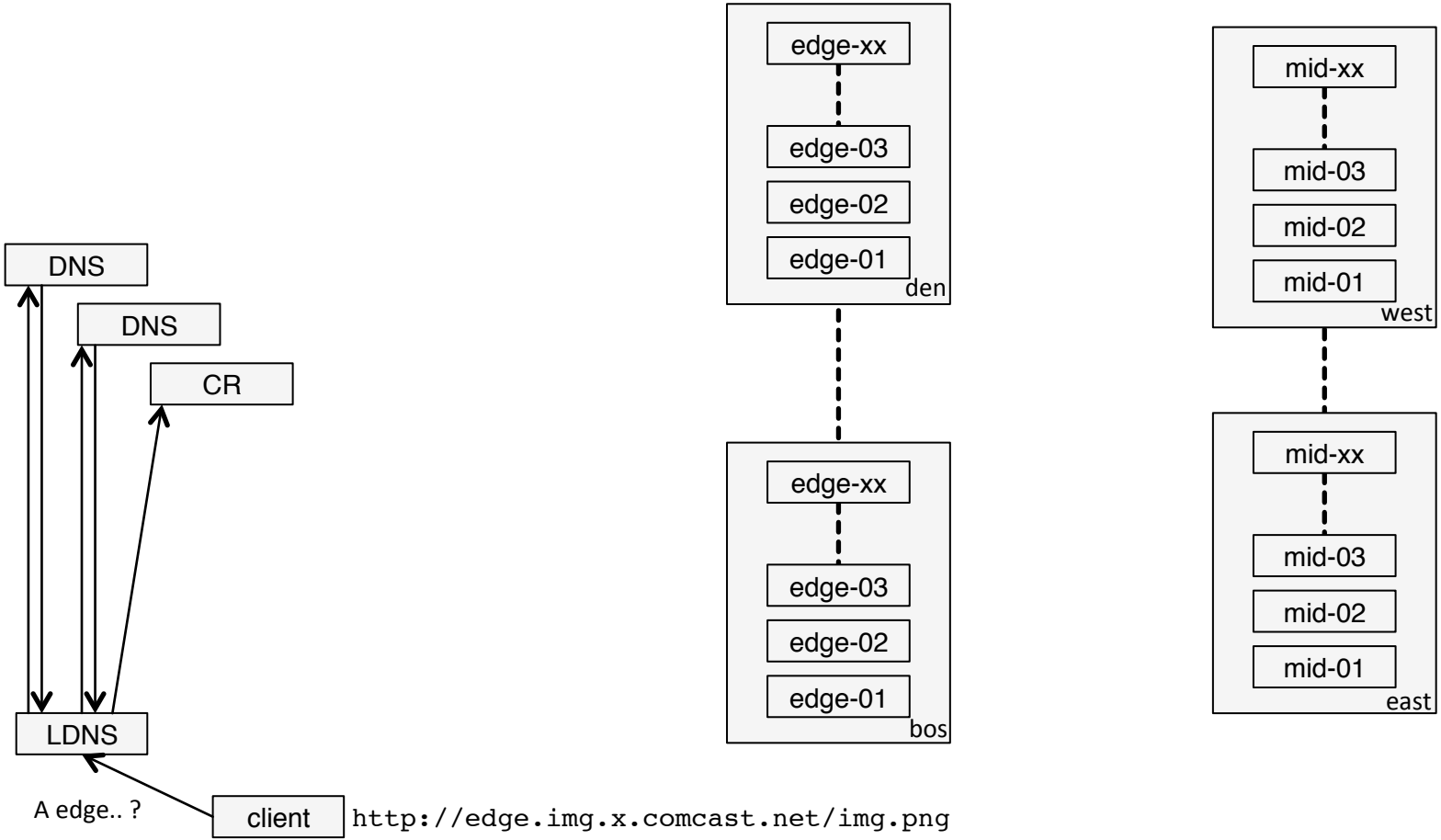
# About Content Routing

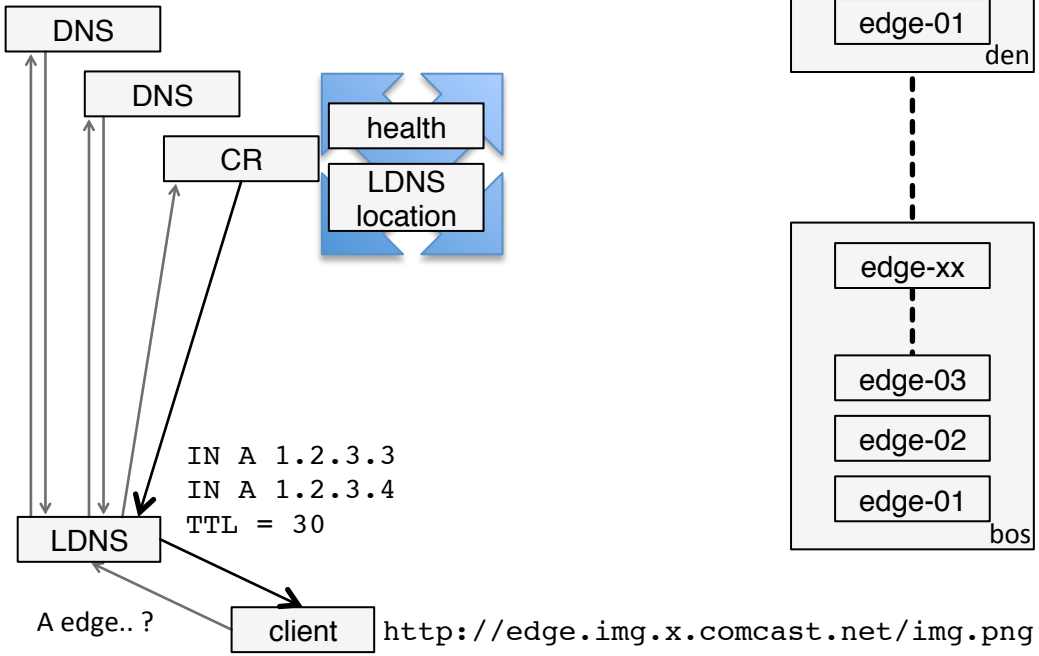
- Goal is to get the client to the best cache *for the content requested and the location it is requested from*
  - Distance / network hops
  - Network link Quality / speed
  - Availability of content in cache

# DNS content routing

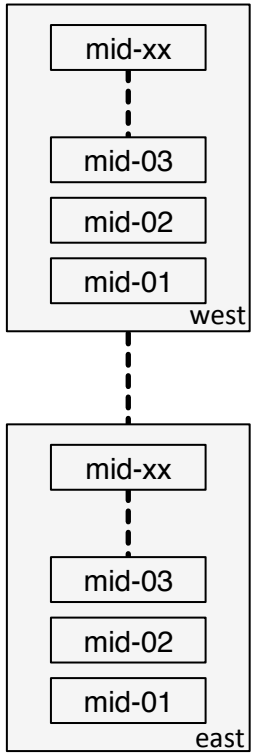
- Content Router is DNS Auth
- CR makes decision based on resolver, not based on client
- CR only knows the hostname
  - Unaware of path in URL, HTTP headers, query string, etc
- Fast
  - Usually used for getting web-page objects, images, etc

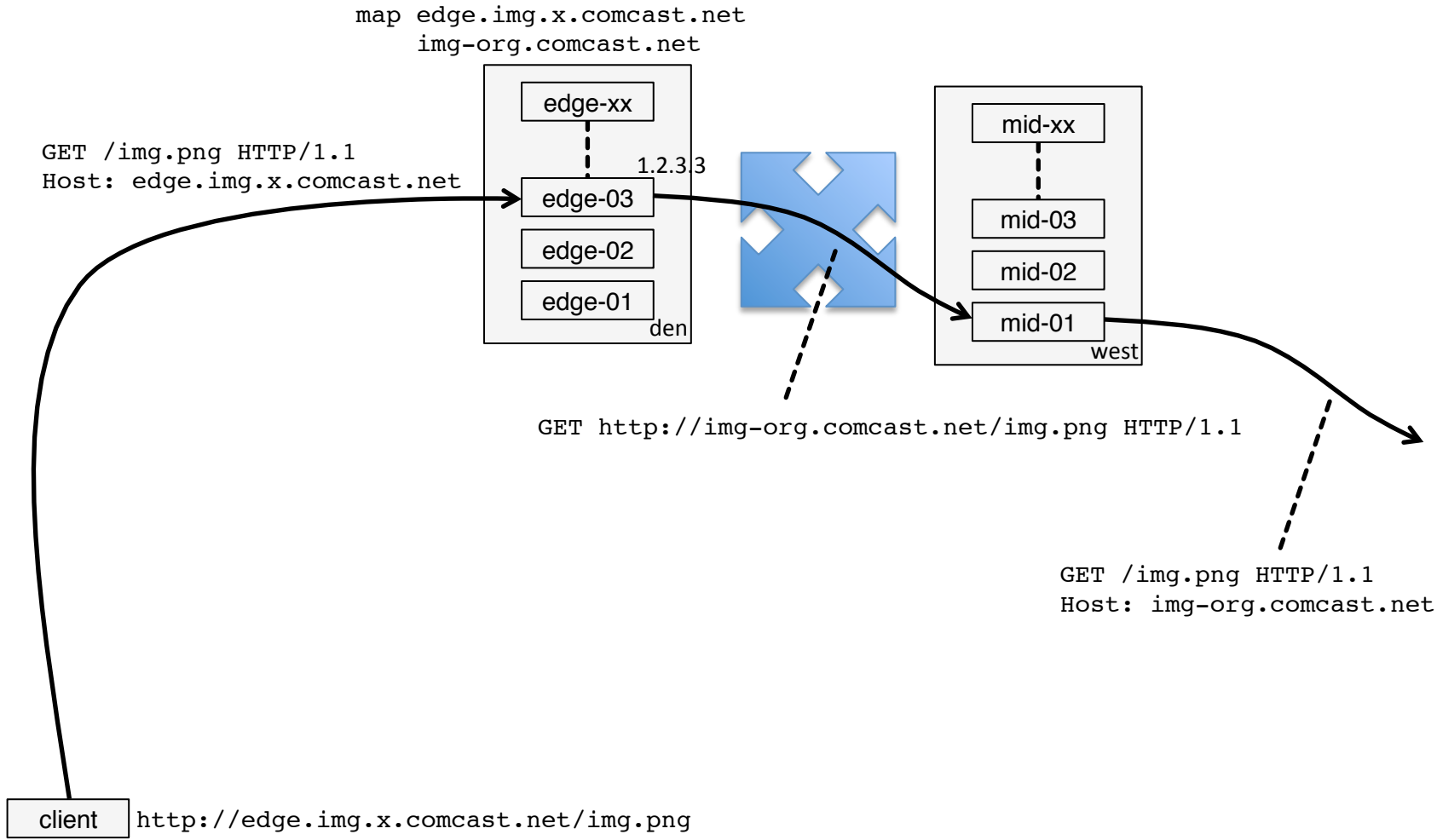

DNS
DNS
CR
LDNS
edge-xx
edge-03
edge-02
edge-01
den
edge-xx
edge-03
edge-02
edge-01
bos
mid-xx
mid-03
mid-02
mid-01
west
mid-xx
mid-03
mid-02
mid-01
east
client
http://edge.img.x.comcast.net/img.png

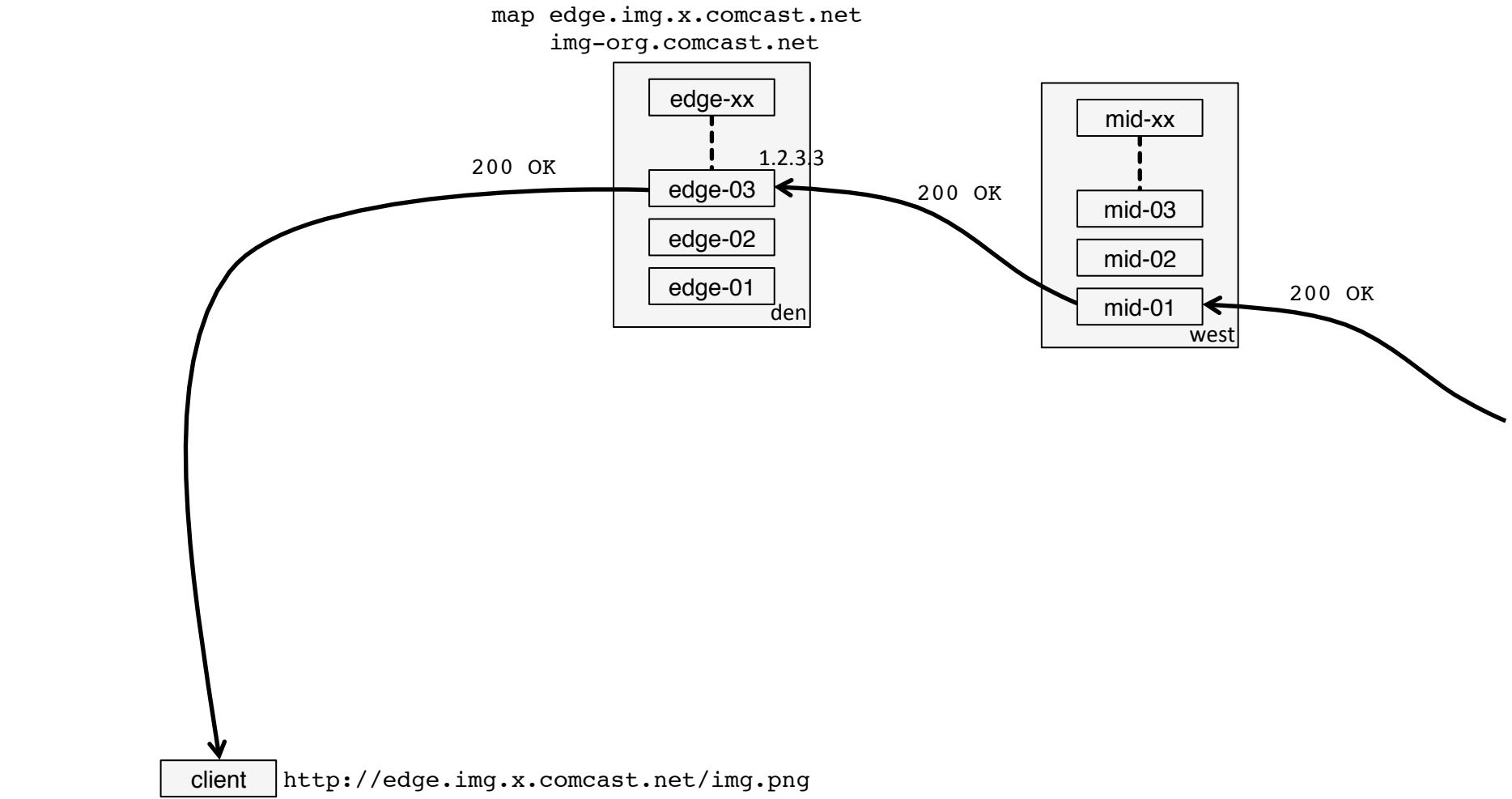

edge-xx
edge-03
edge-02
edge-01
den
mid-xx
mid-03
mid-02
mid-01
west
edge-xx
edge-03
edge-02
edge-01
bos
mid-xx
mid-03
mid-02
mid-01
east
DNS
DNS
CR
LDNS
A edge.. ?
client
http://edge.img.x.comcast.net/img.png


DNS
DNS
CR
health
LDNS location
IN A 1.2.3.3
IN A 1.2.3.4
TTL = 30
LDNS
A edge.. ?
client
http://edge.img.x.comcast.net/img.png


edge-xx
1.2.3.3
edge-03
edge-02
edge-01
den
edge-xx
edge-03
edge-02
edge-01
bos

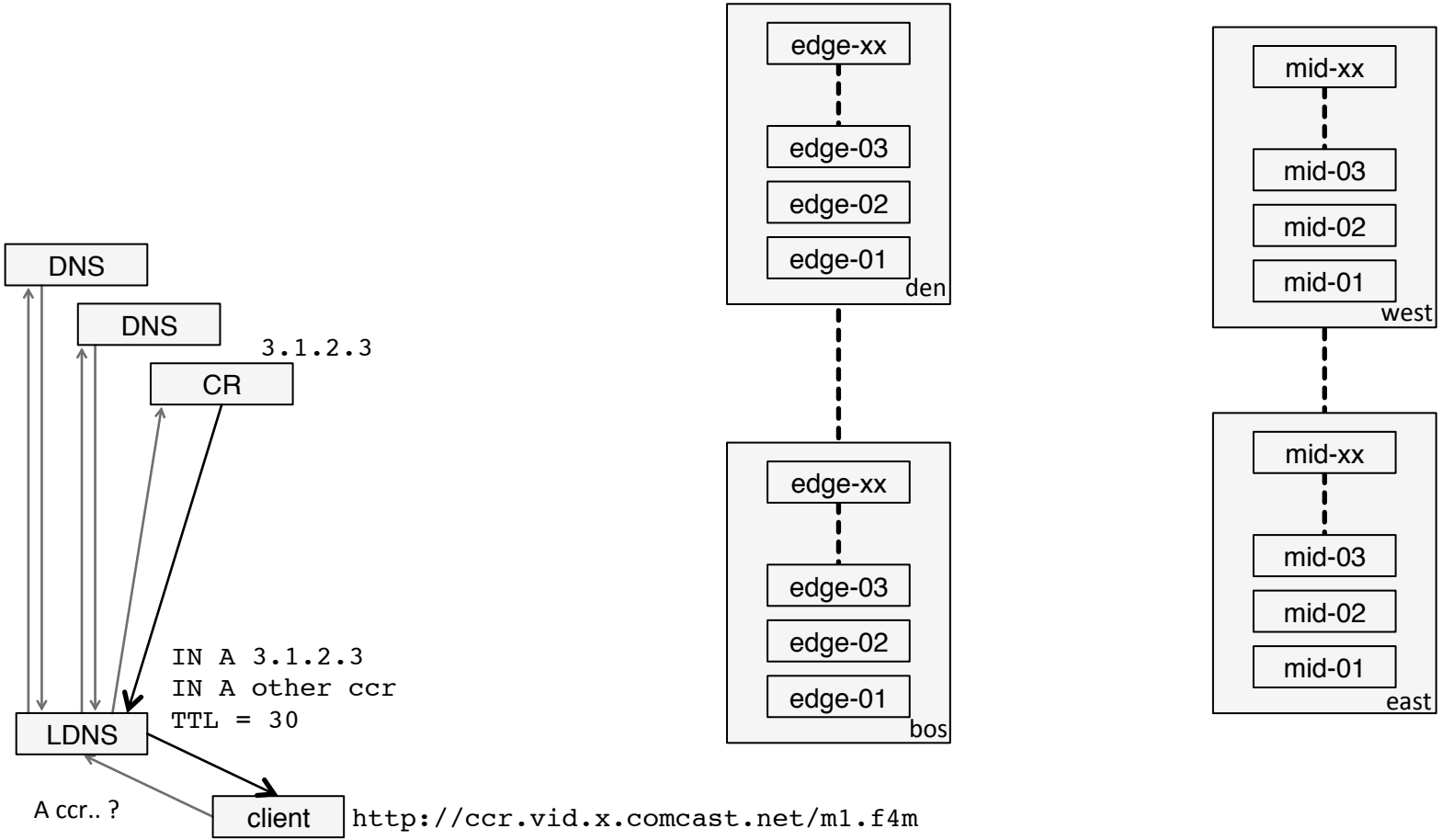

mid-xx
mid-03
mid-02
mid-01
west
mid-xx
mid-03
mid-02
mid-01
east

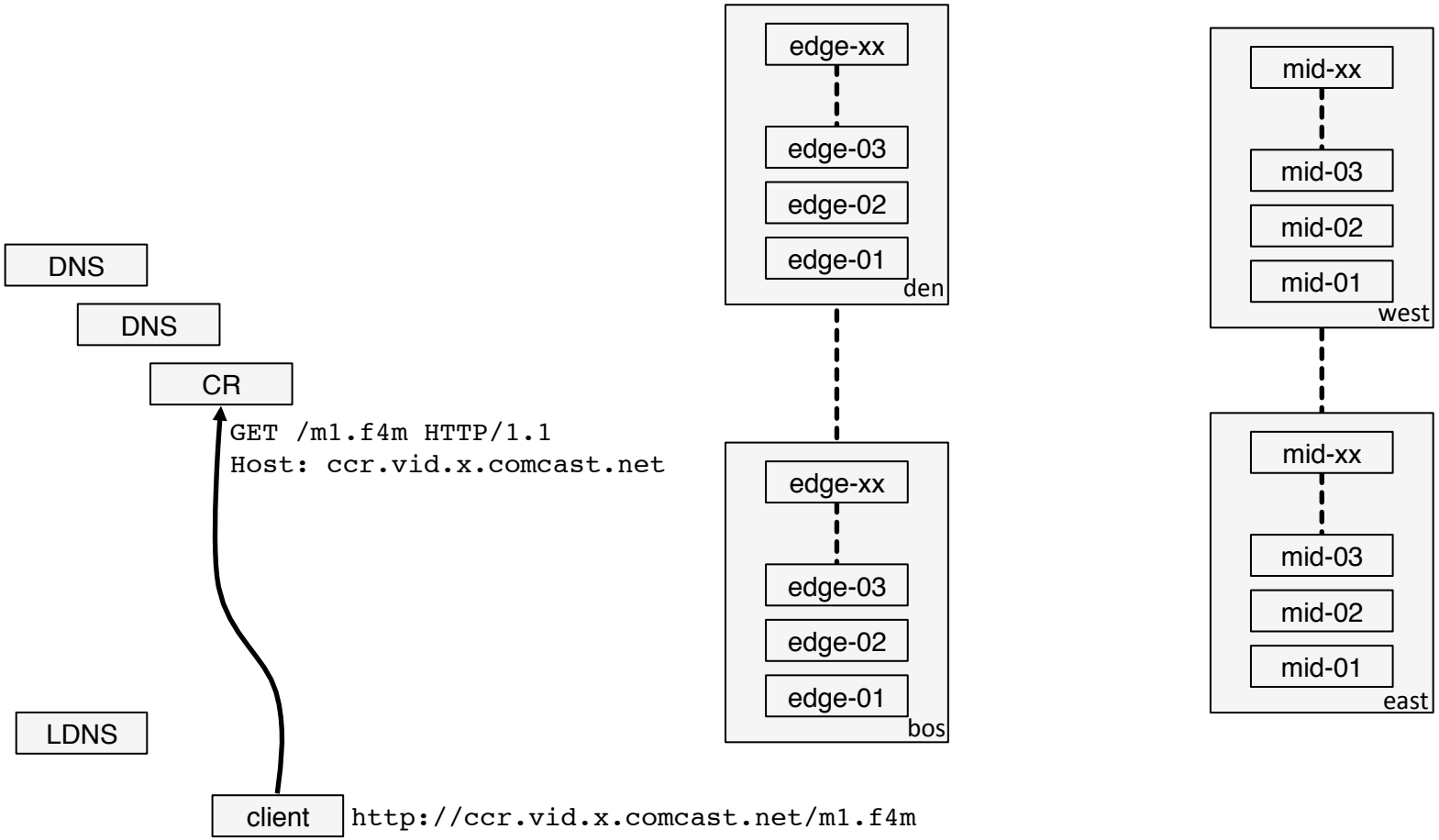

map edge.img.x.comcast.net
img-org.comcast.net
edge-xx
1.2.3.3
edge-03
edge-02
edge-01
den
mid-xx
mid-03
mid-02
mid-01
west
GET /img.png HTTP/1.1
Host: edge.img.x.comcast.net
GET http://img-org.comcast.net/img.png HTTP/1.1
GET /img.png HTTP/1.1
Host: img-org.comcast.net
client
http://edge.img.x.comcast.net/img.png

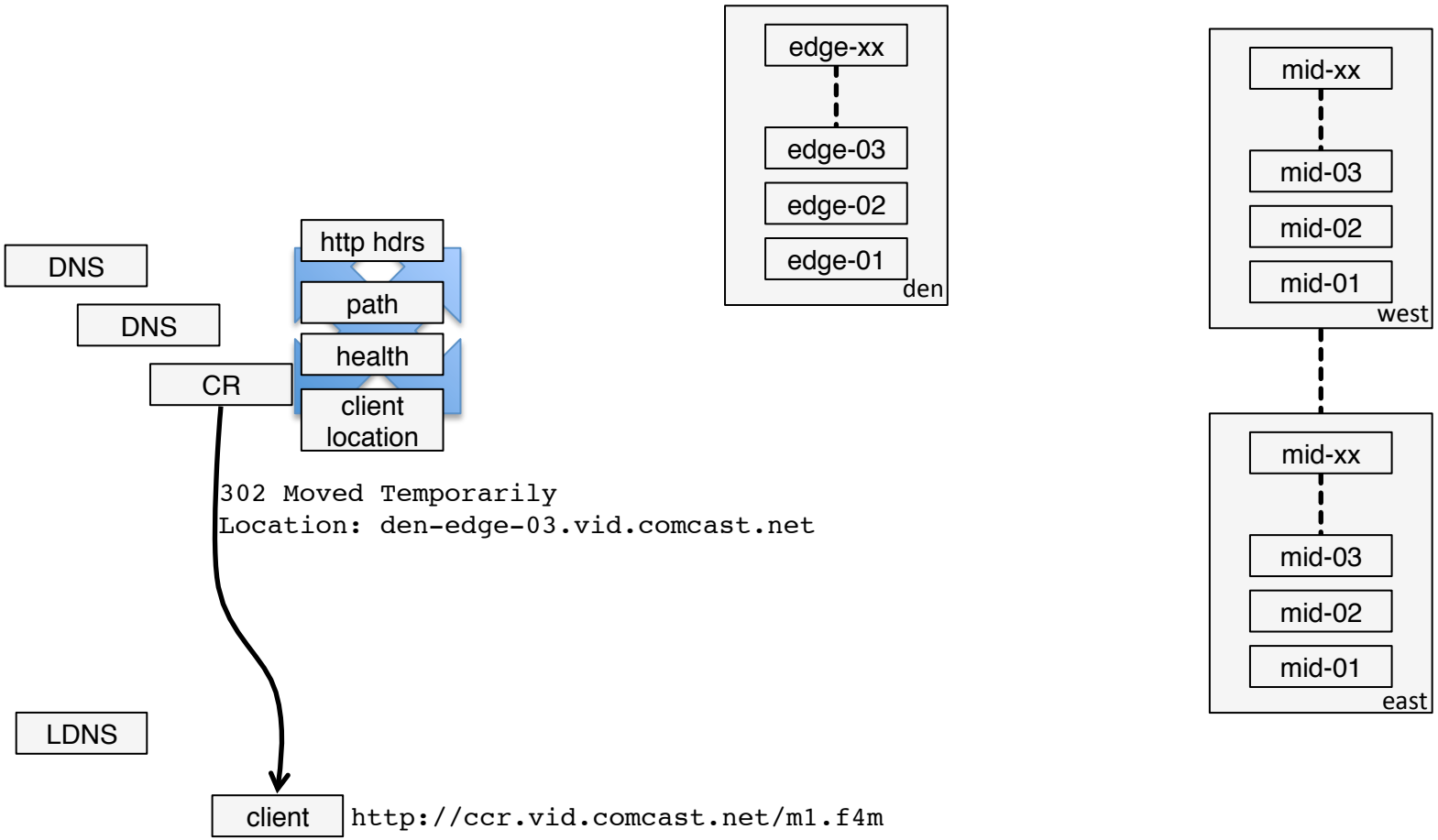

map edge.img.x.comcast.net
img-org.comcast.net
edge-xx
1.2.3.3
edge-03
edge-02
edge-01
den
mid-xx
mid-03
mid-02
mid-01
west
200 OK
200 OK
200 OK
client
http://edge.img.x.comcast.net/img.png

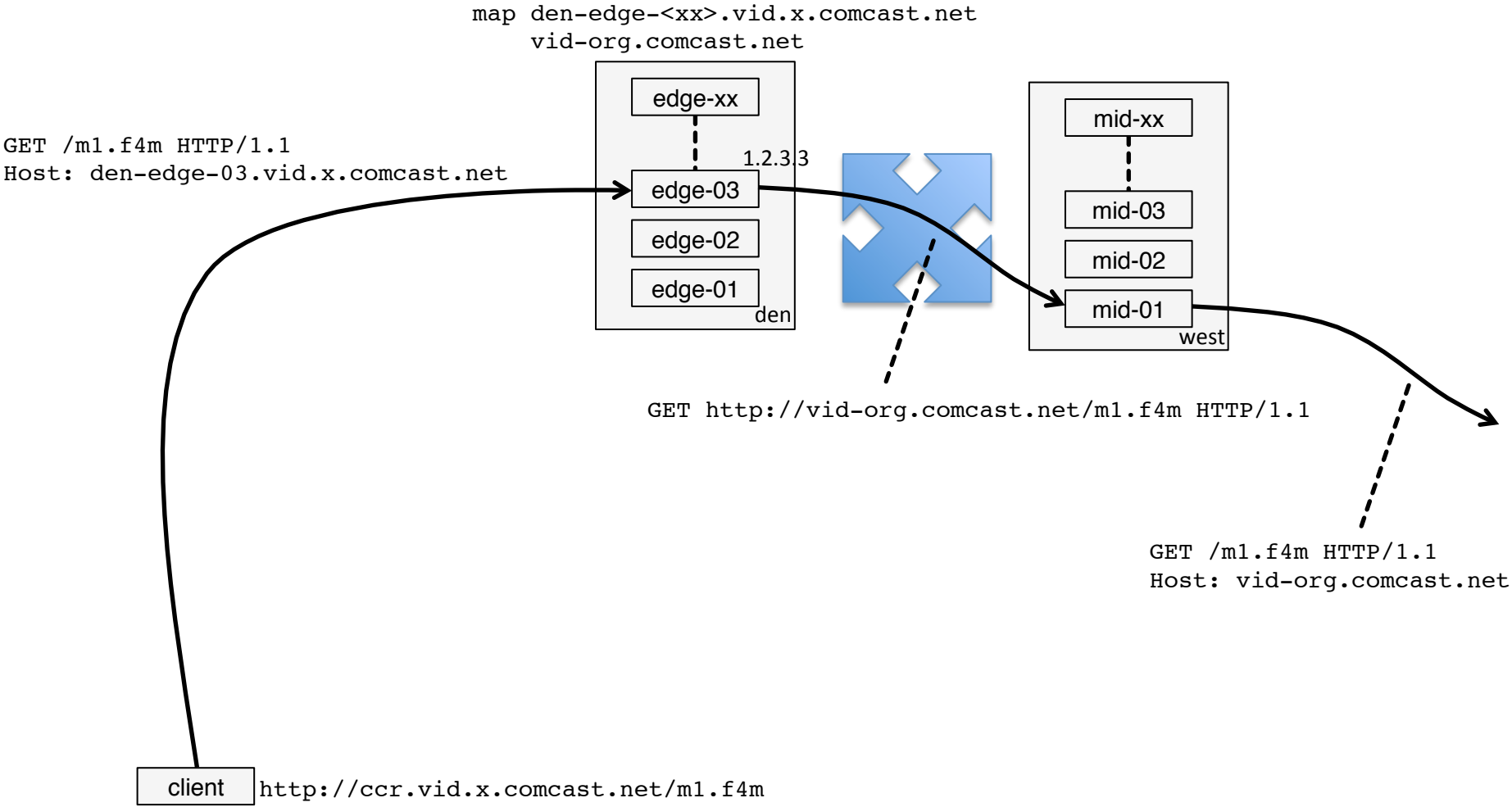
# HTTP Content Routing

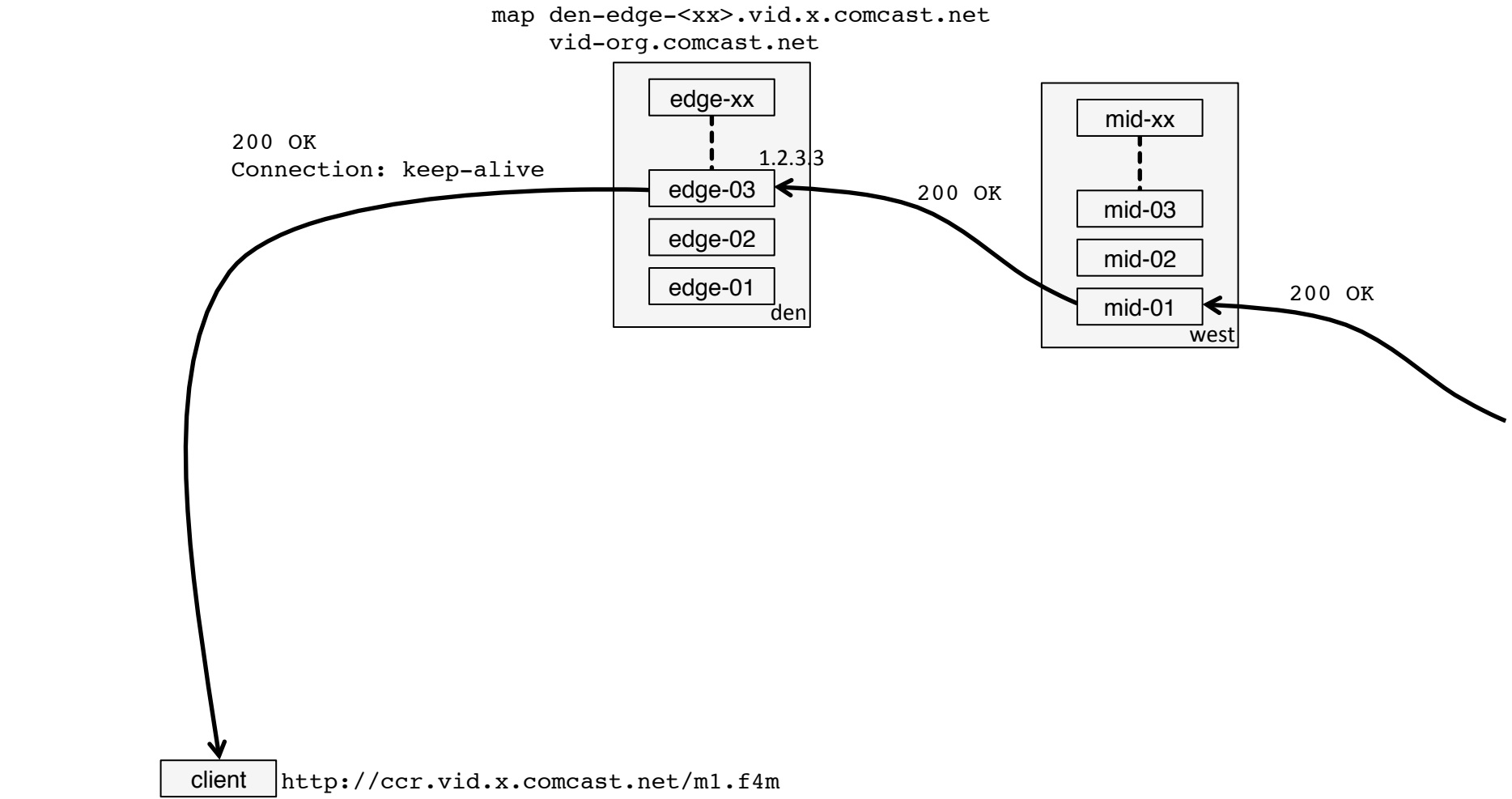
- CR is DNS auth, but always responds with own IP address to DNS query
- Client then does HTTP connection to CR
  - CR now knows all the HTTP stuff
  - CR now also knows client IP address
- Slower, but much more “precise”; usually used for longer sessions, like ABR video sessions


DNS
DNS
3.1.2.3
CR
IN A 3.1.2.3
IN A other ccr
TTL = 30
LDNS
A ccr.. ?
client
http://ccr.vid.x.comcast.net/m1.f4m
edge-xx
edge-03
edge-02
edge-01
den
edge-xx
edge-03
edge-02
edge-01
bos
mid-xx
mid-03
mid-02
mid-01
west
mid-xx
mid-03
mid-02
mid-01
east


edge-xx
edge-03
edge-02
edge-01
den
mid-xx
mid-03
mid-02
mid-01
west
DNS
DNS
CR
GET /m1.f4m HTTP/1.1
Host: ccr.vid.x.comcast.net
edge-xx
edge-03
edge-02
edge-01
bos
mid-xx
mid-03
mid-02
mid-01
east
LDNS
client
http://ccr.vid.x.comcast.net/m1.f4m


edge-xx
edge-03
edge-02
edge-01
den
mid-xx
mid-03
mid-02
mid-01
west
mid-xx
mid-03
mid-02
mid-01
east
DNS
DNS
CR
http hdrs
path
health
client location
302 Moved Temporarily
Location: den-edge-03.vid.comcast.net
LDNS
client
http://ccr.vid.comcast.net/m1.f4m


map den-edge-<xx>.vid.x.comcast.net
vid-org.comcast.net
edge-xx
1.2.3.3
edge-03
edge-02
edge-01
den
mid-xx
mid-03
mid-02
mid-01
west
GET /m1.f4m HTTP/1.1
Host: den-edge-03.vid.x.comcast.net
GET http://vid-org.comcast.net/m1.f4m HTTP/1.1
GET /m1.f4m HTTP/1.1
Host: vid-org.comcast.net
client
http://ccr.vid.x.comcast.net/m1.f4m


map den-edge-<xx>.vid.x.comcast.net
vid-org.comcast.net
edge-xx
1.2.3.3
edge-03
edge-02
edge-01
den
mid-xx
mid-03
mid-02
mid-01
west
200 OK
200 OK
200 OK
Connection: keep-alive
client
http://ccr.vid.x.comcast.net/m1.f4m

# Why Trafficserver

- Any HTTP 1.1 Compliant cache will work
- We chose Apache Traffic Server (ATS)
  - Top Level Apache project (NOT httpd!)
  - Extremely scalable and proven
  - Very good with our VoD load
  - Efficient storage subsystem uses raw disks
  - Extensible through plugin API
  - Vibrant and friendly development community
  - Added handful of plugins for specific use cases

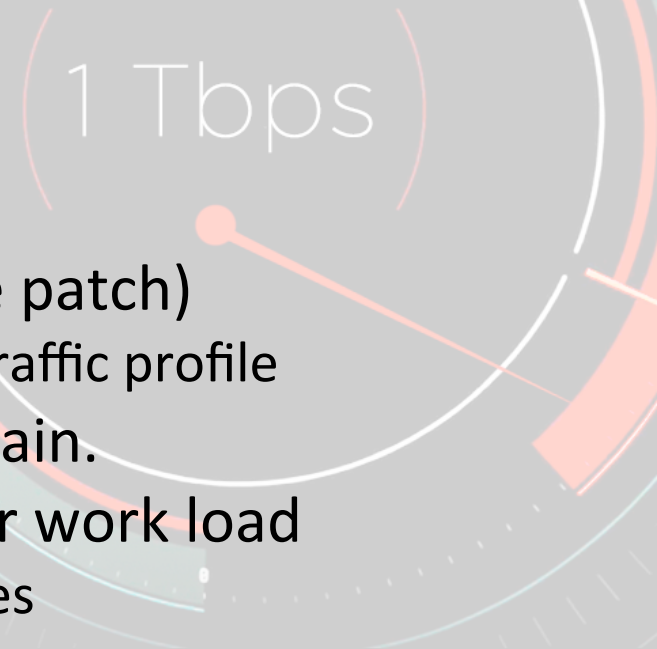
# Our Gen1 cache

- Off the shelf hardware - ride Moore's Law!
- Spinning disks (!)
  - 24 900Gb SAS disks for caching
  - 2 mirrored OS drives

- 192 Gbyte of memory, for live TV
- 1x10GE initially, 2x10GE upgrades being planned
- Connected to Aggregation Routers (first server to do so)
- Linux CentOS 6.1 / 6.2

# Trafficserver performance

- Tested very well without major tweaks
  - Pushing 10 Gbps on Gen1 box with very wide VoD like dispersion
  - Disk util is even and at almost 100%
- Using traffic separation feature (volume patch)
  - up to 40 Gbps per server with a realistic traffic profile
- Not sure if we want more… Failure domain.
- Published tests seem to not apply to our work load
  - Ended up writing many test tools ourselves

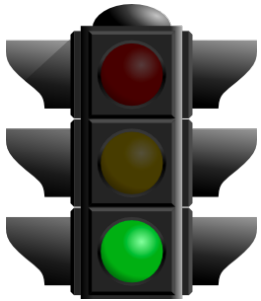


# Open Source and Support

- Not having a support number is scary!
  - Most Open Source projects now have third parties selling support… but we're flying solo on that as well
  - Often more FUD than actual rational reasons
- The active community is really important here
- DevOps model
- DIY surgical patches for your problem are usually much faster than a release (from either a vendor, or an Open Source project)
- Get someone on staff to become part of the Open Source community

# Current status

- Serving ~ 4 TByte / day (!)
- Over 250 caches deployed and serving traffic
  - Over 5 PByte total storage capacity
  - each has single 10GE (starting to upgrade to 20GE)
- 25 edge cache groups (“clusters”)
  ~ 1.7 Tbps total edge capacity
  ~ 320 Gbps served at highest peak
  ~ 220 Gbps daily peak
- 3 mid-tier cache groups (“clusters”)
- Origin off load > 85%
- All IPv6 / IPv4
  - client decides based on connectivity, CDN is IPv6 all the way



# Future Plans

- Double size of CDN in 2014
- Lots of “first screen service” additions
- New Mid-Tier cache(4RU, 512G RAM, 288TB disk)
- Next Gen Edge Tier (Still defining, probably SSD)
- Dedicated “Live TV” caches
- Deeper deployment
- Better tools for Operations

# Traffic Server wish list

- Improved Operations
  - More traffic_line –x, less restart
    - Should be able to change any config without having to restart
  - Better stability
- Configuration flexibility
  - Dare I mention VCL?

Questions?

# Building a large scale CDN with Apache Trafficserver

Jan van Doorn

jan_vandoorn@cable.comcast.com