How Comcast Built An Open Source Content Delivery Network

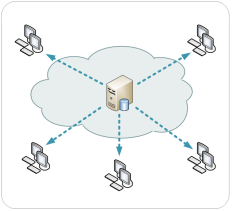
National Engineering
& Technical Operations

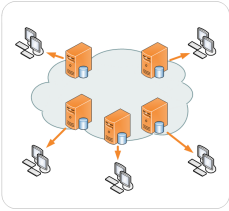
nēte

Jan van Doorn
Distinguished Engineer
VSS CDN Engineering

Comcast

What is a CDN?





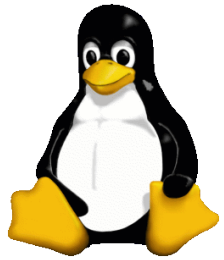
- **Content Router**
  - get customer to best cache for his requested content in his location
- **Health Protocol**
  - a way to tell CR what caches are able to take work
- **Hundreds of Caches**
  - the HTTP/1.1 compatible work horses in multiple tiers and edge locations
- **Management and Monitoring System**
  - a way to manage a geographically disperse set of servers
- **Reporting System**
  - log file analysis of edge, mid and CR contacts

Our Overall Design Goals

- Open standards based
- No vendor or system lock-in
- Cost effective
- All customer facing parts are IPv6 and IPv4
- Horizontally scalable
- Optimized for video, but not exclusively for video
- Loosely coupled components, stateless
- 100% availability, handle component failure gracefully
- Simple

The Caches – Hardware & OS

- Off the shelf hardware - ride Moore's Law!
- Spinning disks (!)
  - 24 900Gb SATA disks for caching
  - 2 mirrored OS drives
- Lots of memory, for linear “TV”
- 1x10GE initially, 2x10GE upgrades being planned
- Connected to Aggregation Routers (first application to do so)
- Linux CentOS 6.1 / 6.2

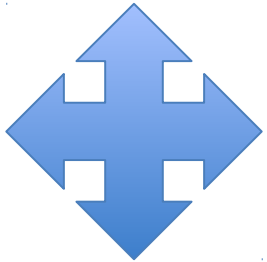


The Caches - Software

- 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 development community
  - Added handful of plugins for specific use cases

The Comcast Content Router (CCR)

- Tomcat Java application built in-house
- Multiple VMs around the country in DNS Round Robin
- Routes “by” DNS, HTTP 302, or REST
- Can route based on:
  - Regexp on URL host name (DNS and HTTP 302 redirect)
  - Regexp on URL Path and headers (HTTP 302 redirect)
  - Client location
    - Coverage Zone File from network
    - Geo IP lookup
  - Edge cache health
  - Edge cache load

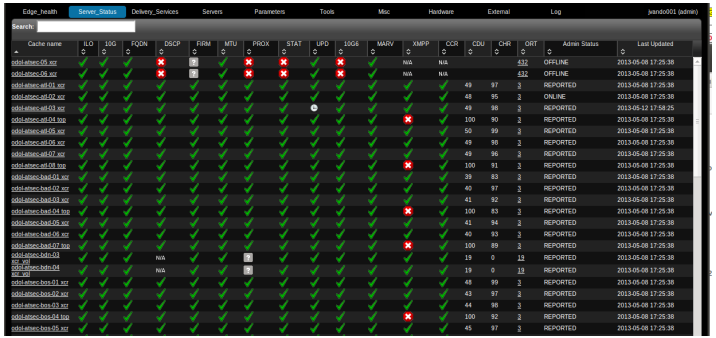


The Health Protocol – Rascal Server

- Tomcat Java application built in-house
- HTTP GETs vital stats from each cache every 5 seconds
  - Modified stats_over_http plugin on caches exposes app and system stats
- Determines and exposes state of caches to CRs
- Allows for real time monitoring / graphing of CDN stats
- Exposes 5 min avg/min/max to NE&TO Service Performance Database
- Redundant by having 2 instances running independent of each other.
  - CRs pick one randomly

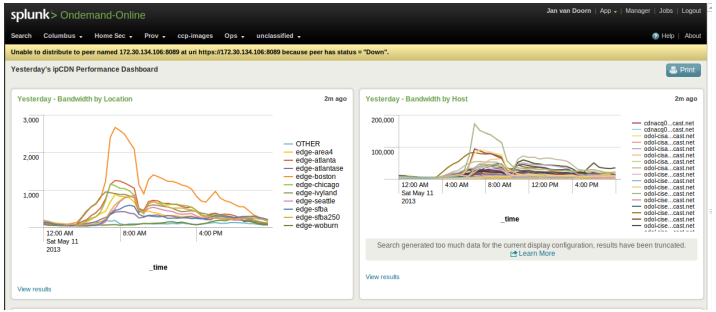
Configuration Management

- Twelve Monkeys tool built in-house
- Web based jQuery UI
- Mojolicious Perl framework
- MySQL database
- REST interfaces
- Integrated into standard Ops methods and best practices from day one
- Monitoring from Health Protocol through Rascal server



Log files and Reporting

- Splunk>
- The only commercial product we used
- Well defined interfaces
  No vendor lock-in possible
- Easy to get started, may move to Open Source on this later
- ipCDN usage metrics by delivery service



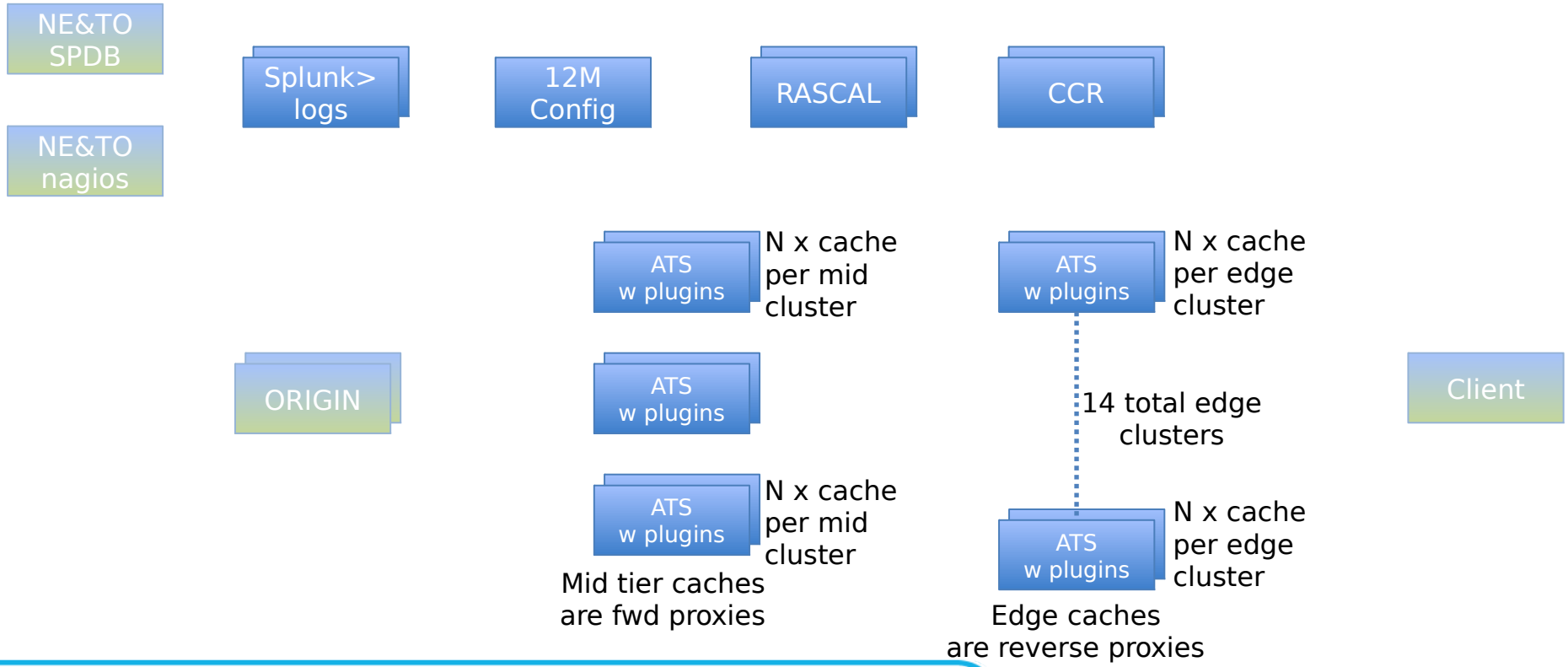
Choosing Open Source Components

- License
- Functionality
- Performance
- Plugin architecture / flexibility
- Development Community
  - Vibrant, active project
  - Friendly / Helpful

What About Support?

- Most Open Source projects now have third parties selling support
- The active community is really important here
- Keep dev close to ops (DevOps model)
- DIY surgical patches for your problem are usually much faster than a release (from either a vendor, or a Open Source project)
- Get someone on staff to become part of the Open Source community

Comcast ipCDN



Comcast ipCDN Summary

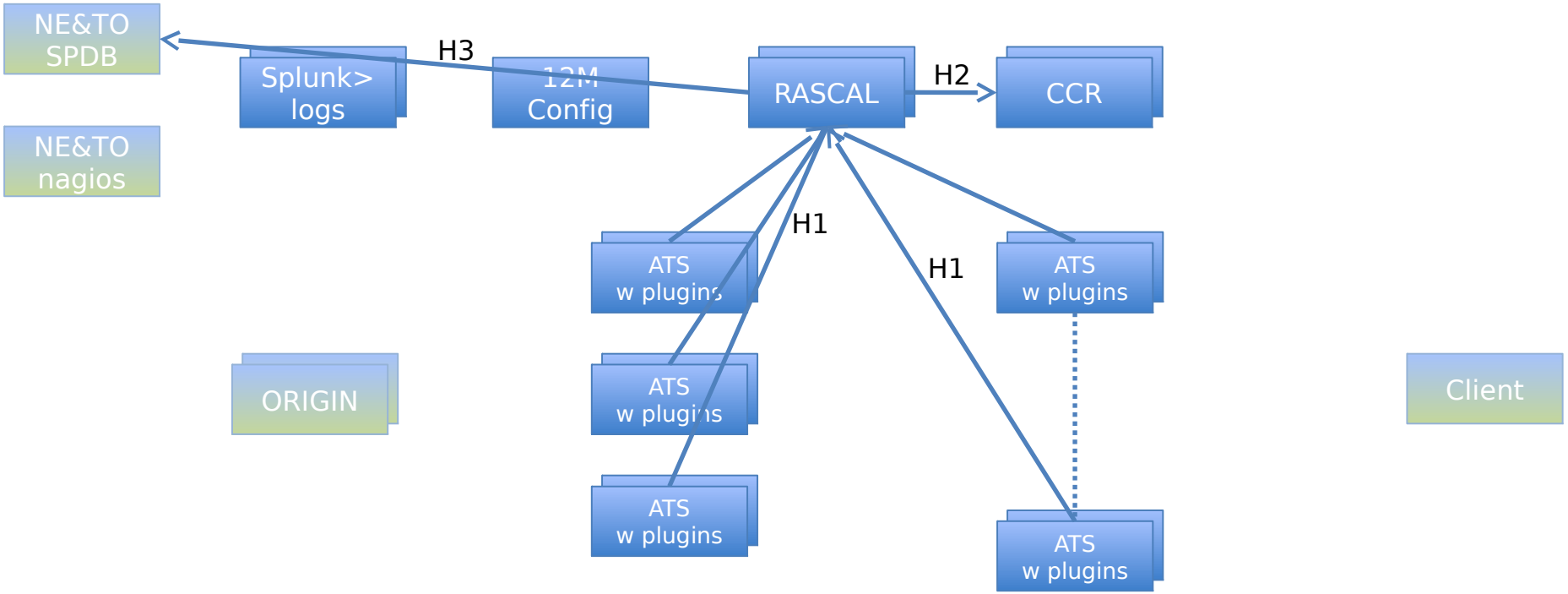
- **C**omcast **C**ontent **R**outer
  - Stateless
  - DNS Round Robin
- **Rascal** Health Monitoring
- **12 Monkeys** Configuration Management
- **Splunk** Logging Collection and Analytics

- **ATS** Caches in hierarchy
  - 3 Mid Tier locations
  - 14 Edge Tier locations
  - 7 – 12 Caches / location
  - 22TB disk / cache
  - No ATS clustering, content affinity by url hash
  - Small number of plugins

National Engineering
& Technical Operations

Comcast

Comcast ipCDN - Health

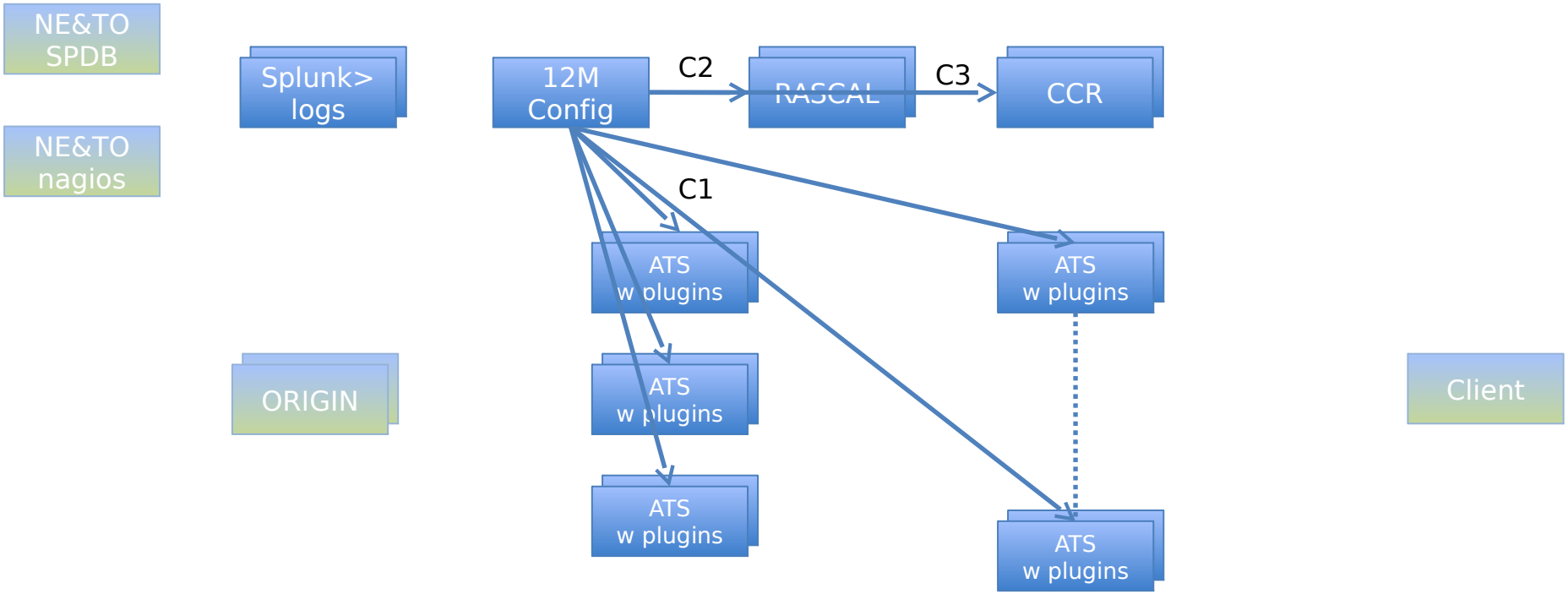


Comcast ipCDN - Health

- **H1**: Rascal does HTTP GET for key application and system stats from an ATS plugin every 5 second
- **H2**: CCR does HTTP GET of combined edge status every 1 second
- **H3**: SPDB front-end pulls avg/min/max of certain key stats into SPDB by HTTP GET every 5 mins

  Rascal is redundant by having 2 separate instances of it running completely independent of each other

Comcast ipCDN - Config

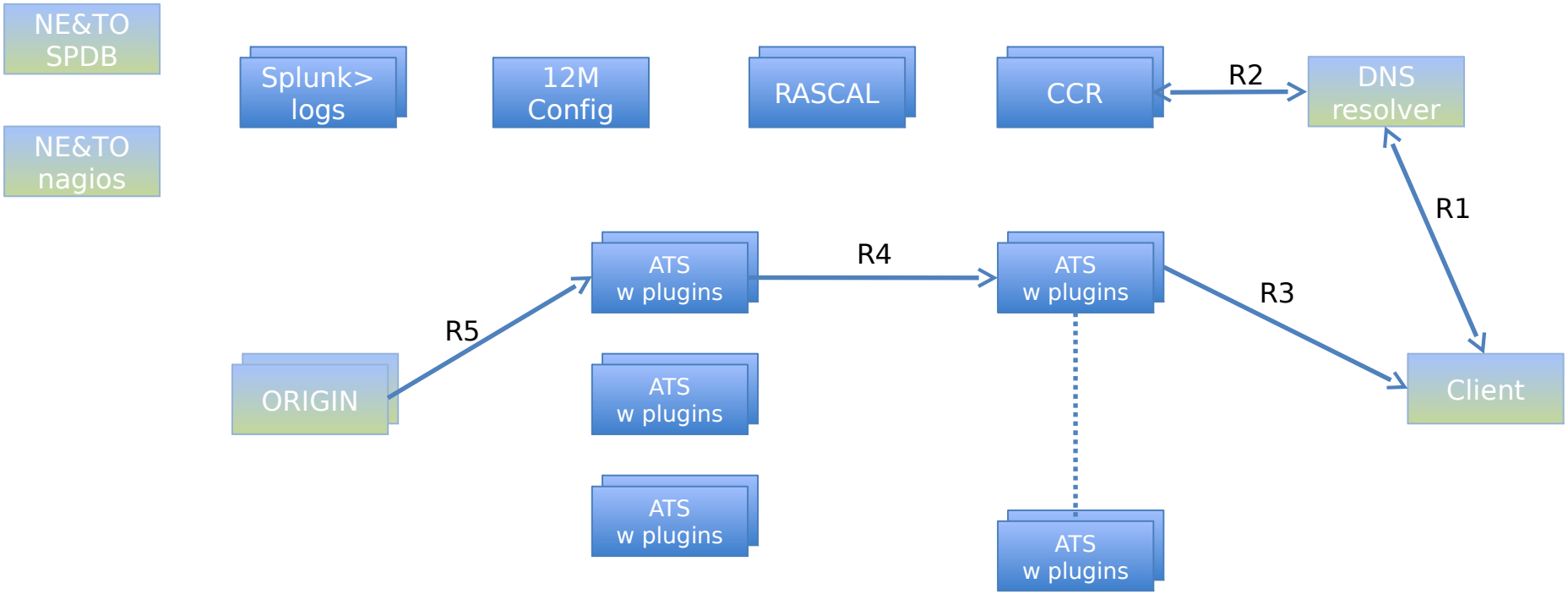


Comcast ipCDN - Config

- **C1**: Each Cache does a HTTP GET for it's configuration every 15 minutes; auto-applies “safe” changes
- **C2**: Each Rascal does a HTTP GET for it's configuration every X seconds; auto-applies
- **C3**: Each CCR does a HTTP GET
  for it's configuration every X seconds; auto-applies.

  12 Monkeys is not redundant at this time

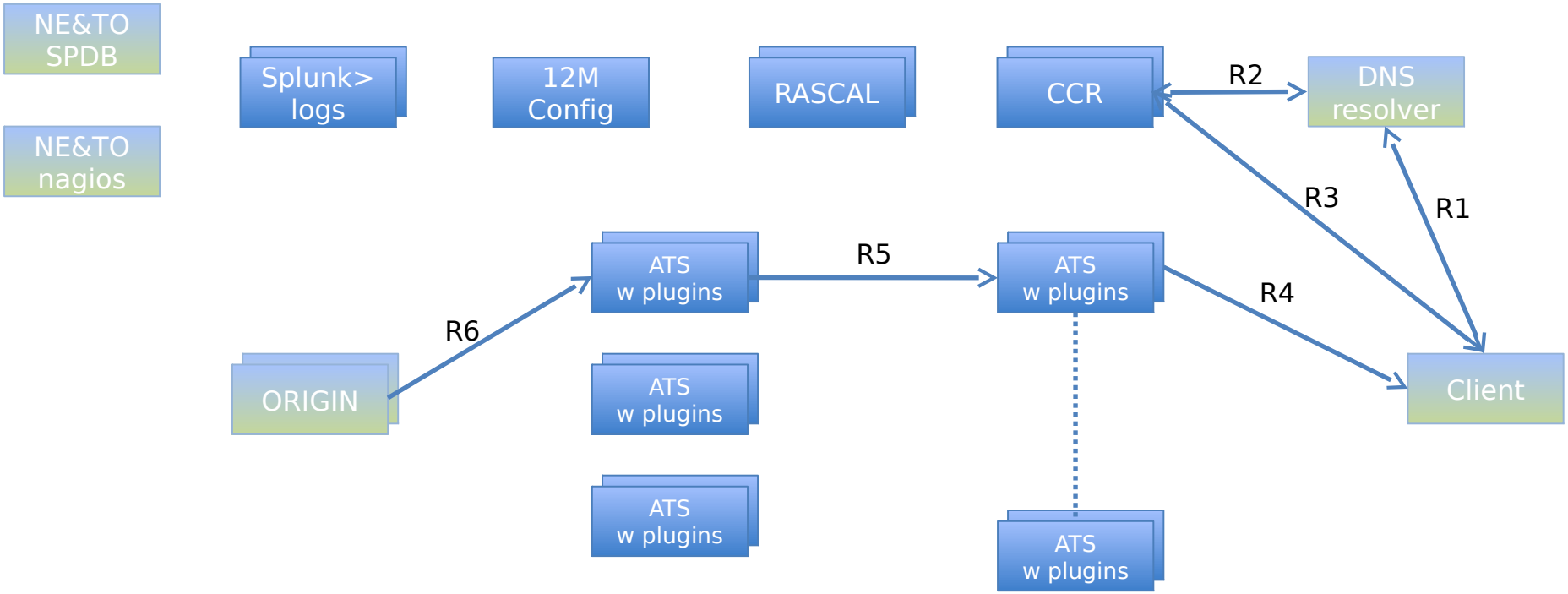
Comcast ipCDN – Client Request (DNS)



Comcast ipCDN – Client Request (DNS)

- **R1**: Client does DNS Q for host in delivery URL
- **R2**: DNS resolver queries DNS auth servers, last one is CCR; CCR answers with IP address of best cache based on DNS Resolver IP
- **R3**: Client does HTTP GET for /path to edge, edge applies reverse proxy rule to map to org server name
- **R4**: On miss, edge selects mid from parent location based on /path hash, does HTTP GET for http://org/path (fwd proxy request)
- **R5**: On miss mid does HTTP GET for /path on org

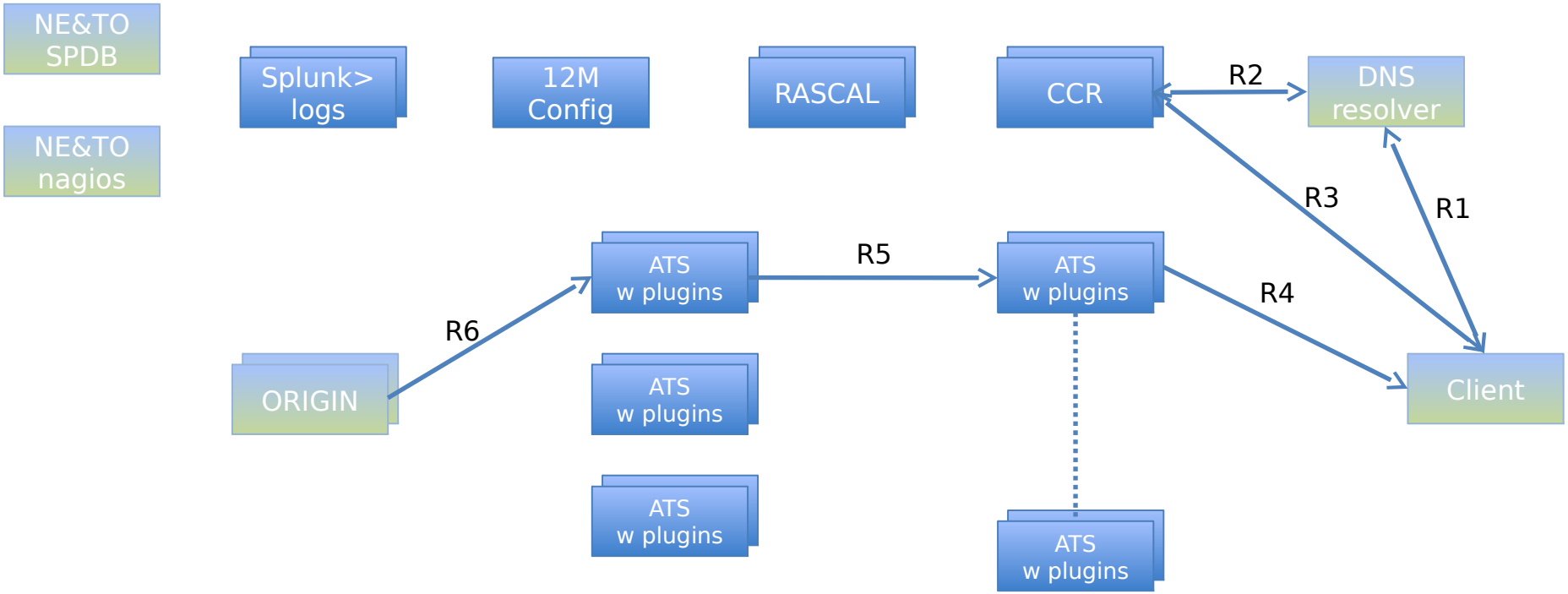
Comcast ipCDN – Client Request (302)



Comcast ipCDN – Client Request (302)

- **R1**: Client does DNS Q for host in delivery URL
- **R2**: DNS resolver queries DNS auth servers, last one is CCR; CCR returns it's own IP
- **R3**: Client does HTTP GET for /path to CCR, CCR responds with 302 of best edge cache
- **R4**: Client does HTTP GET for /path to edge, edge applies reverse proxy rule to map to org server name
- **R5**: On miss, edge selects mid from parent location based on /path hash, does HTTP GET for http://org/path (fwd proxy request)
- **R6**: On miss mid does HTTP GET for /path on org

Comcast ipCDN – Client Request (ALT)

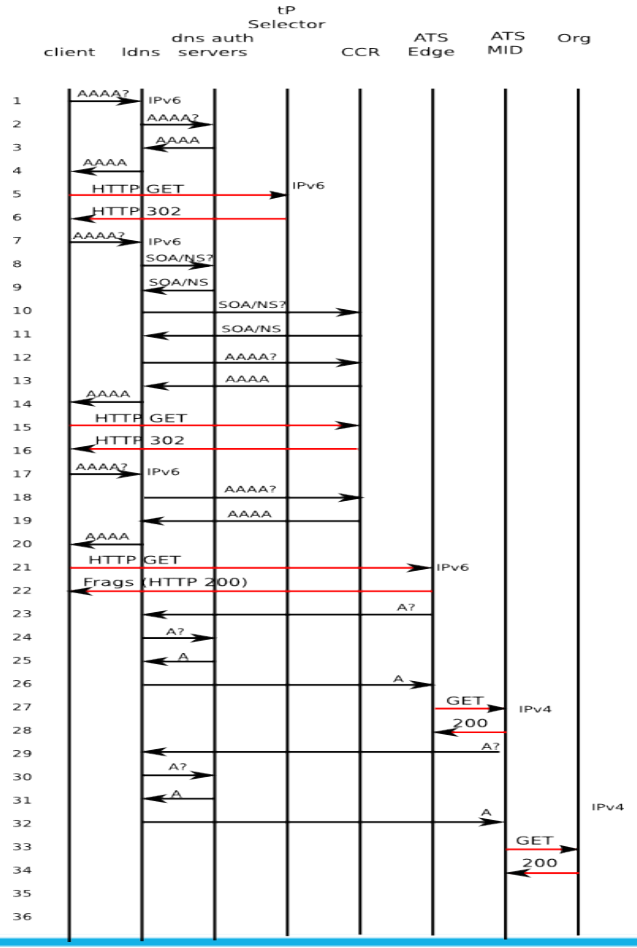


Comcast ipCDN – Client Request (ALT)

- **R1**: Client does DNS Q for host in delivery URL
- **R2**: DNS resolver queries DNS auth servers, last one is CCR; CCR returns it's own IP
- **R3**: Client does HTTP GET for /path**?format=json** to CCR, CCR responds with 200 OK and json pointing to best edge cache
- **R4**: Client does HTTP GET for /path to edge, edge applies reverse proxy rule to map to org server name
- **R5**: On miss, edge selects mid from parent location based on /path hash, does HTTP GET for http://org/path (fwd proxy request)
- **R6**: On miss mid does HTTP GET for /path on org

ipCDN and IPv6

- Client decides to do IPv6 by doing a AAAA DNS query; if it gets a response on the AAAA request, and has a non link-local IPv6 address itself, it'll use that. If not, it'll fall back to IPv4
- All hostnames and domain names are the same for IPv6 and IPv4
- All Edge contact points are be IPv6
- On the Edge cache, the same remap rule applies for IPv6 and IPv4
- Our Mid <=> Edge traffic is IPv4 initially
- Our Mid <=> ORG traffic is IPv4
- Our management and monitoring traffic is IPv4



Questions / Links

jan_vandoorn@cable.comcast.com
NETO-VSS-CDNENG@cable.comcast.com

- ATS: http://trafficserver.apache.org/
- OmniTI (ATS support): http://omniti.com/
- Mojolicious: http://mojolicio.us/
- jQuery: http://jquery.com/

National Engineering
& Technical Operations

Comcast