The Multi-CDN Dilemma

Aggregated Edge Networks at Scale

A technical paper prepared for presentation at SCTE TechExpo24

Ben Rosenblum
Principal Software Engineer
Vecima Networks
Ben.Rosenblum@vecima.com

Nick Dunkin
Director Product & Innovation
Vecima Networks
Nick.Dunkin@vecima.com

Table of Contents

List of Figures

List of Tables

List of Equations

1. Introduction

As consumer use of the internet for entertainment-grade video delivery steadily grows, associated increases in bandwidth present several challenges. Internet Service Providers (ISPs) must manage increases in network congestion, and consumers often experience degraded stream quality, leaving Content Service Providers (CSPs) in a struggle to consistently and dependably deliver high-bitrate content. One solution to the problem of network congestion presents itself: bring the content closer to the viewer by caching that content within the Service Provider's network. However, with over 2,242 [1] ISPs in the US alone, that could mean an enormous number of Content Delivery Network (CDN) integrations.

Open Caching, a non-proprietary specification developed by the Streaming Video Technology Alliance (SVTA), provides a uniform interface for the configuration of CDN infrastructure and traffic delegation. This provides a fabric of interoperability essential for the development of a sustainable multi-vendor ecosystem but unfortunately only solves in part the problem of integration between CSPs and CDNs. While it eliminates the effort of implementing proprietary interfaces, it does not address the ever-increasing burden of configuring a multitude of different CDNs, each with their own supported functionality and features.

A system that can aggregate multiple edge network CDNs into a single homogenous global CDN would allow the configuration and utilization of deep edge caching without extended effort.

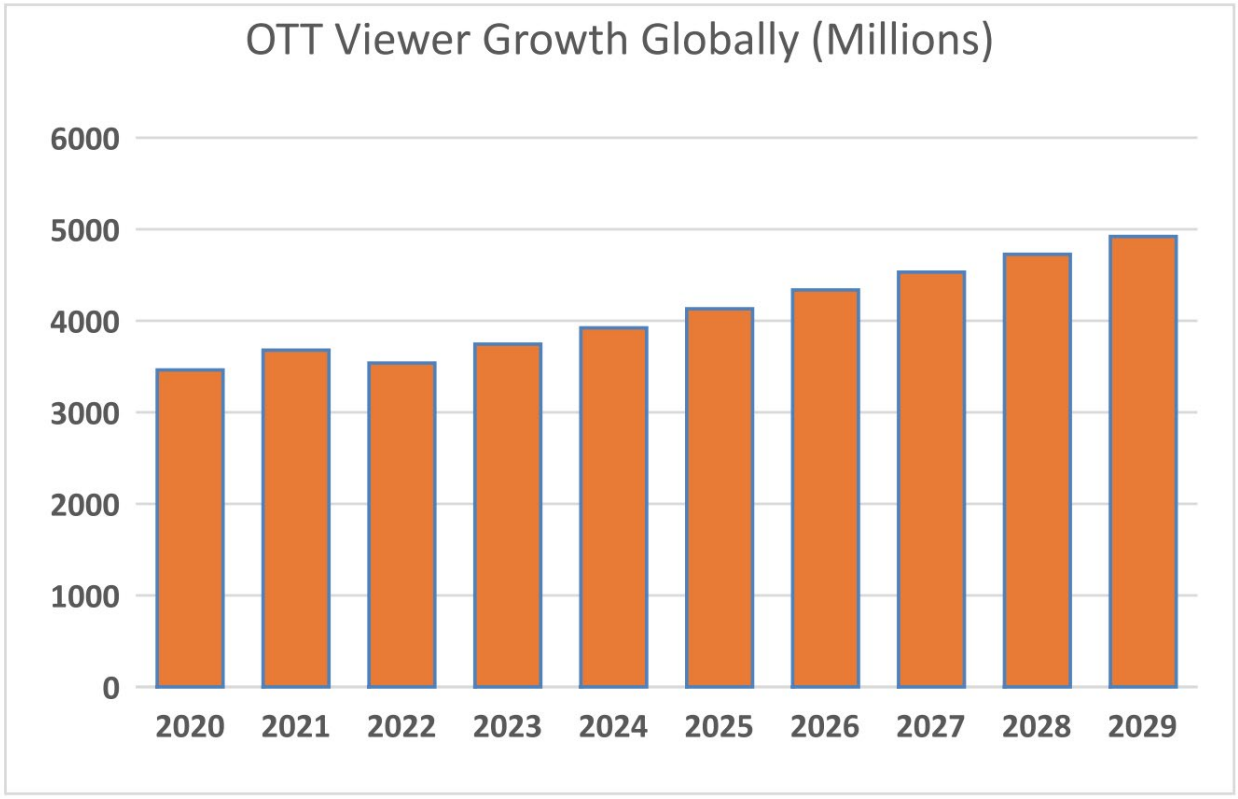
In this paper, we describe such a system. Independent ISPs serving different regional markets are consolidated behind a single global Open Caching Control Plane and Open Caching Request Router that propagates configurations to downstream caches located deep inside edge networks adjacent to subscribers and manages the delegation of streaming sessions from origination at the CSP to the appropriate edge caching node. Reporting, logging, and observability metrics are also aggregated for delivery to an upstream CSP, presenting to the delegating entity as a single CDN.

1.1. Network Congestion Through Increasing Internet Video Demand

As consumers continue to "cut the cable" and switch to the internet for their video needs, the associated increase in bandwidth usage can cause congestion throughout the entire video delivery system. The ISPs see an increase in the downstream data flowing across their network, the consumers experience degraded stream quality as their upstream networks become contended, and the CSPs struggle to satisfy their end users who want a dependable high-quality experience.

Streaming video subscribers expect the highest quality video experiences and will often blame the CSPs or their ISP when they do not receive it. Viewers expect to see streams that start playback immediately, have zero rebuffering events and consistently present the highest available bitrate.

It is projected that the number of global viewers of over-the-top (OTT) video content, 3.92 billion as of 2024, will reach 4.9 billion by 2029 [2]. Additionally, there is continued growth in the so-called Free and Ad Supported Television (FAST) market, with more than 1.1 billion users expected by 2027. [3]



(Source: https://www.statista.com/forecasts/1207843/ott-video-users-worldwide)

Figure 1: OTT Viewer Growth Forecast

In addition to a growth in general OTT consumption, we are beginning to see growth in "OTT exclusive" events, such as sports, that are not available through traditional broadcast channels. These events cause high spikes of data usage throughout the ISP's network. In January 2024, Peacock streamed an exclusive AFC Wild Card NFL game, an event only available via the internet. This game was the most accessed live stream in U.S. history and drove internet traffic to its largest single day usage, consuming 30% of all internet traffic that day. Nielson reported 27.6 million viewers watching this exclusive content online [4].

High volume events like this can push an ISP's network to the breaking point as each viewer retrieves a unicast copy of the content from the internet. As shown in Figure 2, the total unicast viewing demand must be transited across the last mile network.

The congestion in this system happens when the ISP has no way to cache video data, often originating from a public internet CDN, inside of its own network. For a high-volume event with millions of viewers, that equates to a high likelihood of network problems, increased download time to subscriber devices, and events that impact Quality of Experience (QoE), including buffering, quality down-shifting, stalling, and other user agent failure modes.

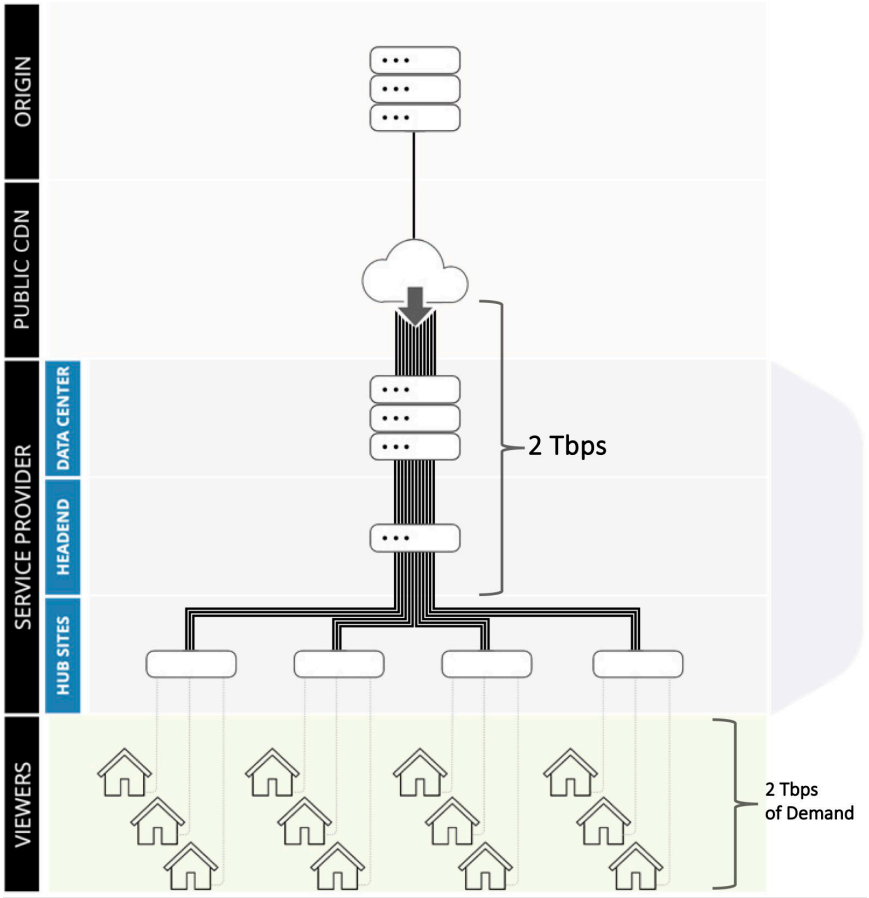


Figure 2: Network Congestion During a High Volume Event

A solution presents itself: cache the video data inside of the ISP's network, reducing network transit to the segment between the last mile cache and the user agent. The same total subscriber demand can be met within the last mile, massively reducing the required throughput in the rest of the network.

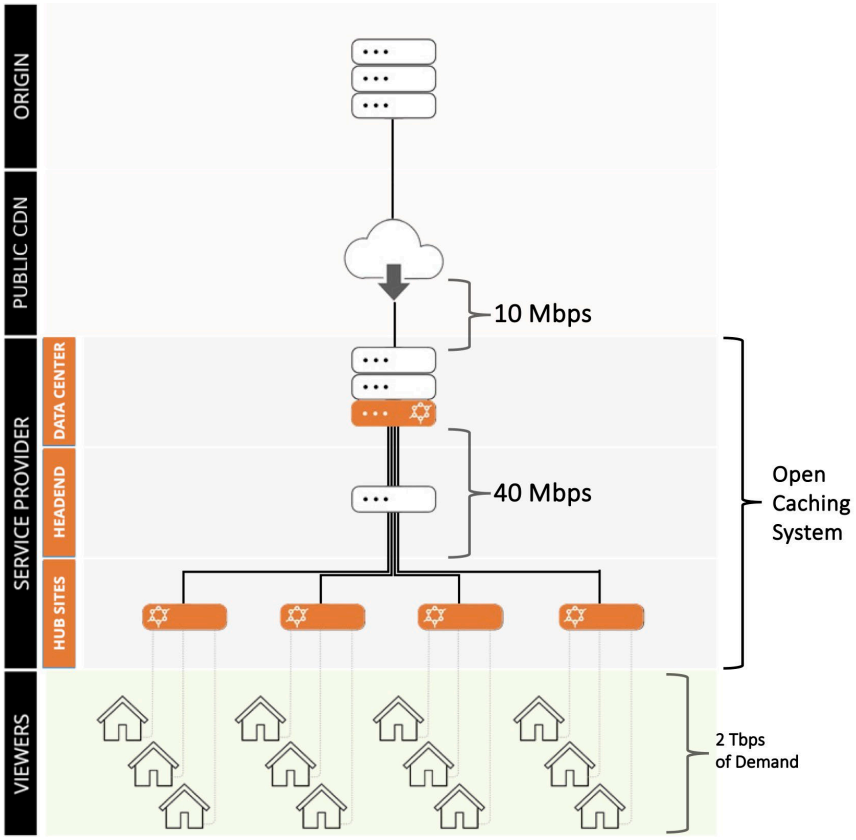


Figure 3: Caching in the Last Mile

The caching infrastructure within the Service Provider's network must present a control plane which allows configuration and management by CSPs, each with different functional requirements. Open Caching is one option for the implementation of that control plane. It exists to standardize the way CSPs and CDNs communicate, allowing any CSP to use a standardized API for provisioning and caching their content across different CDNs, include both large public CDNs and deep caching in ISP edge networks.

With the deployment of infrastructure which implements the Open Caching APIs, Service Providers can cache third-party video content directly within ISP networks, easing upstream utilization, lowering the expenditure on external CDNs, and improving application performance.

Once Open Caching Systems are widely deployed within ISP networks, a new problem is presented: scale. Unlike deploying to the handful of public internet-facing CDNs, CSPs would face challenges in performing integrations with potentially thousands of ISPs. Instead of working individually with every edge network, an aggregation service provider can provide a single Open Caching compliant endpoint that advertises the combined footprint of a block of ISPs, reducing the implementation overhead and simplifying traffic delegation and reporting.

2. The Open Caching Specification

Created by the Open Caching Working Group of the Streaming Video Technology Alliance beginning in 2016 [5], Open Caching is a specification for content delivery unencumbered by proprietary technologies

which defines four fundamental pillars of CDN functionality: traffic delegation, configuration, observability, and content management. Each area is addressed by its own set of specification documents, freely available at the SVTA website [6].

Open Caching extends Content Delivery Network Interconnection (CDNI) [7], a series of proposed standards drafted by the CDNI Working Group of the IETF (Internet Engineering Task Force). Open Caching enhancements to the CDNI specification are reintroduced via the IETF as revisions and additions to the existing CDNI Proposed Standard Requests for Comment (RFCs), ensuring that the specification remains open and accessible to all participants in the CDN ecosystem.

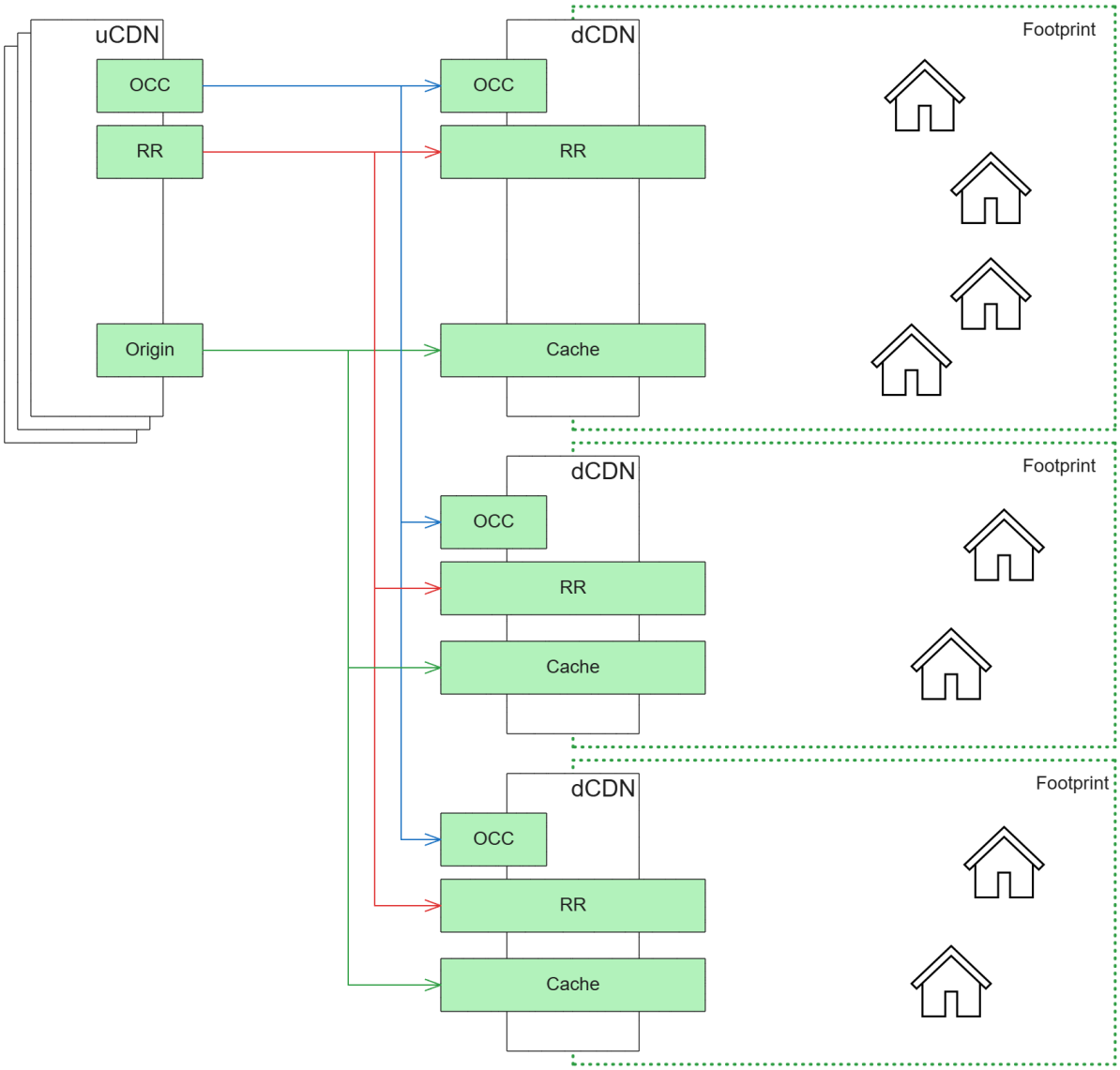


Figure 4: Open Caching Architecture

2.1. Traffic Delegation

In Open Caching, the relationship between content source and content delivery agent is defined in terms of an upstream CDN (uCDN) and a downstream (dCDN). The uCDN may be an originating source, such as a Content Provider, or it may be an intermediary CDN that is passing the traffic to another CDN.

Open Caching specifies two primary methods of delegating client traffic from an upstream CDN (uCDN) to downstream CDN (dCDN): DNS and HTTP redirection. In either case, the destination for the request is selected by the Request Router (RR).

With DNS traffic delegation, a DNS zone is delegated via an NS record to the responsible RR. When the client performs a DNS resolution for the entry point of the CDN, the RR resolver returns a record that sends the client to the selected destination. As DNS records may be cached by an intermediary DNS server, this can limit the ability to route individual clients to different destinations when they share a common DNS server. Additionally, limited information about the client might be available for utilization in making the routing decision. There are various methods for circumventing this limitation such as client-specific DNS names and EDNS.

HTTP redirection provides a simpler approach: the client request to the CDN entry point is evaluated by the RR, and an HTTP 302 response is generated which directs the client to the correct dCDN caching server. Open Caching specifies two methods of HTTP delegation: iterative and recursive.

With iterative delegation, the initial RR may redirect the client to a subsequent RR which will, in turn, send the client to another RR or to a caching server. While simple to implement, this method of delegation can result in long redirection chains.

Recursive delegation is enabled by the Open Caching Request Routing Specification which describes an API for querying the routing decision of subsequent RRs. The initial RR, rather than immediately redirecting the client, may first query the next RR which will in turn utilize the recursive API to query the next RR in the chain until arriving at the ultimate routing decision. This decision is propagated back to the initial RR and returned to the client, resulting in only one HTTP 302 response that sends the client directly to the chosen caching server. [8]

The RR decision-making workflow can be complex and remains outside the scope of the Open Caching specification. In the context of a CDN aggregator, many factors can affect the client routing decision as discussed in detail in Section 5.

2.2. Configuration

CDNs have varying support across a broad list of features and functionality, and this can present difficulty in facilitating interoperability between them without substantial work in developing custom integration. To alleviate this necessity, Open Caching provides two fundamental components: an advertisement interface which allows the dCDN to present its supported capabilities and a configuration interface which allows the uCDN to publish specific configuration values for each of the dCDN's supported features. [9] [10]

Each interface is structured as a set of JSON objects which can be individually supported, allowing variance in the capabilities between CDNs, but this presents its own problem when dealing with multiple partners. If each CDN supports a different feature set, how should a uCDN approach managing its configuration across providers? This problem is addressed in Section 3.

2.3. Observability

Three interfaces are currently provided for observability by the Open Caching specification, Capacity, Telemetry, and Logging. Capacity and Telemetry live together in the same document, the Open Caching Capacity Insights Interface Specification, while logging is a standalone document, the Open Caching Logging Interface [11].

Capacity Insights provides guidance on the levels of traffic a uCDN is permitted to delegate to the dCDN using well-defined units including egress bits per second, requests per second, total storage size, total object count, session county, and total cache size. These limits are published via the FCI advertisement interface through the `FCI.CapacityLimits` object.

Alongside the published limits, the specification allows for near-real-time telemetry that informs the uCDN of current utilization. A simple mechanism allows metrics to be embedded inside the `FCI.CapacityLimits` object alongside the corresponding limit, but for more general usage, the specification also defines an `FCI.Telemetry` object that holds a reference to an external service, e.g. a Prometheus endpoint.

2.4. Content Management

Licensing agreement terms may require strict adherence when ensuring that expired content is promptly removed from any backing storage. Additionally, an operator may wish to pre-warm caches in expectation of new releases or live events. Open Caching provides a content management interface [11] which allows for both object purging and content pre-positioning.

It is the responsibility of the aggregating CDN to distribute content management requests received at its OCC to all affected dCDNs, gather the resulting operation statuses, and return a composite reply to the uCDN. With pre-positioning, a best effort is often acceptable, but content purging typically requires full compliance due to contractual licensing obligations; for this reason, a dCDN that does not implement content purge operations should probably not be considered for use.

3. Configuration Aggregation

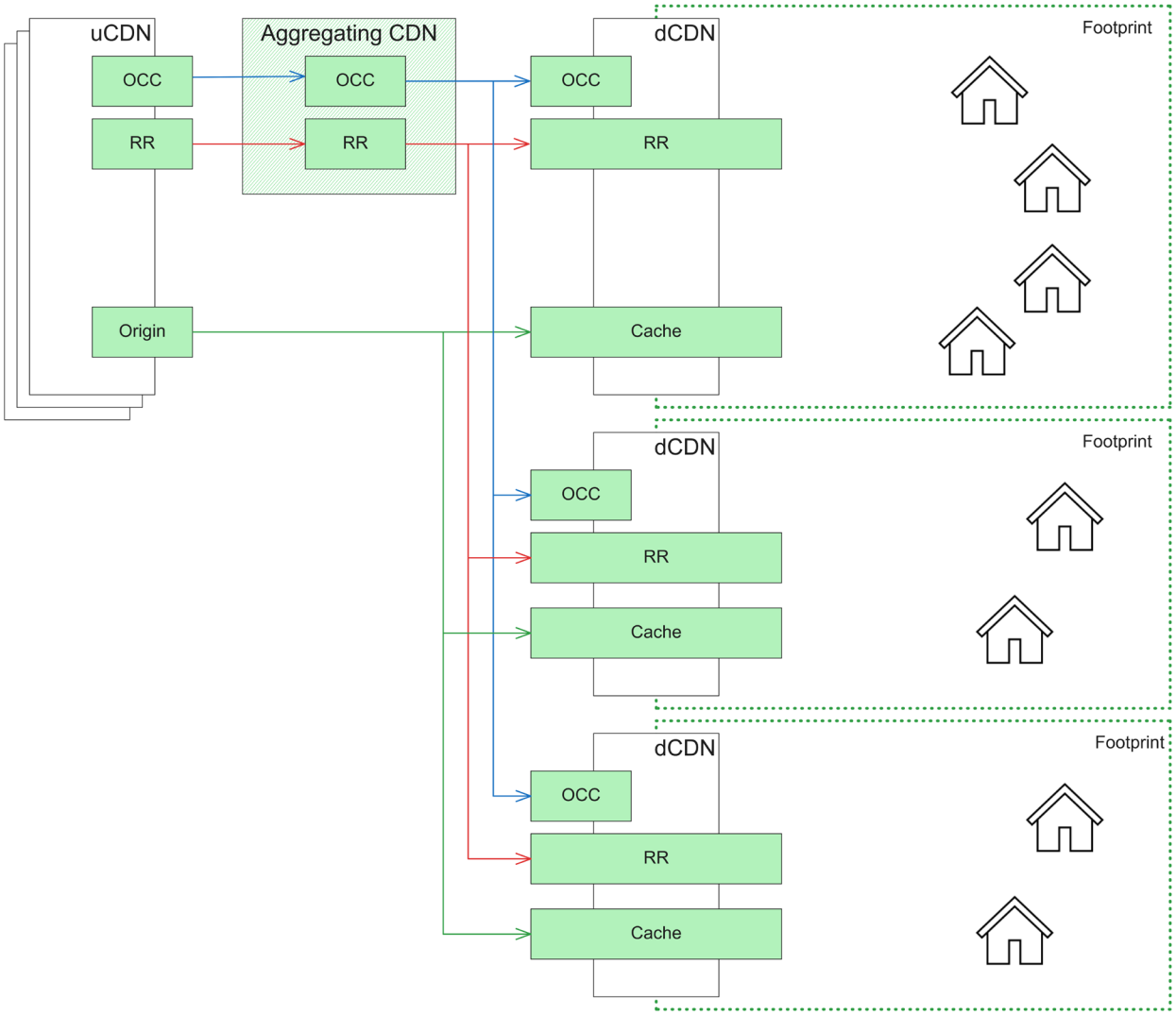


Figure 5: Aggregating CDN Architecture

Presenting an aggregated dCDN facade to the uCDN presents significant challenges. Each dCDN has its own unique coverage area defined by a set of `FCI.Footprint` objects, a mix of supported Open Caching features, different quotas for traffic and request rates, and unique feeds for logging and telemetry in support of observability.

It is the responsibility of the aggregator to coalesce this disparate set into a unified dCDN. The advertisement presented by the OCC should either collapse the dCDN footprints to the minimal set or present a single global footprint. Both options have their own tradeoffs and will be explored below.

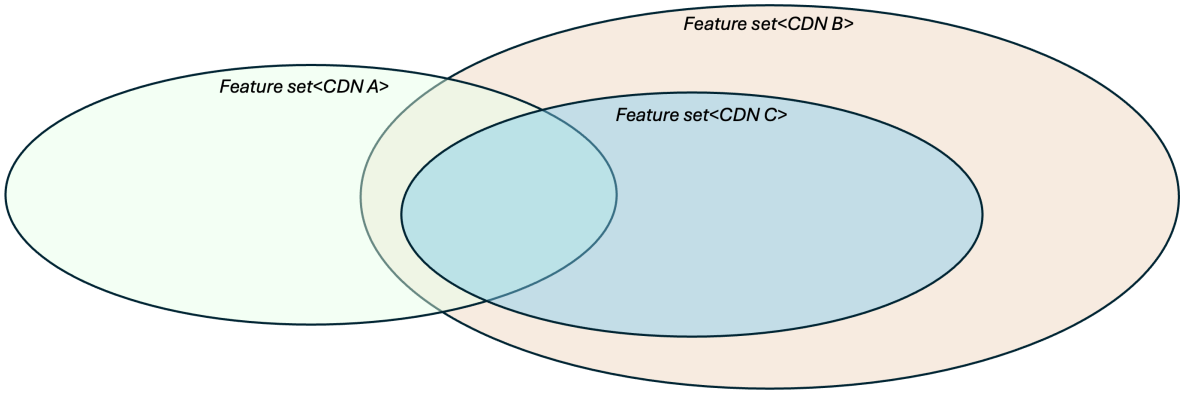


Figure 6: Overlapping and Disparate CDN Features

Log files and metrics must be collected or received from each dCDN and collated for distribution to the uCDN. Downstream logging configuration must utilize appropriate request tagging to ensure correlation of log records for each uCDN. While this arrangement may seem similar to that of an intermediary CDN in a multi-tier Open Caching architecture, the configuration in this case is not a pass-through for a single tenant provisioned across the entire combined CDN, but a gateway which manages the relationships on both the upstream and downstream sides.

Capacity planning can also be complex. If the features required by the uCDN are mandatory and supported on only a subset of the dCDNs, then the capacity of those dCDNs cannot be utilized. When you consider that usage of unsupported features may happen on both a host and a path level, then the capacity might differ on a request-by-request basis, leaving the existing Open Caching mechanism of `FCI.CapacityLimits` is insufficient for advertising the limit values at an appropriate granularity. Consideration of the granular quotas on a request basis may happen at the aggregator Request Router, but another feedback mechanism might be required to explain why incoming traffic is being rejected by the CDN when, according to the footprint-defined limit, plenty of headroom is still available. The alternative is advertising the most restrictive limit, ensuring that there is available capacity on dCDNs for the most featureful request regardless of what fraction of the overall traffic requires these features.

Some relief for the disparate feature sets might come in the form of official feature profiles. A configuration profile would consist of a set of Open Caching metadata objects that the dCDN must support to be compliant. Combined with a postulated certification program, this may cultivate a sufficient level of feature support among participating dCDNs to make aggregation viable without burdensome decisioning on the handling of unsupported configuration. A required profile at the aggregator, after some base participation is already established, could drive overall adoption of future Open Caching features among dCDNs.

3.1. Footprint Aggregation

The Open Caching advertisement requires specification of a footprint attached to every Capability object, but this description is non-uniform. An `FCI.Footprint` may be defined in terms of CIDR mask, ASN, or geographic region. The aggregated CDN must merge the dCDN advertisements, being careful not to combine conflicting capability objects, and publish additional objects that represent a facade on top of the underlying functionality, e.g. an aggregator provided Kafka logging transport that streams logs received in file form from a dCDN.

In cases where some of the dCDNs do not support a feature, the aggregator must decide how to advertise the discrepancy. To enable delegation of traffic supporting the superset of every feature supported across all dCDNs, the aggregator could include separate capabilities for each set of dCDNs which overlap in feature and footprint. This would result in a complex advertisement with footprint carve-outs for every small permutation of supported features, each with a corresponding `FCI.RedirectTarget` for the footprint.

Alternatively, the aggregator could reduce the advertised capabilities to either the least or the greatest common set under a single set of footprint objects. The least common set restricts the types of configurations the advertised CDNs can accept, even if those features are only unsupported by a small subset of the dCDNs, but it may be an acceptable solution with the advent of Open Caching configuration profiles.

With the superset of supported features, the aggregator can accept all traffic and then dynamically route based on which dCDNs can support the request, using the configuration metadata attached to the host (`MI.HostMetadata`) and path (`MI.PathMetadata`). However, in cases with widely disparate features which are required in the same request, this may result in the inability to serve the request at all, as the required features are split between multiple CDNs, none of which support everything. For the uCDN, use of a particular feature may not be absolutely required; in this case, this conflict may be resolved with better support for optionality in Open Caching configuration. Future changes to `FCI.Metadata` could allow advertisement of fine-grained support of metadata beyond the object level. A similar mechanism on the configuration side could allow requests to pass when lack of implementation for an accepted configuration object is not an error condition, allowing the uCDN to understand and accept the partial support.

3.2. Configuration Propagation

When new configuration metadata is pushed to the aggregator by a uCDN, it is the responsibility of the aggregator to reconfigure the dCDNs appropriately. Open Caching provides two mechanisms for publishing configuration. The Simple Configuration Metadata API and the Orchestration API [11].

The Simple API provides no feedback mechanisms during the deployment lifecycle aside from completion or failure and no ability to validate configuration before deployment. When deploying configuration across a set of CDNs, this can result in synchronization issues, leaving each CDN in a different state.

The Orchestration API, once the full specification is completed, will provide lifecycle management for service configuration. This would allow the aggregating CDN to coordinate configuration updates across the fleet of dCDNs and ensure that newly received traffic affected by the configuration change is not re-delegated to a dCDN until the updated configuration has been applied.

4. Observability and Reporting

Logging and telemetry are essential to a CDN for many purposes, including operational monitoring and troubleshooting, traffic delegation decision-making, QoS measurement, auditing, and billing. Every service has its own requirements for these data, expecting a variety of record formats, file types, and transports.

4.1. Logging

Open Caching provides several mechanisms for providing both real-time and batch reporting to the uCDN, but the specifications currently have little to say about aggregation. The Logging Specification [11] provides for the existence of an aggregation component on the dCDN, but it is concerned with aggregation of logs across the dCDN's own nodes into a single report to the upstream. A dCDN aggregating logs across its own infrastructure and in control of its internal data formats and transport mechanisms does not face the same challenges as an intermediary CDN which must combine reports across multiple dCDNs each with their own supported features.

For each uCDN, the aggregating CDN must synthesize the varying reports from each dCDN that has accepted traffic for that uCDN into a single data stream. Log records must be coalesced and transformed into the expected format as configured by the uCDN via the `MI.LoggingMetadata` configuration object and then transmitted or stored accordingly.

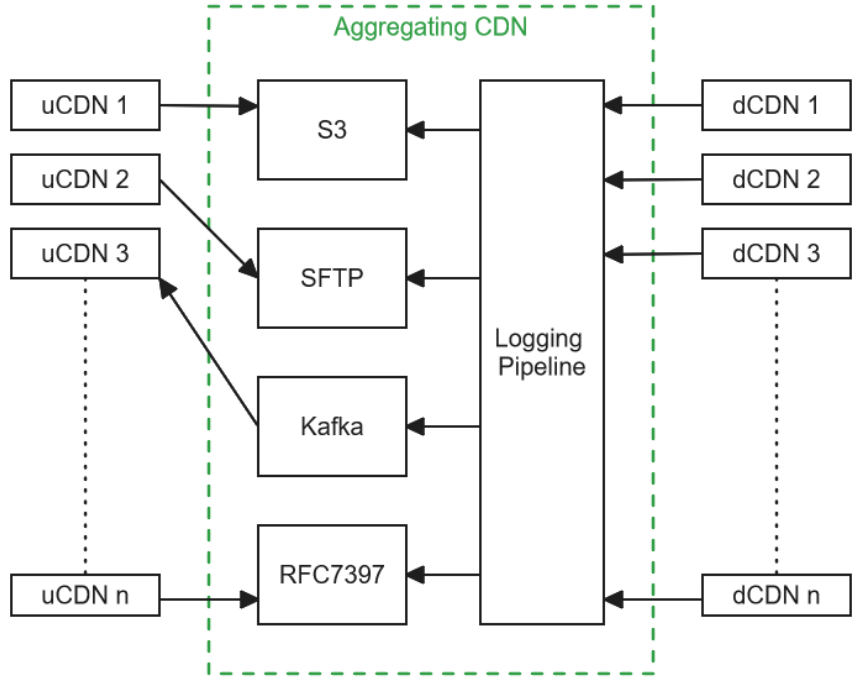


Figure 7: Log Aggregation

A transformation pipeline may be employed to coerce the dCDN logs into the appropriate format, containing the requested fields, for each uCDN. The aggregating CDN is responsible for configuring each dCDN via the Open Caching metadata interface to output a superset of the necessary fields so that all data remains available for selection by the pipeline when building each file.

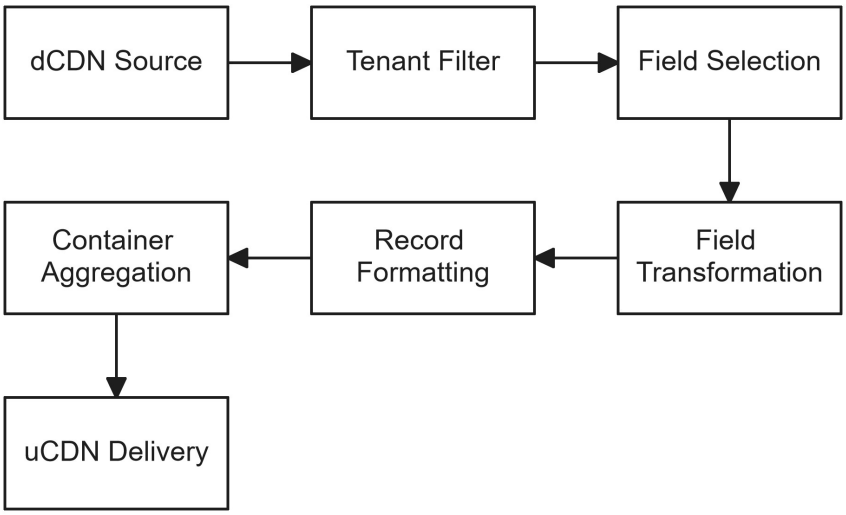


Figure 8: Logging Pipeline

Upon receipt of a log file from a dCDN, the pipeline first filter by each uCDN tenant as each uCDN has its own logging configuration (Tenant Filter). The configured fields, according to the configured Open Caching logging record type, are selected (Field Selection) and passed to a transformation pipeline (Field Transformation). The logging specification allows for various transformations to be applied at the field level, e.g. encryption, truncation, masking, and other textual changes. Additionally, base fields might be enriched at this point in the pipeline, expanding fields like client IP address into a set of geolocation fields or performing a user agent analysis to return operating system and client device.

The transformed fields must then be formatted as defined by the selected logging record type. This can mean JSON, CSV, and protobuf, but future versions of the Open Caching logging specification should offer the ability for a uCDN to configure an entirely custom format (Record Formatting).

Depending on the transport configuration, the records are then packaged into a file (JSON, newline delimited, or protobuf), and also possibly packaged with other log files into a tarball archive (Container Aggregation). The log files may be sliced by time, size, or other criteria, and then shipped to S3 or made available on an SFTP or HTTPS endpoint. Alternatively, with a Kafka transport, the records are immediately streamed to the destination endpoint (uCDN Delivery).

4.2. Telemetry

Real-time and near-real-time telemetry are also essential, particularly for making traffic delegation decisions and informing the uCDN of available capacity. Two mechanisms are available for real-time metrics: a “current” property in-band of the Capacity Insights Interface [11] and a reference to an external telemetry feed that can be of any type.

The embedded telemetry mechanism allows a single metric to be published alongside a corresponding limit value. This mechanism is limited solely to feedback for traffic steering and provides a point-in-time value without history.

```
{
  "capability-type": "FCI.CapacityLimits",
  "capability-value": {
    "limits": [
      {
        "id": "us_na1_egress",
        "limit-type": "egress",
        "maximum-hard": 60000000000000,
        "maximum-soft": 55000000000000,
        "current": 93330032449,
        "telemetry-source": {
          "id": "us_na1_metrics",
          "metric": "egress_5m"
        }
      }
    ]
  }
}
```

Figure 9: In-band Telemetry with FCI.CapacityLimit

Also provided is the facility to reference an external source of telemetry via the `FCI.Telemetry` object. The specification does not yet define any specific formats for this telemetry, providing only a "Generic" type with the actual format to be define out-of-band, but it is expected that future drafts will incorporate support for commonly used services like Prometheus.

```
{
  "capability-type": "FCI.Telemetry",
  "capability-value": {
    "sources": [
      {
        "id": "us_na1_metrics",
        "type": "generic",
        "metrics": [
          {
            "name": "egress_5m",
            "time-granularity": 300,
            "data-percentile": 50,
            "latency": 1500
          }
        ],
        "configuration": {
          "url": "https://telemetry.dcdn.com/v1/cpm1/egress"
        }
      }
    ]
  }
}
```

Figure 10: External telemetry source with FCI.Telemetry

Difficulty can arise in determining the immediate utilized capacity for each uCDN as the provided metrics are an aggregate across footprint without distinction by hostname. Once a session is delegated to a dCDN, the only feedback to the aggregating CDN on the passed traffic is in the form of this aggregate metric which consists of all traffic delegated to the dCDN, including traffic from other uCDN tenants. Several possible solutions present themselves.

The telemetry could be enhanced with tenant data through a future enhancement to the Open Caching telemetry interface. Separate metrics could be provided for each content host, or a custom value could be passed via URL or HTTP header to be utilized as a key. Telemetry might also allow configuration by the uCDN, similar to the use of `MI.LoggingMetadata`, with separate explicit configuration per content host.

Estimation could also be sufficient for feedback controlling traffic delegation. The aggregating CDN is aware of how many sessions it passes from each uCDN to each dCDN, and it has the aggregate current traffic value from each dCDN telemetry source. If the dCDNs also provide a current session count (supported by the existing Open Caching telemetry interface), the aggregating CDN can calculate an average session length, and in turn, provide an estimate of the current traffic for each uCDN based on the rate of session starts. This rough total is likely good enough, given that in this proposed arrangement with an aggregating CDN, the uCDN has no decisioning to make for delegation, and the telemetry is merely informative and not functionally required. Internal to the aggregating CDN, the session delegation rate is also likely sufficient to make determinations of any immediate breech of tenant quotas, while delegation itself is unaffected by the lack of metric granularity as the only information needed on a request-by-request basis is total available capacity.

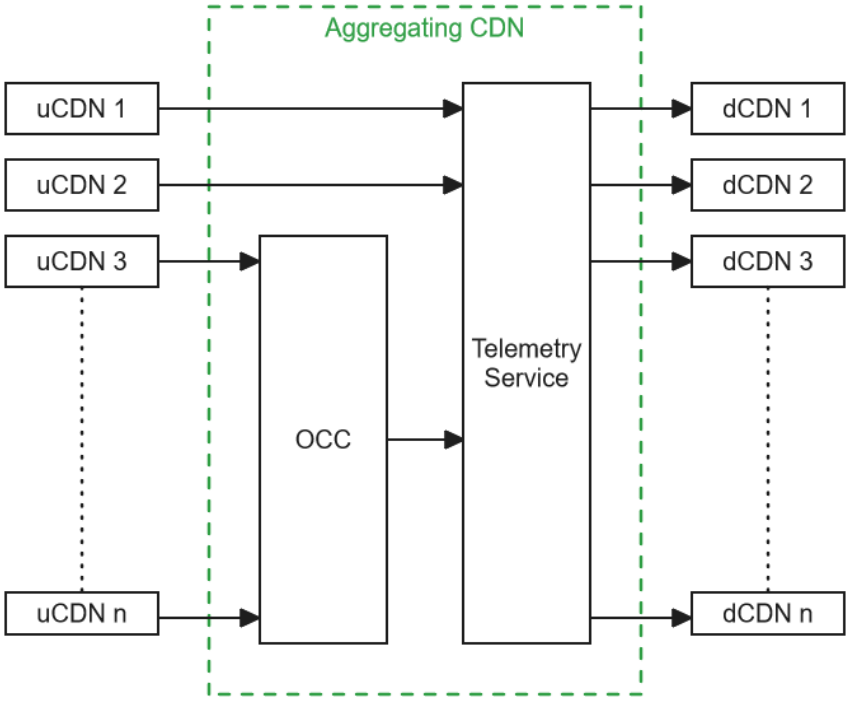


Figure 11: Telemetry Aggregation

If accuracy is absolutely required and near-real-time log records are available (e.g. via a Kafka message stream), then tenant-specific metrics could be computed as an output of the log processing pipeline. As a standalone solution, this is quite expensive when compared against metric sampling, but if the

aggregating CDN is already performing near-real-time log processing, this could potentially be accomplished at minimal additional cost.

5. Delegation and Multi-CDN Selection

5.1. Delegation and Multi-CDN Selection

For every request that arrives at the Request Router of the aggregating CDN, a decision must be made to determine where to delegate the incoming session. The list of dCDNs must first be filtered to those capable of handling the session, and then the remaining dCDNs are assigned a score derived from a set of criteria after which the highest scoring dCDN is selected for delegation.

5.2. Features

Depending on the configuration published by the uCDN which has delegated the session, some dCDNs may be excluded from consideration due to lack of available feature support. As covered in Section 2, even if a CDN supports Open Caching, it may only implement a subset of the specification's features. If a uCDN makes use of a configuration object (e.g. `MI.ProcessingStages`) that is only supported on a subset of the dCDNs, the selection list is narrowed accordingly.

As configuration happens independent of the request delegation lifecycle, the list of valid dCDNs for each configured host and URL path prefix may be computed ahead of time when configuration is newly applied.

5.3. Network Distance

For any given user session, multiple dCDNs may be capable of handling the request, but some will be more optimally positioned than others. A public CDN, operating at global scale, can handle any internet connected client, but it will likely provide an inferior experience to a deep edge cache positioned only a few hops away inside the user's ISP.

The network distance is not considered explicitly as part of the score, because the ultimate effect of this metric is made evident when considering QoE. Instead, the list of dCDNs is culled based on matching footprints.

5.4. Score

The criteria that can be utilized for scoring a dCDN is vast. Here, we narrow it down to five terms, each of which is described below. Each term is multiplied by a weight, provided to the aggregating CDN by configuration. The final score, S, consists of the following weighted sum:

$$S = Q + C + T - E - R$$

Equation 1: CDN Selection Score

Table 1: Terms for CDN Selection (Equation 1)

Term	Definition
Q	Quality of Experience (QoE)
C	Capacity
T	Traffic Commitment
E	Delivery Error Rate
R	Financial Cost

5.5. Quality of Experience

Many CSPs give top priority to the user video playback Quality of Experience (QoE) and rank the affecting metrics accordingly. According to the Nielsen Total Audience Report, 77% of viewers consider streaming and playback quality to be extremely or very important [13]. In computing this term, a variety of data from multiple sources may be considered from aggregate client telemetry and CDN infrastructure observability metrics.

A difficulty arises here in that the aggregated CDN is generally not privy to proprietary telemetry generated by the user agent and reported directly to the uCDN via a third-party service provider. As it is in the interest of the uCDN to have their client sessions delegated to the dCDN with the best QoE, the uCDN might make these metrics available to the aggregating CDN. Open Caching does not currently provide a mechanism for sharing this information, but future additions to the Telemetry Interface could offer a solution.

Common Media Client Data (CMCD) [12] provides a standardized mechanism for direct transmission of metrics from the user agent to the edge cache node via query parameters or HTTP request headers. While CMCD will allow collection of metrics by the serving CDN, to be considered by the aggregating CDN when calculating a routing score, the metrics must be transmitted upstream from the collecting dCDN. Again, a future draft of the Open Caching Telemetry Interface could provide the necessary transport.

According to the QoE Working Group of the SVTA, the following should be considered Key Delivery Metrics [14]:

- Video Startup Time (seconds)
- Re-buffering Ratio
- Average Media Bitrate (bps)
- Video Start Failure

As a possible formula for determining a CDN's QoE score, we can sum the above metrics, taking a current sample of the rolling average at the time of calculation and applying a weight function to each. The bitrate in particular should be weighted down to bring it in line with the other terms.

$$Q = B - R - F - V$$

Equation 2: QoE Score

Table 2: Terms for QoE Score (Equation 2)

Term	Definition	Range
B	Average Bitrate	0 – MAX(B)
R	Re-buffering Ratio	0 – 1.0
F	Video Start Failure Ratio	0 – 1.0
V	Video Startup Time	0 – MAX(V)

Example weights and scoring for a dCDN Q value:

Table 3: Example QoE Score

Term	Value	Weight	Computed Term
B	67000000	0.0000001	67
R	0.013	100	13.0
F	0.0012	100	1.2
V	3.23	1	3.23
Q	-	-	49.57

5.6. CDN Health

Beyond consideration of QoE, the delegating CDN must also be aware of the immediate health of the dCDNs for which it is considering delegation. Delivery error rates, such as HTTP 4xx response codes, can be observed from dCDN provided telemetry feeds or derived from log records.

5.7. Capacity

In the absence of a traffic commitment, traffic should be balanced across eligible dCDNs with available capacity to handle the delegation. The Open Caching Capacity Insights Interface [11] provides a mechanism for the dCDN to communicate capacity limits and provide near-real-time feedback on the current traffic levels as observed by the dCDN.

The Capacity term is the difference between the current reported egress utilization and the soft limit advertised by the dCDN in bits per second. The Capacity Insights Interface supports additional limit types (e.g. request rate) that may be a useful consideration for certain types of applications, and those elements may be summed with this term if required.

5.8. Traffic Commitment

Depending on the business arrangement between the aggregating CDN and a dCDN, the aggregated CDN may be responsible for meeting certain traffic delegation obligations in order to maintain bulk pricing agreements. This consideration could weight the decision to delegate to a particular dCDN in its favor even if it is lagging other dCDNs on the other score terms.

We calculate T as the difference between the outstanding traffic commitment and the current total delegated traffic, denominated in Gigabytes, clamped to a minimum of 0.

5.9. Financial Cost

Balancing dCDN selection based on cost versus the other terms can be an interesting and complicated exercise for business analysis that lies outside the scope of this paper. Here, we apply a pre-determined weight against the CDN egress price, denominated in a currency unit per gigabyte, bearing in mind that this price may be dynamic.

5.10. Example Score

Table 4: Example Calculation for dCDN Score (S)

Term	Value	Weight	Computed Term
Q	49.56	2	99.12
C	56296649990700	1e-12	56.2966
T	584994	1e-4	58.4994
E	0.0015	1000	1.5
R	0.0009	50000	45
S	-	-	167.416

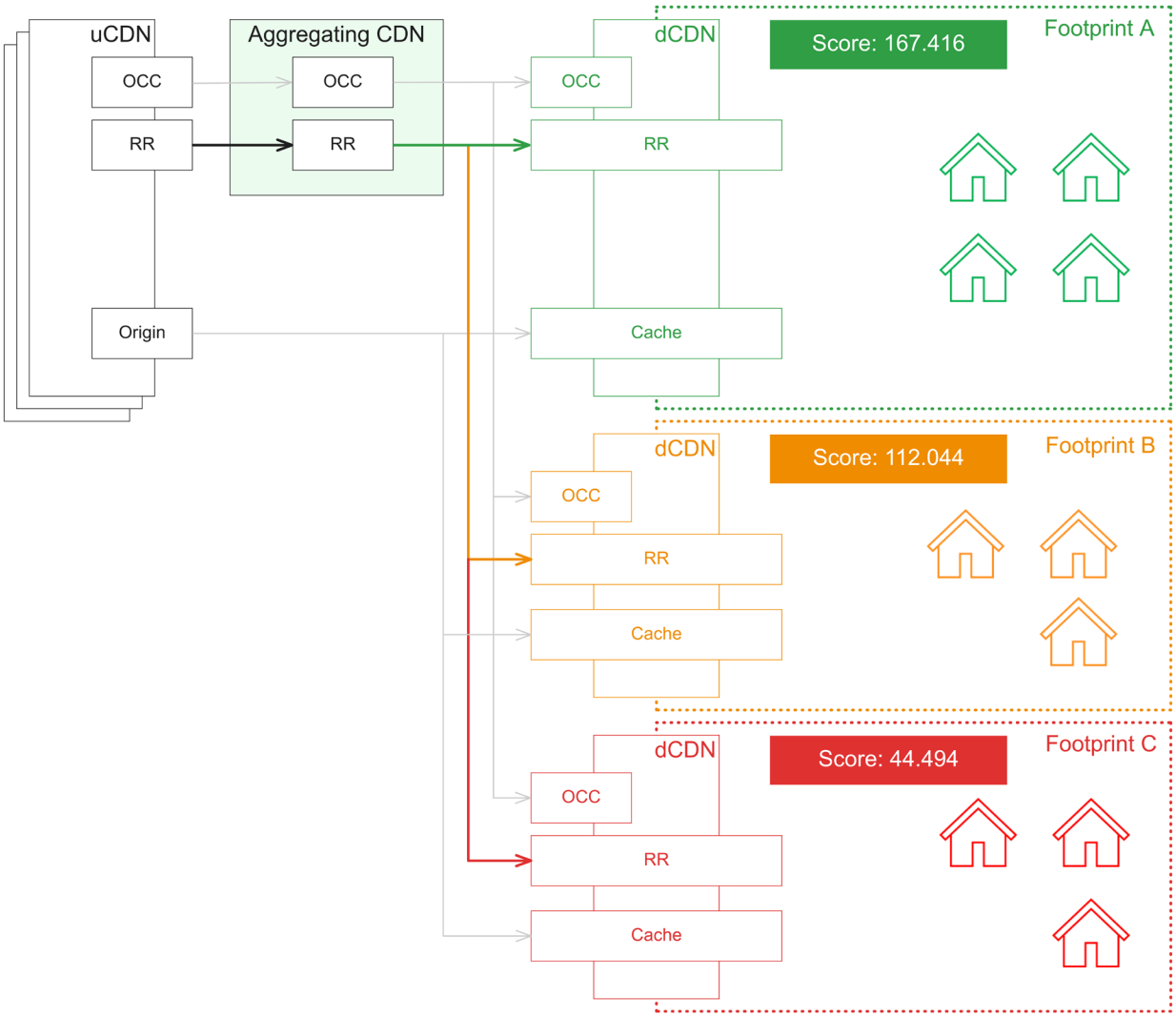


Figure 12: dCDN Scoring

Figure 13 shows an example selection between three competing dCDNs. Footprint A represents a highly scoring dCDN with solid scores across all criteria. Footprint B represents a similar dCDN that has a reduced score due to lack of available capacity (C). Footprint C is a dCDN that has poor QoE (Q) and high financial cost (R).

6. Conclusion

The predicted growth of internet video will place an untenable strain on ISP networks. This congestion will impact subscriber QoE and create customer churn. It is in the interest of CSPs and ISPs to find better ways of delivering this video data and satisfying their customers. This network strain can be mitigated by deploying caches deep in ISP networks, creating islands of CDN capability. However, for practical integration with CSPs, this proliferation of *islands* calls for a consistent and manageable control plane that aggregates the combined capacity into a single multi-footprint CDN.

Open Caching solves part of this problem by providing an API specification that defines a consistent control plane, but the multitude of last mile CDNs must be managed and maintained under an aggregated framework which allows the entire system to be treated as one CDN. Throughout this paper we have highlighted the challenges and potential solutions to the aggregation of disparate footprints, configurations, reporting and observability requirements, and have demonstrated a model for delegating traffic based on a scoring system which considers highly relevant observability metrics.

The future for deep edge caching is bright, but it remains full of significant challenges. Beyond initial integration lies other possibilities that bear consideration. Other industries have faced similar challenges, and there are lessons that may be applicable here. The internet advertising industry has developed an open framework for real-time bidding on ad impressions (OpenRTB [15]). A similar approach with deep edge caching may reduce the friction of establishing the necessary relationships and pricing models. With an aggregating CDN acting as a broker between uCDNs and dCDNs, a bidding marketplace could allow for cross-CDN spot pricing and dynamic pricing for capacity reservations, reducing overall cost to CSPs, increasing viewer QoE, decreasing internet backbone traffic, and producing additional revenue opportunities for ISPs to monetize their excess internal delivery capacity.

Abbreviations

{Delete these instructions: Put all abbreviations in this section. Words should not be capitalized unless they are formal names. See examples below.

Examples below should be deleted if they are not contained in this document.}

AP	access point
bps	bits per second
FEC	forward error correction
HD	high definition
Hz	hertz
K	kelvin
SCTE	Society of Cable Telecommunications Engineers

Bibliography & References

Include an annotated bibliography of key resources providing additional background information on your topic.