CDN Request Routing to Reduce Network Access Cost

Varun Khare Beichuan Zhang
{vkhare, bzhang}@cs.arizona.edu
University of Arizona

***Abstract*—Content Delivery Networks (CDN) are overlay network of servers being used to deliver growing traffic demands on the Internet. As a result, CDNs are facing ever-increasing operating costs. Internet Service Providers (ISP) charge CDNs on server traffic, computed using common usage-based charging models, e.g. 95th Percentile charging. We propose Network Cost Aware Request Routing, *NetReq*, that assign user requests to reduce server charging volume. We compare *NetReq* against nearest-available server request routing in large scale simulations for both web and multicast traffic requests. *NetReq* reduces charging volume for both traffic request types, thereby reducing cost. *NetReq* provides comparable network performance for multicast traffic by introducing end-to-end delay as a constraint in the request-routing. *NetReq* marginally increases network performance for web traffic, when content maybe available at every server.**

I. Introduction

CDN is an overlay network of servers offering reduced access latency for requested content. CDN redirects users to servers and pay ISPs for traffic generated by servers. The ISPs compute server charging volume using Percentile-based charging, and translate it to monetary cost. This operational ISP cost is the dominant factor affecting pricing of CDN services and therefore can impact commercial competitiveness of CDN.

The research challenge is to design a CDN Request Routing that reduces ISP cost. Traditional CDN Request Routing improves application performance metrics such as network delay [7], server load [6] and system throughput [20], but incurs significant ISP cost. In contrast CDN Request Routing in [10] reduces ISP cost by exploiting *concave* nature of ISP charging functions. In this paper, we explore ways to reduce CDN server charging volume as a means to reduce ISP cost for CDN.

The ISP cost and application performance are orthogonal metrics that cannot be optimized simultaneously. Prior research has presented solutions that optimize particular metric and treats other metric as constraint. These solutions are proposed for different but closely related networking problems. GFA [8] distributes traffic over multiple provider links of a multi-homed network to optimize network performance under certain cost constraints. Entact [23] distributes traffic over hundreds of provider and peer outgoing links from data centers of online service provider network after allowing operator to choose an optimal point on performance-cost trade-off. GFA and Entact solutions may seem applicable for CDN as each server is a logical link where user requests are redirected.

Specific characteristics of CDN make prior solutions a non-starter. First, CDNs manage geographically distributed infrastructure with servers deployed in various ISPs. In contrast, GFA performs TE for local provider links of a multi-homed network and Entact performs TE for data centers that are globally distributed but hardly as diverse as a CDN infrastructure. Second, CDNs expect user requests to be geographically distributed and deploy servers close to edge to reduce network delay [13]. In contrast, GFA manages traffic for multi-homed network where traffic request source is local and Entact manages traffic for data centers where traffic requests are distributed but offer higher delay due to increased distance between server and user.

The architectural richness of CDN allow any user to be redirected to multiple server choices deployed in different ISPs. The request routing needs to consider ISP cost and network performance trade-offs associated with these choices. The size of CDN magnifies the number of choices available thereby making prior solutions unscalable. The number of servers deployed for a medium to large scale CDN ranges in thousands [7], [15], which is three orders of magnitude more than provider links of multi-homed networks and two orders of magnitude more than the number of external links for online service provider networks.

The CDN Request Routing needs to reduce ISP cost under performance constraints after considering various user-server assignment choices. The request routing controls how traffic is distributed amongst servers. The ISPs compute server charging volume based on that traffic. Therefore request routing can reduce ISP cost by reducing server charging volume. We explore ways to exploit 95th percentile charging model used by ISPs to compute charging volume. The main observation is that under 95th percentile ISP do not charge for 5% charging intervals, which we refer to as available burst intervals. Each server has available burst intervals depending upon its ISP charging.

We propose greedy heuristics to use available burst intervals to reduce charging volume. First, server bandwidth during available burst interval can absorb any surge in traffic demand without incurring cost. For instance, say 2 servers need to support surge in traffic demand for 10% of charging intervals.

The material in this article is based upon the work partially supported by the National Science Foundation under Grants No. 1039615 and by Open Project of Shenzhen Key Lab of Cloud Computing Technology & Applications (SPCCTA). Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the sponsors.

Consider a scheme where user requests are assigned to one server for 5% and to second server for remaining 5% of charging intervals. Under 95th percentile both servers avoid incurring any cost for extra traffic.

Second, server bandwidth during available burst interval can absorb regular traffic demand to reduce cost. For instance, say 20 servers are charged at 95th percentile then total number of available burst interval equals total number of charging intervals. In effect for every charging interval there exists a server available burst interval that can absorb portion of regular traffic demand without incurring cost.

We devise a CDN Request Routing, called *NetReq* combines proposed greedy heuristics. *NetReq* computes server traffic assignment that reduces charging volume after factoring in available burst intervals. *NetReq* imposes server traffic assignment in the form of bandwidth capacity constraints. Thereafter *NetReq* assigns user requests with preference for application performance under cost-saving bandwidth capacity constraints.

In evaluation, we use simulations to compare *NetReq* against nearest-available routing [7], [15]. *NetReq* reduces CDN servers charging volumes that translates into ISP cost reduction. *NetReq* presents comparable network performance for end users requesting streaming content facilitated by an overlay tree over CDN infrastructure. *NetReq* marginally increases network delay for users in the case where web content is available at each server.

The rest of paper is organized as follows: In Section II, we introduce CDN network and ISP charging model and motivate need to reduce ISP cost. Section III presents CDN Request Routing problem formulation. Section IV presents techniques to reduce server charging volume. Section V presents *NetReq* Request Routing. We present methodology and evaluation results in Section VI. Section VII presents an overview of related work. Section VIII concludes with discussion of future work.

II. ISP Charging and CDN Network Model

ISPs charge business customers for accessing their network. The cost incurred is usually based on customer traffic volume, i.e., cost = c(x) where x is charging volume and c is ISP charging function that maps charging volume to cost.

ISP charging function is proprietary information and not known publicly. But overall trend of ISP charging function reflects general pricing practice of decreased unit cost as purchased bandwidth increases. In [24], ISPs are reported to charge their customers using concave charging functions. In [8], various ISPs charging functions are reported as being complex and piece-wise linear but still follow general pricing trend.

ISPs also need to determine charging volume for traffic sent by its customers over a charging period, which is usually a month. There are two popular ways to determine charging volume: Percentile-based and Total-volume based. In former approach traffic volume is recorded for every 5 minute interval during charging period and over that ISP specified q*th* percentile is used as charging volume. In latter approach total traffic volume generated by customer is charging volume. In the case

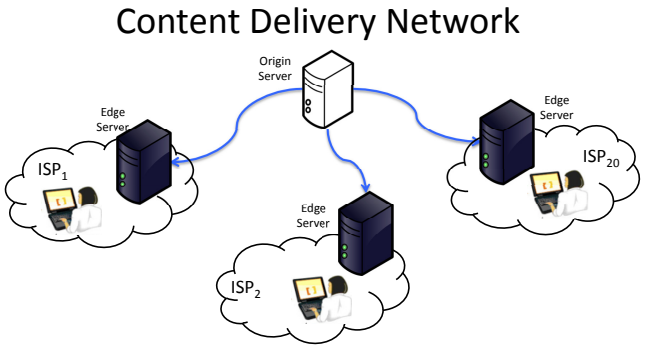

Fig. 1. Content Delivery Network with users being assigned to Edge Severs

of CDNs, ISP are reported to set prices per network port, using 95*th* percentile traffic for charging [16].

Content Delivery Network is composed of three main components: a content delivery infrastructure, a request routing algorithm and a distribution mechanism [17]. The content delivery infrastructure is composed of dedicated servers with reserved bandwidth assignment, that are deployed in strategic locations on the Internet such as Points of Presence (PoP) of various ISPs [7], [15]. The request routing algorithm redirects user requests to a joining server. The distribution mechanism depends upon the specific application for which content is being requested, i.e., for static web content the joining server either has the requested content or can access it at the origin server but for streaming web content the joining server must join an overlay multicast tree to access requested content. Several protocols have been designed to build such overlay multicast tree, e.g., OMNI [4], AMCast [18], HMTP [21] and Narada [5].

In this paper we focus upon the request routing algorithm of a CDN. Commercial CDNs have adopted *"nearest-available"* routing [7], where user is assigned to nearest server with available bandwidth. There are several other variants of nearest-available routing that consider factors such as network conditions [17], reliability [20] and server load [6].

III. *NetReq* Request Routing Problem

We motivate need to reduce ISP cost by showing that optimizing for network performance alone can result in higher cost. We show by example the impact of Request Routing on ISP cost under percentile-based charging.

Consider Figure 1 with CDN having 20 servers in 20 separate ISPs, each having a user. And the traffic demand is such that only one user requests content in any given interval. The nearest-available routing assigns user to server within same ISP, causing each server to have charging volume of one and overall cost to be 20. In another approach users in every interval can be assigned to one server, allowing that server to have charging volume of one and reducing cost to one. In yet another approach, users can be assigned to one server for 5% intervals and then to another server for next 5% intervals and so on, causing every server to have zero charging volume and reducing cost to zero.

TABLE I
NOTATION FOR USER ASSIGNMENT PROBLEM FORMULATION

K	Servers deployed in various ISPs. Use k as index.
B	Bandwidth reserved for servers
I	Time intervals in a charging period. (i is interval index)
c_k	ISP Charging function for SRV_k.
q_k	ISP Charging percentile for SRV_k e.g. q_k = 0.95 when ISP charges at 95th-percentile.
f_k	Available burst intervals for $SRV_k = (1 - q_k) \cdot I$
$u_j^{[i]}$	user u_j bandwidth consumed in interval i. (j is user index)
$v^{[i]}$	User Traffic Demand in interval i = $\sum_j u_j^{[i]}$ Time series V = $\{v^{[i]} \mid 1 \leq i \leq I\}$.
$u_{j,k}^{[i]}$	Assignment of user u_j to SRV_k in interval i.
$t_k^{[i]}$	Traffic volume assigned to SRV_k in interval i = $\sum_j u_{j,k}^{[i]} \cdot u_j^{[i]}$. Time series $T_k = \{t_k^{[i]} \mid 1 \leq i \leq I\}$
qt(X, q)	The $\lceil q \cdot \mid X \mid \rceil$ -th value in X_{sorted} (or 0 if q $\leq$ 0)
p_k	Charging volume of SRV_k, (i.e. p_k = qt(T_k, q_k)).
V_o	Sum of charging volumes of the servers $\sum_k p_k$.

Problem Specification: We introduce notation in Table I and formally state the problem. Find traffic volume assignment $t_k^{[i]}$ for every server that

$$minimize \quad ISP\ cost \quad \sum_{k=1}^{K} c_k(p_k)\ for\ CDN \tag{1}$$

$$s.t. \qquad v^{[i]} = \sum_{k=1}^{K} t_k^{[i]} \quad \forall\ intervals\ i \tag{2}$$

$$t_k^{[i]} \leq B \quad \forall\ SRV_k\ \forall\ intervals\ i \tag{3}$$

Sum of Charging Volume of Servers: The sum of server charging volume, $\sum_k p_k$, controls ISP cost for facilitating any user traffic demand. It is based on the observation that ISP cost has a monotonicity property with respect to sum of server charging volume. This monotonicity property suggests that to minimize ISP cost $\sum_{k=1}^{K} c_k(p_k)$, we need to minimize sum of server charging volume $V_o = \sum_{k=1}^{K} p_k$. *NetReq* focuses on reducing sum of server charging volume to reduce ISP cost.

IV. *NetReq* CHARGING VOLUME OF SERVERS

We present techniques to reduce sum of charging volumes using available burst intervals of servers. We present both offline and online algorithms that combine these techniques to reduce sum of charging volumes of servers and in the process we compute the charging volume of individual servers.

A. Techniques to Reduce Sum of Charging Volumes

NetReq reduces the sum of charging volumes by distributing traffic amongst the available burst intervals of servers. The available burst intervals of a server are the $f_k = (1 - q_k) \cdot I$ charging intervals for which ISP does not charge, e.g., when ISP uses 95th percentile traffic as charging volume then there are $f_k = 5\% \cdot I$ such charging intervals where traffic can be burst without incurring any charges. We present separate techniques which take advantage of the available burst intervals to reduce the sum of charging volumes: (1) Absorbing Peak Traffic Demand and (2) Rotating Traffic Demand.

1) Absorbing Peak Traffic Demand: CDN can reduce ISP cost for surge in traffic demand by assigning it to servers during their available burst intervals. Essentially, Absorption of Peak Traffic Demand is equating the surge in traffic demand during given intervals with the servers' bandwidth during their burst intervals.

The sum of charging volumes, $V_o = qt(V, 1 - m)$, is minimized when m, the number of charging intervals for which traffic demand can be absorbed by available burst intervals, is maximized. The value of m depends upon total number of available burst intervals and available bandwidth of servers in those intervals. Consider when there exists no bandwidth constraints on servers, then m is maximized to $F = \sum_k f_k$. So a CDN with 5 servers being charged at 95th percentile traffic volume can reduce its V_o to 75th percentile of traffic demand. This happens because each server can absorb total traffic demand during its burst interval.

Algorithm 1 Absorption of Peak Traffic Demand

```
i = 1; assignable = false;
while assignable is false AND 1 ≤ i ≤ I do
    V_est = v^[i] {V_est is immediate estimate for V_o}
    assignable = isTrafficAssignable(V_est)
end while
V_o = V_est
isTrafficAssignable (V_est)
for all v^[j] in 1 ≤ j ≤ I s.t. v^[j] > V_est do
    pv^[j] = v^[j] - V_est {peak traffic volume to be absorbed}
    ns^[j] = ⌈pv^[j]/B⌉ {No. of servers' burst intervals needed}
    if (decrement f_k for ns^[j] servers) fails
        return false
end for
return true
```

We present absorption of peak traffic demand, in Algorithm 1, that minimizes V_o where server bandwidth capacity constraints do exist. Due to bandwidth capacity constraints more servers with their burst intervals are needed to absorb surge in traffic demand. The minimum value of V_o is searched iteratively in $V = \{v^{[i]} \mid 1 \leq i \leq I\}$. For any estimate of V_o, i.e. $V_{est} = v^{[i]}$, we check whether traffic is assignable for every interval. During intervals with $v^{[j]} \leq V_{est}$ there is no consumption of available burst intervals of servers. But for interval with $v^{[j]} > V_{est}$, the available burst intervals of $ns^{[j]} = \lceil pv^{[j]}/B \rceil$ servers have to absorb peak traffic of $pv^{[j]} = v^{[j]} - V_{est}$. The least value of V_{est} with traffic assignment in every interval is the minimum value of V_o.

2) Rotating Traffic Demand: CDN can also reduce ISP cost for serving regular traffic demand by assigning it to servers with burst interval. Consider a set of 20 servers, each being charged at 95th percentile traffic volume, then the total number of available burst intervals of these servers equals the total number of charging intervals. In this special case, for every interval there exists at least one server with burst interval whose bandwidth can be used to absorb regular traffic demand.

Essentially, Rotating Traffic Demand maximizes absorption of regular traffic demand by servers in every charging interval.

Rotating traffic demand attempts to maximize traffic that can be absorbed in every interval without impacting server charging volume. Lets assume regular traffic demand to be $RTD \simeq v^{[i]}$ in every interval i. A certain portion of RTD, lets say absorbed traffic demand ATD, in every interval is carried by servers with burst intervals. Since total burst intervals of K servers $F = \sum_{k=1} fk \geq I$, multiple servers can contribute towards ATD in every interval. The maximum number of servers that can absorb RTD in each interval is z = $\lfloor F/I \rfloor$. We dimension server bandwidth to carry static traffic volume during every interval and carry extra traffic contributing towards ATD during burst interval. So if the static traffic is x for each server, then $B - x$ is traffic from z servers for ATD.

$$K \cdot x + z \cdot (B - x) = RTD \tag{4}$$

$$x = \frac{RTD - z \cdot B}{K - z} \tag{5}$$

The sum of charging volume of servers, V_o is $K \cdot x$ and charging volume of individual servers, p_k is x.

B. NetReq Server Management

NetReq classifies servers as peak and rotation servers based on how their burst intervals are used to reduce charging volumes. The burst intervals of peak servers are used to absorb surge in traffic demand. The burst intervals of rotation servers are used to absorb as much of the regular traffic demand.

NetReq pre-computes maximum number of rotation servers from the condition that sum of their burst intervals need to be a factor of total number of charging intervals. In case of 20 rotation servers, each being charged at 95th percentile traffic, total number of burst intervals equals number of charging intervals which means z=1 i.e. only one server can be used to absorb regular traffic demand in an interval. *NetReq* classifies S = $\lfloor K/20 \rfloor \cdot 20$ servers as rotation servers allowing z = $\lfloor K/20 \rfloor$ servers' bandwidth to be used for absorbing regular traffic demand in each interval. The remaining $K - S$ servers are classified as peak servers that absorb surge in traffic demand.

CDN deploys servers in AS locations where majority of user requests are expected. *NetReq* attempts to assign both peak and rotation servers in AS locations that are closer in terms of inter-AS delay. The user traffic requests will be redirected to peak and rotation servers when their available burst interval is being consumed to reduce charging volume. In case the peak and rotation server is distant to the user it will increase the last-hop delay for user requests.

C. NetReq Offline Computation of Charging Volumes

NetReq Offline, presented in Algorithm 2, combines both the techniques: (1) Absorbing Peak Traffic Demand and (2) Rotating Traffic Demand. *NetReq* Offline iteratively searches for minimum value of V_o within the search space $V = \{v^{[i]} \mid 1 \leq i \leq I\}$, sorted in non-decreasing order of $v^{[i]}$. *NetReq* Offline considers each V_{est}, estimate of V_o, as regular traffic demand to be absorbed by rotation servers. Any traffic demand greater than V_{est} in any interval is considered as surge to be absorbed by peak servers. In case traffic volume is not assignable, V_{est} is increased to reduce traffic load on peak servers and also reduce number of intervals for which peak servers absorb surge traffic. But increasing V_{est} increases traffic load on rotation servers, leading to increase in server charging volume.

The minimum value of V_{est} equals V_o when traffic volume is assignable amongst peak and rotation servers. The charging volume for rotation and peak servers is computed based on V_o traffic assignment. The runtime of *NetReq* Offline algorithm is $O(I^2)$.

Algorithm 2 *NetReq* Offline computation of V_o

S = $\lfloor K/20 \rfloor \cdot 20$ {Total No. of Rotation Servers}
z = $\lfloor F/I \rfloor$ {z Servers to absorb RTD in every interval}
Offline V_o computation
i=1; assignable = false;
while assignable is false AND 1 $\leq$i $\leq$ I **do**
  RTD = $v^{[i]}$
  x = $(RTD - z \cdot B)/(K - z)$ {x Static Traffic Volume}
  p_k = x {Charging Volume of each Server}
  $\forall$ K-S peak servers, set $f_k = (1 - q_k) \cdot I$
  assignable = isTrafficAssignable(RTD)
end while
$V_o = K \cdot x$

D. NetReq Online Computation of Charging Volumes

NetReq Offline assumes the user traffic demand for each charging interval is known in advance. However in practice user traffic demand for every charging interval is not known in advance. In this section we present *NetReq* Online Algorithm 3 to compute and maintain reduced charging volume of servers for any given real-time change in user traffic demand.

Algorithm 3 Online V_o calculation

$burst^{[i]}$ = server burst intervals used to support $v^{[i]}$
V_o = $v^{[j]}$ {immediate minimum value of V_o}
$burst_{avail}$ = $burst^{[1]}$ {burst intervals consumed for $v^{[1]}$}
$burst_{need}$ = $burst^{[I+1]}$ {burst intervals consumed for $v^{[I+1]}$}
Online V_o Computation
if $burst_{avail} < burst_{need}$ **then**
  while assignable is false AND j+1 leq i $\leq$ I+1 **do**
    V_{est} = $v^{[i]}$
    $\forall$ peak servers $f_k = (1 - q_k) \cdot I$
    assignable = isTrafficAssignable(V_{est})
  end while
  $V_o = V_{est}$
end if

With change in user traffic demand we need to update V_o to maintain reduced charging volume. In case the immediate value of V_o is underestimated it will cause burst intervals of servers to exhaust too early without being able to absorb traffic

demand for all charging intervals. This will force traffic demand in those intervals to be part of the charging volume of servers, i.e., CDN will have to pay ISPs for it. We maintain a sliding window of length equal to size of charging period, which is usually one-month, and after every single day's worth of traffic demand we update V_o and charging volume of each server.

NetReq Online processes day's user traffic demand using most recent value of V_o. *NetReq* Online checks whether V_o and charging volume of servers needs to be updated due to present day's user traffic demand. *NetReq* Online attempts to match server burst intervals used to support first day's traffic demand, i.e. $v^{[1]}$, with server burst intervals used to support present day's user traffic demand, i.e. $v^{[I+1]}$. In case more server burst intervals are needed to support $v^{[I+1]}$ then V_o and charging volume of servers needs to be updated. Whenever value of V_o needs to be updated it is again searched iteratively using *NetReq* Offline Algorithm but starting from $v^{[I+1]}$.

V. *NetReq* Request Routing

In this section, we present *NetReq* Request Routing algorithm for web traffic and multicast streaming applications. *NetReq* Offline and Online algorithms are used to compute charging volume of servers that reduces ISP cost for CDN. The reduction in ISP cost is guaranteed as long as traffic assignment by Request Routing algorithm maintains server charging volume computed by *NetReq*. *NetReq* Request Routing takes advantage of this fact to improve network performance for users while reducing ISP cost for CDN.

A. Web Traffic Request Routing

NetReq Request Routing redirects user requests while adhering to cost-efficient server charging volume and thereafter providing best network performance. The charging volume, p_k, of a server means for $q_k \cdot I$ intervals server can deliver p_k traffic volume and for $(1-q_k) \cdot I$ intervals server can deliver B traffic volume. *NetReq* sets bandwidth capacity constraints, as either p_k or B, depending upon whether server is being used to burst traffic or not. *NetReq* Request Routing redirects web traffic requests to nearest-available server with available bandwidth. The nearest server is determined by network proximity for redirecting web traffic requests.

B. Multicast Traffic Request Routing

Overlay multicast protocols are involved in (1) assigning users to servers (2) organizing participating servers of a multicast group into a dissemination tree to deliver content from root to end users and (3) maintaining the overlay tree with changing user membership. Overlay tree construction can be performed using any of the following protocols: OMNI, HMTP, NICE and AMCast depending upon particular objective function being optimized. *NetReq* Request Routing for multicast traffic deals with the issue of handling requests from surfers, for short session duration, and viewers, for extended duration, that have been reported to be present in recent group membership studies [19], [9] conducted over live streaming workload from large content providers such as Akamai.

Algorithm 4 User Assignment

```
pUser = new user request consuming b bandwidth
pUser.nearestSRV = Servers ordered by overall delay (overlay tree + last-hop delay)
for all pSRV ∈ pUser.nearestSRV do
    User assignment preference:
    (1) peak traffic (if pSRV is peak server)
    (2) rotating traffic (if pSRV is rotation server)
    (3) static traffic
end for
```

NetReq Request Routing for multicast traffic, as presented in Algorithm 4, treats every new user join requests as surfer. *NetReq* redirects surfer requests to nearest peak servers which can send another stream of data to user without violating its bandwidth capacity constraint. But cases may emerge where either peak servers are saturated or during certain intervals there are no peak servers available for absorbing the user requests. In such cases *NetReq* redirects surfer requests to nearest rotation server with available bandwidth. The nearest server is determined by overall delay, i.e., the overlay tree delay and last-hop delay for redirecting multicast traffic requests.

Algorithm 5 User Movement

```
DelayCompliantSRV = Top 5 SRV in pUser.nearestSRV
SRV_x marked for user movement
Find SRV_y nearest SRV_x s.t. SRV_y.treeDelay is minimum
for all pUser underneath SRV_x do
    if pUser.delay > SRV_y.treeDelay + dist(pUser, SRV_y) OR
    SRV_y ∈ DelayCompliantSRV then
        pUser Move to SRV_y
    end if
end for
```

NetReq adopts a user movement strategy to handle changes in group membership with the objective of maintaining servers' charging volume. The user movement occurs in following scenarios: (i) user movement between rotation servers and (ii) user movement from peak servers to rotation servers. The users on a server are moved only when available burst intervals of server has been exhausted.

User movement is implemented in a distributed fashion through local interactions between servers that are physically close to each other. For every rotation server marked for user movement we find candidate server that is closest to it and offering minimum overlay tree delay. Thereafter attempts are made to move user users from marked rotation server to candidate servers with two conditions in place: (i) improve the overall network performance of user being moved and (ii) candidate server be within a certain number of hops from moved user. Each server maintains local users in a FIFO list to ensure that users with least duration, i.e. local surfers, are given preference when it comes to user movement between different servers.

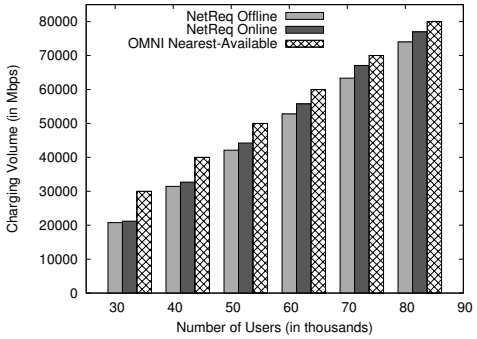

Fig. 2. Charging Volume for Multicast Traffic

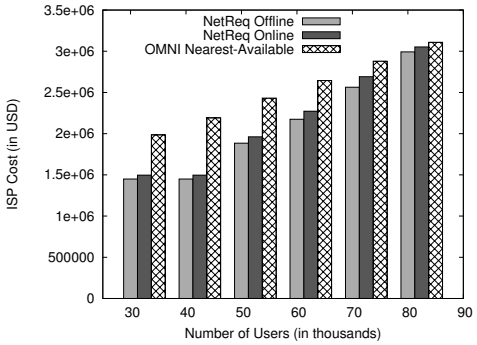

Fig. 3. ISP cost with each server being charged by same ISP charging function (Multicast Traffic)

VI. Evaluation

Servers and users are present within ASes where server deployment is known apriori but users can request content within any AS. To simulate realistic CDN we take AS-level topology [3] and attach servers and users to various ASes. We construct a latency map that captures delay between servers and users, from reported delay information of inter-AS links available at iPlane [14].

We distribute users amongst various ASes following a Zipf distribution. Group membership studies [19], [9] report existence of spatial properties such as clustering and diversity in user population of multicast groups. Clustering points at skew in user population and diversity points at large number of distinct locations where popular sessions are accessed. The Zipf distribution captures both these properties. We distribute servers amongst top AS locations consisting mostly of Tier-1 and Tier-2 networks.

We compare *NetReq* against nearest-available routing for both web and multicast traffic on various metrics: server charging volume, ISP cost and user network performance. We generate web traffic load by (a) fixing number of users requesting content in every interval within a range to capture both regular and surge in traffic demands and (b) choosing random users to request content. We generate multicast traffic by splitting group lifetime into phases: (a) join phase where random users join (b) stable phase having no change in membership and (c) leave phase where random users leave.

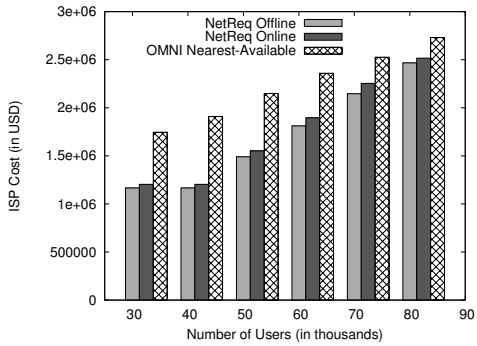

Fig. 4. ISP cost with each server being charged by different ISP charging function (Multicast Traffic)

A. Charging Volume

CDN Request Routing controls server traffic and therefore impacts server charging volume. Figure 5 and 2 compare sum of server charging volume for increasing amount of web and multicast traffic loads. *NetReq* reduces sum of server charging volume for both traffic types when compared to nearest-available routing.

The nearest-available routing is unable to control server charging volume without exploiting available burst intervals. In nearest-available routing user locality and user traffic demand determines server charging volume. Due to clustering in user locality, traffic demand can keep getting assigned to same servers thereby hindering any opportunity to exploit server burst intervals. Furthermore surge in traffic demand within a region can cause nearby servers to saturate causing unnecessary increase in overall charging volume. With nearest-available routing, CDNs are unable to control user locality and therefore unable to exploit server burst intervals to reduce cost.

NetReq offline and online reduce server charging volume by using available burst intervals. *NetReq* identifies peak servers where traffic can be burst for limited intervals and rotation servers amongst which traffic can be burst in every interval. *NetReq* uses server burst intervals to absorb surge and regular traffic demands. *NetReq* allows CDN to reduce server charging volume without controlling user locality and traffic demand.

Figure 2 and 5 show that difference between *NetReq* and nearest-available routing in server charging volume is significant for small traffic load as compared to large traffic loads. Reduction in server charging volume depends upon (a) number of available burst intervals and (b) available bandwidth during those burst intervals. The number of available burst intervals are same for every traffic load but available bandwidth decreases with increasing traffic load. As available bandwidth becomes limited the reduction in overall charging volume also becomes limited.

B. Savings in ISP Cost

CDNs want to control ISP cost since it impacts content delivery pricing. Some CDNs, such as Akamai, are able to negotiate transit contracts with ISPs on a national level [16].

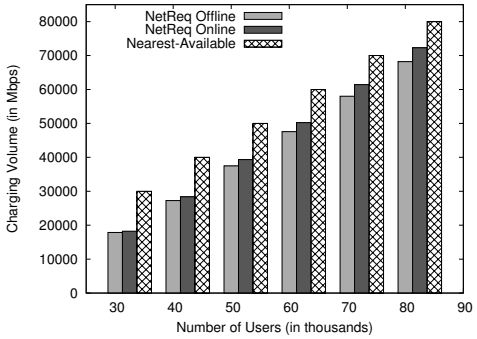

Fig. 5. Charging Volume for Web Traffic

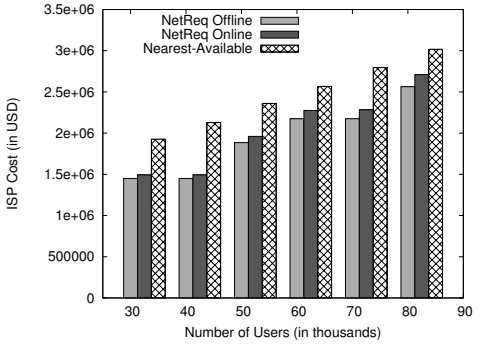

Fig. 6. ISP cost with each server being charged by same ISP charging function (Web Traffic)

Therefore we compare ISP cost for following scenarios: (1) CDN servers charged using a nationwide charging function in Figure 3 and 6 and (2) CDN servers charged using separate ISP Charging function in Figure 7 and 4 for multicast and web traffic respectively. *NetReq* does not explicitly minimize ISP cost but presents lowest ISP cost for both traffic types as a consequence of reduction in server charging volume.

C. User Network Performance for Multicast Traffic

CDNs are being used commercially to deliver live streaming content on Internet-scale to users [1]. The network performance offered by CDN can impact the decision of content providers to use a CDN for delivering their content. *NetReq* redirect user requests to delay-efficient servers under cost-efficient bandwidth constraints. *NetReq* uses server join protocol of HMTP [22] to build overlay tree. In comparison, nearest-available server minimizes last-hop delay and OMNI [4] protocol is used for overlay tree. OMNI protocol attempts to minimize average user delay through local overlay tree transformations whenever change occurs in network conditions and user memberships. Figure 8 compares average user delay for *NetReq* against nearest-available during every 5-minute charging interval.

NetReq offline and online assign users to server offering lowest overall delay. This presents a more holistic approach at improving user network performance. The user delay can be split into two parts: (i) last-hop delay between user and CDN server and (ii) overlay tree delay between origin and

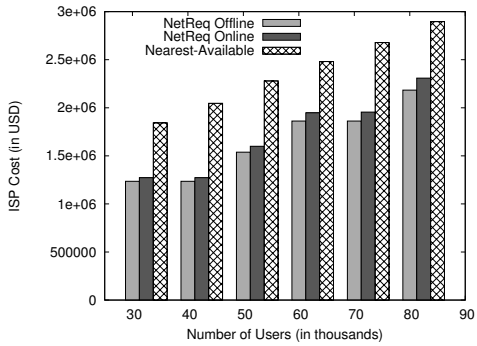

Fig. 7. ISP cost with each server being charging by different ISP charging function (Web Traffic)

CDN server. The overlay tree delay dominates user delay and depends upon number of overlay hops from origin to CDN server. *NetReq* marginally increases last-hop delay by not redirecting to nearest-available server. However, overlay tree delay is smaller as user is redirected to CDN server much closer to origin in the overlay tree. In contrast, nearest available server offers minimum last hop delay but overlay tree delay remains sub-optimal that increases overall delay for users.

We observe that *NetReq* Offline offers slightly better network performance for users when compared to *NetReq* Online. The reason being adjustment in value of V_o needed for drastic changes in user traffic demand. But in every time snapshot user network performance with *NetReq* online is still better than nearest-available, suggesting that *NetReq* Online can handle real-time group membership changes.

D. User Network Performance for Web Traffic

CDNs are also used by content sites such as YouTube, Hulu and CNN to deliver popular web content to numerous users [2]. We compare *NetReq* and nearest-available routing for popular content. Figure 9 compares network performance of *NetReq* against nearest-available routing for user traffic requests in every 5-minute charging intervals. Since the requested content is available at every server, user delay is essentially last-hop delay between joining server and user. In such scenario, nearest available server offers best network performance for users due to its network proximity. Both *NetReq* Online and Offline can only increase user delay by attempting to balance trade-off between ISP cost and network performance.

NetReq marginally increases network performance of users as compared to nearest-available routing. *NetReq* server classification ensures that there are peak and rotation servers available within every region. Therefore user requests are either served as static traffic volume of nearby servers or when needed served as rotating or peak traffic volume of servers within the region. The only difference is that peak and rotation servers within a region are marginally farther away from nearest available server.

VII. Related Work

In recent times Internet-scale dissemination content is being achieved through CDNs where dedicated servers act as proxies

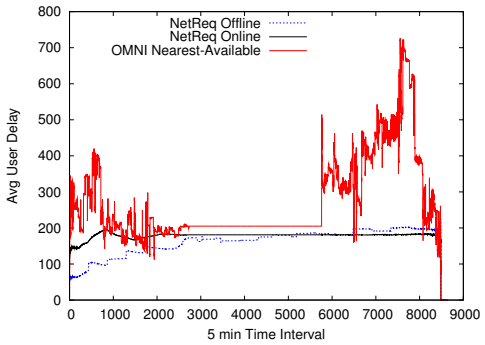

Fig. 8. User Network Performance comparison for Multicast Traffic

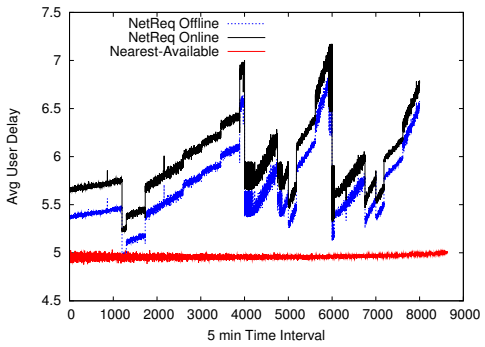

Fig. 9. User Network Performance comparison for Web Traffic

facilitating the multicast groups. The advantage of CDN is that users send or receive only one copy of data packets during session, and the work of duplicating packets is shifted from data sources to servers. Researchers [11], [12] consider deployment of such architectures as a network design problem.

Our work relates to recent work on exploring the trade-off between network access cost and delay performance. Goldenberg et. all, in [8], propose a number of algorithms which assign the flows of multi-homed network to provider links to optimize network access cost or provide good network performance under given cost constraints. Zhang et. all, in [23], propose traffic engineering solutions for online service provider networks having data centers connecting to hundreds of ISPs to find the optimal sweet spot in cost-performance tradeoff. Our work differs from these algorithms in a few major ways. First we look at the CDN Request Routing problem where the choices for traffic assignment are orders of magnitude greater than multi-homed network and data centers. Second, we exploit the nature of the specific Percentile-based charging models used by ISPs to compute their charging bill.

Traditional CDN Request Routing algorithms [7], [6] are designed with the objective of improving application performance metrics without considering the ISP cost they may incur. Our work provides CDN the opportunity to explore the trade-off between ISP cost and network performance of users.

VIII. Conclusion and Future Work

In this paper we propose different techniques to reduce the charging volume of servers. The evaluation shows that reducing charging of servers reduces the ISP cost for CDN. Using simulations based on realistic server and user distribution and ISP charging functions, we show that *NetReq* Request Routing can effectively minimize ISP cost while providing good network performance to end users. There are other avenues for future work. In this paper, we focus on algorithmic design and evaluation through simulation. A natural next step is to implement these algorithms on an actual CDN.

References

[1] 2010 ncaa march madness live and on demand. http://mmod.ncaa.com/.

[2] Google relies on akamai to stream youtube live: 700k concurrent viewers. http://www.YouTubeLive.

[3] Internet topology collection. http://irl.cs.ucla.edu/topology.

[4] S. Banerjee, C. Kommareddy, K. Kar, B. Bhattacharjee, and S. Khuller. Construction of an efficient overlay multicast infrastructure for real-time applications. In *Proceedings of IEEE INFOCOM*, 2003.

[5] Y.-H. Chu, S. G. Rao, and H. Zhang. A case for end system multicast. In *Proceedings of ACM Sigmetrics*, 2000.

[6] R. D. Day. Meta content delivery network. *US Patent 7185052*, 2007.

[7] J. Dilley, B. Maggs, J. Parikh, H. Prokop, R. Sitaraman, and B. Weihl. Globally distributed content delivery. *IEEE Internet Computing*, 6(5):50–58, 2002.

[8] D. K. Goldenberg, L. Qiuy, H. Xie, Y. R. Yang, and Y. Zhang. Optimizing cost and performance for multihoming. *SIGCOMM Comput. Commun. Rev.*, 34(4):79–92, 2004.

[9] J. hong Cui, M. Faloutsos, D. Maggiorini, M. Gerla, and K. Boussetta. Measuring and modelling the group membership in the internet. In *Proceedings of Internet Measurement Conference (IMC)*, 2003.

[10] V. Khare and B. Zhang. Towards economically viable infrastructure-based overlay multicast network. *IEEE INFOCOM*, 2009.

[11] L. Lao, J.-H. Cui, and M. Gerla. Multicast service overlay design. In *Proceedings of SPECTS*, 2005.

[12] L. Lao, J.-H. Cui, and M. Gerla. Toma: A viable solution for largescale multicast service support. In *Proceedings of IFIP*, 2005.

[13] T. Leighton. Improving performance on the internet. *Commun. ACM*, 52:44–51, February 2009.

[14] H. V. Madhyastha, T. Isdal, M. Piatek, C. Dixon, T. Anderson, A. Krishnamurthy, and A. Venkataramani. iPlane: an information plane for distributed services. In *Proceedings of the 7th symposium on Operating systems design and implementation*, OSDI '06, pages 367–380, 2006.

[15] G. Pallis and A. Vakali. Insight and perspective for content delivery networks. In *Communications of the ACM*, 2006.

[16] A. Qureshi, R. Weber, H. Balakrishnan, J. Guttag, and B. Maggs. Cutting the electric bill for internet-scale systems. *SIGCOMM Comput. Commun. Rev.*, 39(4):123–134, 2009.

[17] D. M. Shaw. Global hosting system. *US Patent 6108703*, 2000.

[18] S. Shi and J. Turner. Routing in overlay multicast networks. In *Proceedings of IEEE INFOCOM*, 2002.

[19] K. Sripanidkulchai, B. Maggs, , and H. Zhang. An analysis of live streaming workloads on the internet. In *Proceedings of Internet Measurement Conference (IMC)*, 2004.

[20] L. W. Vivek, L. Wang, V. Pai, and L. Peterson. The effectiveness of request redirection on cdn robustness. In *In Proc. 5th OSDI*, pages 345–360, 2002.

[21] B. Zhang, S. Jamin, and L. Zhang. Host multicast: A framework for delivering multicast to end users. In *Proceedings of IEEE Infocom*, 2002.

[22] B. Zhang, W. Wang, S. Jamin, D. Massey, and L. Zhang. Universal ip multicast delivery. *Journal on Computer and Telecommunications Networking*, 2006.

[23] Z. Zhang, M. Zhang, A. Greenberg, Y. C. Hu, R. Mahajan, and B. Christian. Optimizing cost and performance in online service provider networks. In *Proceedings of NSDI'10*, Berkeley, CA, USA. USENIX Association.

[24] Y. Zhu, C. Dovrolis, and M. Ammar. Combining multihoming with overlay routing. In *Proceedings of IEEE INFOCOM*, 2007.