Noname manuscript No.
(will be inserted by the editor)

A Hybrid Edge-Cloud Architecture for Reducing On-Demand Gaming Latency

Sharon Choy · Bernard Wong · Gwendal Simon · Catherine Rosenberg

Received: INSERT DATE / Accepted: INSERT DATE

Abstract The cloud was originally designed to provide general purpose computing using commodity hardware and its focus was on increasing resource consolidation as a means to lower cost. Hence, it was not particularly adapted to the requirements of multimedia applications that are highly latency sensitive and require specialized hardware, such as graphical processing units.

Existing cloud infrastructure is dimensioned to serve general purpose workloads and to meet end-user requirements by providing high throughput. In this paper, we investigate the effectiveness of using this general-purpose infrastructure for serving latency-sensitive multimedia applications. In particular, we examine on-demand gaming, also known as cloud gaming, which has the potential to change the video game industry. We demonstrate through a large-scale measurement study that the existing cloud infrastructure is unable to meet the strict latency requirements that is necessary for acceptable on-demand game play. Furthermore, we investigate the effectiveness of incorporating edge servers, which are servers located near end-users (e.g. CDN servers), to improve

Sharon Choy
University of Waterloo
Tel.: +1-519-888-4567 ext 36641
E-mail: s2choy@uwaterloo.ca

Bernard Wong
University of Waterloo
Tel.: +1-519-888-4567 ext 31301
E-mail: bernard@uwaterloo.ca

Gwendal Simon
TELECOM Bretagne
Tel.: +33-2-99-12-70-48
E-mail: gwendal.simon@telecom-bretagne.eu

Catherine Rosenberg
University of Waterloo
Tel.: +1-519-888-4510
E-mail: cath@uwaterloo.ca

end-user coverage. Specifically, we examine an edge-server only infrastructure and a hybrid infrastructure that consists of using edge servers in addition to the cloud. We find that a hybrid infrastructure significantly improves the number of end-users served. However, the number of satisfied end-users in a hybrid deployment largely depends on the various deployment parameters. Therefore, we evaluate various strategies that determine two such parameters, namely, the location of on-demand gaming servers and the games that are placed on these servers. We find that, through both a careful selection of on-demand gaming servers and the games to place on these servers, we significantly increase the number of end-users served over the basic random selection and placement strategies.

Keywords On-demand gaming · Cloud Support · Cloud Computing · Content Distribution Networks

1 Introduction

Cloud computing has recently become the predominant environment for hosting web applications and services. Its rapid adoption largely stems from the benefits of resource consolidation, such as higher resource utilization resulting in lower costs. Moreover, cloud computing is attractive for end-users as it provides more elastic resource acquisition, thereby diminishing the need for accurate growth forecasting. However, in order to maximize the benefits of resource consolidation, cloud providers dimension their infrastructure for high throughput and use commodity hardware to offer general purpose computing resources. These resources are found in a relatively small number of large datacenters. Furthermore, cloud datacenter locations are typically chosen to minimize cooling and electricity costs, rather than to minimize latency to end-users.

Unfortunately, these architectural decisions are not favourable to with the needs of an emerging class of multimedia applications that is interactive (hence highly latency-sensitive), and requires specialized hardware resources, such as graphical processing units (GPU) and fast memory. Therefore, these applications providers must either provide their own datacenters, or cloud computing providers (e.g. Amazon) must upgrade their existing infrastructure to include GPUs and fast memory. However, until GPUs are fully deployed in these cloud infrastructures, the average latency to a cloud-gaming capable datacenter will be higher than to a general-purpose datacenter. As a result, public cloud infrastructure may become a feasible, but perhaps suboptimal platform for hosting these applications.

In this paper, we select on-demand gaming as a representative case-study for this new class of applications. On-demand gaming, also known as cloud gaming, is a new video gaming application/platform. Instead of requiring end-users to have sufficiently powerful computers to play modern games, on-demand gaming performs the intensive game computation, including the game graphics generation, remotely with the resulting output streamed as a video back to the end-users. The shift from traditional gaming platforms, such as game consoles and desktop computers, to the cloud is the result of users wanting platform independence. For example, end-users may wish to play computationally intensive games on portable hardware, such as tablets, that do not have the required hardware.

The two main technical challenges to on-demand gaming pertain to latency and the need for servers with expensive, specialized hardware that cannot simultaneously serve multiple gaming sessions. By offloading computation to a remote host, on-demand gaming suffers from encoding latency, that is, the time to compress the video output, and network latency, which is the delay in sending the user input and video output back and forth between the end-user and the cloud. Although the video encoding latency will likely fall with faster hardware encoders, a significant portion of network latency is unavoidable as it is bounded by the speed of light in fibre. Past studies [21,22,32] have found that players begin to notice a delay of 100 ms [33]. However, 20 ms of this latency may be attributed to playout and processing delay [13]. Therefore, 80 ms is the threshold above which network latency begins to appreciably affect user experience. Because of this strict latency requirement, servers are restricted to serving end-users that are located in the same vicinity. In addition to low-latency, on-demand gaming requires specialized hardware such as GPUs. These reasons justify the need for a solution that distributes many specialized servers over the target coverage area as opposed to a centralized deployment.

To validate this statement, we first confirm the need for a new cloud paradigm by performing a large-scale measurement study consisting of latency measurements from PlanetLab and EC2 to more than 2,500 end-users. We find that EC2 is capable of providing a latency of 80 ms or less to fewer than 70% of our measured end-hosts. Additionally, we discover that a substantial increase in the total number of datacenters is required to significantly increase user coverage. These results, originally presented in [19], suggest that the existing cloud infrastructure is a poor platform for hosting highly latency-sensitive applications, as a sizeable portion of the population would experience significantly degraded quality of service. Furthermore, we find that number of served end-users increases by 28% when we add to the cloud a small number of edge servers that are co-located at Internet service provider (ISP) sites as part of a content distribution network (CDN) deployment [19].

In this work, we build on our previous findings in [19] and explore in detail alternative infrastructures such as an edge-server only deployment and a datacenter and edge-server hybrid deployment. We compare these two alternative infrastructures to a datacenter only deployment (i.e. the cloud solution). A datacenter can host a large number of games and serve multiple end-users at the same time. An edge server is limited in its hosting capacity (number of games it hosts) as well as in its serving capacity (due to the computational and GPU requirements of games). However, an edge-server only deployment will likely offer significantly better latency to *serve* users as edge servers are distributed within the service area. Since the maintenance and deployment cost of many edge servers is greater than that of a single datacenter, an edge-only deployment is probably not cost-competitive. To leverage the advantages provided by the edge-only and datacenter-only deployments, we propose a hybrid infrastructure, which consists of using edge servers in addition to the cloud, in order to increase the number of end-users served.

There are many factors to consider when using edge servers in a deployment. For example, as end-user demand varies by location, it may be cost prohibitive or unnecessary to deploy a significant number of edge servers in a single area. Also, as edge servers have limited storage capacity, we must effectively select which games to place on an edge server. To address this issue, we investigate various game-placement heuristics and present a voting based strategy that significantly improves the number of end-users served when compared to a random heuristic.

Overall, this work builds on our work in [19] and makes three new contributions. Firstly, we compare various infrastructures (datacenter-only, edge-server only, and hybrid) and determine their effectiveness at serving end-users. Secondly, we explore various heuristics for edge-server site selection and game placement and determine their effect on the ability of the system to serve end-users. Lastly, we quantify the impact of stricter latency requirements on the ratio of end-users served, and we evaluate the effectiveness of our hybrid deployment and our presented heuristics.

In Section 2, we provide an overview of related work that pertains to peer-assisted delivery architectures and the use of edge servers. Section 3 explains the latency that is experienced by the end-user. Furthermore, in Section 4, we show that the existing cloud infrastructure is unable to provide an acceptable on-demand gaming experience for a significant fraction of the end-users. We also explore the effectiveness of an edge-server-only deployment in Section 5 and a hybrid deployment in Section 6. We conclude our findings in Section 7.

2 Related Work

On-demand gaming is attractive to many end-users since it offers hardware independence, which allows end-users to play games on devices that do not support games, by offloading computation to the cloud. Providers and systems that provide on-demang gaming include OnLive [7], Gaikai [3], and GamingAnywhere [31]. Studies such as [17, 22, 32] demonstrate that short latency times are required in order to maintain an enjoyable user-experience; however, we show in this paper that end-users geographically distant from these datacenters experience unacceptable latency. The serious financial difficulties experienced by OnLive can be attributed to client dissatisfaction [34,50], which resulted in loss of customer revenue. Although there are many facets to the reasons of OnLive's failure (e.g. poor selection of games [52]), we focus on the technical challenge of reducing network latency, which is an inhibitor for the adoption of on-demand gaming.

Works such as [15,48,53], which are summarized in Table 1, aim to reduce latency for massively multiplayer online games (MMOGs). However, the requirements for MMOG infrastructure are significantly different from those of on-demand gaming. MMOGs may have more relaxed latency constraints due to their nature. For example, many MMOGs are role-playing games [10], which have more relaxed latency requirements than action games [21]. Moreover, graphics rendering, which can account for a significant amount of latency, normally occurs at the client. Unlike MMOG servers, on-demand gaming servers provide the platform in which games are rendered and executed, and these servers must receive input from the client and stream video in addition to coordinating multiple players.

Work	**Approach**
Süselbeck et al. [53]	Distribute MMOG server's functionality among several servers which can be co-located with the cloud.
Shelley and Katchabaw [48]	Optimistically predict the user action and continue execution of the game without waiting for user input.
Beskow et al. [15]	Migrate game region to a server that is closer to the end-user.

Table 1 Summary of Works that Reduce Latency in MMOGs

Many technologies for on-demand gaming are closely related to technologies for video streaming, because on-demand gaming consists of sending a video stream of the game back to the client. Current large-scale video-on-demand delivery solutions include dynamic datacenter provisioning (*e.g.* [39]) and peer-assisted delivery architectures (*e.g.* [12,42,44,55]). However, the goals of such systems are to effectively utilize bandwidth for video streaming applications so that service providers incur a lower bandwidth cost. The aforementioned mechanisms ensure that video is effectively distributed; however, these mechanisms may not be applicable for the gaming environment as games are far more latency-sensitive, and on-demand gaming cannot benefit from large video playback buffers to improve user experience. A summary of these works is provided in Table 2.

Similarly, video conferencing or collaboration systems also stream video to clients; however, these systems can tolerate up to 200 ms [45] of latency, which is significantly more than on-demand gaming's latency tolerance. Instead, works such as [20,27,51] focus on addressing the limited available bandwidth for video conferencing and video streams. We summarize their contributions in Table 3.

Since existing technologies are not suitable to satisfy the requirements of on-demand gaming, we proposed in [19] to use edge servers, particularly CDN servers, to serve on-demand gaming end-users in order to reduce network latency. CDN providers such as Akamai [40] also use edge servers in order to provide their customers with greater performance and reliability for their Internet applications. However, the demands for content distribution are different from the demands for on-demand gaming. Firstly, Akamai's servers focus on delivering static web content and streaming media. Secondly, serving web

Work	**Approach**
Niu et al. [39]	Predicts demand and reserves minimum bandwidth resources from multiple datacenters for efficient video streaming.
Park and Pai [42]	Presents a large file transfer service which runs on top of an HTTP content distribution network.
Peterson et al. [44]	Presents a metric that allows a content distribution system to efficiently manage resources such as bandwidth. This metric allows hosts to quickly determine the allocation of resources that maximizes the peers' utility.
Wu and Lui [55]	Presents algorithms to determine the optimal replication ratio for movies in order to reduce a servers' workload

Table 2 Summary of Works for Video-on-Demand and Peer-Assisted Content Propagation

Work	**Approach**
Smilkov et al. [51]	Detect link bottlenecks using passive monitoring and modify streaming routes. Peers who have a significant amount of free bandwidth can act as relays.
Chu et al. [20]	This work evaluates the effectiveness of implementing multicast functinality at the edge. Participants organize themselves in an overlay network and adapt themselves to their application's requirements and network environment.
Gong et al. [27]	This work proposes a link-based congestion management algorithm that controls the bandwidth of each video stream while taking into account the stream's priority.

Table 3 Summary of Works in Video Collaboration and Video Conferencing

content is not as computationally intensive as rendering graphics and game data; thus, a single CDN edge server can concurrently serve many end-users whereas an on-demand gaming server might not be able to serve more than one end-user. Lastly, on-demand gaming is highly interactive; as a result, latency becomes a far more significant issue.

Nonetheless, works such as [23, 36] confirm that edge servers are not only used for serving static content. As studied in [43], current CDN infrastructure has the ability to serve millions of end-users and is well-positioned to deliver game content and software [11]. However, CDN edge servers are generally built from commodity hardware that have relative weak computational capabilities and often lack GPUs. Moreover, although these servers are designed to serve thousands of simultaneous end-users, current virtualization technologies do not enable many instances of a game engine to run concurrently on the same physical machine [13]. Additionally, GPU requirements for games are sufficiently high such that running more than one game is currently not feasible. Therefore, the transition to on-demand gaming requires CDN providers to modify their existing infrastructure, such as increasing CPU capacity and adding GPUs to CDN servers, to support the requirements of on-demand gaming.

3 Background

In order to understand the latency experienced by the end-user, it is necessary to determine the response time of on-demand gaming. The interactive *response time* is defined as the elapsed time between when an action of the user is captured by the system and when the result of this trigger can be perceived by the user. Past studies have found on-demand gaming to be highly demanding with respect to response time [21, 22, 30]. The work in [33] demonstrates that a latency around 100 ms is highly recommended for dynamic, action games while response time of 150 ms is required for slower-paced games.

The overall interactive response time T of an application includes several types of delays, are defined as follows:

$$T = t_{client} + \overbrace{t_{access} + t_{isp} + t_{transit} + t_{datacenter}}^{t_{network}} + t_{server}$$

We define t_{client} as the *playout delay*, which is the time spent by the client to (*i*) send action information (*e.g.* initiating character movement in a game) and (*ii*) receive and play the video. Only the client's hardware is responsible for t_{client}.

Additionally, we define t_{server} as the *processing delay*, which refers to the time spent by the server to process the incoming information from the client, to generate the corresponding video information, and to transmit the information back to the client. For the purpose of gaming, the work in [13] shows that *processing delay* is affected by the amount of computational resources provisioned by the provider (which could be a cloud or CDN provider), and this delay ranges from 10 ms to more than 30 ms. The provider is responsible for the *processing delay*.

Both playout and processing delays can only be reduced with hardware changes. For our purposes, we optimistically estimate that playout and processing amount to 20 ms of delay; however, we recognize that this value can vary. By subtracting the 20 ms playout and processing delay from the target 100 ms latency, it confirms that 80 ms is the threshold network latency for on-demand gaming. The remaining contribution of total latency comes from the network. We further divide the network latency into four components: t_{access}, t_{isp}, $t_{transit}$, and $t_{datacenter}$.

Firstly, t_{access} is the data transmission time between the client's device and the first Internet-connected router. Three quarters of end-users who are equipped with a DSL connection experience a t_{access} greater than 10 ms when the network is idle [25], and the average access delay exceeds 40 ms on a loaded link [54]. The behaviour of different network access technologies can greatly vary, as the latency of the access network can differ by a factor of three between different Internet Service Providers (ISP) [54]. Additionally, the home network configuration and the number of concurrent active computers per network access can double access link latency [24].

The second component of network delay is t_{isp}, which corresponds to the transmission time between the access router and the peering point connecting the ISP network to the next hop transit network. During this phase, data travels exclusively within the ISP network. Although ISP networks are generally fast and reliable, major ISPs have reported congestion due to the traffic generated by new multimedia services [29].

The third component is $t_{transit}$, which is defined as the delay from the first peering point to the front-end server of the datacenter. The ISP and provider are responsible for $t_{transit}$; however, the networks along the path are often owned by third-party network providers. Nonetheless, the ISP and the cloud provider is responsible for to ensuring good network connectivity for their clients.

Lastly, $t_{datacenter}$ is defined as the transmission delay between the front-end server of the datacenter and the hosting server for the client. The provider is responsible for $t_{datacenter}$. Network latencies between two servers in modern datacenters are typically below 1 ms [46].

4 A Reality Check on Cloud Infrastructure for On-Demand Gaming

In the following sections, we study the effectiveness of various infrastructures to offer on-demand gaming services. We focus on the network latency since the other latencies, especially the generation of game videos, have been studied in previous work [18, 32]. We evaluate an EC2-only deployment, an edge-server only deployment, and a hybrid design that combines edge servers (e.g. CDN nodes) and EC2.

4.1 Measurement Settings

To determine the ability of today's cloud to provide the on-demand gaming service, we conduct two measurement experiments to evaluate the performance and latency of cloud gaming services on existing cloud infrastructures in the US. Firstly, we perform a measurement campaign on the Amazon EC2 infrastructure during May 2012. Although EC2 is one of today's largest commercial clouds, our measurements show that it has some performance limitations. Secondly, we use PlanetLab [14] nodes to serve as additional datacenters in order to estimate the behaviour of a larger, more geographically diverse cloud infrastructure.

In our model, a datacenter (either Amazon EC2 or PlanetLab) is able to host all games and to serve all end-users that are within its latency range as it has a significant amount of storage and computational resources. This model is based on public information available regarding the peak number of concurrent end-users using on-demand gaming today (less than 1800 [38]) and the size of modern cloud datacenters (hundreds of thousands of servers [28]). Although on-demand gaming may have significant promise, it is difficult to estimate the potential peak number of concurrent users with existing industry knowledge. We believe that 1800 concurrent clients is a good starting point since it is the peak number of concurrent users on OnLive, which has been one of the largest on-demand gaming providers to date. Moreover, the use of 1800 clients will also allow us to understand OnLive's performance behaviour.

Therefore, this suggests that a datacenter is sufficiently provisioned to handle all concurrent on-demand gaming end-users. In contrast, an edge server has limited capacity; thus, each smart edge can serve far fewer end-users.

As emphasized in previous network measurement papers [25,54], it is challenging to determine a representative population of real clients in large scale measurement experiments. For our measurements, we utilize a set of 2,504 IP addresses, which were collected from twelve different BitTorrent [2] swarms. These BitTorrent clients were participating in popular movie downloads. Although 2,504 IP addresses represent a fraction of the total population in the United States, these IP addresses likely represent home users who are using their machines for entertainment purposes. Therefore, we believe that these users are representative of those who use their computers for entertainment purposes, which includes gaming.

We choose BitTorrent as the platform for our measurement experiments since we believe that BitTorrent provides a realistic representation of end-users and their geographic distribution. Using the *GeoIP* [6] service, we locate our collected peers, and we filter out end-users that are located outside of the US. We use the *GeoIP* service to restrict our clients to the United States, which is the focus of this measurement study. Moreover, the *GeoIP* service allows us to determine the approximate geographical locations of our end-users, which are used as a parameter for many of our measurement experiments. We note that we do not require fine-grained accuracy (e.g. street address) for the locations of our end-users, and that current GeoIP services provide sufficient accuracy for our purposes [16,47]. The geographic distribution of the end-users, whom we refer to as *clients*, is illustrated in Figure 1. We believe this user distribution to be similar to the user distribution of on-demand gaming since the collected clients are also using their machines for recreational purposes. After determining the clients, we use TCP measurement probe messages to determine latency between servers and clients. Note that we do not download content from BitTorrent participants as we only require the latency rather than the bandwidth.

One main advantage of retrieving IP addresses from a BitTorrent system is that BitTorrent tracker provides both an IP address and an open TCP port. By connecting to an open TCP port, we can measure the round-trip-time from the initial TCP handshake, which is more reliable than a traditional ICMP *ping* message, since the *pings* are frequently filtered by network operators.

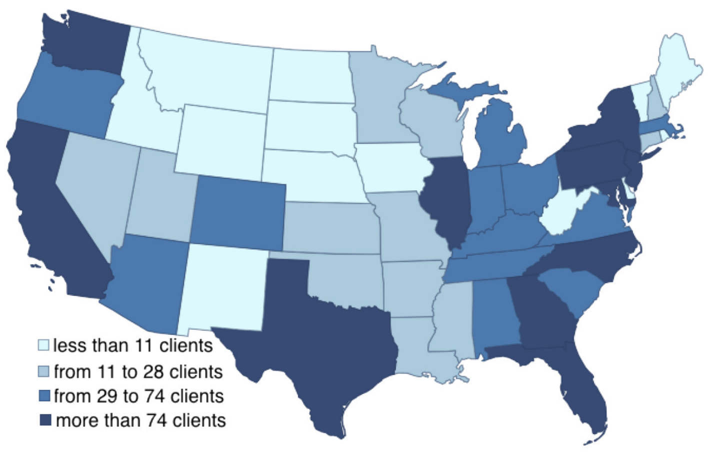

Fig. 1 Geographic distribution of clients.

4.2 Case study: Amazon EC2

The Amazon EC2 cloud offers three datacenters in the US to its customers. We obtain a virtual machine instance in each of the three datacenters. Every 30 minutes, over a single day, we measure the latency between each datacenter to all of the 2,504 clients. We use the median value from ten measurements to represent the latency between an end-host to a PlanetLab node or EC2. Figure 2 depicts the ratio of covered end-users that have at least one network connection to one of the three datacenters for a given latency target. Two observations can be made from this graph:

– *More than one quarter of the population cannot play games from an EC2-powered cloud gaming platform.* The thin, vertical gray line in Figure 2 represents the 80 ms threshold network latency yielding a 70% coverage.
– *Almost 10% of the potential clients are essentially unreachable.* In our study, unreachable clients are clients that have a network latency over 160 ms, which renders them incapable of using an on-demand gaming service. Although we filter out the IP addresses that experienced highly variable latency results, we still observe that a significant proportion of the clients have a network latency over 160 ms. This result confirms the measurements made by previous work, which identified that home gateways can introduce a significant delay on data transmission [54].

The introduction of additional datacenters in the Southern United States may benefit users in that area. However, we show in Section 4.3 that a larger cloud infrastructure, which is geographically distributed across various regions including the Southern United States, remains unable to achieve the 80 ms for a significant fraction of the end-user population. Therefore, a larger deployment may not be sufficient to provide all end-users with acceptable latency. As

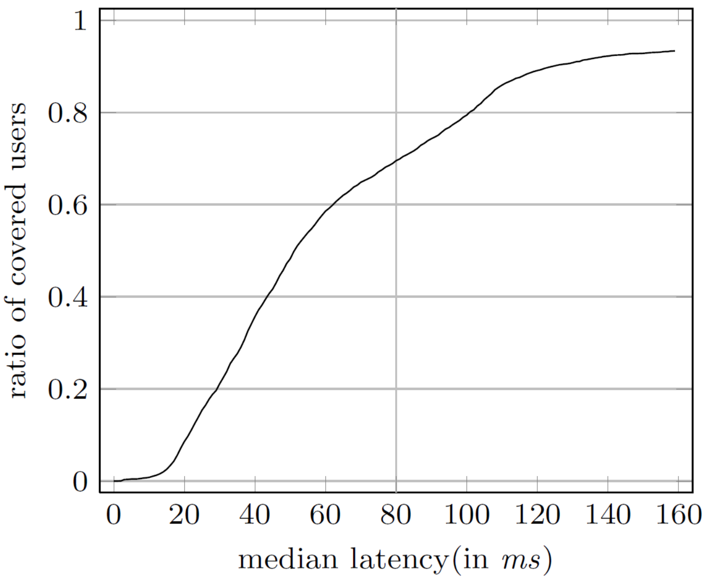

Fig. 2 Population covered by EC2 cloud infrastructure as a function of the median latency

a result, we find that the current cloud datacenter placement offers latency results that are close to optimal.

4.3 Effects of a larger cloud infrastructure

Although using existing cloud datacenters provides unacceptable end-user coverage, the gaming industry has the potential to bring new cloud operators into the cloud computing market. An alternative to deploying a small number of large datacenters is to instead use a large number of smaller datacenters. Providers such as Gaikai [3] or OnLive [7] claim to possess up to a dozen datacenters within the US [4, 5] in order to improve their population coverage. A large datacenter is generally more cost-efficient than a small datacenter; therefore, cloud providers should carefully determine if it is economically beneficial to build a new datacenter. In this section, we investigate the gain in population coverage when new datacenters are added into the existing EC2 infrastructure.

We create a simulator which uses our collected BitTorrent latencies in order to determine how many users are able to meet the latency-requirement for gaming. We utilize 44 geographically diverse PlanetLab [14] nodes in the United States as possible locations for installing datacenters. We consider a cloud provider that can choose from the 44 locations to deploy a k-datacenter cloud infrastructure. We determine latencies between clients and PlanetLab nodes using the result of our measurement campaign. Afterwards, we determine the end-user coverage when using PlanetLab nodes as additional datacenters.

We design two strategies for deciding the location of datacenters:

- **Latency-based strategy**: the cloud provider wants to build a dedicated cloud infrastructure for interactive multimedia services. The network latency is the *only* driving criteria for the choice of the datacenter locations. For a given number k, the cloud provider places k datacenters such that the number of covered end-users is maximal.[1]
- **Region-based strategy**: the cloud provider tries to distribute datacenters over an area. We divide the US into four regions as set forth by the US Census Bureau: Northeast, Midwest, South, and West. Every datacenter is associated with its region. In every region, the cloud provider chooses *random* datacenter locations. For a given total number of datacenters k, either $\lfloor \frac{k}{4} \rfloor$ or $\lceil \frac{k}{4} \rceil$ datacenters are randomly picked in every region.

For cloud providers, the main concern is determining the minimum number of datacenters required to cover a significant portion of the target population. Figure 3 depicts the ratio of covered users as a function of the response time target for two targets network latencies: 80 ms, which enable good response times for action games, and 40 ms for even more demanding games. Although different games have different latency requirements [21, 35], it is in the best interest of the on-demand gaming provider to be able to host a variety of games, including games that have stricter latency requirements. By doing so, on-demand gaming can become more attractive to potential end-users. Consequently, we select 80 ms as a network latency target since previous works [21, 33, 41] indicate that 100 ms is the latency threshold that is required for realism and acceptable gameplay for action games. Because at least 20 ms can be attributed to playout and processing delay [13], network latency can account for up to 80 ms of the total latency. We select 40 ms as a stricter requirement for games that require a significant amount of processing or multiplayer coordination.

We observe that a large number of datacenters is required if one wants to cover a significant proportion of the population. Typically, a cloud provider, which gives priority to latency, reaches a disappointing coverage ratio of 0.85 with ten datacenters for a target latency of 80 ms. Using the region-based strategy requires nine datacenters to reach a (poor) 0.8 ratio. In all cases, a 0.9 coverage ratio with a 80 ms response time is not achievable without a significant increase in the number of datacenters (around 20 datacenters). For more demanding games that have a lower latency requirement (e.g. 40 ms), we find that cloud provides exceedingly poor coverage. Even if 20 datacenters are deployed, less than half of the population would have a response time of 40 ms.

We also emphasize that EC2 is a reasonable 3-datacenter deployment. Particularly, it performs as well as a latency-based 3-datacenter deployment for the 40 ms target response time. The performance of EC2 is also comparable to a region-based 3-datacenter deployment that targets 80 ms response time.

[1] When k is greater than four, we approximate the optimal results by taking the best k-subset out of five thousand randomly generated subsets.

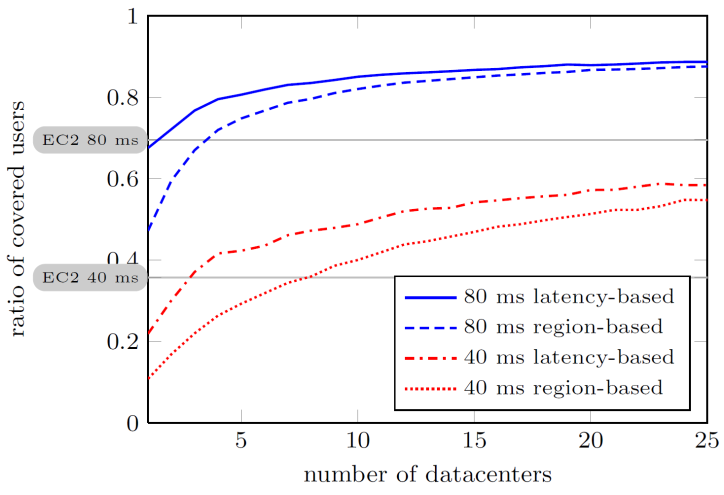

Fig. 3 Coverage vs. the number of deployed datacenters

The similarity between our measurements from PlanetLab and EC2 suggests that PlanetLab nodes can simulate datacenter sites.

We then focus on the performance of two typical cloud infrastructures: a 5 and 20-datacenter infrastructure. We assume a region-based location strategy since it is a realistic trade-off between cost and performance. We present the ratio of covered populations for both infrastructures in Figure 4.

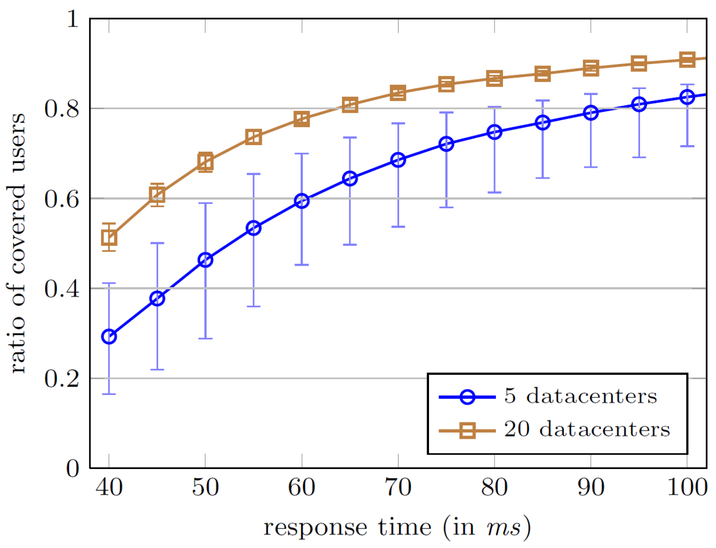

Fig. 4 User coverage for a region-based datacenter location strategy (average with min and max from every possible set of locations)

We observe that there can be significant performance gaps between a 5 and 20-datacenter deployment. Moreover, five datacenters do not guarantee good performance, despite the expectation that a region-based location strategy provides good coverage. Typically, a well-chosen 5-datacenter deployment can achieve 80% coverage for 80 ms. However, a poorly chosen 5-datacenter deployment can result in a disastrous 0.6 coverage ratio. In contrast, a 20-datacenter deployment exhibits insignificant variances in the coverage ratio.

Studies such as [17, 22, 32] show that short latency times are required in order to maintain an enjoyable user-experience, and the use of on-demand gaming suffers from additional latency due to the network. If end-users experience poor game-play while using the on-demand gaming service, they are likely to seek alternative gaming platforms. Although, current on-demand gaming providers such as OnLive [7] and Gaikai [3] utilize a relatively large datacenter deployment, on the order of 10 to 20 datacenters, our simulations show that even with a large deployment, many end-users may still experience unacceptable latency. Consequently, an alternative distribution infrastructure that improves client coverage by offering lower latency would allow on-demand gaming providers such as OnLive to improve their end-users' experience. Therefore, we explore alternative infrastructures for serving on-demand gaming end-users.

5 An Edge-Only Deployment

In the previous section, we show that a cloud-only deployment for on-demand gaming is unable to serve a significant fraction of the target population. In addition, we show that a significantly larger cloud infrastructure, consisting of 20 datacenters, can achieve at most 90% end-user coverage. An alternative to using datacenters is to use a pure edge-server deployment that employs CDN servers, which are co-located at ISPs, to serve on-demand gaming end-users. Although edge servers are in close proximity to end-users, they can only serve a limited number of end-users and host a smaller set of games due to physical limitations. In the following sections, we explore the effectiveness of various edge-server deployments.

5.1 Experimental Settings

We evaluate the effectiveness of a deployment or configuration by the number of end-users that it is able to serve. In all of our simulations, we only model active users, and they are statically matched to either a datacenter or an available edge server that has the requested game and meets the user's latency requirement. For our experiments, an edge server can only serve one end-user at a time. An end-user is served (or satisfied) if one of the following conditions are true:

– Its latency to a datacenter is less than its required latency.

– It is matched to an edge server that is within its latency requirement and hosts its requested game.

An end-user may be unmatched if a datacenter cannot meet its latency requirement and all suitable edge servers are matched to other end-users.

In simulating an edge-only deployment, we select 1,500 clients out of 2,504 of our BitTorrent clients at random to be on-demand gaming end-users. Of the remaining BitTorrent clients that have not been selected to be end-users, we select a subset of them at random to represent edge sites. An edge site represents a location of an edge server, and it may host 1, 2, or 5 on-demand gaming edge servers, where each edge server serves a single end-user. We explore these different configurations to determine their effect on the percentage of end-users served.

5.1.1 Determining Edge Server to End-User Latency

An edge server's ability to satisfy an end-user depends on its ability to meet the latency requirement of the end-user. Therefore, our simulator requires a latency matrix between all of our collected BitTorrent clients. A BitTorrent client may be used to represent an edge server or an end user. Since we do not have control of our collected BitTorrent clients, we estimate client-to-client latency by mapping a Client C_1 to its closest PlanetLab node, P. Suppose we wish to determine the latency between Client C_1 and C_2. This latency is the sum of P's latency to C_2 and a fuzzing factor that is between 0 ms to 15 ms. We assume that client C_1 is located relatively near its closest PlanetLab node P; thus, the additional 0 ms to 15 ms accounts for the latency between C_1 and P. Furthermore, an edge server may only serve an end-user if it hosts the end-user's demanded game.

5.1.2 Random Game Placement

In addition to latency, an edge server's ability to serve an on-demand gaming end-user depends on the games it hosts. For many of our simulations, each edge server hosts 5 games out of a total of 100 games, and game popularity is based on popularity distribution provided by Steam [8]. For this simulation, games are randomly placed on edge servers according to this distribution; thus, if a game is more popular according to Steam, it is highly likely that there are more copies of it in the system. The information found on Steam and Game Stats [9] indicates a peak number of players for the top 100 games. Intuitively, we perform a variation of random-weighted sampling to determine the games that are placed on each edge server and the games that the end-users demand. For example, if the statistics on Steam indicate that half of the population plays a game g, then the probability that g is placed on an edge server is 0.5. To do this, we construct a multiset S, which may contain multiple copies of each game. The number of copies of a game g in S is proportional to g's popularity. For example, if a game has X number of peak players, then $\alpha \cdot X$

copies of the game are placed in our set S. We note that α may be any positive constant that the user can vary. A larger α results in a larger pool of games (S) that the user can choose from. To assign games to end-users, we randomly select a game in S. To assign games to an edge server, we randomly select a game (call it g) in S, remove all copies of g from S, and continue to select and remove games until an edge server has reached its full capacity for games. Additionally, in this experiment, all end-users have a latency requirement of 80 ms.

5.2 The Effectiveness of a Edge-Only Deployment

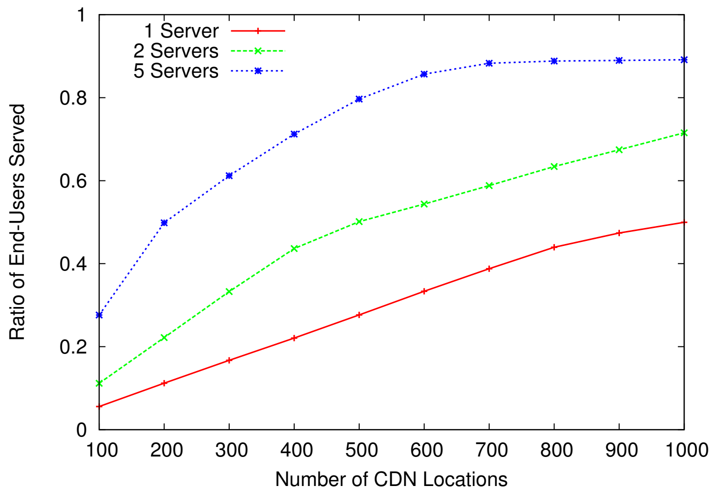

Fig. 5 Ratio of end-users served versus the number of CDN locations for the 80 ms latency target. The number of servers at each CDN locations varies, and each server hosts 5 out of 100 games. There are 1,500 clients.

From our simulations, we find that we need a significant number of edge sites in order to serve 50% of end-users using an edge-only deployment. As depicted in Figure 5, when selecting edge sites and placing games randomly, we find that at least 1000 edge servers are required (200 edge sites with 5 servers each, or 500 edge sites with 2 servers each) to serve half of the end-users. For a deployment which consists of 400 edge sites, each with 5 servers, we find that we can only serve 70% of end-users, which is comparable to a cloud-only deployment. Despite allocating 2000 servers for 1500 end-users, a significant fraction of the target population remains unable to use on-demand gaming. Therefore, we explore a hybrid infrastructure that consists of using edge servers in addition to existing cloud infrastructure.

6 Towards a Hybrid Infrastructure

As discussed in Sections 4.2 and 5, the current cloud is not well suited to meet the latency requirements of on-demand gaming end-users, and an edge-only deployment requires a significant number of edge sites and edge servers. Therefore, in this section, we explore the effectiveness of a hybrid infrastructure. We call an edge server participating in our hybrid infrastructure a *smart edge*.

6.1 Experimental Settings

Similar to our experiment in Section 5, out of the 2,504 IP addresses collected in our measurement study, unless otherwise specified, we select 1,500 of these IP addresses to serve as on-demand gaming end-users. Of the remaining IP addresses, we select 300 of them to represent smart edges. Client-to-client latency is determined using the method described in Section 5.1.1. A served end-user is as defined in Section 5.1.

6.2 Determining the Size of the Augmented Infrastructure

We now focus on the 80 ms target response time (for reasons that are described in Section 4.3), and we consider the factors that affect the performance when additional servers are added to the existing cloud infrastructure. Upon closer inspection, we are able determine whether clients are covered/served by EC2 or not. The *EC2-uncovered clients* can then also be differentiated between those who may be covered by a smart edge and those who are unreachable for a given response time.

In this experiment, each smart edge hosts one game, and there is only one game in the system. Furthermore, we restrict smart edges to serve a single user as opposed to many users. Figure 6 shows that approximately 10% of end-users are unable to meet the 80 ms latency target using EC2 or be served by smart edges. These end-users exhibit excessive delay to all smart edges and datacenters, which is likely due to non-network delays that are outside of our system's control. Therefore, the system's performance with respect to the ratio of covered end-users is limited by this ceiling.

6.3 Challenges in Using CDNs for On-demand Gaming

The results of our measurement study point to a new infrastructure that combines existing cloud datacenters with CDN servers. Because CDN servers are in closer proximity to end-users, they are able to provide lower latency for end-users than distantly located cloud datacenters. Although existing cloud infrastructure can greatly benefit from the addition of CDN servers, there are still many challenges that need to be addressed. One challenge is determining the selection of smart edges that maximizes user-coverage, which is depicted

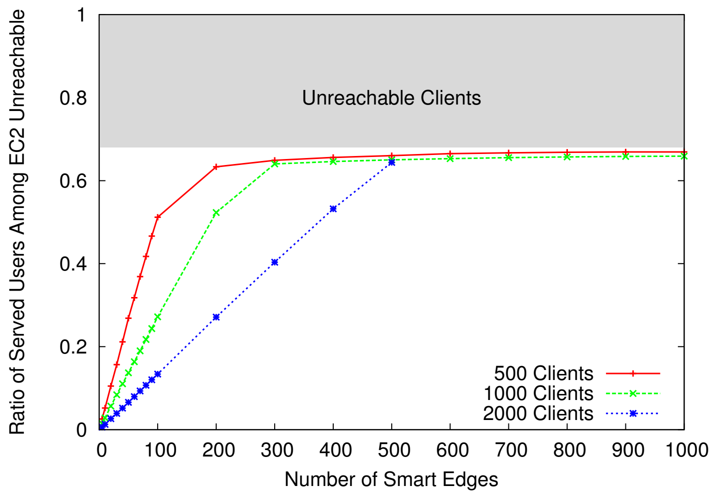

Fig. 6 Ratio of served end-users among the EC2-uncovered end-users. Each smart edge can host one game, and there is only one game in the system. One smart edge can serve up to one on-demand gaming end-user. The grey area indicates the percentage of end-users that cannot be served by both smart edges and EC2 datacenters.

in Figure 7. Unfortunately, this is an instance of the facility location problem which is NP-hard. Thus, we explore various heuristics which allow us to determine a solution. Given this selection of edge servers, we determine which games to place on each edge server. Furthermore, since smart edges cannot host an infinite number of games, due to physical limitations and cost considerations, we must strategically place games on smart edges such that we can achieve a maximal matching between end-users and smart edges.

We investigate various heuristics for smart edge selection and game placement since these two aspects affect the percentage of end-users served. Smart edge selection affects the number of potential end-users our deployment can serve, and game placement determines the ability of a smart edge to serve end-users. We further investigate the effectiveness of our heuristics in various on-demand gaming environments, such as varying the number of games each smart edge can host, the total number of games available, and different latency requirements for end-users.

6.4 Simulation Settings

In order to determine the effectiveness of our proposed heuristics, which we describe in Sections 6.5 and 6.6, we implement a simulator to determine the percentage of end-users served. Unless otherwise indicated, each smart edge hosts one on-demand gaming server and therefore has the capacity to serve one end-user. Using our collected BitTorrent clients, we assign them to represent either end-users or smart edges. For our simulations, we repeat each experi-

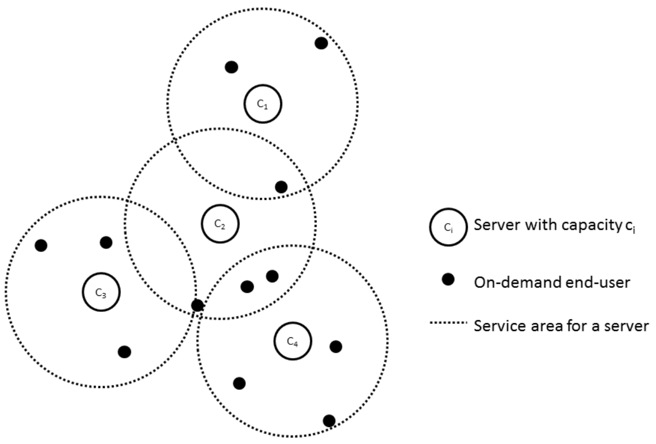

Fig. 7 Our model consists of servers (facilities) which are used to serve on-demand gaming end-users (clients). This figure is for illustrative purposes only as the network latency space is not a two-dimensional Euclidean space.

ment 16 times, and the presented ratio of end-users served is the average of all experiments. Again, end-user to end-user latency is determined using the method described in Section 5.1.

6.4.1 Creating End-Users and Smart Edges

In order to simulate an on-demand gaming environment, we begin by randomly selecting a subset of our BitTorrent clients to be on-demand gaming end-users. In our simulation, each on-demand gaming end-user wishes to play a single game, which we call a *demanded game*. An end-user is randomly assigned its corresponding demanded game according to the popularity of all games in our system. Unless otherwise indicated, game popularity follows the distribution provided by Steam and Game Stats [9], and games are assigned to end-users and smart edges in the method described in Section 5.1

After we have selected our end-users from our pool of BitTorrent clients, we determine our set of smart edges from the remaining clients. As smart edges are intended to be located in close proximity to end-users, we select a subset of our collected BitTorrent clients, which represent real end-users, to be smart edges. We select smart edges using various heuristics which are evaluated in Section 6.5. A smart edge's ability to serve an end-user is also affected by the games it hosts since a smart edge can only serve an end-user if it hosts the end-user's demanded game. We assign games to smart edges according to various heuristics which are described in Section 6.6.1.

6.4.2 Matching End-Users with Smart Edges

Following the selection of our smart edges and end-users, we construct a matching between end-users and smart edges. To accomplish this, we model the ability of a smart edge to serve end-users as a bipartite graph. If an end-user can be served by a datacenter, then it is removed from our constructed graph. Our set of V vertices includes end-users, which are not served by datacenters, and smart edges. Let u represent an end-user, and v represent a smart edge, where $u, v \in V$. An edge (u, v) exists if a smart edge hosts the end-user's demanded game, and the smart edge is able to meet the end-user's latency requirement, which is determined by our constructed latency matrix. After constructing the graph representing end-users, smart edges, and the ability for a smart edge to serve an on-demand gaming client, we determine a bipartite matching between end-users and smart edges. The size of our bipartite matching and the number of end-users that can be covered by datacenters represent our system's served end-users.

Our model represents end-users and smart edges as vertices on a bipartite graph. On-demand gaming is a platform that consists of end-users offloading computation to smart edges. In the single-player scenario, end-users do not interact with each other; thus, a greater number of servers will result in greater end-user capacity. As a result, smart edges do not communicate with each other once they are matched with an end-user. As end-users arrive, they may be matched up to a smart edge using an online bipartite matching algorithm [26]. Therefore, in our model, introducing additional smart edges does not introduce any synchronization latency. However, if on-demand gaming is used for multiplayer games, we can imagine that there is additional coordination latency between servers. We can account for coordination latency by lowering our network latency target.

6.5 Determining the Distribution of On-Demand Gaming Servers

Using our simulator, we first investigate various strategies for determining the location of on-demand gaming servers. In our model, a server is only allowed to serve an end-user if it is able to meet the aforementioned end-user's latency requirements. Due to factors such as variation in on-demand gaming usage (as a result of population density) and proximity to a datacenter, a region may have many CDN locations participate in the augmented infrastructure or none at all.

In Section 6.3, we established that our server site selection problem is an instance of the facility location problem, which has been studied in [37, 49]. Since the facility location problem is NP-hard [49], we therefore investigate various heuristics for placing on-demand gaming servers at CDN locations.

– **Random - RND:** Given our remaining set of BitTorrent clients (e.g. after we have selected our end-users), we randomly select some to represent

content distribution network servers that are responsible for serving on-demand gaming end-users.
– **Population: fractional based - POPF:** We distribute on-demand gaming servers in accordance with population distribution. In order to accomplish this, we determine the fraction of the population a region contains, and we assign this fraction multiplied by the total number of smart edges to the region.
– **Region-based - RB:** We ensure that each region has at least x on-demand gaming servers. We define a region as a geographical area that is bounded by predetermined latitude and longitude values. We recognize that some regions may have less than x smart edges. In this case, after smart edges have been selected for all regions, we randomly select smart edges from our pool of BitTorrent clients until we have reached the number of allocated smart edges. There are four regions in our simulator.
– **Voting for Closest Latency - VCL:** Every BitTorrent client determines its closest neighbour, in terms of latency, and votes for its closest neighbour to become a smart edge. The BitTorrent clients with the greatest number of votes are selected to be smart edges.
– **Voting for All Clients - VAC:** In this smart edge selection strategy, a client C casts a weighted vote to all other BitTorrent clients that have less Xms of latency to C. The weight of the vote is $1/n$, where n is the number of reachable clients. All selected end-users vote, and we select the BitTorrent clients with the highest number of votes to be our smart edges.

When analyzing the effectiveness of smart edge selection strategies, we must recognize that the effectiveness of a smart edge selection strategy is affected by the game-placement strategy. In the following figure, we present the ratio of end-users served versus the number of smart edges. In this simulation, there are 1500 end-users, 100 available games for end-users to choose from, and each smart edge can host 5 games. All end-users have a latency requirement of 80 ms. We randomly assign 5 games to each smart edge. End-users and smart edges are assigned games in the same method as described in Section 5.1.

As depicted in Figure 8, there are some differences in the ratio of end-users served between the various smart edge-selection strategies as described previously. In particular, end-user based voting heuristics, VCL and VAC, are able to cover approximately an additional 1-2% end users when compared to a random smart edge-selection heuristic. However, this minor improvement may simply be a function of measurement noise; therefore, for a 80 ms latency requirement, a random selection algorithm is as effective as any heuristic that we have evaluated.

Figure 8 also illustrates that if we have 1,500 clients, we can achieve close to 86% of end-users served with 1000 smart edges if each smart edge can host only 5 out of the 100 available games, and if games are assigned randomly. This indicates that a significant fraction of the population of clients is within 80 ms of latency to either a datacenter of a smart edge.

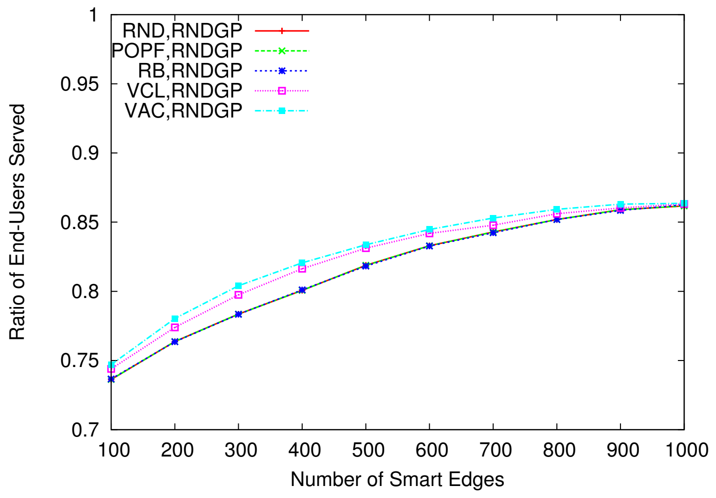

Fig. 8 Ratio of end-users served for various smart edge selection strategies. 1,500 clients, 5 games per smart edge, and 100 games in the entire system. We find that the voting all clients (VAC) smart edge selection heuristic provides the greatest ratio of end-users served. The random, population, and region-based heuristics are able to cover fewer end-users.

6.6 The Evaluation of Various Game Placement Strategies

Although there appears to be ample storage in a single hard-disk to store many games, many titles are released each year, games are increasing in size, and many games require updates and patches that consume a significant amount of storage. Thus, it is not feasible to maintain a high number of games on each server. For example, Steam, a digital distribution platform developed by Valve Corporation, hosts over 1800 games alone. Therefore, it is neither timely nor cost-efficient to store all games on every CDN-based, on-demand gaming server; thus, game placement in CDN networks becomes important since the goal of our new infrastructure is to satisfy as many users as possible. Additionally, there is an opportunity cost of storing an unpopular game on a server. If a CDN provider charges for the amount storage used at each CDN site, it would not be beneficial for a on-demand gaming provider to store a game that is rarely played such that it does not generate revenue.

6.6.1 Game Placement Strategies

In addition to the number of games we place on each CDN server, we also need to consider which games we place on each CDN server since games have varying popularity among the end-users. Therefore, we investigate various heuristics for game placement.

– **Random - RNDGP:** We randomly assign games to smart edges based on the popularity of the game. For simulations that have a total of 100 games or less in the system, this is based on game popularity collected from Steam.

- **Top X most popular games - TXGP:** We ensure that a significant fraction of games stored on CDN servers are popular, and there are fewer provisions for less popular games. Since we expect that more popular games would be played, we wish to provide many instances of them for end-users. A top X game is a game that is the X^{th} most popular game in the overall gaming market. In our simulations, we set aside at least 10% of the slots on each smart edge to host the top 60% popular games.
- **Equal opportunity - EOGP:** We mix popular games with unpopular games. Each game is placed an equal number of times regardless of its popularity. The intuition behind this technique is to provide a wider distribution of unpopular games so that end-users requesting these games have a greater chance of being satisfied.
- **Voting based - VBGP:** In this heuristic, each user votes for a game to be hosted on a server. An end user's vote has a weight which is based on the number of servers which can serve the aforementioned end-user. The intuition behind this technique is give priority to end-users which can only be satisfied by few servers. If an end-user can be served by many servers in different CDN sites, then its vote has less weight than an end-user that can be served by few servers. This ensures that end-users, which have few server options, are served. After voting is completed, the smart edge selects games based on the number of votes each game received according to a Zipfian distribution. We can accomplish the proposed voting mechanism by taking into account the end-user's gaming preferences. Many game distribution platforms such as Steam [8] and Battle.Net [1] track users' game preferences (e.g. games played, and time spent on each game), and this information can be used to constitute a user's vote. Our voting algorithm is described in Algorithm 1. The voting-based game placement heuristic needs to be bootstrapped if an on-demand gaming system does not have any information regarding its users' preferences. To collect votes, the system can survey end-users regarding their game preferences as they enter the system. Moreover, an on-demand gaming client may track users' preferences and vote accordingly.

Algorithm 1 Voting Based Protocol

```
for all end-users do
    Select an end-user, $e$, that has not voted
    Determine the servers that can serve $e$. Call this set of servers $a$
    Calculate the end-user's fractional vote for its desired game $1/a$. Call this value $b$
    Assign the value $b$ for a particular game on each server that can serve $e$
    Allocate the games with the most votes on servers once all end-users have participated
    in voting
```

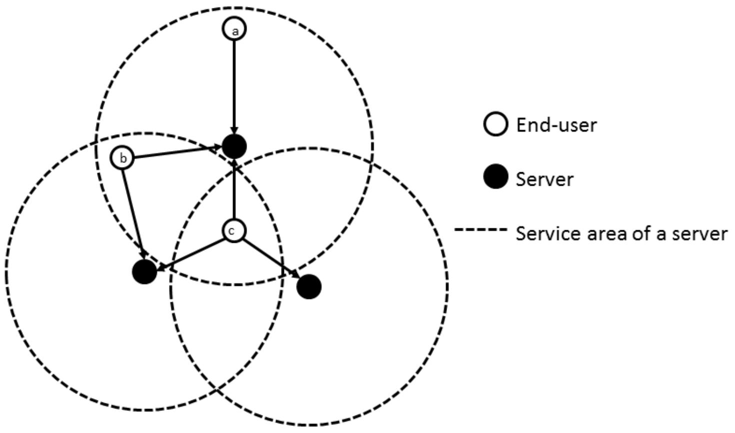

Fig. 9 End-users have the opportunity to vote for their demanded game; smart edges collect votes and host games that receive the most vote. An end-user receives one vote (valued at 1.0) and splits its vote depending on the number of servers that it can use. End-user A assigns a vote of weight 1 to the server it is able to use. End-user B assigns a vote of weight 0.5 to each of the two servers it is able to use. End-user C assigns a vote of weight 0.33 to each of the three servers it is able to use.

In our experiments, we first investigate the effects of placing 5 games on each smart edge out of a total of 100 possible games. The number of on-demand gaming end-users is fixed at 1500, and we vary the number of smart edges that are deployed. Since we have shown in Section 6.5 that differences between various smart edge-selection strategies are small, we fix our smart edge selection strategy and vary the game placement strategies.

We find that the end-user voting-based strategy and the top-X strategy provides the greatest ratio of served end-users regardless of the smart edge deployment size. Furthermore, if the voting based game placement strategy is used, we find that we require only 500 smart edges to satisfy almost all reachable end-users (recall that 10% of end-users have greater than 80 ms latency to datacenters and other BitTorrent clients). By provisioning for more popular games or taking into consideration the end-user demand for games, our architecture is able to achieve a greater number of served end-users than if games were assigned to smart edges randomly. In particular the top-X strategy provides up to an improvement of 8%, and the voting-based heuristic provides up to an improvement of 12%.

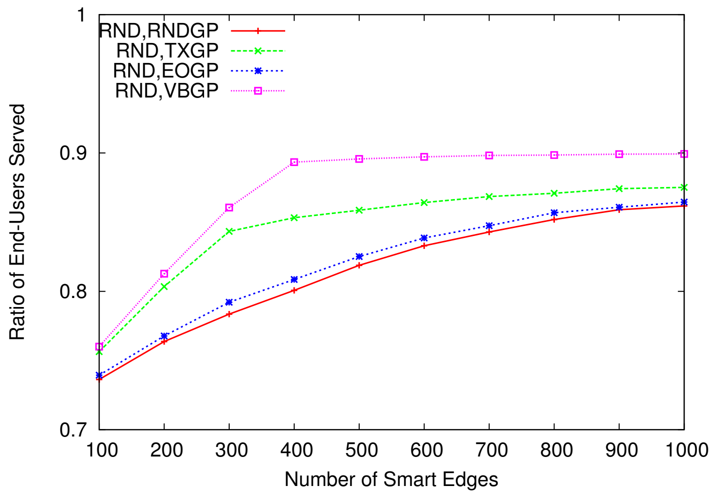

Fig. 10 Ratio of served end-users versus the number of smart edges for various game placement strategies. All simulations in this graph use the random smart edge selection heuristic. There are 1500 clients 100 total games, 5 games per smart edge, 80 ms latency requirement for all end-users. We find that the voting-based (VAC) game placement strategy provides the greatest ratio of served end-users.

Although we have demonstrated that we can achieve significant improvement in the ratio of served end-users by utilizing either the voting-based or top-X heuristic, the effectiveness of our game-placement strategies greatly depends on our on-demand gaming environment, namely the number of games that can be hosted on each smart edge and the number of games that exist in our game space. In order to understand the effects of a varying on-demand gaming environment, we investigate the effectiveness of our game-placement strategies by changing the number of games we can place on each smart edge and the total number of games in our game space. We further investigate the impact of our presented heuristics in different on-demand gaming environments.

6.7 Alternative Deployment Scenarios

In our previous sections, we consider games that have an 80 ms latency target, each server hosts five games, and there are 100 available games in our system. However, as previously described, in practice, there are many more games. Furthermore, these games may have varying latency requirements. In this section, we explore different on-demand gaming environments

6.7.1 Quantity of Games on Each Server

In order to address storage limitations, we investigate the effects of varying the number of games available on each server. Although having more games on a single server results in greater flexibility for end-user matching, which in turn leads to greater number of end-users served, there is a storage cost associated with hosting an excessive number of games on a CDN server and a maintenance cost to keep these games up to date. Additionally, there may be marginal returns for having more games on a single CDN server. Therefore, we investigate placing between 5 to 100 games on each CDN server out of a total of 1,800 available games, which is the number of games that is available on Steam. Since Steam only provides statistics for top 100 most popular games, we extend the popularity distribution to include the remaining 1,700 games.

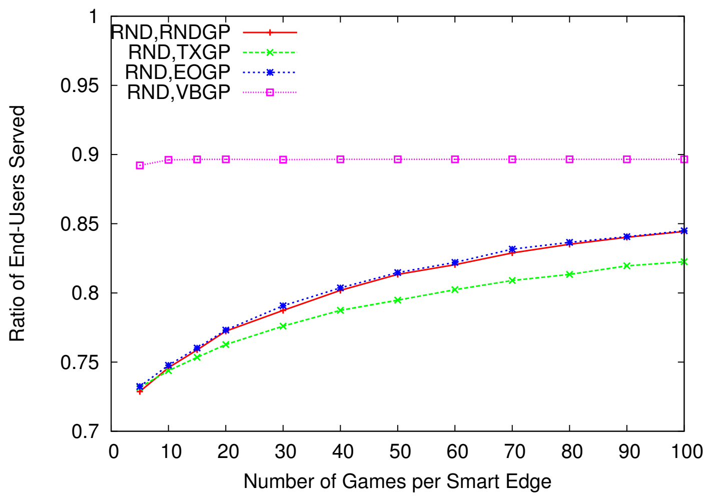

Fig. 11 Ratio of end-users served versus the number of games per smart edge. The number of games on each smart edge is varied from 5 to 100 out of a total of 1800 possible games, and here are a total of 500 smart edges and 1500 clients. We find that the voting-based game placement strategy provides the greatest ratio of end-users served, while the top-x strategy looses effectiveness as the number of games increase.

In Figure 11, we vary the number of games that we place on each smart edge. In the experiment depicted in Figure 11, there are 500 smart edges and 1,500 end-users. We compare the various game placement strategies while using the random smart edge selection strategy. We find that if there are few games hosted on a smart edge, the game placement strategy significantly impacts the number of served end-users. For example, if five games (out of 1800) are hosted on each smart edge, the voting-based game placement heuristic exhibits a 22% improvement (or an additional 16% of served end-users) over the random game placement heuristic.

However, as more games are placed on smart edges, the benefit of using a non-random game placement strategy diminishes. Since more games are hosted on smart edges, there are more opportunities to for end-users to be matched with smart edges. Therefore, there is less of a need to intelligently place games on smart edges. For example, we find that if 100 games are placed on each smart edge, then a voting-based game placement strategy provides a 6% (or an additional 5% of served end-users) increase over the random game placement strategy.

6.8 Considering Game Characteristics

In the previous section, we considered a changing environment where there are significantly many more games available to end-users, and smart edges are able to host many more games. However, in addition to game popularity, games have other characteristics that would affect the deployment of our augmented infrastructure. One game characteristic we consider is the *genre of the game*, which in turn affects the game's latency requirement. In our work, we have established that we require 80 ms for network latency; however, this latency target may be lower for faster-paced games. Therefore, we investigate the effectiveness of our smart edge selection and game placement heuristics in an environment where end-users have stricter latency requirements.

Past work [33] suggests that end-user experience is not degraded if delay is above 100 ms (80 ms of which is attributed to network delay); however, we can imagine that games in the future would be more demanding such that more time is needed for playout and processing delay, thereby reducing the amount of available time for network delay. In general, a stricter latency requirement results in fewer end-users being covered. Nonetheless, if end-users have low latency requirements, the effectiveness of our presented heuristics is more pronounced.

We find that the voting-based (VAC) smart edge selection strategy again provides the greatest ratio of end-users served. In our experiments, we determine the ratio of served end-users for the 40 ms latency target. As demonstrated in Figure 12, we find that the voting-based (VAC) strategy provides up to an additional 5 % of end-users served when compared to the random smart edge selection strategy for the 40 ms latency target. This is an improvement of 12 % which is greater than the 2 % improvement found in the 80 ms latency target.

In Figure 13, we depict the difference between the random and voting-based game placement strategies for various latency targets. Regardless of the end-user's latency target, we find that our voting-based game-placement (VBGP) heuristic provides the greatest ratio of served end-users. When compared to the random game-placement heuristic, the voting-based heuristic provides up to an additional 19% of served end-users for the 40 ms latency target.

Overall, we discover that a lower latency results in a greater difference in the number of served end-users between the voting-based (VAC) smart edge selection strategy and random smart edge selection strategy. Similarly, our experiments show that the voting-based game placement still provides the greatest ratio of served end-users. Therefore, as latency requirements become

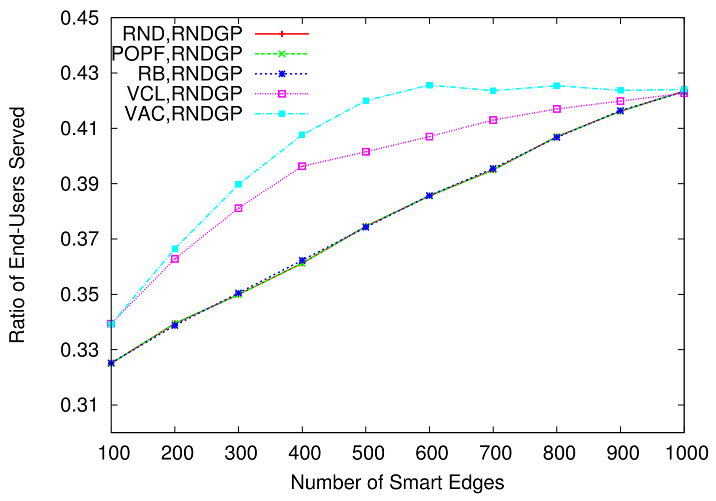

Fig. 12 Ratio of end-users served versus the number of smart edges for a 40 ms latency target. Each smart edge hosts 5 out of 100 games, and there are 1500 clients. Again, we find the voting-based (VAC) smart edge selection strategy is able to serve the most end-users, and random (RND), population (POPF), and region (RB) based strategies provide similar number of end-users served.

more strict, the advantages of using our presented heuristics becomes more pronounced.

6.9 Summary of Results

In this section, we propose a hybrid deployment, which includes using edge servers in addition to cloud resources. We propose augmenting existing cloud infrastructure with smart edges, which are located in close proximity to end-users, so that end-users may experience lower-latency. As the end-user coverage problem is hard, we present lightweight heuristics in order to improve upon the ratio of served end-users achieved by random smart edge selection and random game placement. In our evaluation, we find that we can achieve improvements in the ratio of end-users served when we employ voting-based smart edge selection strategies and game placement strategies.

Using a voting-based smart edge selection and game placement strategy, as opposed to random smart edge selection and game placement, allows an additional 10% of end-users to achieve the 80 ms network latency target. The difference between the voting-based and random strategies are even more pronounced when the network latency target is lower. For a 40 ms latency target, we find that the voting-based strategies provide up to a 50% improvement (or 19% additional served end-users).

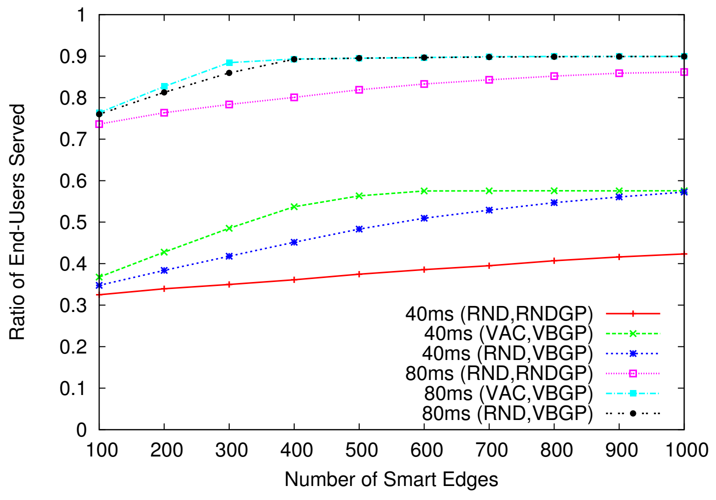

Fig. 13 Ratio of end-users served versus the number of smart edges for the 40 ms and 80 ms latency targets. Each smart edge hosts 5 out of 100 games, and there are 1,500 clients. We find that a voting-based smart edge selection strategy and game placement strategy (VAC, VBGP) provide greater number of served end-users than random strategies (RND,RNDGP) regardless of the latency target.

7 Conclusions

In this paper, we demonstrate through a large scale measurement study that existing cloud infrastructure is unable to meet the requirements of an emerging class of latency-sensitive of multimedia applications. In particular, we examine the cloud's ability to support on-demand gaming and find that only 70% of end-users are able to meet the 80 ms latency target that is required for acceptable on-demand game-play (Section 4). We then explore an edge-only deployment where all end-users are served by edge servers. We find that an excessive number of edge servers are required in order to achieve a 90% ratio of served-users (Section 5).

Therefore, to support latency-sensitive applications, we propose adopting alternative architectures that use edge servers, which are co-located at content distribution network locations that are in close proximity to the end-users (Section 6). We further investigate different approaches to deploying a hybrid infrastructure and determined their impact on the number of end-users that can be served. We find that using a voting-based heuristic for determining game placement can significantly increase the number of end-users that we can serve compared to random game placement. Overall, by using voting-based mechanisms for site selection and game placement, we can serve 90% of end-users. Therefore, by augmenting the cloud with edge servers, we can significantly improve the feasibility of on-demand gaming.

Acknowledgements This work is supported by the National Science and Engineering Research Council, Canada.

References

1. Battle.net. `http://us.battle.net/en/`. Accessed on 03/17/2014
2. Bittorrent. `http://www.bittorrent.com/`. Accessed on 17/03/2014
3. Gaikai open cloud gaming platform. `http://www.gaikai.com`. Accessed on 03/18/2014
4. Gaikai will be fee-free, utilize 300 data centers in the us. `http://www.joystiq.com/2010/03/11/gaikai-will-be-fee-free-utilize-300-data-centers-in-the-us/`. Accessed on 18/03/2014
5. Gdc09 interview: Onlive founder steve perlman wants you to be skeptical. `http://www.joystiq.com/2009/04/01/gdc09-interview-onlive-founder-steve-perlman-wants-you-to-be-sk`. Accessed on 18/03/2014
6. Maxmind - geoip python api. `https://github.com/maxmind/geoip-api-python`. Accessed on 03/18/2014
7. Play on-demand video games over the internet. `http://www.onlive.com/`. Accessed on 03/18/2014
8. Steam. `http://store.steampowered.com/`. Accessed on 03/17/2014
9. Steam and game stats. `http://store.steampowered.com/stats/`. Accessed on 05/30/2012
10. What is an mmo server? `http://www.wisegeek.com/what-is-an-mmo-server.htm`. Accessed on 03/18/2014
11. Alexander, K.: Fat client game streaming or cloud gaming. `https://blogs.akamai.com/2012/08/part-2-fat-client-game-streaming-or-cloud-gaming.html` (2012). Accessed on 18/03/2014
12. Androutsellis-Theotokis, S., Spinellis, D.: A survey of peer-to-peer content distribution technologies. ACM Comput. Surv. **36**(4), 335–371 (2004)
13. Barker, S.K., Shenoy, P.: Empirical evaluation of latency-sensitive application performance in the cloud. In: Proceedings of the first annual ACM SIGMM conference on Multimedia systems, pp. 35–46. ACM (2010)
14. Bavier, A.C., Bowman, M., Chun, B.N., Culler, D.E., Karlin, S., Muir, S., Peterson, L.L., Roscoe, T., Spalink, T., Wawrzoniak, M.: Operating system support for planetary-scale network services. In: First Symposium on Networked Systems Design and Implementation, vol. 4, pp. 19–19 (2004)
15. Beskow, P.B., Halvorsen, P., Griwodz, C.: Latency reduction in massively multiplayer online games by partial migration of game state pp. 153–163 (2007)
16. Bradley Huffaker Marina Fomenkov, K.C.: Geocompare: a comparison of public and commercial geolocation databases. `http://www.caida.org/publications/papers/2011/geocompare-tr/geocompare-tr.pdf`. Accessed on 03/18/2014
17. Chen, K., Huang, P., Wang, G., Huang, C., Lei, C.: On the Sensitivity of Online Game Playing Time to Network QoS. In: Proceedings of the 25th Conference on Computer Communications (2006). DOI 10.1109/INFOCOM.2006.286
18. Chen, K.T., Chang, Y.C., Tseng, P.H., Huang, C.Y., , Lei, C.L.: Measuring the latency of cloud gaming systems. In: Proceedings of the 19th ACM international conference on Multimedia, pp. 1269–1272. ACM (2011)
19. Choy, S., Wong, B., Simon, G., Rosenberg, C.: The brewing storm in cloud gaming: A measurement study on cloud to end-user latency. In: Proceedings of the 11th Annual Workshop on Network and Systems Support for Games, p. 2. IEEE (2012)
20. Chu, Y., Rao, S., Seshan, S., Zhang, H.: Enabling conferencing applications on the internet using an overlay muilticast architecture. In: ACM SIGCOMM Computer Communication Review, vol. 31, pp. 55–67. ACM (2001)
21. Claypool, M., Claypool, K.T.: Latency and player actions in online games. Communications of The ACM **49**, 40–45 (2006). DOI 10.1145/1167860

22. Claypool, M., Claypool, K.T.: Latency can kill: precision and deadline in online games. In: Proceedings of the first annual ACM SIGMM conference on Multimedia systems, pp. 215–222. ACM (2010)
23. Desertot, M., Escoffier, C., Donsez, D.: Towards an autonomic approach for edge computing. Concurrency and Computation: Practice and Experience **19**(14), 1901–1916 (2007)
24. DiCioccio, L., Teixeira, R., Rosenberg, C.: Impact of home networks on end-to-end performance: controlled experiments. In: Proceedings of the 2010 ACM SIGCOMM workshop on Home networks, pp. 7–12. ACM (2010)
25. Dischinger, M., Haeberlen, A., Gummadi, P.K., Saroiu, S.: Characterizing residential broadband networks. In: Internet Measurement Comference, pp. 43–56 (2007)
26. Feng, X.: Online bipartite matching: A survey and a new problem Accessed on 03/18/2014
27. Gong, Y., Wang, W., Liu, C.H.: Efficient prioritized congestion management for social network based live sharing. In: Computer Communications Workshops (INFOCOM WKSHPS), 2011 IEEE Conference on, pp. 247–252. IEEE (2011)
28. Greenberg, A., Hamilton, J., Maltz, D.A., Patel, P.: The cost of a cloud: research problems in data center networks. ACM SIGCOMM Computer Communication Review **39**(1), 68–73 (2008). DOI 10.1145/1496091.1496103. URL `http://doi.acm.org/10.1145/1496091.1496103`
29. Higginbotham, S.: Smart TVs cause a net neutrality debate in S. Korea. `http://gigaom.com/2012/02/10/smart-tvs-cause-a-net-neutrality-debate-in-s-korea/`. Accessed on 03/17/2014
30. Hoßfeld, T., Schatz, R., Varela, M., Timmerer, C.: Challenges of QoE management for cloud applications. Communications Magazine, IEEE **50**(4), 28–36 (2012)
31. Huang, C.Y., Hsu, C.H., Chang, Y.C., Chen, K.T.: GamingAnywhere: An Open Cloud Gaming System . In: Proceedings of the 4th ACM Multimedia Systems Conference, pp. 36–47. ACM (2013)
32. Jarschel, M., Schlosser, D., Scheuring, S., Hoßfeld, T.: An evaluation of qoe in cloud gaming based on subjective tests. In: 2011 Fifth International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS),, pp. 330–335. IEEE (2011). DOI 10.1109/IMIS.2011.92
33. Jarschel, M., Schlosser, D., Scheuring, S., Hoßfeld, T.: Gaming in the clouds: QoE and the users' perspective. Mathematical and Computer Modelling **57**(11), 2883–2894 (2013). DOI http://dx.doi.org/10.1016/j.mcm.2011.12.014
34. Kuchera, B.: Onlive demoed: lag, graphics are a problem. `http://arstechnica.com/gaming/2010/01/onlive-demoed-lag-graphics-are-a-problem/`. Accessed on 03/17/2014
35. Lee, Y.T., Chen, K.T., Su, H.I., Lei, C.L.: Are all games equally cloud-gaming-friendly? an electromyographic approach. In: Proceedings of the 11th Annual Workshop on Network and Systems Support for Games, p. 2. IEEE (2012)
36. Leff, A., Rayfield, J.T.: Alternative edge-server architectures for enterprise javabeans applications. In: Proceedings of the 5th ACM/IFIP/USENIX international conference on Middleware, Middleware '04, pp. 195–211. Springer-Verlag New York, Inc., New York, NY, USA (2004)
37. Liu, B.: Facility location problem. Theory and Practice of Uncertain Programming pp. 157–165 (2009)
38. de Matos, X.: Source: Onlive averaged 1800 concurrent users, ceo promised to protect patents against gaikai. `http://www.joystiq.com/2012/08/17/source-onlive-ceo-showed-no-remorse-when-announcing-layoffs/`. Accessed on 03/17/2014
39. Niu, D., Xu, H., Li, B., Zhao, S.: Quality-assured cloud bandwidth auto-scaling for video-on-demand applications. In: The 31st Annual IEEE International Conference on Computer Communications, pp. 460–468. IEEE (2012)
40. Nygren, E., Sitaraman, R.K., Sun, J.: The akamai network: a platform for high-performance internet applications. SIGOPS Opererating Systems Review **44**(3), 2–19 (2010). DOI 10.1145/1842733.1842736. URL `http://doi.acm.org/10.1145/1842733.1842736`

41. Pantel, L., Wolf, L.C.: On the impact of delay on real-time multiplayer games. In: Proceedings of the 12th international workshop on Network and operating systems support for digital audio and video, pp. 23–29. ACM (2002)
42. Park, K., Pai, V.S.: Scale and performance in the coblitz large-file distribution service. In: Proceedings of the 3rd conference on Networked Systems Design & Implementation - Volume 3, NSDI'06, pp. 3–3. USENIX Association, Berkeley, CA, USA (2006). URL `http://dl.acm.org/citation.cfm?id=1267680.1267683`
43. Passarella, A.: Review: A survey on content-centric technologies for the current internet: Cdn and p2p solutions. Comput. Commun. **35**(1), 1–32 (2012). DOI 10.1016/j.comcom.2011.10.005. URL `http://dx.doi.org/10.1016/j.comcom.2011.10.005`
44. Peterson, R.S., Wong, B., Sirer, E.G.: A content propagation metric for efficient content distribution. In: Proceedings of the ACM SIGCOMM 2011 conference, SIGCOMM '11, pp. 326–337. ACM, New York, NY, USA (2011). DOI 10.1145/2018436.2018474. URL `http://doi.acm.org/10.1145/2018436.2018474`
45. Raghavendra, R., Belding, E.M.: Characterizing high-bandwidth real-time video traffic in residential broadband networks. In: 2010 Proceedings of the 8th International Symposium on Modeling and Optimization in Mobile, Ad Hoc and Wireless Networks, pp. 597–602. IEEE (2010)
46. Rumble, S.M., Ongaro, D., Stutsman, R., Rosenblum, M., Ousterhout, J.K.: It's time for low latency. In: Proceedings of the 13th USENIX conference on Hot topics in operating systems, pp. 11–11. USENIX Association (2011)
47. Shavitt, Y., Zilberman, N.: A study of geolocation databases. arXiv preprint arXiv:1005.5674 (2010)
48. Shelley, G., Katchabaw, M.: Patterns of optimism for reducing the effects of latency in networked multiplayer games. In: In Proceedings of FuturePlay 2005, East (2005)
49. Shmoys, D.B., Tardos, E., Aardal, K.: Approximation algorithms for facility location problems (extended abstract). In: Proceedings of the twenty-ninth annual ACM symposium on Theory of computing, STOC '97, pp. 265–274. ACM, New York, NY, USA (1997). DOI 10.1145/258533.258600. URL `http://doi.acm.org/10.1145/258533.258600`
50. Shrout, R.: Onlive game service preview - is this the future of pc gaming? `http://www.pcper.com/reviews/Graphics-Cards/OnLive-Game-Service-Preview-future-PC-gaming?aid=859&type=expert&pid=1`. Accessed on 03/17/2014
51. Smilkov, D., Zhao, H., Dettori, P., Nogima, J., Schaffa, F.A., Westerink, P., Wu, C.W.: Non-intrusive adaptive multi-media routing in peer-to-peer multi-party video conferencing. In: 2010 IEEE International Symposium on Multimedia (ISM), pp. 105–112. IEEE (2010)
52. Stuart, K.: Why onlive's brave venture failed. `http://www.guardian.co.uk/technology/gamesblog/2012/aug/21/what-happened-to-onlive` (2012). Accessed on 18/03/2014
53. Sueselbeck, R., Schiele, G., Becker, C.: Peer-to-peer support for low-latency Massively Multiplayer Online Games in the cloud. In: Proceedings of the 8th Annual Workshop on Network and Systems Support for Games (2009). DOI 10.1109/NETGAMES.2009.5446229
54. Sundaresan, S., de Donato, W., Feamster, N., Teixeira, R., Crawford, S., Pescapè, A.: Broadband internet performance: a view from the gateway. In: ACM SIGCOMM Computer Communication Review, vol. 41, pp. 134–145. ACM (2011)
55. Wu, W., Lui, J.C.S.: Exploring the optimal replication strategy in P2P-VoD systems: Characterization and evaluation. In: The 30th IEEE International Conference on Computer Communications (2011). DOI 10.1109/INFCOM.2011.5934900