SMOG: A Cloud Platform for Seamless Wide Area Migration of Online Games

Virajith Jalaparti, Matthew Caesar
University of Illinois, Urbana-Champaign
{jalapar1, caesar}@illinois.edu

Seungjoon Lee, Jeffery Pang
AT&T Labs - Research
{slee,jeffpang}@research.att.com

Jacobus Van der Merwe
University of Utah
kobus@cs.utah.com

***Abstract*—Highly interactive network applications such as online games are rapidly growing in popularity but remain challenging for game providers to support, due to their inherent need for low latency. While cloud computing has proven a useful infrastructure for other applications, existing cloud computing facilities are insufficient for games, due to the unpredictability of their workload, their demands on latency and scale, and the need to support game-specific requirements (e.g., players may wish to play with certain other players already in the game). In this work, we explore whether dynamic optimization of latency and scaling of games can be achieved by supplementing cloud computing infrastructure with *seamless wide area virtual machine migration* using *network based route control*. We propose *SMOG*, a framework that dynamically migrates game servers to their optimal location, and uses orchestrated route control to optimize the network path to the server to minimize observable effects of live server migration. Through deployment of a prototype implementation on a Tier-1 ISP's backbone and a user study, we found SMOG can decrease average end-user latency by up to 60% while performing migration in a manner transparent to game players. While this paper's focus is online games, SMOG is general enough to be used for a variety of latency-sensitive interactive applications such as video conferencing and interactive video streaming.**

I. Introduction

"Cloud computing" has recently emerged with enormous success as a new paradigm for distributed computing. This computing model offers significant economic advantages of scale for purchasing, operations, and energy usage since management and machine costs can be reduced by sharing hardware across multiple applications. However, there is a particular class of applications that are more difficult to support in today's cloud environments: highly interactive network applications, such as online games.

Online gaming applications face several unique challenges. They have stringent quality of service (QoS) demands, where even small variations in network characteristics can interfere with game play as shown in several studies [16], [23], [24], [29]. Recent work has shown how applications can obtain predictable local performance in virtualized cloud environments (e.g., [15]), but the QoS of the wide area path between clients and servers remains central for a good user experience in online games. Hence, they have strong constraints on where servers can be placed. The unpredictability of network QoS coupled with the difficulty in predicting where a game will become popular make it hard to statically place these services. Even worse, unlike web users, players of certain types of games like Counter Strike: Source [1], Quake III [9], etc. often have strong affinity to particular servers (see Section II). This makes it difficult to use traditional load balancing and replication mechanisms (e.g., CDNs) to serve game clients. There have been application-specific schemes proposed for network games to deal with network problems (e.g., by partitioning state [18], [19]), but they require substantial changes in the way game developers architect their games.

These challenges are unfortunate because there are two broad classes of games that would obtain substantial benefit from shared cloud infrastructure that can meet their QoS needs: (a) games that are currently hosted by individual players, like Counter Strike: Source, and (b) the majority of games that have not yet achieved a player population large enough to justify deploying a dedicated server infrastructure around the world. Shared cloud services would reduce their operating costs and improve players' gameplay experiences. Similar benefits could be accrued by other highly interactive applications such as interactive video streaming [6].

In our work, we explore whether *dynamic migration* of cloud computing resources can simplify the problem of hosting game servers. We propose a platform for *Seamless Migration of Online Games* (SMOG), which dynamically optimizes geographic server placement while minimizing the observable effects of live server migration to players. To effectively address this problem, we take the relatively unexplored approach of *integrating cloud services with wide area network services*. Most importantly, we demonstrate the utility and practicality of using wide area migration to optimize the placement of live game servers. Our system builds upon recent proposals [32], [39] for integration of VPLS/VLANs with data center infrastructure to achieve a mechanism that seamlessly migrates gaming applications in the wide area. SMOG is practical to deploy because, unlike previous work [7], [20], [36], it requires no modifications to network components and still enables wide area migrations with sub-second down times. Moreover, SMOG requires *no application-level modifications*, is *application-agnostic* and uses the capabilities of the hypervisors in virtualized systems to achieve its goals.

In this paper, we consider the utility of SMOG for games where players may select the server that they play on for reasons other than latency. These reasons include skill-level, social, or historical preferences. This server selection model, which doesn't rely on a matchmaking entity, is used by a large number of games like Counter Strike, Quake etc. While

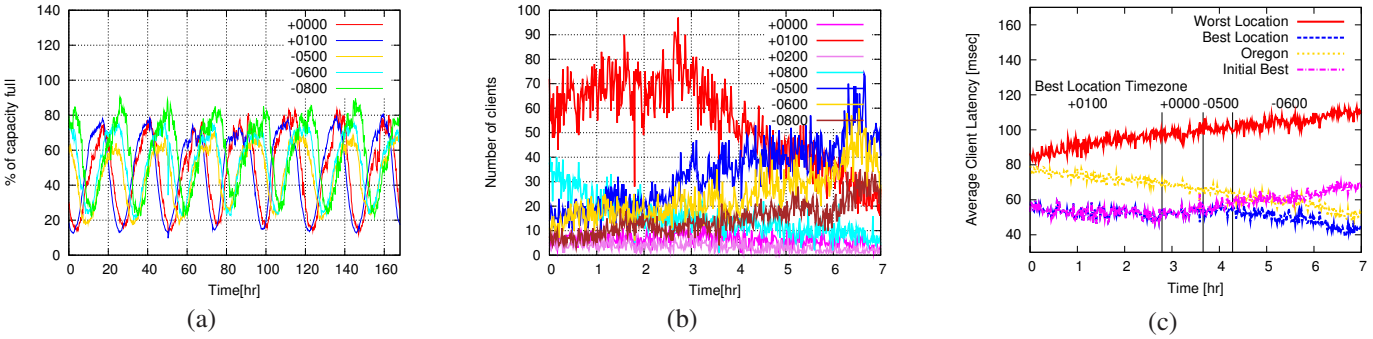

Fig. 1: Time zone level statistics (UTC=+0000): (a) Percentage of occupied server capacity in various time zones over a period of 1 week, starting from 9:30PM on June 8, 2010 EDT, (b) Number of players connected to the mshmro.com server from various time zones, (c) Variation of average player latency for different server locations.

our techniques can be used for any online game, they are particularly useful for games in which players choose the server and is complementary to other approaches to match a player to a nearby server.

This paper makes three main contributions: (1) We conduct a measurement study showing that there is substantial opportunity for both resource savings and improvement in end-user latency, if game servers could be dynamically relocated in the wide area. (2) We designed, implemented and deployed SMOG on data centers connected to a tier-1 ISP, enabling seamless game server migration in the wide area. (3) We show with a user study that the short interruption caused by SMOG's migration in the wide area is not noticeable to game players. Since previous work has shown that reducing latency improves both subjective and objective player satisfaction measures [24], [37], using SMOG's migration primitive to optimize player latency strictly improves the gameplay experience. We also show that seamless migration is feasible with SMOG in a range of realistic operating conditions.

II. Motivation

In this paper, we argue that *dynamic, seamless server migration* can significantly simplify hosting of online games in cloud environments while improving the experience of game players. This is motivated by the insights garnered from a measurement study we performed of Counter Strike Source (CSS), one of the most popular First Person Shooter (FPS) games with 1.716 billion player minutes per month [10]. (We observed similar results for Quake III [9] and Unreal Tournament [11]). We found several unique features of games:

Games need *dynamic*, not *static* allocation of resources: First, we collected a list of the 1000 most popular CSS servers from Game Tracker [4] and measured the number of players present on each of them over a period of one week, starting from 9:30PM on June 8, 2010 EDT. All servers are probed at 10-minute intervals using QStat [8]. With IP geolocation tools [5], we inferred the time zone of each server. From this study, we make two key observations. As others have previously observed [22], we found that *server loads change greatly over time*. Figure 1(a) shows the load on servers in various time zones over the 7-day period. We define load as the ratio of the total number of players in a time zone to the sum of the maximum capacities of the servers in that time zone. We observe a strong diurnal pattern with peak loads of 4-5 times the troughs. Second, we find that *peak loads shift over time according to the timezones of the servers' geographic locations.* For example, load peaks in the eastern hemisphere occur when load in the western hemisphere is lowest, and vice versa. This suggests that *game providers can benefit from dynamic hosting of resources*; game providers could *migrate* servers from lightly loaded regions into peak regions to cope with load, and to react to dynamics in network path quality.

Games need *migration*, not *replication* of servers: Next, we studied how users on a server are geographically distributed. Here, we used a 7-hour `tcpdump` trace collected on April 11, 2004 starting at 9AM PDT from mshmro.com [2], a long running popular CSS server located in Portland, Oregon, USA. Again using an IP geolocation service [5], we mapped the player IP addresses to time zones. From this, we make two key observations. First, we find that *users may express a strong affinity to specific servers*. Although it is well known that players prefer servers that have lower latency [24], our measurements show that geographic and network distance are not the sole criteria in server selection. For example, Figure 1(b) shows that players from all over the world (i.e., several time zones) choose to play on the Oregon CSS server even though there were several other servers closer to them.

Second, we find that *the optimal server position varies dynamically* and *server migration can substantially improve client performance*. Figure 1(c) shows the estimated average client latency given different server placements. For this experiment, we use clients from the mshmro.com trace as the clients connected to the game server. We assume the game server can be located at multiple physical locations corresponding to multiple data centers in a Tier-1 ISP. We estimate the latency experienced by clients using their *air-mile distance* to the server [13]. The air-mile distance is obtained by mapping the IP addresses of the clients and servers to a *(latitude,longitude)* coordinate pair using IP geolocation [5] and then finding the *great-circle distance* between the pairs.

Vertical lines in Figure 1(c) indicate a change in the time zone of the best server location. By using migration (Best Location), the average client latency to the server can always be kept close to 50ms, the highest latency that doesn't degrade FPS player performance under any type of game play [27]. In contrast, the two static server placements (Oregon and Initial Best) have up to 60% higher latencies, compared to the Best Location. The Worst Location line shows that a static placement could be suboptimal by more than 100%. These results suggest that *game providers can benefit from migration of servers*. Unlike traditional services that use replication and redirect user requests to the appropriate (e.g., closest) replica, a user's affinity to a particular online game server means that existing servers need to be moved closer to clients.

Games need *seamless*, not *paused* migration: Next, to understand how long a user is connected to a particular server, we studied the session times of players, using the above week long trace from the top 1000 servers (results omitted due to lack of space). We found that *session durations are very long and have large variance*. While the median session time was around 20 minutes, some players go on to play for several hours with the 99%ile session time being as high as 3.5hrs. This suggests that *migration must be performed while users are playing*, introducing the need for *seamless migration*. The existence of multi-hour sessions means that traditional *dryout* techniques used for non-gaming services (redirecting new users to another server and waiting for the old server to empty) may take too long to react to diurnal patterns or flash crowds in online games.

III. SMOG: Design and Implementation

To address the unique requirements for hosting online game servers in a cloud environment (Section II), we designed a platform for Seamless Migration of Online Games (SMOG). In this section, we present the architecture of SMOG and the techniques it uses to solve two key problems: how to migrate application services across the *wide area* without service disruption, and when and where to migrate them.

SMOG Architecture: The architecture of SMOG is depicted in Figure 2. The cloud service provider deploys a set of *hosting platforms*, connected to a wide area network (WAN). These hosting platforms act as facilities where game servers may be hosted. Each hosting platform consists of (i) a number of *physical servers* running virtualization software that supports VM migration, (ii) *datacenter networking* equipment that provides connectivity within the platform as well as connectivity to the WAN, and (iii) an *SMOG node controller* which is responsible for local orchestration of compute and networking resources to realize SMOG functionality. The SMOG node controllers help in VM migration, manipulate routes, redirect traffic, and observe traffic patterns and routing tables within the network in order to support migration of servers. SMOG node controllers in turn are collectively orchestrated by an *SMOG service controller* which makes migration decisions in order to realize the service objectives of the hosted application.

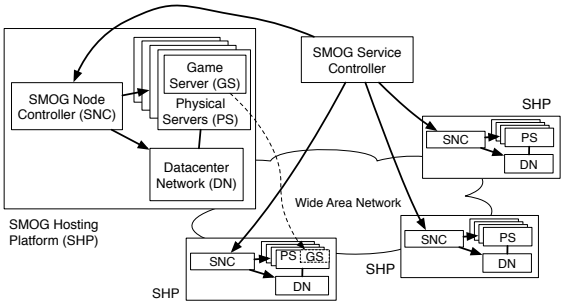

Fig. 2: SMOG provides a distributed platform, across the wide area, for hosting online games.

Wide area Migration: To adapt to changing application workloads, SMOG needs to perform server migration between different hosting platforms, i.e., across LANs, across IP subnets, and potentially across Autonomous Systems (ASes) to provide the best service to the users. To overcome the limitations of traditional virtual machine migration techniques (e.g., Xen [26]) which target migration within a single Ethernet-based IP subnet, SMOG *integrates network route control with VM migration*.

Figure 3 depicts the mechanisms which SMOG uses in migrating an application server (P) from location A to location B. Each application server is allocated a small IP subnet (e.g., a /29). Under normal operating conditions, this prefix is advertised by the datacenter router (Ra) via BGP to the provider edge router (PEa) to allow the application server to be reached from the Internet. The application server uses Ra as its default gateway in order to reach users in the Internet. SMOG utilizes a virtual private LAN service (VPLS) (Va to Vb via PEa and PEb), between physical servers Sa and Sb [38] to interconnect the LAN switches SWa and SWb at layer two and create a broadcast domain between the two sites [1]. This in turn enables the use of "normal" LAN VM migration techniques [26] to migrate the application server (P) from Sa to Sb. After migration to location B, P issues a gratuitous ARP response and will be still reachable from the Internet via the path Ra, SWa, Va (now encapsulated in a VPLS frame), Vb, SWb, Sb and P. P will still be using Ra as its default gateway and hence, packets from P will follow the same path in reverse.

Having traffic follow this "dogleg" path is of course suboptimal, and the SMOG service control corrects this by (a) changing the default gateway of P to Rb, and (b) advertising a more specific prefix (e.g., a /31) to P via the BGP session between Rb and PEb. This causes the path via PEb to be preferred over the path via PEa. Once SMOG monitoring has confirmed that traffic is no longer using the path via PEa/Ra to reach P, Ra is instructed to withdraw the /29 it is advertising, leaving the path via PEb as the only available path to reach P. Finally, Rb in turn is instructed to advertise the shorter prefix (/29) so that we are back to the original state,

[1]The VPLS endpoints Va and Vb are in fact encapsulated within the datacenter routers (Ra and Rb), but we show them separately to simplify the exposition.

except that both P and its associated prefix are now associated with location B. Note that in this mechanism there is always at least one (and for some time two) valid paths to P which prevents packet loss due to the route changes.

Cross-AS Migration: With the above techniques, as long as hosting platforms A and B are in the same AS, no inter-AS advertisements are generated during a migration. However, if the hosting points are located in different ASes, external routing updates may be generated. Inducing BGP routing advertisements could trigger flap damping or routing protocol reconvergence, which in turn could lead to large outages intolerable by online games. Further, route updates for prefixes more specific than /24 are often filtered by edge routers of ASes [21] which can prevent the updates sent by SMOG from propagating across ASes.

To address this, we propose the following mechanism: all the hosting platforms used by the SMOG framework form an SMOG-AS, an Autonomous System that can span multiple networks. The SMOG-AS peers with each AS in which a hosting platform is present. It announces the prefixes corresponding to the game servers hosted on the SMOG framework. For example, SMOG-AS may purchase transport from each AS it peers with and form an internal VPN which is used to migrate the VMs between hosting points. This mechanism ensures that the VMs remain within a single AS (the SMOG-AS), removing the need for inter-AS BGP announcements. Further, since ASes use hot-potato routing [35], the routes to the SMOG-AS are kept short.

Optimized server placement in SMOG: While the above subsection shows how SMOG can carry out migration of application servers, we still need a mechanism to decide when to move the servers. The goal is to minimize the number of players with latency greater than the satisfactory threshold. Given the capacity of each hosting point, the latencies from each client to each hosting point, and the players in each game server, this optimization problem can then be formulated as a variant of the *generalized assignment problem* [34]. Player latencies can be collected using ping or passive latency estimation tools [13], [28]. To solve this optimization problem, we can use existing algorithms that find approximate solutions efficiently [14], [34] (e.g., in tens of milliseconds on commodity hardware for instances equivalent to having hundreds of game servers and around ten hosting points). SMOG solves this optimization problem at regular time intervals to determine the optimal position of each server. If the new optimal location is better than the current location by a pre-defined threshold, SMOG initiates a migration. This threshold is configurable and can be set based on the load on the network. The experiment in Figure 1(c) shows that setting it to 10% of the current value results in just 4 migrations within a total of 7 hours. This shows that migrations are often infrequent and take place once every few hours. Thus, every time SMOG decides to migrate a VM, it has ample time for (a) migrating the entire VM (as shown in our implementation below), and (b) BGP convergence (which typically takes a few minutes [30]). In general, the threshold can be set by the system administrator considering application requirements and the current load on the data-centers.

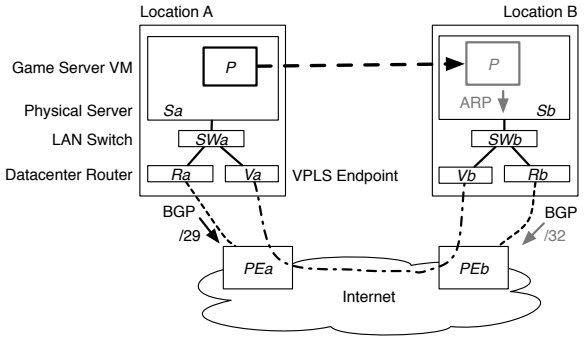

Fig. 3: SMOG Migration

#	Migration Step	FV	PV	Mod
1	Migration started at loc_2	0.239	0.096	0.146
2	VM suspended at loc_1	10.844	9.160	9.355
3	VM destroyed at loc_1	11.073	9.187	9.674
4	Received all pages at loc_2	11.079	9.173	9.424
5	Domain restored at loc_2	11.430	9.436	9.507
6	1st packet received by VM at loc_2	11.646	9.622	9.523
	Downtime = (Time of #6 - Time of #2)	0.802	0.462	0.168

TABLE I: Typical timeline (in seconds) for Xen VMs migrated from loc_1 to loc_2 for a fully-virtualized (FV) VM, paravirtualized (PV) VM, and modified Xen (Mod). Migration starts at loc_1 at $t = 0$ sec.

Implementation of SMOG: To determine the feasibility of our approach, we implemented and deployed a prototype of SMOG on ShadowNet [25], an operational trial network and compute platform deployed on carrier-grade equipment and located within a tier-1 ISP. We used DRBD [3] to ensure that the logical disk used by the VM on the physical machines are synchronized. SMOG builds on the migration capabilities of Xen [12]. In our prototype, we used the Xen 3.4.3 hypervisor for the virtual machines responsible for hosting the game servers. When using the migration capabilities of the default hypervisor, we observed that SMOG's migration interrupted the user's game experience. For example, when migrating a Quake III game server, a game client connected to it experiences a downtime of around 500msec during migration and displays *server disconnected* message, even with paravirtualization enabled. A downtime of 800msec is seen when using a fully virtualized server. To study this problem, we instrumented the Xen migration tools to micro-benchmark the sources of delays, as shown in Table I. In the timeline, we note that the VM is completely unreachable to any clients from when it is suspended at loc_1 (step 2) to when it receives the first packet at loc_2 (step 6). We noticed that a large fraction of the downtime is due to the hypervisor at loc_2 waiting for the hypervisor at loc_1 to close the connection used to transfer the VM state during migration. We modified Xen to avoid this wait condition (Mod). With this extension, we found that the downtime reduced to around 170 msec. Our user study (Section IV-A) shows that this downtime is in fact not noticeable by humans during gameplay.

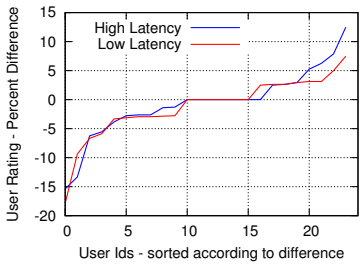

Fig. 4: Percentage difference in user ratings

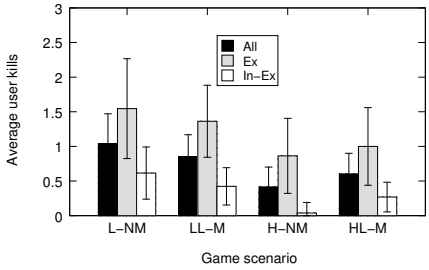

Fig. 5: Average user Kill Count

IV. Evaluation

We next study the performance of SMOG in practice. First, to understand the effect of SMOG end-user game experience, we conduct a user study (Section IV-A). Second, to understand performance issues in SMOG under varying workloads, we instrument a deployment of SMOG to directly measure the relevant performance metrics (Section IV-B).

A. User Study

SMOG aims to minimize the effects of migration on end users. To evaluate how well it achieves this goal, we conducted a user study using Quake III running on SMOG.

Scenarios presented to users: In this study, we consider four different scenarios: (a) L-NM, a low latency setting with no migration, (b) LL-M, a low latency setting with migration to another low latency server, (c) H-NM, a high latency setting with no migration and (d) HL-M, a high latency setting with migration to a low latency server. We use 50ms as the RTT in the low latency (L) scenarios, since it is known to be the limit above which user performance degrades [27]. The high latency (H) value of 200ms is the 95th percentile value of RTTs to the top 500 Quake III servers around the world [4] from a host on the UIUC campus.

Procedure: In our user study, each participant plays a trial consisting of a pair of game scenarios, either (L-NM, LL-M) or (H-NM, HL-M). To remove any bias caused by the order in which these scenarios are played, we randomize the trial order by having each player play each trial pair twice, in both possible orders (4 trial pairs and 8 games total). The users are told that they would be playing the game under network conditions that may differ across trials, but they are not given the specific details of the differences between them.

Before the experiment, each user plays a practice match for 5 minutes on the `q3dm1` map of Quake III over a LAN environment. Each game round in the experiment lasts one minute. All rounds are *death match* games in which the user plays against a computer bot on the game server. We use the `q3dm1` map since it is small enough for a controlled experiment. In every round, both the human player and the bot start from fixed *spawn points*. For scenarios with migration, the migration completes around half-way into the game. This ensures that users have (almost) equal time to play under different latencies, before and after the server's migration.

Results: Our user study consisted of a total of 24 participants and 96 trials. We use two metrics to measure SMOG's effectiveness: (a) *User Ratings*: In each round, users rate the quality of their game experience on a scale of 1-10 based on the lag experienced during the round, with 10 being no observable lag at all and 1 being completely unplayable. This provides a subjective evaluation of the user's experience. (b)*Kill Counts*: To get an objective measure of the user's performance, we measure the number of kills a user accomplishes in each of the rounds played. We divide all the users (All) into two groups: experienced (Ex) and inexperienced (In-Ex). Experienced players are those who reported that they played FPS games regularly at least once a week at some point in their life (around 50% of users in our study).

Figure 4 shows the percentage difference between the average ratings given by the users for the scenarios with and without migration for a particular latency case. We find that this difference between the ratings varies only between -15% to 10% for both high and low latency settings. The distribution of differences is centered on 0 (no difference) and has about equal weight on either side. This allows us to conclude that the downtime due to migration in SMOG cannot be perceived by users. We note that the same results hold when experienced (Ex) and inexperienced users (In-Ex) are considered separately.

Figure 5 shows the average kill count of all the users for the different scenarios. The error bars indicate the 95% confidence interval. Comparing the NM and M scenarios for a particular latency setup, we see that presence of migration does not have a significant negative affect on user performance in the game, when all the users or just the experienced users are considered. In fact, migration significantly increases the kill count for inexperienced users in the high latency scenario (H-NM vs. HL-M) showing that migration actually helps users.

B. Performance Analysis

To generalize our results, we studied the performance of SMOG under a variety of network conditions. We use *application downtime* as the main metric to evaluate SMOG's performance as it directly affects user experience in online games. In these experiments, we varied the bandwidth available for migration from 50Mbps to 1000Mbps (in steps of 50Mbps) and the RTT of the link used for migration from 25msec to 200msec (in steps of 25msec). All experiments were performed using our modified Xen hypervisor (Section III) with a 1GB paravirtualized VM. We found that SMOG works seamlessly for bandwidths more than around 125Mbps and RTTs less than 120msec. For higher RTTs, we found

that the downtime can be noticed by the users of Quake III. However, using other optimizations to Xen's migration (e.g., [38]), we can further decrease the downtime and SMOG can work seamlessly for bandwidths as low as 50Mbps and RTTs as high as 180msec.

V. Related Work

Related work is discussed throughout the paper, but we revisit two key related areas here:

Live migration of virtual machines: Virtual machine migration techniques have been proposed earlier (e.g., Xen [26]) but they assume that the physical machines are on a single LAN and cannot be directly adopted in wide area settings. Other approaches [7], [20] require modifications to the network or application components. In contrast, SMOG targets wide area settings, provides techniques to minimize outages to a few hundreds of milliseconds and requires no modifications to applications or the network components. Building on [32] and [39], SMOG supports highly-interactive applications by using VPLS/VLANs to perform seamless dynamic WAN migration and route control to reap the benefits offered by such migration. Our implementation of SMOG builds on the migration primitives provided by Xen [12] and proposes modifications to ensure seamless migration.

Dynamic resource management for games: Many solutions for resource management in games [31], [33] focus on determining *how many* servers are needed to handle player load while SMOG determines *when* and *where* to move those servers. Further, our work is complementary to other approaches [17] which aim at migrating individual objects rather than the entire game state.

VI. Conclusions

This paper presented SMOG, a cloud framework for highly-interactive applications such as online games. Unlike traditional cloud computing platforms, SMOG's infrastructure is distributed across the wide area and integrated with network support to provide enhanced service to online games. We showed that games can achieve better end user experiences using SMOG's central primitive: dynamic, seamless, wide area virtual machine migration. We realized a working prototype of SMOG on a Tier-1 ISP's backbone network, achieving only around 200msec as application downtime during migration. Our evaluation results show that SMOG's dynamic hosting is transparent to the end-users and performs well under a variety of operating conditions. While this paper focuses on highly interactive online games, SMOG can also be used for improving the performance of other latency-sensitive network applications. We believe SMOG is a first step towards the broader challenge of integrating cloud resources with the network, a requirement for realizing the full potential of cloud computing for highly-interactive applications.

References

[1] Counter-strike: Source on steam. http://store.steampowered.com/css.
[2] Counter-Strike trace. http://www.thefengs.com/wuchang/work/cstrike/.
[3] DRBD. http://www.drbd.org/.
[4] Game tracker. http://www.gametracker.com/.
[5] IpInfoDB. http://ipinfodb.com/.
[6] OnLive. http://www.onlive.com.
[7] Open Flow Demo. http://www.openflowswitch.org/wp/2008/12/video-of-mobile-vms-demo/.
[8] QStat. http://www.qstat.org/.
[9] Quake III Arena. http://www.quake3arena.com/.
[10] Steam Game Statistics. http://steampowered.com/status/game_stats.html.
[11] Unreal Tournament. http://www.unrealtournament.com/.
[12] Xen. http://www.xen.org.
[13] S. Agarwal and J. R. Lorch. Matchmaking for online games and other latency-sensitive P2P systems. In *SIGCOMM*, 2009.
[14] H. A. Alzoubi, S. Lee, M. Rabinovich, O. Spatscheck, and J. V. der Merwe. Anycast CDNS revisited. In *WWW*, 2008.
[15] H. Ballani, P. Costa, T. Karagiannis, and A. Rowstron. Towards predictable datacenter networks. In *SIGCOMM*, 2011.
[16] T. Beigbeder, R. Coughlan, C. Lusher, J. Plunkett, E. Agu, and M. Claypool. The effects of loss and latency on user performance in unreal tournament 2003. In *NetGames*, 2004.
[17] P. B. Beskow, K.-H. Vik, P. Halvorsen, and C. Griwodz. Latency reduction by dynamic core selection and partial migration of game state. In *NetGames*, 2008.
[18] A. Bharambe, J. R. Douceur, J. R. Lorch, T. Moscibroda, J. Pang, S. Seshan, and X. Zhuang. Donnybrook: Enabling large-scale, high-speed, peer-to-peer games. *SIGCOMM CCR*, 38(4), 2008.
[19] A. Bharambe, J. Pang, and S. Seshan. Colyseus: a distributed architecture for online multiplayer games. In *NSDI*, 2006.
[20] R. Bradford, E. Kotsovinos, A. Feldmann, and H. Schiöberg. Live wide-area migration of virtual machines including local persistent state. In *VEE*, 2007.
[21] M. Caesar and J. Rexford. BGP routing policies in ISP networks. In *IEEE Network Magazine*, 2005.
[22] C. Chambers, W. Feng, S. Sahu, and D. Saha. Measurement-based characterization of a collection of on-line games. In *IMC*, 2005.
[23] K. Chen, P. Huang, and C. Lei. Effect of network quality on player departure behavior in online games. *IEEE Trans. Parallel Distrib. Syst.*, 2009.
[24] K. Chen, P. Huang, G. Wang, C. Huang, and C. Lei. On the sensitivity of online game playing time to network qos. In *INFOCOM*, 2006.
[25] X. Chen, Z. Morley, M. Jacobus, and V. Merwe. Shadownet: A platform for rapid and safe network evolution. In *Usenix ATC*, 2009.
[26] C. Clark, K. Fraser, S. Hand, J. G. Hansen, E. Jul, C. Limpach, I. Pratt, and A. Warfield. Live migration of virtual machines. In *NSDI*, 2005.
[27] M. Claypool and K. Claypool. Latency and player actions in online games. *Commun. ACM*, 49(11), 2006.
[28] K. P. Gummadi, S. Saroiu, and S. D. Gribble. King: Estimating latency between arbitrary Internet end hosts. In *IMW*, 2002.
[29] T. Henderson. *The effects of relative delay in networked games*. PhD thesis, University of London, Apr. 2003.
[30] C. Labovitz, A. Ahuja, A. Bose, and F. Jahanian. Delayed internet routing convergence. In *SIGCOMM*, 2000.
[31] P. Quax, J. Dierckx, B. Cornelissen, G. Vansichem, and W. Lamotte. Dynamic server allocation in a real-life deployable communications architecture for networked games. In *NetGames*, 2008.
[32] K. K. Ramakrishnan, P. Shenoy, and J. Van der Merwe. Live data center migration across WANs: a robust cooperative context aware approach. In *INM*, 2007.
[33] A. Shaikh, S. Sahu, M. Rosu, M. Shea, and D. Saha. Implementation of a service platform for online games. In *NetGames*, 2004.
[34] D. Shmoys and E. Tardos. An approximation algorithm for the generalized assignment problem. 1993.
[35] R. Teixeira, A. Shaikh, T. Griffin, and J. Rexford. Dynamics of hot-potato routing in ip networks. In *SIGMETRICS*, 2004.
[36] V. Valancius, N. Feamster, J. Rexford, and A. Nakao. Wide-area route control for distributed services. In *USENIX ATC*, 2010.
[37] A. F. Wattimena, R. E. Kooij, J. M. van Vugt, and O. K. Ahmed. Predicting the perceived quality of a first person shooter: The Quake iv G-model. In *NetGames*, 2006.
[38] T. Wood, K. K. Ramakrishnan, P. Shenoy, and J. van der Merwe. Cloudnet: dynamic pooling of cloud resources by live wan migration of virtual machines. In *VEE*, 2011.
[39] T. Wood, P. Shenoy, A. Gerber, K. K. Ramakrishnan, and J. Van der Merwe. The case for enterprise-ready virtual private clouds. In *HotCloud*, 2009.