G-COPSS: A Content Centric Communication Infrastructure for Gaming Applications

Jiachen Chen, Mayutan Arumaithurai, Xiaoming Fu
Institute of Computer Science, University of Goettingen, Germany.
Email: {jiachen, arumaithurai, fu}@cs.uni-goettingen.de

K.K.Ramakrishnan
AT&T Labs Research, Florham Park, NJ, U.S.A.
Email: kkrama@research.att.com

***Abstract*—With users increasingly focused on an online world, an emerging challenge for the network infrastructure is the need to support Massively Multiplayer Online Role Playing Games (MMORPG). This is an application domain that is attracting more players than ever before, very often with players distributed over a metropolitan area. Currently, MMORPG are built on an IP infrastructure with the primary responsibility on servers to do the work of disseminating control messages and having to predict/retrieve objects in each player's view. Limited server resources significantly impair the user's interactive experience. Modern fast-paced action games that run on a client/server architecture limit the number of players who can interact simultaneously since the server needs to handle the frequent updates and disseminate them. Scale and timeliness are major challenges of such a server-oriented gaming architecture.**

We propose Gaming over COPSS (G-COPSS), a communication infrastructure using a Content-Oriented Pub/Sub System (COPSS) to enable efficient decentralized information dissemination in MMORPG, exploiting the network and the end-systems for player management and information dissemination. We emulate an application that is particularly emblematic of MMORPG – Counter-Strike – but one in which all the players share a hierarchical structured map. Using trace-driven simulation, we demonstrate that G-COPSS can achieve high scalability and tight timeliness requirements of MMORPG. The simulator is parameterized using the results of careful microbenchmarking of the open-source CCN implementation and of standard IP-based forwarding. Our evaluations show that G-COPSS provides considerable performance improvement in terms of aggregate network load and update latency compared to that of a traditional IP server-based infrastructure.

I. Introduction

Supporting Massively Multiplayer Online Role Playing Games (MMORPG) [1] is a significant challenge. MMORPGs have become very popular because of their attractive structuring and creative scenarios and the realization of real-world human interactions. World of Warcraft and Counter-Strike are examples of such games and are characterized by the need for high interactivity (very low network latency), since every action an individual player performs needs to be communicated to all the related players and the players need to react according to the 'current' environment and the cumulative actions of all the players up to that point. Games like Second Life involve a large number of players (possibly within a metropolitan area, although not necessarily restricted as such) and require a persistent view of the world that is usually managed by a dedicated server (e.g., one that is hosted by the game's publisher). The load on such a server for player management and communication can be significant, and is likely to be a source of substantial latency. One of the problems in designing a MMORPG is that of determining the related players for every action and disseminate these actions and the changed environment to the relevant players in a scalable manner with very low latency. As we observe in this paper, for games where the environment is divided into regions that different groups of players may have varying amounts of visibility, it is desirable to sub-divide the environment into hierarchical regions. We envisage incorporating the notion of a "multi-layer hierarchical map". While such capabilities exist in some limited form in selected games, such as Second Life (where players share a global map), they may be useful in other interactive MMORPG. A server-based infrastructure is likely to have more difficulty with providing this capability because of the communication and processing (customized for each individual player) requirements.

Publish/Subscribe (pub/sub) systems are particularly suited for large scale information dissemination, and provide the flexibility for users to only subscribe to the information of interest to them. They decouple the information delivered to them from all the other information available and transmitted by a publisher. Moreover, with the use of an appropriate interface, users can select and filter the information desired, irrespective of the publisher of this information. We see that such a capability is eminently suitable in our support of MMORPG.

Content Centric Networking (CCN [1], NDN [2]) is a novel networking paradigm centered around content distribution rather than host-to-host connectivity. This change from a *host centric* to a *content centric* communication capability removes the need for receivers to know and establish context with specific sources of information and for publishers to have an apriori knowledge of the intended recipients. Our recent proposal of a Content-Oriented Publish/Subscribe System (COPSS) [3] enhances NDN with a push based multicast capability and uses the notion of hierarchical Content Descriptors (CDs) that are employed by users to subscribe to information that is published by any end-system in the network. COPSS facilitates a highly dynamic and large scale pub/sub environment and is able to deliver content in a timely manner. In this work, we develop

[1] http://en.wikipedia.org/wiki/Massively_multiplayer_online_role-playing_game

G-COPSS, a content centric communication infrastructure for a decentralized gaming environment leveraging the advantages provided by COPSS. We evaluate the performance of G-COPSS by using a data trace of the Counter-Strike game and show performance gains in terms of aggregate network load and update latency.

The key contributions of G-COPSS to provide an efficient communication infrastructure for MMORPG include:

- G-COPSS is designed as a decentralized gaming platform that leverages the content-centric push-based multicast capability provided by COPSS. Additionally, G-COPSS adapts COPSS to provide the features necessary for a gaming environment.
- G-COPSS provides games with "multi-layer hierarchical map" functionality which is a further step of current map partitioning solution. This enables players to have different size of vision based on their altitude and thus only need to send/receive updates pertaining to that vision. The hierarchical CD based multicast provided by COPSS allows G-COPSS to send updates efficiently in such hierarchical map.

We review the related work in §II and present a brief background of COPSS. We present the design of the G-COPSS infrastructure in §III and evaluation results are given in §IV. We conclude our work and outline future work in §V.

II. Related Work

Modern fast-paced action games that run on a Client/Server (C/S) architecture limit the number of players who can interact simultaneously since the server needs to handle the frequent updates and disseminate it. Feng et. al. in [4] show that Counter Strike (CS), which is a popular server based game can host on an average of 22 players/game on a server. Another popular MMORPG, Second Life (SL), has dedicated servers to support each of its 18,000 regions [5]. Nevertheless, studies such as [5]–[7] show that SL makes intensive use of network resources and that an Avatar action consumes about 20Kbps in the down-link and a movement made by the avatar could consume upto 110 Kbps on the down-link. Stenio et. al. in [6] also show that the management of a region with only 5,000 rigid-body objects requires about 72% of the server computational power.

Peer-to-peer (P2P) approaches such as those proposed in [8]–[12] manage the virtual worlds in a distributed manner by leveraging end-user resources and therefore provide a scalable and cheap alternative to C/S approaches. E.g., Varvello et. al. propose a DHT based architecture for SL in [12], [13] to overcome the limitations of a client/server based SL. Bharambe et. al. proposed Donnybrook [11], a system that is designed to handle games without server support. The shortcomings of such DHT based solutions are that in incurs management overhead and network overhead since it is agnostic to the underlying topology.

G-COPSS leverages the network layer based content centric benefits provided by both NDN [2] and COPSS [3] and adapts COPSS to provide a decentralized gaming platform that is

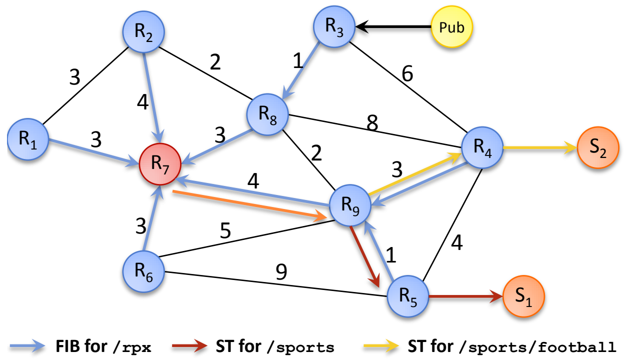


Fig. 1: One step dissemination model of COPSS.

scalable for use by MMORPG. G-COPSS is therefore also able to utilize the network topology to provide efficient and timely content dissemination.

To achieve communication efficiency, we proposed **Content-Oriented Publish/Subscribe System (COPSS)** in [3], which enhances NDN using hierarchical content descriptors, and uses a push-based dissemination framework. This also relieves the users in NDN from having to know the name of every piece of data beforehand. Instead, they express interests in *Content Descriptor*s (CDs), e.g., `/sports/soccer`. Data providers (publishers) send announcements related to a CD when they have a new piece of data. CDs are grouped in hierarchical structure so that subscribers of higher level CDs can also receive announcements of lower level CDs, e.g., a subscriber of `/sports` can also receive announcements of `/sports/soccer`, `/sports/swimming`, etc. NDN requires a new forwarding engine to perform the basic CCN-related operations. The forwarding engine contains a FIB (Forwarding Information Base), a Content Store and PIT (Pending Interest Table). The FIB is used to forward Interest packets toward potential source(s) of matching Data. COPSS aware routers are equipped with an additional Subscription Table (ST) that maintains CD-based subscription information downstream of them in a distributed, aggregated manner, as in IP multicast.

Fig. 1 illustrates COPSS's support for push-based information delivery. R_7 serves as the rendezvous point rp_x that serves CD `/sports`. R_7 will then have to propagate this information to the whole COPSS aware network that would then create a shortest path tree in the FIB (prefix=`/rpx`) in a decentralized manner. Note that R_7 can serve as other rendezvous points (i.e. rp_y, etc.) too. When a subscriber sends $Subscribe$ packet with prefix `/sports`, its 1st hop router will update the ST and send the $Subscribe$ packet upstream based on the FIB entry for `/rpx`. In Fig. 1, S_1 subscribes to `/sports` implying that he would like to receive all the data addressed with the prefix `/sports` and S_2 subscribes to `/sports/football` thereby implying that he would like to receive all the data addressed with the prefix `/sports/football`. When Pub sends a $Multicast$ packet, its 1st hop router (R_3) will encapsulate it into an $Interest$ packet with prefix `/rpx` and the COPSS network will forward it to R_7. R_7, then decapsulates it and sends $Multicast$ packet(s) based on its ST. Although this packet will have to be delivered to 2 groups, only one packet will be sent to R_9. On receiving the packet, R_9 will perform an ST lookup, replicate the packet and forward them to R_4 and R_5. Both S_1 and S_2 will thus receive this packet. If Pub sends a $Multicast$ packet with prefix `/sports/swimming`, R_9 will only send it to R_5 and S_1 will be the only recipient.

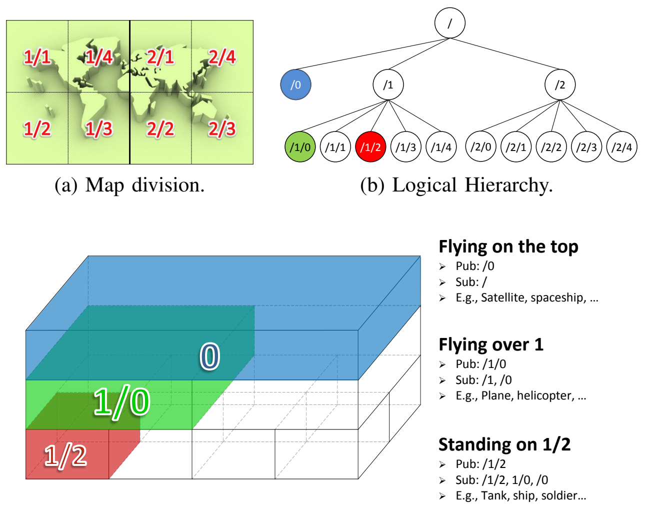
III. G-COPSS: An Efficient Communication Infrastructure for MMORPG

Map partitioning (e.g. Binary Space Partitioning, k-d tree, Octree) is a well-known technique to help increase the update efficiency in 3D game engines like Quake-derived engines, Cube 2, and Doom engines since it can relieve the server from performing tasks such as location approximation and barrier detection. However, it is desirable to further sub-divide the environment into hierarchical regions which enables players to have a different area of vision based on their altitude and thus only need to send/receive updates pertaining to that vision. We believe that it is even more desirable to have a content centric network-layer support for this kind of feature. After mapping areas into CDs, players can publish and subscribe to higher level CDs representing the whole area instead of the leaf CDs that compose that area. This helps to reduce the number of states maintained in the network and the network load caused by the dissemination of updates.

G-COPSS is designed as a decentralized framework to support this feature. It utilizes the benefits provided by COPSS for efficient content dissemination. While COPSS is designed to support content oriented publish-subscribe environments in general, G-COPSS fine-tunes COPSS to support the specific needs of a game environment. G-COPSS utilizes COPSS as a predominantly push based framework to ensure that the players receive timely updates. It is a content-centric network solution (which we envisage will eventually be part of a network layer) which overcomes the disadvantages of server-based and P2P solutions that are agnostic to the underlying network topology.

In G-COPSS, we make the basic assumption that all players have access to the game-map via the game client that was downloaded apriori. For practical reasons, such as efficient broadcast of updates, it is quite natural for game designers of online games to partition the game map into various regions. In G-COPSS, we take that optimization a step further by introducing the need for a multi-layer hierarchical relationship between the various areas in the environment. G-COPSS therefore divides the game-map into a set of multi-layered hierarchical areas as shown in Fig. 2 and uses the hierarchical CD based groups of COPSS to represent these zones. Players subscribe to the groups that represent the areas that they are currently involved or located in.

We create prefix-free virtual rendezvous points (RP) that are responsible for receiving updates from players and disseminating it to the other players belonging to the same or higher groups in the hierarchy. The term prefix-free mandates



(a) Map division.

(b) Logical Hierarchy.

(c) Hierarchical subscription.

Fig. 2: Hierarchical map partition.

that a prefix is served by only one RP, e.g., if an RP is serving the prefix `/1/1`, there would not be RP serving `/1` or `/1/1/1`, ensuring that messages will only be sent to one RP for every CD associated with it. G-COPSS uses a central server only to maintain an up-to-date snapshot of various zones by subscribing to them. This is done to ensure that whenever a player moves to a new zone, the server would be in a position to send the current snapshot to the newly arriving player. This significantly reduces the load on the server and thereby ensures that the queueing delays at the server do not impair the playing experience.

A. *Hierarchy Creation*

As explained above, G-COPSS exploits the game designer's partitioning of the game-map as a multi-layer hierarchy. This allows the mapping of the zones/areas to hierarchical CD based groups. G-COPSS allows map designers to divide map into arbitrary layers, but for simplicity, we only use 3 layers in this paper. Fig. 2a shows a world map which is first divided into 2 regions (marked 1 and 2) and each region is further divided into 4 zones (marked $1/1-2/4$). The hierarchy created for the map is shown in Fig. 2b. Note that we create a `/0` for every non-leaf CD in the hierarchy, i.e., `/0` for top (`/`) and `/1/0` for `/1`. These `/0`s are used to represent the areas above that e.g. represent the area where planes are flying(shown in 3D partition Fig. 2c) so that every area in the game world is represented by a leaf node in the logical hierarchy. E.g., the green area above $1/1$ to $1/4$ is represented by `/1/0`, and the blue area above $1/0$ and $2/0$ is represented by `/0`.

B. *Update Message Dissemination for Online Players*

Using this modified hierarchy, a player can send an update using a leaf CD representing the area that contains the object he has modified. He can also subscribe to the leaf CDs that represents the areas he has visibility into (his Areas of Interest

(AoI)). Furthermore, subscription to CDs can be aggregated at some higher level in the hierarchy. According to [4], almost all of the packets in a gaming application are under 200 bytes. Therefore the one-step model of COPSS, where the data is directly pushed to the subscribers, is used by G-COPSS to disseminate update/control messages. Moreover, in a game, a player is a publisher as well as a subscriber, and therefore G-COPSS allows players to publish and subscribe using different CDs according to the game semantics. Now, we describe how G-COPSS maps the game logic onto COPSS pub/sub relationships.

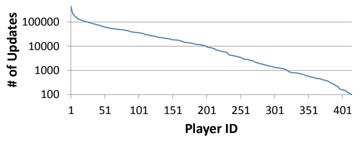
Hierarchical Publishing: Players need to publish the updates they make to all interested-players (interested-players are those that view the same AoI/objects). The player's client is therefore responsible for sending the updates to the RPs that are responsible for the groups comprising the interested-players. E.g., If a player moves a satellite in the blue area in Fig. 2c, he will send the message to the RP serving the CD `/0`; if he shoots at a plane in the green area, he will disseminate the message using the CD `/1/0`; and if a soldier is moving in the red area, he will disseminate it using the associated CD `/1/2`.

Hierarchical Subscriptions: According to the semantics of the hierarchical map, players should be able to see all the updates below and vice versa. Therefore, a player will subscribe to the area he is in and all the `/0`s along the hierarchy. E.g., a player standing on $1/2$ should subscribe to `/0`, `/1/0` (the `/0`s along the hierarchy) and `/1/2` (the area he is in). This allows him to see the units standing on $1/2$, the planes flying over zone 1, and the satellite at top (so that he will not be shot without knowing who did that). Likewise, a player flying over 1 will see the units standing on $1/1 - 1/4$, those flying over 1 (area $1/0$) and also those on top (area 0). Note that the CDs of `/1/1` to `/1/4` and `/1/0` could be aggregated to `/1` implying that the player can therefore subscribe to `/0` (the `/0`s along the hierarchy) and `/1` (the area he is in).

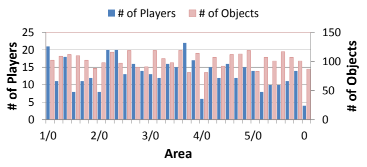
C. *Usage of COPSS vs. COPSS+IP*

COPSS+IP was introduced in [3] as an incremental deployment step wherein the COPSS enabled routers at the edge are used to provide content centric functionality while the intermediate IP routers provide the forwarding efficiency. In this section, we describe how G-COPSS can be designed to function in a COPSS+IP environment.

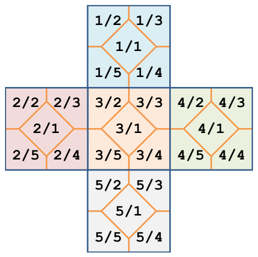
The key to adopting G-COPSS in a COPSS+IP environment (hybrid-G-COPSS) is to map the multitude of hierarchical CDs to the limited IP multicast space. An ideal scenario would be to perform a one-to-one mapping of the leaf CDs. But taking into consideration the fact that billions of CDs could exist in a content centric environment, this could result in more than one CD being mapped to a single IP multicast address. In G-COPSS, based on the available IP multicast address space, the COPSS enabled edge routers would hash the high level CDs on to a single IP multicast address rather than directly hashing the leaf CDs. This allows the mapping tables at the edge COPSS routers to aggregate mapping entries. Moreover, this mechanism allows a message received by `/1/1/1` to be forwarded to the players subscribed to `/1/1` and `/1` too. Due to the mapping of multiple CDs on to one IP multicast group, unwanted messages too may be forwarded along branches of the network. E.g, if `/1` and `/2` are mapped to one IP group, a message to `/1` would be sent to all the subscribers including those subscribed to `/2`. The COPSS enabled router close to the receiver side is then entrusted with the task of filtering out the unwanted messages and forwarding only those messages intended for the receiver.



(a) # of updates per player.



(b) # of players per area.



(c) Game map.

Fig. 3: Simulation Setup.

IV. Experimental Evaluation

A. *Experimental Setup*

To evaluate G-COPSS, we use a self developed simulator and use a game model derived from the data trace of the Counter-Strike (CS) game.

*1) **Event trace:*** We use a CS data trace obtained during the peak of a day [14] as a representative trace of the stress on a CS server. It consists of a Wireshark trace collected on a busy CS server in a $7h05m25s$ period, which totaled $20,000,000$ packets sent to and from the server by $32,765$ different addresses ($59,294$ different address:port). Since the packets contained indecipheareable information and represents a server based game, we performed the following operations to filter out packets that represent a de-centralized gameplay: 1) discarded all the packets sent from the server since G-COPSS does not require a server, 2) discarded packets with address:port that send less than 100 packets to obtain the trace of the established connections (clients really play games) since the trace consisted of many attempted connections (clients echo server to get RTT) too, and 3) we assumed that every unique address represents a unique player. This resulted in a data trace consisting of 414 unique players and $10,686,950$ packets (updates) in the same period of time as the original trace. Fig. 3a shows the number of updates performed by the the different players.

*2) **Game map and player distribution:*** The 414 unique players share a global game-map shown in Fig.3c. The game-map is converted to a hierarchical map by dividing it into

5 regions (marked $1-5$) and further dividing every region into 5 zones (marked $1/1-5/5$). This results in 31 leaf CDs (twenty-five for bottom layer ("zones") marked $1/1-5/5$, five for middle layer ("regions") marked $1/0-5/0$ and one for the top layer ("world") marked 0) in the hierarchy. At the start of the simulation, the players are randomly distributed into the various areas as shown in Fig.3b resulting in a maximum of 22 players and a minimum of 4 players per area. Furthermore, as shown in Fig.3b, the areas are further divided into objects ranging from 80 to 120. A player would be able to see and modify these objects depending on his location in the game and the hierarchy of the area he belongs to as defined in §III.

3) Network topology: We use the Rocketfuel [15] backbone topology (id=3967) for the core routers. In addition, we have a total of 200 edge routers that are homed on the 79 core routers, with each core router having 1-3 edge router(s) connected to it. We uniformly distributed the 414 players on the edge routers. The link weights between the core routers were obtained from the topology and interpreted as delays (in milliseconds). The delay between each edge router and its associated core router is set to 5 ms; the delay between each host and its associated edge router is set to 10 ms.

We randomly distribute the RP(s) on the core routers to evaluate COPSS. To make a fair comparison, we place the server(s) on the router(s) which served as RP(s) for the COPSS evaluation to perform the IP server based simulations. We also use the same map (area-RP/Server) for both the evaluations.

B. *Update Message Dissemination for Online players*

In this section, we evaluate the performance gain of G-COPSS in disseminating updates compared to that of a game operating in an IP network with servers. Here, we assume a stable state where all the players are attached to their areas of interest and do not move for the whole duration of the simulation.

The need for decentralization: We emulate the game-play using the first 100,000 update packets from the event trace to evaluate the average latency incurred in delivering an update from the publisher (player who performed the update) to all the other subscribers (players subscribed to the CDs the update is intended for) for different number of RPs and servers. The average update frequency observed in the event trace is about $2.40ms$. In our simulations, an RP's processing time (including FIB lookup, packet decapsulation and ST lookup) is set to $3.3ms$ (based on benchmark measurements perfomed on standard PCs [3]), and the the server processing time is twice the RP processing time since it is an application layer function. Table I shows that the update latency incurred in the case of 1 or 2 RPs is high because of congestion at the RPs (the processing time at an RP is higher than the update frequency observed at each RP). When we added another RP, this mitigated the effect of congestion. When the number of RPs is equal to 3 or more, no congestion is observed. Fig. 4 shows the minimum, maximum and average update latency of every update in a 3-RP situation. We observe that the update latency lies below $200ms$ which is below

TABLE I: Performance of G-COPSS and IP server with different RP(s)/Server(s)

Type	# of RP/Server	Update Latency (μs)	Network Load (byte)
G-COPSS	1	47,680,290.87	5,590,798,485
	2	558,179.55	5,657,133,239
	3	94,885.80	5,557,641,484
IP Server	1	249,679,699.91	9,735,117,116
	2	71,991,918.39	10,001,374,835
	3	21,448,167.83	9,622,658,081

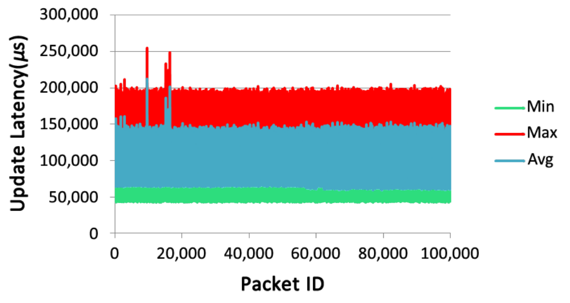


Fig. 4: Simulation Time vs. Update Latency (3-RP)

the generally considered acceptable latency for such games ($300ms$ to $1s$ [16]).

Another observation in Table I is that update latency and aggregate network load in G-COPSS is smaller than that of an IP server scenario with the same number of servers despite the fact that IP routers are 50-500 times faster than NDN routers according to [3]. This is mainly due to the fact that the IP servers need to disseminate the information via unicast to all the individual subscribers wherein G-COPSS is able to perform hierarchical CD based multicast. The IP server based approach needs more computational power and network bandwidth even in decentralized case.

V. Future Work

In this work, we presented G-COPSS that functions as an efficient decentralized communication infrastructure for a gaming environment by leveraging the advantages provided by a content-centric network. We showed that it is able to outperform a pure IP server-based gaming environment in terms of aggregate network traffic and update latency. We are currently working on a full fledged gaming platform. Additionally, we are performing extensive evaluations of the proposed communication model and comparing it to the state of the art gaming solutions.

References

[1] V. Jacobson, D. K. Smetters, J. D. Thornton, M. F. Plass, N. H. Briggs, and R. L. Braynard, "Networking named content," in *CoNEXT*, 2009.

[2] L. Zhang, D. Estrin, J. Burke, V. Jacobson, and J. Thornton, "Named data networking (ndn) project," PARC, Tech. Report NDN-0001, 2010.

[3] J. Chen, M. Arumaithurai, L. Jiao, X. Fu, and K. K. Ramakrishnan, "Copss: An efficient content oriented publish/subscribe system," in *ANCS*, 2011.

[4] W. chang Feng, F. Chang, W. chi Feng, and J. Walpole, "A traffic characterization of popular on-line games," *Networking, IEEE/ACM Transactions*, vol. 13, no. 3, pp. 488 – 500, 2005.

[5] S. Kumar, J. Chhugani, C. Kim, D. Kim, A. Nguyen, P. Dubey, C. Bienia, and Y. Kim, "Second life and the new generation of virtual worlds," *IEEE Computer*, vol. 41, no. 9, p. 4653, 2008.

[6] F. Stenio, K. Carlos, S. Djamel, M. Josilene, and A. Rafael, "Traffic analysis beyond this world: the case of second life," in *NOSSDAV*, 2007.

[7] M. Varvello, S. Ferrari, E. Biersack, and C. Diot, "Exploring second life," *Transaction of Networking*, vol. 19, no. 1, p. 8091, 2010.

[8] J. Keller and G. Simon, "Solipsis: A massively multi-participant virtual world," in *PDPT*, 2003.

[9] A. Bharambe, J. Pang, and S. Seshan, "Colyseus: A distributed architecture for online multiplayer games," in *NSDI*, 2006.

[10] S.-Y. Hu, J.-F. Chen, and T.-H. Chen, "Von: A scalable peer-to-peer network for virtual environments," *IEEE Network*, vol. 20, no. 4, p. 2231, 2006.

[11] 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," in *Sigcomm*, 2008.

[12] M. Varvello, S. Ferrari, E. Biersack, and C. Diot, "A distributed avatar management for second life," in *Netgames*, 2009.

[13] M. Varvello, C. Diot, and E. Biersack, "P2p second life: experimental validation using kad," in *Infocom*, 2009.

[14] W. Feng, "On-line games," http://www.thefengs.com/wuchang/work/cstrike/.

[15] R. Mahajan, N. Spring, D. Wetherall, and T. Anderson, "Inferring link weights using end-to-end measurements," in *IMW*, 2002.

[16] M. Claypool and K. Claypool, "Latency and player actions in online games," *ACM Communication*, vol. 49, no. 11, p. 4045, 2006.