Evaluation Of The HyperVerse Avatar Management Scheme Based On The Analysis Of Second Life Traces

Markus Esch
University of Luxembourg
Email: markus.esch@uni.lu

Wei Tsang Ooi
National University of Singapore
Email: ooiwt@comp.nus.edu.sg

Ingo Scholtes
University of Trier
Email: scholtes@syssoft.uni-trier.de

***Abstract*—Massive Multiuser Virtual Environments (MMVEs) and the idea of a global scale 3D Web have grown popular in recent years. While commercial precursors of such environments for the most part rely on centralized client/server architectures, it is commonly accepted that a global scale virtual online world can only be realized in a distributed fashion. Within the HyperVerse project, we have developed and recently presented a two-tier Peer-to-Peer (P2P) architecture that incorporates a loosely structured P2P overlay of user peers and a highly structured overlay of server machines constituting a reliable backbone service. In such a distributed environment, an essential question is how avatars are tracked and interconnected in order to allow mutual rendering and interaction. We have previously proposed a hybrid avatar management scheme that utilizes the backbone service for avatar tracking if necessary, but handles tracking in a P2P fashion when peers can track each other to reduce the backbone load. This paper presents a detailed performance analysis of this algorithm under a realistic scenario, using traces from a large scale MMVE called Second Life. Moreover this paper presents and evaluates an optimization for the hybrid avatar tracking scheme that can be utilized under a weaker condition.**

***Keywords*-P2P, MMVE, DVE**

I. Introduction

Virtual online environments like Second Life or World Of Warcraft currently attract a lot of attention. Though much research work is done in the field of distributed architectures for such environments, the commercial solutions for the most part still rely on centralized client/server architectures due to advantages in terms of manageability and controllability by the publisher. The drawback of such a centralized solution is the limited scalability. Having global scale scenarios like a 3D Web in mind, it is commonly accepted that distributed Peer-to-Peer (P2P) technologies need to be applied. Within the HyperVerse project, we study the feasibility of P2P technologies for a global scale virtual online environment. We envision the 3D Web to be a combination of virtual globes like Google Earth and avatar-based interaction. Similar to reality, a user can move through a virtual representation of the real world in order to meet friends, undertake a sightseeing tour, shop and so forth. In [1], we have proposed a two-tier P2P architecture as basic infrastructure for such a 3D Web scenario. This architecture incorporates a loosely structured P2P overlay of user machines and a highly structured overlay of reliable server machines. The federation of server machines is responsible for the reliable and persistent hosting of the online world while the client overlay is primarily utilized for data distribution. Similar to today's Web, for the provision of the server machines, we rely on the incentive of being able to publish information in the 3D Web.

In centralized client/server systems, the interconnection of peers in virtual proximity is not an issue, since the server typically has a global view and hence is able to make two peers in virtual proximity mutually aware. Due to the absence of a global view, interconnection of peers in virtual proximity becomes an issue in P2P systems. If the graph falls apart into several connected components, avatars in different components will not be able to interact. In [3], we have presented an avatar management scheme for the HyperVerse infrastructure that handles avatar interaction in a hybrid fashion utilizing both the backbone service and the user machines. In opposite to many other P2P-based infrastructures for virtual environments, in HyperVerse, it would be possible to utilize a centralized solution for the avatar management, since the backbone service is existing. It is obvious that such an approach would not fit the requirements of a global scale environment, since it is likely to overburden the backbone service even if this service is a federation of multiple server machines. Especially in highly populated regions of the world, the server responsible for such a region may become overloaded. Utilizing our hybrid scheme, the server load is reduced in a self-scaling fashion. Though the backbone service is available as a reliable fallback solution, the avatar management is automatically handled in a pure P2P fashion whenever the peer density in a certain region is high enough.

This paper presents a detailed analysis and evaluation of HyperVerse's avatar management scheme using actual avatar traces from Second Life. Additionally, this paper presents and evaluates an optimization for our avatar interaction scheme that can be utilized under weaker conditions than we previously proposed, leading to lower load on the backbone servers, especially in lower density regions.

Section II first briefly describes our hybrid avatar manage-

ment scheme and presents the aforementioned optimization. The evaluation is presented in Section III. Before concluding the paper in Section V, Section IV presents other approaches related to our work.

II. HyperVerse Avatar Interaction

The HyperVerse infrastructure relies on a two-tier P2P architecture incorporating two P2P overlays, a loosely structured and a highly structured overlay. The basic scheme of this architecture is shown in Figure 1. The highly structured backbone overlay of reliable server machines is responsible for the reliable hosting of the virtual world, while the loosely structure overlay of user clients is primarily utilized for data distribution. A detailed description of this scheme has been presented in [1]. One of the tasks of the backbone service is tracking of the avatar positions to interconnect clients in virtual proximity. To reduce the load of the backbone, especially in highly populated areas of the world, a hybrid avatar management scheme has been presented in [3]. This scheme automatically manages the avatar interaction in a pure P2P fashion among the clients once the avatar density in a certain region is high enough.

This scheme uses a certain condition for the decision when to manage the avatar interaction in P2P fashion. While we have previously considered only a *sufficient* condition, this paper considers both the *necessary and sufficient* condition under which the avatars can interact in a P2P fashion, leading to slightly more complex computations. We aim to study how considering both conditions affects the performance of the algorithm. The following subsection describes the algorithm and the sufficient condition, while the necessary and sufficient condition is presented in subsection II-B. To simplify matters, from now on we refer to the necessary and sufficient condition just by necessary condition.

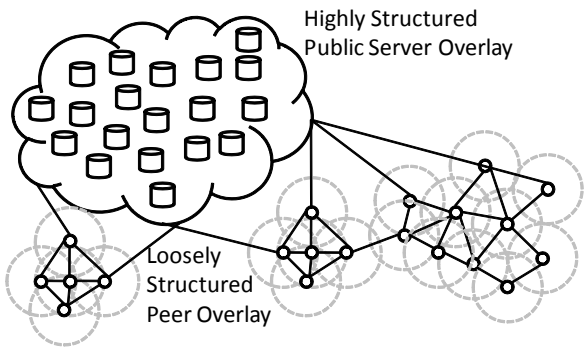

Figure 1. Two Tier HyperVerse Infrastructure

A. Sufficient Condition

In our avatar management scheme, each avatar basically has an *Area of Interest (AoI)* which is defined by a circle with a certain radius around the avatars current position. For the avatar management, the three dimensional avatar positions are projected to the two dimensional plane To allow mutual rendering and interaction among avatars in virtual proximity, all avatars with intersecting AoIs need to be interconnected. This requires the tracking of avatar positions. Utilizing solely the backbone service for this task would overburden this service in highly populated regions of the world because of the huge amount of position update messages. For this reason, we proposed a scheme that reduces the server load by handling avatar tracking and interconnecting in a pure P2P fashion if a certain condition is fulfilled. We classify a client as either in *backbone-mode* or *cluster-mode*. While a node in backbone-mode sends movement updates to the backbone, a node in cluster-mode stops sending these updates. By default a node is in backbone-mode and switches to cluster-mode if the node is *covered* according to the following sufficient condition:

Definition. *A node is covered if the whole AoI fringe of the node is covered by overlapping AoIs of neighbors.*

To clarify this definition, Figure 2 depicts nodes with their AoIs. In this example, A is a node in cluster-mode since the whole AoI fringe is overlapped by AoIs of neighbors. Node B in this figure is in backbone-mode since not the whole AoI fringe is covered. Note that not the whole AoI area of a node needs to be covered but only the fringe of the AoI. If a node is covered, the node and all its neighbors form a *cluster*. A cluster can grow in size if new nodes join the cluster. We distinguish two types of nodes in a cluster. Covered nodes are so-called *core-nodes*, the uncovered nodes at the boundary of a cluster are *border-nodes*.

Each node can assess whether it is in cluster- or in backbone-mode based on local information about its own position and the positions of its neighbors. If a node detects that it became core-node of a cluster, it stops sending position updates to the backbone service and henceforth just sends updates to the neighbors. Note that the updates to the neighbors have to be sent anyway in order to allow mutual rendering, hence this does not imply additional overhead.

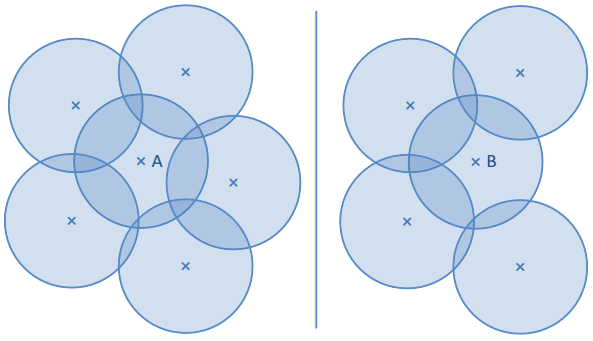

Figure 2. Covered vs. Uncovered. Node A Is Covered While Node B Is Uncovered. The Figure Shows The Nodes With Their Area Of Interest (AoI)

Since the AoI fringe of all nodes in the cluster core is covered by the AoIs of their neighbors, it is impossible for a node to move inside another core node's AoI without crossing the AoI of at least one neighbor. Consequently each core node can be interconnected with new nodes by the neighbors. Node *A* in Figure 3 for instance is a core node with its whole AoI fringe being covered by neighboring AoIs. If node N moves inside *A*'s AoI, it first needs to pass the AoI of *B*. Since *B* is a border node, *N* and *B* are connected by the backbone. Consequently, at some point in time *B* is a neighbor of both *A* and *N*, and therefore *B* receives movement updates from both nodes. Based on this information *B* is able to observe when the AoIs of *A* and *N* intersect and thereupon can interconnect both nodes.

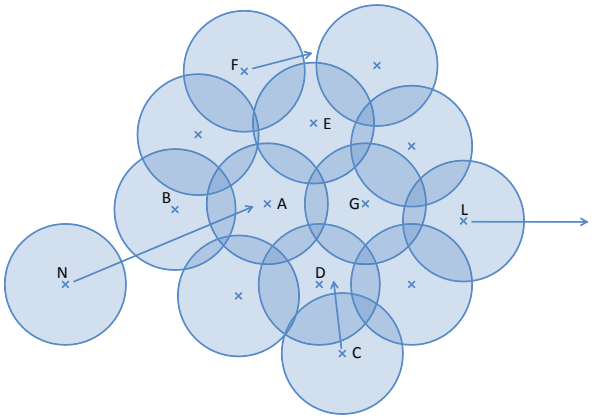

Figure 3. A Cluster Of Peers

This way it is possible to handle the avatar tracking and interconnection of nodes in the core of a cluster in pure P2P fashion. If a core-node detects that it is not covered any more and that it, hence is not in cluster mode any more, it starts sending update messages to the backbone service again. For the correctness of the avatar tracking in the cluster, it is crucial that the border of a cluster is managed correctly. For that reason we have to consider the following four situations that may occur at the cluster border and that need to be handled correspondingly:

- **A new node moves into the cluster:** Consider for example node N in Figure 3. This node moves towards the cluster, thus the AoI of N intersects with one of the cluster border nodes at some point (in this case node B). Since the backbone service has location information about the nodes constituting the border, it is able to interconnect node B to N, which then also becomes part of the cluster border.
- **A core node becomes part of the border:** This situation occurs either if a core node itself moves towards the border, or if a border node previously covering a core node moves away so that the core node is no longer covered. Consider node C in Figure 3 which moves towards the core and passes D, so that D is no longer part of the core. A former core node like D becoming border node can detect this situation based solely on local information about its neighbors. It then sends a position update to the backbone service in order to indicate that it is no longer inside of a cluster core.
- **A border node becomes part of the core:** A border node can become part of the core either if (a) the node itself moves to a point where its whole AoI fringe is covered, or if (b) other nodes move in such a way that they cover the whole AoI of the border node. For example consider node E and F in Figure 3. E becomes part of the core because F moves to a position where it covers E's AoI fringe. Another example is node C that becomes part of the core by moving into the core itself. A node observing that it has become core node sends a message to the backbone to inform the backbone about not expecting any further updates until the node leaves the cluster again and sends its new position.
- **A node leaves the cluster:** If a node leaves the cluster, e.g., node L in Figure 3, this has no effect on the node itself. As it was previously already part of the border and it had to send movement updates to the backbone anyway. This may, however, causes a core node to become a border node (node G in the example).

B. Necessary Condition

In this section, we present the necessary condition that can be used for the decision of switching to cluster-mode as well. Though this condition implies a slightly higher computational complexity, it does not affect the message complexity. As we will see in section III, in the most cases it can further reduce the backbone load by increasing the number of nodes in cluster-mode.

Following the sufficient condition described above, a node switches to cluster-mode if the whole AoI fringe is covered by overlapping AoIs of neighbors. In the case, that the AoI of all nodes has the same size and is fixed, this condition can be generalized by considering the fact that, though not the whole AoI fringe is covered, it may still be impossible for a new node to enter the AoI without crossing a neighbor (See Figure 4). In this example, the AoI fringe of *A* is not completely covered, but, nevertheless it is obviously not possible for node *B* to move inside *A*'s AoI without crossing *C* and *D*. For that reason we can define the following necessary and sufficient condition for the coverage of a node:

Definition. *In the case that all nodes have the same fixed AoI radius, a node* A *is covered if no gap between two covered segments of the AoI fringe exists that allows the movement of a new node into* A*'s AoI without crossing a neighbor of A.*

In order to verify this condition mathematically the situ-

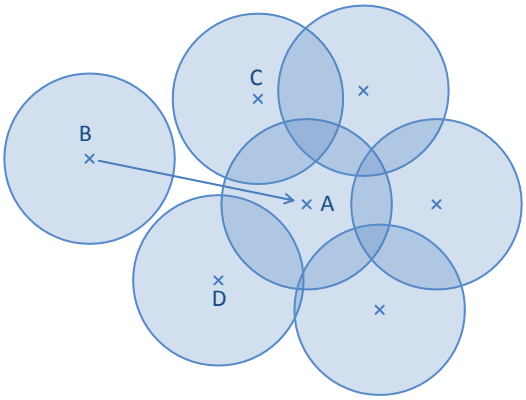

Figure 4. Optimized Condition Example

ation depicted in Figure 4 needs to be considered. Let node A be the node in question. The whole AoI fringe of A is covered except the gap between node C and D. In order to figure out whether this gap is large enough to allow a new node to enter A's AoI or not, first the intersection point p of the two circles with radius $2r$ (with r being the AoI radius) and center A and B respectively has to be calculated. A node can not move closer than to this point p towards B and A without intersecting the AoI of at least one of them. By checking whether the circle with radius r and center p intersects with C's AoI we can figure out whether the gap is too large. If, as shown in Figure 5, the circle does not intersect, it is obvious that it is possible for a new node to enter A's AoI without crossing a neighbor and A is not covered. Otherwise, as depicted in Figure 6, if the circle with center p intersects with C's AoI, it is not possible to enter A's AoI without crossing C. Hence A is covered, though not the whole AoI fringe is overlapped by AoIs of neighbors.

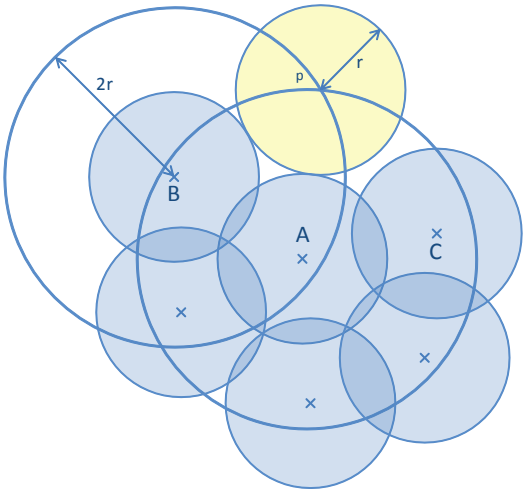

Figure 5. Verifying The Optimized Condition: A Is Not Covered

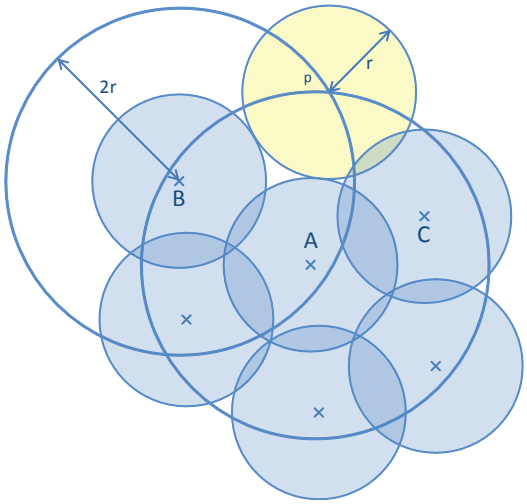

Figure 6. Verifying The Optimized Condition: A Is Covered

III. Evaluation

This section presents the performance analysis of our avatar tracking scheme. We evaluated our scheme by analyzing avatar traces from Second Life with respect to our algorithm. Traces from four different Second Life regions, each trace spanning a period of 24 hours where analyzed. Table I shows the number of nodes per region, the average number of avatars online, the average network density as well as the dates the data was collected. The network density is given by the quotient of the area covered by all nodes and the simulation area. The three regions *Isis*, *Freebies* and *Pharm* are very popular and crowded. In contrast, the region *Ross* is less popular and exhibits a lower peer density. For more detailed information about the traces and how they were collected we refer to [7].

We analyzed the traces with respect to three different metrics. First, we measured the ratio of peers in cluster-mode for each timestamp of the traces to find out how much the load of the backbone service is reduced. Second, we evaluated the stability of the clusters by measuring the time a node stays in cluster-mode once it became a core-node. Third, we performed analysis with different AoI radii in order to find out how the node densities affect the number of cluster nodes and their stability.

Name	Avatars	∅ Avatars	∅ Density	Date
Freebies	3153	84.63	14.67	11 Mar 2008, Tue
Isis	2735	83.10	14.32	28 Mar 2008, Fri
Ross	560	24.49	4.31	11 Mar 2008, Tue
Pharm	1537	93.01	16.05	5 Mar 2008, Wed

Table I
Basic Second Life Trace Information

A. Cluster-Node Ratio

Figures 7 shows the results of the cluster-node ratio measurement for the sufficient and the necessary condition.

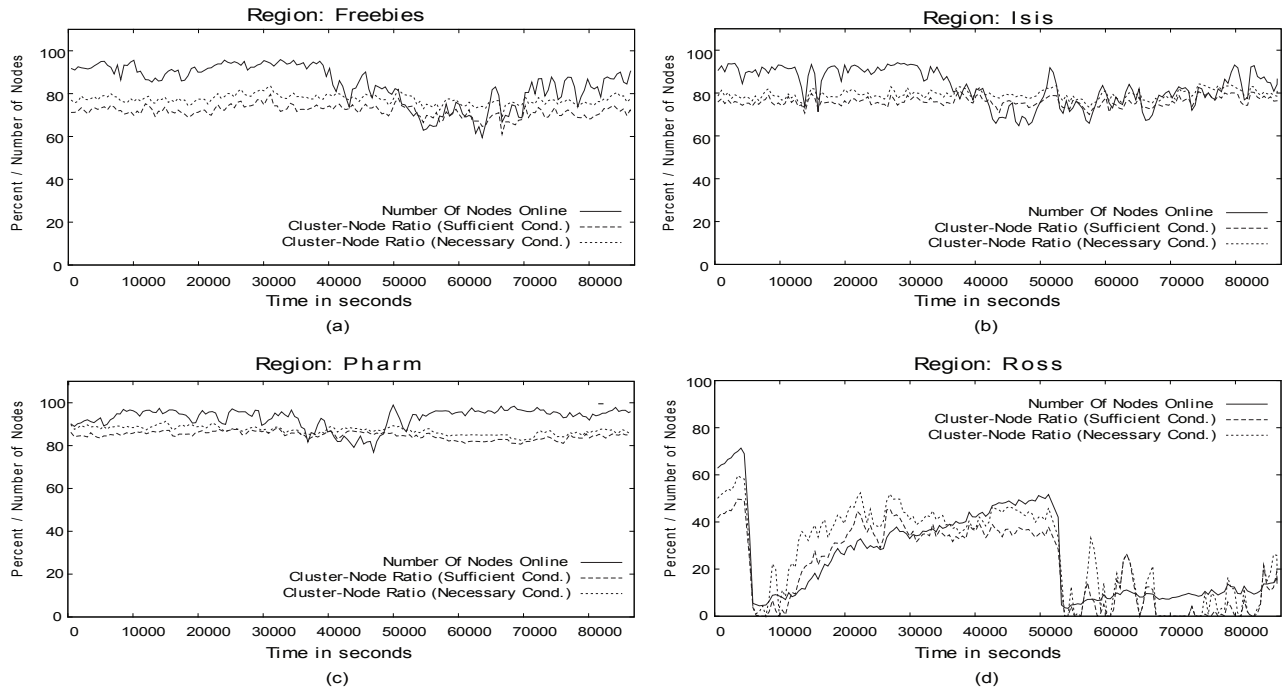

Figure 7. Ratio Of Nodes In Cluster-Mode

We used a fixed AoI radius of 64 meter, which is the standard AoI size in Second Life. Each of the regions has a size of 256×256 meters. To depict that the ratio of nodes in cluster-mode closely correlates with the number of nodes online, the figures also contain the absolute number of nodes online a each timestamp. Especially in the sparsely populated region Ross, we can observe the correlation of cluster-node ratio and population. The other three more crowded regions, all have a cluster-node ratio of about 80% during the whole time using the sufficient condition and a slightly higher ratio using the necessary condition. The absolute number of nodes in this regions is relatively constant around $90 - 100$ nodes. This shows that our scheme reduces the server load in densly populated regions considerably. In the three regions, Freebies, Isis and Pharm, at any time only around 20% of the nodes require the backbone service.

Considering the absolute number of nodes using the backbone and comparing the populated regions with the less populated region Ross, one observes that the backbone load in the highly populated regions is not much higher than in Ross. This is shown in Figure 8, where the absolute number of nodes in backbone-mode for all regions using the sufficient condition is depicted. At some points, the backbone load in Ross actually exceeds the load in the highly populated regions. Only if the number of users in Ross is very low $(0-20)$, the backbone load in Ross is considerably lower. That means using our scheme, the number of nodes requiring the backbone is bounded above, independent of the number of users online. Another interesting point is that, the region with the highest population, Pharm, with 93.01 nodes being online on average, exhibits the lowest server utilization. As we will see in the following section, this can be explained by the usage characteristic of this region.

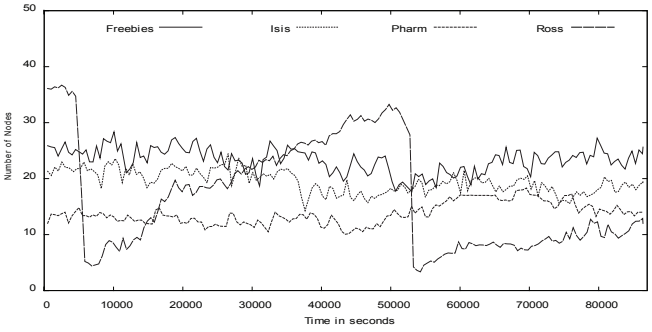

Figure 8. Absolute Number Of Nodes In Backbone-Mode

B. Cluster Stability

Another important criterion for the usability of the scheme is how stable the clusters are. That means for how long a node stays in cluster-mode once it became a core-node. It is important that nodes stay stable in cluster-mode because only in this case the backbone load is reduced. If the nodes constantly oscillate between cluster- and backbone-mode, one could still measure a good cluster-node ratio at any point in time. But the backbone load would not be reduced, since the nodes constantly send messages to the backbone when they switch the mode. To find out how

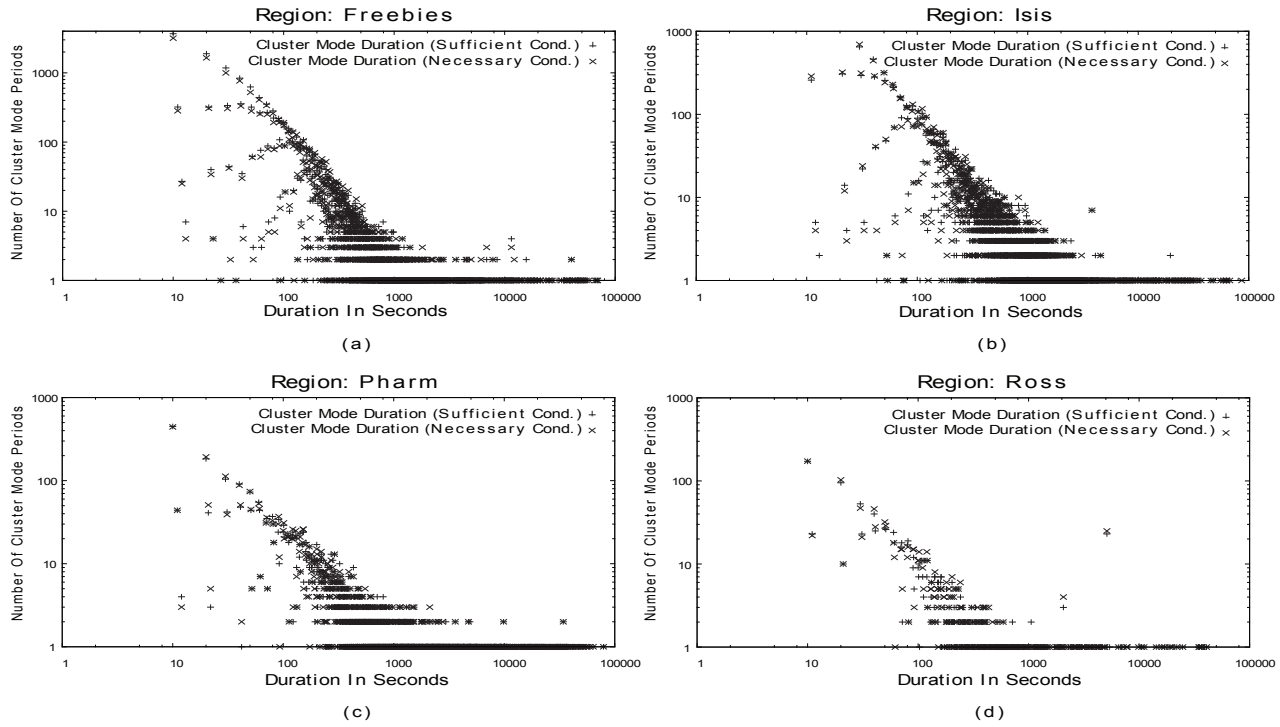

Figure 9. Cluster Stability

stable the clusters are, we measured the time a node stays in cluster-mode based on the Second Life traces. Figure 9 shows the results of this analysis for the four regions. The figures depict the distribution of the occurred cluster-mode durations for the sufficient condition and the necessary condition, while the difference between both measurements is just marginal. It can be observed that the distribution is quite similar in all four regions and the majority of cluster-mode periods lasts between 100 and 1000 seconds. Hence it can be assessed that, in all regions, the nodes remain relatively stable in cluster-mode. This is also substantiated by the average cluster-mode time presented in Table II and the cumulative distribution function (CDF) of the duration of the cluster periods depicted in Figure 10. Considering the average values, it stands out, that the average cluster period of 1739 seconds in the region Pharm is very long and much higher than in the other regions. This can be explained by the usage characteristics of the different regions. Pharm is a camping regions and as shown in [7] the users stay for a long time in this region and pause for a extraordinary long time at the same position. 40 avatars pause for more than 3 hours and the longest observed pause time is over 14 hours. Moreover the majority of users remains in a relatively small area of the region. This leads to the extraordinary long average cluster-mode time. Another oddity can be observed in the regions Isis and Pharm, here the average cluster-mode time using the necessary condition is slightly lower than with the sufficient condition. This is due to the fact that, using the necessary condition, more nodes are in cluster mode compared to the sufficient condition. Further, it seems that some of these additional nodes are not as stable. Thus, for the aim of more stable clusters, the necessary condition is not always advantageous, depending on the characteristic of the user behavior.

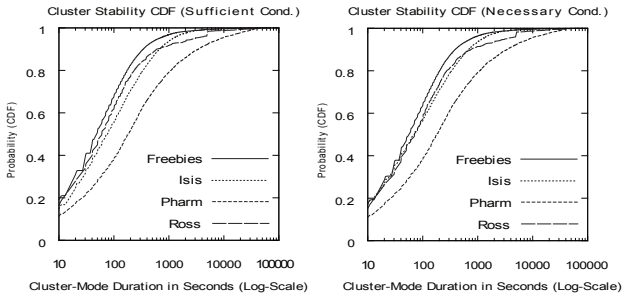

Figure 10. Cumulative Distribution Function (CDF) Of The Cluster Periods For The Basic And The Optimized Condition.

It needs to be noted that the shortest cluster-mode period measured (see Figure 9) is 10 seconds. This is due to the fact that the timestamp interval of the traces equals to 10 seconds. For this reason, the evaluation result cannot achieve accuracy less than 10 seconds, and some of the periods stated with 10 seconds are actually shorter. But since the majority of the measured periods is above 10 seconds and since it can be expected that the duration of the periods under 10

Region	∅ Period(Sufficient Cond.)	∅ Period(Necessary Cond.)
Freebies	276	309
Isis	385	382
Pharm	1739	1730
Ross	675	704

Table II
AVERAGE CLUSTER-MODE PERIOD

seconds is more or less uniformly distributed between 0 and 10 seconds, the bias does not affect the results by the order of magnitudes.

C. Density

In order to analyze the behavior of our scheme under different peer densities, we measured the cluster node ratio as well as the stability using different AoI radii. For this purpose, in addition to the Second Life standard AoI radius of 64 meters, we used radii of 32, 16 and 8 meters. The results are shown in Figure 11 and Figure 12. Figure 11 shows the average cluster node ratio with the different AoI radii. As expected the cluster-node ratio drops with decreasing AoI radius. But we also can assess that the decrement of the cluster-node ratio is very different in the four regions. In Pharm, for example, the reduction is very low, while the cluster-node ratio in Freebies drops very fast. This can be explained by the different usage characteristics. As shown in [7] Freebies is a highly dynamic region with high moving speed and users are widespread over the region. In contrast to this Pharm exhibits lower dynamic and the users remain within a small part of the region. Another interesting result is the fact that the necessary condition becomes more advantageous if the peer density drops. This can especially be asserted in Freebies and Isis. While the advantage in a dense network (AoI radius = 64 meters) is just marginal the necessary condition gets more beneficial if the density drops.

With respect to the stability (see Figure 12) we can assess similar results for two regions. In Pharm and Isis, the necessary condition is very advantageous because the clusters become more stable in denser networks. In opposite, in Ross and Freebies, if the density is low, again the phenomenon occurs that the clusters are less stable using the necessary condition. This is due to the fact that the absolute number of clusters is higher using the necessary condition and obviously in this two regions these additional clusters are less stable. In summary, whether the necessary condition is advantageous with respect to the cluster stability depends on the usage characteristic and the density of the regions.

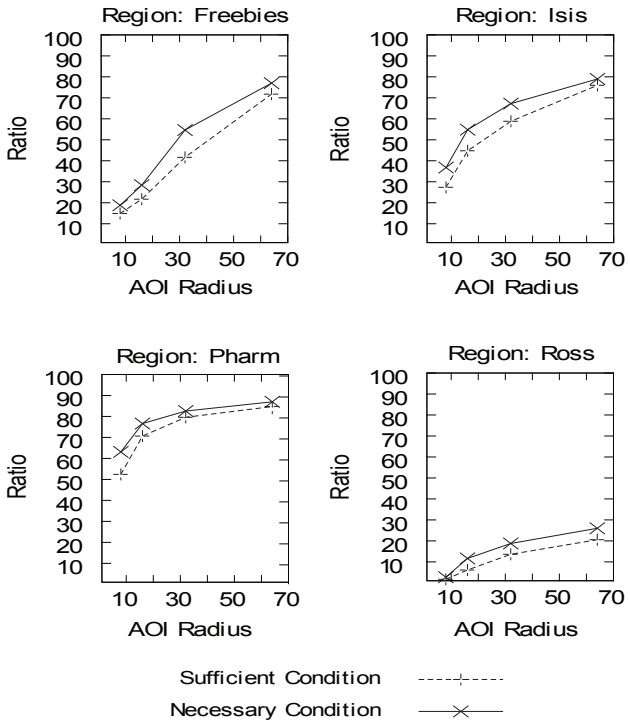

Figure 11. Average Cluster-Node Ratio Using Different AoI Radius (Lines are drawn to guide the eye)

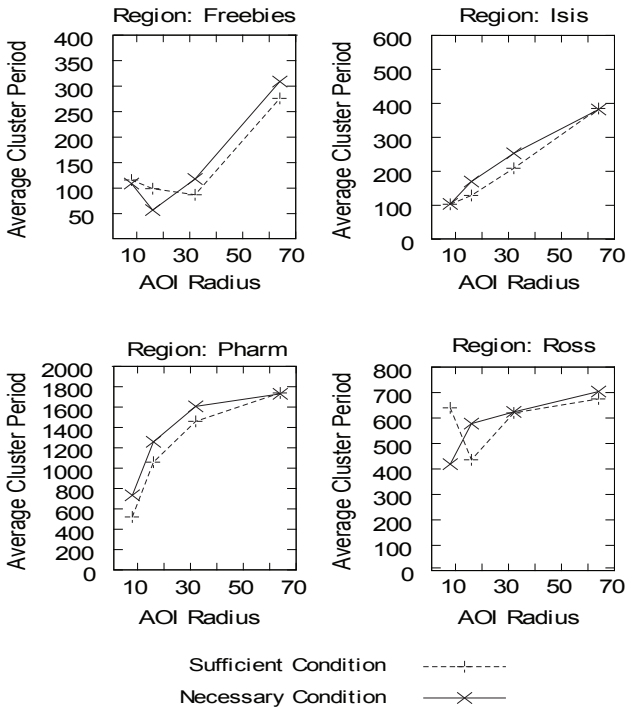

Figure 12. Average Cluster Stability Using Different A0I Radius (Lines are drawn to guide the eye)

IV. RELATED WORK

The impracticality of centralized approaches for the provision of a global scale 3D Web scenario is well recognized. Hence several P2P-based approaches for Massive Multiuser Virtual Environments (MMVEs) exist. Since most of them

rely on a pure P2P scheme without a backbone infrastructure, the avatar tracking in this systems is implicitly handled in a pure P2P fashion. This contrasts to our HyperVerse approach, where we propose the utilization of a federated backbone service in order to guarantee the reliable and persistant provision of the online environment.

A well-known approach in this field is the *Solipsis* project [6], providing a virtual online world in a pure P2P fashion. To allow avatar tracking and interaction, a mesh-like overlay interconnecting clients is applied. Due to the estimated high churn rates of user clients, high maintenance costs for the mesh overlay must be expected. FLoD [5] provides a framework for pure P2P-based 3D scene streaming that is build upon VON [4], a Voronoi-based P2P overlay network distributing the virtual world among peers. Since leaving nodes may result in extensive reorganizations of the Voronoi overlay, this approach is also vulnerable to high churn rates. In [8], an architecture similar to FLoD is described that uses super peers, so-called *connectivity peers*, to interconnect the peers. *VastPark* [9] also utilizes a highly-structured user client overlay in order to form a P2P virtual environment. Here a quadtree in combination with a *Chord* overlay is applied. In order to find other peers, the Hydra [2] architecture provides a central tracker service that can be realized as a single server or as a Distributed Hash Table (DHT).

V. Conclusion

This paper presented a performance analysis of our avatar management scheme published in [3] under realistic conditions using avatar traces collected from Second Life. In addition to the sufficient condition considered previously, this paper presents and evaluates a necessary and sufficient condition that can be used if all peers have the same fixed AoI radius in order to further reduce the backbone load.

The performance analysis presented in this paper shows that the load of the backbone service is considerably reduced by our avatar interaction scheme. Hence the server load is automatically reduced in highly populated regions without the need of any central control. In a self-scalable manner, peers switch to cluster-mode if a certain condition, which can be checked based on local knowledge, is fulfilled. For this reason, our scheme can effectively tackle the problem of flash crowds. Flash crowd refers to the phenomenon in virtual environments that certain regions suddenly and unforeseeable attract a lot of attention, causing a sudden rise of users in this region. Since flash crowds are often unforeseeable it is difficult to handle such situations with proactive countermeasures. Using our avatar management scheme, peers in highly populated parts of the world are likely to be in cluster mode, hence the server is not burdened with avatar tracking in this area at all.

The HyperVerse infrastructure features a two tier infrastructure with a federated backbone service. Though the avatar management scheme discussed in this paper has been developed with respect to this infrastructure, the scheme is not limited to this domain. The definition as well as the evaluation of the scheme does not make any assumptions about the properties of the backbone service. For this reason it is not required to be a federated service, rather the scheme can also be applied in classic client/server systems in order to reduce the server load. All the evaluation results from this paper can be transfered to this setting.

Of course the actual reduction of the backbone load depends on the user behavior in the virtual world. But the analysis of the Second Life traces has shown that, in a typical virtual environment setting, the amount of clusters and the associated reduction of backbone load is very high.

In order to reduce not only the server load but also the network load we intend to extend our scheme that way, that the communication among the nodes is optimized by aggregating updates and sending them in a single message. For this purpose the saved server capacities can be utilized as well in order to get a good tradeoff between network and server load.

References

[1] J. Botev, A. Hohfeld, H. Schloss, I. Scholtes, P. Sturm, and M. Esch. The hyperverse concepts for a federated and torrent based 3d web. *International Journal of Advanced Media and Communication*, 2(4):331–350, 2008.

[2] L. Chan, J. Yong, J. Bai, B. Leong, and R. Tan. Hydra: a massively-multiplayer peer-to-peer architecture for the game developer. In *NetGames '07: Proceedings of the 6th ACM SIGCOMM workshop on Network and system support for games*, pages 37–42, New York, NY, USA, 2007. ACM.

[3] M. Esch, J. Botev, H. Schloss, and I. Scholtes. P2p-based avatar interaction in massive multiuser virtual environments. In *Proceedings of the 3th International Conference on Complex, Intelligent and Software Intensive Systems (CISIS 2009)*, 2009.

[4] S.-Y. Hu, J.-F. Chen, and T.-H. Chen. Von: a scalable peer-to-peer network for virtual environments. *IEEE Network Magazine*, 20(4):22–31, 2006.

[5] S.-Y. Hu, T.-H. Huang, S.-C. Chang, W.-L. Sung, J.-R. Jiang, and B.-Y. Chen. Flod: A framework for peer-to-peer 3d streaming. In *The 27th Conference on Computer Communications (IEEE INFOCOM '08)*, 2008.

[6] J. Keller and G. Simon. Solipsis: A massively multi-participant virtual world. In *PDPTA*, pages 262–268, 2003.

[7] H. Liang, R. D. Silva, W. T. Ooi, and M. Motani. Avatar mobility in user-created networked virtual worlds: measurements, analysis, and implications. *Multimedia Tools and Applications*, 45(1–3):163–190, 2009.

[8] J. Royan, P. Gioia, R. Cavagna, and C. Bouville. Network-based visualization of 3d landscapes and city models. *IEEE Comput. Graph. Appl.*, 27(6):70–79, 2007.

[9] E. Tanin, A. Harwood, and H. Samet. Using a distributed quadtree index in peer-to-peer networks. *The VLDB Journal*, 16(2):165–178, 2007.