# A Scale-free And Self-organized P2P Overlay For Massive Multiuser Virtual Environments

Markus Esch
*University of Luxembourg*
*Faculté des Sciences, de la Technologie et de la Communication*
*Luxembourg, Luxembourg*
*Email: markus.esch@uni.lu*

Ingo Scholtes
*University of Trier*
*Department of Computer Science*
*Trier, Germany*
*Email: scholtes@syssoft.uni-trier.de*

***Abstract*—Massive Multiuser Virtual Environments have recently grown popular, and commercial virtual online worlds like Second Life or World of Warcraft attract a lot of attention. In this context, for the research on distributed systems, especially the idea of a 3D Web as a global scale virtual environment is very interesting, since it poses severe technical challenges to the underlying infrastructure. It is generally accepted, that the realization of such a global scale scenario can not be realized in a traditional centralized fashion. For this reason in the course of the HyperVerse project we have developed a two-tier Peer-to-Peer (P2P) architecture as basic infrastructure for a federated and scalable 3D Web. Our approach relies on a concept, that incorporates a loosely-structured P2P overlay of user clients and an overlay that connects a federation of reliable server machines constituting a reliable backbone service. This paper proposes a self-organized and scale-free network overlay for the reliable backbone in the HyperVerse architecture. The overlay incorporates advantages of scale-free networks, self-organization and epidemic aggregation in order to tackle the severe challenges of the scenario in a fully distributed fashion without any central control.**

***Keywords*-MMVE, DVE, P2P Overlay, Self-organization, Complex Networks**

## I. Introduction

The popularity of Massive Multiuser Virtual Environments (MMVEs) like Internet communities (e.g. *Second Life*) or Massive Multiplayer Online Games (MMOGs) (e.g. World of Warcraft) is constantly increasing. This surge of interest influences also the research on distributed systems, since a lot of work is done in the field of distributed virtual environments. In this context, especially the idea of a *3D Web* as combination of MMVEs and today's WWW attracts a lot of attention and provides a variety of interesting opportunities. One may envision a fusion of today's Web content and avatar based interaction. A user can move with an avatar through a 3D Online world in order to meet friends, undertake a sightseeing tour, shop and so forth. Provided on a global-scale and in combination with foreseeable advances in human interface technologies, such a 3D Web allows for instance for virtual mass events and immersive interaction. While the opportunities of such a scenario sound promising, its realization on a global-scale poses severe technical challenges. The central question is how scalability and interactivity along with consistency and persistency can be reached in a way that potentially all 3D Web users are able to use the same instance of the virtual world at the same time. The commercial precursors of a 3D Web, for the most part rely on centralized client/server architectures and the parallel provision of hundreds of separated world instances, each supporting at most a few thousand concurrent participants. While this has advantages in terms of manageability and controllability for the providers, these approaches can not reach the envisioned global-scale user number of a 3D Web. It is however generally accepted that realizing a global-scale MMVE based on a traditional client/server architecture with a centralized server farm is not practicable.

In a current research project called HyperVerse, we aim at the provision of a federated infrastructure that supports such a scenario on a global-scale while retaining the decentralized nature and reliability of the WWW. In [4] we have presented the concept of a two-tier Peer-to-Peer network as an infrastructure for a global-scale MMVE. The basic idea is to combine a loosely structured Peer-to-Peer network, that interconnects the user clients for a Torrent-based data distribution with a highly-structured Peer-to-Peer overlay of reliable server machines (so-called Public Servers) in order to provide a reliable and persistent backbone service. Tasks of the federated backbone service involve avatar tracking as well as object- hosting and indexing. For their interconnection, we have developed a overlay network, allowing for the particular properties of MMVEs, like high avatar dynamics and non uniform load distribution. This overlay utilizes the expected scale-free distribution of Public Server capacities to form a network, that emerges in a self-organized fashion into a scale-free state with a power law degree distribution. By this means, the overlay benefits form the particular advantages of scale-free networks like short average path length and robustness. This paper presents the concept of this self-organized and scale-free P2P overlay network, that was especially designed for our setting of a global scale Distributed Virtual Environment (DVE).

In section II a brief introduction of the HyperVerse archi-

*Digital Object Identifier: 10.4108/ICST.COLLABORATECOM2009.8279*
*http://dx.doi.org/10.4108/ICST.COLLABORATECOM2009.8279*

tecture is presented in order to enable a better understanding of the entire concept. In the subsequent section III objectives as well as the idea of the concept is presented, before section IV describes the actual realization of the scale-free P2P overlay network in a self-organized fashion. Section V compares our work to other approaches in this field. The paper concludes with a discussion of our main contributions, open issues and future work in section VI.

## II. The HyperVerse Infrastructure

In this section we give a brief introduction to the HyperVerse project. Figure 1 presents a schematic overview of the architecture. Our concept distinguishes between reliable peers, so-called Public Servers, and relatively unreliable user clients. Public Servers are machines that resemble today's Web Servers and they are not required to be under control of any centralized authority. Similar to today's Web for their provision we rely on the incentive of being able to publish information in the HyperVerse. The federation of these Public Servers constitutes the backbone of the HyperVerse and provides the huge amount of data in an efficient and - most important - reliable manner to clients. This is required since the huge amount of world data in our scenario is highly dynamic, due to fact that world objects can be added, modified and removed at any time by users or content providers. For this reason a predistribution of world data, like it is done in today's MMOGs, is not feasible.

Due to the fact that the clients' bandwidth is constantly increasing, it is mandatory to engage the clients in the data distribution. Since it is foreseeable that the clients exhibit high churn rates we propose a loosely structured peer-to-peer overlay to interconnect them. Our approach is based on a scheme similar to the BitTorrent protocol [6]. Each client makes cached data accessible in a Torrent-like manner, i.e. data are split into individually addressable pieces. By this means it is possible to download object and world data in parallel from a set of clients. In order to figure out which clients hold data of interest we can utilize the fact that clients in virtual proximity need to possess a similar set of data, since they have to render more or less the same objects. Hence a client can always download the required data from other users in its virtual proximity once client density is high enough. Taking advantage of the supposed, primarily continuous movement through the virtual world we can enhance this mechanism by applying prefetching and caching strategies. A detailed description of the Torrent-based data distribution mechanism in HyperVerse is presented in [4] and [19]. One main advantage of this Torrent mechanism is that it implicitly tackles the problem of flash crowds. Due to the rise of available bandwidth for data distribution with increasing user numbers in a given region, the flash crowd problem can be handled in a self-organizing manner.

Each Torrent system relies on a tracker service that enables the clients to find the required data pieces. In HyperVerse the Torrent tracking can be realized as piggyback mechanism to the avatar interaction. Because clients in virtual proximity have with a high probability the same object data and these clients anyway need to be informed about each other in order to allow mutual rendering and interaction. In our setting the backbone service is responsible for the interconnection of clients in proximity. For this the clients have to send periodical position updates to the backbone. In order to keep the update frequency low we utilize a scheme that differentiates a client's Field of View (FoV) and its Area of Interest (AoI), which is also described in detail in [4]. Receiving frequent movement updates of the clients, the backbone service is able to interconnect two clients when their AoIs intersect.

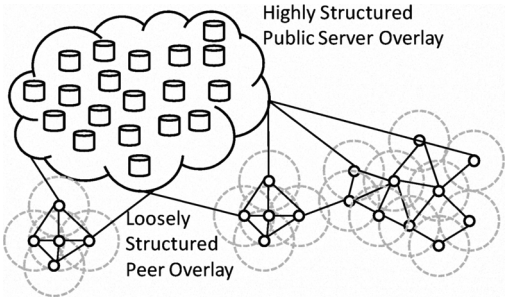


Figure 1. Overview Of The Two-Tier HyperVerse Infrastructure

In addition to the avatar tracking further tasks of the backbone service involve object hosting and object indexing. As mentioned, world objects are distributed in a Torrent-based fashion whenever possible. However in order to provide a persistent and reliable world and to realize the initial distribution of the object data, the backbone hosts all world objects and acts as initial seed for these objects. Hereby an object is hosted by its publisher. For this reason someone who wants to publish an object in the HyperVerse has to run or have access to a Public Server that hosts this object and constitutes a part of the backbone overlay. This is similar to today's WWW where publishing information at least requires access to some Web server. To enable a client to load the objects in its proximity the HyperVerse backbone works as indexing service mapping an object's position to the Public Server that hosts this object. Summing up, a Public Server has two independent tasks, at first hosting of world objects and second being a part of the backbone overlay that is responsible for avatar tracking and object indexing. An avatar entering a certain region first obtains a list of avatars and objects in proximity from the backbone and thereupon starts downloading the object data from the providing Public Server or, if possible, in Torrent fashion. For this purpose the whole surface of the world needs to be distributed among all Public Servers, each managing a certain area of the world. Moreover the Public Servers need to be interconnected by any scheme in order to allow

Digital Object Identifier: 10.4108/ICST.COLLABORATECOM2009.8279
http://dx.doi.org/10.4108/ICST.COLLABORATECOM2009.8279

efficient routing as well as node and object lookups on the backbone overlay. For this purposes, of managing the federation of Public Servers constituting the HyperVerse backbone, the scale-free overlay described in the subsequent sections has been developed.

## III. Objectives and Basic Concept

This section presents our concept for a dynamic backbone overlay utilized to interconnect the Public Servers of the HyperVerse infrastructure presented in the previous section. This overlay forms the second tier of the architecture and is responsible for the reliable and persistent hosting of the virtual environment.

Our concepts relies on two basic assumptions about the properties of a virtual online world:

- **Unequally distributed avatars and objects:** It can not be expected that objects and avatars in a virtual world are uniformly distributed in the world. Rather hotspots with a high object and avatar density are likely to emerge. While the object distribution is relatively static and is expected to change slowly, the avatar distribution is very dynamic and can change fast and unexpectedly.
- **Varying Public Server capacities:** It can not be assumed that all Public Servers of the overlay have the same capacity in terms of bandwidth and computational power. For this reason, each Public Server is able to host a different fraction of the world. Like in today's Web, the capacity basically depends on the popularity of the hosted content. Today, the providers of Web content need to make sure that the servers, hosting a Website, are able to handle the incoming traffic. For this reason, very popular and frequently used Websites, for example Google, are hosted by powerful server farms, while small private Websites can be hosted by less powerful single machines. Translating the principle, that the content provider needs to provide sufficient server capacity, to the HyperVerse scenario, the object providers have to care for the required server capacity. Often visited and used objects in the world need to be hosted by more powerful servers than less popular objects.

We argue, that any feasible concept for the backbone infrastructure of the HyperVerse scenario, needs to consider these two basic assumptions. On the one hand, problems like high avatar dynamics need to be tackled. On the other hand, properties of the environment have to be considered in order to utilize the full potential of the available resources. A scheme, equally and statically distributing the virtual world among the Public Servers has little prospect for success, because of the avatar dynamics, the unequal avatar distribution and the inhomogeneous servers. Our concept for the backbone infrastructure considers these points by enabling a dynamic reorganization of servers based on the current avatar and object distribution while distributing the load according to the server's capacities.

Our concept relies on the advantages of scale-free networks, like resilience against node failures and small diameter, for this reason section III-A gives a brief introduction to this topic. The subsequent section III-B describes the basic idea of our scale-free backbone overlay.

### A. Scale Free Networks

Studying large and complex networks, describing the exact state of each vertex in the network implies huge efforts, due to the large number of nodes and their dynamics. Fortunately, it has been shown, that statistical information and macroscopic properties of the entire network can also be used to characterize such complex systems. Since not all nodes in a network have the same degree, the node degree distribution is one important property characterizing a complex network. Typically large scale networks have been described by the random graph theory of Erdös and Renyi. While for these Erdös/Renyi graphs the degree distribution is a Poisson distribution and the degree of all nodes is close the average degree, in [2] it has been shown that in reality the degree distribution of large complex networks emerges to a power law distribution. The probability $P(k)$ that a random node has exactly $k$ edges is given by: $P(k) \sim k^{\gamma}$. If the power law exponent $\gamma$ is in a certain range (typically $2 < \gamma < 3$), such networks are called scale-free networks. For many large networks it has been shown that they exhibit a scale-free degree distribution. Be it constructed networks like the World Wide Web [1] and the Internet [9], social networks like the graph of scientific collaborations [16] or cellular networks like the metabolic network [14].

The question, which fundamental mechanism leads to emergence of scale-freedom in large networks has first been answered by the *Barábasi/Albert model* [3]. This model mimics the two basic mechanisms responsible for the emergence of scale-freedom: *growth* and *preferential attachment*. The basic idea is to constantly add new nodes to the network and connect these nodes to existing nodes in the network with a probability equivalent to the existing nodes' degree. This means a new node is connected to a node with high degree with a higher probability, resembling the principle *the rich get richer.*

It has been shown that scale-free networks generated by the Barábasi/Albert model have a lot of advantageous properties in terms of reliability as well as performance. Important for the performance of a network is the distance between nodes in the network, since a small path length allows fast routing. In [7] it has been proven that scale-free networks with power law exponents $\gamma \in (2, 3)$, exhibit an ultra-small network diameter $d \propto ln(ln(N))$. Thus the average path length grows very slow with increasing network size, allowing fast routing even in huge networks. The resilience against random node failures of a network

Digital Object Identifier: 10.4108/ICST.COLLABORATECOM2009.8279
http://dx.doi.org/10.4108/ICST.COLLABORATECOM2009.8279

is determined by the tolerance against node failures. Here two different scenarios need to be distinguished. Random node failures and targeted attacks against the network. It has been shown that scale-free networks are very robust against random node failures, since the probability that a high degree node fails is extremely low. But, since attacks are targeted on the most important nodes, scale-free networks are less robust against selective attacks compared to Erdös/Renyi random networks.

### B. Concept

This section presents our concept for a scale-free P2P overlay incorporating advantages of scale-free networks, epidemic information aggregation and self-organization. The overlay has been built considering the above mentioned assumptions of unequal load distribution and different server capacities. The basic idea is to build an overlay, that self-organized emerges to a scale-free state in order to benefit from the particular advantages of this network type.

The task of the backbone is indexing of world objects, as well as interconnecting clients in virtual proximity in order to allow interaction and mutual rendering. For this purpose the world surface gets subdivided into small cells, each cell managed by one Public Server. For subdividing the plane we are using a Voronoi diagram. To define a Voronoi diagram distributing the plane, a virtual position on the world surface is assigned to all Public Servers. In the resulting Voronoi decomposition each Public Server manages the Voronoi cell surrounding its virtual position. To allow for the assumptions about the particular properties of an MMVE, servers handle regions with a object and avatar density equivalent to the own capacities. By this means hotspots get managed by machines with high capacities, while less crowded areas can be handled by less powerful machines. In order to be able to adapt the overlay to dynamic load shifts in the online world, the virtual positions are not fixed. Rather, the servers modify their position according to the load induced by the users of the online world. The virtual positions are managed in a self-organized fashion, based on a set of rules applied by each server as described in section IV-A.

In addition to the distribution of the world surface, it is an important issue how the Public Servers are linked together. In first place all nodes have links to the nodes managing the bordering Voronoi cells. Additionally links to other nodes in the network are established in order to get a scale-free degree distribution and to allow fast routing to all positions in the virtual world. To get a network, that emerges into a scale-free state, links are established by a preferential attachment scheme, based on bandwidth and computational power. This mechanism is described in section IV-B.

For the management of both, the virtual positions and the link structure, the servers require certain aggregated information like e.g. load distribution and network size. But, a global system view is not existing, since the system is totally distributed and self-organized. For this reason we are using epidemic aggregation to enable peers to obtain certain information about the network state. The aggregation mechanism is explained in section IV-C.

## IV. Realization Of The Scale-free P2P Overlay

This section describes how the concept presented in the previous section can be realized. The self-organized plane distribution based on the application of a set of rules is explained in section IV-A. Section IV-B explains how links between nodes can be established in a distributed fashion, so that a scale-free node degree distribution emerges. The epidemic aggregation mechanisms used to gather certain information required for the algorithm are presented in section IV-C.

### A. Plane Distribution

As already mentioned virtual positions of Public Servers need to be assigned in a way that crowded regions are managed by machines with high bandwidth and computational power, while less powerful machines are responsible for less crowded regions. Public Server apply several simple rules to adjust their virtual position to the current load distribution in a way that ensures this requirement. We distinguish so-called *local* rules and *global* rules. The local rules require information about the own state and the state of the bordering nodes and have only local effect. Their purpose is to adapt the local world distribution to local load variances. The global rules are applied to equalize global imbalances in the load distribution detected based on aggregated information.

*1) Local Rules:* Using the local rules each Public Server balances the load distribution between itself and its bordering neighbors. For the application of subsequent rules a *mass* is assigned to each object or avatar, its mass reflecting the load it induces on the server backbone. As the backbone's purpose is object and avatar tracking and indexing rather than actual data hosting, an object's mass is independent of the transmission size and depends on its velocity. This is because faster moving items induce a higher load, due to the required position tracking. The mass $m$ of an entity $x$ is given by:

$$m(x) = m_0 + v(x) \tag{1}$$

With $m_0$ being a constant rest mass equal for all entities, and $v(x)$ being an entities velocity. Using these masses, a Public Server can calculate the absolute mass of the objects in its cell as well as the center of mass. While the absolute mass of a cell is just the sum of the masses, the center of mass $C$ of a cell is calculated by the following formula:

$$C = \frac{\sum_{i=1}^{n} p_i \cdot m(i)}{\sum_{i=1}^{n} m(i)} \tag{2}$$

Digital Object Identifier: 10.4108/ICST.COLLABORATECOM2009.8279
http://dx.doi.org/10.4108/ICST.COLLABORATECOM2009.8279

With $p_1, ..., p_n$ being the positions of entities $1, ...i$ in the cell. Depending on its bandwidth and computational power, a maximum payload $p$ (indicating the mass it can handle) is assigned to each Public Server. Based on these information, Public Servers adjust their position according to the following local rules:

I) **Centering:** Subdividing the world surface into cells managed by different Public Server implies traffic on the backbone overlay if peers cross cell borders. For this reason our goal is to minimize the number of cell crossing. Hence the Public Servers adjust their virtual positions so that their position is close to the cell's center of mass.

II) **Keeping The Mass-Order:** As already mentioned the load of a server should correspond to its payload. In order to assure this in the vicinity of each node, the Public Servers modify their positions accordingly. For this purpose each node monitors the mass of its own as well as bordering cells and compares these masses to the maximum payloads. If one of the neighbors has a higher payload but a lower cell mass, or vice-versa, this imbalance needs to be adjusted. For this purpose the node with the lower payload moves away from the node with the higher payload, in order to increase the cell size and with it the cell mass of the node with higher payload. This is depicted in figure 2. In this example node $A$ has a lower payload but a higher mass than node $B$. For this reason, node $A$ moves away from node $B$, to keep the mass order. The direction of $A$'s movement, is given by the normalized vector $\vec{BA}$ and the speed is defined by the cell mass difference between $A$ and $B$. If a node has to move away from several neighbors at the same time, the direction of movement is given by the sum of the movement vectors. This is illustrated in figure 3. In this example node $A$ moves away from $B$ and $F$. The resulting motion vector $\vec{v}$ is the sum of the vector $\vec{b}$ and $\vec{f}$, being the vectors pointing away from $B$ and $F$ respectively.

III) **Unload Neighbors:** If a Public Server detects that one of its neighbors is overburdened (i.e. its cell mass being larger than the maximum payload), this Public Server moves towards the overburdened node in order to unload it. This is depicted in figure 4. In this example node $B$ is overburdened and for this reason node $A$ moves towards $B$. As above the direction is given by the normalized vector $\vec{AB}$ between the nodes and the speed is defined by the amount of $B$'s overload. If several neighbors of a given node are overburdened the resulting movement vector is again given by the sum of all vectors.
Since an overburdened Public Server negatively affects the operativeness of the backbone service, neighbors have to move towards nodes running the risk of getting overburdened before they are actually overburdened. It is future work to find out what is a good tradeoff for the reserve capacity a Public Server keeps, so that the utilization of the available resources is as good as possible while the danger of overburdening a server is as low as possible.
As also depicted in figure 4 the motion of one node towards an overburdened neighbor may result in pulling other nodes, like node $E$ (also $D$ and $F$) in the example. $E$ is a neighbor of $A$. Since $A$ moves towards $B$ the load for $A$ may rise, resulting in the risk of overburdening $A$. This causes $E$ to follow $A$. By this means server capacities are automatically concentrated around regions with a high mass.

IV) **Swapping:** It makes no sense to shrink a Public Server's cell by any order, since this would imply an extraordinary overhead by avatars that constantly cross cell borders. For this reason we define a minimum cell size. If this size is reached neighbors can not move closer to a node. Thus in scenarios with very high density hotspots, the mass in a certain cell may become too high for the managing Public Server, while neighbors can not move closer to this node in order to unload it. To handle this issue, not only the absolute mass of cells (as done by the local rule $II$), but also the positions of high density hotspots need to be considered. For this, Public Servers detect the hotspot within their own cell, a hotspot being defined as the position in the cell with the highest mass within a certain radius around this spot. This radius can be adjustable and further evaluations are required in order to investigate appropriate values. Based on the hotspot masses, the Public Servers are organized in a way, that server payloads correspond to the hotspot masses within the managed cell. For this purpose each Public Server compares the own hotspot mass with the hotspot masses of its bordering neighbors. If one of the neighbors has a heavier hotspot but a smaller payload, Public Servers swap their cells. For this operation, it is required, that the server with lower payload is able to handle the current cell mass of the other server. If this is not the case, the cell size of the server with higher load first needs to be downsized in order to reduce the mass. An example for the application of the swapping rules is shown in figure 5. In order to avoid a permanent swapping of cells, an appropriate threshold is required that prevents thrashing and at the same time guarantees that the nodes manage hotspots corresponding to their payload. A study of different threshold values is ongoing research.
Since all nodes in the concerned cells need to change their managing Public Server, swapping is a comparatively expensive operation. We argue that this can be

*Digital Object Identifier: 10.4108/ICST.COLLABORATECOM2009.8279*
*http://dx.doi.org/10.4108/ICST.COLLABORATECOM2009.8279*

justified by two reasons: At first this rule essentially contributes to the stability of the overlay since heavy hotspots are always managed by the most powerful machines available. Second, it can be expected that swapping will be necessary quite infrequently, due to the threshold and the relative stability of hotspots. That means, popular and crowded regions are expected to remain hotspots for quite a while and due to the rule *Centering*, a hotspot once managed by a powerful server remains in this cell.

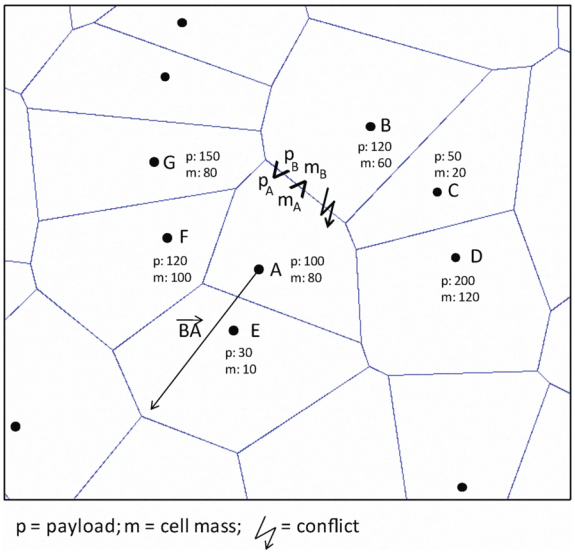


Figure 2. Applying The Local Rule *Keeping The Mass-Order*

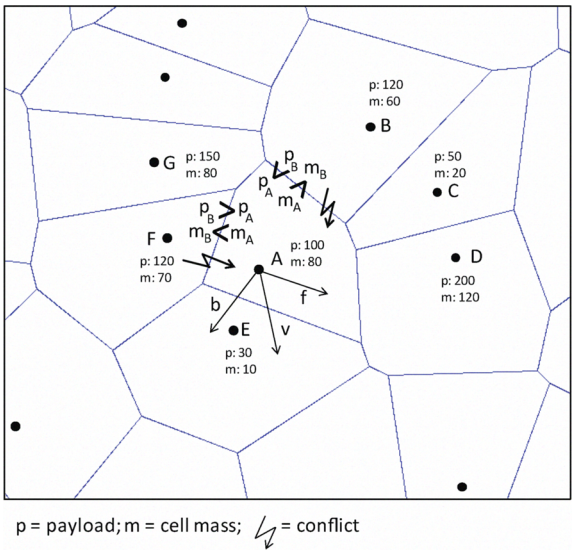


Figure 3. Applying The Local Rule *Keeping The Mass-Order* For Several Neighbors

It is likely that the rules $I$, $II$ and $III$ need to be applied at the same time. For this reason the absolute motion vector

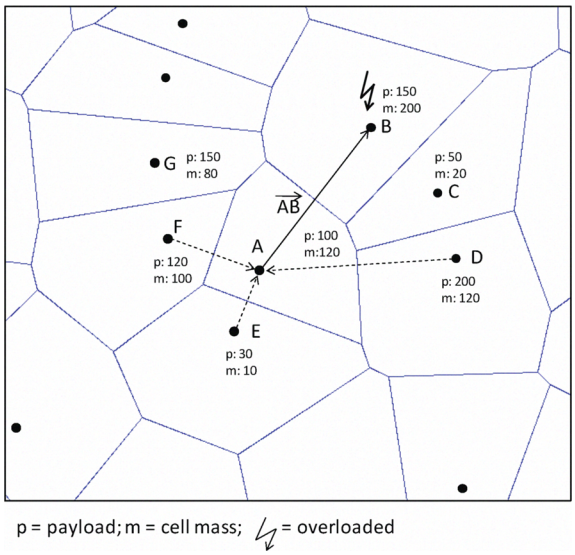


Figure 4. Applying The Local Rule *Unload Neighbors*

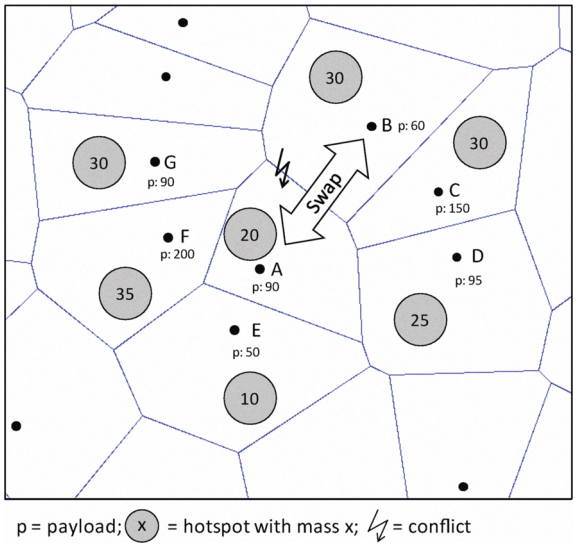


Figure 5. Applying The Local Rule *Swapping*

$\vec{m}$ of a Public Server can be calculated by the following formula:

$$\vec{m} = \vec{PC} + \sum_{i=1}^{n} \frac{\vec{L_iP}}{|\vec{L_iP}|} \cdot l_i + \sum_{i=1}^{m} \frac{\vec{PO}i}{|\vec{PO}_i|} \cdot o_i \qquad (3)$$

With $P$ being the current position, $L_1, ..., L_n$ being the nodes with reversed mass order, $l_i$ being the speed away from $L_i$, $O_1, ..., O_n$ being the overburdened neighbors and $o_i$ being the speed towards $O_i$.

For the adjustment of the algorithm it may be useful to weight a certain rule more than others. For this reason we introduce weights $w_1$, $w_2$ and $w_3$ that can take values between 0 and 1. Using the weights the motion vector is

*Digital Object Identifier: 10.4108/ICST.COLLABORATECOM2009.8279*
*http://dx.doi.org/10.4108/ICST.COLLABORATECOM2009.8279*

given by:

$$\vec{m} = w_1 \cdot \vec{PC} + w_2 \cdot \sum_{i=1}^{n} \frac{\vec{L_i P}}{|\vec{L_i P}|} \cdot l_i + w_3 \cdot \sum_{i=1}^{m} \frac{\vec{PO}i}{|\vec{PO}_i|} \cdot o_i \quad (4)$$

To avoid constantly moving and oscillating Public Servers it may be necessary to introduce a notion of inertia damping the motion of the Public Servers. Finding an appropriate damping function that on the one hand guarantees a good load distribution and on the other hand keeps the dynamic in the backbone overlay as low as possible needs to be considered future work.

*2) Global Rules:* The local rules described above manage the load distribution based on local information in the neighborhood of each node. By this means, a collective behavior, distributing the load according to the server's payload, emerges in a self-organized fashion without any central control.

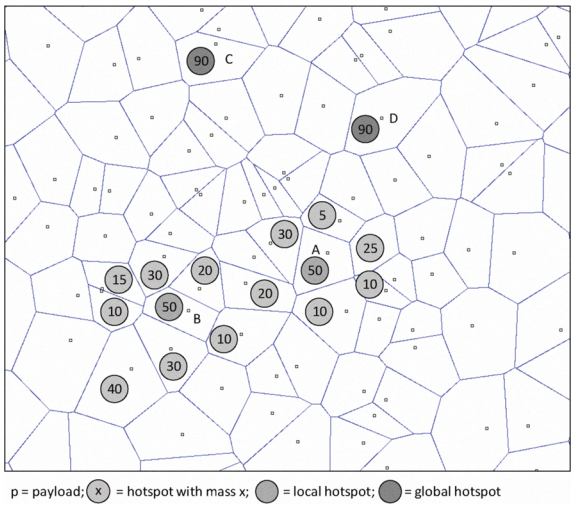


Figure 6. Emergence Of Local Maximums By The Application Of Local Rules Only

In addition to the local rules, some global rules have been defined. These are required to balance global inequalities in the load distribution, that can not be detected and solved solely based on local knowledge and rules. Because, situations can occur where all nodes remain stable and balanced around local load maximums. Since they only have local knowledge, they don't know about any global load maximums bigger than the known local maximum. Such a situation is depicted in figure 6. In this example the nodes $A$ and $B$ remain at their hotspot, being a local maximum, and the condition is fulfilled in the surrounding of $A$ and $B$. But since $A$ and $B$ have only local knowledge, they can not become aware of the global maximums at $C$ and $D$. For this reason, to overcome the border of the local maximums some global rules utilizing aggregated information need to be used. In order to apply the global rules, the following global information needs to be aggregated using gossip-based aggregation as described in IV-C:

- **Density Distribution:** The Public Servers require information about the distribution of the avatar and object density in order to be able to adjust their positions accordingly. It is obviously not possible to determine the position of all objects and avatars in order to get an exact map of the density distribution. Because this would on the one hand imply a unjustifiable overhead, and on the other hand it is likely to be impossible due to the avatar dynamics. For this reason only information about the biggest hotspots is required. However, in a growing and scalable environment it is not possible to define a constant number of hotspots for which information is aggregated. For this reason, a fixed preview radius $r$ is defined, and all nodes aggregate only information about this preview area. Since this area has a fixed size, we can define a constant value $n$, and aggregate the $n$ most crowded hotspots within this area. For the hotspot aggregation we are using the same definition of a hotspot as applied in the local rule *Swapping*.
- **Payload:** Along with information about the hotspots, the maximum payload of the nodes currently managing the Voronoi cell surrounding the hot spots is aggregated.

Using these information, the following global rules are used by the Public Servers:

I) **Jumping:** Based on the aggregated hotspot information a Public Server can detect if there is a position within its preview radius suiting better to its payload. For example consider figure 7, here node $A$ becomes aware of the hotspot $h_1$. Since $A$ has a much high payload than required at its current position, it sets the hotspot $h_1$ as new position. Node $B$ is thereupon pushed aside by the application of the local rules. The nodes previously bordering $A$ than have to manage its old cell and may be moved towards $A$'s old position by local rules in order to handle the load. Here again a threshold needs to be defined in order to avoid constantly jumping servers. By application of this rule the most powerful server machines concentrate around the heaviest hotspots and the payload of the Public Servers decreases with increasing distance from hotspots. After jumping to another position, the Public Servers have a new preview radius. Hence, the jumping may recursively proceed until the servers have found their global optimum.

II) **Active Search:** If a Public Server with a high excess payload can not find an appropriate hotspot within its own preview area, it can start a active search for a hotspot matching its payload. For this purpose the server sends requests to other world regions outside

*Digital Object Identifier: 10.4108/ICST.COLLABORATECOM2009.8279*
*http://dx.doi.org/10.4108/ICST.COLLABORATECOM2009.8279*

the own preview radius to find heavier hotspots. Due to the scale-free link structure of the network and the associated small path length (see section III-A), this hotspot search can be performed very efficiently. If an appropriate hotspot with a managing server having a lower payload is found, a swapping is performed. A threshold has to ensure that always the rule *Jumping* is applied first before executing this rule. Only, if no hotspot using the server to full capacity can be found the active search is started.

III) **Active Pull:** If a Public Server managing a hotspot region is running into the risk of getting overburdened and the minimum cell size is already reached and no server with higher payload is available within the own preview area, it can actively pull more powerful machines. Like for the *Active Search* rule, a search outside of the own preview area is started and if a Public Server with higher payload, managing currently a less crowded region, is found these two servers can swap their cells.

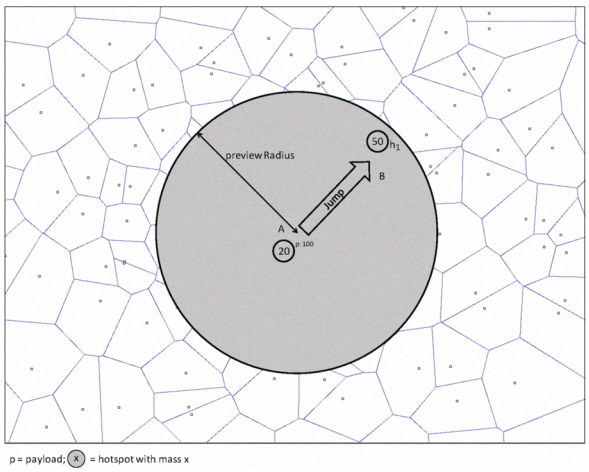


Figure 7. Applying The Global Rule *Jumping*

These global rules enable that the condition, that Public Servers remain at positions with a load corresponding to their payload, can be achieved globally. Because applying the local rules only, could lead to a situations where the condition is fulfilled locally in the surrounding of each node, but not globally, since information about the density in other regions is missing. Applying the global rules in the example shown in figure 6, the nodes $A$ and $B$ can get aware of the global hotspots and change their position if necessary.

### B. Scale-free link structure

As mentioned above the aim is to construct a overlay network with a scale-free link structure, in order to benefit from the advantages of scale-free networks in terms of reliability, resilience and short path length. In order to reach this, our concept proposes two basic steps. At first, establishing links to newly added nodes by a preferential attachment scheme similar to the one proposed in [2]. Second, monitoring the power-law exponent in the network and adjusting it by using local rules, as described in [20].

The algorithm presented in [2], specifies two ingredients leading to the emergence of a scale-free networks: *growth* and *preferential attachment*. The preferential attachment is realized by connecting a new node with probability $\Pi$ to an existing node $i$, depending on the degree $k_i$ of $i$:

$$\Pi(k_i) = \frac{k_i}{\sum_j k_j} \tag{5}$$

With $k_i$ being the degree of node $i$. This means, new nodes are connected to existing nodes with a high node degree with a higher probability.

Due to the Web like scenario in which new information provider appear one after another the prerequisite growth is presumably fulfilled automatically. In order to adopt also the preferential attachment scheme to our scenario we have to resemble the way preferential attachment evolves in real networks like the WWW. In the WWW one can assert, that the number of links to popular Web-sites increases faster due to their high profile. At the same time, the providers take care for allocating sufficient server capacities. That means, the scale-free link structure and the existence of hubs with sufficient capacities emerges in a self-organizing manner from the different popularity of Web sites. Transferring this observation to our scenario of a virtual online environment, it means, that Public Servers hosting very popular world objects automatically exhibit higher capacities, since this is automatically ensured by the object providers. Since the scale-free capacity distribution thus emerges automatically, we just have to construct the link structure accordingly. For this reason, we establish links to Public Servers with high capacities, with a higher probability. Hence a new node is connected to an existing node $i$ with probability $\Pi$ depending on the payload $p_i$ of $i$:

$$\Pi(p_i) = \frac{p_i}{\sum_j p_j} \tag{6}$$

Now the question is, how such a link distribution can be realized in our fully distributed scenario without global knowledge about the capacity of all nodes. For this, we need to aggregate the average node payload as well as the number of nodes in the network. Based on this information, a joining node can establish its links to other nodes in the network.

If a new node joins the network, the following steps, have to be taken:

- A new node $v$ first has to connect to a node $i$ already part of the overlay. This initial node redirects the new node to a random node $r$ in the network, by choosing a random position in the world and forwarding the new node to the Public Server hosting the surrounding Voronoi cell.

*Digital Object Identifier: 10.4108/ICST.COLLABORATECOM2009.8279*
*http://dx.doi.org/10.4108/ICST.COLLABORATECOM2009.8279*

- Based on the information of $r$ about the density distribution in its preview area, the appropriate initial position $x$ for $v$ in this preview area is identified. Thereupon $v$ is inserted at this position and gets connected to all bordering node $b_1, ..., b_n$.
- Utilizing the neighbors $l_1, ..., l_k$ of $n's$ bordering nodes, it is possible to establish $m$ ($m$ is a fixed value, independent of the network size) links to other nodes in the network. In order to avoid that $v$ has the same links as its bordering neighbors, the neighbors of $l_1, ..., l_k$ are used as potential links. We denote this set of potential links with $s$. Thereupon $v$ establishes links to the nodes in $s$ with a probability corresponding to equation 6, until the required number of $m$ links is reached. The sum of the node capacities in equation 6 can be estimated by using the aggregated information about the network size and the average density. The basic idea of the algorithm is that each node maintains a list of the $n$ heaviest hotspots within its preview area. In periodical time intervals the node selects a random neighbor and exchanges the hotspot list with this neighbor. Both nodes thereupon update their local lists by merging it with the list received from the neighbor. It has been shown, that using this algorithm it is possible to aggregate global information very fast after only few intervals. Moreover the overhead of the algorithm is constant since each node per time interval communicates with just one neighbor.

Using this algorithm a power-law network is constructed based on the preferential attachment paradigm of Barábasi and Albert. Once having established a power-law network, the algorithm presented in [20] can be used to adapt the properties of the network by influencing the critical power law exponent. This algorithm uses a gossip scheme to make nodes aware of the power law exponent of the network, and applies reconnection rules to adapt the exponent. This is important, since the power law exponent has crucial influence on the topological properties of the network.

Routing and node lookup in our overlay can most efficiently be realized using geographic routing. This way the virtual geography existing in our MMVE scenario is in some sense utilized as scheme for identifying nodes. Since all entities in the virtual world, like objects, avatars and Public Servers have virtual positions, routes can always be targeted to the virtual position of an entity. Applying well studied geographic routing algorithms [10], known mainly from the field of mobile ad-hoc networks, routing can be performed in an efficient and robust manner. Moreover, due to the short average path length of power-law networks, routing on this overlay can expected to be very efficient.

### C. Information Aggregation

For inserting a new node into the network as well as for the application of the global rules described in the previous sections, information about the network state needs to be aggregated by the nodes. It has been shown, that epidemic aggregation is very useful for the fast and efficient aggregation of information in large, highly dynamic networks. Our epidemic aggregation scheme is similar to the one described in [13]. We are using a push-pull anti-entropy [8] protocol, because this scheme has properties advantageous for our scenario. It is extremely reliable even under high dynamics and the overhead for the aggregation is constant.

Summing up the requirements from the previous sections, the following information about the network state needs to be aggregated:

- Hotspots within a certain preview radius, along with the payload of the nodes hosting the hotspots.
- Average server payload.
- Network size.

In order to retrieve an estimation for the network size and the average server payload, the algorithm described in [13] can be utilized without any changes. For the aggregation of the hotspots within the preview area, this basic algorithm needs to be slightly adjusted. These adjustments have been presented in [19]. Basically the algorithm is adopted with respect to the avatar dynamics and the fact that nodes require only information from a certain preview area.

## V. 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 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 [15], 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 [12] provides a framework for pure P2P-based 3D scene streaming that is build upon VON [11], 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 [17], an architecture similar to FLoD is described that uses super peers, so-called *connectivity peers*, to interconnect the peers. *VastPark* [21] 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 [5] architecture provides a central tracker service that

Digital Object Identifier: 10.4108/ICST.COLLABORATECOM2009.8279
http://dx.doi.org/10.4108/ICST.COLLABORATECOM2009.8279

can be realized as a single server or as a Distributed Hash Table (DHT).

## VI. Conclusion and Future Work

This paper presented the concept for a self-organized and scale-free backbone overlay for the HyperVerse infrastructure. The overlay allows for non-uniform object and avatar distribution as well as for the high avatar dynamics expected in an MMVE scenario. The world surface is distributed among the Public Servers by a Voronoi diagram, each server managing one Voronoi cell. Thereby the load is distributed in a fashion that the load of a server depends on its payload. This is reached by assigning hotspot regions to powerful machines and less crowded regions to less powerful servers, using a self-organized scheme that defines a set of rules applied by each node. In order to establish a scale-free link structure among the Public Servers the scale-free distribution of server capacities, that automatically emerges in our scenario, is utilized. This way we benefit from the advantages of scale-free networks in terms of resilience against random node failures and short path length.

The presented concept defines a number of adjustable threshold values influencing the overall characteristics of the network. It is future work to study the behavior of the network using different threshold values, in order to find appropriate values for different scenarios and requirements. Moreover performance measurements in terms of scalability, routing performance as well as churn resilience need to be performed. In order to enable these studies, we are currently implementing the proposed overlay network for evaluations using the complex network generator and simulator TopGen [18].

## References


[1] R. Albert, H. Jeong, and A. L. Barabasi. Internet: Diameter of the world-wide web. *Nature*, 401(6749):130–131, September 1999.

[2] A.-L. Barabasi and R. Albert. Emergence of scaling in random networks. *Science*, 286:509, 1999.

[3] A.-L. Barabási, R. Albert, and H. Jeong. Mean-field theory for scale-free random networks, July 1999.

[4] J. Botev, M. Esch, A. Höhfeld, H. Schloss, and I. Scholtes. The hyperverse - concepts for a federated and torrent-based "3d web". The 1st International Workshop on Massively Multiuser Virtual Environments (MMVE), 2008.

[5] 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.

[6] B. Cohen. Incentives build robustness in bittorrent, 2003. citeseer.ist.psu.edu/cohen03incentives.html.

[7] R. Cohen and S. Havlin. Scale-free networks are ultrasmall. *PHYS.REV.LETT*, 90:058701, 2003.

[8] A. Demers, D. Greene, C. Hauser, W. Irish, J. Larson, S. Shenker, H. Sturgis, D. Swinehart, and D. Terry. Epidemic algorithms for replicated database maintenance. In *PODC '87: Proceedings of the sixth annual ACM Symposium on Principles of distributed computing*, pages 1–12, New York, NY, USA, 1987. ACM.

[9] M. Faloutsos, P. Faloutsos, and C. Faloutsos. On power-law relationships of the internet topology. In *SIGCOMM '99: Proceedings of the conference on Applications, technologies, architectures, and protocols for computer communication*, pages 251–262, New York, NY, USA, 1999. ACM.

[10] S. Giordano, I. Stojmenovic, and L. Blazevic. Position based routing algorithms for ad hoc networks: A taxonomy. In *Ad Hoc Wireless Networking*, pages 103–136. Kluwer, 2001.

[11] 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.

[12] 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.

[13] M. Jelasity and A. Montresor. Epidemic-style proactive aggregation in large overlay networks. In *ICDCS '04: Proceedings of the 24th International Conference on Distributed Computing Systems (ICDCS'04)*, pages 102–109, Washington, DC, USA, 2004. IEEE Computer Society.

[14] H. Jeong, B. Tombor, R. Albert, Z. N. Oltvai, and A. L. Barabasi. The large-scale organization of metabolic networks. *Nature*, 407(6804):651–654, October 2000.

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

[16] M. E. J. Newman. The structure of scientific collaboration networks. *PROC.NATL.ACAD.SCI.USA*, 98:404, 2001.

[17] 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.

[18] I. Scholtes, J. Botev, M. Esch, A. Höhfeld, H. Schloss, and B. Zech. Topgen - internet router-level topology generation based on technology constraints. In *SimuTools*, page 30, 2008.

[19] I. Scholtes, J. Botev, M. Esch, and P. Sturm. Minimizing load delays in distributed virtual enviornments using epidemic hoarding. In *Proceedings of the 4th International Conference on Collaborative Computing Networking, Applications and Worksharing (CollaborateCom)*, Orlando, FL, USA, 2008.

[20] I. Scholtes, J. Botev, A. Höhfeld, H. Schloss, and M. Esch. Awareness-driven phase transitions in very large scale distributed systems. In *SASO '08: Proceedings of the 2008 Second IEEE International Conference on Self-Adaptive and Self-Organizing Systems*, pages 25–34, Washington, DC, USA, 2008. IEEE Computer Society.

[21] 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.



*Digital Object Identifier: 10.4108/ICST.COLLABORATECOM2009.8279*
*http://dx.doi.org/10.4108/ICST.COLLABORATECOM2009.8279*