Towards Decentralized Networlds

Current Developments in the P2Life Project

Hauke Coltzau
Department of Communication Networks
FernUniversität in Hagen
Hagen, Germany
hauke.coltzau@fernuni-hagen.de

***Abstract*—Decentralized networlds can serve as platforms both for interaction between users and for content and service delivery. They may well be the next evolutionary step of the WWW. To gain acceptance of users and businesses, content and service providers must be able to keep their independence as in today's WWW and still maintain the impression of a coherent unlimited virtual world. Existing approaches emerged from the area of networked virtual environments have so far failed to fulfill these basic requirements. In this article, we present an architecture to build unlimited networlds, in which interaction between user and content provider is handled directly. This allows among others for real money trading as in the WWW. Additionally, we discuss our approach to build and maintain dynamic maps of the virtual world without need for centralized instances or any other form of global knowledge.**

Keywords-networlds; networked virtual environments.

I. Introduction

In the late 1990s, the question came to focus, how users would experience the upcoming WWW in their everyday life. Among others, the dutch architect Rem Koolhaas observed hopes, expectations and social changes through the integration of the WWW as follows:

> What about the widely heralded hope that cyberspace will be the new street, the piazza in our sprawling cities of connected isolation; that *cyberflâneurs* will promenade around the Net like Baudelaire taking a stroll around Paris? (see [1])

In the context of this article, it is not so much the question whether these changes actually have come into effect today, but instead, it is the term *cyberflâneur* that shall be of interest. According to Koolhaas (and others unmentioned here), the vision existed that users would be strolling the WWW from site to site, without the need for a specific goal to go to. The early WWW supported this kind of browsing paradigm, since no centralized (and, therefore, dominant) instances as, e.g., search engines existed. The users were "forced" to stroll, and they could do so anonymously.

Fifteen years later, the publicist Evgeny Morozov takes on the idea of cyberflânerie – and declares it dead [2]. He argues that almost all browsing activity is channelled through a few central nodes, especially search engines. Moreover, in his opinion, the anonymity of the early WWW got lost due to the omnipresence of social networks, in which everyone shares everything with everybody else.

Koolhaas and Morozov do not provide scientific foundation for their arguments, but rather illustrate personal experiences and implications. Nevertheless, with the idea of cyberflânerie, they give a sketch of an interesting and enriching way of using and experiencing contents as well as services in the WWW. In other words: A coherent 3D networld consisting of virtual objects representing content and services, in which users can interact with each other but also perform trading with content and service providers as in today's WWW, could be the next evolutionary step of what we call 'the net' today.

As approaches like *SecondLife* [3], *Kaneva* [4] and *3D-City* [5] show, it has already been tried to put the idea of a coherent, browseable and intuitively understandable virtual content- and service environment into practice. They all have in common to provide a platform for a virtual environment, into which users can add interactive objects to display content and provide services. Although a remarkable interest from the user side has been brought towards such interaction and trading platforms (e.g., *SecondLife* has $\approx$ 60.000.000 users), none of the existing approaches has come so far yet to be recognized as a replacement for even just a few current WWW contents or services.

Even though virtual worlds provide new ways of displaying products and of interacting with potential customers, businesses are remarkably cautious about investing resources into virtual worlds and use them merely as an additional platform for advertising, but not for actually trading their products and services. This is mainly due to three reasons:

1) Businesses cannot control the availability (visibility) of their objects, but instead have to rely completely on the availability of the platform provider.
2) An unjustifiable amount of unidirectional trust into the platform provider is necessary, since all virtual objects, which may need to contain internal business information, must be hosted on the providers machines.
3) Usually, trade is done on base of virtual currencies (e.g., the *Lindendollar* in SecondLife). This leads to unforeseeable exchange rate risks for businesses, rendering safe trading almost impossible.

These problems can be overcome by distributed architectures, as they have emerged from Massively Multiplayer

 ISBN: 978-1-61208-300-1

Online Games (MMOGs). Decentralized virtual environments, usually referred to as Networked Virtual Environments (NVEs, [6]), have in common that the software components needed for managing a coherent and consistent virtual world are linked together on the base of local algorithms and without any global knowledge or control.

NVEs have remarkable advantages over centralized approaches for users and businesses:

- **Reduced dependency and necessity of trust:** Availability and visibility of contents and services do no longer have to depend on single platform providers. There is no need to upload business critical data to a third party platform provider.
- **Scalability:** Decentralized approaches can scale with both the number of users as well as the size of the virtual world.

But, none of the existing approaches for NVEs can be used as base for a next evolutionary step of the WWW, because they all expect deeper cooperation between the participating peers, whereas in a decentralized interaction and *trading* platform, the autonomy of each single participant (i.e., WWW server) is of highest importance.

A structure that provides both a coherent virtual world without making service and content providers dependent on centralized instances or other participants of the virtual world, shall be referred to as *networld* for the remaining part of this article.

In Section II, a brief overview is given on concepts and projects related to this topic. Section III displays a summary of requirements on networlds, which we have already developed in previous works (see e.g., [7] and [8]). In Section IV, the architecture of *P2Life*, our approach to build and maintain virtual decentralized networlds is presented. Section V shows, how this structure can be used to build maps of the whole virtual world efficiently and still with high redundancy without using centralized instances. Section VI concludes the article and gives an outlook on future work.

II. Related Work

Since networked virtual environments focus on interaction (and not, as networlds, on flânerie), the most important aspect in terms of scalability certainly is, how event multicasting and filtering is performed.

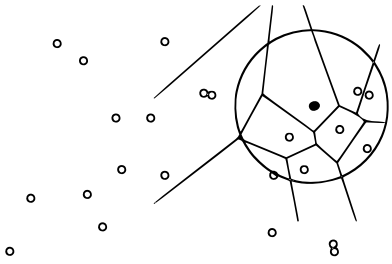


Fig. 1. Nodes in a Voronoi-based Overlay Network (VON).

Makbily et. al. (see [9] and [10]) suggest, that participants use their current position to calculate *Update-Free-Regions*

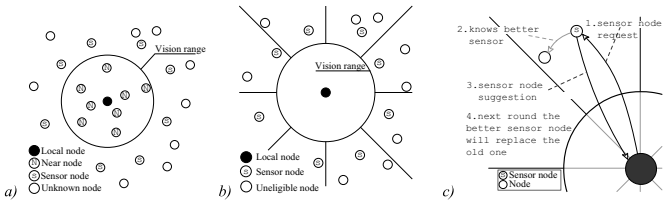


Fig. 2. Network from the perspective of a single node in *pSense*. a) Point of view and roles of the nodes, b) segmentation to select *sensor-nodes*, c) Finding the optimal *sensor-node* of a segment (Source: [24])

(UFR) pairwise for every other participant. These are regions of the virtual world, in which participants cannot propagate events to each other. They could result, e.g., from obstacles, which make participants invisible for each other. But since the UFRs have to be calculated pairwise amongst the participants, the calculation does not scale, because the number of messages needed is in $O(N^2)$. Steed and Angus also filter events based on mutual visibility [11] [12]. In their approach, the virtual world is partitioned into cells. For each cell, a rough estimation is possible, which other cells are visible from there. This information can be used to generate so-called *Frontier Sets*, out of which users can *not* see each other. Bharambe et. al. propose *Donnybrook* [13], in which the number of event messages is reduced by estimating, on which other participants a user currently focuses (*Interest Set*). A similar approach has been introduced by Najaran and Krasic [14].

Approaches that are not only used for event management but also for object management actually create and maintain virtual worlds. They can be classified into *static* and *dynamic segmentation* approaches, the first partitioning the virtual world into segments of fixed size and location, while the latter dynamically decides on size, shape, number and location of segments.

A. Static Segmentation

Knutsson et. al., who authored *SimMud* [15], assume that the structure and content of the virtual world is static and hence is available offline on every client. For avatar management, the world is partitioned into fixed segments, in which events are forwarded to all avatars located in it. Participants can register for events of several regions. Events starting from single participants are forwarded in a Pastry network [16] [17] using Scribe multicasts [18] [19].

In the *Zoned Federation of Game Servers* proposed by Iimura et. al. [20], each partition (zone) is assigned to a zone server managing all objects and events within. The zone servers are linked together based on a dynamic hashtable, which allows users to find out, which zone server is responsible for any given zone. Similar approaches for static segmentation with only minor variations have amongst others been proposed by Lee and Sun [21], Yamamoto et. al. [22], and de Oliveira et. al. [23].

 ISBN: 978-1-61208-300-1

B. Dynamic Segmentation

Although a lot more approaches for dynamic segmentation exist, only two shall be mentioned here. Schmieg et. al. [24] assign each object to a node (peer). Objects (including avatars) are not bound to a position, but can freely move through the virtual environment. Each node manages a circle-shaped field of view around it. All other nodes within this circle (*near-nodes*) receive events from the node in the center. Additionally, each node links itself to eight other nodes outside of its field of view (*sensor-nodes*), one – namely the closest – in each eighth of the compass rose (see Fig. 2).

The most advanced project based on dynamic segmentation was proposed by Hu and Chen [25] [26] and is based on a Voronoi-based Overlay Network (VON) [27]. Each node is linked to its closest neighbours (in terms of distance in the VE, see Fig. 1). This structure is used to provide *spatial publish-subscribe*, i.e., dynamically forming regions of arbitrary size, in which events are forwarded and received. The task of filtering and forwarding is no longer assigned to single peers, but instead solved cooperatively amongst all peers in a region. The approach is prone to consistency problems (see [28]), which can nevertheless be handled locally.

All dynamic segmentation approaches have in common that they do not have to limit the size of the virtual world.

III. Requirements on Decentralized Networlds

A decentralized networld, which shall be used as both an interaction and a *trading* platform must allow trading based on real money to happen in the same way as in today's WWW. Businesses and customers shall be able to use well known and well established payment methods as, e.g., credit cards or online payment services for transactions. Existing approaches for NVEs, of which a lot can undoubtly be used as interaction platforms, are however not designed to support trading.

A system that aims to merge the advantages of today's WWW with the advantages of NVEs, must fulfill specific requirements, which mostly result from real world trading and, therefore, go far beyond requirements on NVEs. In fact, when real money comes into consideration, a change of paradigm for creating a decentralized coherent virtual environment is necessary, in which cooperation between the participating peers is reduced to a minimum. Each provider must be able to maintain its autonomy and thus be in control of the data and system security.

Requirement 1 – Object ownership: In most legal systems, trading is understood as a bilateral legal act, in which the participants have to be uniquely identifiable. Since trading in the WWW as well as in the proposed networld is performed through virtual objects, these objects need to be assigned to a legal body, i.e., the contracting party of the trade. In contrast, current approaches for NVEs dynamically assign objects to peers based on structural circumstances, i.e., often an explicit owner for each object is not given.

Requirement 2 – Independent management layer: For an object to be visible in the virtual world, a management layer is needed that allows to lookup, which objects are located on a given position or within a given region. When the management layer is based solely on the structure of the virtual world, the peers managing the *neighbourhood* of an object could limit its visibility. Therefore, the management layer must be independent from the structure of the virtual world.

Requirement 3 – Unlimited virtual world: The size of the virtual world, i.e., the number of participants, is not limited. No artificial shortage of the resource *space* is allowed.

When looking at the existing NVE approaches, none of them fulfills all three requirements, which leads to the conclusion that a new approach is needed to create virtual networlds as interaction- and trading platforms.

IV. Architecture

The main part of the proposed approach, which we have named *P2Life*, contains an independent decentralized management layer, in which content providers can register their contents at specific coordinates and users (browsers) can lookup all objects located at a given coordinate or region.

A. Structure of the P2Life Decentralized Networld

The unlimited P2Life networld is divided into square-shaped parcels of equal size. Other shapes as, e.g., hexagons could be used, but nevertheless we stay with square parcels for simplicity reasons. Each parcel is either assigned to exactly one provider, or it is empty. The provider assigned to a parcel 'owns' it, i.e., he is the one to be contacted for any content located on it.

Interaction between users and providers happens in the same manner as in today's WWW. When a user enters a parcel, the browser contacts the provider of the parcel to get a description of the objects existing on it, e.g., in form of a markup language like 3DMLW. It is not necessary to decide for *one* system-wide object description language. It is well possible to use application specific languages, as long as the browser is able to understand them. Having received the object descriptions, the browser is now able to render the objects for the user and handle interactions between user and objects, possibly leading to subsequent data exchange between user and provider (analogue to input fields in HTML).

The illusion of a coherent virtual world is created by the user's browser. Whenever a parcel is visited, the browser automatically contacts the providers of the neighbouring parcels and gathers information about their displayable objects. This information can be used to display a coherent environment, in which parcel borders need not to be visible at all. It solely depends on the user's *Area of Interest* and the available data transfer rate, which parcels are displayed.

B. Registration and Lookup Architecture

To be assigned to a previously empty parcel, a provider has to register its (IP-)address for the parcel. Users can then generate a lookup for the according coordinate and receive the address of the provider to contact for more information. It is worth mentioning that using human readable names for identifying a host as in the *domain name service* still remains

 ISBN: 978-1-61208-300-1

possible. Identifying a provider using its coordinate is to be understood as an additional way of addressing, specifically useful in virtual environments.

To achieve this, a registration and lookup infrastructure is needed, which is the core architectural component of the networld described here. According to the autonomy requirement, this infrastructure must be organized in a decentralized manner. Additionally, it must not rely on the structure of the virtual world itself.

The management layer is based on a *Content-Addressable-Network* (CAN), which is a natural choice due to its planary structure. Providers, who have occupied at least one parcel in the virtual world, are automatically assigned to maintain a region in the CAN. Other DHT-like architectures could also be used, yet, the most important advantage of CANs in this context is their robustness against *man-in-the-middle* attacks, since they provide a large amount of disjunct paths between any two peers.

Coordinates are mapped to 2-dimensional keys in $[0..1] \times [0..1]$ using a system wide hash function with an adaptive number of digits. This way, the unlimited character of the coordinate space is mapped into the arbitrary granularity of the key space. The hashing guarantees that the structure of the virtual world is not reflected in the management layer's neighbourhood. The CAN-Peer managing the key for any given coordinate holds a reference (address) to the provider assigned to the respective parcel. If no provider address is assigned to that key, the parcel is assumed to be empty.

When a provider wants to register itself for any given coordinate, it generates a registration request containing the hash value of the requested coordinate and the provider's IP-address. The request is signed using the provider's credentials and then forwarded to an arbitrary CAN-peer. The peer can either directly handle the request, because it manages the required hash value, or it can forward it in the CAN until the managing CAN-peer is found. This peer can now directly contact the provider and ask for the coordinate of the parcel, for which the provider wants to register.

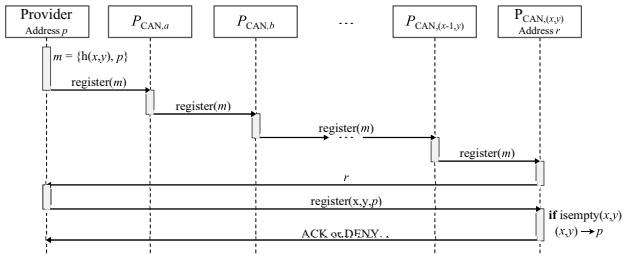


Fig. 3. Registration of a provider

If the required coordinate is empty, the CAN-peer can acknowledge the request and store the provider's IP-address in its local database. Otherwise, the request is declined.

The lookup for finding out, which provider is registered for any given coordinate, is performed in a similar way. The requesting user (browser) generates the hash value for the

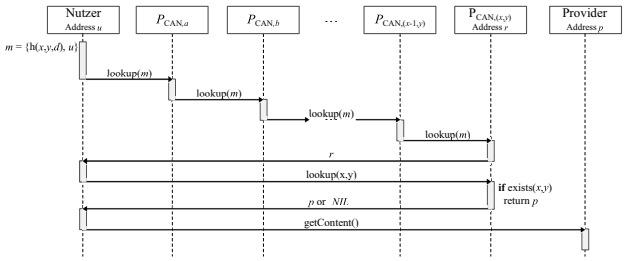


Fig. 4. Lookup scheme

requested coordinate and forwards a lookup request containing the hash value and the user's IP-address to any CAN-peer. The message is forwarded in the CAN until the managing CAN-peer is found. This CAN-peer can contact the browser directly and ask for the plaintext coordinate. If a provider address is stored for this coordinate, it is returned to the client. Otherwise, the requested parcel is assumed to be empty.

C. Hashing

The average number of messages to reach an arbitrary peer in the CAN is in $O(\sqrt{N})$. While this is efficient enough for registering a provider (because it can be assumed that this happens only once per provider) as well as for lookups, browsing would become a problem. As described above, the illusion of a coherent virtual world is created, because the browser automatically looks up and contacts the providers in the neighbourhood of a requested parcel. But since neighboured parcels are usually not managed by neighboured CAN-Peers, each of these lookups would also be in $O(\sqrt{N})$ time complexity, which would disturb a smooth browsing experience.

For this reason, each CAN peer maintains for each coordinate it manages, direct links to the CAN peers that manage the four neighboured parcels of that coordinate. The resulting structure allows to route requests in keyspace, thus independent from the structure of the virtual world, but additionally provides constant time lookups for neighboured coordinates.

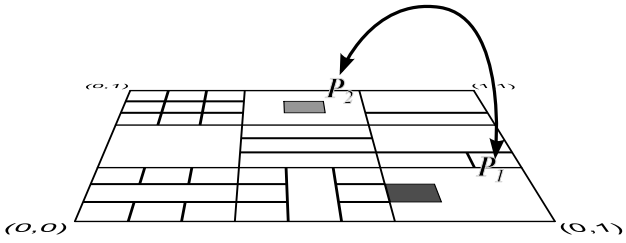


Fig. 5. Direct link between to distant CAN-peers managing keys for neighboured coordinates

But since these links have to be maintained for every key (even for those representing empty parcels) and since each key represents an unlimited number of coordinates (due to the fact that the key has a limited number of digits), a scalability problem arises: Each peer would have to maintain links to (almost) all other CAN peers. A solution for this problem is

 ISBN: 978-1-61208-300-1

to use a hashfunction with a characteristic, which could best be described as *collision-preserving*:

Let (x_{c1}, y_{c1}) and (x_{c2}, y_{c2}) be two arbitrary coordinates mapping on the same d-digit hashvalue, i.e., $h_i(x_{c1}, y_{c1}, d) = h_i(x_{c2}, y_{c2}, d)$. Furthermore, let $(x_{c1}+1, y_{c1})$ and $(x_{c2}+1, y_{c2})$ be the right neighbours of (x_{c1}, y_{c1}) and (x_{c2}, y_{c2}). Then these two values also have to be mapped on an identical hashvalue, i.e., $h_i(x_{c1}+1, y_{c1}, d) = h_i(x_{c2}+1, y_{c2}, d)$. This also needs to be true for the other three neighbouring coordinates, so that for each key only four links to other CAN peers need to be maintained.

This way, the Content-Addressable-Network evolves to a structure, which reflects both the lattice structure of the virtual world *and* the independent keyspace, in which the neighbourhood of providers is deliberately broken up. The advantages of both structures can be used without being forced to accept their disadvantages.

D. Summary

The proposed architecture allows to manage an unlimited coherent virtual networld built by independent content and service providers. Each of the providers can act the same way as today's WWW servers, which includes direct real money trade with their customers.

Due to the nature of the underlying Content-Addressable-Network, neighbourhood related attacks as well as man-in-the-middle attacks are more difficult than in previous decentralized approaches for virtual environments. Still, with the imprinted lattice structure, neighbourhood based routing and, as a result, flânerie is supported in an efficient manner.

V. Decentralized Maps

In the current version of P2Life, no restrictions exist on which parcels a provider can register for. Especially, no relation exists between the position of a provider and its content. Hence, with a growing amount of participating providers, navigation and orientation become a challenge for the user.

A likely solution for this problem is to let the system provide a coherent map of the virtual environment, through which the user can navigate in different levels of detail. In the highest level of detail, a map fraction representing the top view on a single parcel is displayed. On the lowest level of detail, all nonempty parcels are shown in a single map for a general overview. The intermediate levels let the user display map fractions containing more (lower level of detail) or less (higher level of detail) parcels.

The algorithms used to create, maintain and provide these maps of course have to be decentralized to avoid generating a dependency on centralized services. Additionally, the algorithms have to comply with the general requirements on decentralized networlds (see Section III):

- **Authorative Owners:** Each Provider manages the top views for each of their parcels on their own.
- **Verifyability:** Except for a single top view, each request for a map fraction can be answered by several independent peers so that malicious behaviour of single peers can be revealed.
- **Redundancy:** With growing level of a map fraction (i.e., with increasing number of parcels, a map fraction represents), the number of peers maintaining this map fraction also increases. Therefore, local disturbances only have local influence.

Fulfilling these requirements leads to an architecture, in which malicious behaviour of single peers can neither disturb the consistency nor the availability of the map as a whole or of its fractions. The approach described in this section covers two aspects:

1) A feasible **map structure** to manage and link the map fractions
2) A strategy for an efficient **redundant assignment** of data sets to peers maintaining them.

A. Map Structure

The map data structure is organized in such a way that different zooming levels can be provided. Therefore, the virtual world is divided into square disjunct map fractions for each level e. The division depends on a system constant b, which controls the size of the map fractions in combination with the zooming level as follows: In the lowest zooming level $e_0 = 0$, each map fraction contains a top view on a single provider, on the highest level $e_{max} \approx \log N$, the only existing map fraction represents the whole nonempty part of the unlimited virtual world. Between those two extremes, additional levels $0 < e < e_{max}, e \in \mathbb{N}$ exist, in which each map fraction represents b^{2e} parcels (see Fig. 6b).

The centre coordinates (x, y) of the map fractions are chosen in such a way that each map fraction in level $e > 0$ fully contains exactly b^2 map fractions of level $e-1$:

$$x = c_x \cdot b_m^e + \frac{b_m^e - 1}{2}, y = c_y \cdot b_m^e + \frac{b_m^e - 1}{2}, c_x, c_y \in \mathbb{Z} \qquad (1)$$

It is obvious that an appropriate data structure to manage the map fractions on different levels must have the nature of a tree, in which each node represents a part of the map. Nodes close to the root represent map fractions of higher levels (i.e., of lower detail), while leaf nodes represent top views on single providers.

The map is generated bottom-up, i.e., the providers manage the information of the leaf nodes. Each provider maintains for each of its parcels all information necessary to display them in the map. The map fractions of the higher levels are generated using the information available from their child map fractions of lower levels. Changes in the map are also generated on level 0, e.g., by adding or removing providers. For each map fraction a dataset exists that contains the following elements:

1) one or more displayable *representations* of the map fraction.
2) either 0 or 2 to b_m^2 links to *child maps*, i.e., map fractions that represent a part of the virtual world that is located within the same area as the current map fraction. Hence, child maps always have a lower level than the map they are a child of.

 ISBN: 978-1-61208-300-1

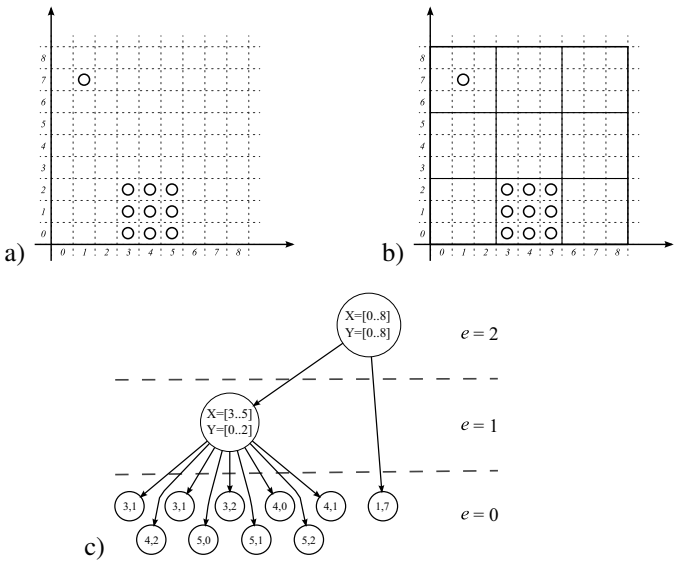


Fig. 6. a) Providers in an example virtual world, b) map partitioning, and c) resulting tree of map fractions.

3) 0 or 1 link to a *parent map*, i.e., the map fraction that this map is a child map of. If no such map exists, the current map is the root map and represents the whole nonempty virtual world.

The tree contains only relevant map fractions. There is no need to maintain a map fraction that contains no data, because no providers are located within the respective area of the virtual world. In the same way, a map fraction that only has one child map is obsolete, since it is fully represented by that single child map. The child maps are chosen in such a way that starting from any of the b_m^2 (max.) child maps, each map fraction of level 0 (i.e., each single provider) that is represented by the respective child map, can be reached. Additionally, the child maps are chosen to have the lowest level possible. This reduces the number of reorgs necessary to handle changes in the map datastructure.

In Fig. 6.a, an example of a virtual world with 10 providers (circles) is shown. There is a area with higher provider density around parcel $(4,1)$ and a single outlying provider on parcel $(1,7)$. Fig. 6.b shows the resulting partitioning on levels 0, 1 and 2. The base for the partitioning has a value of $b = 3$. The resulting tree of map fractions in Fig. 6.c starts with a root map of level 2, i.e., with an edge length of $3^2 = 9$ and its center at $(4,4)$. The area with higher provider density around $(4,1)$ is represented by an intermediate map of level 1, which contains 9 child maps of level 0. The outlying provider at $(1,7)$ is instead directly linked to the root map.

This way, the number of map fractions necessary to represent any given area of the virtual world roughly scales with the number of *providers* within it instead of its *extent*. Additionally, the time and message complexity to reach an arbitrary map fraction of level 0 starting from the root map remains in $O(\log N)$.

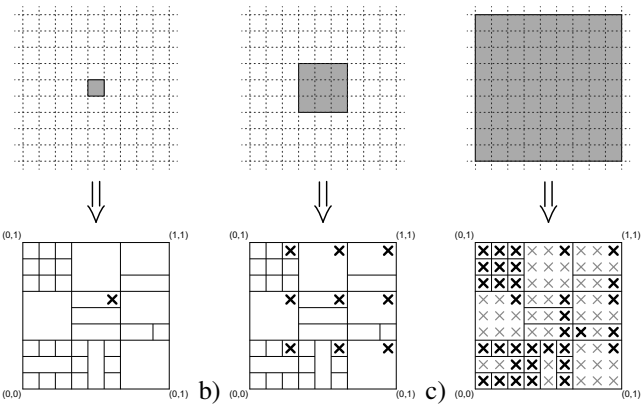


Fig. 7. Carrier assignment example for a) level $e = 0$, b) level 1 and c) level 2 in a CAN with base $b = 3$.

B. Assignment of map fractions to peers

The maintenance of the map fractions is integrated into the already existing management structure (see Section IV-B). A CAN-Peer, who manages a given map fraction, is called a *carrier* of it. Only the map fractions of level 0 have a single carrier. All map fractions of higher levels have multiple carriers. The assignment of a map fraction of level to its carriers is done by building the 2-dimensional hash value (k_x, k_y) of the map's center coordinate. through the key (see Section IV-C). A CAN-Peer is carrier for a map fraction of level e, who's center coordinate is mapped to $k = (k_x, k_y)$ in hashspace, when the CAN region of the peer covers a key matching k with the leading e digits masked out. For a single provider, i.e., for $e = 0$, the CAN-Peer managing k is the only carrier. For the root map, i.e., $e = e_{max}$, (almost) all CAN-Peers are carriers.

In Fig. 7, the assignment of map fractions to carriers is demonstrated with three example maps having the same center coordinate but different levels. In subfigure c), some over-determined carrier assignments can be seen, i.e., a CAN-Peer manages several keys matching the masked hashvalue of the according map's center coordinate. In such a case, the carrier will of course only maintain a single copy of the map fraction.

Events, which can influence the appearance of the map fractions, are always initiated on level 0, i.e., by providers.

- **Registration:** A provider registers for a formerly empty parcel. The according carrier is contacted during the registration process and receives all information necessary to manage a map fraction of level 0 from the provider.
- **Update:** The information for a single parcel has changed. The provider generates an update message for the according carrier.
- **Unregistration or provider fail:** A provider has unregistered or is assumed to be ultimatively offline for other reasons.

In all three cases, the according carrier on level 0 is contacted by the provider. The carrier performs the necessary changes on the map fraction data and then contacts the b^2 carriers of the next higher level (i.e., the carriers of the parent

map), which can be done with local knowledge. Each of these carriers updates the map fraction data for the parent map and forwards the event to their parent map carriers, and so on.

To maintain locality and to reduce network load, each carrier of level e forwards events only to carriers within the same CAN-region (see Fig. 8), i.e., each carrier regardless of its level only has to generate a maximum of b^2 parallel update messages to perform the forwarding.

Attack- or failure scenarios derived from misbehaviour of single peers can be handled, if

1) each carrier of level e sends an event to *few* arbitrary carriers of level $e+1$ *outside* of the current carrier's CAN-region. Additionally, the carrier forwards the event to 1 or 2 arbitrary carriers of level e *within* its CAN-region.
2) carriers of identical map fractions randomly cross-check their data from time to time.

This way, updates do reach a carrier on different ways without influencing the overall performance. Malicious behaviour of single peers can be revealed in short time.

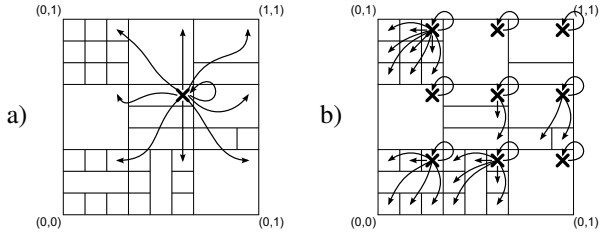


Fig. 8. Example event forwarding from a) level 0 to level 1, b) level 1 to 2.

C. Performance

The most important aspect in practical matters is, whether the proposed architecture allows users to navigate through the map in such a way, that they are able to contact any peer in the CAN and reach the carrier for any map fraction efficiently.

It is obvious that each node in the tree of map fractions can be reached in $O(\log N)$ time complexity, when the CAN distance between two carriers is ignored. Hence, if each carrier maintains a small cache with direct links to carriers of parent- and child maps, the number of hops to reach an arbitrary map fraction carrier is also in $O(\log N)$. Without such a cache, the number of hops is in $O(\sqrt{N})$.

The second aspect of the performance evaluation covers the handling of map events. To avoid that each event has to be forwarded up all the way to the root node, updates are only forwarded when they perform a noticeable change in the parent map. Otherwise, the update event is stored locally and merged with further events, until an effect on the parent map actually occurs. This way, the *average* number of forwards in the tree for each event is constant, i.e., in $O(1)$.

Due to limited space, a proof is omitted here and only an example simulation showing the average number of hops over different network sizes is given in Fig. 9a.

As mentioned above, for each forwarding of an event to the next higher level, each carrier has to send b^2 physical messages through the CAN. Each of this messages has an average path length of $\sqrt{N/b^{2e}}$. Hence, the total number of physical messages to forward an event from level e to level $e+1$ is

$$b^{2(e+1)} \cdot \sqrt{N/b^{2e}} = b^{e+2} \cdot \sqrt{N} \tag{2}$$

If again a threshold Δ is used to limit forwarding to events that actually have an effect on the next higher level, the total number of messages sums up to

$$\frac{\sqrt{N}}{\Delta} \sum_{e=0}^{e_{max}-1} \frac{1}{b^e} < \frac{\sqrt{N}}{\Delta} \frac{1}{b-1} \in O(\sqrt{N}) \tag{3}$$

without using caches (see Fig 9b).

When caches are used, i.e., when instead of using the CAN for message delivery, each carrier can contact the b^2 carriers of the next higher level directly, the average number of physical messages is reduced to

$$\frac{1}{\Delta} \sum_{e=0}^{e_{max}-1} \frac{b^{2(e+1)}}{b^{2(e+1)}} = \frac{e_{max}-1}{\Delta} \in O(\log N) \tag{4}$$

D. Summary

As long as no semantic navigation is possible in virtual worlds, navigable maps are an essential tool for orientation. Since the main idea behind this article is to provide a *decentralized* networld, creation and maintenance of such maps may not be conceded solely to centralized instances. The proposed approach utilizes the already existing architecture to store a tree of map fractions in a massively redundant way. By using simple threshold mechanisms, both maintenance and usage of the resulting maps can be done efficiently.

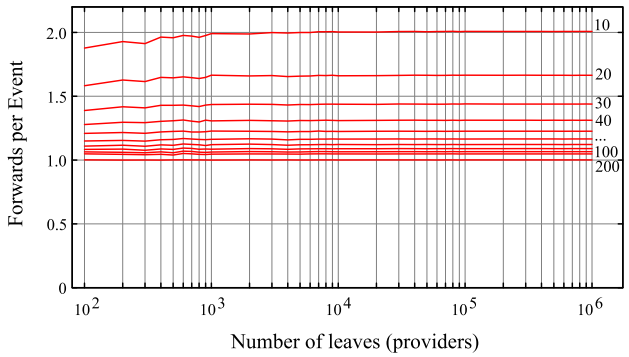
VI. Conclusion

Virtual networlds as platforms for interaction, content and service delivery do have the potential to be the next evolutionary step of the WWW. To do so, they have to allow content and service providers to keep their independence as in today's WWW and still maintain the impression of a coherent unlimited virtual world. Existing approaches in the area of networked virtual environments fail to fulfill these basic requirements. Therefore, we have presented an architecture to build unlimited networlds, through which users can browse as cyberflâneurs without the need for a specific target or to give up their anonymity. Interaction between user and provider is handled directly, which allows among others for real money trading as in the WWW.

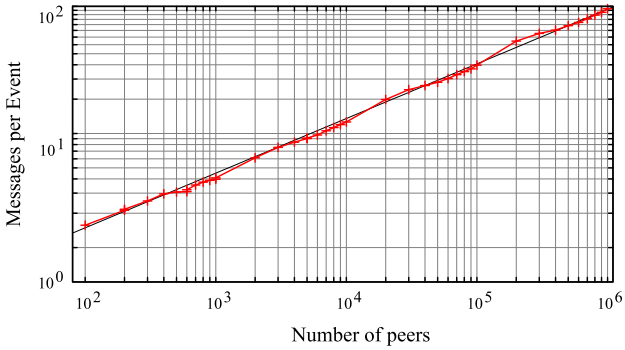
Navigation and Orientation is provided through dynamic maps of the nonempty parts of the networld. The maps are stored in a decentralized way utilizing the already existing architecture to maintain the networld, with a high level of redundancy and yet with low maintenance costs in terms of hops and physical messages necessary.

Yet, a single reality virtual world will inevitably reach its limits, when the number of content and service providers increases. It is therefore necessary to allow a kind of semantic navigation, i.e., to build individual views onto the available content based on a user's current context.

 ISBN: 978-1-61208-300-1



(a) Average number of forwards in the tree of map fractions to handle an event originating from level 0 using thresholds from 10 to 200.



(b) Average number of *physical* messages in the CAN to handle an event, when a threshold Δ exists and no caches are used.

Fig. 9. Simulation results for handling events in the tree of map fractions in a decentralized manner.

REFERENCES

[1] N. Gardels, "The changing global order: world leaders reflect". Wiley-Blackwell, 1997.

[2] E. Morozov, "The death of the cyberflâneur," http://www.nytimes.com/2012/02/05/opinion/sunday/the-death-of-the-cyberflaneur.html, retrieved June 2013

[3] SecondLife website, http://secondlife.com/, retrieved June 2013

[4] Kaneva project website, http://www.kaneva.com, retrieved June 2013

[5] 3D-City project website, http://www.3dcity.de/, retrieved June 2013

[6] S. Singhal and M. Zyda, "Networked virtual environments: design and implementation", ser. SIGGRAPH series. Addison-Wesley, 1999.

[7] H. Coltzau, "P2life: An infrastructure for networked virtual marketplace environments," *IJIIP*, vol. 1, no. 2, 2010, pp. 1–13.

[8] H. Coltzau and B. Ulke, "Navigation in the p2life networked virtual marketplace environment," in Autonomous Systems: Developments and Trends, ser. Studies in Computational Intelligence, H. Unger, K. Kyamakya, and J. Kacprzyk, Eds. Springer Berlin / Heidelberg, vol. 391, 2012, pp. 213–227.

[9] Y. Makbily, C. Gotsman, and R. Bar-Yehuda, "Geometric algorithms for message filtering in decentralized virtual environments," in Proceedings of the 1999 symposium on Interactive 3D graphics, ser. I3D '99. New York, NY, USA: ACM, 1999, pp. 39–46.

[10] A. Goldin and C. Gotsman, "Geometric message-filtering protocols for distributed multiagent environments," *Presence: Teleoper. Virtual Environ.*, vol. 13, no. 3, Jul. 2004, pp. 279–295.

[11] A. Steed and C. Angus, "Enabling scalability by partitioning virtual environments using frontier sets," in Presence: Teleoper. Virtual Environ., vol. 15, no. 1, Feb. 2006, pp. 77–92.

[12] S. Avni and J. Stewart, "Frontier sets in large terrains," in Proceedings of Graphics Interface 2010, ser. GI '10. Toronto, Ont., Canada, Canada: Canadian Information Processing Society, 2010, pp. 169–176.

[13] A. Bharambe, J. R. Douceur, J. R. Lorch, T. Moscibroda, J. Pang, S. Seshan, and X. Zhuang, "Donnybrook: enabling large-scale, high-speed, peer-to-peer games," SIGCOMM Comput. Commun. Rev., vol. 38, no. 4, 2008, pp. 389–400.

[14] M. T. Najaran and C. Krasic, "Scaling online games with adaptive interest management in the cloud," in Proceedings of the 9th Annual Workshop on Network and Systems Support for Games, ser. NetGames '10. Piscataway, NJ, USA: IEEE Press, 2010, pp. 9:1–9:6.

[15] B. Knutsson, H. Lu, J. C. Mogul, and B. Hopkins, "Architecture and performance of server-directed transcoding," in ACM Trans. Internet Techn., vol. 3, no. 4, 2003, pp. 392–424.

[16] A. Rowstron and P. Druschel, "Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems," in IFIP/ACM International Conference on Distributed Systems Platforms (Middleware), Nov. 2001, pp. 329–350.

[17] M. Castro, M. B. Jones, A.-M. Kermarrec, A. Rowstron, M. Theimer, H. Wang, and A. Wolman, "An evaluation of scalable application-level multicast built using peer-to-peer overlays," in Proceedings of Infocom'03, Apr. 2003, pp. 1510 – 1520.

[18] A. Rowstron, A.-M. Kermarrec, M. Castro, and P. Druschel, "Scribe: The design of a large-scale event notification infrastructure," in Networked Group Communication, Third International COST264 Workshop (NGC'2001), ser. Lecture Notes in Computer Science, J. Crowcroft and M. Hofmann, Eds., vol. 2233, Nov. 2001, pp. 30–43.

[19] M. Castro, P. Druschel, A.-M. Kermarrec, and A. Rowstron, "Scribe: A large-scale and decentralized application-level multicast infrastructure," IEEE Journal on Selected Areas in Communication (JSAC), vol. 20, no. 8, 2002, pp. 100–110.

[20] T. Iimura, H. Hazeyama, and Y. Kadobayashi, "Zoned federation of game servers: a peer-to-peer approach to scalable multi-player online games," in Proceedings of 3rd ACM SIGCOMM workshop on Network and system support for games, ser. NetGames '04. New York, NY, USA: ACM, 2004, pp. 116–120.

[21] H.-H. Lee and C.-H. Sun, "Load-balancing for peer-to-peer networked virtual environment," in Proceedings of 5th ACM SIGCOMM workshop on Network and system support for games, ser. NetGames '06. New York, NY, USA: ACM, 2006, p. Art.14.

[22] S. Yamamoto, Y. Murata, K. Yasumoto, and M. Ito, "A distributed event delivery method with load balancing for mmorpg," in NETGAMES, 2005, pp. 1–8.

[23] J. C. de Oliveira, D. T. Ahmed, and S. Shirmohammadi, "Performance enhancement in mmogs using entity types," in Proceedings of the 11th IEEE International Symposium on Distributed Simulation and Real-Time Applications, ser. DS-RT '07. Washington, DC, USA: IEEE Computer Society, 2007, pp. 25–30.

[24] A. Schmieg, M. Stieler, S. Jeckel, B. Kabus, Patric Kemme, and A. Buchmann, "psense - maintaining a dynamic localized peer-to-peer structure for position based multicast in games," in Proceedings of the 8th International Conference on Peer-to-Peer Computing 2008 (P2P 2008), pp. 247–256.

[25] S.-Y. Hu, C. Wu, E. Buyukkaya, C.-H. Chien, T.-H. Lin, M. Abdallah, J.-R. Jiang, and K.-T. Chen, "A spatial publish subscribe overlay for massively multiuser virtual environments," in International Conference on Electronics and Information Engineering (ICEIE), Aug. 2010, pp. 314–318.

[26] S.-Y. Hu and K.-T. Chen, "Vso: Self-organizing spatial publish subscribe," in Proceedings of IEEE SASO, Oct 2011, pp. 21–30.

[27] 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, 2006 pp. 22–31.

[28] H. Backhaus and S. Krause, "Voronoi-based adaptive scalable transfer revisited: gain and loss of a voronoi-based peer-to-peer approach for mmog," in Proceedings of the 6th ACM SIGCOMM workshop on Network and system support for games, ser. NetGames '07. New York, NY, USA: ACM, 2007, pp. 49–54.

 ISBN: 978-1-61208-300-1